# 4. IMPLEMENTASI SISTEM

Pada bab ini menjelaskan mengenai implementasi sistem sesuai dengan desain pada bab sebelumnya. Implementasi sistem merupakan rancangan sistem berdasarkan desain yang dibuat di bab sebelumnya dan membahas mengenai *framework* dan bahasa pemrograman yang digunakan dalam membuat aplikasi ini.

## 4.1 *Framework* yang digunakan

*Framework* adalah sebuah kerangka kerja yang digunakan untuk mengembangkan aplikasi, dengan bantuan *framework* maka aplikasi tidak perlu dibuat dari awal. Pada tabel di bawah ini merupakan *framework* yang digunakan beserta versinya.

| No | *Framework* | Versi |
|---|---|---|
| 1 | @apollo/server | 4.10.0 |
| 2 | express | 4.18.2 |
| 3 | expo | 51.0.8 |
| 4 | react | 18.2.0 |
| 5 | react-native | 0.74.1 |

Tabel 4. 1 Framework yang digunakan beserta versinya

## 4.2 Login

*Login* diperlukan pengguna sebelum masuk ke aplikasi.

Segmen Program 4. 1 Login

```
const windowWidth = Dimensions.get("window").width;
const windowHeight = Dimensions.get("window").height;
const LOGIN_USER = gql`
  mutation Login($loginInput: LoginInput!) {
    login(loginInput: $loginInput) {
      access_token
    }
  }
`;
export default function Login({ navigation }) {
  const [email, setEmail] = useState("");
```

```
  const [password, setPassword] = useState("");
  const [isTyping, setIsTyping] = useState(false);
  const authContext = useContext(AuthContext);
  const [mutateFunction, { data, loading, error }] =
useMutation(LOGIN_USER, {
    onCompleted: (data) => {
      console.log(data);
    },
  });
  const handleLogin = async () => {
    try {
      const loginInput = { email, password };
      const result = await mutateFunction({ variables: {
loginInput } });
    } catch (error) {
      console.error(error);
    }
  };
  useEffect(() => {
    if (data?.login && !error) {
      save("accessToken", data.login.access_token).then(() =>
{
        authContext.setIsSignedIn(true);
      });
    }
  }, [data]);
  const onFocus = () => {
    setIsTyping(true);
  };
  const onBlur = () => {
    setIsTyping(false);
  };
  return (
    <SafeAreaProvider>
      <SafeAreaView style={styles.container}>
```

```
        <View
          style={{ flex: 5, justifyContent: "center",
alignItems: "center" }}
        >
          <Image style={styles.logo}
source={require("../assets/logo.png")} />
          <TextInput
            style={styles.input}
            onChangeText={setEmail}
            onFocus={onFocus}
            onBlur={onBlur}
            value={email}
            placeholder="Email"
          />
          <TextInput
            style={styles.input}
            onChangeText={setPassword}
            onFocus={onFocus}
            onBlur={onBlur}
            secureTextEntry
            value={password}
            placeholder="Password"
          />
          <TouchableOpacity
            style={styles.buttonLogin}
            onPress={() => {
              handleLogin();
            }}
          >
            <Text style={{ color: "white" }}>Log in</Text>
          </TouchableOpacity>
        </View>
        {isTyping ? null : (
          <View
```

```
            style={{ flex: 1, justifyContent: "center",
alignItems: "center" }}
          >
            <TouchableOpacity
              onPress={() => {
                navigation.navigate("Register");
              }}
            >
              <View style={styles.buttonCreateAccount}>
                <Text>Create new account</Text>
              </View>
            </TouchableOpacity>
          </View>
        )}
      </SafeAreaView>
    </SafeAreaProvider>
  );
}
```

**4.2 Registrasi**

Registrasi merupakan fitur yang digunakan oleh pengguna untuk melakukan registrasi akun, jadi pengguna yang belum memiliki akun tapi ingin menggunakan aplikasi wajib untuk registrasi sehingga pengguna dapat menggunakan aplikasi tukang cukur ini.

Segmen Program 4. 2 Registrasi

```
const REGISTER_USER = gql`
  mutation CreateUser($userInput: UserInput!) {
    createUser(userInput: $userInput) {
      _id
      name
      username
      email
      password
    }
  }
```

```
`;

const windowWidth = Dimensions.get("window").width;
const windowHeight = Dimensions.get("window").height;

export default function Register({ navigation }) {
  const [name, setname] = useState("");
  const [username, setusername] = useState("");
  const [email, setemail] = useState("");
  const [password, setpassword] = useState("");
  const [isTyping, setIsTyping] = useState(false);

  const [register, { data, loading, error }] =
useMutation(REGISTER_USER);

  const handleRegister = async () => {
    try {
      const userInput = { username, email, password, name };
      const result = await register({ variables: { userInput }
});
      navigation.navigate("Login");
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {}, [data]);

  const onFocus = () => {
    setIsTyping(true);
  };

  const onBlur = () => {
    setIsTyping(false);
  };
```

```
  return (
    <SafeAreaProvider>
      <SafeAreaView style={styles.container}>
        <View
          style={{ flex: 5, justifyContent: "center",
alignItems: "center" }}
        >
          <TextInput
            style={styles.input}
            onChangeText={setname}
            onFocus={onFocus}
            onBlur={onBlur}
            value={name}
            placeholder="Name"
          />
          <TextInput
            style={styles.input}
            onChangeText={setusername}
            onFocus={onFocus}
            onBlur={onBlur}
            value={username}
            placeholder="Username"
          />
          <TextInput
            style={styles.input}
            onChangeText={setemail}
            onFocus={onFocus}
            onBlur={onBlur}
            value={email}
            placeholder="Email"
          />
          <TextInput
            style={styles.input}
            onChangeText={setpassword}
```

```
            onFocus={onFocus}
            onBlur={onBlur}
            secureTextEntry
            value={password}
            placeholder="Password"
          />
          <TouchableOpacity
            style={styles.buttonRegister}
            onPress={() => {
              handleRegister();
            }}
          >
            <Text style={{ color: "white" }}>Register</Text>
          </TouchableOpacity>
        </View>
        {!isTyping && (
          <View
            style={{ flex: 1, justifyContent: "center",
alignItems: "center" }}
          >
            <TouchableOpacity
              style={styles.backLogin}
              onPress={() => {
                navigation.goBack();
              }}
            >
              <Text>Already have an account</Text>
            </TouchableOpacity>
          </View>
        )}
      </SafeAreaView>
    </SafeAreaProvider>
  );
}
```

### 4.3 Melihat Barbershop

Fitur melihat barbershop ini digunakan oleh pengguna untuk melihat barbershop mana saja yang pengguna ingin pilih sesuai dengan keinginan pengguna.

Segmen Program 4. 3 Melihat Barbershop

```
const GET_BARBERS = gql`
  query Query {
    getBarbershop {
      _id
      alamat
      image
      name
    }
  }
`;
export default function Booking({ navigation }) {
  const authContext = useContext(AuthContext);

  const { loading, error, data } = useQuery(GET_BARBERS);

  if (loading) return <Text>"Loading..."</Text>;
  if (error) return <Text>`Error! ${error.message}` </Text>;

  const renderItem = ({ item }) => (
    <CardBarber
      barberId={item._id}
      alamat={item.alamat}
      image={item.image}
      name={item.name}
      navigation={navigation}
    />
  );

  return (
    <SafeAreaProvider>
```

```
      <SafeAreaView style={styles.container}>
        <FlatList
          data={data.getBarbershop}
          renderItem={renderItem}
          keyExtractor={(item) => item._id}
          contentContainerStyle={{ paddingBottom: 16 }}
        />
      </SafeAreaView>
    </SafeAreaProvider>
  );
}
```

**4.4 Melihat Riwayat Barbershop**

Fitur melihat riwayat barbershop digunakan oleh pengguna untuk melihat riwayat barbershop yang pengguna gunakan sebelumnya untuk potong rambut.

Segmen Program 4. 4 Melihat Riwayat Barbershop

```
export const MY_HISTORY = gql`
  query GetHistory {
    getHistory {
      _id
      invoice
      name
      date
      price
      queue
      userID
      orderBy
      status
    }
  }
`;
const History = () => {
  const { loading, error, data } = useQuery(MY_HISTORY);
  const navigation = useNavigation();
```

```
  const handleItemPress = (item) => {
    // Navigate to the "Paid" page or any other page you
desire
    navigation.navigate("Paid", { invoiceId: item._id });
  };
  if (loading) {
    return (
      <View>
        <Text>Loading...</Text>
      </View>
    );
  }
  if (error) {
    return (
      <View>
        <Text>Error: {error.message}</Text>
      </View>
    );
  }
  const renderItem = ({ item }) => (
    <View style={styles.cardContainer}>
      <Text style={styles.cardText}>{item.invoice}</Text>
      <Text style={styles.cardText}>{item.name}</Text>
      <Text style={styles.cardText}>Order By:
{item.orderBy}</Text>
      <Text style={styles.cardText}>Antrian:
{item.queue}</Text>
      <Text style={styles.cardText}>Tanggal:
{item.date}</Text>
      <Text style={styles.cardText}>Price: {item.price}</Text>
     <Textstyle={styles.cardText}>Status: {item.status}</Text>
      <TouchableOpacity
        style={styles.button}
        onPress={() => handleItemPress(item)}
      >
```

```
        <Text style={styles.buttonText}>Go to Paid</Text>
      </TouchableOpacity>
    </View>
  );
  return (
    <View style={styles.container}>
      <Text style={styles.title}> History</Text>
      <FlatList
        data={data.getHistory}
        keyExtractor={(item) => item._id}
        renderItem={renderItem}
      />
    </View>
  );
};
```

**4.5 Daftar Antrian**

Fitur daftar antrian digunakan untuk melakukan pendaftaran barbershop yang pengguna ingin mendaftar untuk potong rambut.

Segmen Program 4. 5 Daftar Antrian

```
<TouchableOpacity
            style={styles.bookingButton}
            onPress={() => {
              updateQ();
              navigation.navigate("Invoice", {data:
data.getOne });
         }}>
     <Text style={styles.bookingButtonText}>Book Now</Text>
          </TouchableOpacity>
```

**4.6 Melihat Antrian**

Fitur melihat antrian digunakan oleh pengguna untuk melihat antrian yang masih berjalan di barbershop yang dipilih

Segmen Program 4. 6 Melihat Antrian

```
<Text style={styles.barberName}>{data?.getOne?.name}</Text>
<Text style={styles.queueText}>
        Queue:{data?.getOne?.queue}
</Text>)}
```

**4.7 Pembayaran**

Fitur pembayaran digunakan oleh pengguna untuk melakukan pembayaran setelah mendaftar di barbershop yang diinginkan.

Segmen Program 4. 7 Pembayaran

```
const { invoiceId, price, name } = route.params;
  const [url, setUrl] = useState('');
  const checkPayment = async () => {
    try {
      const { data } = await
axios.post("http://13.215.203.92:3000/payment", {
        invoiceId: invoiceId,
        price: price,
        name: name
      });
      console.log(data);
      setUrl(data.redirect_url);
      // if (true) {
      //   navigation.navigate("Success");
      // }
    } catch (error) {
      console.log(error, "<<<");
    }
  };
  useEffect(() =>{
    checkPayment();
```

```
    console.log(invoiceId, '<--invoiceId', price , '<--
price')
  }, [])

  const handleOnSuccess = (result) => {
    console.log(result, '<---')
    if(result.canGoBack && result.url.includes('backtohome'))
{
      navigation.navigate('Home')
    }
  }
//   const { url } = route.params;
  return (
    <View style={{ flex: 1, paddingBottom: 100 }}>
      <WebView
        source={{ uri: url || "" }}
        // onLoad={() => setLoading(false)}
        javaScriptEnabled={true}
        javaScriptCanOpenWindowsAutomatically={true}
        domStorageEnabled={true}
        cacheEnabled={true}
        allowFileAccessFromFileURLs={true}
        allowFileAccess={true}
        cacheMode="LOAD_NO_CACHE"
        onNavigationStateChange={handleOnSuccess}
      />
      {/* <Button title="check payment" onPress={checkPayment}
/> */}
    </View>
  );
}
```

**4.8 Logout**

Fitur logout merupakan fitur yang digunakan pengguna untuk keluar dari aplikasi.

Segmen Program 4. 8 Logout

```
const authContext = useContext(AuthContext);
  return (
    <>
      <Button
        title="Logout"
        onPress={() => {
          authContext.setIsSignedIn(false);
        }}
      ></Button>
    </>
```

**4.9 Login Admin**

Fitur login admin merupakan fitur yang digunakan oleh admin barbershop untuk masuk ke dalam website khusus admin.

Segmen Program 4. 9 Login Admin

```
function Login(){
    const navigate = useNavigate();
    let [loginForm, setLoginForm] = useState({
        email:"",
        password:""
    });
    async function submitHandler(e){
        try {
            e.preventDefault();
            // console.log(loginForm);
            const {data} = await axios.post('/user/login',
{email: loginForm.email, password: loginForm.password})
            // console.log(data, 'ini di login');
            localStorage.access_token = data.access_token;
            navigate('/');
        } catch (err) {
            // console.log(err);
            if(err.message === 'Network Error'){
                Swal.fire({
                    icon: "error",
                    title: "Oops...",
                    text: "Please check your network!",
                  }).then(function(result){
                    if(result.value){
                        window.location = '/login';
                    }
                  });;
            } else if(err.response.status === 400){
                Swal.fire({
                    icon: "error",
```

```
                    title: "Oops...",
                    text: `${err.response.data.message}`,
                  }).then(function(result){
                    if(result.value){
                        window.location = '/login';
                    }
                  });;
            }else if(err.response.status === 401){
                Swal.fire({
                    icon: "error",
                    title: "Oops...",
                    text: `${err.response.data.message}`,
                  }).then(function(result){
                    if(result.value){
                        window.location = '/login';
                    }
                  });;
            } else{
                Swal.fire({
                    icon: "error",
                    title: "Oops...",
                    text: "Internal server error!",
                  }).then(function(result){
                    if(result.value){
                        window.location = '/login';
                    }
                  });;
            }
        }
    }
```

**4.10 List Barbershop**

Fitur list barbershop untuk admin merupakan fitur yang digunakan oleh admin barbershop untuk melihat list barbershop yang tersedia.

Segmen Program 4. 10 List Barbershop

```
const barbershop = useSelector(state =>{
        return state.barbershop.list
    })
    const loading = useSelector(state =>{
        return state.barbershop.loading
    })
    const dispatch = useDispatch();


    useEffect(()=>{
```

```
        dispatch(fetchBarbershop());
    }, []);

    if(loading){
        return(
        <>
            <iframe src={loadingGif} width="480" height="480"
allowFullScreen></iframe>
            <h1>Loading...</h1>
        </>
        )
    }
    return(
        <>
        <Navbar/>
        <div className="container" style={{marginTop:
'100px'}}>
            <h1>List Barbershop</h1>
            <div className="container">
             <table className="table table-bordered">
                <thead>
                    <tr>
                        <th scope="col">Nama</th>
                        <th scope="col">Alamat</th>
                        <th scope="col">Gambar</th>
                        <th scope="col">Antrian</th>
</tr>
                </thead>
                <tbody>
                    {
                         barbershop.map(barbershop =>{
                            return <TableBarbershop
barbershop={barbershop}/>
                        })
                    }
```

```
                </tbody>
             </table>
            </div>
        </div>
        </>
    )

//TableBarbershop
<>
<tr>
                <th>{barbershop.name}</th>
                <td>{barbershop.alamat}</td>
                <td><img src={barbershop.image} style={{width:
'263px', height: '197px', objectFit:'fill'}} onError={event =>
{
                    event.target.src =
"https://cdn.vectorstock.com/i/preview-1x/65/30/default-image-
icon-missing-picture-page-vector-40546530.jpg"
                    event.onerror = null}}/></td>
                <td>{barbershop.queue}</td>
            </tr>
</>
```