BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Aplikasi pengingat perawatan sepeda motor merupakan media atau alat bantu yang dapat digunakan oleh para pengguna sepeda motor untuk memberikan penanda waktu bahwa jadwal perawatan sepeda motor telah tiba. Aplikasi ini juga digunakan sebagai pencatatan riwayat servis sehingga pengguna dapat terhindar dari tindak penipuan bengkel. Pengguna dapat membaca informasi seputar analisis perawatan dan konten edukasi yang disediakan oleh aplikasi. Melalui aplikasi ini, pengguna dapat memonitor perawatan lebih dari satu sepeda motor. Hasil penelitian perancangan antarmuka pada aplikasi ini dijabarkan pada penjelasan berikut.

4.2 IMPLEMENTASI DESAIN ANTARMUKA

Berikut merupakan beberapa implementasi desain antarmuka dari halaman aplikasi yang dikembangkan.

4.2.1 Halaman *On Boarding*

Halaman *on boarding* merupakan halaman pembuka yang muncul ketika aplikasi atau program pertama kali dijalankan. Pada tampilan ini, pengguna dapat memilih menu pendaftaran maupun login. Tampilannya dapat dilihat pada gambar 4.1.

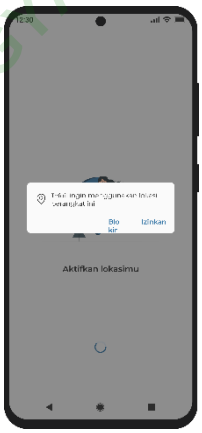
Gambar 4.1 Halaman *On Boarding*

Gambar 4.1 merupakan salah satu tampilan dari halaman *on boarding*. Adapun kode program yang digunakan yaitu sebagai berikut.

```
import 'package:get/get.dart';

class OnboardingController extends GetxController {
  final Rx<int> _currentOnboardingIndex = 0.obs;
  int get currentOnboardingIndex =>
_currentOnboardingIndex.value;

  void setCurrentOnboardingIndex(int index) {
    _currentOnboardingIndex.value = index;
  }
}
```

4.2.2 Halaman Perizinan Akses Lokasi

Halaman perizinan akses lokasi merupakan yang muncul sebelum pengguna memasuki beranda aplikasi. Pengguna dapat memilih opsi yang ada terkait perizianan lokasi. Tampilannya dapat dilihat pada gambar 4.2.



Gambar 4.2 Halaman Perizinan Lokasi

Gambar 4.2 merupakan halaman perizinan lokasi. Adapun kode program yang digunakan yaitu sebagai berikut.

```
import 'package:get/get.dart';
import 'package:permission_handler/permission_handler.dart';
```

```
class PermissionController extends GetxController {
  Future<bool> requestLocationPermission() async {
    final status = await Permission.location.request();
    if (status.isGranted) {
      return true;
    }
    return false;
  }
}
```

4.2.3 Halaman Registrasi

Halaman ini merupakan tampilan pertama setelah pengguna memilih *button* pendaftaran di halaman sebelumnya. Pengguna diwajibkan untuk mengisi nomor hp yang masih aktif. Halaman tersebut dapat dilihat pada gambar 4.3.

Gambar 4.3 Salah satu halaman registrasi

Gambar 4.3 merupakan salah satu tampilan dari halaman registrasi yang telah disebutkan sebelumnya. Adapun kode program yang digunakan yaitu sebagai berikut.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
```

```
class RegisterController extends GetxController {
  final PageController _pageController = PageController();
  PageController get pageController => _pageController;

  final int _currentPage = 0;
  int get currentPage => _currentPage;

  void previousPage() {
    pageController.previousPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeInOut,
    );
  }

  void nextPage() {
    pageController.nextPage(
      duration: const Duration(milliseconds: 300),
      curve: Curves.easeInOut,
    );
  }

  final Rx<TextEditingController> _phoneController =
      TextEditingController().obs;
  TextEditingController get phoneController =>
_phoneController.value;

  final Rx<TextEditingController> _otpController =
TextEditingController().obs;
  TextEditingController get otpController =>
_otpController.value;

  final FirebaseAuth _auth = FirebaseAuth.instance;
  FirebaseAuth get auth => _auth;

  final SupabaseClient _supabaseClient =
```

```
Supabase.instance.client;
  SupabaseClient get supabaseClient => _supabaseClient;

  final RxString _verificationId = ''.obs;
  String get verificationId => _verificationId.value;

  Future<void> verifyPhoneNumber(String phoneNumber) async {
    if (phoneNumber.isEmpty) {
      return;
    }
    if (!phoneNumber.startsWith('8')) {
      Get.snackbar(
        'Nomor HP tidak valid',
        'Nomor HP harus diawali dengan angka 8',
        snackPosition: SnackPosition.TOP,
      );
      return;
    }

    await auth.verifyPhoneNumber(
      phoneNumber: '+62$phoneNumber',
      verificationCompleted: (PhoneAuthCredential credential)
async {},
      verificationFailed: (FirebaseAuthException e) {
        debugPrint(e.message);
      },
      codeSent: (String verificationId, int? resendToken) async {
        debugPrint('Code send : $verificationId');
        _verificationId.value = verificationId;
      },
      codeAutoRetrievalTimeout: (String verificationId) {},
      timeout: const Duration(seconds: 60),
    );
  }

  Future<void> signInWithPhoneNumber(String smsCode) async {
```

```
    PhoneAuthCredential credential =
PhoneAuthProvider.credential(
      verificationId: verificationId,
      smsCode: '123456',
    );
    await auth.signInWithCredential(credential);
  }

  Future<void> signInWithEmailPasswordSupabase(
      String email, String password) async {
    await supabaseClient.auth.signInWithPassword(
      email: email,
      password: password,
    );
  }

  final Rx<TextEditingController> _userNameController =
      TextEditingController().obs;
  TextEditingController get userNameController =>
_userNameController.value;

  final Rx<TextEditingController> _emailController =
      TextEditingController().obs;
  TextEditingController get emailController =>
_emailController.value;

  final FirebaseFirestore _firebaseFirestore =
FirebaseFirestore.instance;
  FirebaseFirestore get firebaseFirestore => _firebaseFirestore;

  Future<void> setUserNameAndEmail(String name, String email)
async {
    await
firebaseFirestore.collection('users').doc(auth.currentUser!.uid).
set({
      'name': name,
```

```
        'email': email,
      });
    }
  }
```

4.2.4 Halaman Beranda

Halaman beranda merupakan tampilan yang pasti dijumpai ketika membuka aplikasi. Pada bagian ini terdapat fitur pengaturan, beralih motor, melihat riwayat, dan informasi umum lainnya. Tampilan halaman beranda disajikan pada gambar 4.4.

Gambar 4.4 Halaman Beranda

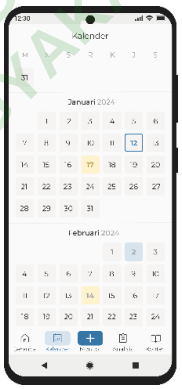
Gambar 4.4 merupakan tampilan dari halaman beranda. Adapun kode program yang digunakan yaitu sebagai berikut.

```
import 'package:get/get.dart';
import 'package:supabase_flutter/supabase_flutter.dart';

class HomeController extends GetxController {
  final Rx<double> _gaugeValue = 0.0.obs;
  double get gaugeValue => _gaugeValue.value;

  final SupabaseClient _client = Supabase.instance.client;
  SupabaseClient get client => _client;

  void setGaugeValue(double value) {
    _gaugeValue.value = value;
```

```
  }

  void fetchDataFromSupabase() async {
    final response = await _client.from('vehicle').select();
    print(response);
  }

  @override
  void onReady() async {
    fetchDataFromSupabase();
    super.onReady();
  }
}
```

4.2.5 Halaman Kalender

Halaman kalender memuat tanggal, bulan, dan tahun, serta penanda suatu tindakan yang dilakukan ketika perawatan motor. Tampilan halaman ini dapat dilihat pada gambar 4.5.



Gambar 4.5 Halaman Kalender

Gambar 4.5 menampilkan halaman kalender yang ada di aplikasi. Pada halaman kalender, terdapat penanda tindakan ketika melakukan perawatan sepeda motor dan keterangan riwayatnya. Adapun kode program halaman kalender yang digunakan yaitu sebagai berikut.

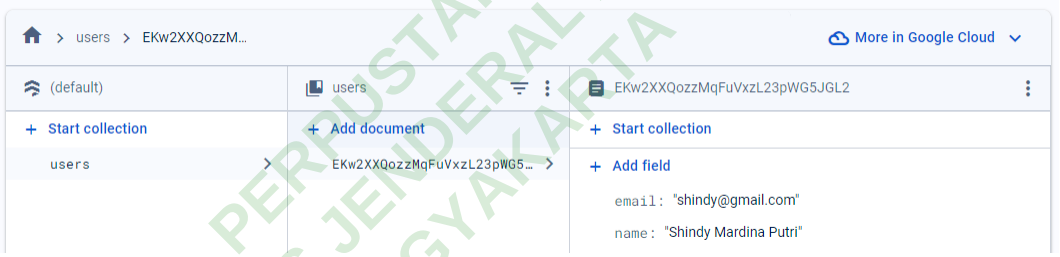
```
import 'package:get/get.dart';
import 'package:intl/intl.dart';
```

```
import 'package:supabase_flutter/supabase_flutter.dart';

class CalendarController extends GetxController {
  final SupabaseClient _client = Supabase.instance.client;
  SupabaseClient get client => _client;

  final RxString selectedDate = ''.obs;

  final RxList<Schedule> schedules = <Schedule>[].obs;

  @override
  void onReady() async {
    await fetchSchedule();
    super.onReady();
  }

  Future<void> fetchSchedule() async {
    final response = await _client
        .from('schedule')
        .select('*, users(uid)')
        .eq('users.uid', _client.auth.currentUser!.id);

    for (var element in response) {
      final Schedule schedule = Schedule.fromJson(element);
      schedules.add(schedule);
    }
    update(['calendar']);
  }
}

class Schedule {
  String type;
  String dateInString;

  Schedule({
    required this.type,
```

```
    required this.dateInString,
  });

  factory Schedule.fromJson(Map<String, dynamic> json) {
    DateTime date = DateTime.parse(json['date']);
    return Schedule(
      type: json['type'],
      dateInString: DateFormat('yyyy-MM-dd').format(date),
    );
  }
}
```

4.3 LAYANAN DAN BASIS DATA

Layanan Firebase Firestore yang digunakan untuk mengelola validasi pengguna dapat dilihat pada gambar 4.6.



Gambar 4.6 Validasi Pengguna

Gambar 4.6 menunjukkan bagaimana data dalam Firestore diorganisir dalam koleksi dan dokumen, serta bagaimana pengelolaan data tersebut melalui antarmuka Firestore di Google Cloud. Sedangkan tabel basis data Supabase yang digunakan dapat dilihat pada penyataan berikut.

4.3.1 Tabel *Vehicle*

Tabel *vehicle* yang digunakan untuk menyimpan data jarak kendaraan pada antarmuka aplikasi ini dapat dilihat pada gambar 4.7.

Name	Description	Data Type	Format
id	No description	bigint	int8
name	No description	text	text
distance_traveled	No description	bigint	int8
created_at	No description	timestamp with time zone	timestamptz

Gambar 4.7 Tabel *Vehicle*

Gambar 4.7 memperlihatkan struktur tabel yang terdiri dari kolom id, name, distance_traveled, dan created_at.

4.3.2 Tabel *Schedule*

Tabel *schedule* digunakan untuk menyimpan data riwayat servis yang dapat dilihat pada gambar 4.8.

Name	Description	Data Type	Format
user_id	No description	bigint	int8
id	No description	bigint	int8
type	No description	text	text
date	No description	date	date
created_at	No description	timestamp with time zone	timestamptz

Gambar 4.8 Tabel *Schedule*

Gambar 4.8 memperlihatkan struktur tabel dalam database, termasuk jenis data yang disimpan di setiap kolom dan format yang digunakan. Tabel tersebut terdiri dari kolom user_id, id, type, date, dan created_at.

4.3.3 Tabel *Users*

Tabel *users* digunakan untuk menyimpan data pengguna dapat dilihat pada gambar 4.9.

Name	Description	Data Type	Format
id	No description	bigint	int8
uid	No description	text	text
name	No description	text	text
created_at	No description	timestamp with time zone	timestamptz

Gambar 4.9 Tabel *User*

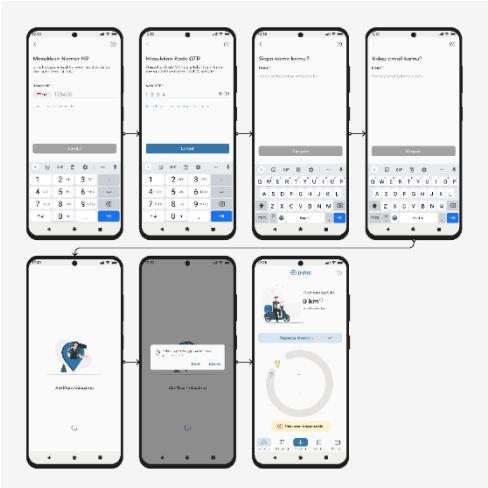
Gambar 4.9 memperlihatkan struktur tabel yang terdiri dari kolom id, uid, name, dan created_at. Selain itu, juga ditampilkan tipe data beserta formatnya.

4.4 FITUR-FITUR APLIKASI

Dalam antarmuka aplikasi ini, ada beberapa fitur yang dikembangkan selama penelitian berlangsung diantaranya daftar dan login akun, pengaturan dan pusat bantuan, beralih motor, kalender dan monitor, analisis, serta konten edukasi. Untuk lebih detailnya, dijabarkan pada penjelasan berikut.

4.4.1 Daftar dan Login Akun

Daftar dan login akun merupakan bagian yang wajib diisi pengguna ketika pertama kali membuka aplikasi. Bagian ini memberikan pengalaman pengguna yang dipersonalisasikan sesuai dengan keadaan dan sepeda motor yang dimilikinya. Fitur daftar dapat dilihat pada gambar 4.10.



Gambar 4.10 Fitur Daftar

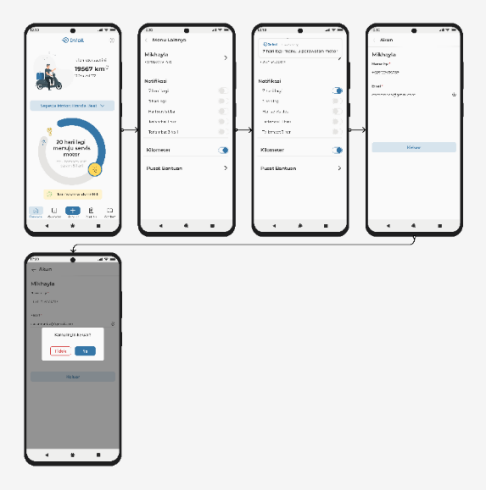
Gambar 4.10 memperlihatkan alur dan tampilan fitur daftar, dari awal hingga dapat masuk ke beranda aplikasi. Langkah pertama pengguna untuk mendaftar yaitu dengan mengisi nomor telepon yang aktif. Setelah itu, pengguna akan menerima kode OTP yang dapat secara otomatis terisi pada formulir. Pengguna juga perlu menuliskan nama dan email sebagai identitas akun. Aplikasi juga meminta izin untuk mengakses lokasi perangkat dan akhirnya pengguna dapat memasuki beranda aplikasi. Sedangkan untuk fitur login, dapat dilihat pada gambar 4.11.

Gambar 4.11 Fitur Login

Gambar 4.11 memperlihatkan alur dan tampilan fitur login, dari awal hingga dapat masuk ke beranda aplikasi. Fitur login dapat digunakan ketika pengguna telah memiliki akun pada aplikasi ini. Fitur ini terdiri dari 3 langkah, yaitu pengisian nomor telepon, verifikasi kode OTP, dan perizinan lokasi perangkat.

4.4.2 Pengaturan dan Pusat Bantuan

Fitur pengaturan berguna untuk custominasi beberapa kondisi sesuai dengan keinginan pengguna, sedangkan pusat bantuan memberikan jawaban dari masalah dasar yang terjadi ketika menggunakan aplikasi. Fitur pengaturan dapat dilihat pada gambar 4.10.



Gambar 4.12 Fitur Pengaturan

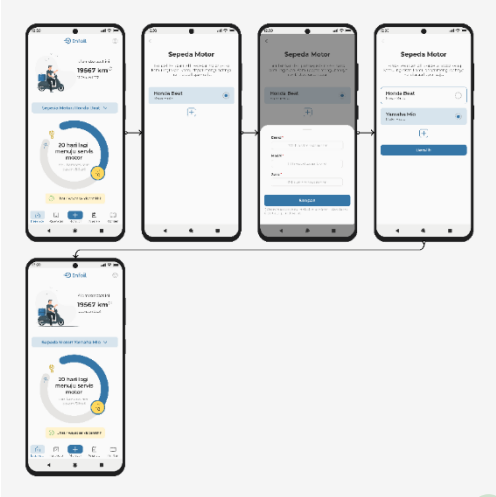
Gambar 4.12 memperlihatkan alur dan pengaturan yang berisi informasi akun, pengaturan notifikasi dan kilometer otomatis, serta pusat bantuan. Pengguna dapat mengaktifkan atau menonaktifkan notifikasi dan kilometer otomatis sesuai keinginan dengan men-*drag* tombol *switch*. Sedangkan pusat bantuan, dapat dilihat pada gambar 4.13.

Gambar 4.13 Pusat Bantuan

Gambar 4.13 memperlihatkan tampilan pusat bantuan yang berada di bagian terakhir pengaturan. Sebagai contoh pada gambar tersebut, apabila pengguna mengalami kendala saat melakukan login, maka penguna dapat memakai verifikasi email aktif untuk dapat masuk ke aplikasi.

4.4.3 Beralih Motor

Fitur ini berguna untuk mengganti informasi motor dengan yang lainnya apabila pengguna memiliki kendaraan lebih dari satu. Untuk detailnya dapat dilihat pada gambar 4.14.

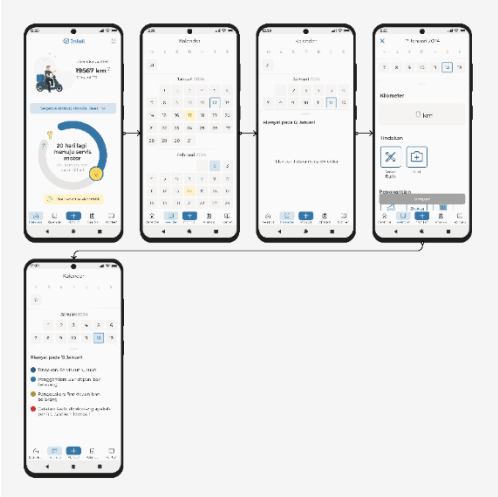


Gambar 4.14 Fitur Beralih Motor

Gambar 4.14 memperlihatkan tampilan alur untuk beralih motor. Dari beranda, pengguna dapat menekan *button* sepeda motor lalu dapat menambahkan sepeda motor yang baru. Setelah itu, pengguna dapat menekan pilihan motor yang diinginkan lalu menekan *button* beralih. Maka, informasi sepeda motor akan berganti sesuai dengan motor yang diinginkan.

4.4.4 Kalender dan Monitor

Fitur ini memuat hal-hal yang pernah dilakukan berdasarkan waktu dan terjadi selama perawatan sepeda motor. Untuk lebih jelasnya dapat dilihat pada gambar 4.15.



Gambar 4.15 Kalender dan Monitor

Gambar 4.15 memperlihatkan alur dan tampilan untuk melihat riwayat perawatan maupun menambahkannya. Pengguna dapat melihat riwayat melalui navigasi kalender, sedangkan untuk menambahkan riwayat, pengguna dapat menekan *button* monitor di bagian navigasi. Pada bagian monitor, pengguna dapat menuliskan *update* kilometer terakhir sebelum servis. Hal ini dikarenakan, kilometer pada bagian beranda dapat saja tidak akurat. Selain itu, pengguna juga dapat menambahkan foto kuitansi perawatan.

4.4.5 Analisis

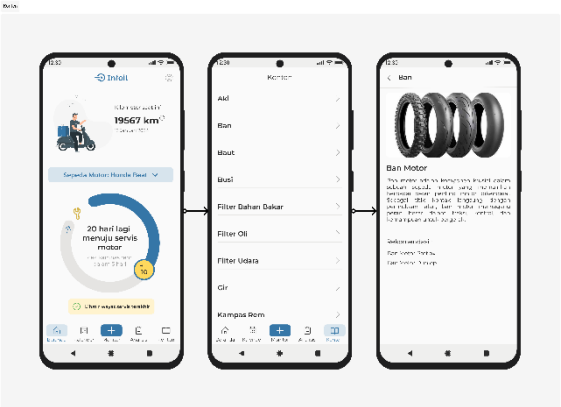
Fitur ini memuat akumulasi perawatan sepeda motor yang terekam pada aplikasi. Gambar 4.16 adalah fitur analisis yang dimaksud.

Gambar 4.16 Fitur Analisis

Gambar 4.16 memperlihatkan alur dan tampilan fitur analisis. Fitur ini akan menghitung dan memberikan informasi terkait siklus perawatan motor bulanan serta frekuensi pengecekan dan penggantian *sparepart*.

4.4.6 Konten Edukasi

Fitur ini berisi informasi dasar seputar sparepart dan rekomendasinya berdasarkan jenis dan *brand* sepeda motor. Contoh informasi tersebut dapat dilihat pada gambar 4.17.



Gambar 4.17 Fitur Konten Edukasi

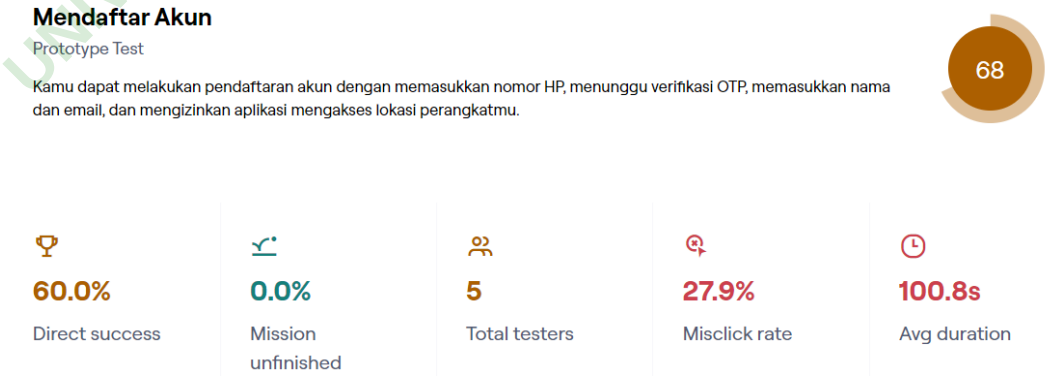
Gambar 4.17 memperlihatkan alur dan tampilan fitur konten edukasi. Pengguna dapat menemukannya melalui bilah navigasi.

4.5 USABILITY TESTING

Pengujian ini dilakukan dengan metode *remote usability testing*. Hasil dari *Usability Testing* (UT) antarmuka pengguna pada aplikasi pengingat perawatan sepeda motor dijabarkan pada penjelasan berikut.

4.5.1 *Single Ease Question* (SEQ)

Pengujian *Single Ease Question* (SEQ) pada skenario pendaftaran akun dan pengaktifan notifikasi dilakukan dengan menggunakan aplikasi Maze. Hasil pengujian skenario pendaftaran akun dapat dilihat pada gambar 4.18.



Gambar 4.18 Hasil SEQ Skenario Pendaftaran Akun

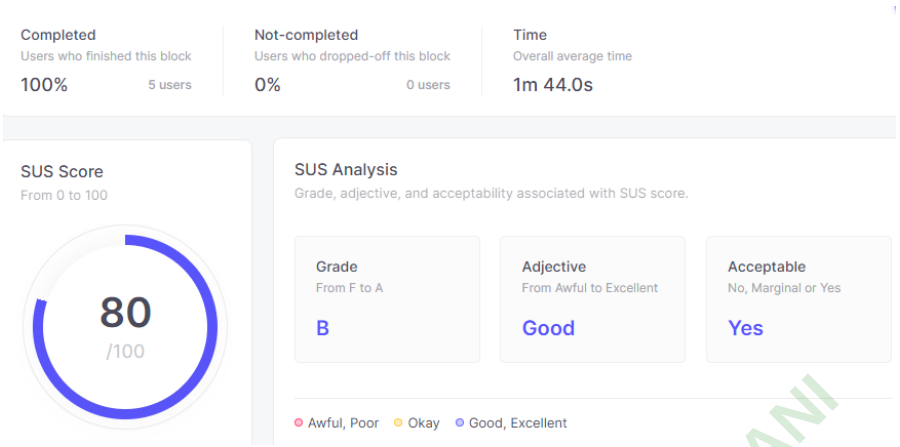
Gambar 4.18 menunjukkan bahwa *usability score* pada skenario pendaftaran akun yang didapatkan sebesar 68. Hasil tersebut dipengaruhi dengan pencapaian *success metric* sebesar 60% dengan durasi rata-rata selama 100,8 detik. Namun, tingkat kesalahan klik tergolong besar yaitu sebesar 27,9% yang dikarenakan pengguna memerlukan pembiasaan ketika pertama kali berinteraksi dengan aplikasi yang diuji. Pada bagian analisis pengoptimalan, ada 5 *screen* yang disarankan untuk ditinjau ulang. Sedangkan pengujian skenario pengaktifan notifikasi dapat dilihat pada gambar 4.19.

Gambar 4.19 Hasil SEQ Skenario Pengaktifan Notifikasi

Gambar 4.19 menunjukkan bahwa *usability score* pada pengaktifan notifikasi yang didapatkan sebesar 47. Hasil tersebut dipengaruhi dengan pencapaian *success metric* sebesar 40% dengan durasi rata-rata selama 46,7 detik. Namun, tingkat kesalahan klik tergolong besar yaitu sebesar 41,2% yang dikarenakan pengguna mengaktifkan notifikasi secara acak sedangkan skenario yang dibuat pada aplikasi penguji mengharuskan pengaktifan secara berurutan. Oleh karena itu, pada bagian analisis pengoptimalan, ada 1 *screen* yang disarankan untuk diperbaiki dan 2 *screen* yang disarankan untuk ditinjau ulang.

4.5.2 *System Usability Scale* (SUS)

Pengujian SUS pada skenario menambahkan riwayat servis menggunakan aplikasi Useberry. Hasil pengujian skenario penambahan riwayat servis dapat dilihat pada gambar 4.20.

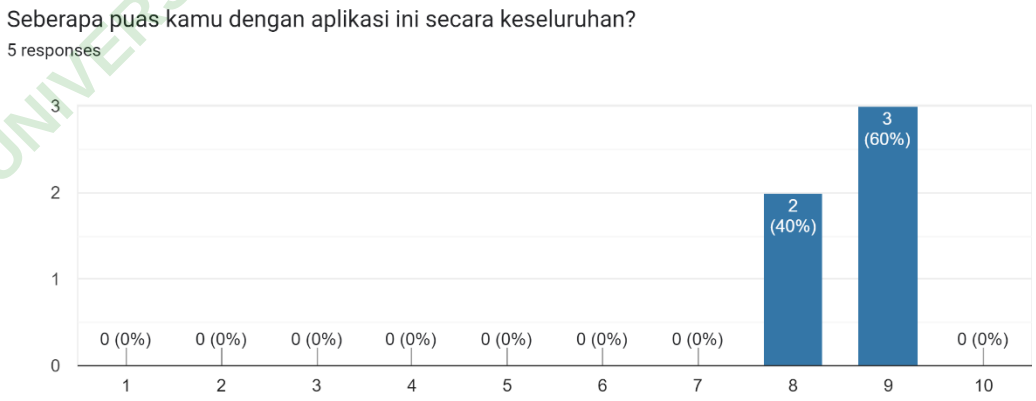


Gambar 4.20 Hasil SUS Skenario Penambahan Riwayat Servis

Gambar 4.20 menunjukan bahwa SUS *score* yang diperoleh sebesar 80/100 yang masuk pada peringkat "B" pada *Grade Scales*, mendapat predikat "*Good*" pada penilaian *Adjective Ratings*, serta masuk pada kategori *"Acceptable"* berdasarkan *Acceptables Ranges.*

4.5.3 *Feedback* Pengguna

Feedback pengguna yang diberikan meliputi bagian pertanyaan umum, penggunaan, kegunaan (*usability*), desain, serta kepuasan dan rekomendasi. Bagian ini didapatkan dari pengisian Google Form oleh 5 pengguna. Secara keseluruhan, grafik kepuasan pengguna dapat dilihat pada gambar 4.21.



Gambar 4.21 Grafik Kepuasan pengguna

Gambar 4.21 memperlihatkan kepuasan pengguna yang berada di angka 8 dan 9. Angka tersebut menunjukkan bahwa pengguna puas terhadap aplikasi yang dikembangkan. Seluruh pengguna pada pengujian antarmuka aplikasi ini merasa mudah menggunakan aplikasi karena tata letak yang nyaman dilihat dan tidak membingungkan. Selain itu, pengguna dengan mudah dapat menemukan informasi dan fitur yang dicari karena fitur mendetail, familiar, dan penggunaan Bahasa yang dapat dimengerti. Namun, 3 dari 5 pengguna memerlukan pembiasaan dalam menjelajahi antarmuka aplikasi ini.

Kesan pertama pengguna pada pengujian yang dilakukan yaitu tampilan yang menarik, natural, serta sangat terkejut karena berpikiran mengenai dampak positif yang diperoleh melalui aplikasi ini. Secara umum, pengguna juga merasa cocok dengan elemen desain yang dipakai. Disamping itu, pengguna juga memberikan beberapa rekomendasi, yaitu penambahan alamat bengkel yang tersambung ke aplikasi *maps*, pengoptimalan tampilan dan variasi sepeda motor, *tutorial* penggantian *sparepart*, serta sebuah aplikasi yang tidak membutuhkan banyak ruang penyimpanan jika diinstal.

4.6 PEMBAHASAN

Bagian ini mengacu pada penelitian yang telah selesai dilakukan sebelumnya. Pembahasan hasil penelitian lebih lanjut akan dijabarkan pada penjelasan berikut.

4.6.1 Rancangan Antarmuka Pengguna yang Optimal

Berdasarkan penelitian yang telah dilakukan, rancangan antarmuka pengguna pada aplikasi pengingat perawatan sepeda motor yang optimal dapat direalisasikan dengan meninjau hasil dari penelitian-penelitian sebelumnya serta terjun langsung ke lapangan untuk memahami pengguna secara mendalam termasuk kebutuhan, keinginan, dan tantangan yang dihadapi. Penelitian sebelumnya menggunakan metode *Extreme Programming*, sedangkan penelitian ini menggunakan metode *Design Thinking*. Dalam penelitian ini juga dilakukan pengujian untuk mengetahui apakah antarmuka aplikasi telah menjadi solusi bagi masalah yang dihadapi pengguna. Pengujian ini bermanfaat untuk mengevaluasi

antarmuka aplikasi sekaligus menjadi bahan pertimbangan pada pengembangan selanjutnya.

Hasil pengujian yang didapatkan bervariasi pada tiap skenario. *Usability Score* yang diperoleh pada SEQ skenario pendaftaran akun sebesar 68/100. Artinya skenario pendaftaran tersebut cukup baik diterima oleh pengguna. Sedangkan SEQ skenario skenario pengaktifan notifikasi memperoleh sebesar 47/100 yang berarti belum baik. Hasil tersebut didapatkan karena ketidakefektifan tombol *switch* yang digeser. Hal ini berbeda dengan nilai SUS skenario penambahan riwayat servis yang diperoleh sebesar 80/100. Artinya, skenario ini baik (*good*) diterima oleh pengguna. Namun, secara keseluruhan tingkat kepuasan pengguna terhadap aplikasi berada di angka 8 dan 9 dari rentang nilai 1-10. Angka ini diperoleh dari *feedback* 5 pengguna.

Jika dibandingkan dengan penelitian sebelumnya pada aplikasi "*MRA My Reminder App*", sudut pandang penelitian ini berbeda karena terfokus pada pengoptimalan antarmuka aplikasi. Pengujian yang dilakukan pun berbeda. Jika penelitian sebelumnya menggunakan pengujian *Black-Box* dari segi fungsionalitasnya, maka penelitian ini menggunakan metode *Remote Usability Testing* sebagai pengujian antarmuka aplikasinya. Pengujian ini menggunakan *platform* pengujian Useberry. Berdasarkan *platform* tersebut, hasil analisis antarmuka yang dikembangkan berada di peringkat "B" pada *Grade Scales*, mendapat predikat "*Good*" pada penilaian *Adjective Ratings*, serta masuk pada kategori *"Acceptable"* berdasarkan *Acceptables Ranges*. Sehingga ditemukan bahwa penggunaan metode *Design Thinking* dapat digunakan sebagai pelengkap metode lain untuk mengoptimalkan tampilan aplikasi.

4.6.2 Penerapan Metode Perancangan Antarmuka

Penelitian yang telah dilakukan menggunakan 2 metode yang berbeda, yaitu Scrum dan *Design Thinking*. Metode Scrum digunakan untuk meningkatkan produktivitas dan kualitas pengembangan produk dengan mengelola pekerjaan dalam iterasi pendek selama penelitian, sedangkan metode *Design Thinking* digunakan untuk memahami masalah secara mendalam, mengeksplorasi berbagai

solusi, dan menciptakan produk yang memenuhi kebutuhan pengguna dengan cara yang inovatif. Kedua metode tersebut juga disesuaikan lagi dengan kebutuhan pengembangan. Dalam penelitian ini, kolaborasi keduanya telah terbukti efektif dan efisien untuk mengembangkan antarmuka aplikasi yang sesuai dengan kebutuhan pengguna dalam waktu yang terbatas. Hal tersebut dibuktikan dengan tercapainya hasil penelitian dengan waktu 104 hari.

4.6.3 Hasil Produk Antarmuka Pengguna

Penelitian yang telah dilakukan menghasilkan produk antarmuka pengguna berupa *high-fidelity prototype* dan MVP (*Minimum Viable Product*). *High-fidelity prototype* merupakan purwarupa yang menyerupai produk akhir dalam hal tampilan, interaktivitas, dan fungsionalitas serta digunakan untuk menguji kesesuaian solusi dengan masalah yang ada. Purwarupa ini dirancang menggunakan aplikasi Figma. Pengguna dapat melihat dan mencoba rancangan awal antarmuka aplikasi lebih cepat jika dibandingkan dengan MVP (*Minimum Viable Product*). Namun, pengguna hanya dapat berinteraksi dengan tampilan sehingga tidak ada data yang tersimpan. Skenario yang dilalui pengguna juga terbatas karena pembuatan hanya difokuskan untuk pengujian antarmuka.

Sedangkan MVP (*Minimum Viable Product*) merupakan produk minimal yang sudah dapat dipakai. Produk awal ini dirancang menggunakan *framework* Flutter. Pengguna dapat berinteraksi dengan aplikasi yang dapat menyimpan data. Selain itu, pengguna dapat dengan bebas menjelajahi aplikasi yang tidak terbatas pada skenario di *high-fidelity prototype*. MVP ini menggunakan OTP (*One-Time Password*) sebagai autentifikasi pengguna yang dikirimkan melalui SMS (*Short Message Service*) ke nomor telepon yang diisikan. Teknologi lain yang dipakai yaitu pengaksesan lokasi perangkat dan notifikasi. Akan tetapi, beberapa spesifikasi tersebut belum berjalan sebagaimana mestinya dan hanya dapat dilihat antarmukanya sehingga aplikasi belum dapat di-*install* oleh pengguna.