4. IMPLEMENTASI SISTEM

Pada bagian ini akan di jelaskan mengenai implementasi dari desain yang sudah di buat dalam bab 3. Terdapat tiga bagian utama dalam melakukan implementasi sistem, yaitu implementasi server, implementasi IoT menggunakan Arduino, dan implementasi aplikasi menggunakan Flutter.

4.1 Pemerograman Server Backend

Pada bagian ini akan membahas mengenai implementasi pada bagian *backend* berupa implementasi *source code* dan sistem yang di implementasikan untuk berinteraksi antara aplikasi Android dan sistem IoT.

4.1.1 Setup Server

Implementasi server menggunakan Node JS dengan Library Express.js dan Server juga di buat menggunakan Typescript untuk memudahkan melakukan *debugging*.

```
import express, { Request, Response } from "express";
import http from "http";
import { Server } from "socket.io";
import dotenv from "dotenv";
import cors from "cors";
```

Segmen Program 4.1 Setup Dependency

Pada segmen program 4.1, menggambarkan mengenai *dependency* yang di gunakan untuk *setup* server Express.js dan *library* yang di gunakan melakukukan komunikasi. Express.js dan http dimana server dapat menerima *Request* dan mengirim *Response*, Socket.io untuk komunikasi *websocket*, dotenv untuk mengakses *environment file*, dan cors untuk mengatur akses server.

```
const app = express();
const server = http.createServer(app);
const io = new Server(server);
app.use(express.json());

dotenv.config();

const port: number = parseInt(process.env.PORT as string) | 3000;
```

Segmen Program 4.2 Setup Server

Pada bagian segmen program 4.2, Kode ini bertujuan untuk menginisialisasi aplikasi Express untuk menangani permintaan HTTP, membuat *server* HTTP menggunakan aplikasi Express, dan menyiapkan server Socket.IO untuk komunikasi *real-time*. Kode "express.json" adalah middleware yang di gunakan untuk memproses permintaan JSON yang masuk dan "dotenv.config()" di gunakan memuat variabel lingkungan dari file .env. Server mendengarkan pada port yang ditentukan oleh variabel lingkungan PORT, defaultnya adalah 3000 jika tidak ditentukan. Selain itu, CORS diaktifkan untuk mengizinkan permintaan dari asal mana pun.

4.1.2 Database Connection

Pada bagian ini akan di bahas mengenai pembuatan dan koneksi ke *database*. *Database* di buat menggunakan suatu ORM (*Object Relation Mapping*) untuk memudahkan melakukan prubahan terhadap struktur *database*, serta memudahkan akses tabel dan koneksi terhadap database.

```
import { PrismaClient } from "@prisma/client";

const globalForPrisma = globalThis as unknown as { prisma:
PrismaClient };

export const prisma = globalForPrisma.prisma || new
PrismaClient();

if (process.env.NODE_ENV !== "production") globalForPrisma.prisma
= prisma;
```

Segmen Program 4.3 Setup Database Connection

Kode pada segmen 4.3 mengimpor PrismaClient dari paket @prisma/client dan menyiapkan variabel global untuk menyimpan *instance* PrismaClient untuk menghindari terbuatnya beberapa Prisma *instance*. Kode berikutnya menginisialisasi prisma dengan *instance* PrismaClient yang ada dari variabel global jika tersedia, atau membuat *instance* baru. Jika aplikasi tidak berjalan dalam produksi (yaitu, process.env.NODE_ENV bukan "produksi"), aplikasi tersebut akan menugaskan instance PrismaClient yang baru dibuat ke variabel global. Hal ini memastikan hanya satu *instance* PrismaClient yang terbuat, dan membantu mencegah masalah berkaitan dengan koneksi database lebih dari satu.

4.1.3 Auth

Pada bagian ini, akan di bahas mengenai logika untuk melakukan Authorization, login, signup, dan implementasi Json Web Token.

```
import jwt from "jsonwebtoken";
import dotenv from "dotenv";
import bcrypt from "bcrypt";
import { Request, Response, NextFunction } from "express";
import { prisma } from "./server.db.js";

dotenv.config();

interface JwtPayload {
  email: string;
  userId: number;
}

interface JwtOptions {
  expiresIn: string | number;
}

const JWT_SECRET = process.env.JWT_SECRET || "default_jwt_secret_key";
```

Segmen Program 4.4 Auth Dependency

Kode pada segmen program 4.4 mengimpor *library* yang diperlukan seperti jsonwebtoken, dotenv, dan bcrypt, serta *Request*, *Respons*, dan *nextFunction* dari Express, dan *instance* prisma dari file lokal. Dotenv di gunakan untuk memuat variabel lingkungan dari file .env. Dua *interface*, JwtPayload dan JwtOptions, menentukan struktur payload dan *option* yang digunakan untuk JWT. JWT_SECRET menampung nilai variabel dari process.env.JWT_SECRET atau default ke "default_jwt_secret_key" jika tidak ditentukan/tidak di dapat dari .env.

```
export function generateToken(
  payload: JwtPayload,
  options: JwtOptions
): string {
  return jwt.sign(payload, JWT_SECRET, options);
}

export function verifyToken(req: Request, res: Response, next:
NextFunction) {
  const token = req.headers.authorization?.split(" ")[1];
  if (!token) {
    return res
      .status(401)
      .json({ message: "Access denied. No token provided." });
  }

  try {
    const decoded = jwt.verify(token, JWT_SECRET) as JwtPayload;
    (req as any).user = decoded;
    next();
  } catch (error) {
    return res.status(403).json({ message: "Invalid token." });
  }
}
```

Segmen Program 4.5 JsonWebToken generate token & verify token

Kode pada segmen program 4.5 mengekspor dua *function*, yaitu generateToken dan verifikasiToken, untuk menangani penggunaan JSON Web Token (JWT) untuk *authentification*. *Function* generateToken menerima *payload* dan *option* sebagai argumen dan mengembalikan *token* yang di tandatangani menggunakan *library* jsonwebtoken dan JWT_SECRET.

Function verifikasiToken adalah middleware yang memeriksa token pada *header Authorization* pada *request* yang masuk. Jika token tidak ada, akan di berikan respons dengan status 401 dan pesan "Akses ditolak". Dan jika token ada, token tersebut akan di verifikasi menggunakan JWT_SECRET. Jika verifikasi berhasil, *function* akan mengirimpkan payload yang di *decrypt* ke objek *request* dan memanggil next() untuk meneruskan *request* ke middleware berikutnya. Jika verifikasi gagal, akan di kirim *response* dengan status 403 dan pesan "Token tidak valid". Hal ini memastikan bahwa hanya permintaan yang sudah di *authenticate* yang dapat mengakses rute yang di lindungi.

```
export async function login(
  email: string,
  password: string
): Promise<string | null> {
  // Find user by email
  const user = await prisma.user.findFirst({
    where: {
      email: email,
    },
  });
  console.log("try find user");
  console.log(email);
  if (!user) return null; // User not found
  console.log("user found");
  if (!user.password) return null; // User password not null

  // Check if the provided password matches the hashed password
  const passwordMatch = bcrypt.compareSync(password, user.password);
  if (!passwordMatch) return null; // Password doesn't match

  // Generate JWT token
  const payload: JwtPayload = { email: user.email, userId: user.id };
  const options: JwtOptions = { expiresIn: "7d" };
  return generateToken(payload, options);
}
```

Segmen Program 4.6 Auth Login

Kode pada segmen program 4.6 mengekspor dua *function*, yaitu *login* dan *signin*. *Function* logic menerima email dan password sebagai *argument*, dan *function* ini akan mencari email pada *database*. Jika email tidak ditemukan maka akan mengembalikan *null,* dan jika email ditemukan akan berlanjut untuk mengambil password dari email tersebut dan memastikan bahwa hasil *encryption* password yang diterima *function* sama dengan *encryption* yang di simpan di *database*. Jika password yang di terima sesuai dengan password yang ada pada *database*, *function* akan membuat token JWT yang akan *expire* dalam waktu 7 hari dan mengembalikan token JWT tersebut.

```
export async function signIn(
  email: string,
  password: string
): Promise<string | null> {
  const existingUser = await prisma.user.findFirst({
    where: {
      email: email,
    },
  });
  if (existingUser) return null;
  console.log("no user exist");
  // Hash the password
  const saltRounds = 10;
  const passwordHash = bcrypt.hashSync(password, saltRounds);
  console.log("encrypt pass");
  // Add the user to the database
  const newUser = await prisma.user.create({
    data: {
      email: email,
      password: passwordHash,
    },
  });

  // Generate JWT token
  const payload: JwtPayload = { email, userId: newUser.id };
  const options: JwtOptions = { expiresIn: "7d" };
  return generateToken(payload, options);
}
```

Segmen Program 4.7 Auth Signin

Function signup juga menerima email dan password sebagai *argument*. *Function* ini akan memasukan email dan password baru ke dalam *database* jika emal tidak ada di dalam *database*. Password akan di *encrypt* sebelum di simpan ke dalam *database*. *Function* akan menghasilkan *token* yang akan *expire* dalam waktu tujuh hari.

```
app.post("/login", async (req, res) => {
  console.log("test");
  const { email, password } = req.body;
  if (!email || !password) {
    return res.status(400).json({ message: "Email and password are
required" });
  }
  console.log(`email: ${email}`);
  console.log(`pass: ${password}`);
  const token = await login(email, password);
  if (token) {
    res.json({ token });
  } else {
    res.status(401).json({ message: "Invalid email or password" });
  }
});
```

Segmen Program 4.8 Login Access Point

Segmen program 4.8 merupakan implementasi mengenai *access point* untuk melakukan login, dimana *access point* ini akan mengambil email dan password dalam *request body*. Jika password dan email tidak ada atau "null", akan di kirim *response* berstatus 400 dengan pesan "Emai and password are required". Jika email dan password ada, maka akan di masukan ke *function* login, jika login berhasil maka token akan di kirim dalam bentuk json pada *response*, dan jika login gagal akan di kirim *response* dengan status 401 dan pesan "Invalid email or password".

```
app.post("/signup", async (req, res) => {
  console.log("test signup");
  const { email, password } = req.body;
  if (!email || !password) {
    return res.status(400).json({ message: "Email and password are
required" });
  }
  const token = await signIn(email, password);
  console.log(`token: ${token}`);
  if (token) {
    res.json({ token });
  } else {
    res.status(409).json({ message: "User already exists" });
  }
});
```

Segmen Program 4.9 Signup / Signin Access Point

Segmen program 4.9 merupakan implementasi mengenai *access point* untuk melakukan signup, dimana *access point* ini akan mengambil email dan password dalam *request body*. Jika

password dan email tidak ada atau “null”, akan di kirim *response* berstatus 400 dengan pesan “Emai and password are required”. Jika email dan password ada, maka akan di masukan ke *function* signup, dan jika signup berhasil. maka token akan di kirim dalam bentuk json pada *response*, dan jika signup gagal akan di kirim *response* dengan status 409 dan pesan “User already exists”.

4.1.4 Farm

Pada bagian ini akan membahas mengenai implementasi program dan logika yang di buat untuk menjalankan fungsi untuk membuat data, mengubah data, dan menghapus data pada tabel *farm*. Implementasi ini di buat pada file *farm_service.ts* dan memiliki *access point* pada file *server.ts*.

```
import { Farm } from "@prisma/client";
import { prisma } from "../utils/server.db.js";
import { farmFieldData } from "../utils/types.js";
```

Segmen Program 4.10 Farm Dependency

```
export type farmFieldData = {
  farm_name: string;
  farm_room_id: string;
};
```

Segmen Program 4.11 Farm Field Data Type

Segmen 4.10 adalah *dependency* yang di butuhkan pada file *farm_service.ts*, dimana terdapat jenis data “Farm” dari @prisma/client, prisma dari server.db.js pada segmen program 4.3 untuk *query* database, dan *farmFieldType* yang di *export* dari "../utils/types.js".

```
export async function getUserFarms(userId: number):
Promise<Farm[]> {
  try {
    const userFarms = await prisma.user_farm_list.findMany({
      where: {
        user_id: userId,
      },
      include: {
        farm: true,
      },
    });

    return userFarms.map((entry) => entry.farm);
  } catch (error) {
    console.error("Error fetching user farms:", error);
    throw error;
  }
}
```

Segmen Program 4.12 Get User Farm

Segmen 4.12 adalah *function* yang bertujuan untuk menggambil semua data *farm* pada satu user, dimana *function* menerima *userId* dan menjalankan *query* menggunakan prisma untuk mendapat semua data *farm* yang di miliki oleh *user*. *Function* akan mengembalikan data dalam bentuk *array of object*.

```
export async function getFarmDetails(farmId: number):
Promise<Farm | null> {
  try {
    const farmDetails = await prisma.farm.findUnique({
      where: {
        id: farmId,
      },
    });

    return farmDetails;
  } catch (error) {
    console.error("Error fetching farm details:", error);
    throw error;
  }
}
```

Segmen Program 4.13 Get Farm Details

Segmen program 4.13 adalah *function* yang bertujuan untuk mengambil detail data *farm*, yaitu data yang bersangkutan dengan *farm* tersebut. *Function* akan menerima *idFarm* dan mengambil data yang bersangkutan dengan ID tersebut.

```
export async function createFarm(
  userId: number,
  farmName: string,
  farmRoomId: string
): Promise<Farm | null> {
  try {
    const existingFarm = await prisma.user_farm_list.findFirst({
      where: {
        user_id: userId,
        farm: {
          farm_name: farmName,
        },
      },
    });
    if (existingFarm) {
      throw new Error(
        `A farm with the name '${farmName}' already exists in the
user's farm list.`
      );
    }
    const existingFarmRoomId = await
prisma.user_farm_list.findFirst({
      where: {
        user_id: userId,
        farm: {
          farm_room_id: farmRoomId,
        },
      },
    });
    if (existingFarmRoomId) {
      throw new Error(
        `A farm room id with the name '${farmRoomId}' already
exists in the user's farm list.`
      );
    }
    const newFarm = await prisma.farm.create({
      data: {
        farm_name: farmName,
        farm_room_id: farmRoomId,
      },
    });
    await prisma.user_farm_list.create({
      data: {
        user_id: userId,
        farm_id: newFarm.id,
      },
    });
    return newFarm;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.14 Create Farm

Segmen program 4.14 adalah *function* yang bertujuan untuk membuat data farm baru pada *database*. *Function* ini menerima *userId*, *farmName*, dan *farmRoomId* sebagai argumen dan akan melakukan cek bila nama farm dengan userId yang diberikan sudah ada atau tidak pada tabel *user_farm_list*. Jika ada, maka *function* akan mengembalikan *error*, sedangkan jika tidak ada maka akan melanjutkan cek bila *farmRoomId* dengan *userId* yang diberikan sudah ada di dalam tabel *user_farm_list*. Jika sudah maka akan di kembalikan *error*, dan jika belum akan dibuat *farm* pada *farm* tabel dan *user_farm_list* tabel dalam *database*.

```
export async function updateFarm(
  farmId: number,
  updatedFields: farmFieldData
): Promise<Farm | null> {
  try {
    // Update the farm name and room ID
    const updatedFarm = await prisma.farm.update({
      where: {
        id: farmId,
      },
      data: {
        farm_name: updatedFields.farm_name,
        farm_room_id: updatedFields.farm_room_id,
      },
    });

    console.log(`Farm with ID ${farmId} updated successfully.`);
    return updatedFarm;
  } catch (error) {
    console.error("Error updating farm:", error);
    throw error;
  }
}
```

Segmen Program 4.15 Update Farm

Segmen program 4.15 adalah *function* yang berfungsi untuk memperbarui data *farm*. *Function* menerima *farmId* dan *updatedField* sebagai argument. *Function* ini akan memperbarui data *farm* dengan id yang sama dengan argument *farmId*.

```
export async function deleteFarm(userId: number, farmId: number)
{
  try {
    // Delete the farm from the user's farm list
    await prisma.user_farm_list.deleteMany({
      where: {
        user_id: userId,
        farm_id: farmId,
      },
    });

    // Delete the farm itself
    await prisma.farm.delete({
      where: {
        id: farmId,
      },
    });

    console.log(`Farm with ID ${farmId} deleted successfully.`);
  } catch (error) {
    console.error("Error deleting farm:", error);
    throw error;
  }
}
```

Segmen Program 4.16 Delete Farm

Segmen program 4.16 adalah *function* yang di gunakan untuk menghapus *farm* dari database dimana *userId* dan *farmId* akan di terima sebagai argumen dan *function* akan menghapus data *farm* yang ada pada *user_farm_list* dan *farm* table.

```
app.post("/farm", verifyToken, async (req: Request, res:
Response) => {
  const { userId } = req.body;
  if (!userId) {
    return res.status(400).json({ message: "User ID is required"
});
  }
  try {
    const farmData = await getUserFarms(userId);
    res.json(farmData).status(200);
  } catch (error) {
    console.error("Error fetching farms:", error);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Segmen Program 4.17 Access Point Get Farm

```
app.post("/add-farm", verifyToken, async (req: Request, res:
Response) => {
  const { userId, farmName, farmRoomId } = req.body;
  if (!userId || !farmName || !farmRoomId) {
    return res.status(400).json({ message: "User ID is required"
});
  }
  try {
    const farmData = await createFarm(userId, farmName,
farmRoomId);
    res.json(farmData).status(200);
  } catch (error) {
    console.error("Error fetching farms:", error);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Segmen Program 4.18 Access Point Add Farm

```
app.post("/edit-farm", verifyToken, async (req: Request, res:
Response) => {
  const { farmId, updatedFieldData } = req.body;
  if (!farmId || !updatedFieldData) {
    return res.status(400).json({ message: "User ID is required"
});
  }
  try {
    const farmData = await updateFarm(farmId, updatedFieldData);
    res.json(farmData).status(200);
  } catch (error) {
    console.error("Error fetching farms:", error);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Segmen Program 4.19 Access Point Edit Farm

Pada segmen program 4.17, 4.18, dan 4.19 merupakan implementasi *access point* untuk menggunakan *function* getUserFarm untuk mengambil semau data *farm* suatu *user*, createFarm, dan updateFarm untuk membuat dan merubah data *farm* dari suatu *user*.

```
app.post("/delete-farm", verifyToken, async (req: Request, res:
Response) => {
  const { userId, farmId } = req.body;
  if (!farmId || !userId) {
    return res.status(400).json({ message: "User ID is required"
});
  }

  try {
    await deleteFarm(userId, farmId);
    res.json({ msg: "farm successfully deleted" }).status(200);
  } catch (error) {
    console.error("Error fetching farms:", error);
    res.status(500).json({ error: "Internal server error" });
  }
});
```

Segmen Program 4.20 Access Point Delete Farm

4.1.5 User Setting

Pada bagian ini akan di jelaskan mengenai implementasi dan program yang di buat untuk mengatur semua data yang berkaitan dengan *user settings*. Implementasi di buat dan di tempatkan pada *file user_setting_service.ts*.

```
import {
  Fan_settings,
  Light_settings,
  User_settings,
  Watering_settings,
} from "@prisma/client";
import { prisma } from "../utils/server.db.js";
import {
  CreateFanSettingInput,
  SensorData,
  ThresholdExceedance,
  Thresholds,
  UserSettingsUpdateInput,
  createLightingSettingInput,
  createWateringSettingInput,
} from "../utils/types.js";
```

Segmen Program 4.21 User Setting Dependency

Segmen program 4.21 adalah bagian dependency yang di gunakan pada user_setting_service.ts. Dependency yang terdapat pada file user_setting_service.ts terdiri dari jenis

data untuk argumen suatu function hingga tipe data dari prisma untuk melakukan query dengan prisma.

```
export type CreateFanSettingInput = {
  fan_on_time: Date;
  fan_off_time: Date;
  active_status: boolean;
};
export type SensorData = {
  humidity: number;
  ppm: number;
  ph: number;
};
export type ThresholdExceedance = {
  humidityExceedance: boolean;
  ppmExceedance: boolean;
  phExceedance: boolean;
};
export type Thresholds = {
  minHumidity: number;
  maxHumidity: number;
  minPpm: number;
  maxPpm: number;
  minPh: number;
  maxPh: number;
};
export type UserSettingsUpdateInput = {
  user_setting_name: string;
  min_humidity: number;
  max_humidity: number;
  min_ppm: number;
  max_ppm: number;
  min_ph: number;
  max_ph: number;
  ultrasonic_shot_hour: number;
  in_use: boolean;
};
export type createLightingSettingInput = {
  light_on_time: Date;
  light_off_time: Date;
  active_status: boolean;
};
export type createWateringSettingInput = Omit<
  Watering_settings,
  "id" | "user_setting_id"
>;
```

Segmen Program 4.22 User setting data types

```
export async function getUserSettings(
  userId: number
): Promise<User_settings[] | null> {
  try {
    const userSettings = await prisma.user_settings.findMany({
      where: {
        user_id: userId,
      },
    });

    return userSettings;
  } catch (error) {
    console.error("Error fetching user settings:", error);
    throw error;
  }
}
```

Segmen Program 4.23 Get User Settings

```
export async function getSpecificUserSetting(
  userSettingId: number
): Promise<User_settings | null> {
  try {
    const userSetting = await prisma.user_settings.findFirst({
      where: {
        id: userSettingId,
      },
    });

    return userSetting;
  } catch (error) {
    console.error("Error fetching user setting:", error);
    throw error;
  }
}
```

Segmen Program 4.24 Get specific user setting

Pada kedua segmen program 4.23 dan 4.24 adalah program yang di gunakan untuk mengambil informasi mengenai user settings yang di miliki oleh user dan untuk menggambil informasi detail dari suatu user setting pada suatu user.

```
export async function getUserSettingOfFarm(
  farmId: number
): Promise<User_settings[] | null> {
  try {
    const userSetting = await prisma.user_settings.findMany({
      where: {
        farm_id: farmId,
      },
    });
    return userSetting;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.25 Get user setting off farm

Segmen Program 4.25 merupakan *function* yang di gunakan untuk mendapatkan *user setting* yang di gunakan oleh suatu *farm. Function* ini menerima *farmId* sebagai argumen dan mengembalikan *user setting* yang di gunakan pada *farm* dengan ID yang sama dengan argumen yang di berikan.

```
export async function addUserSettings(
  userId: number,
  farmId: number,
  settingsData: UserSettingsUpdateInput
): Promise<User_settings | null> {
  try {
    const newUserSettings = await prisma.user_settings.create({
      data: {
        ...settingsData,
        user: { connect: { id: userId } },
        farm: { connect: { id: farmId } },
      },
    });
    return newUserSettings;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.26 Add user setting

Segmen program 4.26 adalah function yang berfungsi untuk menambah user setting dimana function menerima userId, farmId, dan settingsData dengan tipe data UserSettingsUpdateInput dari segmen program 4.22.

```
export async function updateUserSettings(
  userSettingsId: number,
  updatedFields: UserSettingsUpdateInput
): Promise<User_settings | null> {
  try {
    const updatedUserSettings = await
prisma.user_settings.update({
      where: { id: userSettingsId },
      data: updatedFields,
    });
    return updatedUserSettings;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.27 Update user settings

```
export async function deleteUserSettings(userSettingsId: number)
{
  try {
    await prisma.user_settings.delete({
      where: { id: userSettingsId },
    });
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.28 Delete user settings

Function pada segmen program 4.27 dan 4.28 adalah *function updateUserSettings* yang di gunakan untuk memperbarui *user setting* yang di miliki oleh *user* tertentu dan deleteUserSettings yang di gunankan untuk menghapus *user setting* yang di miliki *user setting* tertentu.

```
export async function createFanSetting(userSettingsId: number,
fanSettingData: CreateFanSettingInput
): Promise<Fan_settings> {
  try {
    const newFanSetting = await prisma.fan_settings.create({
      data: {
        ...fanSettingData,
        user_settings: { connect: { id: userSettingsId } },
      },
    });
    return newFanSetting;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.29 Create fan setting

```
export async function deleteFanSetting(fanSettingId: number):
Promise<void> {
  try {
    await prisma.fan_settings.delete({
      where: { id: fanSettingId },
    });
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.30 Delete fan setting

```
export async function getAllFanSettings(userSettingsId: number):
Promise<Fan_settings[]> {
  try {
    const fanSettings = await prisma.fan_settings.findMany({
      where: {
        user_setting_id: userSettingsId,
      },
    });
    return fanSettings;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.31 Get all fan setting

Segmen program 4.29, 4.30, dan 4.31 adalah *functions* yang bertujuan untuk mengatur data *fan Setting*, di mana *createFanSetting* pada segmen 4.29 untuk membuat data *fan setting* pada *user settings* yang berkaitan, segmen 4.30 untuk menghapus *fan setting* tertentu, dan 4.31 untuk mengambil semua *fan settings* pada suatu *user settings.*

```
export async function createLightingSetting(userSettingsId:
number,  lightingSettingData: createLightingSettingInput
): Promise<Light_settings> {
  try {
    const newLightingSetting = await
prisma.light_settings.create({
      data: {
        ...lightingSettingData,
        user_settings: { connect: { id: userSettingsId } },
      },
    });
    return newLightingSetting;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.32 Create light setting

```
export async function deleteLightingSetting(lightingSettingId:
number): Promise<void> {
  try {
    await prisma.light_settings.delete({
      where: { id: lightingSettingId },
    });
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.33 Delete light setting

Pada segmen program 4.32 dan 4.33 adalah function yang di gunakan untuk membuat lighting setting dan menghapus lighting setting pada suatu user settings. Sedangkan pada segmen program 4.34 di gunakan untuk mengambil semua data mengenai fan setting yang bersangkutan dengan user setting tertentu.

```
export async function getAllLightingSettings(userSettingsId:
number): Promise<Light_settings[]> {
  try {
    const lightingSettings = await
prisma.light_settings.findMany({
      where: {
        user_setting_id: userSettingsId,
      },});
    return lightingSettings;
  } catch (error) {
    throw error;
  }}
```

Segmen Program 4.34 Get all light settings

```
export async function createWateringSetting(
  userSettingsId: number,
  wateringSettingData: createWateringSettingInput
): Promise<Watering_settings> {
  try {
    const newWateringSetting = await
prisma.watering_settings.create({
      data: {
        ...wateringSettingData,
        user_settings: { connect: { id: userSettingsId } },
      },});
    return newWateringSetting;
  } catch (error) {
    throw error;
  }}
```

Segmen Program 4.35 Create watering setting

```
export async function deleteWateringSetting(
  wateringSettingId: number
): Promise<void> {
  try {
    await prisma.watering_settings.delete({
      where: { id: wateringSettingId },
    });
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.36 Create watering setting

```
export async function getAllWateringSettings(
  userSettingsId: number
): Promise<Watering_settings[]> {
  try {
    const wateringSettings = await
prisma.watering_settings.findMany({
      where: {
        user_setting_id: userSettingsId,
      },
    });
    return wateringSettings;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.37 Get all watering settings

Pada segmen program 4.35, 4.36, dan 4.37 adalah implementasi program yang berkaitan dengan pengaturan *watering setting* dan pengambilan data *watering setting* dari suatu *user settings. createWateringSetting* pada segmen 4.35 berfungsi untuk membuat data *watering setting* baru untuk suatu *user settings. deleteWateringSetting* pada segmen 4.36 berfungsi untuk menghapus data *watering setting* dari *database.*

```
export function checkSensorThresholds(
  sensorData: SensorData,
  thresholds: Thresholds
): ThresholdExceedance {
  const { humidity, ppm, ph } = sensorData;
  const { minHumidity, maxHumidity, minPpm, maxPpm, minPh, maxPh
} = thresholds;
  const humidityExceedance = humidity < minHumidity || humidity
> maxHumidity;
  const ppmExceedance = ppm < minPpm || ppm > maxPpm;
  const phExceedance = ph < minPh || ph > maxPh;
  return {
    humidityExceedance,
    ppmExceedance,
    phExceedance,
  };
}
```

Segmen Program 4.38 Check sensor thresholds

Pada segmen program 4.38 berfungsi untuk melakukan pengecekan antars parameter dalam *user settings* dan data terbaru dari sensor. *Function* akan menerima data *user setting* dan data sensor terbaru dan membandingkan jika data sensor berada di dalam atau di luar parameter *user setting.*

```
app.post("/user-setting", verifyToken, async (req: Request, res:
Response) => {
  const { userId } = req.body;
  if (!userId) {
    return res.status(400).json({ message: "User ID is required"
});
  }
  try {
    const userSettingData = await getUserSettings(userId);
    res.json({ userSettingData: userSettingData }).status(200);
  } catch (error) {
    res.status(500).json({ error: "Internal server error" });
  }
});

app.post("/user-setting-per-farm", verifyToken, async (req:
Request, res: Response) => {
    const { farmId } = req.body;
    if (!farmId) {
      return res.status(400).json({ message: "Farm ID is
required" });
    }
try {
      const userSettingData = await
getUserSettingOfFarm(farmId);
      res.json({ userSettingData: userSettingData
}).status(200);
    }
  }
);
```

Segmen Program 4.39 user setting access point

Segmen program 4.39 adalah implementasi access point untuk menggunakan getUserSettings dan getUserSettingOfFarm. Access point akan mengecek data yang masuk, jika data valid, maka akan menjalankan function lalu mengirimkan response dan jika data tidak valid maka akan di beri response error.

```
app.post(
  "/user-setting-detail",
  verifyToken,
  async (req: Request, res: Response) => {
    const { userSettingId } = req.body;
    if (!userSettingId) {
      return res.status(400).json({ message: "User ID is
required" });
    }
    try {
      const userSettingData = await
getSpecificUserSetting(userSettingId);
      res.json({ userSettingData: userSettingData
}).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
app.post(
  "/user-setting-detail/schedule", verifyToken,
  async (req: Request, res: Response) => {
    const { userSettingId } = req.body;
    if (!userSettingId) {
      return res.status(400).json({ message: "User ID is
required" });
    }
    try {
      const fanScheduleData = await
getAllFanSettings(userSettingId);
      const ledScheduleData = await
getAllLightingSettings(userSettingId);
      const wateringScheduleData = await
getAllWateringSettings(userSettingId);
      res.json({
          scheduleData: {
            fanScheduleData: fanScheduleData,
            ledScheduleData: ledScheduleData,
            wateringScheduleData: wateringScheduleData,
          },
        }).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.40 user setting detail & schedule access point

```
app.post(  "/user-setting-detail/add-schedule",  verifyToken,
  async (req: Request, res: Response) => {
    const { onTime, offTime, type, userSettingId } = req.body;
    if (!onTime || !offTime || !type || !userSettingId) {
      return res.status(400).json({ message: "schedule data is required" });
    }
    let scheduleData;
    try {
      const [hourStringOn, minuteStringOn] = onTime.split(":");
      const [hourStringOff, minuteStringOff] = offTime.split(":");
      const hourOn = parseInt(hourStringOn, 10);
      const minuteOn = parseInt(minuteStringOn, 10);
      const hourOff = parseInt(hourStringOff, 10);
      const minuteOff = parseInt(minuteStringOff, 10);
      const tempDateOn = new Date();
      const tempDateOff = new Date();
      tempDateOn.setUTCHours(hourOn, minuteOn, 0);
      tempDateOff.setUTCHours(hourOff, minuteOff, 0);
      switch (type) {
        case "fan":
          let tempDataFan: CreateFanSettingInput = {
            fan_on_time: tempDateOn,
            fan_off_time: tempDateOff,
            active_status: true,
          };
          scheduleData = await createFanSetting(userSettingId, tempDataFan);
          break;
        case "LED":
          let tempDataLED: createLightingSettingInput = {
            light_on_time: tempDateOn,
            light_off_time: tempDateOff,
            active_status: true,
          };
          scheduleData = await createLightingSetting(
            userSettingId,
            tempDataLED
          );
          break;
        case "watering":
          let tempDataWatering: createWateringSettingInput = {
            watering_time: tempDateOn,
            watering_for: tempDateOff,
            active_status: true,
          };
```

```
          scheduleData = await createWateringSetting(
            userSettingId,
            tempDataWatering
          );
          break;
        default:
          break;
      }
      res.json({ scheduleData: scheduleData }).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.41 User setting add schedule access point

Segmen program 4.40 adalah implementasi access point untuk mengakses funcitons untuk mengambil user setting detail dan schedule dari penyiraman, penyalaan lampu, dan kipas. Selain itu segmen program 4.41 adalah implementasi access point untuk mengakses function untuk menambahkan data watering setting, fan setting, dan lighting setting.

```
app.post(  "/user-setting-detail/delete-schedule",  verifyToken,
  async (req: Request, res: Response) => {
    const { scheduleId, type } = req.body;
    if (!scheduleId || !type) {
      return res.status(400).json({ message: "schedule ID is
required." });
    }
    try {
      switch (type) {
        case "Fan":
          await deleteFanSetting(scheduleId);
          break;
        case "LED":
          await deleteLightingSetting(scheduleId);
          break;
        case "Watering":
          await deleteWateringSetting(scheduleId);
          break;
        default:
          break;
      }
      res.json({ message: "delete complete" }).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  });
```

Segmen Program 4.42 User setting delete schedule access point

Pada segmen program 4.42 adalah *access point* yang di gunakan untuk mengakses *function*s *deleteFanSetting, deleteLightingSetting,* dan *deleteWateringSetting* untuk menghapus *schedule* setting *lighting, watering,* dan *fan* dari *database.*

```
app.post(  "/add-user-setting",  verifyToken,
  async (req: Request, res: Response) => {
    const { userId, farmId, settingsData } = req.body;
    if (!userId || !farmId) {
      return res
        .status(400)
        .json({ message: "User ID and Farm ID is required" });
    }
    if (
      !settingsData.user_setting_name ||
      !settingsData.max_humidity ||
      !settingsData.max_ppm ||
      !settingsData.min_ph ||
      !settingsData.max_ph ||
      !settingsData.ultrasonic_shot_hour
    ) {
      return res
        .status(400)
        .json({ message: "User setting complete data is
required" });
    }
    try {
      const userSettingData = await addUserSettings(
        userId,
        farmId,
        settingsData as UserSettingsUpdateInput
      );
      res.json(userSettingData).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.43 Add user setting access point

Pada segmen program 4.43 adalah *access point* untuk mengakses *function* untuk menambah *user setting* untuk suatu *user.* Dimana *access point* akan mengecek jika data yang di butuhkan sudah valid, jika sudah valid maka akan berlanjut ke pembuatan *user setting*.

```
app.post(  "/delete-user-setting",  verifyToken,
  async (req: Request, res: Response) => {
    const { userSettingId } = req.body;
    if (!userSettingId) {
      return res.status(400).json({ message: "User Setting ID is
required" });
    }
    try {
      await deleteUserSettings(userSettingId);
      res.json({ msg: "user setting successfully deleted."
}).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.44 delete user setting access point

Segmen program 4.44 adalah *access point* untuk mengakses *function deleteUserSettings* untuk menghapus *user settings* dari suatu *user.*

```
app.post(
  "/user-setting-detail/edit-user-setting",
  verifyToken,
  async (req: Request, res: Response) => {
    const { userSettingId, settingsData } = req.body;
    if (!userSettingId) {
      return res
        .status(400)
        .json({ message: "User ID and Farm ID is required" });
    }
    if (
      !settingsData.user_setting_name ||
      !settingsData.max_humidity ||
      !settingsData.max_ppm ||
      !settingsData.min_ph ||
      !settingsData.max_ph ||
      !settingsData.ultrasonic_shot_hour
    ) {
      return res
        .status(400)
        .json({ message: "User setting complete data is required" });
    }
    try {
      const userSettingData = await updateUserSettings(
        userSettingId,
        settingsData as UserSettingsUpdateInput
      );
      res.json(userSettingData).status(200);
    } catch (error) {
      console.error("Error fetching user settings:", error);
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.45 Edit user setting access point

Segmen program 4.45 merupakan access point yang di gunakan untuk mengakses function updateUserSettings untuk melakukan perubahan terhadap user setting pada database dimana access point akan melakukan validasi terhadap data pada request body. Jika data valid, maka updateUserSettings akan di jalankan untuk mengubah data user settings.

4.1.6 Grow Cycle

Pada bagian ini akan di jelaskan mengenai implementasi dan program yang di buat untuk mengatur semua data yang berkaitan dengan *grow cycle*. Implementasi di buat dan di tempatkan pada *file grow_cycle_service.ts*.

```
import {
  grow_cycle,
  humidity_per_cycle,
  ph_per_cycle,
  plant_high_per_cycle,
  ppm_per_cycle,
  temperature_per_cycle,
} from "@prisma/client";
import { prisma } from "../utils/server.db.js";
import {
  GrowCycleWithSettings,
  HumidityData,
  LatestData,
  PhData,
  PlantHighData,
  PpmData,
  TemperatureData,
  createGrowCycleInput,
} from "../utils/types.js";
```

Segmen Program 4.46 Grow cycle service dependency

Segmen program 4.46 merupakan *dependency* yang di gunakan pada *file grow_cycle_service.ts*. *Dependency* terdiri dari prisma untuk *query database* dan tipe data dari @prisma/client dan types.js pada segmen 4.47.

```
export interface GrowCycleWithSettings {
  growCycle: grow_cycle;
  plant_high_per_cycle: plant_high_per_cycle[];
  userSettings: User_settings;
  wateringSettings: Watering settings[];
  fanSettings: Fan_settings[];
  lightSettings: Light_settings[];
}
export type HumidityData = Omit<humidity_per_cycle, "id" | "grow_cycle_id">;
export type PhData = Omit<ph_per_cycle, "id" | "grow_cycle_id">;
export type PlantHighData = Omit<plant_high_per_cycle, "id" | "grow_cycle_id">;
export type PpmData = Omit<ppm_per_cycle, "id" | "grow_cycle_id">;
```

```
export type TemperatureData = Omit<  temperature_per_cycle,
"id" | "grow_cycle_id">;
export interface LatestData {
  ppm: number | null;
  ph: number | null;
  temperature: number | null;
  humidity: number | null;
}
export type createGrowCycleInput = Omit<  grow_cycle,  "id" |
```

Segmen Program 4.47 Grow cycle type & interface

```
export async function createGrowCycle(  farmId: number,
userSettingsId: number,
  growCycleData: createGrowCycleInput
): Promise<grow_cycle> {
  try {
    const newGrowCycle = await prisma.grow_cycle.create({
      data: {
        ...growCycleData,
        user_settings: { connect: { id: userSettingsId } },
        farm: { connect: { id: farmId } },
      },
    });
    return newGrowCycle;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.48 Create grow cycle

Segmen program 4.48 adalah *function* yang di gunakan untuk membuat *grow cycle* dari suatu *farm*, dimana *function* akan menerima *farmId, userSettingsId* dan *growCycleData* lalu membuat data *grow cycle* baru di *database.*

```
export async function updateGrowCycle(growCycleId: number,
updatedFields: Partial<grow_cycle> ): Promise<grow_cycle> {
  try {
    const updatedGrowCycle = await prisma.grow_cycle.update({
      where: { id: growCycleId },
      data: updatedFields,
    });
    return updatedGrowCycle;
  } catch (error) {
    throw error;
  }
}
export async function updateGrowCycleFinishDate(
  growCycleId: number,
  finishDate: Date
): Promise<grow_cycle> {
  try {
    const updatedGrowCycle = await prisma.grow_cycle.update({
      where: { id: growCycleId },
      data: {
        finish_date: finishDate,
        active_status: false,
      },
    });
    return updatedGrowCycle;
  } catch (error) {
    throw error;
  }
}
export async function updateGrowCycleYield(
  growCycleId: number,
  harvest: number
): Promise<grow_cycle> {
  try {
    const updatedGrowCycle = await prisma.grow_cycle.update({
      where: { id: growCycleId },
      data: {
        total_harvest: harvest,
      },
    });
    return updatedGrowCycle;
  } catch (error) {
    throw error;
  }
}
```

```
export async function deleteGrowCycle(growCycleId: number):
Promise<void> {
  try {
    await prisma.grow_cycle.delete({
      where: { id: growCycleId },
    });
  } catch (error) {
    throw error;
  }}
```

Segmen Program 4.49 Update & delete grow cycle

Function yang ada pada segmen 4.49 bertujuan untuk mengubah data pada *grow cycle* dan untuk menghapus suatu *grow cycle.*

```
export async function getAllGrowCycles(farmId: number):
Promise<grow_cycle[]> {
  try {
    const growCycles = await prisma.grow_cycle.findMany({
      where: {
        farm_id: farmId,
      },
    });
    return growCycles;
  } catch (error) {
    throw error;
  }
}
export async function getAllGrowCyclesPerUser(  userId: number):
Promise<grow_cycle[]> {
  try {
    const userFarms = await prisma.user_farm_list.findMany({
      where: {
        user_id: userId,
      },
      select: {
        farm_id: true,
      },});
    const farmIds = userFarms.map((userFarm) => userFarm.farm_id);
    const growCycles = await prisma.grow_cycle.findMany({
      where: {
        farm_id: {
          in: farmIds,
        },
```

```
      },
    });
    return growCycles;
  } catch (error) {
    throw error;
  }
}
export async function getGrowCyclesByFarmRoomId(
  roomId: string
): Promise<grow_cycle | null> {
  try {
    const farm = await prisma.farm.findFirst({
      where: {
        farm_room_id: roomId,
      },
      include: {
        grow_cycles: {
          where: {
            active_status: true, // Filter grow cycles by
active_status
          },
          take: 1,
        }, // Include the associated grow cycles
      },
    });
    if (!farm) {
      throw new Error(`Farm with room ID ${roomId} not found`);
    }
    const activeGrowCycle = farm.grow_cycles[0] || null;
    return activeGrowCycle;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.50 Get grow cycle

Pada segmen program 4.50 adalah *functions* yang di gunakan untuk mengambil data *grow cycle* dari *database* berdasarkan *farm, user,* atau *farm room id*.

```
export async function createHumidityPerCycle(  growCycleId: number,
  humidityData: HumidityData
): Promise<humidity_per_cycle> {
  try {
    const newHumidityPerCycle = await prisma.humidity_per_cycle.create({
      data: {
        ...humidityData,
        grow_cycle: { connect: { id: growCycleId } },
      },
    });
    return newHumidityPerCycle;
  } catch (error) {
    throw error;
  }
}
export async function createTemperaturePerCycle( growCycleId: number,
  temperatureData: TemperatureData
): Promise<temperature_per_cycle> {
  try {
    const newTemperaturePerCycle = await prisma.temperature_per_cycle.create({
      data: {
        ...temperatureData,
        grow_cycle: { connect: { id: growCycleId } },
      },
    });
    return newTemperaturePerCycle;
  } catch (error) {
    throw error;
  }
}
export async function createPpmPerCycle( growCycleId: number,  ppmData:
PpmData
): Promise<ppm_per_cycle> {
  try {
    const newPpmPerCycle = await prisma.ppm_per_cycle.create({
      data: {
        ...ppmData,
        grow_cycle: { connect: { id: growCycleId } },
      },
    });
    return newPpmPerCycle;
  } catch (error) {
    throw error;
  }
}
```

```
export async function createPhPerCycle( growCycleId: number,  phData: PhData
): Promise<ph_per_cycle> {
  try {
    const newPhPerCycle = await prisma.ph_per_cycle.create({
      data: {
        ...phData,
        grow_cycle: { connect: { id: growCycleId } },
      },
    });
    return newPhPerCycle;
  } catch (error) {
    throw error;
  }
}
export async function createPlantHighPerCycle( growCycleId: number,
plantHighData: PlantHighData
): Promise<plant_high_per_cycle> {
  try {
    const newPlantHighPerCycle = await prisma.plant_high_per_cycle.create({
      data: {
        ...plantHighData,
        grow_cycle: { connect: { id: growCycleId } },
      },});
    return newPlantHighPerCycle;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.51 Create sensor data

Pada segmen program 4.51 adalah *functions* yang di gunakan untuk melakukan pembuatan data sensor baru ke dalam *database*. Terdapat empat jenis data sensor yang dapat di buat, yaitu data sensor pH, PPM/TDS, kelembaban, temperature, dan Panjang tanaman.

```
export async function getAllCycleData(growCycleId: number):
Promise<{
  humidity: humidity_per_cycle[];
  temperature: temperature_per_cycle[];
  ppm: ppm_per_cycle[];
  ph: ph_per_cycle[];
  plantHeight: plant_high_per_cycle[];
}> {
```

```
try {
    const humidityData = await prisma.humidity_per_cycle.findMany({
      where: {
        grow_cycle_id: growCycleId,
      },
    });
    const temperatureData = await prisma.temperature_per_cycle.findMany({
      where: {
        grow_cycle_id: growCycleId,
      },
    });
    const ppmData = await prisma.ppm_per_cycle.findMany({
      where: {
        grow_cycle_id: growCycleId,
      },
    });
    const phData = await prisma.ph_per_cycle.findMany({
      where: {
        grow_cycle_id: growCycleId,
      },
      take: 10,
    });
    const plantHeightData = await prisma.plant_high_per_cycle.findMany({
      where: {
        grow_cycle_id: growCycleId,
      },
    });
    return {
      humidity: humidityData,
      temperature: temperatureData,
      ppm: ppmData,
      ph: phData,
      plantHeight: plantHeightData,
    };
  } catch (error) {
    throw error;
  }
}
export async function getLatestDataForGrowCycle( growCycleId: number
): Promise<LatestData> {
  const [latestPpm, latestPh, latestTemperature, latestHumidity] =
    await Promise.all([
      prisma.ppm_per_cycle.findFirst({
        where: { grow_cycle_id: growCycleId },
        orderBy: { time: "desc" },
      }),
```

```
      prisma.ph_per_cycle.findMany({
        where: { grow_cycle_id: growCycleId },
        orderBy: { time: "desc" },
        take: 10,
      }),
      prisma.temperature_per_cycle.findFirst({
        where: { grow_cycle_id: growCycleId },
        orderBy: { time: "desc" },
      }),
      prisma.humidity_per_cycle.findFirst({
        where: { grow_cycle_id: growCycleId },
        orderBy: { time: "desc" },
      }),
    ]);
  let latestMeanpH: number = 0;
  for (let i = 0; i < latestPh.length; i++) {
    latestMeanpH += latestPh[i].ph;
  }
  latestMeanpH = latestMeanpH / latestPh.length;
  return {
    ppm: latestPpm?.ppm || null,
    ph: latestMeanpH || null,
    temperature: latestTemperature?.temperature || null,
    humidity: latestHumidity?.humidity || null,
  };
}
export async function getAllActiveGrowCyclesWithSettings(): Promise<
  GrowCycleWithSettings[]
> {
  try {
    const activeGrowCycles = await prisma.grow_cycle.findMany({
      where: {
        active_status: true,
      },
      include: {
        plant_high_per_cycle: {
          orderBy: { time: "desc" },
          take: 5,
        },
        user_settings: {
          include: {
            watering_settings: true,
            fan_settings: true,
            light_settings: true,
          },
        },
```

```
      },
    });
    const result: GrowCycleWithSettings[] = activeGrowCycles.map(
      (growCycle) => ({
        growCycle,
        plant_high_per_cycle: growCycle.plant_high_per_cycle,
        userSettings: growCycle.user_settings,
        wateringSettings: growCycle.user_settings.watering_settings,
        fanSettings: growCycle.user_settings.fan_settings,
        lightSettings: growCycle.user_settings.light_settings,
      })
    );
    return result;
  } catch (error) {
    throw error;
  }
}
```

Segmen Program 4.52 Get Grow cycle data & sensor data

Pada segmen program 4.52 adalah *functions* yang berfungsi untuk mengambil data *grow cycle*, data sensor, dan *user setting* yang bersangkutan dengan *grow cycle* tersebut.

```
app.post("/grow-cycle", verifyToken, async (req: Request, res: Response) => {
  const { farmId } = req.body;
  if (!farmId) {
    return res.status(400).json({ message: "User ID is required" });
  }
  try {
    const growCycleFarmData = await getAllGrowCycles(farmId);
    res.json(growCycleFarmData).status(200);
  } catch (error) {
    res.status(500).json({ error: "Internal server error" });
  }
});
app.post(
  "/add-grow-cycle",
  verifyToken,
  async (req: Request, res: Response) => {
    const { farmId, userSettingId, growCycleData } = req.body;
    if (!farmId || !userSettingId || !growCycleData) {
      return res.status(400).json({ message: "User ID is required" });
    }
    const currentDate = new Date();
    currentDate.setUTCHours(currentDate.getUTCHours() + 7);
```

```
    const newGrowCycleData = {
      ...growCycleData,
      start_date: currentDate.toISOString(),
    };
    try {
      const growCycleFarmData = await createGrowCycle(
        farmId,
        userSettingId,
        newGrowCycleData
      );
      res.json(growCycleFarmData).status(200);
    } catch (error) {
      console.error("Error fetching farms:", error);
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
app.post(
  "/finish-grow-cycle",
  verifyToken,
  async (req: Request, res: Response) => {
    const { growCycleId } = req.body;
    if (!growCycleId) {
      return res.status(400).json({ message: "Grow Cycle data is required"
});
    }
    const currentDate = new Date();
    currentDate.setUTCHours(currentDate.getUTCHours() + 7);
    try {
      const growCycleFarmData = await updateGrowCycleFinishDate(
        growCycleId,
        currentDate
      );
      const getFarmData = await getFarmDetails(growCycleFarmData.farm_id);
      if (getFarmData) {
        for (let i = 0; i < 8; i++) {
          const offComamnd = {
            type: "control",
            hardware: "relay",
            pin: i, // pin for relay like LED, Fan, Watering
            run: "off", // on, off, or echo
          };
          toggleRelay(getFarmData.farm_room_id, offComamnd);
        }
      }
      res.json(growCycleFarmData).status(200);
```

```
    } catch (error) {
      console.error("Error fetching farms:", error);
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
app.post(
  "/add-harvest-grow-cycle",
  verifyToken,
  async (req: Request, res: Response) => {
    const { growCycleId, harvest } = req.body;
    if (!growCycleId || !harvest) {
      return res.status(400).json({ message: "Grow Cycle data is required"
});
    }
    const currentDate = new Date();
    currentDate.setUTCHours(currentDate.getUTCHours() + 7);
    try {
      const growCycleFarmData = await updateGrowCycleYield(
        growCycleId,
        parseInt(harvest)
      );
      res.json(growCycleFarmData).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
app.post("/grow-cycle-data",  verifyToken,  async (req: Request, res:
Response) => {
    const { growCycleId } = req.body;
    if (!growCycleId) {
      return res.status(400).json({ message: "Grow cycle ID is required" });
    }
    try {
      const growCycleData = await getAllCycleData(growCycleId);
      res.json(growCycleData).status(200);
    } catch (error) {
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
app.post(
  "/all-user-grow-cycle-data",
```

```
  verifyToken,
  async (req: Request, res: Response) => {
    const { userId } = req.body;
    if (!userId) {
      return res.status(400).json({ message: "User ID is required" });
    }

    try {
      const growCycleData = await getAllGrowCyclesPerUser(userId);
      res.json(growCycleData).status(200);
    } catch (error) {
      console.error("Error fetching farms:", error);
      res.status(500).json({ error: "Internal server error" });
    }
  }
);
```

Segmen Program 4.53 Grow cycle access point

Pada segmen program 4.53 adalah implementasi *access point* untuk semua *functions* yang berhubungan dengan *grow cycle*, yaitu *function* untuk mengambil data *grow cycle* dan data sensor, mengubah data, dan menyelesaikan *grow cycle*.

4.1.7 Socket IO

Pada bagian ini akan di jelaskan mengenai implementasi dan program yang di buat untuk melakukan komunikasi antara server, sistem IoT, dan aplikasi mobile. Implementasi ini di lakukan pada *server.ts*.

```
export function toggleRelay(room: string, data: relayToggleData) {
  let dataSend = {
    type: "control",
    hardware: "relay",
    run: data.run,
    pin: data.pin,
  };
  io.in(room).emit("cmd", dataSend);
}
export function emitUltrasonicCommand(room: string) {
  let data = {
    type: "control",
    hardware: "ultrasonic",
    run: "echo",
    pin: 0,
```

```
  };
  io.to(room).emit("cmd", data);
}
```

Segmen Program 4.54 toggleRelay function & emitUltrasonicCommand

Pada segmen program 4.54 adalah *functions* yang di gunakan untuk mengirimkan instruksi melalui *websocket* untuk menjalankan komponen elektronik dan sensor *ultrasonic*.

```
io.on("connection", (socket) => {
  socket.on("joinRoom", (data: joinRommData) => {
    let room_id = data.roomID;
    socket.join(room_id);
  });
  socket.on("toggleRelay", (data) => {
    toggleRelay(data.room, data.data as relayToggleData);
  });
  const pingInterval = setInterval(() => {
    socket.emit("ping");
  }, 20000);
  socket.on("pong", () => {
    console.log("get ponged");
  });
  const checkingGrowCycleData = setInterval(checkGrowCycleData, 120 * 1000);
  socket.on("data", async (payload) => {
    const getGrowCycle: grow_cycle | null = await getGrowCyclesByFarmRoomId(
      payload.roomID
    );
    if (payload.code == "relay") {
      io.to(payload.roomID).except(socket.id).emit("roomMessage", {
        hardware: payload.hardware,
        status: payload.status,
      });
    }
    if (getGrowCycle != null) {
      const currentDate = new Date();
      currentDate.setUTCHours(currentDate.getUTCHours() + 7);
      if (payload.code == "sensor") {
        if (payload.type == "TDS" && payload.value != null) {
          await createPpmPerCycle(getGrowCycle.id, {
            time: currentDate,
            ppm: payload.value,
          });
        } else if (payload.type == "PH" && payload.value != null) {
```

```
          await createPhPerCycle(getGrowCycle.id, {
            time: currentDate,
            ph: payload.value,
          });
        } else if (payload.type == "HUMIDITY" && payload.value !=
null) {
          await createHumidityPerCycle(getGrowCycle.id, {
            time: currentDate,
            humidity: payload.value,
          });
        } else if (payload.type == "AIR-TEMPRATURE" && payload.value
!= null) {
          await createTemperaturePerCycle(getGrowCycle.id, {
            time: currentDate,
            temperature: payload.value,
          });
        }
      } else if (payload.type == "DISTANCE" && payload.value != null)
{
        console.log(payload);
        await createPlantHighPerCycle(getGrowCycle.id, {
          time: currentDate,
          index: payload.index,
          high: payload.value,
        });
      }
    }
  });
  socket.on("disconnect", () => {
    clearInterval(pingInterval);
    clearInterval(checkingGrowCycleData);
    const rooms = Object.keys(socket.rooms);
    rooms.forEach((room) => {
      socket.leave(room);
    });
  });
});
```

Segmen Program 4.54 Socket.io onConnect & onDisconnect

Segmen program 4.54 adalah implementasi program untuk melakukan komunikasi antara sistem IoT, aplikasi mobile, dan server di mana terjadi pengiriman perintah/instruksi dari aplikasi ke sistem IoT dan perintah/instruksi dari server ke sistem IoT. Pada bagian ini juga terdapat pengecekan data sensor terbaru dengan *user setting* dan pembuatan data sensor terbaru dari sistem IoT.

4.1.8 Check User Setting

Pada bagian ini akan di jelaskan mengenai implementasi dan programyang di buat untuk melakukan check antara parameter dalam *user setting* dan data sensor terbaru, serta mengirimkan instruksi ke sistem IoT untuk menyalakan dan mematikan komponen elektronik pada sistem IoT. Implementasi di lakukan pada *file checkingLogic.ts.*

```
import { emitUltrasonicCommand, toggleRelay } from "../server.js";
import { getFarmDetails } from "../services/farm_service.js";
import {
  getAllActiveGrowCyclesWithSettings,
  getLatestDataForGrowCycle,
} from "../services/grow_cycle_service.js";
```

Segmen Program 4.55 Check user setting dependency

```
function hasDatePassed(comparedDate: Date): boolean {
  const currentDate = new Date();
  const normalizedCurrentDate = new Date(
    currentDate.getFullYear(),
    currentDate.getMonth(),
    currentDate.getDate()
  );
  const normalizedComparedDate = new Date(
    comparedDate.getFullYear(),
    comparedDate.getMonth(),
    comparedDate.getDate()
  );
  return normalizedCurrentDate > normalizedComparedDate;
}
function isTimeWithinPeriod( currentTime: Date, startTime: Date,
endTime: Date): boolean {
  const currentHours = currentTime.getHours();
  const currentMinutes = currentTime.getMinutes();
  const startHours = startTime.getUTCHours();
  const startMinutes = startTime.getUTCMinutes();
  const endHours = endTime.getUTCHours();
  const endMinutes = endTime.getUTCMinutes();
  const current = currentHours * 60 + currentMinutes;
  const start = startHours * 60 + startMinutes;
  const end = endHours * 60 + endMinutes;
  if (start <= end) {
    return current >= start && current <= end;
```

```
  } else {
    return current >= start || current <= end;
  }
}
export function addDurationToTime(time: Date, duration: Date):
Date {
  const endTime = new Date(time);

  const timeHours = time.getUTCHours();
  const timeMinutes = time.getUTCMinutes();
  const durationHours = duration.getUTCHours();
  const durationMinutes = duration.getUTCMinutes();
  endTime.setUTCHours(timeHours + durationHours);
  endTime.setUTCMinutes(timeMinutes + durationMinutes);

  return endTime;
}
```

Segmen Program 4.56 hasDatePassed, isTimeWithinPeriod, dan addDurationToTime

Pada segmen program 4.56 adalah implementasi functions untuk melakukan perhitungan waktu dan pengecekan jika suatu waktu berada dalam suatu periode atau jika waktu sekarang sudah melewati waktu tertentu.

```
export async function checkGrowCycleData() {
  try {
    const activeGrowCycles = await getAllActiveGrowCyclesWithSettings();
    for (const {
      growCycle,
      userSettings,
      plant_high_per_cycle,
    } of activeGrowCycles) {
      const latestData = await getLatestDataForGrowCycle(growCycle.id);
      const getFarm = await getFarmDetails(growCycle.farm_id);
      if (plant_high_per_cycle.length <= 0) {
        if (getFarm != null) {
          emitUltrasonicCommand(getFarm.farm_room_id);
        }
      } else if (plant_high_per_cycle.length > 0) {
        const getLastPlantHeightDate = plant_high_per_cycle[0].time;
        if (hasDatePassed(getLastPlantHeightDate)) {
          if (getFarm != null) {
            emitUltrasonicCommand(getFarm.farm_room_id);
          }
```

```
} else if (plant_high_per_cycle.length > 0) {
  const getLastPlantHeightDate = plant_high_per_cycle[0].time;
  if (hasDatePassed(getLastPlantHeightDate)) {
    if (getFarm != null) {
      emitUltrasonicCommand(getFarm.farm_room_id);
    }
  }
}
const currentTime = new Date();
if (latestData.ppm !== null) {
  if (latestData.ppm < userSettings.min_ppm) {
    setTimeout(() => {
      const commandData = {
        type: "control",
        hardware: "relay",
        pin: 7, // pin for relay like LED, Fan, Watering
        run: "on", // on, off, or echo
      };
      if (getFarm != null) {
        toggleRelay(getFarm.farm_room_id, commandData);
      }
    }, 1500);
  } else if (latestData.ppm > userSettings.max_ppm) {
    setTimeout(() => {
      const commandData = {
        type: "control",
        hardware: "relay",
        pin: 7, // pin for relay like LED, Fan, Watering
        run: "off", // on, off, or echo
      };
      if (getFarm != null) {
        toggleRelay(getFarm.farm_room_id, commandData);
      }
    }, 1500);
  } else {
    const commandData = {
      type: "control",
      hardware: "relay",
      pin: 7, // pin for relay like LED, Fan, Watering
      run: "off", // on, off, or echo
    };
    setTimeout(() => {
      if (getFarm != null) {
        toggleRelay(getFarm.farm_room_id, commandData);
      }
    }, 1500);
  }
```

```
    }
    if (latestData.ph !== null) {
      if (latestData.ph < userSettings.min_ph) {
        setTimeout(() => {
          const commandData = {
            type: "control",
            hardware: "relay",
            pin: 5, // pin for relay like LED, Fan, Watering
            run: "on", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData);
          }
        }, 2000);
      } else if (latestData.ph > userSettings.max_ph) {
        setTimeout(() => {
          const commandData = {
            type: "control",
            hardware: "relay",
            pin: 6, // pin for relay like LED, Fan, Watering
            run: "on", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData);
          }
        }, 2000);
      } else {
        setTimeout(() => {
          const commandData1 = {
            type: "control",
            hardware: "relay",
            pin: 5, // pin for relay like LED, Fan, Watering
            run: "off", // on, off, or echo
          };
          const commandData2 = {
            type: "control",
            hardware: "relay",
            pin: 6, // pin for relay like LED, Fan, Watering
            run: "off", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData1);
            toggleRelay(getFarm.farm_room_id, commandData2);
          }
        }, 2000);
      }
    }
```

```
    if (latestData.humidity !== null) {
      if (latestData.humidity < userSettings.min_humidity) {
        setTimeout(() => {
          const commandData = {
            type: "control",
            hardware: "relay",
            pin: 3, // pin for relay like LED, Fan, Watering
            run: "on", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData);
          }
        }, 3000);
      } else if (latestData.humidity > userSettings.max_humidity) {
        setTimeout(() => {
          const commandData = {
            type: "control",
            hardware: "relay",
            pin: 3, // pin for relay like LED, Fan, Watering
            run: "off", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData);
          }
        }, 3000);
      } else if (
        latestData.humidity < userSettings.max_humidity &&
        latestData.humidity > userSettings.min_humidity
      ) {
        setTimeout(() => {
          const commandData = {
            type: "control",
            hardware: "relay",
            pin: 3, // pin for relay like LED, Fan, Watering
            run: "off", // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, commandData);
          }}, 3000);
      }
    }
    let fanCmd = "-";
    let lampCmd = "-";
    let WateringCmd = "-";
    for (const { lightSettings } of activeGrowCycles) {
      for (const lightSetting of lightSettings) {
        if (lightSetting.active_status) {
```

```
            const lightOnTime = new Date(lightSetting.light_on_time);
            const lightOffTime = new Date(lightSetting.light_off_time);
            if (isTimeWithinPeriod(currentTime, lightOnTime, lightOffTime)) {
              lampCmd = "on";
            } else {
              if (lampCmd == "-") {
                lampCmd = "off";
              }
            }
          }
        }
        setTimeout(() => {
          const lampCommandData = {
            type: "control",
            hardware: "relay",
            pin: 0, // pin for relay like LED, Fan, Watering
            run: lampCmd, // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, lampCommandData);
          }
        }, 4000);
      }
      for (const { fanSettings } of activeGrowCycles) {
        for (const fanSetting of fanSettings) {
          if (fanSetting.active_status) {
            const fanOnTime = new Date(fanSetting.fan_on_time);
            const fanOffTime = new Date(fanSetting.fan_off_time);
            if (isTimeWithinPeriod(currentTime, fanOnTime, fanOffTime)) {
              fanCmd = "on";
            } else {
              if (fanCmd == "-") {
                fanCmd = "off";
              }
            }
          }
        }
        setTimeout(() => {
          const fanCommandData = {
            type: "control",
            hardware: "relay",
            pin: 4, // pin for relay like LED, Fan, Watering
            run: fanCmd, // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, fanCommandData);
          }
}, 4500);
```

```
      }
      for (const { wateringSettings } of activeGrowCycles) {
        for (const wateringSetting of wateringSettings) {
          if (wateringSetting.active_status) {
            const wateringTime = new Date(wateringSetting.watering_time);
            const wateringDuration = new Date(wateringSetting.watering_for);
            const wateringEndTime = addDurationToTime(
              wateringTime,
              wateringDuration
            );
            if (
              isTimeWithinPeriod(currentTime, wateringTime, wateringEndTime)
            ) {
              WateringCmd = "on";
            } else {
              if (WateringCmd == "-") {
                WateringCmd = "off";
              }
            }
          }
        }
        setTimeout(() => {
          const wateringCommandData = {
            type: "control",
            hardware: "relay",
            pin: 1, // pin for relay like LED, Fan, Watering
            run: WateringCmd, // on, off, or echo
          };
          if (getFarm != null) {
            toggleRelay(getFarm.farm_room_id, wateringCommandData);
          }
        }, 5000);
      }
    }
  } catch (error) {
    console.error("Error checking grow cycle data:", error);
  }
}
```

Segmen Program 4.57 checkGrowCycleData

Pada segmen 4.57 adalah implementasi suatu *function* untuk melakukan pengecekan terhadap *user setting* dan data sensor terbaru dimana jika data tersebut berada di luar parameter *user setting*, maka akan di kirim instruksi dari server ke sistem IoT untuk menyalakan suatu komponen elektronik yang berkaitan dengan data sensor tersebut.

4.1.9 Backend API Access Point

Method	URL	Penjelasan
POST	/login	Akses poin untuk melakukan login
POST	/signup	Akses poin untuk melakukan signup
POST	/farm	Akses poin untuk mendapat semua data farm
POST	/add-farm	Akses poin untuk menambah data farm baru
POST	/edit-farm	Akses poin untuk mengubah data farm
POST	/delete-farm	Akses poin untuk menghapus data farm
POST	/user-setting	Akses poin untuk membuat user setting
POST	/user-setting-per-farm	Akses poin untuk mendapat user setting pada suatu farm
POST	/user-setting-detail	Akses poin untuk mendapat informasi detail mengenai suatu user setting
POST	/user-setting-detail/schedule	Akses poin untuk mendapat informasi mengenai jadwal penyiraman, lampu, dan kipas pada suatu iser setting
POST	/user-setting-detail/add-schedule	Akses poin untuk menambah data jadwal pada suatu user setting
POST	/user-setting-detail/delete-schedule	Akses poin untuk menghapus data jadwal pada suatu user setting
POST	/user-setting-detail/edit-user-setting	Akses poin untuk mengubah data pada suatu user setting
POST	/add-user-setting	Akses poin untuk menambah data user setting baru
POST	/delete-user-setting	Akses poin untuk menghapus suatu user setting
POST	/grow-cycle	Akses poin untuk mendapatkan data grow cycle pada suatu farm
POST	/add-grow-cycle	Akses poin untuk menambah data grow cycle baru pada suatu farm
POST	/finish-grow-cycle	Akses poin untuk mengubah status suatu grow cycle menjadi ‘selesai’
POST	/add-harvest-grow-cycle	Akses poin untuk menambahkan informasi mengenai jumlah hasil panen microgreen
POST	/grow-cycle-data	Akses poin untuk mendapat informasi lebih detail mengenai suatu grow cycle
POST	/all-user-grow-cycle-data	Akses poin untuk untuk mendapat informasi mengenai semua grow cycle yang di miliki suatu user

4.1.10 Implementasi Backend Pada Server

Pada bagian ini akan di jelaskan mengenai langkah yang di lakukan untuk mengimplementasikan aplikasi *backend* ke *server*. Akan di gunakan *server* dari Biznet Gio dengan jenis NEO LITE MS 4.4 dengan OS Ubuntu 22.04.1.

1. Melakukan akses ke dalam VPS menggunakan SSH.
2. Melakukan ‘sudo apt-get update’ dan ‘sudo apt install untuk memastikan package dalam VPS sudah di perbaharui.
3. Melakukan ‘sudo apt install mysql-server’ untuk install database MySQL.
4. Masuk ke dalam mysql dengan menggunakan perintah ‘sudo mysql’.
5. Buat *database* baru dengan nama ‘microgreen_iot_database’ dan ‘microgreen_iot_database’ untuk melakukan *migration* dari Prisma ORM.
6. Buat *user* baru dengan perintah ‘CREATE USER microgreen-backend@'localhost' IDENTIFIED WITH mysql_native_password BY microgreen; ‘.
7. Buat *user* baru dengan perintah ‘CREATE USER microgreen-backend@'%' IDENTIFIED WITH mysql_native_password BY microgreen; ‘.
8. Masukan perintah ‘GRANT ALL PRIVILEGES ON microgreen_iot_database TO 'microgreen-backend'@'%' WITH GRANT OPTION;’.
9. Masukan perintah ‘GRANT ALL PRIVILEGES ON microgreen_iot_database_shadow.* TO 'microgreen-backend'@'%' WITH GRANT OPTION;’.
10. Lakukan hal yang sama dengan *user* 'microgreen-backend'@'localhost.
11. Keluar dari mysql dan install node js versi 18.20.2 dan npm
12. Install pm2 dengan perintah ‘sudo npm install pm2 -g’ untuk menjalankan aplikasi node
13. Lakukan clone repository server backend dari github.
14. Masuk ke folder repository dan lakukan ‘npm i’ untuk install semua *dependency* yang di gunakan *server backend.*
15. Lakukan ‘npx prisma generate’ untuk menghasilkan *prisma client* dan *compile* Typescript menjadi Javascript menggunakan *CLI Tool* tsc.
16. Masuk ke dalam *folder* ‘dist’ dan jalankan ‘pm2 start server.js --name microgreen-backend’

4.2 Pemerograman Arduino

Pada bagian ini akan di jelaskan mengenai implementasi dan program pada sistem IoT berupa implementasi *source code* dan sistem yang di implementasikan untuk berinteraksi antara sistem IoT dengan server dan aplikasi untuk mengirimkan data sensor dan menerima instruksi.

4.2.1 Implementasi pada NodeMCU

Pada bagian ini akan di jelaskan mengenai implementasi program untuk melakukan koneksi dari sistem IoT ke server, program ini di gunakan pada *microcontroller* NodeMCU.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WebSocketsClient.h>
#include <SocketIoClient.h>
#include <EEPROM.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
```

Segmen Program 4.58 NodeMCU Library

Pada segmen program 4.58 adalah *library* yang digunakan pada program untuk *microcontroller* NodeMCU.

```
ESP8266WebServer server(80);

SocketIOclient socketIO;

unsigned long lastReconnectAttempt = 0;
const unsigned long reconnectInterval = 10;

#define USE_SERIAL Serial
#define JSON_DOC_SIZE 2048

#define NODEMCU_RX_PIN D2
#define NODEMCU_TX_PIN D1
SoftwareSerial MegaSerial(NODEMCU_RX_PIN, NODEMCU_TX_PIN); //
RX, TX

unsigned long duration;
bool isJoinRoom = false;

struct settings {
  char ssid[30];
  char password[30];
```

```
  char accountID[30];
} user_wifi = {};
```

Segmen Program 4.59 Variable NodeMCU

Pada segmen program 4.59 adalah *variable* yang di gunakan pada *microcontroller* NodeMCU. Dimana di inisialisasi server untuk *hosting web page input wifi & farm room* serta *struct user_wifi* . Selain itu juga di inisialisasi SocketIOcilent dan variable untuk *serial communication*.

```
bool isPing(uint8_t* payload, size_t length) {
    if (length < 8) {
        return false;
    }
    char charArray[length + 1];
    memcpy(charArray, payload, length);
    charArray[length] = '\0';
    return strcmp(charArray, "[\"ping\"]") == 0;
}
void socketIOEvent(socketIOmessageType_t type, uint8_t * payload, size_t
length) {
    switch(type) {
        case sIOtype_DISCONNECT:
            USE_SERIAL.printf("[IOc] Disconnected!\n");
            isJoinRoom = false;
            break;
        case sIOtype_CONNECT:
            USE_SERIAL.printf("[IOc] Connected to url: %s\n", payload);
            // join default namespace (no auto join in Socket.IO V3)
            socketIO.send(sIOtype_CONNECT, "/");
            break;
        case sIOtype_EVENT:
            if (isPing(payload, length)) {
                USE_SERIAL.println("[IOc] Received ping!");
                // Handle ping message as needed

            } else {
              USE_SERIAL.printf("[IOc] get event: %s\n", payload);
                handleMessage(payload, length);
            }
            break;
        case sIOtype_ACK:
            USE_SERIAL.printf("[IOc] get ack: %u\n", length);
            hexdump(payload, length);
            break;
```

```
        case sIOtype_ERROR:
            USE_SERIAL.printf("[IOc] get error: %u\n", length);
            hexdump(payload, length);
            break;
        case sIOtype_BINARY_EVENT:
            USE_SERIAL.printf("[IOc] get binary: %u\n", length);
            hexdump(payload, length);
            break;
        case sIOtype_BINARY_ACK:
            USE_SERIAL.printf("[IOc] get binary ack: %u\n", length);
            hexdump(payload, length);
            break;
    }
}
        case sIOtype_ERROR:
            USE_SERIAL.printf("[IOc] get error: %u\n", length);
            hexdump(payload, length);
            break;
        case sIOtype_BINARY_EVENT:
            USE_SERIAL.printf("[IOc] get binary: %u\n", length);
            hexdump(payload, length);
            break;
        case sIOtype_BINARY_ACK:
            USE_SERIAL.printf("[IOc] get binary ack: %u\n", length);
            hexdump(payload, length);
            break;
    }
}
```

Segmen Program 4.60 Socket IO event

Pada segmen program 4.60 adalah implementasi functions untuk melakukan ping ke server, dan untuk memproses event yang di terima oleh socket io client.

```
void handleMessage(uint8_t * payload, size_t length){
  DynamicJsonDocument doc(JSON_DOC_SIZE);
  DeserializationError error = deserializeJson(doc, payload, length);
  if (error) {
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    return;
  }
  const char* type = doc[1]["type"];
  const char* hardware = doc[1]["hardware"];
  const char* run = doc[1]["run"];
  int pin = doc[1]["pin"];
  DynamicJsonDocument docCommand(1024);
  if(strcmp(type,"control")==0){
    if(strcmp(hardware,"relay")==0){
      const char* command = (strcmp(run,"on")==0) ? "ON" : "OFF";
      docCommand["command"] = command;
      docCommand["hardware"] = hardware;
      docCommand["pin"] = pin;
      sendJson(docCommand);
      Serial.println(command);
    } else if(strcmp(hardware,"ultrasonic")==0 &&
strcmp(run,"echo")==0){
      docCommand["command"] = "ECHO";
      docCommand["hardware"] = "ultrasonic";
      docCommand["pin"] = pin;
      sendJson(docCommand);
      Serial.println("ultrasonic shot");
    }
  }
}
```

Segmen Program 4.61 Handle message

Pada segmen program 4.61 adalah implementasi untuk memproses message yang di terima dari event socket IO dari server. Message akan di proses menjadi JSON object di mana data object akan di cek dan berdasarkan data tersebut akan di kirim instruksi ke microcontroller Arduino Mega melalui serial communication.

```
void sendJson(JsonDocument& doc) {
  Serial.print("[JSON-IN]");
  String jsonString;
  serializeJson(doc, jsonString);
  Serial.println(jsonString);
}
void sendDataJsonOut(String dataType, String code, String
Inputdata) {
    DynamicJsonDocument input(1024);
    deserializeJson(input, Inputdata);
    String jsonString;

    DynamicJsonDocument data(1024);
    JsonArray arrayData = data.to<JsonArray>();

    arrayData.add("data");

    JsonObject allData = data.createNestedObject();
    String roomID = user_wifi.room;
    allData["roomID"] = roomID;
    allData["code"] = code;

    if (dataType == "SENSOR") {
        allData["type"] = input["type"];
        allData["value"] = input["value"];
        serializeJson(data, jsonString);
    }
    else if (dataType == "RELAY") {
        allData["hardware"] = input["hardware"];
        allData["status"] = input["status"];
        serializeJson(data, jsonString);
    }
    else if (dataType == "ULTRASONIC") {
        allData["type"] = input["type"];
        allData["value"] = input["value"];
        allData["index"] = input["index"];
        serializeJson(data, jsonString);
    }
    socketIO.sendEVENT(jsonString);
}
```

Segmen Program 4.62 sendJson

Pada segmen program 4.62 adalah implementasi untuk sistem IoT pada NodeMCU untuk mengirim serial message berupa perintah dalam bentuk JSON lalu di kirim ke microcontroller Arduino Mega.

```
void setup() {
  Serial.begin(9600);
  MegaSerial.begin(9600);
  EEPROM.begin(sizeof(struct settings) );
  EEPROM.get( 0, user_wifi );
  WiFi.mode(WIFI_STA);
  WiFi.begin(user_wifi.ssid, user_wifi.password);
  byte tries = 0;
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
    if (tries++ > 30) {
      WiFi.mode(WIFI_AP);
      WiFi.softAP("Setup Portal", "microgreen");
      break;
    }
  }
  Serial.println("Connected to WiFi");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.on("/",  handlePortal);
  server.on("/changeWifi",  handleChange);
  server.begin();
  socketIO.begin("103.150.191.218", 3000, "/socket.io/?EIO=4");
  socketIO.onEvent(socketIOEvent);
  socketIO.setReconnectInterval(5000);
}
void loop() {
  server.handleClient();
  socketIO.loop();
  uint64_t now = millis();
  if (socketIO.isConnected()) {
    if(isJoinRoom == false){
      Serial.println("client is connected.");
      Serial.println("client is joining room.");
      isJoinRoom = true;
      DynamicJsonDocument room(1024);
      JsonArray arrayJoinRoom = room.to<JsonArray>();
      arrayJoinRoom.add("joinRoom");
      String roomID = user_wifi.room;
      String accountID = user_wifi.accountID;
      JsonObject roomData = arrayJoinRoom.createNestedObject();
```

```
      roomData["roomID"] = roomID;
      String outputRoom;
      serializeJson(room, outputRoom);
      socketIO.sendEVENT(outputRoom);
    }
    if (MegaSerial.available() > 0) {
      String line = MegaSerial.readStringUntil('\n');
      if (line.startsWith("[DATA-OUT]")) {
        Serial.println(line);
        Serial.println("Data out");
        String jsonData = line.substring(10);
        sendDataJsonOut("SENSOR","sensor",jsonData);
      }
      else if (line.startsWith("[DATA-RELAY-OUT]")) {
        Serial.println(line);
        String jsonData = line.substring(16);
        sendDataJsonOut("RELAY","relay",jsonData);
      }
      else if (line.startsWith("[DATA-ULTRASONIC-OUT]")) {
        Serial.println(line);
        String jsonData = line.substring(21);
        sendDataJsonOut("ULTRASONIC","ultrasonic",jsonData);
      }
      else {
        Serial.println("Unknown data format from Mega");
      }
    }
    lastReconnectAttempt = millis();
  }
}
```

Segmen Program 4.63 Setup & Loop NodeMCU

Pada segmen 4.63 adalah implementasi pada setup dan loop functions di NodeMCU. inisialisasi nilai variabel pada setup function dan pembacaan pesan dari serial communication pada loop function.

```
void handlePortal() {
  if (server.method() == HTTP_POST || WiFi.status() == WL_CONNECTED) {
    strncpy(user_wifi.ssid,     server.arg("ssid").c_str(),
sizeof(user_wifi.ssid) );
    strncpy(user_wifi.password, server.arg("password").c_str(),
sizeof(user_wifi.password) );
    strncpy(user_wifi.room, server.arg("room").c_str(),sizeof(user_wifi.room) );
    user_wifi.ssid[server.arg("ssid").length()] =
user_wifi.password[server.arg("password").length()] = '\0';
    EEPROM.put(0, user_wifi);
    EEPROM.commit();
    server.send(200,   "text/html",  "<!doctype html><html lang='en'><head><meta
charset='utf-8'><meta name='viewport' content='width=device-width, initial-
scale=1'><title>Wifi Setup</title><style>*,::after,::before{box-sizing:border-
box;}body{margin:0;font-family:'Segoe UI',Roboto,'Helvetica Neue',Arial,'Noto
Sans','Liberation Sans';font-size:1rem;font-weight:400;line-
height:1.5;color:#212529;background-color:#f5f5f5;}.form-
control{display:block;width:100%;height:calc(1.5em + .75rem + 2px);border:1px
solid #ced4da;}button{border:1px solid transparent;color:#fff;background-
color:#007bff;border-color:#007bff;padding:.5rem 1rem;font-size:1.25rem;line-
height:1.5;border-radius:.3rem;width:100%}.form-signin{width:100%;max-
width:400px;padding:15px;margin:auto;}h1,p{text-align: center}</style> </head>
<body><main class='form-signin'> <h1>Wifi Setup</h1> <br/> <p>Your settings have
been saved successfully!<br />Please restart the device.</p><br/><a
href='/changeWifi'><button>Change setting</button></a></main></body></html>" );
  } else {
    server.send(200,   "text/html", "<!doctype html><html lang='en'><head><meta
charset='utf-8'><meta name='viewport' content='width=device-width, initial-
scale=1'><title>Wifi Setup</title> <style>*,::after,::before{box-sizing:border-
box;}body{margin:0;font-family:'Segoe UI',Roboto,'Helvetica Neue',Arial,'Noto
Sans','Liberation Sans';font-size:1rem;font-weight:400;line-
height:1.5;color:#212529;background-color:#f5f5f5;}.form-
control{display:block;width:100%;height:calc(1.5em + .75rem + 2px);border:1px
solid #ced4da;}button{cursor: pointer;border:1px solid
transparent;color:#fff;background-color:#007bff;border-
color:#007bff;padding:.5rem 1rem;font-size:1.25rem;line-height:1.5;border-
radius:.3rem;width:100%}.form-signin{width:100%;max-
width:400px;padding:15px;margin:auto;}h1{text-align: center}</style> </head>
<body><main class='form-signin'> <form action='/' method='post'> <h1
class=''>Wifi Setup</h1><br/><div class='form-floating'><label>SSID</label><input
type='text' class='form-control' name='ssid'> </div><div class='form-
floating'><br/><label>Password</label><input type='password' class='form-control'
name='password'></div><div class='form-floating'><br/><label>room</label><input
type='text' class='form-control'
```

```
name='room'></div><div class='form-floating'><br/><label>account
ID</label><input type='text' class='form-control'
name='accountID'></div><br/><br/><button
type='submit'>Save</button><p style='text-align:
right'></p></form></main> </body></html>" );
  }
}
void handleChange() {
  memset(&user_wifi, 0, sizeof(user_wifi));
  EEPROM.put(0, user_wifi);
  EEPROM.commit();
  handlePortal();
  WiFi.begin(user_wifi.ssid, user_wifi.password);
}
```

Segmen Program 4.64 handlePortal & handleChange

Pada segmen program 4.64 adalah implementasi functions untuk mengakses web page untuk memperbarui data wifi dan farm room id.

4.2.2 Implementasi pada Arduino Mega

Pada bagian ini akan menjelaskan mengenai implementasi sistem IoT pada *microcontroller* Arduino Mega.

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <DHT.h>
#include <GravityTDS.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

Segmen Program 4.65 Arduino Mega dependency

Pada bagian segmen program 4.65 adalah library yang di gunakan pada program untuk microcontroller Arduino Mega.

```
GravityTDS gravityTds;
#define MEGA_RX_PIN 12
#define MEGA_TX_PIN 13
SoftwareSerial NodeMCUSerial(MEGA_RX_PIN, MEGA_TX_PIN);
#define ONE_WIRE_BUS 26
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature waterTempSensors(&oneWire);
```

```
struct Hardware {
    int pin;
    bool state; // true for ON, false for OFF
    String name; // Name of the hardware
};
Hardware relayHardware[] = {
    {2, false, "lamp"},
    {3, false, "WPump"},
    {4, false, "MPump"},
    {5, false, "HumidifierFan"},
    {6, false, "Fans"},
    {7, false, "PeristalticpHUp"},
    {8, false, "PeristalticpHDown"},
    {9, false, "PeristalticNutrient"},
};
const int echoPins[] = {51, 49, 47, 45, 43, 41, 39, 37, 35, 33, 31,
29};
const int trigPins[] = {50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30,
28};
float h;
float t;
float f;
float hif;
float hic;
const int ph_pin = A0;
float Po = 0;
float PH_step;
int analog_PH_value;
double PH_Volte;
float pH4 = 1.97;
float pH7 = 1.73;
float waterTemperature = 25;
const int tds_pin = A2;
float tdsValue = 0;
int EC_Isolator = 10; // 3906 PNP TYPE TRANSISTOR
int EC_GND_Wire = 11; // 2N2222 NPN TRANSISTOR
int indexUltrasonic = 0;
bool runUltrasonic = false;
bool startMixing = false;
unsigned long previousMillisRelayUpdate = 0;
const long intervalRelayUpdate = 10000; // 3 seconds
int currentState = 0;
unsigned long previousSensorMillis = 15000;
const long sensorInterval = 15000; // Interval for all sensors
```

Segmen Program 4.66 Inisialsiasi variabel Arduino Mega

Pada segmen program 4.66 adalah inisialisasi variabel yang akan di gunakan untuk program pada microcontroller Arduino Mega. Terdapat struct, array of object, dan variabel yang di gunakan untuk data sensor.

```
void setup() {
  Serial.begin(9600); // For debugging
  NodeMCUSerial.begin(9600); // Set the baud rate to match the NodeMCU
  pinMode(ph_pin, INPUT);
  waterTempSensors.begin();
  dht.begin();
  gravityTds.setPin(tds_pin);
  gravityTds.setAref(5);
  gravityTds.setAdcRange(1024);
  gravityTds.begin();

  pinMode(EC_Isolator, OUTPUT);
  pinMode(EC_GND_Wire, OUTPUT);

  digitalWrite(EC_Isolator, HIGH);
  digitalWrite(EC_GND_Wire, LOW);

  for (int i = 0; i < sizeof(relayHardware) /
sizeof(relayHardware[0]); i++) {
      pinMode(relayHardware[i].pin, OUTPUT);
      digitalWrite(relayHardware[i].pin, HIGH);
  }

  for (int i = 0; i < 12; i++) {
    pinMode(trigPins[i], OUTPUT);
    pinMode(echoPins[i], INPUT); // Turn off all relays initially
  }
}
void loop() {
  if (NodeMCUSerial.available() > 0) {
    delay(20);
    // Serial.println("NodeMCU serial avaliable");
    String line = NodeMCUSerial.readStringUntil('\n');
    //Serial.println(jsonString);
    if (line.startsWith("[JSON-IN]")) {
      // Extract the JSON string excluding the [JSON] marker
      String jsonString = line.substring(9);
      handleJson(jsonString);
    }
```

```
  }
  unsigned long currentMillisLoop = millis();
  handleSensorReads();
  if (currentMillisLoop - previousMillisRelayUpdate >=
intervalRelayUpdate) {
    Serial.println("send data relay out");
    previousMillisRelayUpdate = currentMillisLoop;
    sendAllHardwareStates();
  }
  if(runUltrasonic == true){
      if(indexUltrasonic < 12){
      echoUltrasonic(indexUltrasonic);
      indexUltrasonic++;
      }else{
        runUltrasonic = false;
        indexUltrasonic = 0;
      }
    }
    if(startMixing == true && relayHardware[2].state == false){
      turnOnHardwareByIndex(2);
    }else if(startMixing == false &&relayHardware[2].state ==
true){
      turnOffHardwareByIndex(2);
    }
}
```

Segmen Program 4.67 Setup & loop Arduino Mega

Pada bagian segmen program 4.67 adalah implementasi *functions* setup dan loop untuk menginisialisasi serial, pin, dan sensor pada *function* setup. Sedangkan pada *function* loop, di implementasikan pembacaan data yang di terima dari *serial communication*, pembacaan data dari sensor.

```
void handleJson(String jsonString) {
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, jsonString);
  Serial.print("[RECIEVE]: ");
  Serial.println(jsonString);
  const char* command = doc["command"];
  int pin = doc["pin"];
  if (strcmp(command, "ON") == 0) {
    Serial.print("Turn on pin ");
    Serial.println(pin);
    turnOnHardwareByIndex(pin);
    if(pin == 7){// >= 5 if ph sensor work
```

```
      startMixing = true;
    }
  } else if (strcmp(command, "OFF") == 0) {
    Serial.print("Turn off pin ");
    Serial.println(pin);
    turnOffHardwareByIndex(pin);
    if(relayHardware[5].state == false && relayHardware[6].state
== false && relayHardware[7].state == false){
      startMixing = false;
    }
  } else if(strcmp(command,"ECHO")==0){
    runUltrasonic = true;
  }
}
void sendDataOut(String dataType, String type, float value,
String hardware = "", int status = -1, int index = -1) {
    DynamicJsonDocument data(1024);
    data["dataType"] = dataType;
    if (dataType == "PARAMETER" || dataType == "ULTRASONIC") {
        data["type"] = type;
        data["value"] = value;
        if (index != -1)
            data["index"] = index;
    }
    if (index != -1)
        data["index"] = index;
    if (dataType == "RELAY") {
        data["status"] = status;
        data["hardware"] = hardware;
    }
    String output;
    serializeJson(data, output);
    String prefix = "";
    if (dataType == "PARAMETER")
        prefix = "[DATA-OUT]";
    else if (dataType == "ULTRASONIC")
        prefix = "[DATA-ULTRASONIC-OUT]";
    else if (dataType == "RELAY")
        prefix = "[DATA-RELAY-OUT]";
    NodeMCUSerial.println(prefix + output);
    Serial.print(prefix);
    Serial.println(output);
}
```

Segmen Program 4.68 handleJson & sendDataOut Arduino Mega

Pada segmen program 4.68 adalah implementasi *functions* handleJson dan sendDataOut dimana *function* handleJson berfungsi untuk memproses perintah berbentuk JSON yang di terima melalui *serial communication*

```
void turnOnHardwareByIndex(int index) {
    Serial.print("on: ");
    Serial.println(relayHardware[index].name);
    digitalWrite(relayHardware[index].pin, LOW);
    relayHardware[index].state = true;
    Serial.println(relayHardware[index].state);
    sendDataOut("RELAY", "", -1, relayHardware[index].name,
relayHardware[index].state);
}

void turnOffHardwareByIndex(int index) {
  Serial.print("off: ");
    Serial.println(relayHardware[index].name);
    digitalWrite(relayHardware[index].pin, HIGH);
    relayHardware[index].state = false;
    sendDataOut("RELAY", "", -1, relayHardware[index].name,
relayHardware[index].state);
}

void sendAllHardwareStates() {
  if(currentState < 8){
    sendDataOut("RELAY", "", -1,
relayHardware[currentState].name,
relayHardware[currentState].state);
    currentState++;
  } else {
    currentState = 0;
  }
}
```

Segmen Program 4.69 turnOnHardwareByIndex, turnOffHardwareByIndex, dan sendAllHardwareStates

Pada segmen program 4.69 adalah implementasi *functions* yang berfungsi untuk menyalakan dan mematikan *relay* untuk mengontrol komponen elektronik pada sistem IoT, dan *function* sendAllHardwareStates untuk mengirim status komponen elektronik jika komponen ada pada kondisi mati atau menyala.

```
void echoUltrasonic(int index){
  long duration;
  int distance;
  digitalWrite(trigPins[index], LOW);
  delayMicroseconds(2);
  digitalWrite(trigPins[index], HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPins[index], LOW);
  duration = pulseIn(echoPins[index], HIGH);
  distance = duration * 0.034 / 2;
  float sendDistance = distance * 1.0;
  delay(1000);
  sendDataOut("ULTRASONIC", "DISTANCE", sendDistance, "",-
1,index);
}
void PHRead() {
  float sum_PH_value = 0;
  int num_readings = 10;
  for (int i = 0; i < num_readings; i++) {
    sum_PH_value += analogRead(ph_pin);
    delay(10); // Add a small delay to allow time between
readings
  }
  float mean_PH_value = sum_PH_value / num_readings;
  Serial.print("Mean Analog pH value: ");
  Serial.println(mean_PH_value);
  float PH_Volte = 3.3 / 1024.0 * mean_PH_value;
  Serial.print("Mean pH volte value: ");
  Serial.println(PH_Volte);
  PH_step = (pH4 - pH7) / 3.0;
  Po = 7.00 + ((pH7 - PH_Volte) / PH_step);
  Serial.print("pH value: ");
  Serial.println(Po);
  sendDataOut("PARAMETER", "PH", Po);
}
void dhtRead() {
  h = dht.readHumidity();
  t = dht.readTemperature();
  delay(250);
  sendDataOut("PARAMETER", "HUMIDITY", h);
  sendDataOut("PARAMETER", "AIR-TEMPRATURE", t);
}
void waterTempRead() {
  waterTempSensors.requestTemperatures();
  waterTemperature=waterTempSensors.getTempCByIndex(0);
  sendDataOut("PARAMETER", "WATER-TEMP", waterTemperature);
}
```

```
void tdsRead() {
  waterTempRead();
  const int numSamples = 10;
  float tdsSum = 0;
  for (int i = 0; i < numSamples; i++) {
    gravityTds.setTemperature(waterTemperature);
    gravityTds.update();
    tdsSum += gravityTds.getTdsValue();
    delay(100);  // Small delay between readings
  }
  tdsValue = tdsSum / numSamples;
  sendDataOut("PARAMETER", "TDS", tdsValue);
}
```

Segmen Program 4.70 Pembacaan data sensor

Pada segmen program 4.70 adalah implementasi *functions* untuk melakukan pembacaan sensor kelembaban, temperatur, TDS, pH, dan sensor ultrasonic. Setiap pembacaan selesai, data yang di dapat akan di kirim ke server dengan *websocket* melalui *serial communication* ke NodeMCU.

```
void handleSensorReads() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousSensorMillis >= sensorInterval) {
    previousSensorMillis = currentMillis;
    digitalWrite(EC_Isolator, HIGH);
    digitalWrite(EC_GND_Wire, LOW);
    delay(1000);  // Allow some time for stabilization
    Serial.println("TDS off");
    PHRead();
    delay(1000);
    digitalWrite(EC_Isolator, LOW);
    digitalWrite(EC_GND_Wire, HIGH);
    delay(1000);  // Allow some time for stabilization
    Serial.println("TDS on");
    tdsRead();
    delay(1000);
    dhtRead();
    digitalWrite(EC_Isolator, HIGH);
    digitalWrite(EC_GND_Wire, LOW);
  }
}
```

Segmen Program 4.71 handleSensorReads

Implementasi program pada segmen program 4.71 adalah *function* yang bertujuan untuk menjalankan *functions* untuk membaca data dari sensor setiap 15 detik dan mengisolasi sensor TDS agar tidak menggangu saat proses pembacaan oleh sensor pH.

4.3 Pemerograman Aplikasi Flutter

Pada bagian ini akan di bahas mengenai implementasi *source code* dan UI yang di buat pada bagian aplikasi mobile menggunakan Flutter. Aplikasi memiliki kemampuan untuk membuat akun dan *login*, membuat dan mengatur *user setting,* mengontrol sistem IoT, serta melihat data lingkungan terbaru.

```
dependencies:
  flutter:
    sdk: flutter
  http:
  shared_preferences:
  jwt_decoder:
  socket_io_client: ^2.0.3+1
  intl: ^0.19.0
  fl_chart: ^0.67.0
```

Segmen Program 4.72 Flutter dependency

Pada segmen program 4.72 adalah *library* yang di gunakan untuk membuat *feature* UI dan implementasi logika yang di gunakan pada aplikasi mobile menggunakan Flutter.

4.3.1 Implementasi Authentication

Pada bagian ini akan di bahas mengenai implementasi *source code* logika *authentication* pada aplikasi Flutter. Implementasi di buat dalam Bahasa pemrogaman Dart pada *file checkToken.dart.*

```
import "package:flutter/material.dart";
import "package:shared_preferences/shared_preferences.dart";
import "package:jwt_decoder/jwt_decoder.dart";
```

Segmen Program 4.73 checkToken.dart dependency

Pada segmen program 4.73 adalah *library* yang di gunakan untuk mengimplementasikan *function* untuk melakukan pengecekan pada JWT token dan penyimpanan token.

```
void checkTokenAndNavigate(BuildContext context, String previousRoute,
    String currentRoute, String nextRoute) async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  String? token = prefs.getString('token');
  if (token == null || token.isEmpty) {
    Navigator.pushReplacementNamed(context, '/login');
    return;
  }
  Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
  if (decodedToken.containsKey('exp')) {
    int expiryTimeInSeconds = decodedToken['exp'];
    int currentTimeInSeconds = DateTime.now().millisecondsSinceEpoch
~/ 1000;
    if (expiryTimeInSeconds <= currentTimeInSeconds) {
      Navigator.pushReplacementNamed(context, '/login');
      return;
    }
  }
```

Segmen Program 4.74 checkTokenAndNavigate

Pada segmen program 4.74 adalah implementasi dari suatu *function* untuk melakukan pengecekan terhadap JWT token, jika token tidak valid, halaman aplikasi akan di pindah ke halaman *login*.

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;
import 'package:shared_preferences/shared_preferences.dart';
Future<void> loginUser(
    String email, String password, BuildContext context) async {
  String apiUrl = '${globalVar.ApiUrlBase}/login';
  try {
    final response = await http.post(
      Uri.parse(apiUrl),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: jsonEncode(<String, String>{
        'email': email,
        'password': password,
      }),
```

```
    );

    if (response.statusCode == 200) {
      final Map<String, dynamic> responseData = json.decode(response.body);
      final String token = responseData['token'];

      SharedPreferences prefs = await SharedPreferences.getInstance();
      await prefs.setString('token', token);

      Navigator.pushReplacementNamed(
          context, '/home'); // Replace '/home' with your home screen route
    } else {
      final Map<String, dynamic> responseData = json.decode(response.body);
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text('Login failed. Please try again.'),
        backgroundColor: Colors.red,
      ));
    }
  } catch (error) {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
      content: Text('An error occurred. Please try again later.'),
      backgroundColor: Colors.red,
    ));
  }
}

class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  TextEditingController emailController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: Center(
        child: Padding(
          padding: const EdgeInsets.all(25.0),
```

```
          child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  const Padding(
                    padding: EdgeInsets.only(bottom: 30.0),
                    child: Text(
                      "LOGIN",
                      style: TextStyle(fontSize: 50, color:
Color(0xFFd5dddf)),
                    ),
                  ),
                  TextFormField(
                    controller: emailController,
                    style: const TextStyle(color: Color(0xFFb2c5b2)),
                    decoration: const InputDecoration(
                      label: Text(
                        'Email',
                        style: TextStyle(
                          color: Color(0xFFb2c5b2),
                        ),
                      ),
                      hintText: 'Enter your email',
                      hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
                      enabledBorder: OutlineInputBorder(
                          borderRadius: BorderRadius.all(Radius.circular(5)),
                          borderSide: BorderSide(color: Colors.white)),
                      focusedBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.white,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                      errorBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.red,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                    ),
                  ),
                  const SizedBox(
                    height: 20.0,
```

```
                  ),
                  TextFormField(
                    controller: passwordController,
                    style: const TextStyle(color: Color(0xFFb2c5b2)),
                    decoration: const InputDecoration(
                      label: Text(
                        'Password',
                        style: TextStyle(
                          color: Color(0xFFb2c5b2),
                        ),
                      ),
                      hintText: 'Enter your password',
                      hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
                      enabledBorder: OutlineInputBorder(
                          borderRadius:
BorderRadius.all(Radius.circular(5)),
                          borderSide: BorderSide(color: Colors.white)),
                      focusedBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.white,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                      errorBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.red,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                    ),
                  ),
                  const SizedBox(
                    height: 20,
                  ),
                  Padding(
                    padding: const EdgeInsets.symmetric(vertical: 10),
                    child: SizedBox(
                      width: 300.0,
                      height: 50.0,
                      child: ElevatedButton(
                        style: ButtonStyle(
                            shape: MaterialStateProperty.all(
                              RoundedRectangleBorder(
                                borderRadius: BorderRadius.circular(10),
                              ),
```

```
                            ),
                            backgroundColor:
                                const MaterialStatePropertyAll(Colors.green)),
                          onPressed: () async {
                            await loginUser(emailController.text,
                                passwordController.text, context);
                          },
                          child: const Text(
                            "Login",
                            style: TextStyle(color: Colors.white),
                          ),
                        ),
                      ),
                    ),
                    const Padding(
                      padding: EdgeInsets.symmetric(vertical: 5),
                      child: Text("or"),
                    ),
                    SizedBox(
                      width: 300,
                      height: 50,
                      child: ElevatedButton(
                        style: ButtonStyle(
                          backgroundColor:
                              const MaterialStatePropertyAll(Color(0xFFb2c5b2)),
                          shape: MaterialStateProperty.all(
                            RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10),
                            ),
                          ),
                        ),
                        onPressed: () {
                          emailController.text = "";
                          passwordController.text = "";
                          Navigator.pushNamed(context, '/signup');
                        },
                        child: const Text("Signup"),
                      ),
                    )
                  ],
                )),
          ),
        ));
  }
}
```

Segmen Program 4.75 Login page

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class SignupPage extends StatelessWidget {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  SignupPage({super.key});

  Future<void> signupUser(
      String email, String password, BuildContext context) async {
    String apiUrl = '${globalVar.ApiUrlBase}/signup';

    try {
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Content-Type': 'application/json; charset=UTF-8',
        },
        body: jsonEncode(<String, String>{
          'email': email,
          'password': password,
        }),
      );

      if (response.statusCode == 200) {
        final Map<String, dynamic> responseData = json.decode(response.body);
        final String token = responseData['token'];

        SharedPreferences prefs = await SharedPreferences.getInstance();
        await prefs.setString('token', token);

        Navigator.pushReplacementNamed(
            context, '/home'); // Replace '/home' with your home screen route
      } else {
```

```
        final Map<String, dynamic> responseData =
json.decode(response.body);
        print(responseData["message"]);
        String res = responseData['message'] as String;

        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Signup failed. $res'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      print('Error: $error');
      // Show error message if an exception occurs during signup
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text('An error occurred. Please try again later.'),
        backgroundColor: Colors.red,
      ));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
            color: Color(0xFFd5dddf),
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/login');
          },
        ),
      ),
      resizeToAvoidBottomInset: false,
      body: Center(
        child: Padding(
          padding: const EdgeInsets.all(25.0),
          child: Form(
              key: _formKey,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  const Padding(
                    padding: EdgeInsets.only(bottom: 30.0),
```

```
      child: Text(
        "SIGN UP",
        style: TextStyle(fontSize: 50, color: Color(0xFFd5dddf)),
      ),
    ),
    TextFormField(
      controller: emailController,
      style: const TextStyle(color: Color(0xFFb2c5b2)),
      decoration: const InputDecoration(
        label: Text(
          'Email',
          style: TextStyle(
            color: Color(0xFFb2c5b2),
          ),
        ),
        hintText: 'Enter your email',
        hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
        enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.all(Radius.circular(5)),
            borderSide: BorderSide(color: Colors.white)),
        focusedBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Colors.white,
            width: 2,
          ),
          borderRadius: BorderRadius.all(Radius.circular(10)),
        ),
        errorBorder: OutlineInputBorder(
          borderSide: BorderSide(
            color: Colors.red,
            width: 2,
          ),
          borderRadius: BorderRadius.all(Radius.circular(10)),
        ),
      ),
    ),
    const SizedBox(
      height: 20.0,
    ),
    TextFormField(
      controller: passwordController,
      style: const TextStyle(color: Color(0xFFb2c5b2)),
      decoration: const InputDecoration(
        label: Text(
          'Password',
          style: TextStyle(
            color: Color(0xFFb2c5b2),
```

```
            ),
          ),
          hintText: 'Enter your password',
          hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
          enabledBorder: OutlineInputBorder(
              borderRadius: BorderRadius.all(Radius.circular(5)),
              borderSide: BorderSide(color: Colors.white)),
          focusedBorder: OutlineInputBorder(
            borderSide: BorderSide(
              color: Colors.white,
              width: 2,
            ),
            borderRadius: BorderRadius.all(Radius.circular(10)),
          ),
          errorBorder: OutlineInputBorder(
            borderSide: BorderSide(
              color: Colors.red,
              width: 2,
            ),
            borderRadius: BorderRadius.all(Radius.circular(10)),
          ),
        ),
      ),
      const SizedBox(
        height: 20,
      ),
      Padding(
        padding: const EdgeInsets.symmetric(vertical: 10),
        child: Column(
          children: [
            SizedBox(
              width: 300.0,
              height: 50.0,
              child: ElevatedButton(
                style: ButtonStyle(
                    shape: MaterialStateProperty.all(
                      RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                    ),
                    backgroundColor: const MaterialStatePropertyAll(
                        Colors.green)),
                onPressed: () async {
                  await signupUser(emailController.text,
                      passwordController.text, context);
                },
                child: const Text(
```

```
                                "Signup",
                                style: TextStyle(color: Colors.white),
                              ),
                            ),
                          ),
                        ],
                      ),
                    ),
                  ],
                )),
          ),
        ),
      );
  }
}
```

Segmen Program 4.75 Signup page

Pada segmen program 4.74 dan 4.75 adalah implementasi UI dan logika untuk melakukan login dan signup pada aplikasi mobile menggunakan Flutter.

4.3.2 Utility Function dan Tipe data

Pada bagian ini akan di jelaskan mengenai implementasi *function* yang di gunakan untuk fungsi utilitas dan tipe data yang di gunakan untuk *function* parameter, tipe data *response*, dan parameter Flutter *component.*

```
class FarmData {
  final int farmId;
  final String farmName;
  final String farmRoomId;
  FarmData(this.farmId, this.farmName, this.farmRoomId);
}
```

Segmen Program 4.76 Farm tipe data

```
class GrowCycleData {
  int id;
  String growCycleName;
  DateTime startDate;
  DateTime? finishDate;
  bool activeStatus;
  int userSettingId;
  int farmId;
```

```
  int? totalHarvest;

  GrowCycleData({
    required this.id,
    required this.growCycleName,
    required this.startDate,
    this.finishDate,
    required this.activeStatus,
    required this.userSettingId,
    required this.farmId,
    this.totalHarvest,
  });
}

class Humidity {
  final int id;
  final DateTime time;
  final double humidity;
  final int growCycleId;

  Humidity({
    required this.id,
    required this.time,
    required this.humidity,
    required this.growCycleId,
  });

  factory Humidity.fromJson(Map<String, dynamic> json) {
    return Humidity(
      id: json['id'],
      time: DateTime.parse(json['time']),
      humidity: json['humidity'].toDouble(),
      growCycleId: json['grow_cycle_id'],
    );
  }
}

class Temperature {
  final int id;
  final DateTime time;
  final double temperature;
  final int growCycleId;

  Temperature({
    required this.id,
    required this.time,
    required this.temperature,
```

```
    required this.time,
    required this.temperature,
    required this.growCycleId,
  });

  factory Temperature.fromJson(Map<String, dynamic> json) {
    return Temperature(
      id: json['id'],
      time: DateTime.parse(json['time']),
      temperature: json['temperature'].toDouble(),
      growCycleId: json['grow_cycle_id'],
    );
  }
}

class PPM {
  final int id;
  final DateTime time;
  final double ppm;
  final int growCycleId;

  PPM({
    required this.id,
    required this.time,
    required this.ppm,
    required this.growCycleId,
  });

  factory PPM.fromJson(Map<String, dynamic> json) {
    return PPM(
      id: json['id'],
      time: DateTime.parse(json['time']),
      ppm: json['ppm'].toDouble(),
      growCycleId: json['grow_cycle_id'],
    );
  }
}

class PH {
  final int id;
  final DateTime time;
  final double ph;
  final int growCycleId;

  PH({
    required this.id,
    required this.time,
```

```
    required this.ph,
    required this.growCycleId,
  });

  factory PH.fromJson(Map<String, dynamic> json) {
    return PH(
      id: json['id'],
      time: DateTime.parse(json['time']),
      ph: json['ph'].toDouble(),
      growCycleId: json['grow_cycle_id'],
    );
  }
}

class Height {
  final int id;
  final DateTime time;
  final double high;
  final int index;
  final int growCycleId;

  Height({
    required this.id,
    required this.time,
    required this.high,
    required this.index,
    required this.growCycleId,
  });

  factory Height.fromJson(Map<String, dynamic> json) {
    return Height(
      id: json['id'],
      time: DateTime.parse(json['time']),
      high: json['high'].toDouble(),
      index: json['index'],
      growCycleId: json['grow_cycle_id'],
    );
  }
}

class CombinedResponse {
  final List<Humidity> humidity;
  final List<Temperature> temperature;
  final List<PPM> ppm;
  final List<PH> ph;
  final List<Height> height;
```

```
  CombinedResponse({
    required this.humidity,
    required this.temperature,
    required this.ppm,
    required this.ph,
    required this.height,
  });
  factory CombinedResponse.fromJson(Map<String, dynamic> json) {
    return CombinedResponse(
      humidity: (json['humidity'] as List<dynamic>)
          .map((item) => Humidity.fromJson(item))
          .toList(),
      temperature: (json['temperature'] as List<dynamic>)
          .map((item) => Temperature.fromJson(item))
          .toList(),
      ppm: (json['ppm'] as List<dynamic>)
          .map((item) => PPM.fromJson(item))
          .toList(),
      ph: (json['ph'] as List<dynamic>)
          .map((item) => PH.fromJson(item))
          .toList(),
      height: (json['plantHeight'] as List<dynamic>)
          .map((item) => Height.fromJson(item))
          .toList(),
    );
  }
}
```

Segmen Program 4.77 Grow cycle tipe data

```
import 'dart:convert';

class UserSetting {
  final int id;
  final String userSettingName;
  final double minHumidity;
  final double maxHumidity;
  final double minPPM;
  final double maxPPM;
  final double minPH;
  final double maxPH;
  final int ultrasonicShotHour;
```

```
  final int userId;
  final int farmId;
  final bool inUse;

  UserSetting({
    required this.id,
    required this.userSettingName,
    required this.minHumidity,
    required this.maxHumidity,
    required this.minPPM,
    required this.maxPPM,
    required this.minPH,
    required this.maxPH,
    required this.ultrasonicShotHour,
    required this.userId,
    required this.farmId,
    required this.inUse,
  });

  factory UserSetting.fromJson(Map<String, dynamic> json) {
    return UserSetting(
      id: json['id'],
      userSettingName: json['user_setting_name'],
      minHumidity: json['min_humidity'].toDouble(),
      maxHumidity: json['max_humidity'].toDouble(),
      minPPM: json['min_ppm'].toDouble(),
      maxPPM: json['max_ppm'].toDouble(),
      minPH: json['min_ph'].toDouble(),
      maxPH: json['max_ph'].toDouble(),
      ultrasonicShotHour: json['ultrasonic_shot_hour'],
      userId: json['user_id'],
      farmId: json['farm_id'],
      inUse: json['in_use'],
    );
  }
}

List<UserSetting> parseUserSettings(String responseBody) {
  final Map<String, dynamic> parsed = jsonDecode(responseBody);
  final List<dynamic> userSettingsJson = parsed['userSettingData'];
  return userSettingsJson
      .map<UserSetting>((json) => UserSetting.fromJson(json))
      .toList();
}
```

Segmen Program 4.78 User settings tipe data

```
import 'package:flutter/material.dart';

void notification(BuildContext context, error) {
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content:
        Text('An error occurred. Please try again later.
${error.toString()}'),
    backgroundColor: Colors.red,
  ));
}
void notificationGreen(BuildContext context, notifiaction) {
  ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content: Text(notifiaction.toString()),
    backgroundColor: Colors.green,
  ));
}
```

Segmen Program 4.79 Snackbar notification

```
import 'package:intl/intl.dart';
String formatTime(String timeString) {
  DateTime dateTime = DateTime.parse(timeString);
  return DateFormat('HH:mm').format(dateTime);
}
String formatTimeFull(String timeString) {
  DateTime dateTime = DateTime.parse(timeString);
  return DateFormat('dd-MM-yyyy').format(dateTime);
}
String convertDateToDay(String date) {
  DateTime dateTime = DateTime.parse(date);
  String dayOfWeek = DateFormat('EEEE').format(dateTime);
  return dayOfWeek;
}
double dateToDouble(DateTime date) {
  return date.millisecondsSinceEpoch.toDouble();
}
String doubleToDate(double value) {
  DateTime date =
DateTime.fromMillisecondsSinceEpoch(value.toInt());
  return DateFormat('dd').format(date);
}
```

Segmen Program 4.80 Formating time

```
String ApiUrlBase = "http://103.150.191.218:3000";
```

Segmen Program 4.81 Base API URL

```
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/util/GrowCycleData.dart';
import
'package:microgreen_monitoring_app/util/formatingTime.dart';

List<FlSpot> convertHumidityToFlSpots(List<Humidity>
humidityData) {
  List<FlSpot> spots = [];
  for (Humidity humidity in humidityData) {
    double x = dateToDouble(humidity.time);
    double y = humidity.humidity.toDouble();
    spots.add(FlSpot(x, y));
  }
  return spots;
}

List<FlSpot> convertTemperatureToFlSpots(List<Temperature>
temperatureData) {
  List<FlSpot> spots = [];
  for (Temperature temperature in temperatureData) {
    double x = dateToDouble(temperature.time);
    double y = temperature.temperature.toDouble();
    spots.add(FlSpot(x, y));
  }

  return spots;
}

List<FlSpot> convertPPMToFlSpots(List<PPM> ppmData) {
  List<FlSpot> spots = [];
  for (PPM ppm in ppmData) {
    double x = dateToDouble(ppm.time);
    double y = ppm.ppm.toDouble();
    spots.add(FlSpot(x, y));
  }

  return spots;
}
```

```
List<FlSpot> convertPHToFlSpots(List<PH> phData) {
  List<FlSpot> spots = [];

  for (PH ph in phData) {
    double x = dateToDouble(ph.time);
    double y = ph.ph.toDouble();
    spots.add(FlSpot(x, y));
  }
  return spots;
}

List<FlSpot> convertHeightToFlSpots(List<Height> heightData) {
  List<FlSpot> spots = [];

  for (Height height in heightData) {
    double x = dateToDouble(height.time);
    double y = height.high.toDouble();
    spots.add(FlSpot(x, y));
  }
  return spots;
}

Widget bottomTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold,
fontSize: 13);
  String text = doubleToDate(value);
  return SideTitleWidget(
    axisSide: meta.axisSide,
    space: 4,
    child: Text(text, style: style),
  );
}

Widget leftTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold,
fontSize: 13);
  return SideTitleWidget(
    axisSide: meta.axisSide,
    child: Text(
      '${value.toInt()}',
      style: style,
    ),
  );
}
```

```
double createIntervalHumidity(List<Humidity> humidityData) {
  try {
    return humidityData[humidityData.length - 1]
            .time
            .millisecondsSinceEpoch
            .toDouble() -
        humidityData[0].time.millisecondsSinceEpoch.toDouble();
  } catch (error) {
    return 0.0;
  }
}
double createIntervalTemperature(List<Temperature> temperatureData) {
  try {
    double intervalCalculation =
temperatureData[temperatureData.length - 1]
            .time
            .millisecondsSinceEpoch
            .toDouble() -
temperatureData[0].time.millisecondsSinceEpoch.toDouble();
    print(intervalCalculation);
    return intervalCalculation;
  } catch (error) {
    return 0.0;
  }
}
```

Segmen Program 4.82 Chart tool function

4.3.3 Implementasi Home Page

Pada bagian ini akan di jelaskan mengenai implementasi program dari halaman Home dengan menggunakan Flutter.

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

import '../auth/CheckToken.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
```

```
@override
void initState() {
  super.initState();
  checkTokenAndNavigate(context, "/login", '/home', "");
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Home', style: TextStyle(color: Color(0xFFd5dddf))),
      actions: [
        Padding(
          padding: const EdgeInsets.only(right: 20.0),
          child: GestureDetector(
            onTap: () async {
              SharedPreferences prefs = await SharedPreferences.getInstance();
              await prefs.remove('token'); // Remove token on logout
              Navigator.pushReplacementNamed(context, '/login');
            },
            child: const Icon(
              Icons.logout,
              size: 26.0,
            ),
          ),
        ),
      ],
    ),
    body: GridView.count(
      primary: false,
      padding: const EdgeInsets.all(20),
      crossAxisSpacing: 10,
      mainAxisSpacing: 10,
      crossAxisCount: 2,
      children: <Widget>[
        ElevatedButton(
          style: ElevatedButton.styleFrom(
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(10),
            ),
          ),
          child: const Text(
            "Farm",
            style: TextStyle(
              color: Color(0xFFb2c5b2),
            ),
          ),
```

```
            onPressed: () {
              Navigator.pushReplacementNamed(context, '/farm');
            },
          ),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            onPressed: () {
              Navigator.pushReplacementNamed(context, '/settings');
            },
            child: const Text(
              "Settings",
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
          ),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            child: const Text(
              "Statistic",
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
            onPressed: () {
              Navigator.pushReplacementNamed(context, '/statistic');
            },
          ),
        ],
      ),
    );
  }
}
```

Segmen Program 4.83 Home page

Pada segmen program 4.83 adalah implementasi dari halaman Home menggunakan Flutter.

4.3.4 Implementasi Farm Page

Pada bagian ini akan membahas mengenai implementasi fitur dan halaman Farm.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:jwt_decoder/jwt_decoder.dart';
import
'package:microgreen_monitoring_app/home/farm/AddFarmDialog.dart';
import 'package:microgreen_monitoring_app/util/FarmData.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import
'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class FarmPage extends StatefulWidget {
  const FarmPage({super.key});

  @override
  State<FarmPage> createState() => _FarmPageState();
}

class _FarmPageState extends State<FarmPage> {
  int countFarm = 0;
  List<dynamic> farms = [];

  Future<void> fetchData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/farm';
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');
      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken =
JwtDecoder.decode(token);
      print(decodedToken);
      int userId = decodedToken['userId'];
      print("user id: $userId");
      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
```

```
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, int>{'userId': userId}));
      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body);
        print(responseData);
        setState(() {
          farms = responseData;
          countFarm = farms.length;
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error: $error');
    }
  }

  @override
  void initState() {
    super.initState();
    fetchData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Farm", style: TextStyle(color:
Color(0xFFd5dddf))),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow back ios,
            color: Color(0xFFd5dddf),
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/home');
          },
        ),
      ),
      body: Column(
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 20),
            child: SizedBox(
```

```
            height: 70,
            width: 300,
            child: ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFF3c5148),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                ),
                onPressed: () {
                  showDialog(
                    context: context,
                    builder: (BuildContext context) {
                      return const AddFarmDialog();
                    },
                  );
                },
                child: const Text(
                  "Tambah Farm",
                  style: TextStyle(fontSize: 20, color: Color(0xFFb2c5b2)),
                )),
          ),
        ),
        const Center(
          child: Text(
            'LIST FARM',
            style: TextStyle(
                fontWeight: FontWeight.bold,
                fontSize: 25,
                color: Color(0xFFb2c5b2)),
          ),
        ),
        Expanded(
          child: countFarm > 0
              ? Padding(
                  padding: const EdgeInsets.only(top: 4.0),
                  child: ListView.builder(
                    itemCount: countFarm,
                    itemBuilder: (BuildContext context, int index) {
                      return Padding(
                        padding: const EdgeInsets.symmetric(
                            vertical: 5.0, horizontal: 8.0),
                        child: Center(
                          child: Card(
                            child: Column(
                              children: [
                                ListTile(
                                  title: Center(
```

```
                                    child: Text(
                                      'Farm Name: ${farms[index]["farm_name"]}',
                                      style: const TextStyle(
                                          fontWeight: FontWeight.w500,
                                          color: Color(0xFFb2c5b2)),
                                    ),
                                  ),
                                ),
                                ListTile(
                                  title: Center(
                                    child: Text(
                                      'Farm room ID:
${farms[index]["farm_room_id"]}',
                                      style: const TextStyle(
                                        fontWeight: FontWeight.w500,
                                        color: Color(0xFFb2c5b2),
                                      ),
                                    ),
                                  ),
                                ),
                                Row(
                                  children: [
                                    Expanded(
                                      child: Padding(
                                        padding: const EdgeInsets.symmetric(
                                            horizontal: 5.0),
                                        child: ElevatedButton(
                                            style: ElevatedButton.styleFrom(
                                              backgroundColor:
                                                  const Color(0xFF3c5148),
                                              shape: RoundedRectangleBorder(
                                                borderRadius:
                                                    BorderRadius.circular(10),
                                              ),
                                            ),
                                            onPressed: () {
                                              Navigator.pushReplacementNamed(
                                                  context, '/farm-detail',
                                                  arguments: FarmData(
                                                      farms[index]["id"],
                                                      farms[index]
                                                          ["farm_name"],
                                                      farms[index]
                                                          ["farm_room_id"]));},
                                            child: const Text(
                                              "Edit",
                                              style: TextStyle(
                                                  color: Color(0xFFb2c5b2)),
```

```
                                        )),
                                    ),
                                ),
                                Expanded(
                                  child: Padding(
                                    padding: const EdgeInsets.symmetric(
                                        horizontal: 5.0),
                                    child: ElevatedButton(
                                      style: ElevatedButton.styleFrom(
                                        backgroundColor:
                                            Colors.red.shade700,
                                        shape: RoundedRectangleBorder(
                                          borderRadius:
                                              BorderRadius.circular(10),
                                        ),
                                      ),
                                      onPressed: () {},
                                      child: const Text(
                                        "Delete",
                                        style: TextStyle(
                                          color: Color(0xFFb2c5b2),
                                        ),
                                      ),
                                    ),
                                  ),
                                )
                              ],
                            )
                          ],
                        ),
                      );
                    },
                  ),
                )
              : const Text('No Farm Data'),
          )
        ],
      ),
    );
  }
}
```

Segmen Program 4.84 Farm page

Pada segmen program 4.84 adalah implementasi halaman Farm dimana pengguna dapat menambah, merubah, atau menghapus data Farm yang di miliki oleh pengguna.

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:jwt_decoder/jwt_decoder.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class AddFarmDialog extends StatefulWidget {
  const AddFarmDialog({super.key});
  @override
  State<AddFarmDialog> createState() => _AddFarmDialogState();
}

class _AddFarmDialogState extends State<AddFarmDialog> {
  TextEditingController farmNameController = TextEditingController();
  TextEditingController farmRoomIdController = TextEditingController();

  Future<void> addFarm() async {
    String apiUrl = '${globalVar.ApiUrlBase}/add-farm';
    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
      int userId = decodedToken['userId'];
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json', // Specify the content type
        },
        body: jsonEncode(<String, dynamic>{
          'userId': userId,
          'farmName': farmNameController.text,
          'farmRoomId': farmRoomIdController.text
        }),
      );
```

```
      if (response.statusCode == 200) {
        print('Farm added successfully');
        // You can navigate to another page or show a success message here
      } else {
        print('Request failed with status code: ${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      // An error occurred, handle the exception
      print('Error: $error');
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(
            'An error occurred. Please try again later. ${error.toString()}'),
        backgroundColor: Colors.red,
      ));
    }
  }

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      backgroundColor: Color(0xFF6b8e4e),
      title: const Text('Tambah Farm', style: TextStyle(color: Colors.white)),
      content: Column(
        children: [
          const Text('Masukan data farm yang dibutuhkan.',
              style: TextStyle(color: Colors.white)),
          const SizedBox(
            height: 20.0,
          ),
          TextFormField(
            controller: farmNameController,
            style: const TextStyle(color: Color(0xFFb2c5b2)),
            decoration: const InputDecoration(
              label: Text(
                'Nama Farm',
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
              hintText: 'Masukan nama farm.',
```

```
            hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
            enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(5)),
                borderSide: BorderSide(color: Colors.white)),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
            errorBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.red,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
          ),
        ),
        const SizedBox(
          height: 20.0,
        ),
        TextFormField(
          controller: farmRoomIdController,
          style: const TextStyle(color: Color(0xFFb2c5b2)),
          decoration: const InputDecoration(
            label: Text(
              'Farm Room ID',
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
            hintText: 'masukan id room unutk farm',
            hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
            enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(5)),
                borderSide: BorderSide(color: Colors.white)),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
            errorBorder: OutlineInputBorder(
              borderSide: BorderSide(
```

```
                  color: Colors.red,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
        ],
      ),
      actions: <Widget>[
        TextButton(
          onPressed: () {
            Navigator.of(context).pop(); // Close the dialog
          },
          child: const Text(
            'Tutup',
            style: TextStyle(color: Colors.white),
          ),
        ),
        TextButton(
          onPressed: () async {
            await addFarm();
            Navigator.of(context).pop(); // Close the dialog
          },
          child: const Text('Tambah', style: TextStyle(color: Colors.white)),
        ),
      ],
    );
  }
}
```

Segmen Program 4.85 Add farm dialog

Pada segmen program 4.85 adalah implementasi UI untuk melakukan penambahan data farm baru. Komponen UI ini akan muncul dalam bentuk pop up UI Ketika pengguna menekan tombol “Add Farm”.

```
import 'dart:convert';
import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/home/farm/AddGrowCycleDialog.dart';
import 'package:microgreen_monitoring_app/home/farm/GrowCycleCard.dart';
import 'package:microgreen_monitoring_app/util/FarmData.dart';
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
```

```
    as globalVar;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;

class FarmDetailPage extends StatefulWidget {
  const FarmDetailPage({super.key});

  @override
  State<FarmDetailPage> createState() => _FarmDetailPageState();
}

class _FarmDetailPageState extends State<FarmDetailPage> {
  TextEditingController farmNameController = TextEditingController();
  TextEditingController farmRoomIdController =
TextEditingController();

  int countGrowCycle = 0;
  int farmId = 0;
  List<dynamic> growCycle = [];
  bool allComplete = true;

  Future<void> editFarm() async {
    String apiUrl = '${globalVar.ApiUrlBase}/edit-farm';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{
          'farmId': farmId,
          "updatedFieldData": {
            "farm_name": farmNameController.text,
            "farm_room_id": farmRoomIdController.text
          }
        }),
```

```
      );

      if (response.statusCode == 200) {
        Navigator.pushReplacementNamed(context, '/farm');
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
    }
  }

  Future<void> fetchGrowCycleData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/grow-cycle';

    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, dynamic>{'farmId':
farmId}));

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body);
        setState(() {
          growCycle = responseData;
          countGrowCycle = growCycle.length;
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
```

```
      notification(context, error);
      print('Error: $error');
    }
  }

  @override
  void initState() {
    super.initState();
    fetchGrowCycleData();
  }

  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as
FarmData;
    farmNameController.text = args.farmName;
    farmRoomIdController.text = args.farmRoomId;
    farmId = args.farmId;
    for (int i = 0; i < growCycle.length; i++) {
      if (growCycle[i]['active_status'] == true ||
          growCycle[i]['finish_date'] == null) {
        allComplete = false;
      }
    }
    print(args.farmRoomId);
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: const Text(
          "Farm Detail",
          style: TextStyle(
            color: Color(0xFFd5dddf),
          ),
        ),
        actions: [
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            onPressed: () async {
              await editFarm();
            },
            child: const Text(
              "Save",
              style: TextStyle(
```

```
              color: Color(0xFFb2c5b2),
            ),
          ),
        ),
      ],
      leading: GestureDetector(
        child: const Icon(
          Icons.arrow_back_ios,
        ),
        onTap: () {
          Navigator.pushReplacementNamed(context, '/farm');
        },
      ),
    ),
    body: SingleChildScrollView(
      child: Center(
        child: Padding(
          padding: const EdgeInsets.only(left: 10, right: 10, top: 20),
          child: Column(
            children: [
              TextFormField(
                controller: farmNameController,
                style: const TextStyle(color: Color(0xFFb2c5b2)),
                decoration: const InputDecoration(
                  label: Text(
                    'Nama Farm',
                    style: TextStyle(
                      color: Color(0xFFb2c5b2),
                    ),
                  ),
                  enabledBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.all(Radius.circular(5)),
                      borderSide: BorderSide(color: Colors.white)),
                  focusedBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.white,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
```

```
                ),
              ),
              const SizedBox(
                height: 20.0,
              ),
              TextFormField(
                controller: farmRoomIdController,
                style: const TextStyle(color: Color(0xFFb2c5b2)),
                decoration: const InputDecoration(
                  label: Text(
                    'ID Room Farm',
                    style: TextStyle(
                      color: Color(0xFFb2c5b2),
                    ),
                  ),
                  enabledBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.all(Radius.circular(5)),
                      borderSide: BorderSide(color: Colors.white)),
                  focusedBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.white,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                ),
              ),
              Padding(
                padding: const EdgeInsets.only(top: 15.0),
                child: Row(
                  children: [
                    Expanded(
                      child: Padding(
                        padding: const EdgeInsets.only(left: 2.0),
                        child: SizedBox(
                          height: 75,
                          child: ElevatedButton(
                            style: TextButton.styleFrom(
                              shape: RoundedRectangleBorder(
                                borderRadius: BorderRadius.circular(10),
```

```
                                ),
                              ),
                              onPressed: () {
                                if (allComplete == true) {
                                  showDialog(
                                    context: context,
                                    builder: (BuildContext context) {
                                      return AddGrowCycleDialog(
                                        farmId: farmId,
                                      );
                                    },
                                  );
                                }
                              },
                              child: Text(
                                  allComplete == true
                                      ? "Tambah Grow Cycle"
                                      : "Selesaikan grow cycle dulu",
                                  style: const TextStyle(
                                      fontSize: 15, color:
Color(0xFFb2c5b2))),),
                            ),
                          ),
                        ),
                      ],
                    ),
                  ),
                  const Padding(
                    padding: EdgeInsets.only(top: 20.0),
                    child: Text(
                      "Grow Cycle List",
                      style: TextStyle(
                          color: Color(0xFFb2c5b2),
                          fontWeight: FontWeight.bold,
                          fontSize: 25),
                    ),
                  ),
                  ListView.builder(
                    shrinkWrap: true,
                    physics: const NeverScrollableScrollPhysics(),
                    padding: const EdgeInsets.all(8),
                    itemCount: countGrowCycle,
                    itemBuilder: (context, index) {
                      if (growCycle[index]['active_status'] == true ||
                          growCycle[index]['finish_date'] == null) {
                        allComplete = false;
                      }
```

```
                    return growCycle[index]['active_status'] == true
                        ? GrowCycleCard(
                            data: growCycle[index],
                            farmData: args,
                          )
                        : const SizedBox.shrink();
                  },
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Segmen Program 4.86 Farm detail page

Pada segmen program 4.86 adalah implementasi halaman Farm Detail dimana pengguna dapat melihat informasi detail mengenai suatu farm yang di miliki oleh pengguna.

```
import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/completeGrowCycle.dart';
import 'package:microgreen_monitoring_app/util/formatingTime.dart';

class GrowCycleCard extends StatelessWidget {
  const GrowCycleCard({super.key, required this.data, this.farmData});
  final data;
  final farmData;

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          ListTile(
            title: Text(data['grow_cycle_name']),
            titleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 20),
            subtitle: Row(
```

```
              children: [
                Expanded(
                    child: Text(
                        'Start date: ${formatTimeFull(data['start_date'])}')),
                Expanded(
                  child: Text(
                    'Finish date: ${data['finish_date'] != null ?
formatTimeFull(data['finish_date']) : "not finish"}',
                    textAlign: TextAlign.right,
                  ),
                ),
              ],
            ),
            subtitleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 15),
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.end,
            children: <Widget>[
              TextButton(
                child: const Text(
                  'Detail',
                  style: TextStyle(
                      color: Color(0xFFb2c5b2),
                      fontWeight: FontWeight.bold,
                      fontSize: 20),
                ),
                onPressed: () {
                  Navigator.pushReplacementNamed(context, '/grow-cycle-detail',
                      arguments: {data, farmData});
                },
              ),
              const SizedBox(width: 8),
              TextButton(
                child: Text(
                  data['finish_date'] != null ? 'Completed' : "Complete",
                  style: const TextStyle(
                      color: Color(0xFFb2c5b2),
                      fontWeight: FontWeight.bold,
                      fontSize: 20),
                ),
                onPressed: () {
                  if (data['finish_date'] == null ||
                      data['active_status'] == true) {
```

```
                    finishGrowCycle(context, data['id']);
                  }
                },
              ),
              const SizedBox(width: 8),
            ],
          ),
        ],
      ),
    );
  }
}
```

Segmen Program 4.87 grow cycle card

Pada segmen program 4.87 adalah implementasi komponen card untuk menampilkan data grow cycle pada halaman Farm detail.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:microgreen_monitoring_app/util/UserSettingData.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class AddGrowCycleDialog extends StatefulWidget {
  const AddGrowCycleDialog({super.key, required this.farmId});
  final int farmId;

  @override
  State<AddGrowCycleDialog> createState() => _AddGrowCycleDialogState();
}

class _AddGrowCycleDialogState extends State<AddGrowCycleDialog> {
  UserSetting? _selectedUserSetting;
  List<UserSetting> userSettings = [];

  TextEditingController growCycleNameController = TextEditingController();

  Future<void> addGrowCycle() async {
    String apiUrl = '${globalVar.ApiUrlBase}/add-grow-cycle';

    try {
```

```
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{
          'farmId': widget.farmId,
          'userSettingId': _selectedUserSetting!.id,
          'growCycleData': {
            "grow_cycle_name": growCycleNameController.text,
            "active_status": true
          }
        }),
      );

      if (response.statusCode == 200) {
        print('Grow Cycle added successfully');
      } else {
        print('Request failed with status code:
${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      print('Error: $error');
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(
            'An error occurred. Please try again later.
${error.toString()}'),
```

```
        backgroundColor: Colors.red,
      ));
    }
  }

  Future<void> fetchData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/user-setting-per-
farm';
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(
          <String, int>{'farmId': widget.farmId},
        ),
      );

      if (response.statusCode == 200) {
        setState(() {
          userSettings = parseUserSettings(response.body);
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error: $error');
    }
  }

  @override
  void initState() {
    super.initState();
    fetchData();
  }
```

```
@override
Widget build(BuildContext context) {
  return AlertDialog(
    backgroundColor: Color(0xFF6b8e4e),
    title: const Text('Tambah Grow Cycle',
        style: TextStyle(color: Colors.white)),
    content: Column(
      children: [
        const Text('Masukan data Grow Cycle yang dibutuhkan.',
            style: TextStyle(color: Colors.white)),
        const SizedBox(
          height: 20.0,
        ),
        DropdownButtonFormField<UserSetting>(
          value: _selectedUserSetting,
          onChanged: (UserSetting? newValue) {
            setState(() {
              _selectedUserSetting = newValue;
            });
          },
          items: userSettings.map((UserSetting userSetting) {
            return DropdownMenuItem<UserSetting>(
              value: userSetting,
              child: Text(
                userSetting.userSettingName,
                style: const TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
            );
          }).toList(),
          decoration: const InputDecoration(
            label: Text(
              'Pilih user setting',
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
            border: OutlineInputBorder(
              borderSide: BorderSide(color: Colors.white),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
```

```
                borderRadius:
BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
          TextFormField(
            controller: growCycleNameController,
            style: const TextStyle(color: Color(0xFFb2c5b2)),
            decoration: const InputDecoration(
              label: Text(
                'Nama Grow Cycle',
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
              hintText: 'Masukan nama Grow Cycle.',
              hintStyle: TextStyle(color: Color(0xFFb2c5b2)),
              enabledBorder: OutlineInputBorder(
                  borderRadius:
BorderRadius.all(Radius.circular(5)),
                  borderSide: BorderSide(color: Colors.white)),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.white,
                  width: 2,
                ),
                borderRadius:
BorderRadius.all(Radius.circular(10)),
              ),
              errorBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.red,
                  width: 2,
                ),
                borderRadius:
BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
          const SizedBox(
            height: 20.0,
          ),
        ],
      ),
      actions: <Widget>[
```

```
        TextButton(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: const Text(
            'Tutup',
            style: TextStyle(color: Colors.white),
          ),
        ),
        TextButton(
          onPressed: () async {
            await addGrowCycle();
            Navigator.of(context).pop();
          },
          child: const Text('Tambah', style: TextStyle(color:
Colors.white)),
        ),
      ],
    );
  }
```

Segmen Program 4.88 Add grow cycle dialog

Pada segmen program 4.88 adalah implementasi UI untuk melakukan pembuatan grow cycle baru.

4.3.5 Implementasi User Setting Page

Pada bagian ini akan membahas mengenai implementasi UI dan logika untuk halaman User Setting.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:jwt_decoder/jwt_decoder.dart';
import 'package:microgreen monitoring app/util/ScaffoldNotification.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class AddUserSettingData {
  final int farmId;

  AddUserSettingData(this.farmId);
}
```

```
class UserSettingPage extends StatefulWidget {
  const UserSettingPage({super.key});

  @override
  State<UserSettingPage> createState() => _UserSettingPageState();
}

class _UserSettingPageState extends State<UserSettingPage> {
  int countUserSetting = 0;
  List<dynamic> userSettings = [];
  Future<void> deleteUserSetting(int userSettingId) async {
    String apiUrl = '${globalVar.ApiUrlBase}/delete-user-setting';
    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');
      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(
          <String, int>{'userSettingId': userSettingId},
        ),
      );
      if (response.statusCode == 200) {
        print("delete success.");
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
      print('Error: $error');
    }
  }

  Future<void> fetchData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/user-setting';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
```

```
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
      print(decodedToken);
      int userId = decodedToken['userId'];

      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(
          <String, int>{'userId': userId},
        ),
      );

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body);
        setState(() {
          userSettings = responseData['userSettingData'];
          countUserSetting = userSettings.length;
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error: $error');
    }
  }

  @override
  void initState() {
    super.initState();
    fetchData();
}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("User Settings",
            style: TextStyle(color: Color(0xFFd5dddf))),
```

```
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/home');
          },
        ),
      ),
      body: Column(
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 20),
            child: SizedBox(
              height: 70,
              width: 300,
              child: ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                  onPressed: () {
                    Navigator.pushReplacementNamed(context, '/add-
setting');
                  },
                  child: const Text(
                    "Tambah user setting",
                    style: TextStyle(fontSize: 20, color:
Color(0xFFd5dddf)),
                  )),
            ),
          ),
          const Center(
            child: Text(
              'LIST USER SETTING',
              style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 25,
                  color: Color(0xFFd5dddf)),
            ),
          ),
          Expanded(
            child: countUserSetting > 0
                ? Padding(
                    padding: const EdgeInsets.only(top: 4.0),
                    child: ListView.builder(
```

```
                    itemCount: countUserSetting,
                    itemBuilder: (BuildContext context, int index) {
                      return Padding(
                        padding: const EdgeInsets.symmetric(
                            vertical: 5.0, horizontal: 8.0),
                        child: Center(
                          child: Card(
                            child: Column(
                              children: [
                                ListTile(
                                  title: Center(
                                    child: Text(
                                      'Setting Name:
${userSettings[index]["user_setting_name"]}',
                                        style: const TextStyle(
                                            fontWeight: FontWeight.w500,
                                            color: Color(0xFFb2c5b2)),
                                    ),
                                  ),),
                                ListTile(
                                  title: Center(
                                    child: Text(
                                      'Farm ID: ${userSettings[index]["farm_id"]}',
                                      style: const TextStyle(
                                        fontWeight: FontWeight.w500,
                                        color: Color(0xFFb2c5b2),
                                      ),
                                    ),
                                  ),),
                                Row(
                                  children: [
                                    Expanded(
                                      child: Padding(
                                        padding: const EdgeInsets.symmetric(
                                            horizontal: 5.0),
                                        child: ElevatedButton(
                                            style: ElevatedButton.styleFrom(
                                              backgroundColor:
                                                  const Color(0xFF3c5148),
                                              shape: RoundedRectangleBorder(
                                                borderRadius:
                                                    BorderRadius.circular(10),
                                              ),
                                            ),
                                            onPressed: () {
                                              Navigator.pushReplacementNamed(
                                                  context, '/setting-detail',
```

```
                                        arguments:
                                            userSettings[index]
                                                ["id"]);
                                    },
                                    child: const Text(
                                      "Edit",
                                      style: TextStyle(
                                          color: Color(0xFFb2c5b2)),
                                    )),
                              ),
                            ),
                            Expanded(
                              child: Padding(
                                padding: const EdgeInsets.symmetric(
                                    horizontal: 5.0),
                                child: ElevatedButton(
                                  style: ElevatedButton.styleFrom(
                                    backgroundColor:
                                        Colors.red.shade700,
                                    shape: RoundedRectangleBorder(
                                      borderRadius:
                                          BorderRadius.circular(10),
                                    ),
                                  ),
                                  onPressed: () async {
                                    await deleteUserSetting(
                                        userSettings[index]["id"]);
                                  },
                                  child: const Text(
                                    "Delete",
                                    style: TextStyle(
                                      color: Color(0xFFb2c5b2),
                                    ),
                                  ),
                                ),
                              ),
                            )
                          ],
                        )
                      ],
                    ),
                  ),
                ),
              );
            },
          ),
```

```
          )
          : const Text('No Farm Data'),
      )
    ],
   ),
  );
 }
}
```

Segmen Program 4.89 User setting page

Pada segmen program 4.89 adalah implementasi dan logika terhadap halaman user setting dimana pengguna dapat melihat data user setting yang di miliki pengguna, menghapus data user setting, dan berpindah ke halaman detail user setting atau berpindah ke halaman add user setting.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:jwt_decoder/jwt_decoder.dart';
import 'package:http/http.dart' as http;
import
'package:microgreen_monitoring_app/util/ScaffoldNotification.dar
t';
import 'package:shared_preferences/shared_preferences.dart';
import
'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class Farm {
  final int farmId;
  final String farmName;

  Farm(this.farmId, this.farmName);
}

class AddUserSettingDetailPage extends StatefulWidget {
  const AddUserSettingDetailPage({super.key});

  @override
  State<AddUserSettingDetailPage> createState() =>
      _AddUserSettingDetailPageState();
```

```
}

class _AddUserSettingDetailPageState extends
State<AddUserSettingDetailPage> {
  TextEditingController userSettingNameController =
TextEditingController();
  TextEditingController maxHumidityController = TextEditingController();
  TextEditingController minHumidityController = TextEditingController();
  TextEditingController maxPPMController = TextEditingController();
  TextEditingController minPPMController = TextEditingController();
  TextEditingController maxPHController = TextEditingController();
  TextEditingController minPHController = TextEditingController();
  TextEditingController ultrasonicHourController =
TextEditingController();
  TextEditingController inUseController = TextEditingController();
  Farm? _selectedFarm;
  List<Farm> farms = [];

  Future<void> fetchFarmData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/farm';
    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
      print(decodedToken);
      int userId = decodedToken['userId'];
      print("user id: $userId");
      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, int>{'userId': userId}));

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body) as List<dynamic>;
        print(responseData);
        setState(() {
          farms.clear();
```

```
          farms.addAll(
              responseData.map((data) => Farm(data['id'],
data['farm_name'])));
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error: $error');
    }
  }

  Future<void> addUserSetting() async {
    String apiUrl = '${globalVar.ApiUrlBase}/add-user-setting';
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken =
JwtDecoder.decode(token);
      int userId = decodedToken['userId'];

      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{
          'userId': userId,
          'farmId': _selectedFarm!.farmId,
          'settingsData': {
            "user_setting_name": userSettingNameController.text,
            "min_humidity":
int.parse(minHumidityController.text),
            "max_humidity":
int.parse(maxHumidityController.text),
            "min_ppm": int.parse(minPPMController.text),
            "max_ppm": int.parse(maxPPMController.text),
```

```
            "min_ph": int.parse(minPHController.text),
            "max_ph": int.parse(maxPHController.text),
            "ultrasonic_shot_hour":
int.parse(ultrasonicHourController.text),
            "in_use": false
          }
        }),
      );
      if (response.statusCode == 200) {
        print('Farm added successfully');
      } else {
        print('Request failed with status code:
${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      print('Error: $error');
      notification(context, error);
    }
  }

  @override
  void initState() {
    super.initState();
    fetchFarmData();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: const Text("Detail Setting",
            style: TextStyle(color: Color(0xFFd5dddf))),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context,
'/settings');
```

```
          },
        ),
      ),
      body: Column(
        children: [
          DropdownButtonFormField<Farm>(
            value: _selectedFarm,
            onChanged: (Farm? newValue) {
              setState(() {
                _selectedFarm = newValue;
              });
            },
            items: farms.map((Farm farm) {
              return DropdownMenuItem<Farm>(
                value: farm,
                child: Text(
                  farm.farmName,
                  style: const TextStyle(
                    color: Color(0xFFb2c5b2),
                  ),
                ),
              );
            }).toList(),
            decoration: const InputDecoration(
              label: Text(
                'Select a farm',
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
              border: OutlineInputBorder(
                borderSide: BorderSide(color: Colors.white),
              ),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.white,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
          const SizedBox(height: 20),
          TextFormField(
            controller: userSettingNameController,
            style: const TextStyle(color: Color(0xFFb2c5b2)),
            decoration: const InputDecoration(
```

```
              label: Text(
                'Nama user setting',
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
              enabledBorder: OutlineInputBorder(
                  borderRadius: BorderRadius.all(Radius.circular(5)),
                  borderSide: BorderSide(color: Colors.white)),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.white,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
              errorBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.red,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(top: 7.0),
            child: Row(
              children: [
                Expanded(
                  child: TextFormField(
                    controller: maxHumidityController,
                    style: const TextStyle(color: Color(0xFFb2c5b2)),
                    decoration: const InputDecoration(
                      label: Text(
                        'Max Humidity',
                        style: TextStyle(
                          color: Color(0xFFb2c5b2),
                        ),),
                      enabledBorder: OutlineInputBorder(
                          borderRadius:
BorderRadius.all(Radius.circular(5)),
                          borderSide: BorderSide(color: Colors.white)),
                      focusedBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.white,
                          width: 2,
```

```
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                ),
              ),
            ),
            Expanded(
              child: TextFormField(
                controller: minHumidityController,
                style: const TextStyle(color: Color(0xFFb2c5b2)),
                decoration: const InputDecoration(
                  label: Text(
                    'Min Humidity',
                    style: TextStyle(
                      color: Color(0xFFb2c5b2),
                    ),
                  ),
                  enabledBorder: OutlineInputBorder(
                      borderRadius: BorderRadius.all(Radius.circular(5)),
                      borderSide: BorderSide(color: Colors.white)),
                  focusedBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.white,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
```

```
          Padding(
            padding: const EdgeInsets.only(top: 7.0),
            child: Row(
              children: [
                Expanded(
                  child: TextFormField(
                    controller: maxPPMController,
                    style: const TextStyle(color: Color(0xFFb2c5b2)),
                    decoration: const InputDecoration(
                      label: Text(
                        'Max PPM',
                        style: TextStyle(
                          color: Color(0xFFb2c5b2),
                        ),
                      ),
                      enabledBorder: OutlineInputBorder(
                          borderRadius: BorderRadius.all(Radius.circular(5)),
                          borderSide: BorderSide(color: Colors.white)),
                      focusedBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.white,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                      errorBorder: OutlineInputBorder(
                        borderSide: BorderSide(
                          color: Colors.red,
                          width: 2,
                        ),
                        borderRadius: BorderRadius.all(Radius.circular(10)),
                      ),
                    ),
                  ),
                ),
                Expanded(
                  child: TextFormField(
                    controller: minPPMController,
                    style: const TextStyle(color: Color(0xFFb2c5b2)),
                    decoration: const InputDecoration(
                      label: Text(
                        'Min PPM',
                        style: TextStyle(
                          color: Color(0xFFb2c5b2),
                        ),
                      ),
                      enabledBorder: OutlineInputBorder(
```

```
                    borderRadius: BorderRadius.all(Radius.circular(5)),
                    borderSide: BorderSide(color: Colors.white)),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Colors.white,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Colors.red,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
              ),
            ),
          ),
        ],
      ),
    ),
    Padding(
      padding: const EdgeInsets.only(top: 7.0),
      child: Row(
        children: [
          Expanded(
            child: TextFormField(
              controller: maxPHController,
              style: const TextStyle(color: Color(0xFFb2c5b2)),
              decoration: const InputDecoration(
                label: Text(
                  'Max pH',
                  style: TextStyle(
                    color: Color(0xFFb2c5b2),
                  ),
                ),
                enabledBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.all(Radius.circular(5)),
                    borderSide: BorderSide(color: Colors.white)),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Colors.white,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
```

```
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.red,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                  ),
                ),
              ),
              Expanded(
                child: TextFormField(
                  controller: minPHController,
                  style: const TextStyle(color: Color(0xFFb2c5b2)),
                  decoration: const InputDecoration(
                    label: Text(
                      'Min pH',
                      style: TextStyle(
                        color: Color(0xFFb2c5b2),
                      ),
                    ),
                    enabledBorder: OutlineInputBorder(
                        borderRadius: BorderRadius.all(Radius.circular(5)),
                        borderSide: BorderSide(color: Colors.white)),
                    focusedBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.white,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.red,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(top: 7.0),
          child: TextFormField(
```

```
            controller: ultrasonicHourController,
            style: const TextStyle(color: Color(0xFFb2c5b2)),
            decoration: const InputDecoration(
              label: Text(
                'Jam Ultrasonic',
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                ),
              ),
              enabledBorder: OutlineInputBorder(
                  borderRadius: BorderRadius.all(Radius.circular(5)),
                  borderSide: BorderSide(color: Colors.white)),
              focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.white,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
              errorBorder: OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.red,
                  width: 2,
                ),
                borderRadius: BorderRadius.all(Radius.circular(10)),
              ),
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(top: 20.0),
          child: SizedBox(
            height: 75,
            width: 420,
            child: ElevatedButton(
                style: TextButton.styleFrom(
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                ),
                onPressed: () async {
                  await addUserSetting();
                  Navigator.pushReplacementNamed(context, '/settings');
                },
                child: const Text(
                  "Submit",
                  style: TextStyle(fontSize: 20, color: Color(0xFFd5dddf)),
```

```
                    )),
              ),
            )
          ],
        ),
      );
    }
  }
```

Segmen Program 4.90 Add user setting page

Pada segmen program 4.90 adalah implementasi UI dan logika untuk menambah data user setting baru.

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:jwt_decoder/jwt_decoder.dart';
import
'package:microgreen_monitoring_app/home/user_setting/ScheduleCard.dart';
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class Farm {
  final int farmId;
  final String farmName;
  Farm(this.farmId, this.farmName);
}

void _showEditDialog(BuildContext context, String type, int userSettingId) {
  Future<void> addUserSetting(
      int userSettingId, String onTime, String offTime) async {
    String apiUrl = '${globalVar.ApiUrlBase}/user-setting-detail/add-
schedule';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
```

```
        return;
      }
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{
          "userSettingId": userSettingId,
          "onTime": onTime,
          "offTime": offTime,
          "type": type
        }),
      );
      if (response.statusCode == 200) {
        print('Farm added successfully');
      } else {
        print('Request failed with status code:
${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      notification(context, error);
    }
  }

  showDialog(
    context: context,
    builder: (BuildContext context) {
      TextEditingController onTimeController =
TextEditingController();
      TextEditingController offTimeController =
TextEditingController();
      return AlertDialog(
        backgroundColor: const Color(0xFF6b8e4e),
        title: Text(
          'Tambah jadwal $type',
```

```
          style: const TextStyle(
            color: Color(0xFFb2c5b2),
          ),
        ),
        contentTextStyle: const TextStyle(
          color: Color(0xFFb2c5b2),
        ),
        content: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              TextField(
                controller: onTimeController,
                decoration: const InputDecoration(
                  labelText: 'On Time (HH:mm)',
                  labelStyle: TextStyle(
                    color: Color(0xFFb2c5b2),
                  ),
                  enabledBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.all(Radius.circular(5)),
                      borderSide: BorderSide(color: Colors.white)),
                  focusedBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.white,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                )),
              const SizedBox(
                height: 10,
              ),
              TextField(
                controller: offTimeController,
                decoration: const InputDecoration(
                  labelText: 'Off Time (HH:mm)',
                  enabledBorder: OutlineInputBorder(
                      borderRadius:
BorderRadius.all(Radius.circular(5)),
```

```
                    borderSide: BorderSide(color: Colors.white)),
                focusedBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Colors.white,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
                errorBorder: OutlineInputBorder(
                  borderSide: BorderSide(
                    color: Colors.red,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
              ),),
            Row(
              children: [
                const Text('Active Status:'),
                Radio(value: true, groupValue: true, onChanged: (value){}),
                const Text('Active'),
                Radio(value: false, groupValue: true, onChanged: (value){}),
                const Text('Inactive'),
              ],
            ),
          ],
        ),
      ),
      actions: <Widget>[
        TextButton(
          onPressed: () {
            Navigator.of(context).pop();
          },
          child: const Text(
            'Cancel',
            style: TextStyle(
              color: Color(0xFFb2c5b2),
            ),
          ),
        ),
        ElevatedButton(
          onPressed: () {
            addUserSetting(
                userSettingId, onTimeController.text,
offTimeController.text);
```

```
              Navigator.of(context).pop();
            },
            child: const Text(
              'Save',
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
          ),
        ),
      ],
    );
  },
 );
}

class UserSettingDetailPage extends StatefulWidget {
  const UserSettingDetailPage({super.key});

  @override
  State<UserSettingDetailPage> createState() =>
_UserSettingDetailPageState();
}

class _UserSettingDetailPageState extends State<UserSettingDetailPage> {
  int countJadwal = 0;
  Map<String, dynamic> jadwal = {};
  Farm? _selectedFarm;
  List<Farm> farms = [];
  int userSettingId = 0;

  TextEditingController userSettingNameController =
TextEditingController();
  TextEditingController maxHumidityController = TextEditingController();
  TextEditingController minHumidityController = TextEditingController();
  TextEditingController maxPPMController = TextEditingController();
  TextEditingController minPPMController = TextEditingController();
  TextEditingController maxPHController = TextEditingController();
  TextEditingController minPHController = TextEditingController();
  TextEditingController ultrasonicHourController = TextEditingController();
  TextEditingController inUseController = TextEditingController();

  Future<void> fetchFarmData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/farm';

    try {
```

```
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');
      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      Map<String, dynamic> decodedToken =
JwtDecoder.decode(token);
      int userId = decodedToken['userId'];
      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, int>{'userId': userId}));

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body) as
List<dynamic>;
        setState(() {
          farms.clear();
          farms.addAll(
              responseData.map((data) => Farm(data['id'],
data['farm_name'])));
        });
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
    }
  }
  Future<void> fetchUserSettingData() async {
    await fetchFarmData();
    String apiUrl = '${globalVar.ApiUrlBase}/user-setting-
detail';
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
```

```
        print('Token not found');
        return;
      }
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{'userSettingId':
userSettingId}),
      );
      if (response.statusCode == 200) {
        print('user setting fetched.');
        final responseData = jsonDecode(response.body);
        userSettingNameController.text =
            responseData['userSettingData']['user_setting_name'];
        maxHumidityController.text =
            responseData['userSettingData']['max_humidity'].toString();
        minHumidityController.text =
            responseData['userSettingData']['min_humidity'].toString();
        maxPPMController.text =
            responseData['userSettingData']['max_ppm'].toString();
        minPPMController.text =
            responseData['userSettingData']['min_ppm'].toString();
        maxPHController.text =
            responseData['userSettingData']['max_ph'].toString();
        minPHController.text =
            responseData['userSettingData']['min_ph'].toString();
        ultrasonicHourController.text =

responseData['userSettingData']['ultrasonic_shot_hour'].toString();
        setState(() {
          for (var data in farms) {
            if (data.farmId == responseData['userSettingData']['farm_id'])
{
              _selectedFarm = data;
            }
          }
        });
      } else {
        print('Request failed with status code: ${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
```

```
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      print('Error: $error');
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(
            'An error occurred. Please try again later.
${error.toString()}'),
        backgroundColor: Colors.red,
      ));
    }
    await fetchScheduleData();
  }

  Future<void> fetchScheduleData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/user-setting-
detail/schedule';

    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, int>{'userSettingId':
userSettingId}));

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body);
        setState(() {
          jadwal = responseData;
          countJadwal = jadwal.length;
        });
      } else {
```

```
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error: $error');
    }
  }

  Future<void> editUserSetting() async {
    String apiUrl =
        '${globalVar.ApiUrlBase}/user-setting-detail/edit-user-
setting';
    try {
      SharedPreferences prefs = await
SharedPreferences.getInstance();
      String? token = prefs.getString('token');
      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }
      final response = await http.post(
        Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization':
              'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{
          'userSettingId': userSettingId,
          'settingsData': {
            "user_setting_name": userSettingNameController.text,
            "min_humidity":
int.parse(minHumidityController.text),
            "max_humidity":
int.parse(maxHumidityController.text),
            "min_ppm": int.parse(minPPMController.text),
            "max_ppm": int.parse(maxPPMController.text),
            "min_ph": int.parse(minPHController.text),
            "max_ph": int.parse(maxPHController.text),
            "ultrasonic_shot_hour":
int.parse(ultrasonicHourController.text),
            "in_use": false
          }
        }),
      );
```

```
      if (response.statusCode == 200) {
        print('User setting save successfully');
        Navigator.pushReplacementNamed(context, '/user-setting');
      } else {
        print('Request failed with status code:
${response.statusCode}');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text(
              'An error occurred. Please try again later.
${response.statusCode}'),
          backgroundColor: Colors.red,
        ));
      }
    } catch (error) {
      print('Error: $error');
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(
            'An error occurred. Please try again later.
${error.toString()}'),
        backgroundColor: Colors.red,
      ));
    }
  }

  @override
  void initState() {
    super.initState();
    fetchUserSettingData();
  }

  @override
  Widget build(BuildContext context) {
    userSettingId = ModalRoute.of(context)!.settings.arguments as
int;
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: const Text("Detail Setting",
            style: TextStyle(color: Color(0xFFd5dddf))),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/settings');
```

```
          },
        ),
        actions: [
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(10),
              ),
            ),
            onPressed: () async {
              await editUserSetting();
            },
            child: const Text(
              "Save",
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
          ),
        ],
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            DropdownButtonFormField<Farm>(
              value: _selectedFarm,
              onChanged: (Farm? newValue) {
                setState(() {
                  _selectedFarm = newValue;
                });
              },
              items: farms.map((Farm farm) {
                return DropdownMenuItem<Farm>(
                  value: farm,
                  child: Text(
                    farm.farmName,
                    style: const TextStyle(
                      color: Color(0xFFb2c5b2),
                    ),
                  ),
                );
              }).toList(),
              decoration: const InputDecoration(
                label: Text(
                  'Select a farm',
                  style: TextStyle(
```

```
                color: Color(0xFFb2c5b2),
              ),
            ),
            border: OutlineInputBorder(
              borderSide: BorderSide(color: Colors.white),
            ),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
          ),
        ),
        const SizedBox(height: 20),
        TextFormField(
          controller: userSettingNameController,
          style: const TextStyle(color: Color(0xFFb2c5b2)),
          decoration: const InputDecoration(
            label: Text(
              'Nama user setting',
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
            enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(5)),
                borderSide: BorderSide(color: Colors.white)),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
            errorBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.red,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
          ),
        ),
        Padding(
```

```
              padding: const EdgeInsets.only(top: 7.0),
              child: Row(
                children: [
                  Expanded(
                    child: TextFormField(
                      controller: maxHumidityController,
                      style: const TextStyle(color: Color(0xFFb2c5b2)),
                      decoration: const InputDecoration(
                        label: Text(
                          'Max Humidity',
                          style: TextStyle(
                            color: Color(0xFFb2c5b2),
                          ),
                        ),
                        enabledBorder: OutlineInputBorder(
                            borderRadius:
BorderRadius.all(Radius.circular(5)),
                            borderSide: BorderSide(color: Colors.white)),
                        focusedBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.white,
                            width: 2,
                          ),
                          borderRadius:
BorderRadius.all(Radius.circular(10)),
                        ),
                        errorBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.red,
                            width: 2,
                          ),
                          borderRadius:
BorderRadius.all(Radius.circular(10)),
                        ),),),),),
                  Expanded(
                    child: TextFormField(
                      controller: minHumidityController,
                      style: const TextStyle(color: Color(0xFFb2c5b2)),
                      decoration: const InputDecoration(
                        label: Text(
                          'Min Humidity',
                          style: TextStyle(
                            color: Color(0xFFb2c5b2),
                          ),
                        ),
                        enabledBorder: OutlineInputBorder(
```

```
                            borderRadius:
BorderRadius.all(Radius.circular(5)),
                            borderSide: BorderSide(color: Colors.white)),
                        focusedBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.white,
                            width: 2,
                          ),
                          borderRadius: BorderRadius.all(Radius.circular(10)),
                        ),
                        errorBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.red,
                            width: 2,
                          ),
                          borderRadius: BorderRadius.all(Radius.circular(10)),
                        ),),),
                    ),
                  ],
                ),
              ),
              Padding(
                padding: const EdgeInsets.only(top: 7.0),
                child: Row(
                  children: [
                    Expanded(
                      child: TextFormField(
                        controller: maxPPMController,
                        style: const TextStyle(color: Color(0xFFb2c5b2)),
                        decoration: const InputDecoration(
                          label: Text(
                            'Max PPM',
                            style: TextStyle(
                              color: Color(0xFFb2c5b2),
                            ),
                          ),
                          enabledBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.all(Radius.circular(5)),
                              borderSide: BorderSide(color: Colors.white)),
                          focusedBorder: OutlineInputBorder(
                            borderSide: BorderSide(
                              color: Colors.white,
                              width: 2,
                            ),
                            borderRadius: BorderRadius.all(Radius.circular(10)),
```

```
                    ),
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.red,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                  ),
                ),
              ),
              Expanded(
                child: TextFormField(
                  controller: minPPMController,
                  style: const TextStyle(color: Color(0xFFb2c5b2)),
                  decoration: const InputDecoration(
                    label: Text(
                      'Min PPM',
                      style: TextStyle(
                        color: Color(0xFFb2c5b2),
                      ),
                    ),
                    enabledBorder: OutlineInputBorder(
                        borderRadius: BorderRadius.all(Radius.circular(5)),
                        borderSide: BorderSide(color: Colors.white)),
                    focusedBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.white,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                    errorBorder: OutlineInputBorder(
                      borderSide: BorderSide(
                        color: Colors.red,
                        width: 2,
                      ),
                      borderRadius: BorderRadius.all(Radius.circular(10)),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
        Padding(
```

```
              padding: const EdgeInsets.only(top: 7.0),
              child: Row(
                children: [
                  Expanded(
                    child: TextFormField(
                      controller: maxPHController,
                      style: const TextStyle(color: Color(0xFFb2c5b2)),
                      decoration: const InputDecoration(
                        label: Text(
                          'Max pH',
                          style: TextStyle(
                            color: Color(0xFFb2c5b2),
                          ),
                        ),
                        enabledBorder: OutlineInputBorder(
                            borderRadius:
BorderRadius.all(Radius.circular(5)),
                            borderSide: BorderSide(color: Colors.white)),
                        focusedBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.white,
                            width: 2,
                          ),
                          borderRadius: BorderRadius.all(Radius.circular(10)),
                        ),
                        errorBorder: OutlineInputBorder(
                          borderSide: BorderSide(
                            color: Colors.red,
                            width: 2,
                          ),
                          borderRadius: BorderRadius.all(Radius.circular(10)),
                        ),
                      ),
                    ),),
                  Expanded(
                    child: TextFormField(
                      controller: minPHController,
                      style: const TextStyle(color: Color(0xFFb2c5b2)),
                      decoration: const InputDecoration(
                        label: Text(
                          'Min pH',
                          style: TextStyle(
                            color: Color(0xFFb2c5b2),
                          ),
                        ),
                        enabledBorder: OutlineInputBorder(
```

```
                      borderRadius: BorderRadius.all(Radius.circular(5)),
                      borderSide: BorderSide(color: Colors.white)),
                  focusedBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.white,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                  errorBorder: OutlineInputBorder(
                    borderSide: BorderSide(
                      color: Colors.red,
                      width: 2,
                    ),
                    borderRadius: BorderRadius.all(Radius.circular(10)),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(top: 7.0),
        child: TextFormField(
          controller: ultrasonicHourController,
          style: const TextStyle(color: Color(0xFFb2c5b2)),
          decoration: const InputDecoration(
            label: Text(
              'Jam Ultrasonic',
              style: TextStyle(
                color: Color(0xFFb2c5b2),
              ),
            ),
            enabledBorder: OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(5)),
                borderSide: BorderSide(color: Colors.white)),
            focusedBorder: OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.white,
                width: 2,
              ),
              borderRadius: BorderRadius.all(Radius.circular(10)),
            ),
            errorBorder: OutlineInputBorder(
              borderSide: BorderSide(
```

```
                    color: Colors.red,
                    width: 2,
                  ),
                  borderRadius: BorderRadius.all(Radius.circular(10)),
                ),
              ),
            ),
          ),
          Padding(
            padding: const EdgeInsets.only(top: 10.0),
            child: Row(
              children: [
                Expanded(
                  child: Padding(
                    padding: const EdgeInsets.only(right: 2.0),
                    child: SizedBox(
                      height: 60,
                      child: ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10),
                            ),
                          ),
                          onPressed: () {
                            _showEditDialog(context, "fan", userSettingId);
                          },
                          child: const Text(
                            "Tambah Jadwal Kipas",
                            style: TextStyle(
                              color: Color(0xFFb2c5b2),
                            ),
                          )),
                    ),
                  ),
                ),
                Expanded(
                  child: Padding(
                    padding: const EdgeInsets.only(left: 2.0),
                    child: SizedBox(
                      height: 60,
                      child: ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10),
                            ),
                          ),
```

```
                            onPressed: () {
                              _showEditDialog(context, "LED",
userSettingId);
                            },
                            child: const Text(
                              "Tambah Jadwal LED",
                              style: TextStyle(
                                color: Color(0xFFb2c5b2),
                              ),
                            )),
                        ),
                      ),
                    ),
                  ],),),),
              SizedBox(
                height: 60,
                child: Padding(
                  padding: const EdgeInsets.only(top: 7.0),
                  child: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                    ),
                    onPressed: () {
                      _showEditDialog(context, "watering", userSettingId);
                    },
                    child: const Text(
                      "Tambah Jadwal Penyiraman",
                      style: TextStyle(
                        color: Color(0xFFb2c5b2),
                      ),
                    ),
                  ),
                ),
              ),
              countJadwal > 0
                  ? Padding(
                      padding: const EdgeInsets.symmetric(
                          vertical: 5.0, horizontal: 2.0),
                      child: Center(
                          child:
                              ScheduleCard(scheduleData:
jadwal['scheduleData'])),
                    )
                  : const Text('No Schedule Data'),
```

```
            ],
          ),
        ),
      );
    }
}
```

Segmen Program 4.91 User setting detail

Pada segmen program 4.91 adalah implementasi dan logika yang di gunakan untuk membuat halaman user setting detail dimana pengguna dapat melihat informasi mengenai user setting tertenu dan mengubah data user setting dan setting yang berkaitan.

```
import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:microgreen_monitoring_app/util/formatingTime.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class ScheduleCard extends StatefulWidget {
  final Map<String, dynamic> scheduleData;
  ScheduleCard({required this.scheduleData});

  @override
  State<ScheduleCard> createState() => _ScheduleCardState();
}

class _ScheduleCardState extends State<ScheduleCard> {
  Future<void> deleteSchedule(int scheduleId, String type) async {
    String apiUrl =
        '${globalVar.ApiUrlBase}/user-setting-detail/delete-schedule';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');
```

```
      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(
              <String, dynamic>{'scheduleId': scheduleId, "type": type}));

      if (response.statusCode == 200) {
        print("success delete.");
      } else {
        print('Request failed with status code: ${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
      print('Error: $error');
    }
  }

  @override
  Widget build(BuildContext context) {
    List<dynamic> fanScheduleData =
widget.scheduleData['fanScheduleData'];
    List<dynamic> ledScheduleData =
widget.scheduleData['ledScheduleData'];
    List<dynamic> wateringScheduleData =
        widget.scheduleData['wateringScheduleData'];

    return ListView(
        physics: const NeverScrollableScrollPhysics(),
        shrinkWrap: true,
        children: [
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              _buildScheduleList("Fan", 'Fan Schedule', fanScheduleData),
              _buildScheduleList("LED", 'LED Schedule', ledScheduleData),
              _buildScheduleList(
                  "Watering", 'Watering Schedule', wateringScheduleData),
            ],
          ),
```

```
      ]);}
 Widget _buildScheduleList(
     String type, String title, List<dynamic> scheduleList) {
   return Padding(
     padding: const EdgeInsets.all(8.0),
     child: Card(
       color: const Color(0xffd5dddf),
       child: Padding(
         padding: const EdgeInsets.all(8.0),
         child: Column(
           crossAxisAlignment: CrossAxisAlignment.start,
           children: [
             Text(
               title,
               style:
                   const TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
             ),
             const SizedBox(height: 8),
             ListView.builder(
               physics: const NeverScrollableScrollPhysics(),
               shrinkWrap: true,
               itemCount: scheduleList.length,
               itemBuilder: (context, index) {
                 Map<String, dynamic> schedule = scheduleList[index];
                 return Card(
                   child: Padding(
                     padding: const EdgeInsets.all(8.0),
                     child: Row(
                       children: [
                         Expanded(
                           child: Column(
                             crossAxisAlignment: CrossAxisAlignment.start,
                             children: [
                               Text(
                                 type == "Fan"
                                     ? 'Fan Menyala Saat:
${formatTime(schedule['fan_on_time'])}'
                                     : type == "LED"
                                         ? 'LED Menyala Saat:
${formatTime(schedule['light_on_time'])}'
                                         : 'Menyiram saat:
${formatTime(schedule['watering_time'])}',
                                 style: const TextStyle(
                                   color: Color(0xFFb2c5b2),
                                 ),
                               ),
                               Text(
                                 type == "Fan"
```

```
                                        ? 'Fan Mati Saat:
 ${formatTime(schedule['fan_off_time'])}'
                                        : type == "LED"
                                            ? 'LED Mati Saat:
 ${formatTime(schedule['light_off_time'])}'
                                            : 'Menyiram sampai:
 ${formatTime(schedule['watering_for'])}',
                                    style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                    ),
                                  ),
                                  Text(
                                    'Active Status:
${schedule['active_status']}',
                                    style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                    ),
                                  ),
                                ],
                              ),
                            ),
                            Expanded(
                              child: SizedBox(
                                height: 75,
                                child: Padding(
                                  padding: const EdgeInsets.only(left: 10.0),
                                  child: ElevatedButton(
                                    style: ElevatedButton.styleFrom(
                                      shape: RoundedRectangleBorder(
                                        borderRadius: BorderRadius.circular(10),
                                      ),
                                    ),
                                    onPressed: () async {
                                      await deleteSchedule(schedule['id'],
type);

                                      setState(() {});
                                    },
                                    child: const Text(
                                      "Delete",
                                      style: TextStyle(
                                        color: Color(0xFFb2c5b2),
                                      ),
                                    ),
                                  ),
                                ),
                              ),)],),),);
                  },
```

```
                ),
              ],
            ),
          ),
        ),
      );
    }
}
```

Segmen Program 4.92 Schedule card componen

Pada segmen program 4.92 adalah implementasi komponent UI untuk menampilkan jadwal user setting kipas, lampu, dan penyiraman.

4.3.6 Implementasi Grow Cycle Page

Pada bagian ini akan di bahas mengenai implementasi dan logika yang di gunakan untuk membuat halaman grow cycle.

```
import 'dart:async';
import 'dart:convert';

import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import 'package:jwt_decoder/jwt_decoder.dart';
import 'package:microgreen_monitoring_app/home/farm/grow_cycle/HeightCard.dart';
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/completeGrowCycle.dart';
import 'package:microgreen_monitoring_app/home/farm/grow_cycle/HumidityChart.dart';
import 'package:microgreen_monitoring_app/home/farm/grow_cycle/TemperatureChart.dart';
import 'package:microgreen_monitoring_app/util/FarmData.dart';
import 'package:microgreen_monitoring_app/util/GrowCycleData.dart';
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:microgreen_monitoring_app/util/chartFunctionTools.dart';
import 'package:microgreen_monitoring_app/util/formatingTime.dart';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:socket_io_client/socket_io_client.dart' as IO;
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

Widget bottomTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold, fontSize: 13);
```

```
  String text = doubleToDate(value);
  return SideTitleWidget(
    axisSide: meta.axisSide,
    space: 4,
    child: Text(text, style: style),
  );
}

Widget leftTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold, fontSize: 13);

  return SideTitleWidget(
    axisSide: meta.axisSide,
    child: Text(
      '${value.toInt()} %',
      style: style,
    ),
  );
}

class GrowCycleDetail extends StatefulWidget {
  const GrowCycleDetail({super.key});
  @override
  State<GrowCycleDetail> createState() => _GrowCycleDetailState();
}

class _GrowCycleDetailState extends State<GrowCycleDetail> {
  String startDate = "None";
  String finishDate = "-";
  late IO.Socket _socket;
  late Timer _timer;
  bool _disposed = false;

  TextEditingController harvestController = TextEditingController();

  List<Humidity> humidityData = [];
  List<FlSpot> humiditySpots = [];
  List<Temperature> temperatureData = [];
  List<FlSpot> temperatureSpots = [];
  List<PPM> ppmData = [];
  List<FlSpot> ppmSpots = [];
  List<PH> phData = [];
  List<FlSpot> phSpots = [];
  List<Height> plantHeightData = [];
  List<Height> last12plantHeightData = [];
```

```
  List<FlSpot> plantHeightSpots = [];

  String humidityNow = "0";
  String temperatureNow = "0";
  String nutrientNow = "0";
  String pHNow = "0";

  bool fanOn = false;
  bool ledOn = false;
  bool wateringOn = false;
  bool humidityOn = false;

  bool manualDisconnect = false;

  late FarmData farmData;
  late Map<String, dynamic> growCycleData;

  _connectSocket() {
    _socket.onConnect(
        (data) => notificationGreen(context, "Connection established"));
    _socket.onConnectError(
        (error) => notification(context, "Connection error: $error"));
    _socket.onDisconnect(
        (data) => notificationGreen(context, "Socket IO server disconnect"));
    _socket.on('roomMessage', (data) {
      switch (data['hardware']) {
        case "lamp":
          setState(() {
            if (data["status"] == 1) {
              ledOn = true;
              print("lamp is on");
            } else {
              ledOn = false;
              print("lamp is off");
            }
          });
          print(
              'Received hardware: ${data["hardware"]}, with status ${data["status"]
== 1 ? 'on' : 'off'}');
          print('Received status: ${data["status"]}');
          print("status: ${data['status'] == true ? "True" : "False"}");
          break;
        case "WPump":
          setState(() {
            if (data["status"] == 1) {
```

```
              wateringOn = true;
            } else {
              wateringOn = false;
            }
          });
          print(
              'Received hardware: ${data["hardware"]}, with status ${data["status"]
== 1 ? 'on' : 'off'}');
          print('Received status: ${data["status"]}');
          print("status: ${data['status'] == true ? "True" : "False"}");
          break;
        case "HumidifierFan":
          setState(() {
            if (data["status"] == 1) {
              humidityOn = true;
            } else {
              humidityOn = false;
            }
          });
          print(
              'Received hardware: ${data["hardware"]}, with status ${data["status"]
== 1 ? 'on' : 'off'}');
          print('Received status: ${data["status"]}');
          print("status: ${data['status'] == true ? "True" : "False"}");
          break;
        case "Fans":
          setState(() {
            if (data["status"] == 1) {
              fanOn = true;
            } else {
              fanOn = false;
            }
          });
          print(
              'Received hardware: ${data["hardware"]}, with status ${data["status"]
== 1 ? 'on' : 'off'}');
          print('Received status: ${data["status"]}');
          print("status: ${data['status'] == true ? "True" : "False"}");
          break;
        default:
      }
    });
  }
```

```
  _joinRoom() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String? token = prefs.getString('token');

    if (token == null || token.isEmpty) {
      // Handle the case when token is not available
      print('Token not found');
      return;
    }

    Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
    int userId = decodedToken['userId'];

    _socket
        .emit('joinRoom', {"accountID": userId, "roomID":
farmData.farmRoomId});
  }

  _toggleRelay(int pin, bool turnOn, String run) async {
    final data = {
      'room': farmData.farmRoomId,
      'turnOn': turnOn,
      'data': {
        "type": "control",
        "hardware": "relay",
        "pin": pin,
        "run": run
      },
    };
    _socket.emit('toggleRelay', data);
  }

  void fetchGrowCycleEnvironmentData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/grow-cycle-data';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
```

```
      headers: <String, String>{
        'Authorization': 'Bearer $token',
        'Content-Type': 'application/json',
      },
      body: jsonEncode(
          <String, dynamic>{'growCycleId': growCycleData['id']}));

  if (response.statusCode == 200) {
    final Map<String, dynamic> responseData = jsonDecode(response.body);
    setState(() {
      CombinedResponse data = CombinedResponse.fromJson(responseData);

      if (data.humidity.isNotEmpty) {
        humidityData = data.humidity;
        humiditySpots = convertHumidityToFlSpots(humidityData);
        humidityNow =
            data.humidity[data.humidity.length - 1].humidity.toString();
      }
      if (data.temperature.isNotEmpty) {
        temperatureData = data.temperature;
        temperatureSpots = convertTemperatureToFlSpots(temperatureData);
        temperatureNow = data
            .temperature[data.temperature.length - 1].temperature
            .toString();
      }
      if (data.ppm.isNotEmpty) {
        ppmData = data.ppm;
        nutrientNow = data.ppm[data.ppm.length - 1].ppm.toString();
      }
      if (data.ph.isNotEmpty) {
        phData = data.ph;
        double tempPH = 0;
        for (var element in phData) {
          tempPH += element.ph;
        }
        pHNow = (tempPH / phData.length).toString();
      }
      if (data.height.isNotEmpty) {
        plantHeightData = data.height;
        last12plantHeightData = plantHeightData.sublist(
            plantHeightData.length - 12, plantHeightData.length);
      }
    });
  } else {
```

```
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
    }
  }

  void updateHarvestData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/add-harvest-grow-cycle';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, dynamic>{
            'growCycleId': growCycleData['id'],
            "harvest": harvestController.text
          }));

      if (response.statusCode == 200) {
        setState(() {});
      } else {
        print('Request failed with status code:
${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
    }
  }

  double createInterval() {
```

```
    try {
      return humidityData[0].time.millisecondsSinceEpoch.toDouble() -
          humidityData[humidityData.length]
              .time
              .millisecondsSinceEpoch
              .toDouble();
    } catch (error) {
      return 0.0;
    }
  }

  @override
  void initState() {
    super.initState();
    manualDisconnect = false;
    _timer = Timer.periodic(const Duration(seconds: 10), (timer) {
      print("function is going");
      if (!_disposed) {
        fetchGrowCycleEnvironmentData();
      }
    });
  }

  @override
  void dispose() {
    super.dispose();
    _disposed = true;
    _timer.cancel();
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    final args = ModalRoute.of(context)!.settings.arguments as Set<dynamic>;
    growCycleData = args.elementAt(0);
    farmData = args.elementAt(1) as FarmData;
    if (growCycleData['total_harvest'] != null) {
      harvestController.text = growCycleData['total_harvest'].toString();
    }
    startDate = formatTimeFull(growCycleData["start_date"]);
    finishDate = growCycleData["finish_date"] != null
        ? formatTimeFull(growCycleData["finish_date"])
        : "-";
    _socket = IO.io(
        globalVar.ApiUrlBase,
```

```
      IO.OptionBuilder()
          .setTransports(['websocket'])
          .enableForceNew()
          .setQuery({"farmRoomId": farmData.farmRoomId})
          .build());
  if (manualDisconnect == false) {
    _connectSocket();
    _joinRoom();
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text(
        "Grow Cycle Detail",
        style: TextStyle(
          color: Color(0xFFd5dddf),
        ),
      ),
      leading: GestureDetector(
        child: const Icon(
          Icons.arrow_back_ios,
        ),
        onTap: () {
          _socket.disconnect();
          manualDisconnect = true;
          Navigator.pushReplacementNamed(context, '/farm-detail',
              arguments: farmData);
        },
      ),
    ),
    body: SingleChildScrollView(
      child: Column(
        children: [
          Row(
            children: [
              Expanded(
                child: Column(
                  children: [
                    const Text(
                      "Start Date",
                      style: TextStyle(
                          color: Color(0xFFb2c5b2),
                          fontWeight: FontWeight.bold,
```

```
                        fontSize: 20),
                  ),
                  Text(
                    startDate,
                    style: const TextStyle(
                        color: Color(0xFFb2c5b2),
                        fontWeight: FontWeight.bold,
                        fontSize: 20),
                  )
                ],
              ),
            ),
            Expanded(
              child: Column(
                children: [
                  const Text(
                    "Finish Date",
                    style: TextStyle(
                        color: Color(0xFFb2c5b2),
                        fontWeight: FontWeight.bold,
                        fontSize: 20),
                  ),
                  Text(
                    finishDate,
                    style: const TextStyle(
                        color: Color(0xFFb2c5b2),
                        fontWeight: FontWeight.bold,
                        fontSize: 20),
                  )
                ],
              ),
            )
          ],
        ),
        Row(
          children: [
            Expanded(
              child: SizedBox(
                height: 100,
                child: Card(
                  child: Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Column(
                      children: [
                        const Expanded(
                          child: Text(
```

```
                            "Humidity now",
                            style: TextStyle(
                                color: Color(0xFFb2c5b2),
                                fontWeight: FontWeight.bold,
                                fontSize: 13),
                          ),
                        ),
                        Expanded(child: Text(humidityNow))
                      ],
                    ),
                  ),
                ),
              ),
            ),
            Expanded(
              child: SizedBox(
                height: 100,
                child: Card(
                  child: Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: Column(
                      children: [
                        const Expanded(
                          child: Text(
                            "Temperature now",
                            style: TextStyle(
                                color: Color(0xFFb2c5b2),
                                fontWeight: FontWeight.bold,
                                fontSize: 13),
                          ),
                        ),
                        Expanded(child: Text(temperatureNow))
                      ],
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
        Row(
          children: [
            Expanded(
              child: SizedBox(
                height: 100,
                child: Card(
```

```
                    child: Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: Column(
                        children: [
                          const Expanded(
                            child: Text(
                              "Nutrient now",
                              style: TextStyle(
                                  color: Color(0xFFb2c5b2),
                                  fontWeight: FontWeight.bold,
                                  fontSize: 13),
                            ),
                          ),
                          Expanded(child: Text(nutrientNow))
                        ],
                      ),
                    ),
                  ),
                ),
                Expanded(
                  child: SizedBox(
                    height: 100,
                    child: Card(
                      child: Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Column(
                          children: [
                            const Expanded(
                              child: Text(
                                "pH now",
                                style: TextStyle(
                                    color: Color(0xFFb2c5b2),
                                    fontWeight: FontWeight.bold,
                                    fontSize: 13),
                              ),
                            ),
                            Expanded(child: Text(pHNow))
                          ],
                        ),
                      ),
                    ),
                  ),
                )
              ],
            ),
```

```
            Row(
              children: [
                Expanded(
                  child: SizedBox(
                    height: 100,
                    child: Card(
                      child: Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Column(
                          children: [
                            Expanded(
                              child: Text(
                                "LED: ${ledOn == true ? "ON" : "OFF"}",
                                style: const TextStyle(
                                    color: Color(0xFFb2c5b2),
                                    fontWeight: FontWeight.bold,
                                    fontSize: 13),
                              ),
                            ),
                            Expanded(
                                child: TextButton(
                              onPressed: () {
                                if (ledOn == true) {
                                  print("try turning off");
                                  _toggleRelay(
                                      0, false, "off");
                                } else if (ledOn == false) {
                                  print("try turning on");
                                  _toggleRelay(0, true, "on");
                                }
                              },
                              child: Text(
                                  ledOn == false ? "Turn ON" : "Turn OFF",
                                  style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                      fontWeight: FontWeight.bold,
                                      fontSize: 15)),
                            ))
                          ],
                        ),
                      ),
                    ),
                  ),
                ),
                Expanded(
                  child: SizedBox(
```

```
                    height: 100,
                    child: Card(
                      child: Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Column(
                          children: [
                            Expanded(
                              child: Text(
                                "Watering: ${wateringOn == true ? "ON" :
"OFF"}",
                                style: const TextStyle(
                                    color: Color(0xFFb2c5b2),
                                    fontWeight: FontWeight.bold,
                                    fontSize: 13),
                              ),
                            ),
                            Expanded(
                                child: TextButton(
                              onPressed: () {
                                if (wateringOn == true) {
                                  _toggleRelay(1, false, "off");
                                } else if (wateringOn == false) {
                                  _toggleRelay(1, true, "on");
                                }
                              },
                              child: Text(
                                  wateringOn == false ? "Turn ON" : "Turn OFF",
                                  style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                      fontWeight: FontWeight.bold,
                                      fontSize: 15)),
                            ))
                          ],
                        ),
                      ),
                    ),
                  )
                ],),
              Row(
                children: [
                  Expanded(
                    child: SizedBox(
                      height: 100,
                      child: Card(
```

```
                child: Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Column(
                    children: [
                      Expanded(
                        child: Text(
                          "Fan: ${fanOn == true ? "ON" : "OFF"}",
                          style: const TextStyle(
                              color: Color(0xFFb2c5b2),
                              fontWeight: FontWeight.bold,
                              fontSize: 13),
                        ),
                      ),
                      Expanded(
                          child: TextButton(
                        onPressed: () {
                          if (fanOn == true) {
                            _toggleRelay(
                                4, false, "off");
                          } else if (fanOn == false) {
                            _toggleRelay(4, true, "on");
                          }
                        },
                        child: Text(
                            fanOn == false ? "Turn ON" : "Turn OFF",
                            style: const TextStyle(
                                color: Color(0xFFb2c5b2),
                                fontWeight: FontWeight.bold,
                                fontSize: 15)),
                      ))
                    ],
                  ),
                ),
              ),
            ),
          ),
        ),
        Expanded(
          child: SizedBox(
            height: 100,
            child: Card(
              child: Padding(
                padding: const EdgeInsets.all(8.0),
                child: Column(
                  children: [
                    Expanded(
                      child: Text(
```

```
                                "Humidifier: ${humidityOn == true ? "ON" :
"OFF"}",
                                style: const TextStyle(
                                    color: Color(0xFFb2c5b2),
                                    fontWeight: FontWeight.bold,
                                    fontSize: 13),
                              ),
                            ),
                            Expanded(
                                child: TextButton(
                              onPressed: () {
                                if (humidityOn == true) {
                                  _toggleRelay(
                                      3, false, "off"); // if on turn off
                                } else if (humidityOn == false) {
                                  _toggleRelay(3, true, "on");
                                }
                              },
                              child: Text(
                                  humidityOn == false ? "Turn ON" : "Turn
OFF",
                                  style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                      fontWeight: FontWeight.bold,
                                      fontSize: 15)),
                            ))
                          ],
                        ),
                      ),
                    ),),)
              ],
            ),
            HumidityChart(spots: humiditySpots, humidityData: humidityData),
            TemperatureChart(
                spots: temperatureSpots, temperatureData: temperatureData),
            GridView.count(
              physics: const NeverScrollableScrollPhysics(),
              padding: const EdgeInsets.all(10),
              shrinkWrap: true,
              primary: true,
              crossAxisSpacing: 3,
              mainAxisSpacing: 3,
              crossAxisCount: 3,
              children: List.generate(
                12,
                (index) {
```

```
                return HeightCard(
                    data: last12plantHeightData.isNotEmpty
                        ? last12plantHeightData[index]
                        : "none");
              },
            ),
          ),
          growCycleData['active_status'] == false
              ? Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: SizedBox(
                    height: 140,
                    child: Card(
                      child: Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Column(
                          children: [
                            TextFormField(
                              controller: harvestController,
                              style:
                                  const TextStyle(color: Color(0xFFb2c5b2)),
                              decoration: const InputDecoration(
                                label: Text(
                                  "Hasil Panen (Gram)",
                                  style: TextStyle(
                                    color: Color(0xFFb2c5b2),
                                  ),
                                ),
                                enabledBorder: OutlineInputBorder(
                                    borderRadius:
                                        BorderRadius.all(Radius.circular(5)),
                                    borderSide:
                                        BorderSide(color: Colors.white)),
                                focusedBorder: OutlineInputBorder(
                                  borderSide: BorderSide(
                                    color: Colors.white,
                                    width: 2,
                                  ),
                                  borderRadius:
                                      BorderRadius.all(Radius.circular(10)),
                                ),
                                errorBorder: OutlineInputBorder(
                                  borderSide: BorderSide(
                                    color: Colors.red,
                                    width: 2,
                                  ),
```

```
                        borderRadius:
                            BorderRadius.all(Radius.circular(10)),
                      ),
                    ),
                  ),
                  Padding(
                    padding: const EdgeInsets.only(top: 4.0),
                    child: Row(
                      children: [
                        Expanded(
                          child: ElevatedButton(
                              style: ElevatedButton.styleFrom(
                                backgroundColor:
                                    const Color(0xFF3c5148),
                                shape: RoundedRectangleBorder(
                                  borderRadius:
                                      BorderRadius.circular(10),
                                ),
                              ),
                              onPressed: () {
                                updateHarvestData();
                              },
                              child: const Text(
                                "Update panen",
                                style: TextStyle(
                                  color: Color(0xFFb2c5b2),
                                ),
                              )),
                        ),
                      ],
                    ),
                  )
                ],
              ),
            ),
          ),
        ),
      )
    : Row(
        children: [
          Expanded(
            child: Padding(
              padding: const EdgeInsets.all(8.0),
              child: ElevatedButton(
                style: ElevatedButton.styleFrom(
                  backgroundColor: const Color(0xFF3c5148),
```

```
                                shape: RoundedRectangleBorder(
                                  borderRadius: BorderRadius.circular(10),
                                ),
                              ),
                              onPressed: () {
                                if (growCycleData['active_status'] == false) {
                                  finishGrowCycle(context, growCycleData['id']);
                                  setState(() {});
                                }
                              },
                              child: Text(
                                  growCycleData['active_status'] == true
                                      ? "Selesaikan"
                                      : "Sudah selesai",
                                  style: const TextStyle(
                                      color: Color(0xFFb2c5b2),
                                      fontWeight: FontWeight.bold,
                                      fontSize: 15)),
                            ),
                          ),
                        )
                      ],
                    ),
                ],
              ),
            ),
        );
    }
}
```

Segmen Program 4.93 Grow cycle detail

Pada segmen program 4.93 adalah implementasi UI dan logika untuk membuat halaman grow cycle detail.

```
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import 'package:microgreen_monitoring_app/util/GrowCycleData.dart';
import 'package:microgreen_monitoring_app/util/chartFunctionTools.dart';

class HumidityChart extends StatefulWidget {
  final List<FlSpot> spots;
  final List<Humidity> humidityData;
  const HumidityChart({
    super.key,
```

```
    required this.spots,
    required this.humidityData,
  });

  @override
  State<HumidityChart> createState() => _HumidityChartState();
}

class _HumidityChartState extends State<HumidityChart> {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 10, bottom: 10),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Center(
            child: Padding(
              padding: EdgeInsets.only(bottom: 8.0),
              child: Text(
                "Humidity data",
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                  fontWeight: FontWeight.bold,
                  fontSize: 15,
                ),
              ),
            ),
          ),
          SizedBox(
            width: 400,
            height: 200,
            child: LineChart(
              LineChartData(
                lineBarsData: [
                  LineChartBarData(
                    show: true,
                    spots: widget.spots,
                    belowBarData: BarAreaData(show: false),
                    aboveBarData: BarAreaData(
                      show: true,
                      color: Colors.transparent,
                      cutOffY: 100,
                      applyCutOffY: true,
                    ),
                  ),
```

```
                ],
                minY: 0,
                maxY: 100,
                titlesData: FlTitlesData(
                  bottomTitles: AxisTitles(
                    sideTitles: SideTitles(
                      showTitles: true,
                      interval:
                          createIntervalHumidity(widget.humidityData) != 0.0
                              ? createIntervalHumidity(widget.humidityData)
                              : 1,
                      getTitlesWidget: bottomTitleWidgets,
                    ),
                  ),
                  leftTitles: const AxisTitles(
                    sideTitles: SideTitles(
                      showTitles: true,
                      getTitlesWidget: leftTitleWidgets,
                      interval: 20,
                      reservedSize: 50,
                    ),
                  ),
                  topTitles: const AxisTitles(
                    sideTitles: SideTitles(showTitles: false),
                  ),
                  rightTitles: const AxisTitles(
                    sideTitles: SideTitles(showTitles: false),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Segmen Program 4.94 Humidity chart

Pada segmen program 4.94 adalah implementasi komponen UI dari chart kelembaban dimana komponen ini akan muncul saat pengguna membuka halaman grow cycle detail.

```
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import 'package:microgreen_monitoring_app/util/GrowCycleData.dart';
import
'package:microgreen_monitoring_app/util/chartFunctionTools.dart';

class TemperatureChart extends StatefulWidget {
  final List<FlSpot> spots;
  final List<Temperature> temperatureData;

  const TemperatureChart({
    super.key,
    required this.spots,
    required this.temperatureData,
  });

  @override
  _TemperatureChartState createState() => _TemperatureChartState();
}
class _TemperatureChartState extends State<TemperatureChart> {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 10, bottom: 10),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          const Center(
            child: Padding(
              padding: EdgeInsets.only(bottom: 8.0),
              child: Text(
                "Temperature data",
                style: TextStyle(
                  color: Color(0xFFb2c5b2),
                  fontWeight: FontWeight.bold,
                  fontSize: 15,
                ),
              ),
            ),
          ),
          SizedBox(
            width: 400,
            height: 200,
            child: LineChart(
              LineChartData(
                lineBarsData: [
```

```
                  LineChartBarData(
                    show: true,
                    spots: widget.spots,
                  ),
                ],
                minY: 10,
                maxY: 40,
                titlesData: FlTitlesData(
                  bottomTitles: AxisTitles(
                    sideTitles: SideTitles(
                      showTitles: true,
                      interval: createIntervalTemperature(
                                  widget.temperatureData) !=
                              0.0
? createIntervalTemperature(widget.temperatureData)
                          : 1,
                      getTitlesWidget: bottomTitleWidgets,
                    ),
                  ),
                  leftTitles: const AxisTitles(
                    sideTitles: SideTitles(
                      showTitles: true,
                      getTitlesWidget: leftTitleWidgets,
                      interval: 5,
                      reservedSize: 50,
                    ),
                  ),
                  topTitles: const AxisTitles(
                    sideTitles: SideTitles(showTitles: false),
                  ),
                  rightTitles: const AxisTitles(
                    sideTitles: SideTitles(showTitles: false),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Segmen Program 4.95 temperatur chart

Pada sefmen program 4.95 adalah implementasi komponen UI untuk chart temperature dimana komponen ini akan muncul saat pengguna membuka halaman grow cycle detail.

```
import 'package:flutter/material.dart';

class HeightCard extends StatelessWidget {
  final data;
  const HeightCard({super.key, required this.data});

  double calculateHeight(double heightData) {
    double answer = 17 - heightData;
    if (answer < 0) {
      return 0;
    } else {
      return answer;
    }
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          ListTile(
            title: Text(data != "none" ? "Tray ${data.index + 1}" : "No
data"),
            titleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 20),
            subtitle: Row(
              children: [
                Expanded(
                  child: Text(
                    'High: ${data != "none" ? calculateHeight(data.high) :
"none"} cm',
                  ),
                ),
              ],),
            subtitleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 15),
          ),
        ],
```

```
      ),
    );
  }
}
```

Segmen Program 4.96 Height card

Pada segmen program 4.96 adalah implementasi komponen UI height card dimana komponen ini akan muncul saat pengguna membuka halaman grow cycle detail.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

void finishGrowCycle(BuildContext context, int growCycleId) async {
  String apiUrl = '${globalVar.ApiUrlBase}/finish-grow-cycle';
  try {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String? token = prefs.getString('token');
    if (token == null || token.isEmpty) {
      print('Token not found');
      return;
    }
    final response = await http.post(Uri.parse(apiUrl),
        headers: <String, String>{
          'Authorization': 'Bearer $token',
          'Content-Type': 'application/json',
        },
        body: jsonEncode(<String, dynamic>{'growCycleId':
growCycleId}));

    if (response.statusCode == 200) {
      // final Map<String, dynamic> responseData =
jsonDecode(response.body);
    } else {
      print('Request failed with status code:
${response.statusCode}');
```

```
    }
  } catch (error) {
    notification(context, error);
  }
}
```

Segmen Program 4.97 Complete grow cycle

Pada segmen program 4.97 adalah implementasi logika untuk mengubah status suati grow cycle menjadi selesai pada suatu grow cycle.

4.3.7 Implementasi Satatistic Page

Pada bagian ini akan membahas mengenai implementasi dan logika yang di gunakan untuk membuat halaman statistic.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:jwt_decoder/jwt_decoder.dart';
import
'package:microgreen_monitoring_app/home/statistic/GrowCycleCardStatistic.d
art';
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;

class StatisticPage extends StatefulWidget {
  const StatisticPage({super.key});

  @override
  State<StatisticPage> createState() => _StatisticPageState();
}

class _StatisticPageState extends State<StatisticPage> {
  int countGrowCycle = 0;
  int farmId = 0;
  List<dynamic> growCycle = [];
  Future<void> fetchGrowCycleData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/all-user-grow-cycle-data';
    try {
```

```
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      Map<String, dynamic> decodedToken = JwtDecoder.decode(token);
      print(decodedToken);
      int userId = decodedToken['userId'];

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(<String, dynamic>{'userId': userId}));

      if (response.statusCode == 200) {
        final responseData = jsonDecode(response.body);
        setState(() {
          growCycle = responseData;
          countGrowCycle = growCycle.length;
        });
      } else {
        print('Request failed with status code: ${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
      print('Error: $error');
    }
  }

  @override
  void initState() {
    super.initState();
    fetchGrowCycleData();
  }

  @override
  Widget build(BuildContext context) {
```

```
    for (int i = 0; i < growCycle.length; i++) {
      if (growCycle[i]['active_status'] == true ||
          growCycle[i]['finish_date'] == null) {}
    }
    return Scaffold(
      appBar: AppBar(
        title:
            const Text("Statistic", style: TextStyle(color:
Color(0xFFd5dddf))),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/home');
          },
        ),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            ListView.builder(
              shrinkWrap: true,
              physics: const NeverScrollableScrollPhysics(),
              padding: const EdgeInsets.all(8),
              itemCount: countGrowCycle,
              itemBuilder: (context, index) {
                return GrowCycleCardStatistic(
                  data: growCycle[index],
                );
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

Segmen Program 4.98 Statistic page

Pada segmen program 4.98 adalah implementasi dan logika yang di buat untuk halaman statistic dimana pengguna dapat melihat semua grow cycle yang suatu pengguna punya.

```
import 'dart:convert';
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/HeightCard.dart';
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/completeGrowCycle.dart'
;
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/HumidityChart.dart';
import
'package:microgreen_monitoring_app/home/farm/grow_cycle/TemperatureChart.dart';
import 'package:microgreen_monitoring_app/util/GrowCycleData.dart';
import 'package:microgreen_monitoring_app/util/ScaffoldNotification.dart';
import 'package:microgreen_monitoring_app/util/chartFunctionTools.dart';
import 'package:microgreen_monitoring_app/util/formatingTime.dart';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:microgreen_monitoring_app/util/globalVariable.dart'
    as globalVar;
Widget bottomTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold, fontSize: 13);
  String text = doubleToDate(value);
  return SideTitleWidget(
    axisSide: meta.axisSide,
    space: 4,
    child: Text(text, style: style),
  );
}
Widget leftTitleWidgets(double value, TitleMeta meta) {
  const style = TextStyle(
      color: Color(0xFFb2c5b2), fontWeight: FontWeight.bold, fontSize: 13);

  return SideTitleWidget(
    axisSide: meta.axisSide,
    child: Text(
      '${value.toInt()} %',
      style: style,
    ),
  );
}
class GrowCycleDetailStatistic extends StatefulWidget {
  const GrowCycleDetailStatistic({super.key});
  @override
  State<GrowCycleDetailStatistic> createState() =>
```

```
      _GrowCycleDetailStatisticState();
}

class _GrowCycleDetailStatisticState extends
State<GrowCycleDetailStatistic> {
  String startDate = "None";
  String finishDate = "-";

  List<Humidity> humidityData = [];
  List<FlSpot> humiditySpots = [];
  List<Temperature> temperatureData = [];
  List<FlSpot> temperatureSpots = [];
  List<PPM> ppmData = [];
  List<FlSpot> ppmSpots = [];
  List<PH> phData = [];
  List<FlSpot> phSpots = [];
  List<Height> plantHeightData = [];
  List<Height> last12plantHeightData = [];
  List<FlSpot> plantHeightSpots = [];

  String humidityNow = "0";
  String temperatureNow = "0";
  String nutrientNow = "0";
  String pHNow = "0";

  late Map<String, dynamic> growCycleData;

  void fetchGrowCycleEnvironmentData() async {
    String apiUrl = '${globalVar.ApiUrlBase}/grow-cycle-data';

    try {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      String? token = prefs.getString('token');

      if (token == null || token.isEmpty) {
        print('Token not found');
        return;
      }

      final response = await http.post(Uri.parse(apiUrl),
          headers: <String, String>{
            'Authorization': 'Bearer $token',
            'Content-Type': 'application/json',
          },
          body: jsonEncode(
```

```
              <String, dynamic>{'growCycleId': growCycleData['id']}));
      if (response.statusCode == 200) {
        final Map<String, dynamic> responseData =
jsonDecode(response.body);
        setState(() {
          CombinedResponse data = CombinedResponse.fromJson(responseData);
          if (data.humidity.isNotEmpty) {
            humidityData = data.humidity;
            humiditySpots = convertHumidityToFlSpots(humidityData);
            humidityNow =
                data.humidity[data.humidity.length -
1].humidity.toString();
          }
          if (data.temperature.isNotEmpty) {
            temperatureData = data.temperature;
            temperatureSpots =
convertTemperatureToFlSpots(temperatureData);
            temperatureNow = data
                .temperature[data.temperature.length - 1].temperature
                .toString();
          }
          if (data.ppm.isNotEmpty) {
            ppmData = data.ppm;
            nutrientNow = data.ppm[data.ppm.length - 1].ppm.toString();
          }
          if (data.ph.isNotEmpty) {
            phData = data.ph;
            pHNow = data.ph[data.ph.length - 1].ph.toString();
          }
          if (data.height.isNotEmpty) {
            plantHeightData = data.height;
            last12plantHeightData = plantHeightData.sublist(
                plantHeightData.length - 12, plantHeightData.length);
          }
        });
      } else {
        print('Request failed with status code: ${response.statusCode}');
      }
    } catch (error) {
      notification(context, error);
    }
  }

  double createInterval() {
    try {
```

```
      return humidityData[0].time.millisecondsSinceEpoch.toDouble() -
          humidityData[humidityData.length]
              .time
              .millisecondsSinceEpoch
              .toDouble();
    } catch (error) {
      return 0.0;
    }
  }

  @override
  void initState() {
    super.initState();
    fetchGrowCycleEnvironmentData();
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    final args = ModalRoute.of(context)!.settings.arguments;
    print(args);
    growCycleData = args as Map<String, dynamic>;
    startDate = formatTimeFull(growCycleData["start_date"]);
    finishDate = growCycleData["finish_date"] != null
        ? formatTimeFull(growCycleData["finish_date"])
        : "-";
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: const Text(
          "Grow Cycle Detail",
          style: TextStyle(
            color: Color(0xFFd5dddf),
          ),
        ),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/statistic');
          },
        ),
```

```
      return
humidityData[0].time.millisecondsSinceEpoch.toDouble() -
          humidityData[humidityData.length]
              .time
              .millisecondsSinceEpoch
              .toDouble();
    } catch (error) {
      return 0.0;
    }
  }
  @override
  void initState() {
    super.initState();
    fetchGrowCycleEnvironmentData();
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    final args = ModalRoute.of(context)!.settings.arguments;
    print(args);
    growCycleData = args as Map<String, dynamic>;
    startDate = formatTimeFull(growCycleData["start_date"]);
    finishDate = growCycleData["finish_date"] != null
        ? formatTimeFull(growCycleData["finish_date"])
        : "-";
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(
        title: const Text(
          "Grow Cycle Detail",
          style: TextStyle(
            color: Color(0xFFd5dddf),
          ),
        ),
        leading: GestureDetector(
          child: const Icon(
            Icons.arrow_back_ios,
          ),
          onTap: () {
            Navigator.pushReplacementNamed(context, '/statistic');
          },
        ),
```

```
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            Row(
              children: [
                Expanded(
                  child: Column(
                    children: [
                      const Text(
                        "Start Date",
                        style: TextStyle(
                            color: Color(0xFFb2c5b2),
                            fontWeight: FontWeight.bold,
                            fontSize: 20),
                      ),
                      Text(
                        startDate,
                        style: const TextStyle(
                            color: Color(0xFFb2c5b2),
                            fontWeight: FontWeight.bold,
                            fontSize: 20),
                      )
                    ],
                  ),
                ),
                Expanded(
                  child: Column(
                    children: [
                      const Text(
                        "Finish Date",
                        style: TextStyle(
                            color: Color(0xFFb2c5b2),
                            fontWeight: FontWeight.bold,
                            fontSize: 20),
                      ),
                      Text(
                        finishDate,
                        style: const TextStyle(
                            color: Color(0xFFb2c5b2),
                            fontWeight: FontWeight.bold,
                            fontSize: 20),
                      )
                    ],
                  ),
                )
              ],
```

```
            ),
            HumidityChart(spots: humiditySpots, humidityData:
humidityData),
            TemperatureChart(
                spots: temperatureSpots, temperatureData:
temperatureData),
            Row(
              children: [
                Expanded(
                    child: Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: const Color(0xFF3c5148),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                    ),
                    onPressed: () {
                      if (growCycleData['active_status'] == false) {
                        finishGrowCycle(context, growCycleData['id']);
                      }},
                    child: Text(
                        growCycleData['active_status'] == true
                            ? "Selesaikan"
                            : "Sudah selesai",
                        style: const TextStyle(
                            color: Color(0xFFb2c5b2),
                            fontWeight: FontWeight.bold,
                            fontSize: 15)),
                  )))
              ],
            ),
            Row(
              children: [
                Expanded(
                  child: Padding(
                    padding: const EdgeInsets.all(8.0),
                    child: SizedBox(
                      height: 100,
                      child: Card(
                        child: Padding(
                          padding: const EdgeInsets.all(8.0),
                          child: Column(
                            children: [
                              const Expanded(
                                child: Text(
```

```
                                    "Hasil Panen (Gram)",
                                    style: TextStyle(
                                        color: Color(0xFFb2c5b2),
                                        fontWeight: FontWeight.bold,
                                        fontSize: 13),
                                  ),
                                ),
                                Expanded(
                                  child: Text(
growCycleData['total_harvest'].toString() !=
                                            "null"
                                        ? growCycleData['total_harvest']
                                            .toString()
                                        : "NO data yet",
                                    style: const TextStyle(
                                        color: Color(0xFFb2c5b2),
                                        fontWeight: FontWeight.bold,
                                        fontSize: 13),
                                  ),
                                ),
                              ],
                            ),
                          ),
                        ),
                      ),
                    ),
                  )
              ],
            ),
            GridView.count(
              physics: const NeverScrollableScrollPhysics(),
              padding: const EdgeInsets.all(10),
              shrinkWrap: true,
              primary: true,
              crossAxisSpacing: 3,
              mainAxisSpacing: 3,
              crossAxisCount: 3,
              children: List.generate(
                12,
                (index) {
                  return HeightCard(
                      data: last12plantHeightData.isNotEmpty
                          ? last12plantHeightData[index]
                          : "none");
                },
              ),
            ),
```

```
          ],
        ),
      ),
    );
  }
}
```

Segmen Program 4.99 Statistic grow cycle detail page

Pada segmen program 4.99 adalah implementasi UI dan logika untuk membuat halaman statistic grow cycle detail dimana pengguna bisa melihat informasi mengenai data sensor pada grow cycle tersebut.

```
import 'package:flutter/material.dart';
import 'package:microgreen_monitoring_app/util/formatingTime.dart';

class GrowCycleCardStatistic extends StatelessWidget {
  const GrowCycleCardStatistic({super.key, required this.data});
  final data;
  @override
  Widget build(BuildContext context) {
    return Card(
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          ListTile(
            title: Text(data['grow_cycle_name']),
            titleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 20),
            subtitle: Row(
              children: [
                Expanded(
                    child: Text(
                        'Start date:
${formatTimeFull(data['start_date'])}')),
                Expanded(
                  child: Text(
                    'Finish date: ${data['finish_date'] != null ?
formatTimeFull(data['finish_date']) : "not finish"}',
                    textAlign: TextAlign.right,
                  ),
                ),
```

```
              ],
            ),
            subtitleTextStyle: const TextStyle(
                color: Color(0xFFb2c5b2),
                fontWeight: FontWeight.bold,
                fontSize: 15),
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.end,
            children: <Widget>[
              TextButton(
                child: const Text(
                  'Detail',
                  style: TextStyle(
                      color: Color(0xFFb2c5b2),
                      fontWeight: FontWeight.bold,
                      fontSize: 20),
                ),
                onPressed: () {
                  Navigator.pushReplacementNamed(context, '/statistic-detail',
                      arguments: data);
                },
              ),
              const SizedBox(width: 8),
              Text(
                data['finish_date'] != null ? 'Completed' : "Complete",
                style: const TextStyle(
                    color: Color(0xFFb2c5b2),
                    fontWeight: FontWeight.bold,
                    fontSize: 20),
              ),
              const SizedBox(width: 8),
            ],
          ),
        ],
      ),
    );
  }
}
```

Segmen Program 4.100 Statistic grow cycle card componen

Pada segmen program 4.100 adalah implementasi komponen UI dan logika yang di gunakan untuk menampilkan data mengenai grow cycle yang di miliki oleh pengguna pada halaman statistic pada segmen program 4.98.