Proses Pembuatan Aplikasi dan Konektifitas atara Firebase dan Flutter

1st Yusril Yudha Pratama
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
yusrilyudhap@student.telkomuniversity.ac.id

2nd Ahmad Sugiana
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sugianaa@telkomuniversity.ac.id

3rd Junarto Halomoan
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
junartho@telkomuniversity.ac.id

***Abstrak* — Persinyalan kereta api memainkan peran penting dalam memastikan keamanan perjalanan kereta. Namun, di Indonesia, pemantauan persinyalan masih memiliki keterbatasan, karena hanya dapat dilakukan dari ruang kontrol yang terhubung melalui Local Area Network (LAN). Untuk meningkatkan efisiensi pemantauan dan mempercepat respons teknisi dalam menangani gangguan sinyal, penelitian ini mengusulkan solusi menggunakan teknologi Internet of Things (IoT) yang terintegrasi dengan cloud. Aplikasi yang dikembangkan menggunakan Flutter dan Firebase Realtime Database memungkinkan data persinyalan ditampilkan secara real-time di perangkat seluler. Implementasi ini dilengkapi dengan fitur autentikasi pengguna, monitoring, dan logger untuk memudahkan debugging. Hasil pengujian menunjukkan bahwa aplikasi dapat menampilkan data persinyalan dengan akurat, mendukung berbagai orientasi layar, dan memberikan navigasi yang mudah bagi pengguna. Aplikasi ini diharapkan dapat meningkatkan efektivitas pemeliharaan dan perbaikan persinyalan kereta api di Indonesia.**

***Kata kunci*— persinyalan kereta api, Internet of Things, Flutter, Firebase, pemantauan sinyal, aplikasi mobile.**

I. PENDAHULUAN

Kondisi persinyalan kereta api dapat dipantau melalui *Centralized Traffic Control (CTC),* yaitu sebuah sistem monitoring dan kontrol persinyalan kereta api yang berasal dari Persinyalan kereta api merupakan suatu bentuk, warna, atau cahaya yang berfungsi memberikan isyarat kepada petugas perkeretaapian agar perjalanan kereta api berlangsung dengan aman [4]. Jenis persinyalan kereta api terbagi menjadi persinyalan elektrik dan mekanik. Persinyalan elektrik adalah peralatan yang memberikan isyarat berupa cahaya atau warna dengan arti tertentu, biasanya menggunakan lampu *LED (Light Emitting Diode)* [9]. Sebaliknya, sinyal mekanik adalah perangkat yang digerakkan secara mekanik, seperti papan atau lengan instruksi yang dinaikkan dan diturunkan untuk memberi perintah kepada masinis kereta api [5].

Amerika Utara. CTC memungkinkan pemantauan kondisi persinyalan apakah berjalan normal atau mengalami gangguan, yang diawasi oleh operator dan teknisi *maintenance* di ruang kontrol [1].

Di Indonesia, pemantauan persinyalan kereta api mengharuskan operator berada di ruang kontrol. Saat ini, tampilan CTC hanya dibagikan menggunakan sistem *Local Area Network* (LAN) yang memiliki batas jarak, sehingga pemantauan menjadi tidak efisien dan memerlukan waktu yang cukup lama untuk melakukan perbaikan jika terjadi kegagalan pada kondisi persinyalan [3].

Untuk mengatasi ketidakefisienan dalam pemantauan dan masalah waktu yang cukup lama dalam melakukan perbaikan ketika terjadi kegagalan, kami mengusulkan solusi dengan memanfaatkan teknologi *Internet of Things (IoT).* Data persinyalan yang ditampilkan di *SCADA* akan dikirimkan ke cloud, kemudian diteruskan ke smartphone, sehingga operator dan teknisi maintenance dapat memantau kondisi persinyalan tanpa harus berada di ruang kontrol. Pendekatan ini diharapkan dapat meningkatkan kecepatan respons teknisi maintenance dalam perbaikan dan pemeliharaan instrumen pada persinyalan kereta api [2].

II. KAJIAN TEORI

A. Flutter

Flutter adalah SDK yang dikembangkan Google untuk membuat *User Interface* (UI) aplikasi mobile Android, IOS, web, dan desktop dengan lisensi open source. Flutter menggunakan bahasa pemrograman Dart yang dikembangkan oleh Google untuk menggantikan Javascript. Dengan membuat UI di Flutter data yang tersimpan di Firebase Realtime Database dapat dipanggil ke Flutter agar dapat ditampilkan di smartphone dengan UI yang mudah dimengerti pengguna.[2]

B. Firebase

Firebase Realtime *Database* adalah basis data online yang dapat digunakan sebagai media penyimpanan data dari

aplikasi. Firebase memiliki beberapa fitur, diantaranya adalah:

1. Penyimpanan Cloud: *Realtime Database* disimpan secara *cloud*, memungkinkan akses data dari mana saja dengan koneksi internet.
2. *API Realtime*: Layanan ini menggunakan API yang memungkinkan data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung. Setiap perubahan pada data yang tersimpan akan diterima oleh semua user yang terhubung secara otomatis, memastikan bahwa data yang digunakan selalu *up-to-date*.
3. Platform NoSQL: Firebase adalah platform NoSQL yang berarti data tidak disimpan dalam tabel seperti dalam SQL. Oleh karena itu, diperlukan konversi data SQL ke JSON menggunakan skrip Python atau alat lain sebelum data dapat disimpan di Firebase.

Firebase *Authentication* adalah layanan yang menyediakan sistem autentikasi yang aman dan mudah digunakan untuk aplikasi Anda. Fitur-fitur utama dari Firebase *Authentication* meliputi:

4. Otentikasi Multi-Platform: Mendukung berbagai metode autentikasi seperti email/sandi, nomor telepon, dan penyedia identitas terintegrasi seperti Google, Facebook, dan Twitter.
5. Integrasi Mudah: Firebase Authentication mudah diintegrasikan ke dalam aplikasi, memungkinkan pengembang untuk menambahkan fitur login dan sign-up dengan cepat.
6. Keamanan: Menyediakan keamanan tinggi dengan perlindungan autentikasi yang kuat dan mudah diterapkan. Firebase juga mendukung autentikasi dua faktor untuk meningkatkan keamanan.
7. Pengelolaan Pengguna: Menyediakan API untuk mengelola data pengguna, seperti pembuatan, penghapusan, dan pembaruan akun, serta pengelolaan sesi pengguna.

Dengan menggunakan Firebase *Realtime Database* dan Firebase *Authentication*, pengembang dapat dengan mudah membangun aplikasi yang dapat menyimpan dan memanipulasi data secara realtime dan menyediakan sistem autentikasi yang aman dan andal untuk pengguna.

III. METODE

A. Cara Kerja *UI* dan Implementasi nya

1. Inisialisasi Firebase di FlutterMain Dart

Pada tahap ini, aplikasi diinisialisasi menggunakan Firebase sebagai *backend*, yang merupakan *platform* pengembangan aplikasi yang sangat populer. Proses dimulai dengan mengimpor paket-paket yang diperlukan, seperti firebase_core untuk integrasi Firebase dan flutter/material untuk membangun antarmuka pengguna. Inisialisasi Firebase dilakukan melalui file firebase_options.dart, yang berisi konfigurasi spesifik untuk platform yang digunakan.

Metode main bertindak sebagai titik awal aplikasi. Pertama, dilakukan pengikatan widget dengan memanggil WidgetsFlutterBinding.ensureInitialized(), kemudian Firebase diinisialisasi secara asinkron menggunakan Firebase.initializeApp(). Setelah Firebase berhasil diinisialisasi, aplikasi dijalankan dengan memanggil runApp dan menginisialisasi kelas MyApp.

Kelas MyApp merupakan turunan dari StatelessWidget, yang bertugas membangun antarmuka utama aplikasi. Dalam metode build, digunakan MaterialApp untuk mengonfigurasi aplikasi dengan berbagai properti seperti debugShowCheckedModeBanner, theme, dan home. debugShowCheckedModeBanner diatur ke false untuk menyembunyikan banner debug. Tema aplikasi diatur menggunakan ThemeData dengan primarySwatch warna cyan dan canvasColor transparan untuk memberikan tampilan yang bersih dan konsisten. Halaman utama aplikasi ditetapkan ke WelcomePage.

Dengan pendekatan ini, integrasi Firebase dengan Flutter dapat dilakukan secara efektif, memungkinkan aplikasi untuk memanfaatkan berbagai layanan yang disediakan oleh Firebase. Desain kode yang terstruktur dan penggunaan tema yang konsisten juga mendukung maintainability dan kualitas kode.

```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:tap_project/firebase_options.dart';
import 'package:tap_project/pages/pages.dart';
import 'custom_logger.dart'; // Import custom logger

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

  // Cetak log saat aplikasi dimulai
  CustomLogger.log("Aplikasi dimulai");

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    // Cetak log saat widget MyApp dibangun
    CustomLogger.log("Widget MyApp dibangun");

    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.cyan,
        canvasColor: Colors.transparent,
      ),
      home: const WelcomePage(),
    );
  }
}
```

Gambar 1
Source Code main.dart

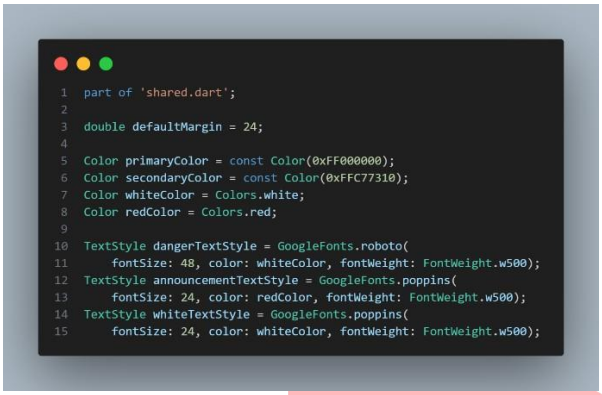
2. Desain Elemen Visual Utamatheme_share.dart

Pada tahap desain capstone project ini, elemen- elemen visual utama yang akan digunakan dalam aplikasi diatur. Dimulai dengan mendefinisikan defaultMargin sebesar 24, yang akan memberikan konsistensi dalam penempatan margin di seluruh antarmuka.

Selanjutnya, beberapa warna utama ditentukan untuk menjaga konsistensi visual aplikasi. primaryColor diatur sebagai warna hitam (0xFF000000), memberikan dasar yang kuat dan kontras. secondaryColor adalah warna oranye gelap (0xFFC77310), yang menambah kesan hangat dan energik. whiteColor dan redColor diambil dari pustaka warna Flutter sebagai warna putih dan merah standar.

Untuk menjaga konsistensi tipografi, pustaka Google Fonts digunakan. dangerTextStyle menggunakan font Roboto dengan ukuran 48, warna putih, dan berat font

sedang (w500), yang dirancang untuk menarik perhatian pada elemen penting seperti peringatan. announcementTextStyle dan whiteTextStyle menggunakan font Poppins dengan ukuran 24, masing- masing berwarna merah dan putih, serta berat font sedang. Gaya ini memberikan kesan modern dan bersih, cocok untuk pengumuman dan teks umum lainnya.



```dart
part of 'shared.dart';

double defaultMargin = 24;

Color primaryColor = const Color(0xFF000000);
Color secondaryColor = const Color(0xFFC77310);
Color whiteColor = Colors.white;
Color redColor = Colors.red;

TextStyle dangerTextStyle = GoogleFonts.roboto(
    fontSize: 48, color: whiteColor, fontWeight: FontWeight.w500);
TextStyle announcementTextStyle = GoogleFonts.poppins(
    fontSize: 24, color: redColor, fontWeight: FontWeight.w500);
TextStyle whiteTextStyle = GoogleFonts.poppins(
    fontSize: 24, color: whiteColor, fontWeight: FontWeight.w500);
```

GAMBAR 2
Source Code theme_shared.dart

3. Halaman Sambutan (Welcome Page)

welcome_page.dart

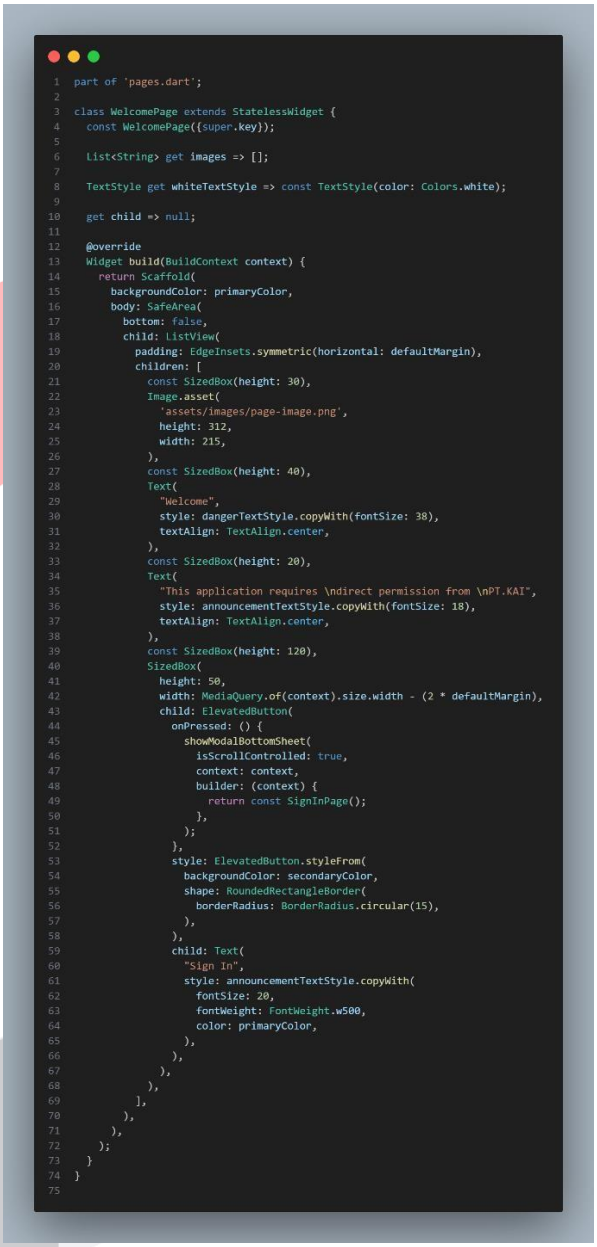
Pada implementasi kode berikut, didefinisikan sebuah kelas bernama WelcomePage yang merupakan turunan dari StatelessWidget dalam kerangka kerja Flutter. Kelas ini memiliki beberapa properti dan metode yang relevan untuk membangun antarmuka pengguna aplikasi.

Metode build di dalam kelas WelcomePage bertugas untuk membangun antarmuka pengguna. Metode ini mengembalikan sebuah Scaffold yang merupakan kerangka dasar untuk membuat visual layout halaman. Scaffold ini memiliki backgroundColor yang diatur ke primaryColor dan sebuah body yang dibungkus dalam SafeArea dengan properti bottom yang diset false agar menghindari area yang tidak aman di layar.

Di dalam SafeArea, terdapat sebuah ListView dengan padding horizontal yang diatur ke defaultMargin. ListView ini mengandung beberapa widget yang disusun secara vertikal, seperti SizedBox, Image.asset, Text, dan SizedBox lagi untuk memberikan jarak antar elemen. Image.asset menampilkan gambar dengan jalur 'assets/images/page-image.png', sementara dua widget Text digunakan untuk menampilkan teks "Welcome" dan "This application requires \ndirect permission from \nPT.KAI" dengan gaya teks yang ditentukan oleh dangerTextStyle dan announcementTextStyle.

Bagian bawah dari ListView ini diisi dengan sebuah SizedBox yang mengandung ElevatedButton. Tombol ini memiliki properti onPressed yang akan memanggil showModalBottomSheet ketika ditekan, untuk menampilkan halaman SignInPage secara modal. Tombol ini juga diatur dengan gaya latar belakang secondaryColor dan bentuk sudut melengkung menggunakan RoundedRectangleBorder. Teks di dalam tombol ini memiliki gaya teks yang diatur dengan announcementTextStyle dengan beberapa modifikasi seperti ukuran font, ketebalan font, dan warna teks yang diatur ke primaryColor.

Secara keseluruhan, kode ini memberikan gambaran tentang bagaimana sebuah halaman sambutan (welcome page) dibangun dalam aplikasi Flutter, dengan menggunakan berbagai widget dan gaya untuk menciptakan antarmuka yang estetis dan fungsional.



```dart
part of 'pages.dart';

class WelcomePage extends StatelessWidget {
  const WelcomePage({super.key});

  List<String> get images => [];

  TextStyle get whiteTextStyle => const TextStyle(color: Colors.white);

  get child => null;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: primaryColor,
      body: SafeArea(
        bottom: false,
        child: ListView(
          padding: EdgeInsets.symmetric(horizontal: defaultMargin),
          children: [
            const SizedBox(height: 30),
            Image.asset(
              'assets/images/page-image.png',
              height: 312,
              width: 215,
            ),
            const SizedBox(height: 40),
            Text(
              "Welcome",
              style: dangerTextStyle.copyWith(fontSize: 38),
              textAlign: TextAlign.center,
            ),
            const SizedBox(height: 20),
            Text(
              "This application requires \ndirect permission from \nPT.KAI",
              style: announcementTextStyle.copyWith(fontSize: 18),
              textAlign: TextAlign.center,
            ),
            const SizedBox(height: 120),
            SizedBox(
              height: 50,
              width: MediaQuery.of(context).size.width - (2 * defaultMargin),
              child: ElevatedButton(
                onPressed: () {
                  showModalBottomSheet(
                    isScrollControlled: true,
                    context: context,
                    builder: (context) {
                      return const SignInPage();
                    },
                  );
                },
                style: ElevatedButton.styleFrom(
                  backgroundColor: secondaryColor,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(15),
                  ),
                ),
                child: Text(
                  "Sign In",
                  style: announcementTextStyle.copyWith(
                    fontSize: 20,
                    fontWeight: FontWeight.w500,
                    color: primaryColor,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

GAMBAR 3
Source Code welcome_page.dart

4. Halaman Masuk (Sign In Page)

signin_page.dart

Implementasi kode berikut mendefinisikan kelas SignInPage yang merupakan turunan dari StatelessWidget. Metode build digunakan untuk membangun antarmuka pengguna untuk halaman masuk aplikasi. Dalam metode ini, dua objek TextEditingController digunakan untuk menangani input teks pada kolom email

dan password. Variabel checkMessageSignin digunakan untuk menampilkan pesan kesalahan jika proses masuk gagal.

Komponen utama halaman ini menggunakan StatefulBuilder, memungkinkan pengaturan state lokal seperti visibilitas teks password. Struktur halaman dibangun menggunakan Wrap yang berisi Container dengan BoxDecoration untuk memberikan warna latar belakang putih dan sudut melengkung.

Di dalam Container, SingleChildScrollView digunakan untuk memungkinkan scrolling saat konten melampaui tinggi layar. Konten diatur dalam Padding dan Container dengan margin horizontal. Bagian atas halaman menampilkan teks "Hello!!!" dan "Sign In" serta tombol tutup. Dua TextField disediakan untuk input email dan password, dengan StatefulBuilder mengatur visibilitasteks password.

Tombol Sign In digunakan untuk memproses masuk, dan jika berhasil, pengguna diarahkan ke halaman utama. Pesan kesalahan ditampilkan jika proses masuk gagal.

```dart
part of 'pages.dart';

class SignInPage extends StatelessWidget {
  const SignInPage({super.key});

  @override
  Widget build(BuildContext context) {
    var emailController = TextEditingController();
    var passwordController = TextEditingController();
    String checkMessageSignin = '';
    return StatefulBuilder(
      builder: (BuildContext context, StateSetter setState) {
        bool isPasswordVisible = false;

        return Wrap(
          children: [
            Container(
              color: Colors.transparent,
              child: Container(
                decoration: BoxDecoration(
                  color: whiteColor,
                  borderRadius: const BorderRadius.only(
                    topRight: Radius.circular(30),
                    topLeft: Radius.circular(30),
                  ),
                ),
                child: SingleChildScrollView(
                  child: Padding(
                    padding: EdgeInsets.only(
                      bottom: MediaQuery.of(context).viewInsets.bottom,
                    ),
                    child: Container(
                      margin: EdgeInsets.symmetric(horizontal: defaultMargin),
                      child: Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          const SizedBox(height: 25),
                          Row(
                            children: [
                              Column(
                                crossAxisAlignment: CrossAxisAlignment.start,
                                children: [
                                  Text(
                                    "Hello!!!",
                                    style: whiteTextStyle.copyWith(
                                      fontSize: 20,
                                      color: Colors.black,
                                    ),
                                  ),
                                  Text(
                                    "Sign In",
                                    style: whiteTextStyle.copyWith(
                                      fontWeight: FontWeight.bold,
                                      fontSize: 30,
                                      color: Colors.black,
                                    ),
                                  ),
                                ],
                              ),
                              const Spacer(),
                              InkWell(
                                onTap: () {
                                  Navigator.of(context).pop();
                                },
                                child: Image.asset(
                                  'assets/images/close.png',
                                  height: 30,
                                  width: 30,
                                ),
                              )
                            ],
                          ),
                          const SizedBox(height: 15),
                          TextField(
                            controller: emailController,
                            decoration: InputDecoration(
                              border: OutlineInputBorder(
                                borderRadius: BorderRadius.circular(10),
                              ),
                              hintText: "name@example.com",
                              labelText: "e-mail from company",
                            ),
                          ),
                          const SizedBox(height: 15),
                          StatefulBuilder(
                            builder: (context, setState) {
                              return TextField(
                                controller: passwordController,
                                obscureText: !isPasswordVisible,
                                decoration: InputDecoration(
                                  border: OutlineInputBorder(
                                    borderRadius: BorderRadius.circular(10),
                                  ),
                                  hintText: "password",
                                  labelText: "password",
                                  suffixIcon: IconButton(
                                    icon: Icon(
                                      isPasswordVisible
                                          ? Icons.visibility
                                          : Icons.visibility_off,
                                    ),
                                    onPressed: () {
                                      setState(() {
                                        isPasswordVisible = !isPasswordVisible;
                                      });
                                    },
                                  ),
                                ),
                              );
                            },
                          ),
```

GAMBAR 4
Source Code Signin_page.dart (1)

```
                    Container(
                      height: 30,
                      alignment: Alignment.centerRight,
                      child: checkMessageSignin == ''
                          ? const SizedBox()
                          : Text(
                              checkMessageSignin,
                              style: const TextStyle(
                                  color: Colors.red, fontSize: 12),
                            ),
                    ),
                    SizedBox(
                      width: 325,
                      height: 60,
                      child: ElevatedButton(
                        onPressed: () async {
                          if (emailController.text == '' &&
                              passwordController.text == '') {
                            setState(() {
                              checkMessageSignin = 'Please input data!';
                            });
                          } else {
                            try {
                              await FirebaseAuth.instance
                                  .signInWithEmailAndPassword(
                                      email: emailController.text,
                                      password: passwordController.text);
                              // ignore: use_build_context_synchronously
                              Navigator.pushAndRemoveUntil(
                                  // ignore: use_build_context_synchronously
                                  context,
                                  MaterialPageRoute(
                                    builder: (context) =>
                                        const HomePage(),
                                  ),
                                  (route) => false);
                            } on FirebaseAuthException catch (e) {
                              if (e.code == 'wrong-password') {
                                checkMessageSignin =
                                    'Wrong email & password. Try again';
                                setState(() {
                                  emailController.clear();
                                  passwordController.clear();
                                });
                              } else if (e.code == 'user-not-found') {
                                // ignore: use_build_context_synchronously
                                ScaffoldMessenger.of(context)
                                    .showSnackBar(SnackBar(
                                  content: const Text(
                                      'User not found. Try again'),
                                  backgroundColor: secondaryColor,
                                ));
                                checkMessageSignin =
                                    'User not found. Try again';
                                setState(() {
                                  emailController.clear();
                                  passwordController.clear();
                                });
                              } else {
                                checkMessageSignin =
                                    'Check your internet connection';
                              }
                            }
                          }
                        },
                        style: ElevatedButton.styleFrom(
                          backgroundColor: secondaryColor,
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(15),
                            side: const BorderSide(
                              color: Colors.white,
                              width: 3,
                            ),
                          ),
                        ),
                        child: Text(
                          "Sign In",
                          style: announcementTextStyle.copyWith(
                            fontSize: 20,
                            fontWeight: FontWeight.w500,
                            color: primaryColor,
                          ),
                        ),
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ],
        ),
      ],
    ),
  ],
);
  }
}
```

GAMBAR 5
Source Code Signin_page.dart (2)

5. Halaman Utama (Home Page)
home_page.dart

Kode berikut merupakan implementasi dari sebuah aplikasi Flutter yang terintegrasi dengan Firebase Realtime Database. Pada aplikasi ini, pengguna dapat melihat status gambar yang diambil dari database Firebase dan menampilkan informasi tersebut dalam format visual. Halaman utama dari aplikasi ini diatur untuk hanya dapat dilihat dalam orientasi landscape.

Untuk mencapai hal ini, pada fungsi initState, orientasi perangkat diatur menggunakan SystemChrome.setPreferredOrientations ke mode landscape. Selanjutnya, data status gambar diambil dari Firebase melalui _fetchImageStatusFromFirebase, yang mendengarkan perubahan data pada node 'DATA_CTC' dan memperbarui tampilan sesuai dengan data terbaru yang diterima.

Setiap gambar memiliki status yang dapat berupa normal, warning, atau station disable, yang ditampilkan dengan menggunakan filter warna pada gambar. Status ini dipetakan menggunakan warna berbeda berdasarkan kategori gambar, seperti SIG atau TRC. Pemetaan kunci Firebase ke path gambar dilakukan melalui map firebaseKeyToImagePath, memungkinkan aplikasi untuk menemukan path gambar yang sesuai dengan data yang diterima dari Firebase.

Aplikasi ini juga menyediakan fitur untuk menampilkan daftar pengguna yang login dalam bentuk dialog, beserta memungkinkan pengguna untuk logout, yang mengembalikan orientasi perangkat ke mode portrait dan mengarahkan pengguna kembali ke halaman WelcomePage. Melalui implementasi ini, aplikasi menyediakan cara interaktif untuk memantau dan menampilkan status berbagai kondisi komponen dalam sistem.

```
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:tap_project/pages/pages.dart';
import 'package:tap_project/shared/shared.dart';

// Contoh data login, ganti dengan data nyata jika ada
final List<String> loginList = [
  'yusrilyudhap@gmail.com',
  'anotherduyy@gmail.com',
];

// Gambar dan status awal yang tanpa perintah di ctc
final Map<String, int> imageStatus = {
  'assets/images/VAR_WKA_SIG_SRT1760_I_RSET.png': 0,
  'assets/images/VAR_WKA_SIG_SRT1789_I_RSET.png': 0,
  'assets/images/rel-120d.png': 0,
  'assets/images/tel-t.png': 0,
};
// Pemetaan key dari Firebase ke path gambar di Flutter
final Map<String, String> firebaseKeyToImagePath = {
  'VAR_WKA_SIG_SHT1764_I_RSET': 'assets/images/VAR_WKA_SIG_SHT1764_I_RSET.png',
  'VAR_WKA_SIG_SHT1785_I_RSET': 'assets/images/VAR_WKA_SIG_SHT1785_I_RSET.png',
  'VAR_WKA_SIG_SHT1762_I_RSET': 'assets/images/VAR_WKA_SIG_SHT1762_I_RSET.png',
  // Tambahkan pemetaan lain di sini
};

void main() {
  // Contoh penggunaan
  String key = 'VAR_WKA_SIG_SHT1787_I_RSET';
  String? imagePath = firebaseKeyToImagePath[key];

  if (kDebugMode) {
    print(imagePath);
  } // Output: assets/images/VAR.WKA.SIG_SHT1764.I_RSET.png
}

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

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

class _HomePageState extends State<HomePage> {
  String currentLoggedInUser =
      ''; // Variable untuk menyimpan email pengguna yang sedang login

  @override
  void initState() {
    super.initState();
    // Set the orientation to landscape
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.landscapeRight,
      DeviceOrientation.landscapeLeft,
    ]);
    _fetchImageStatusFromFirebase();
  }
```

GAMBAR 6
Code theme_shared.dart (I)

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

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

class _HomePageState extends State<HomePage> {
  String currentLoggedInUser =
      ''; // Variable untuk menyimpan email pengguna yang sedang login

  @override
  void initState() {
    super.initState();
    // Set the orientation to landscape
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.landscapeRight,
      DeviceOrientation.landscapeLeft,
    ]);
    _fetchImageStatusFromFirebase();
  }

  @override
  void dispose() {
    super.dispose();
  }

  void _fetchImageStatusFromFirebase() {
    DatabaseReference ref = FirebaseDatabase.instance.ref().child('DATA_CTC');
    ref.onValue.listen((event) {
      final data = event.snapshot.value as Map?;
      print("DATA: $data");

      if (data != null) {
        // print('Data received from Firebase: $data');
        setState(() {
          // Memperbarui status gambar berdasarkan data dari Firebase
          data.forEach((key, value) {
            String imagePath = firebaseKeyToImagePath[key] ?? '';

            print("key : $key | value : ${value['Value']}");

            if (imagePath.isNotEmpty) {
              // print("DIR :  $imagePath");
              print('Updating status for $imagePath to ${value['Value']}');
              imageStatus[imagePath] = value['Value'].toInt();
            }

            // else {
            //   print('Image path $imagePath not found in imageStatus map');
            // }
          });
          print(imageStatus.toString());
        });
      }
    });
  }

  String _getImagePathFromFirebaseKey(String firebaseKey) {
    // Menggunakan pemetaan key dari Firebase ke path gambar di Flutter
    return firebaseKeyToImagePath[firebaseKey] ?? 'assets/image/path.png';
  }

  void _showLoginList(BuildContext context) {
    // Menampilkan daftar login dalam sebuah dialog
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Login List'),
          content: SizedBox(
            width: double.maxFinite,
            child: ListView.builder(
              shrinkWrap: true,
              itemCount: loginList.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(loginList[index]),
                );
              },
            ),
          ),
        ),
```

GAMBAR 7
Code theme_shared.dart (II)

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

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

class _HomePageState extends State<HomePage> {
  String currentLoggedInUser =
      ''; // Variable untuk menyimpan email pengguna yang sedang login

  @override
  void initState() {
    super.initState();
    // Set the orientation to landscape
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.landscapeRight,
      DeviceOrientation.landscapeLeft,
    ]);
    _fetchImageStatusFromFirebase();
  }

  @override
  void dispose() {
    super.dispose();
  }

  void _fetchImageStatusFromFirebase() {
    DatabaseReference ref = FirebaseDatabase.instance.ref().child('DATA_CTC');
    ref.onValue.listen((event) {
      final data = event.snapshot.value as Map?;
      print("DATA: $data");

      if (data != null) {
        // print('Data received from Firebase: $data');
        setState(() {
          // Memperbarui status gambar berdasarkan data dari Firebase
          data.forEach((key, value) {
            String imagePath = firebaseKeyToImagePath[key] ?? '';

            print("key : $key | value : ${value['Value']}");

            if (imagePath.isNotEmpty) {
              // print("DIR :  $imagePath");
              print('Updating status for $imagePath to ${value['Value']}');
              imageStatus[imagePath] = value['Value'].toInt();
            }

            // else {
            //   print('Image path $imagePath not found in imageStatus map');
            // }
          });
          print(imageStatus.toString());
        });
      }
    });
  }

  String _getImagePathFromFirebaseKey(String firebaseKey) {
    // Menggunakan pemetaan key dari Firebase ke path gambar di Flutter
    return firebaseKeyToImagePath[firebaseKey] ?? 'assets/image/path.png';
  }

  void _showLoginList(BuildContext context) {
    // Menampilkan daftar login dalam sebuah dialog
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Login List'),
          content: SizedBox(
            width: double.maxFinite,
            child: ListView.builder(
              shrinkWrap: true,
              itemCount: loginList.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(loginList[index]),
                );
              },
            ),
          ),
        ),
```

GAMBAR 8
Code theme_shared.dart (III)

```
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text('Close'),
          ),
        ],
      );
    },
  );
}

void _logout(BuildContext context) {
  // Mengembalikan orientasi ke potret sebelum beralih ke WelcomePage
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
    DeviceOrientation.portraitDown,
  ]).then((_) {
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
        builder: (context) => const WelcomePage(),
      ),
    );
  });
}

Color _getImageColor(String imagePath, int status) {
  // Menentukan warna berdasarkan status gambar
  if (imagePath.contains('SIG')) {
    if (status == 0) {
      return Colors.transparent; // No color filter, original image color
    } else if (status == 1) {
      return Colors.green; // menyala (hijau)
    } else {
      return Colors.red; // ada kendala (merah)
    }
  } else {
    switch (status) {
      case 0:
        return Colors.grey; // mati
      case 1:
        return Colors.yellow; // menyala
      case 2:
        return Colors.red; // ada kendala
      default:
        return Colors.white; // default color
    }
  }
}
```

GAMBAR 9
Code theme_shared.dart (VI)

```
@override
 Widget build(BuildContext context) {
   return Scaffold(
     backgroundColor: Colors.black, // Set background color to black
     appBar: AppBar(
       backgroundColor: Colors.black,
       actions: [
         ElevatedButton(
           onPressed: () {
             // Set the current logged-in user (example using the first user in the list)
             setState(() {
               currentLoggedInUser = loginList.isNotEmpty ? loginList[0] : '';
             });
             _showLoginList(context);
           },
           style: ElevatedButton.styleFrom(
             backgroundColor:
                 secondaryColor, // Set button color to secondaryColor
           ),
           child: const Text(
             'View Login List',
             style: TextStyle(color: Colors.black), // Set text color to black
           ),
         ),
         IconButton(
           icon: const Icon(Icons.logout),
           color: Colors.white, // Set icon color to white
           onPressed: () {
             _logout(context);
           },
         ),
       ],
     ),
     body: Stack(
       children: [
         // tempat nemabahkan asset image untuk menampilkan komponen ctc aktif
         _buildPositionedImage(
             'assets/images/VAR_WKA_I_TRC_1789_RK.png', 0, 180, 55, 17, 0),
         _buildPositionedImage(
             'assets/images/VAR_WKA_I_TRC_1787_RK.png', 56, 180, 55, 17, 0),

         // tempat menambahkan input sebuah teks
         _buildPositionedText('2B', 529, 95),
       ],
     ),
   );
 }
```

GAMBAR 11
Code theme_shared.dart (VI)

```
void _showImageStatus(BuildContext context, String imagePath) {
    // Menampilkan status gambar dalam sebuah dialog
    int status = imageStatus[imagePath] ?? 1;
    String statusText;
    switch (status) {
      case 0:
        statusText = 'Mati (Abu)';
        break;
      case 1:
        statusText = 'Menyala (Kuning/Hijau)';
        break;
      case 2:
        statusText = 'Ada Kendala (Merah)';
        break;
      default:
        statusText = 'Tidak Diketahui';
    }

    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Image Status'),
          content: Text('Status: $statusText'),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('Close'),
            ),
          ],
        );
      },
    );
  }
```

GAMBAR 10
Code theme_shared.dart (V)

```
  Widget _buildPositionedImage(String imagePath, double left, double top,
      double width, double height, double rotationAngle) {
    int status =
        imageStatus[imagePath] ?? 1; // Default to 1 (menyala) if not found
    return Positioned(
      left: left,
      top: top,
      child: GestureDetector(
        onTap: () {
          _showImageStatus(context, imagePath);
        },
        child: Transform.rotate(
          angle: rotationAngle, // Sudut rotasi dalam radian
          child: ColorFiltered(
            colorFilter: ColorFilter.mode(
              _getImageColor(imagePath, status),
              BlendMode.srcATop,
            ),
            child: Image.asset(
              imagePath,
              width: width,
              height: height,
            ),
          ),
        ),
      ),
    );
  }

  Widget _buildPositionedText(String text, double left, double top) {
    return Positioned(
      left: left,
      top: top,
      child: Text(
        text,
        style: const TextStyle(
          color: Colors.white, // Set text color to white
          fontSize: 12, // Set the font size
        ),
      ),
    );
  }
}
```

GAMBAR 12
Code theme_shared.dart (VII)

6. Logger untuk Mempermudah Pengujian
custom_logger.dart
Kode berikut ini bertujuan untuk mengimplementasikan logger kustom yang akan menambahkan timestamp pada setiap log yang dihasilkan. Dengan menggunakan logger ini, kita dapat memastikan bahwa setiap pesan log dilengkapi dengan informasi waktu, yang sangat penting untuk debugging.

Informasi waktu yang di hasilkan hanya dapar di lihat pada Debug Console yang ada pada aplikasi flutter guna mempermudah pengujian saja.

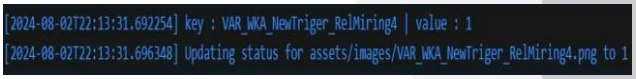
```dart
import 'package:logger/logger.dart';

class CustomLogger {
  static final Logger _logger = Logger(
    printer: PrettyPrinter(
      methodCount: 0, // Disable method count
      errorMethodCount: 5, // Set how many method calls to display on error
      lineLength: 120, // Set the line length
      colors: true, // Enable colors
      printEmojis: true, // Print emojis
      printTime: true, // Print timestamp
    ),
  );

  static void log(String message) {
    _logger.i(message); // Information log
  }

  static void error(String message) {
    _logger.e(message); // Error log
  }

  static void debug(String message) {
    _logger.d(message); // Debug log
  }
}
```

GAMBAR 13
Code custom_logger.dart

Pada file ini, kelas CustomLogger didefinisikan dengan metode statis untuk log informasi (logInfo), peringatan (logWarning), dan kesalahan (logError). Logger menggunakan PrettyPrinter dari paket logger untuk mencetak pesan log dengan format yang rapi dan menyertakan timestamp seperti contoh dari gambar di bawah ini.

```
[2024-08-02T22:13:03.934998] Updating status for assets/images/VAR_WKA_NewTriger_RelMiring4.png to 0.0
[2024-08-02T22:13:03.937325] key : VAR_WKA_NewTriger_RelMiring3 | value : 0.0
```

GAMBAR 14
Informasi pada Debug Console sebelum valueberubah



```
[2024-08-02T22:13:31.692254] key : VAR_WKA_NewTriger_RelMiring4 | value : 1
[2024-08-02T22:13:31.696348] Updating status for assets/images/VAR_WKA_NewTriger_RelMiring4.png to 1
```

GAMBAR 15
Informasi pada Debug Console sesudah valueberubah

7. Firebase Database
firebase_database.dart
Pada file ini, kode bertujuan untuk mengatur koneksi ke Firebase Realtime Database serta menyediakan metode untuk membaca dan menulis data dari/ke database.

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

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Firebase Data Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final DatabaseReference _databaseReference = FirebaseDatabase.instance.ref();

  Map<String, dynamic> _data = {};

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

  void _fetchData() async {
    try {
      DataSnapshot snapshot = (await _databaseReference.child('DATA_CTC').once()) as DataSnapshot;
      if (snapshot.value != null) {
        Map<dynamic, dynamic> data = snapshot.value as Map<dynamic, dynamic>;
        setState(() {
          _data = data.map((key, value) => MapEntry(key.toString(), value));
        });
      } else {
        print('Data tidak ditemukan');
      }
    } catch (e) {
      print('Error saat fetching data: $e');
    }
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Firebase Data'),
      ),
      body: _data.isNotEmpty
          ? ListView.builder(
              itemCount: _data.length,
              itemBuilder: (context, index) {
                String key = _data.keys.elementAt(index);
                dynamic value = _data[key]; // Tipe value dapat bervariasi
                return ListTile(
                  title: Text(key),
                  subtitle: Text(value.toString()),
                );
              },
            )
          : Center(child: CircularProgressIndicator()),
    );
  }
}
```

GAMBAR 16
Code firebase_database.dart

Kelas FirebaseDatabaseService ini menyediakan tiga metode utama:

1. getData: Mendapatkan data dari Firebase Realtime Database pada node tertentu.
2. setData: Menulis data ke Firebase Realtime Database pada node tertentu.
3. listenToDataChanges: Mendengarkan perubahan data pada node tertentu dan memanggil callback onDataChanged ketika ada perubahan.

8. Konfigurasi Firebase
firebase_options.dart
File ini digunakan untuk menyimpan konfigurasi spesifik untuk menginisialisasi Firebase di berbagai platform. Biasanya, file ini dihasilkan secara otomatis oleh Firebase CLI saat mengonfigurasi proyek.

```dart
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for ios - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyDFsShgYehry0KVfPzIr0qPt_tw6Twi9wA',
    appId: '1:536669292127:android:7d0402d9aad9efea767dea',
    messagingSenderId: '536669292127',
    projectId: 'tap-project-1a9ec',
    storageBucket: 'tap-project-1a9ec.appspot.com',
  );
}
```

GAMBAR 17
Code firebase_options.dart

Kelas DefaultFirebaseOptions berisi konfigurasi Firebase untuk proyek Anda. Properti currentPlatform mengembalikan FirebaseOptions yang diisi dengan informasi spesifik proyek seperti apiKey, authDomain, databaseURL, dan lain-lain. Konfigurasi ini diperlukan untuk menginisialisasi Firebase pada aplikasi Flutter.

B.Cara Pengujian sistem

Pengujian sistem dilakukan dengan langkah-langkah berikut:

1. Pendaftaran dan Akses Awal:

a. Verifikasi bahwa pengguna dapat mendaftarkan ID mereka ke pengelola server database.

b. Pastikan pengguna dapat mengakses aplikasi dengan memasukkan ID yang telah terdaftar.

2. Proses Sign-In:

a. Uji bahwa pengguna dapat berhasil melakukan sign-in dan dialihkan ke halaman sistem CTC.

b. Pastikan tampilan dan informasi di halaman CTC terlihat dengan jelas pada perangkat pengguna.

3. Interaksi dengan Komponen CTC:

a. Uji bahwa pengguna dapat menekan komponen yang ada di halaman CTC untuk melihat keterangan detail setiap komponen.

b. Verifikasi bahwa informasi yang ditampilkan sesuai dengan data yang ada didatabase.

4. Fungsi Daftar Pengguna:

a. Uji tombol "View login list" untuk memastikan pengguna dapat melihat daftar orang yang sedang mengakses aplikasi.

b. Pastikan daftar ini diperbarui secara real- time dan menampilkan informasi yang akurat.

5. Proses Logout:

a. Uji fungsi tombol keluar berlogo exit di pojok kanan atas.

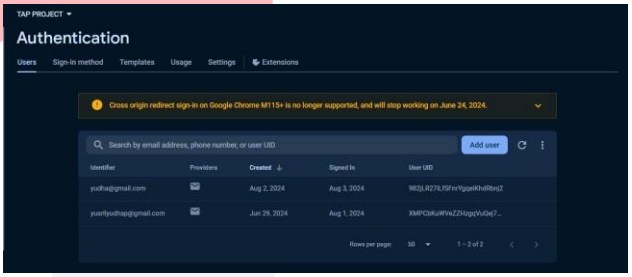
b. Pastikan pengguna dapat keluar dari aplikasi dengan lancar dan orientasi perangkat kembali ke mode awal setelah logout.

Pengujian dilakukan untuk memastikan bahwa setiap fitur aplikasi berfungsi dengan baik dan memenuhi kebutuhan pengguna.

IV. HASIL DAN PEMBAHASAN

A. Pengujian Aplikasi

Mencoba mendaftarkan akun melalui pengelolaserver database pada Firebase.



GAMBAR 18
gambar pengelola pendaftaran akun berhasil

Melakukan uji *UI* aplikasi untuk membuktikan semua berfungsi dengan baik.

TABEL 1
Pengujian fungsi *UI*

No.	Nama Pengujian	Deskripsi Pengujian	Hasil yang Diharapkan	Status
1	Pengujian WelcomePage	Memastikan WelcomePage ditampilkan dengan benar	WelcomePage muncul dengan teks dan tombol sesuai desain.	Lulus
2	Pengujian Tombol Sign In	Memastikan tombol Sign In berfungsi dengan benar	Halaman SignInPage ditampilkan.	Lulus
3	Pengujian Input Email	Memastikan input email pada SignInPage berfungsi dengan benar	Alamat email dimasukkan dengan benar dan tidak ada kesalahan input.	Lulus

4	Pengujian Input Password	Memastikan input password pada SignInPage berfungsi dengan benar	Kata sandi dimasukkan dengan benar dan tidak ada kesalahan input.	Lulus
5	Pengujian Sign In	Memastikan proses Sign In berjalan dengan benar	Pengguna berhasil masuk dan diarahkan ke HomePage.	Lulus
6	Pengujian Sign In Error	Memastikan pesan kesalahan muncul saat Sign In gagal	Pesan kesalahan ditampilkan.	Lulus
7	Pengujian HomePage	Memastikan HomePage ditampilkan dengan benar setelah Sign In	HomePage ditampilkan dengan data dari Firebase.	Lulus
8	Pengujian Data Firebase	Memastikan data dari Firebase ditampilkan dengan benar di HomePage	Data dari Firebase ditampilkan dengan benar di HomePage.	Lulus
9	Pengujian Logout	Memastikan proses logout berjalan dengan benar	Pengguna berhasil logout dan diarahkan kembali ke WelcomePage.	Lulus
10	Pengujian Orientasi	Memastikan aplikasi berjalan dengan baik dalam mode landscape dan portrait	Aplikasi berjalan dengan baik di kedua orientasi.	Lulus
11	Pengujian Logger	Memastikan setiap log memiliki timestamp	Setiap pesan log memiliki timestamp yang benar.	Lulus

B. Analisi Pengujian

Berdasarkan tabel pengujian yang disediakan, semua pengujian pada aplikasi telah berhasil lulus, menunjukkan efektivitas dan keandalan dari implementasi berbagai fitur dan fungsi. Di bawah ini adalah analisis terperinci berdasarkan hasil pengujian tersebut untuk disertakan dalam sebuah jurnal:

Analisis Hasil Pengujian Aplikasi

1. Pengujian WelcomePage

Pengujian ini berhasil memastikan bahwa halaman pembuka aplikasi, atau WelcomePage, muncul sesuai dengan spesifikasi desain, termasuk teks dan tombol yang diperlukan. Hal ini menunjukkan bahwa user interface pertama yang dihadapi pengguna telah diimplementasikan dengan tepat dan sesuai dengan persyaratan desain.

2. Pengujian Tombol Sign In

Tombol Sign In berhasil difungsikan dengan baik, mengarahkan pengguna ke halaman SignInPage. Keberhasilan ini penting untuk memastikan bahwa alur navigasi dalam aplikasi berjalan sesuai dengan rencana pengembangan.

3. Pengujian Input Email dan Password

Fitur input email dan password pada SignInPage bekerja dengan benar, tanpa adanya kesalahan input yang terdeteksi. Ini menunjukkan bahwa validasi form pada aplikasi telah diatur dengan baik, sehingga dapat menghandle input dari pengguna secara efisien.

4. Pengujian Sign In

Proses autentikasi pengguna (Sign In) terbukti berfungsi dengan baik, dengan pengguna yang berhasil masuk diarahkan ke HomePage. Ini menunjukkan bahwa mekanisme autentikasi dan sesi pengguna dalam aplikasi telah diimplementasikan dengan benar.

5. Pengujian Sign In Error

Sistem berhasil menampilkan pesan kesalahan ketika proses Sign In gagal. Keberhasilan ini adalah penting untuk memberikan feedback yang akurat kepada pengguna ketika terjadi kegagalan autentikasi.

6. Pengujian HomePage

Setelah pengguna berhasil masuk, HomePage dengan data yang diambil dari Firebase ditampilkan dengan benar. Hal ini menunjukkan bahwa integrasi aplikasi dengan backend database (Firebase) berfungsi secara efektif.

7. Pengujian Data Firebase

Data yang ditarik dari Firebase ditampilkan dengan benar di HomePage, memverifikasi bahwa aplikasi mampu mengolah dan menampilkan data dari cloud dengan tepat.

8. Pengujian Logout

Fungsi logout berhasil memutus sesi pengguna dan mengarahkan kembali ke WelcomePage, menunjukkan manajemen sesi yang baik dan keamanan aplikasi yang terjaga.

9. Pengujian Orientasi

Aplikasi berfungsi baik dalam orientasi landscape maupun portrait, memperlihatkan bahwa desain responsif telah berhasil diimplementasikan untuk mendukung berbagai jenis perangkat.

10. Pengujian Logger

Setiap log dalam aplikasi memiliki timestamp yang akurat, yang penting untuk keperluan debugging dan pemeliharaan sistem.

V. KESIMPULAN

Dari hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi telah berhasil memenuhi semua persyaratan fungsionalitas dan keandalan yang ditetapkan. Pengujian meliputi aspek navigasi antar halaman, input data, proses autentikasi, manajemen sesi, dan integrasi yang efektif dengan database *backend* seperti Firebase, yang semuanya menunjukkan kinerja yang sangat memuaskan. Hal ini menegaskan bahwa desain aplikasi, termasuk *user interface* dan *user experience*, telah dikembangkan dengan baik, mengutamakan kemudahan penggunaan bagi pengguna. Kegiatan pengujian juga menunjukkan bahwa aplikasi dapat beroperasi dengan baik pada berbagai orientasi layar, menunjukkan adaptasi yang baik terhadap berbagai perangkat. Berdasarkan keberhasilan ini, aplikasi siap untuk diluncurkan dalam lingkungan produksi dengan potensi pengembangan lebih lanjut yang fokus pada peningkatan keamanan dan skalabilitas, serta kemampuan untuk mengintegrasikan fitur baru dengan efisien di masa mendatang.

REFERENSI

[1] J. Alvert, Controlize Traffic Control, 1999.

[2] G. S. Chandra, “Pemanfaatan Flutter dan Electron Framework pada Aplikasi Inventori dan Pengaturan Pengiriman Barang,” 2020.

[3] S. Aisyah, “Prototipe Pemindah Wesel Kereta Menggunakan RFID,” 2017.

[4] L D. S. O. A. Darmawan, “Peningkatan Keamanan Perjalanan Kereta Api Dengan Penggunaan Sistem Axle Counter Dan Media Transmisi Fiber Optic Untuk Hubungan Blok Di Persinyalan Vpi (Studi Kasus Hubungan Blok Stasiun Surodadi – Pemalang),” 2017.

[5] E. Sayuri1, “Perancangan Sistem Persinyalan Elektrik Di Stasiun Berbasis PLC Omron,” 2017.