4. IMPLEMENTASI SISTEM

4.1 Perangkat Lunak yang Digunakan

Aplikasi mobile untuk sekolah ini dibuat dengan menggunakan teknologi Flutter, PHP, MySQL, dan Firebase. Flutter digunakan untuk pengembangan antarmuka pengguna (frontend). Library Flutter yang digunakan antara lain http, dio untuk komunikasi API, shared_preferences untuk penyimpanan data lokal, provider untuk manajemen state, intl untuk format tanggal dan waktu, serta cloud_firestore untuk integrasi dengan Firebase Firestore. PHP digunakan dalam pembuatan logika backend, sebagai penghubung antara frontend dan backend, serta pengiriman data secara get dan post ke dalam database. Database yang digunakan adalah MySQL, yang datanya sudah langsung ada di server sekolah, memastikan penyimpanan data yang aman dan efisien. Firebase digunakan untuk manajemen data real-time, khususnya dalam fitur chat dan notifikasi, memfasilitasi sinkronisasi data dan pemberitahuan instan kepada pengguna.

4.2 Langkah-langkah Implementasi

Untuk mengembangkan aplikasi mobile untuk sekolah ini, dilakukan beberapa langkah implementasi yang terstruktur dan sistematis. Berikut adalah tahapan-tahapan yang dilakukan:

4.2.1 Persiapan Lingkungan Pengembangan

1. Instalasi Flutter dan SDK

   - Mengunduh dan menginstal Flutter SDK dari situs resmi Flutter.
   - Menambahkan Flutter ke PATH sistem untuk akses command line.
   - Memastikan dependensi yang diperlukan seperti Android Studio sudah terinstal.

2. Instalasi Tools Pendukung

   - Menginstal Visual Studio Code sebagai Integrated Development Environment (IDE).
   - Menginstal plugin Flutter dan Dart pada Visual Studio Code

3. Pengaturan Proyek

   - Membuat proyek baru menggunakan perintah flutter create ParentSquare

- Mengatur struktur direktori proyek dan menambahkan library yang diperlukan dalam file pubspec.yaml

4.2.2 Pengembangan *Frontend* dengan Flutter

1. Desain UI/UX

- Membuat desain antarmuka pengguna (UI) menggunakan widget-widget Flutter.
- Mengimplementasikan navigasi antar halaman menggunakan Navigator.

2. Komunikasi API

- Menggunakan library http dan dio untuk mengatur komunikasi API.
- Mengimplementasikan fungsi GET dan POST untuk mengirim dan menerima data dari server

3. Manajemen State

- Menggunakan provider untuk manajemen state, memastikan data dapat diakses di seluruh aplikasi.
- Mengatur state aplikasi untuk fitur-fitur seperti autentikasi pengguna, pengelolaan data siswa, dan notifikasi.

4. Penyimpanan Data Lokal

- Menggunakan shared_preferences untuk menyimpan data pengguna secara lokal di perangkat.

5. Format Tanggal dan Waktu

- Menggunakan intl untuk memformat tanggal dan waktu sesuai dengan kebutuhan aplikasi.

6. Integrasi Firebase

- Mengatur proyek Firebase dan menambahkan Firebase SDK ke proyek Flutter.
- Menggunakan cloud_firestore untuk integrasi dengan Firestore, menyediakan penyimpanan data real-time.

4.2.3 Pengembangan *Backend* dengan PHP

1. Pengaturan Database

- Membuat dan mengatur struktur database MySQL.
- Mengatur koneksi antara PHP dan MySQL, memastikan keamanan dan efisiensi dalam pengiriman dan pengambilan data.

4.2.4 Pengujian dan *Debugging*

1. Pengujian Unit

- Melakukan pengujian unit pada setiap komponen aplikasi untuk memastikan fungsionalitas berjalan sesuai harapan.

2. Pengujian Integrasi

- Menguji integrasi antara frontend dan backend, memastikan data dapat dikirim dan diterima dengan benar.

3. Pengujian dengan User

- Melakukan pengujian aplikasi secara langsung dengan user guru dan orang tua di TK Pelangi Kristus

4.2.5 *Deployment*

1. Deploy Backend

- Mengunggah skrip PHP ke server dan mengonfigurasi pengaturan server untuk mendukung aplikasi.

2. Deploy Frontend

- Menghasilkan build aplikasi Flutter untuk Android
- Melakukan build aplikasi Flutter, dan menghasilkan aplikasi dalam bentuk *apk* dan disebarkan ke guru dan orang tua.

4.3 Implementasi Sistem

Pada bab sebelumnya, terdapat desain sistem yang menjelaskan fitur dan proses apa saja yang terdapat di dalam aplikasi. Kemudian desain sistem tersebut akan diimplementasikan

kedalam pembuatan program dalam bentuk beberapa segmen. Pembagian segmen program akan dipetakan pada tabel 4.1.

Tabel 4. 1 Hubungan Desain Sistem dan Segmen Program

Flowchart	**Segmen**	**Nama segmen**	**Keterangan**
Gambar 3.2	Segmen Program 4.1	Inisialisasi flutter dan firebase	Proses inisialisasi flutter dengan firebase
Gambar 3.2	Segmen Program 4.2	Function stuktur class	Function struktur class dalam sistem
Gambar 3.2	Segmen Program 4.3	Function Login	Proses login dalam sistem
Gambar 3.2	Segmen Program 4.4	Function Change Password	Proses ubah password
Gambar 3.2	Segmen Program 4.5	Function Logout	Proses logout dalam sistem
Gambar 3.3	Segmen Program 4.6	Menampilkan Attendance	Menampilkan absensi siswa
Gambar 3.3	Segmen Program 4.7	Menghapus data Attendance	Menghapus absensi siswa
Gambar 3.3	Segmen Program 4.8	Mengedit data Attendance	Mengedit absensi siswa
Gambar 3.3	Segmen Program 4.9	Membuat data Attendance	Menambahkan data kelas dan tanggal absensi
Gambar 3.3	Segmen Program 4.10	Membuat data Attendance Detail	Menyimpan data status murid
Gambar 3.3	Segmen Program 4.11	Menggunggah Gambar	Menggunggah gambar ke dalam sistem

Gambar 3.4	Segmen Program 4.12	Menampilkan Achievement	Menampilkan *daily achievement* siswa
Gambar 3.4	Segmen Program 4.13	Edit Achievement	Mengedit *daily achievement* siswa
Gambar 3.4	Segmen Program 4.14	Hapus Achievement	Menghapus *daily achievement* siswa
Gambar 3.4	Segmen Program 4.15	Add New Achievement	Menambahkan data *daily achievement* siswa
Gambar 3.7	Segmen Program 4.16	Menampilkan pengumuman	Menampilkan data *announcement*
Gambar 3.7	Segmen Program 4.17	Edit pengumuman	Mengedit data *announcement*
Gambar 3.7	Segmen Program 4.18	Hapus pengumuman	Menghapus data *announcement*
Gambar 3.7	Segmen Program 4.19	Menyimpan data pengumuman	Menyimpan data *announcement*
Gambar 3.7	Segmen Program 4.20	Menampilkan data ijin	Menampilkan data p*ermissions*
Gambar 3.6	Segmen Program 4.21	Update status data ijin	Melakukan update status data *permissions* dari orang tua
Gambar 3.6	Segmen Program 4.22	Hapus data ijin	Menghapus status data *permissions*
Gambar 3.6	Segmen Program 4.23	Menyimpan Riwayat kesehatan	Menambahkan data *health record*

Gambar 3.5	Segmen Program 4.24	Mengedit Riwayat kesehatan	Mengedit data *health record*
Gambar 3.5	Segmen Program 4.25	Menghapus data kesehatan anak	Menghapus data *health record*
Gambar 3.5	Segmen Program 4.26	Menampilkan data kesehatan anak	Menampilkan data *health record*
Gambar 3.8	Segmen Program 4.27	Chat Live	Menampilkan percakapan orang tua dan guru

4.3.1 Inisialisasi aplikasi flutter dan firebase

Inisialisasi aplikasi Flutter dan Firebase dimulai dengan membuat proyek Flutter baru dan menghubungkannya dengan proyek Firebase di Firebase Console. Proses ini melibatkan penambahan file konfigurasi Firebase yang diperlukan untuk platform Android dan iOS, serta menambahkan dependensi Firebase ke dalam proyek Flutter.

Segmen Program 4. 1 Inisialisasi aplikasi flutter dan firebase

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
  ]);

  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  await NotificationService.initialize();

  FirebaseMessaging messaging = FirebaseMessaging.instance;

  // Meminta izin notifikasi
  NotificationSettings settings = await
messaging.requestPermission(
    alert: true,
    badge: true,
    sound: true,
  );
  if (settings.authorizationStatus ==
AuthorizationStatus.authorized) {
    print('User granted permission');
```

```
  } else if (settings.authorizationStatus ==
AuthorizationStatus.provisional) {
    print('User granted provisional permission');
  } else {
    print('User declined or has not accepted permission');
  }
  ThemeHelper().changeTheme('primary');

  String initialRoute = await getInitialRoute();
  runApp(MyApp(initialRoute: initialRoute));
}
```

4.3.2 *Function structur class*

Kelas ini mengatur dasar aplikasi, termasuk pengaturan rute (route) untuk navigasi antar halaman dan konfigurasi notifikasi. Pada bagian pengaturan rute, aplikasi menentukan halaman-halaman mana yang dapat diakses dan bagaimana pengguna berpindah antar halaman tersebut.

Segmen Program 4. 2 *Function Structur Class*

```
class MyApp extends StatelessWidget {
  final String initialRoute;
  MyApp({required this.initialRoute});

  @override
  Widget build(BuildContext context) {
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      RemoteNotification? notification = message.notification;
      AndroidNotification? android =
message.notification?.android;

      if (notification != null && android != null) {
        globalMessengerKey.currentState?.showSnackBar(
          SnackBar(
            content: Text(notification.body ?? ''),
          ),
        );
      }
    });
```

4.3.3 *Function Login*

Fitur login dalam aplikasi bertujuan untuk memverifikasi identitas pengguna, memastikan keamanan data pribadi, memungkinkan pengalaman yang dipersonalisasi, mengelola hak akses berdasarkan peran, dan melacak aktivitas pengguna.

Segmen Program 4. 3 *Function Login*

```
Future<void> login(BuildContext context) async {
  setState(() {
    _isLoading = true;
  });

  try {
    String? token = await FirebaseMessaging.instance.getToken();

    final response = await http.post(
      Uri.parse('$ip2/checklogin.php'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
        'X-API-KEY': apikey,
      },
      body: jsonEncode(<String, String>{
        'username': usernameController.text,
        'password': passwordController.text,
        'token': token ?? "",
      }),
    );

    if (response.statusCode == 200) {
      var data = json.decode(response.body);

      if (data != null && data["jwt"] != null && data["user"] !=
null) {
        SharedPreferences prefs = await
SharedPreferences.getInstance();

        await prefs.setString('username',
data["user"]["username"]);
        await prefs.setString('nama', data["user"]["nama"]);
        await prefs.setString('userRole',
data["user"]["status"]);
        await prefs.setString('no_induk',
data["user"]["no_induk"]);
        await prefs.setString('jwt', data["jwt"]);

        var anak = data["user"]["anak"];
        List<dynamic> children = [];

        if (anak is List) {
          children = anak;
        } else if (anak is Map) {
          children = anak.entries.map((entry) {
            return {...entry.value, "key": entry.key};
          }).toList();
        }

        List<dynamic> activeChildren =
            children.where((child) => child["is_active"] ==
1).toList();
        if (activeChildren.isNotEmpty) {
```

```
          String anakJson = jsonEncode(activeChildren);
          await prefs.setString('children', anakJson);
          await prefs.setString('noIndukAnak',
activeChildren[0]["no_induk"]);
        }

        String userRole = data["user"]["status"];
        String username = usernameController.text;
        if (userRole == "O" && children.isNotEmpty) {
          String noIndukAnak = children[0]["no_induk"];
          if (token != null) {
            await
FirebaseFirestore.instance.collection('users').doc(noIndukAnak).
set({
              'fcmToken': token,
              'nama': username,
              'role': userRole,
              'noInduk': noIndukAnak,
            }, SetOptions(merge: true));
          }
        } else if (userRole == "G") {
          String noIndukGuru = data["user"]["no_induk"];
          if (token != null) {
            await
FirebaseFirestore.instance.collection('users').doc(noIndukGuru).
set({
              'fcmToken': token,
              'nama': username,
              'role': userRole,
              'noInduk': noIndukGuru,
            }, SetOptions(merge: true));
          }
        }

        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) =>
DashboardContainerScreen()),
        );
      } else {
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text("Login failed: incorrect username or
password"),
          backgroundColor: Colors.redAccent,
        ));
      }
    } else {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text("Login failed with status code:
${response.statusCode}"),
      ));
    }
  } catch (e) {
```

```
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text("An error occurred during login"),
    ));
  } finally {
    setState(() {
      _isLoading = false;
    });
  }}
```

4.3.4 *Function Change Password*

Fitur *change password* dalam aplikasi bertujuan untuk memungkinkan pengguna untuk melakukan perubahan kata sandi dalam aplikasi ini.

Segmen Program 4. 4 *Function Change Password*

```
Future<void> changePassword(BuildContext context) async {
  SharedPreferences prefs = await
SharedPreferences.getInstance();
  String? token = prefs.getString('jwt');
  String username = prefs.getString('username') ?? '';

  if (newpasswordController.text !=
confirmpasswordController.text) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text("New password and confirmation do not
match."),
      backgroundColor: Colors.redAccent,
    ));
    return;
  }

  final response = await http.put(
    Uri.parse('https://api.pelangikristus.or.id/users/$username/
password'),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8',
      'X-API-KEY': apikey,
      'Authorization': 'Bearer $token',
    },
    body: jsonEncode(<String, String>{
      'oldpass': oldpasswordController.text,
      'newpass': confirmpasswordController.text,
    }),
  );

  if (response.statusCode == 200) {
    var responseData = json.decode(response.body);
    if (responseData['data'][0] == true) {
      ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text("Password changed
successfully!")));
```

```
      Navigator.pop(context);
    } else {
      print("Failed to change password:
${responseData['errors']}");
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text("Cannot change password: " +
              (responseData['errors'].isEmpty
                  ? "Unknown error"
                  : responseData['errors'][0]))));
    }
  } else {
    var responseData = json.decode(response.body);
    print("Server error: ${responseData['errors']}");
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text(responseData['errors'].isEmpty
            ? "Failed to change password due to server error."
            : responseData['errors'][0])));
  }
}
```

4.3.5 *Function Logout*

Fitur *logout* dalam aplikasi bertujuan untuk opsi yang memungkinkan pengguna untuk keluar dari akun atau sesi yang sedang aktif.

Segmen Program 4. 5 *Function Logout*

```
class DashboardLogout extends StatelessWidget {
  const DashboardLogout({Key? key}) : super(key: key);

  void _logout(BuildContext context) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    await prefs.clear(); // Menghapus semua data yang tersimpan
di SharedPreferences

    Navigator.pushReplacement(
      // Mengganti layar dengan SignInScreen
      context,
      MaterialPageRoute(builder: (context) => SignInScreen()),
    );
  }
```

4.3.6 Menampilkan *Attendance*

Fitur ini berfungsi untuk menyimpan data kehadiran siswa. *User* yang menggunakan adalah guru dan siswa. Guru *login* ke aplikasi dan masuk ke fitur absensi. Sistem akan menampilkan daftar seluruh siswa yang ada, kemudian guru akan memilih siswa berdasarkan

nama siswa dan memilih kehadiran. Kemudian *user* melakukan simpan data, data akan tersimpan di tabel *Attendance*.

Segmen Program 4. 6 Menampilkan *Attendance*

```
Widget _buildItem(AttendanceData attendanceData) {
    return GestureDetector(
      onTap: () {
        showModalBottomSheet(
          context: context,
          backgroundColor:
              Colors.transparent, // Mengatur latar belakang
transparan
          builder: (context) {
            return _buildDetailModal(attendanceData);
          },
        );
      },
      child: Container(
        width: 356.6,
        padding: EdgeInsets.symmetric(horizontal: 15.0,
vertical: 17.0),
        decoration: AppDecoration.outlineBlue900.copyWith(
          borderRadius: BorderRadiusStyle.roundedBorder8,
        ),
        child: Row(
          mainAxisAlignment: MainAxisAlignment
              .spaceBetween, // Menyebarkan anak secara
horizontal
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Expanded(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    attendanceData.studentName,
                    style: CustomTextStyles.titleMediumPrimary,
                  ),
                  SizedBox(height: 13.0),
                  _buildClassButton(attendanceData),
                  SizedBox(height: 13.0),
                  Text(
                    attendanceData.date,
                    style:
theme.textTheme.bodySmall!.copyWith(height: 1.50),
                  ),
                  SizedBox(height: 13.0),
                  Text(
                    attendanceData.status,
```

```
                      style:
theme.textTheme.bodySmall!.copyWith(height: 1.50),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      );
  }
```

4.3.7 Menghapus data *Attendance*

Fitur ini berfungsi untuk menghapus data kehadiran siswa. Fungsi yang digunakan seperti dibawah ini.

Segmen Program 4. 7 Menghapus *data Attendance*

```
Future<bool> _deleteAttendanceData(AttendanceData
attendanceData) async {
    try {
      final response = await dio.delete(
        '$ip2/delete_attendance.php',
        data: {'Attendance_ID': attendanceData.attendanceID},
        options: Options(headers: {'X-API-KEY': apikey}),
      );

      if (response.statusCode == 200) {
        setState(() {
          attendanceList.removeWhere(
              (element) => element.attendanceID ==
attendanceData.attendanceID);
          filteredAttendanceData.removeWhere(
              (element) => element.attendanceID ==
attendanceData.attendanceID);
        });

        // Fetch data ulang setelah penghapusan
        await _fetchAttendanceData();

        // Kirimkan daftar absensi terbaru kembali ke halaman
sebelumnya
        Navigator.pop(context, attendanceList);
        return true;
      } else {
        print('Failed to delete data. Status code:
${response.statusCode}');
        return false;
      }
    } catch (error) {
      print('Error deleting data: $error'); return false; }}
```

4.3.8 Mengedit data *Attendance*

Fitur ini berfungsi untuk melakukan perubahan pada data kehadiran siswa. Fungsi yang digunakan seperti dibawah ini.

Segmen Program 4. 8 Mengedit *data Attendance*

```
Future<void> _saveAttendanceChanges() async {
  List<Future> futures = [];
  for (var student in students) {
    var future = dio.post(
      '$ip2/edit_attendance.php',
      data: {
        'Detail_ID': student.attendanceDetailID,
        'Status': attendanceStatus[student.attendanceDetailID]
?? student.status,
      },
      options: Options(headers: {'X-API-KEY': apikey}),
    );
    futures.add(future);
  }

  try {
    await Future.wait(futures);
  } catch (e) {
    print("Failed to update attendance: $e");
    throw Exception('Update failed');
  }
}

Widget _buildSaveButton(BuildContext context) {
  return Expanded(
    child: CustomElevatedButton(
      text: "Save",
      margin: EdgeInsets.only(left: 6.h),
      onPressed: () async {
        try {
          await _saveAttendanceChanges();
          widget.onRefresh();
          Navigator.pushNamed(context,
AppRoutes.homeFilledScreen);
        } catch (e) {
          print("Error saving changes: $e");

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content:
Text('Failed to save changes')));
        }
      },
    ),
  );}
```

4.3.9 Membuat data *Attendance*

Fitur ini berfungsi untuk menyimpan data kelas dan tanggal yang dibuat. Fungsi yang digunakan seperti dibawah ini.

Segmen Program 4. 9 Membuat *data Attendance*

```
Future<void> _saveAttendanceData(BuildContext context) async {
  try {
    var dateFormat = DateFormat('yyyy-MM-dd');
    var date = dateFormat.format(selectedDate);
    var classID = selectedLevel;

    if (!isValidClassID(classID)) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
        content: Text('Invalid class selected.'),
      ));
      return;
    }

    final queryParams = {
      'Date': Uri.encodeComponent(date),
      'Class': Uri.encodeComponent(classID!),
    };

    final queryString = queryParams.entries.map((e) =>
'${e.key}=${e.value}').join('&');

    final response = await dio.get(
      '$ip2/savedata_attendance.php?$queryString',
      options: Options(headers: {'X-API-KEY': apikey}),
    );
```

4.3.10 Membuat data *Attendance Detail*

Fitur ini berfungsi untuk menyimpan data kehadiran siswa ke server. Fungsi yang digunakan seperti dibawah ini

Segmen Program 4. 10 Membuat *data Attendance Detail*

```
Future<String> getLatestAttendanceId() async {
  final response = await dio.get(
    '$ip2/getdata_attendanceID.php',
    options: Options(headers: {'X-API-KEY': apikey}),
  );
  if (response.statusCode == 200) {
    var data = json.decode(response.data);
    return data['Attendance_ID'];
  } else {
    throw Exception('Failed to load latest Attendance_ID');
  }
}
```

```
Future<void> saveAttendanceDetails(String attendanceId, Student
student) async {
  final response = await dio.post(
    '$ip2/savedata_attendancedetail.php',
    data: jsonEncode({
      'Attendance_ID': attendanceId,
      'Status': student.status,
      'Student_Name': student.name,
      'No_Induk': student.noInduk,
    }),
    options: Options(
      headers: {
        'X-API-KEY': apikey,
        'Content-Type': 'application/json',
      },
    ),
  );
  if (response.statusCode != 200) {
    throw Exception('Failed to save attendance details for
${student.name}');
  }
}
```

4.3.11 Menggunggah gambar

Fitur ini berfungsi untuk mengunggah file gambar di fitur achievement. Gambar akan disimpan di database.

Segmen Program 4. 11 Mengunggah gambar

```
Future<void> _pickImageFromGallery() async {
  final pickedImage = await ImagePicker().getImage(source:
ImageSource.gallery);

  if (pickedImage == null) {
    return;
  }

  setState(() {
    _pickedImage = File(pickedImage.path);
    _image = _pickedImage;
  });
}

Future<String?> uploadImage() async {
  if (_image == null) {
    print('No image selected.');
    return null;
  }

  String fileName = _image!.path.split('/').last;
```

```
  FormData formData = FormData.fromMap({
    'file': await MultipartFile.fromFile(_image!.path, filename:
fileName),
  });

  try {
    Response response = await dio.post(
      ip2 + '/uploadimage.php',
      data: formData,
      options: Options(
        headers: {
          'Content-Type': 'multipart/form-data',
        },
      ),
    );
```

4.3.12 Menampilkan *Achievement*

Fitur ini berfungsi untuk menyimpan catatan harian setiap siswa. Guru dapat *login* kedalam sistem, dan masuk ke fitur *Daily Achievements*. Guru dapat memilih siswa, dimana sistem akan menampilkan riwayat aktivitas siswa tersebut dan guru juga dapat menginputkan aktivitas anak. Setelah diinputkan, guru dapat melakukan simpan data.

Segmen Program 4. 12 Menampilkan *Achievement*

```
Widget _buildStudentItem(String studentName, BuildContext
context) {
  return GestureDetector(
    onTap: () => _navigateToStudentAchievementPage(studentName),
    child: Container(
      width: 356.6,
      padding: EdgeInsets.symmetric(horizontal: 15.0, vertical:
17.0),
      decoration: AppDecoration.outlineBlue900.copyWith(
        borderRadius: BorderRadiusStyle.roundedBorder8,
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Expanded(
            child: Text(
              studentName,
              style: CustomTextStyles.titleMediumPrimary,
            ),
          ),
          Icon(Icons.arrow_forward_ios, size: 20),
        ],
      ),
    ),
  );}
```

4.3.13 Edit *Achievement*

Fitur ini berfungsi untuk melakukan edit catatan harian setiap siswa. Guru dapat *login* kedalam sistem, dan masuk ke fitur *Daily Achievements*. Guru dapat memilih siswa, dimana sistem akan menampilkan riwayat *achievement*.

Segmen Program 4. 13 *Edit Achievement*

```
void saveData(BuildContext context) async {
    final data = {
      'Achievements_ID': widget.achievementsData.achievementsID,
      'Student_name': selectedStudent,
      'Class': selectedLevel,
      'Date': DateFormat('yyyy-MM-dd').format(selectedDate),
      'Description': detailedexplanationController.text,
      'Achievements_name': achievementController.text,
      'Grade': selectedGrade,
    };

    final response = await dio.post(
      '$ip2/edit_achievements.php', // Pastikan URL ini benar
      data: data,
    );


    if (response.statusCode == 200) {
      Navigator.pop(
          context, true); // Return true if data was
successfully updated
    } else {
      ScaffoldMessenger.of(context)
          .showSnackBar(SnackBar(content: Text('Error updating
data')));
    }
 }
```

4.3.14 Delete *Achievement*

Fitur ini berfungsi untuk melakukan penghapusan catatan harian siswa. Guru dapat *login* kedalam sistem, dan masuk ke fitur *Daily Achievements*. Guru dapat memilih siswa, dimana sistem akan menampilkan riwayat *achievement*.

Segmen Program 4. 14 Hapus *Achievement*

```
Future<bool> _deleteDataFromDatabase(AchievementsData
achievementsData) async {
    try {
        final response = await dio.delete(
        '$ip2/delete_achievements.php',
        data: {'Achievements_ID':
achievementsData.achievementsID},
```

```
        options: Options(headers: {'X-API-KEY': apikey}),);
        if (response.statusCode == 200) {
        setState(() {
            achievementsList.removeWhere((element) =>
            element.achievementsID ==
achievementsData.achievementsID);
            filteredAchievementsDataList.removeWhere((element)
=>
            element.achievementsID ==
achievementsData.achievementsID);
        });
        return true;
        } else {
        print('Failed to delete data. Status code:
${response.statusCode}');
        return false;
        }
    } catch (error) {
        print('Error deleting data: $error');
        return false;
    }
}
```

4.3.15 Add New *Achievement*

Fitur ini berfungsi untuk melakukan penyimpanan catatan harian siswa. Guru dapat *login* kedalam sistem, dan masuk ke fitur *Daily Achievements*. Guru dapat memilih siswa, dimana sistem akan menampilkan riwayat *achievement*.

Segmen Program 4. 15 *Add New Achievement*

```
Future<void> _saveAchievements(BuildContext context) async {
    if (_image == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("Please select an image to
upload.")),
      );
      return;
    }

    setState(() {
      _isLoading = true; // Aktifkan indikator loading
    });

    String? imagePath = await uploadImage(); // Upload gambar
dan dapatkan path

    if (imagePath == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Image upload failed. Please try
again.')),
      );
      if (mounted) {
        setState(() {
```

```
          _isLoading =
              false;
        });
      }
      return;
    }

    SharedPreferences prefs = await
SharedPreferences.getInstance();
    String? jwtToken = prefs.getString('jwt');
    if (jwtToken == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text("JWT token is not found. Please
login again.")),
      );
      if (mounted) {
        setState(() {
          _isLoading =
              false; // Nonaktifkan state loading jika token
tidak ditemukan
        });
      }
      return;
    }

    try {
      var dateFormat = DateFormat('yyyy-MM-dd');
      var date = dateFormat.format(selectedDate);
      var student = selectedStudent;
      var classID = selectedLevel;
      var achievementName = achievementController.text;
      var grade = selectedGrade;
      var description = detailedexplanationController.text;
      var noInduk = selectedNoInduk;

      if (!isValidClassID(classID)) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
              content:
                  Text('Invalid class selected. Please choose a
valid class.')),
        );
        if (mounted) {
          setState(() {
            _isLoading = false;
          });
        }
        return;
      }

      final postData = FormData.fromMap({
        'Date': date,
        'Student_name': student,
```

```
        'No_Induk': Uri.encodeComponent(noInduk!),
        'Class': classID,
        'Achievements_name': achievementName,
        'Grade': grade,
        'Description': description,
        'Path': imagePath, // Kirim path gambar sebagai bagian
dari data
      });

      final response = await dio.post(
        '$ip2/savedata_achievements.php',
        data: postData,
        options: Options(
          headers: {
            'Authorization': 'Bearer $jwtToken'
          }, // Menggunakan token JWT sebagai header
Authorization
        ),
      );

      if (response.statusCode == 200) {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => HomeFilledAchievements(),
          ),
        );
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
              content:
                  Text('Failed to save achievements. Please try
again later.')),
        );
      }
    } catch (error) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('An error occurred: $error')),
      );
    } finally {
      if (mounted) {
        setState(() {
          _isLoading =
              false; // Pastikan untuk mematikan indikator
loading ketika proses selesai atau jika ada error
        });
      }
    }
  }
```

4.3.16 Menampilkan Pengumuman

Fitur ini berfungsi menampilkan pengumuman. *User* dari fitur ini adalah guru. *User* dapat *login* ke aplikasi dan membuka fitur pengumuman. Dimana *user* dapat menambahkan atau menginputkan pengumuman baru, kemudian melakukan konfirmasi pengumuman. Setelah dikonfirmasi, pengumuman akan dibagikan. Orang tua akan menerima notifikasi dari pengumuman tersebut.

Segmen Program 4. 16 Menampilkan Pengumuman

```
  Widget _buildAnnouncementItem(Announcement announcement) {
    return Container(
      width: 356.6,
      padding: EdgeInsets.symmetric(horizontal: 15.0, vertical:
17.0),
      decoration: AppDecoration.outlineBlue900.copyWith(
        borderRadius: BorderRadiusStyle.roundedBorder8,
      ),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  announcement.title,
                  style: CustomTextStyles.titleMediumPrimary,
                ),
                SizedBox(height: 13.0),
                Text(
                  announcement.date,
                  style: theme.textTheme.bodySmall,
                ),
                SizedBox(height: 13.0),
                Text(
                  announcement.isi,
                  style:
theme.textTheme.bodySmall!.copyWith(height: 1.50),
                ),
              ],
            ),
          ),
          Transform.translate(
            offset: Offset(10, -10),
            child: PopupMenuButton<String>(
              onSelected: (String result) {
                if (result == 'Edit') {
                  _navigateToEditAnnouncement(context,
announcement);
                } else if (result == 'Delete') {
```

```
                  _confirmDeleteAnnouncement(context,
announcement);
                }
              },
              itemBuilder: (BuildContext context) =>
<PopupMenuEntry<String>>[
                const PopupMenuItem<String>(
                  value: 'Edit',
                  child: Text('Edit'),
                ),
                const PopupMenuItem<String>(
                  value: 'Delete',
                  child: Text('Delete'),
                ),
              ],
              icon: Icon(Icons.more_vert, size: 20),
            ),
          ),
        ],
      ),
    );
```

4.3.17 Edit Pengumuman

Fitur ini berfungsi melakukan edit data pengumuman. *User* dari fitur ini adalah guru. *User* dapat *login* ke aplikasi dan membuka fitur pengumuman. Dimana *user* dapat menambahkan atau menginputkan pengumuman baru, kemudian melakukan konfirmasi pengumuman. Setelah dikonfirmasi, pengumuman akan dibagikan.

Segmen Program 4. 17 *Edit* Pengumuman

```
 void saveData (BuildContext context)async {
     final data = {
     'id': widget.announcement.id,
     'Title': titleController.text,
     'Date': DateFormat('yyyy-MM-dd').format(selectedDate),
     'announcement': detailedexplanationController.text,};

     Final response = await.dio.post(
     '$ip2/edit_announcement.php',
     Data: data,);

     If (response.statusCode == 200) {
     Navigator.pop(context, true);
     } else {
     ScaffoldMessenger.of(context).showSnackbar(Snackbar(contex
t : Text('Error updating data'))); }}
```

4.3.18 Hapus Pengumuman

Fitur ini berfungsi melakukan hapus data pengumuman. *User* dari fitur ini adalah guru. *User* dapat *login* ke aplikasi dan membuka fitur pengumuman. Dimana *user* dapat

menambahkan atau menginputkan pengumuman baru, kemudian melakukan konfirmasi pengumuman. Setelah dikonfirmasi, pengumuman akan dibagikan. Orang tua akan menerima notifikasi dari pengumuman tersebut.

Segmen Program 4. 18 Hapus Pengumuman

```
  void _deleteAnnouncementFromDatabase(Announcement
announcement) async {
    try {
      final response = await dio.delete(
        '$ip2/delete_pengumuman.php',
        data: {'id': announcement.id},
        options: Options(
          headers: {'X-API-KEY': apikey},
        ),
      );
      if (response.statusCode == 200) {
        setState(() {
          announcements.removeWhere((element) => element.id ==
announcement.id);
        });
        print('Announcement deleted successfully');
      } else {
        print('Failed to delete announcement. Status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error deleting announcement: $error');
    }
```

4.3.19 Simpan data pengumuman

Fitur ini berfungsi menyimpan data pengumuman. *User* dari fitur ini adalah guru. *User* dapat *login* ke aplikasi dan membuka fitur pengumuman. Dimana *user* dapat menambahkan atau menginputkan pengumuman baru, kemudian melakukan konfirmasi pengumuman. Setelah dikonfirmasi, pengumuman akan dibagikan. Orang tua akan menerima notifikasi dari pengumuman tersebut.

Segmen Program 4. 19 Menyimpan data pengumuman

```
Future<void> saveData(
      BuildContext context, String title, String date, String
content) async {
    SharedPreferences prefs = await
SharedPreferences.getInstance();
    String? jwtToken = prefs.getString('jwt');

    if (jwtToken == null) {
      print('JWT token is not available');
```

```
      ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text("No JWT token found, please
login again.")));
      return;
    }

    try {
      var response = await dio.post(
        '$ip2/savedata_announcement.php',
        data: jsonEncode({
          'Title': title,
          'Date': date,
          'announcement': content,
        }),
        options: Options(
          headers: {
            'Authorization': 'Bearer $jwtToken',
            'Content-Type': 'application/json; charset=UTF-8'
          },
        ),
      );

      if (response.statusCode == 200) {
        print('Announcement saved successfully');
        await sendNotification();
        Navigator.pop(context, true); // Kirim true sebagai
hasil
      } else {
        print('Failed to save announcement. Status code:
${response.statusCode}');
        print('Response body: ${response.data}');
      }
    } catch (error) {
      print('Error saving announcement: $error');
      if (error is DioError) {
        print('DioError: ${error.response?.data}');
```

4.3.20 Menampilkan data ijin

Fitur ini berfungsi mengambil data ijin dari server. *User* dari fitur ini adalah guru dan ortu. *User* dapat *login* ke aplikasi dan membuka fitur pengumuman. Dimana *user* dapat menambahkan atau menginputkan pengumuman baru, kemudian melakukan konfirmasi pengumuman.

Segmen Program 4. 20 Menampilkan data ijin

```
Widget _buildPermissionItem(Permissions permissionData) {
        return GestureDetector(
          onTap: () {
            showModalBottomSheet(
              context: context,
              backgroundColor: Colors.transparent,
```

```
            builder: (context) {
              return _buildDetailModal(permissionData);
            },
          );
        },
        child: Container(
          padding: EdgeInsets.symmetric(horizontal: 15.0,
vertical: 17.0),
          decoration: AppDecoration.outlineBlue900.copyWith(
            borderRadius: BorderRadiusStyle.roundedBorder8,
          ),
          child: Row(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Expanded(
                child: Column(
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: [
                    Text(
                      permissionData.studentName,
                      style:
CustomTextStyles.titleMediumPrimary,
                    ),
                    SizedBox(height: 13.0),
                    _buildTypeButton(permissionData),
                    SizedBox(height: 13.0),
                    Text(
                      "Tanggal izin :",
                      style: theme.textTheme.bodySmall,
                    ),
                    SizedBox(height: 13.0),
                    Text(
                      "${permissionData.startDate} -
${permissionData.endDate}",
                      style: theme.textTheme.bodySmall,
                    ),
                  ],
                ),
              ),
              if (!permissionData
                  .status) // Tombol delete hanya
ditampilkan jika status false (belum dikonfirmasi)

                GestureDetector(
                  onTap: () => _confirmDelete(context,
permissionData),
                  child: Icon(
                    Icons.delete_forever_rounded,
                    color: Colors.red,
                  ),
                ),
            ]),), );}
```

4.3.21 Update status data ijin

Fitur ini berfungsi melakukan update status data ijin. *User* dari fitur ini adalah guru. *User* dapat *login* ke aplikasi dan membuka fitur *permissions*. Dimana *user* dapat melakukan konfirmasi ijin orang tua.

Segmen Program 4. 21 *Update* status data ijin

```
updatePermissionStatus(String permissionID) async {
    final response = await dio.post(
      '$ip2/update_permission_status.php',
      data: {
        'Permission_ID': permissionID,
        'Status': 1,
      },
      options: Options(
        headers: {'X-API-KEY': apikey},
      ),
    );
    if (response.statusCode == 200) {
      print("Update response: ${response.data}");
    } else {
      print("Failed to update status");
    }
  }
```

4.3.22 Hapus status data ijin

Fitur ini berfungsi melakukan hapus data ijin. *User* dari fitur ini adalah ortu. *User* dapat *login* ke aplikasi dan membuka fitur permissions. Dimana *user* dapat menambahkan atau menghapus ijin.

Segmen Program 4. 22 Hapus data ijin

```
void _deleteFromDatabase(Permissions permissionData) async {
    try {
      final response = await dio.delete(
        '$ip2/delete_permission.php',
        data: {'Permission_ID': permissionData.permissionID},
        options: Options(headers: {'X-API-KEY': apikey}),
      );

      if (response.statusCode == 200) {
        setState(() {
          permissionList.removeWhere(
              (element) => element.permissionID ==
permissionData.permissionID);
        });
        print('Permission deleted successfully');
         await _fetchPermissionData();
```

```
      } else {
        print('Failed to delete Permission. Status code:
${response.statusCode}');
      }
    } catch (error) {
      print('Error deleting Permission: $error');
    }
  }
```

4.3.23 Menyimpan riwayat kesehatan

Fitur ini dirancang untuk memberikan kemudahan akses dan pemantauan terhadap riwayat kesehatan siswa. Orang tua dan guru dapat dengan mudah melihat informasi kesehatan sebelumnya dan perkembangan siswa. Guru dapat memasukkan data-data seperti berat badan, tinggi badan, lingkar kepala, dan kondisi kebersihan siswa.

Segmen Program 4. 23 Menyimpan riwayat Kesehatan

```
Future<void> _saveHealthRecord(BuildContext context) async {
     SharedPreferences prefs = await
     SharedPreferences.getInstance();
   String? jwtToken = prefs.getString('jwt');

   if (jwtToken == null) {
     ScaffoldMessenger.of(context).showSnackBar(
               SnackBar(content: Text("No JWT token found,
    please login again.")));
     return;
   }

   setState(() {
     _isLoading = true;
   });

   try {
     var dateFormat = DateFormat('yyyy-MM-dd');
     var date = dateFormat.format(selectedDate);
     var head = headSizeCmvalueController.text;
     var weight = weightKgvalueController.text;
     var height = heightCmvalueController.text;
     var cleanliness = selectedCleanliness;
     var student = selectedStudent;
     var noInduk = selectedNoInduk;

     if (!isValidClassID(selectedLevel)) {
       ScaffoldMessenger.of(context).showSnackBar(SnackBar(
         content: Text('Invalid class selected.'),
       ));
       if (mounted) {
         setState(() {
           _isLoading = false;
         });
```

```
      }
      return;
    }

    final queryParams = {
      'Date': Uri.encodeComponent(date),
      'Head': Uri.encodeComponent(head),
      'Weight': Uri.encodeComponent(weight),
      'Height': Uri.encodeComponent(height),
      'Class': Uri.encodeComponent(selectedLevel!),
      'Cleanliness': Uri.encodeComponent(cleanliness!),
      'Student_name': Uri.encodeComponent(student!),
      'No_Induk': Uri.encodeComponent(noInduk!),
    };

    final queryString =
              queryParams.entries.map((e) =>
     '${e.key}=${e.value}').join('&');

    final response = await dio.get(
      '$ip2/savedata_health.php?$queryString',
      options: Options(
        headers: {'Authorization': 'Bearer $jwtToken'},
      ),
    );

    if (response.statusCode == 200) {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => HomeFilledHealth(),
        ),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text(
                      'Failed to save health record. Status
     code: ${response.statusCode}')),
      );
    }
  } catch (error) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
          content: Text('An error occurred: $error'),
          backgroundColor: Colors.redAccent),
    );
  } finally {
    if (mounted) {
      setState(() {
        _isLoading = false;
      }); } } }
```

4.3.24 Mengedit riwayat kesehatan

Fitur ini dirancang untuk memberikan kemudahan akses dan pemantauan terhadap riwayat kesehatan siswa. Orang tua dan guru dapat dengan mudah melihat informasi kesehatan sebelumnya dan perkembangan siswa. Guru dapat memasukkan data-data seperti berat badan, tinggi badan, lingkar kepala, dan kondisi kebersihan siswa.

Segmen Program 4. 24 Mengedit riwayat kesehatan

```
  void saveData(BuildContext context) async {
      final data = {
        'Health_ID': widget.healthRecord.healthID,
        'Student_name': selectedStudent,
        'Class': selectedLevel,
        'Date': DateFormat('yyyy-MM-dd').format(selectedDate),
        'Weight': weightKgvalueController.text,
        'Height': heightCmvalueController.text,
        'Head': headSizeCmvalueController.text,
        'Cleanliness': selectedCleanliness,
      };

      final response = await dio.post(
        '$ip2/edit_health.php',
        data: data,
      );

      if (response.statusCode == 200) {
        Navigator.pop(context, true);
      } else {
        ScaffoldMessenger.of(context)
            .showSnackBar(SnackBar(content: Text('Error updating
  data')));
      }
    }

    bool isValidClassID(String? classID) {
      return classID != null && classID.isNotEmpty;
    }
```

4.3.25 Menghapus Data Kesehatan Anak

Fitur ini dirancang untuk menghapus riwayat kesehatan siswa. Orang tua dan guru dapat dengan mudah melihat informasi kesehatan sebelumnya dan perkembangan siswa. Guru dapat memasukkan data-data seperti berat badan, tinggi badan, lingkar kepala, dan kondisi kebersihan siswa.

Segmen Program 4. 25 Menghapus data kesehatan anak

```
Future<bool> _deleteHealthDataFromDatabase(HealthData
healthData) async {
    try {
      final response = await dio.delete(
        '$ip2/delete_health.php',
        data: {'Health_ID': healthData.healthID},
        options: Options(headers: {'X-API-KEY': apikey}),
      );

      if (response.statusCode == 200) {
        return true;
      } else {
print('Failed to delete data. Status code:
${response.statusCode}');
        return false;
      }
    } catch (error) {
      print('Error deleting data: $error');
      return false;
    }
  }
```

4.3.26 Menampilkan Data Kesehatan Anak

Fitur ini dirancang untuk menampilkan data kesehatan siswa. Orang tua dan guru dapat dengan mudah melihat informasi kesehatan sebelumnya dan perkembangan siswa.

Segmen Program 4. 26 Menampilkan data kesehatan anak

```
Widget _buildHealthItem(HealthData healthData) {
    return GestureDetector(
      onTap: () {
        showModalBottomSheet(
          context: context,
          backgroundColor: Colors.transparent,
          builder: (context) {
            return _buildDetailModal(healthData);
          },
        );
      },
      child: Container(
        width: 356.6,
        padding: EdgeInsets.symmetric(horizontal: 15.0,
vertical: 17.0),
        decoration: AppDecoration.outlineBlue900.copyWith(
          borderRadius: BorderRadiusStyle.roundedBorder8,
        ),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Expanded(
```

```
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    healthData.studentName,
                    style: CustomTextStyles.titleMediumPrimary,
                  ),
                  SizedBox(height: 13.0),
                  _buildCleanlinessButton(healthData),
                  SizedBox(height: 13.0),
                  Text(
                    healthData.date,
                    style:
theme.textTheme.bodySmall!.copyWith(height: 1.50),
                  ),
                  SizedBox(height: 12.0),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
```

4.3.27 Chat Live

User dari fitur ini adalah orang tua dan guru. *User* dapat *login* ke aplikasi dan masuk ke fitur *chat live*. Dibagian ini *user* dapat melakukan chat antara guru dan orang tua. Setiap ada chat yang diterima akan memberikan notifikasi pada masing-masing *user*.

Segmen Program 4. 27 Chat Live

```
      Widget _buildMessagesList() {
         return StreamBuilder<QuerySnapshot<Map<String,
     dynamic>>>(
           stream: firestoreDB
               .collection('messages')
               .doc('$noIndukAnak-213001')
               .collection('messages')
               .orderBy('timestamp', descending: true)
               .snapshots(),
           builder: (context, snapshot) {
             if (snapshot.connectionState ==
     ConnectionState.waiting) {
               return Center(child:
     CircularProgressIndicator());
             } else if (snapshot.hasError) {
               return Center(child: Text('Error:
     ${snapshot.error}'));
             } else if (snapshot.hasData &&
     snapshot.data!.docs.isNotEmpty) {
```

```
            final documents = snapshot.data!.docs;
            return ListView.builder(
              reverse: true,
              itemCount: documents.length,
              itemBuilder: (context, index) {
                final message = documents[index].data();
                final messageId =
                    documents[index].id; // Get document ID
from snapshot


      bool isSender = message['senderId'] ==
           (noInduk == '213001' ? '213001' : noIndukAnak);
              bool isRead = message['isRead'] as bool? ??
                  false;
              if (!isSender && !isRead) {
                markMessageAsRead(messageId);
              }
       DateTime messageTime = message['timestamp'] != null
            ? (message['timestamp'] as Timestamp).toDate()
                : DateTime.now();

         return ListTile(
           title: Align(
             alignment:
  isSender ? Alignment.centerRightAlignment.centerLeft,
                child: Container(
                  padding:
                      EdgeInsets.symmetric(horizontal:
12.0, vertical: 8.0),
                  decoration: BoxDecoration(
                    color: isSender ? appTheme.amber500
: Colors.grey[300],
                    borderRadius:
BorderRadius.circular(20),
                  ),
                  child: Column(
                    crossAxisAlignment: isSender
                        ? CrossAxisAlignment.end
                        : CrossAxisAlignment.start,
                    children: [
                      Text(message['text']),
                      SizedBox(height: 5),
                      Text(
                        DateFormat('dd MMM yyyy hh:mm
a').format(messageTime),
                        style:
                            TextStyle(fontSize: 12,
color: Colors.grey[600]),
                      ),
                    ],
                  ),
                ),
```

```
                    ),
                  );
                },
              );
            } else {
              return Center(child: Text('No messages found'))
               }
               },
          );
```