KERJA PRAKTIK - EF234801

Perancangan dan Implementasi Aplikasi myITS Notification di Institut Teknologi Sepuluh Nopember

Kampus Sukolilo, Gedung Pusat Riset Lantai 4, Jl. Teknik Kimia, Keputih, Sukolilo, Surabaya, Jawa Timur, 60117

Periode: 26 Februari 2024 – 26 Mei 2024

Oleh:

Jabalnur	5025201241
Sejati Bakti Raga	5025201007

Pembimbing Departemen
Bintang Nuralamsyah, S.Kom., M.Kom.,
Pembimbing Lapangan
Hadziq Fabroyir, S.Kom., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

KERJA PRAKTIK - EF234801

Perancangan dan Implementasi Aplikasi myITS Notification di Institut Teknologi Sepuluh Nopember

Kampus Sukolilo, Gedung Pusat Riset Lantai 4, Jl. Teknik Kimia, Keputih, Sukolilo, Surabaya, Jawa Timur, 60117

Periode: 26 Februari 2024 – 26 Mei 2024

Oleh:

Jabalnur	5025201241
Sejati Bakti Raga	5025201007

Pembimbing Departemen
Bintang Nuralamsyah, S.Kom., M.Kom.,
Pembimbing Lapangan
Hadziq Fabroyir, S.Kom., Ph.D.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Perancangan dan Implementasi Aplikasi myITS Notification di Institut Teknologi Sepuluh Nopember

Oleh:

Jabalnur	5025201241
Sejati Bakti Raga	5025201007

Disetujui oleh Pembimbing Kerja Praktik:

1. Bintang Nuralamsyah, S.Kom., M.Kom.,
   NIP. 198106202005011003

   (Pembimbing Departemen)

2. Hadziq Fabroyir, S.Kom., Ph.D.,
   NIP. 197007141997031002

   (Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perancangan dan Implementasi Aplikasi myITS Notification di Institut Teknologi Sepuluh Nopember

Nama Mahasiswa	: Jabalnur
NRP	: 5025201241
Nama Mahasiswa	: Sejati Bakti Raga
NRP	: 5025201007
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Bintang Nuralamsyah, S.Kom., M.Kom.,
Pembimbing Lapangan	: Hadziq Fabroyir, S.Kom., Ph.D.

ABSTRAK

myITS Notification adalah sistem pemberitahuan terpadu untuk mengirimkan notifikasi melalui berbagai aplikasi terintegrasi di Institut Teknologi Sepuluh Nopember (ITS). Menggunakan Firebase sebagai backend, aplikasi ini menggunakan Firebase Cloud Functions untuk mengelola notifikasi, Firebase Firestore sebagai basis data, dan Firebase Cloud Messaging (FCM) untuk mengirimkan notifikasi push ke pengguna.

Dibangun dengan Flutter, MyITS Notification menawarkan antarmuka pengguna yang responsif dan memudahkan pengelolaan notifikasi. Integrasi dengan aplikasi ITS memungkinkan penyebaran informasi yang efisien, meningkatkan komunikasi di dalam komunitas ITS.

Kata Kunci : Flutter, Push Notification, Android, Firebase

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Perancangan dan Implementasi Aplikasi myITS Notification di Institut Teknologi Sepuluh Nopember.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bapak Bintang Nuralamsyah, S.Kom., M.Kom., selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Bapak Hadziq Fabroyir, S.Kom., Ph.D selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 1 Juni 2024
Jabalnur dan Sejati Bakti Raga

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam era digital yang serba cepat, kebutuhan akan komunikasi yang efisien dan efektif di lingkungan pendidikan semakin mendesak. Institut Teknologi Sepuluh Nopember (ITS), sebagai salah satu institusi pendidikan terkemuka di Indonesia, menghadapi tantangan untuk menyampaikan informasi penting secara cepat kepada seluruh anggotanya. Saat ini, berbagai aplikasi yang digunakan di ITS, seperti sistem informasi akademik (myITS Akademik), portal mahasiswa, dan aplikasi administratif lainnya, sudah tergabung dalam *Single Sign-On* (SSO) yang sama. Meskipun demikian, fragmentasi informasi tetap menjadi masalah, karena pengguna harus memeriksa setiap aplikasi secara terpisah untuk mendapatkan pemberitahuan terbaru. Oleh karena itu, diperlukan sebuah sistem pemberitahuan terpadu yang dapat mengintegrasikan berbagai sumber informasi ini dan menyampaikan notifikasi secara *real-time* kepada pengguna melalui satu *platform* terpusat.

Pada saat Kuliah Praktik (KP), kami mendapatkan kesempatan untuk merancang sistem myITS Notification. Aplikasi tersebut dirancang untuk mengirimkan notifikasi kepada seluruh pengguna ITS melalui satu platform yang terpusat, memanfaatkan integrasi SSO yang sudah ada. Dengan menggunakan teknologi Firebase untuk backend dan Flutter untuk frontend, myITS Notification mampu menyediakan notifikasi push secara real-time yang dapat diterima di perangkat mobile pengguna. Firebase Cloud Functions digunakan untuk mengelola dan memproses notifikasi yang masuk dari berbagai aplikasi ITS, sedangkan Firebase Firestore berfungsi sebagai basis data

untuk menyimpan informasi notifikasi. Dengan integrasi ini, diharapkan komunikasi dan koordinasi di lingkungan ITS menjadi lebih efisien, sehingga informasi penting dapat disampaikan dengan tepat waktu dan mengurangi risiko terlewatnya informasi oleh pengguna.

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 4 SKS dan membantu Institut Teknologi Sepuluh Nopember (ITS) dalam meningkatkan efisiensi komunikasi dan penyampaian informasi melalui pengembangan aplikasi myITS Notification.

1.3. Manfaat

Manfaat yang diperoleh dengan adanya aplikasi myITS Notification adalah meningkatkan efisiensi penyampaian informasi penting kepada seluruh pengguna ITS, mengurangi fragmentasi informasi dengan mengintegrasikan notifikasi dari berbagai aplikasi ke dalam satu platform terpusat, dan meningkatkan koordinasi serta responsivitas dalam lingkungan ITS.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana arsitektur server yang dapat memberikan layanan aplikasi myITS Notification yang terintegrasi dengan berbagai aplikasi ITS?
2. Bagaimana rekayasa yang dapat dilakukan agar aplikasi myITS Notification memungkinkan DPTSI

ITS mengirimkan notifikasi ke pengguna sesuai dengan aplikasi-aplikasi yang diinginkan?

1.5. Lokasi dan Waktu Kerja Praktik

Pengerjaan kerja praktik ini lakukan secara *hybrid*, yaitu dari Kantor Subdirektorat Aplikasi dan Platform Digital Direktorat Pengembangan Teknologi dan Sistem Informasi ITS.

Adapun kerja praktik dimulai pada tanggal 26 Februari 2024 hingga 26 Mei 2024.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Untuk memahami kebutuhan aplikasi, kami mengikuti rapat bersama tim developer. Sebelum rapat, kami bertemu dengan Pak Hadziq dan Mas Zydhan dari DPTSI ITS, yang memiliki permintaan untuk pembuatan aplikasi myITS Notification. Pada saat rapat, dijelaskan konsep dan proses pemberitahuan terintegrasi yang diinginkan. Setelah penjelasan tersebut, pemimpin tim developer merumuskan fitur-fitur yang akan diterapkan pada aplikasi.

1.6.2. Studi Literatur

Setelah mendapatkan gambaran bagaimana sistem akan berjalan, kami diberi informasi tentang teknologi dan tinjauan yang akan diimplementasikan untuk membuat aplikasi beroperasi. Tinjauan ini meliputi Flutter untuk pengembangan frontend, Firebase Cloud Functions, Firebase Firestore, dan Firebase Cloud Messaging untuk backend. Selain itu, kami dijelaskan tentang aturan-aturan dalam menuliskan konfigurasi agar mudah dipahami oleh pengembang lain.

1.6.3. Analisis dan Perancangan Sistem

Setelah mempelajari tinjauan yang akan digunakan, kami merancang sistem dengan membuat desain arsitektur yang sesuai. Untuk aplikasi myITS Notification, tim developer sepakat menggunakan arsitektur berbasis serverless dengan Firebase sebagai backend dan Flutter sebagai frontend, memastikan integrasi yang lancar dengan sistem SSO ITS.

1.6.4. Implementasi Sistem

Implementasi adalah realisasi dari tahap perancangan. Pada tahap ini, kami melakukan pengembangan dan deployment aplikasi yang telah dirancang oleh tim developer. Kami mengintegrasikan berbagai fitur pemberitahuan dan memastikan aplikasi dapat menangani notifikasi dari berbagai aplikasi ITS.

1.6.5. Pengujian dan Evaluasi

Setelah aplikasi myITS Notification selesai dikembangkan, evaluasi dilakukan untuk menguji apakah aplikasi sesuai dengan harapan DPTSI ITS. Jika masih ada kekurangan atau perlu penambahan fitur, rapat akan diadakan kembali untuk mendiskusikan fitur-fitur yang perlu diperbaiki atau ditambahkan.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan menunjukkan bahwa aplikasi myITS Notification telah memenuhi syarat yang diinginkan dan berjalan dengan baik. Aplikasi ini dapat mengirimkan notifikasi secara efisien dan membantu meningkatkan komunikasi di lingkungan ITS.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum Direktorat Pengembangan Teknologi dan Sistem Informasi ITS mulai dari profil, hingga lokasi perusahaan.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil Direktorat Pengembangan Teknologi & Sistem Informasi ITS

Direktorat Pengembangan Teknologi & Sistem Informasi (DPTSI) Institut Teknologi Sepuluh Nopember (ITS) adalah unit kerja yang bertanggung jawab atas pengembangan, implementasi, dan pengelolaan teknologi informasi dan sistem informasi di lingkungan kampus ITS. Direktorat ini memiliki peran penting dalam mendukung kegiatan akademik, penelitian, dan administrasi melalui penyediaan infrastruktur TI yang andal dan layanan berbasis teknologi.

2.2. Lokasi

Kampus Sukolilo, Gedung Pusat Riset Lantai 4, Jl. Teknik Kimia, Keputih, Sukolilo, Surabaya, Jawa Timur 60117

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Pemrograman Mobile

Pemrograman mobile adalah proses pengembangan aplikasi perangkat lunak yang dirancang untuk dijalankan pada perangkat mobile seperti smartphone dan tablet. Ini mencakup pembuatan aplikasi untuk platform seperti Android dan iOS. Pemrograman mobile melibatkan berbagai tahapan seperti perancangan antarmuka pengguna (UI), pengalaman pengguna (UX), penulisan kode, pengujian, dan distribusi aplikasi melalui toko aplikasi seperti Google Play Store dan Apple App Store [1].

Pada platform Android, bahasa pemrograman utama yang digunakan adalah Java dan Kotlin. Sementara itu, pada platform iOS, bahasa pemrograman yang digunakan adalah Swift dan Objective-C. Dalam proses pengembangan aplikasi mobile, pengembang juga sering menggunakan berbagai framework dan alat bantu seperti Android Studio, Xcode, dan alat lintas platform seperti Flutter dan React Native untuk mempercepat proses pengembangan [2] [3].

3.2. Dart

Dart adalah bahasa pemrograman yang dikembangkan oleh Google, dirancang untuk pengembangan aplikasi web, server, desktop, dan mobile. Dart pertama kali diperkenalkan pada tahun 2011 dan telah berkembang menjadi bahasa yang kuat dengan performa tinggi dan sintaks yang mudah dipahami. Salah satu fitur utama Dart adalah kemampuannya untuk dikompilasi ke

dalam kode JavaScript, memungkinkan aplikasi yang ditulis dalam Dart untuk dijalankan di peramban web [4].

3.3. Flutter

Flutter adalah framework open-source yang dikembangkan oleh Google untuk membangun aplikasi mobile, web, dan desktop dari satu basis kode. Flutter pertama kali dirilis pada tahun 2017 dan telah menjadi salah satu alat utama dalam pengembangan aplikasi lintas platform. Menggunakan bahasa pemrograman Dart, Flutter menyediakan berbagai widget yang memungkinkan pengembang untuk menciptakan antarmuka pengguna yang menarik, responsif, dan berkinerja tinggi [5].

3.4. Firebase

Firebase adalah platform pengembangan aplikasi mobile dan web yang dikembangkan oleh Google. Firebase menawarkan berbagai layanan yang dirancang untuk membantu pengembang dalam membangun, mengelola, dan meningkatkan aplikasi mereka dengan lebih mudah dan cepat. Layanan Firebase meliputi database real-time, otentikasi pengguna, analitik, penyimpanan file, hosting, cloud messaging, dan banyak lagi [6].

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1. Analisis Sistem

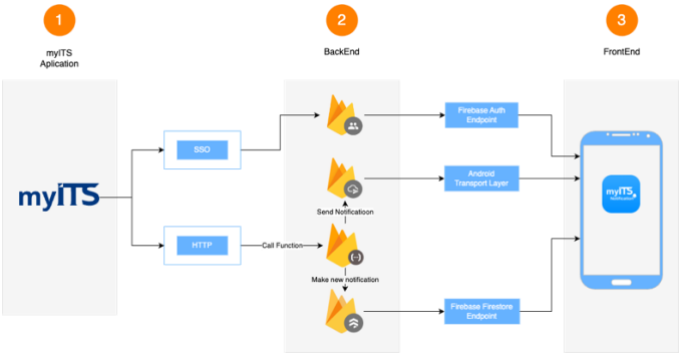
Pada bab ini akan dijelaskan mengenai tahapan dalam membangun infrastruktur aplikasi myITS Notification yaitu analisis dari infrastruktur sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan.

4.1.1. Definisi Umum Aplikasi

Secara umum, aplikasi myITS Notification adalah aplikasi yang dirancang untuk mengirimkan notifikasi kepada seluruh pengguna yang terhubung dengan berbagai aplikasi terintegrasi di Institut Teknologi Sepuluh Nopember (ITS). Aplikasi ini berfungsi sebagai pusat pemberitahuan terpadu, memungkinkan pengguna menerima notifikasi penting secara real-time dari berbagai sistem dan aplikasi yang digunakan dalam lingkungan ITS.

4.2. Perancangan Infrastruktur Sistem

Desain arsitektur pada aplikasi myITS Notification terbagi menjadi 3 poin utama yaitu, *FrontEnd*, *BackEnd*, dan Aplikasi ITS. Desain arsitekturnya dapat dilihat pada Gambar 4.1.



Gambar 4. 1. Desain Arsitektur Sistem

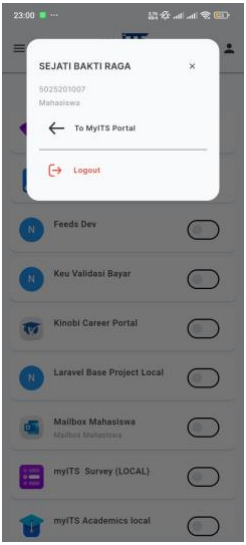
Gambar 4.1 menunjukkan bahwa aplikasi-aplikasi milik ITS akan memanggil API dari *BackEnd* aplikasi myITS Notification untuk menambahkan notifikasi baru. Kemudian notifikasi tersebut akan dikirimkan kepada user (*FrontEnd*) untuk dapat melihat notifikasi tersebut. User dapat menyimpan aplikasi mana saja yang akan diaktifkan notifkasinya.

4.2.1. *Front-End*

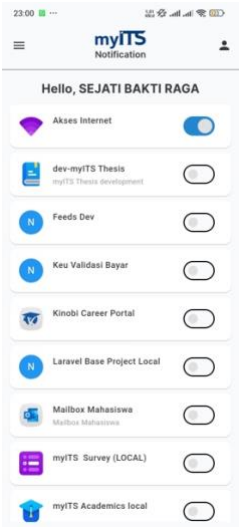
Frontend dari myITS Notification dibangun menggunakan Flutter, sebuah framework *open-source* yang memungkinkan pengembangan aplikasi mobile yang responsif dan berkualitas tinggi. Aplikasi Flutter ini terintegrasi dengan Firebase Cloud Messaging (FCM) untuk menerima notifikasi push secara langsung. Pengguna dapat mengakses notifikasi yang diterima melalui antarmuka pengguna yang intuitif, mengelola preferensi notifikasi, serta menandai notifikasi sebagai telah dibaca. Tampilan aplikasi myITS Notification dibangun menjadi 2 yaitu *light* dan *dark theme*. Adapun desain *light theme* dari *FrontEnd* dapat di lihat pada Gambar 4.2-4.8, sedangkan desain *dark theme* dari *FrontEnd* dapat di lihat pada Gambar 4.9-4.15.

Gambar 4. 2. Light Theme - SplashScreen

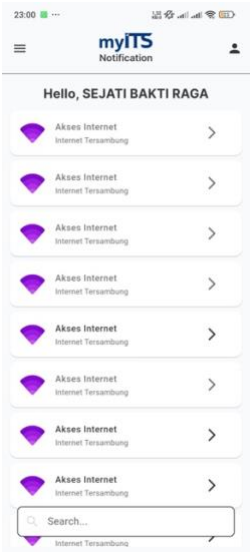
Gambar 4. 3. Light Theme - LoginScreen



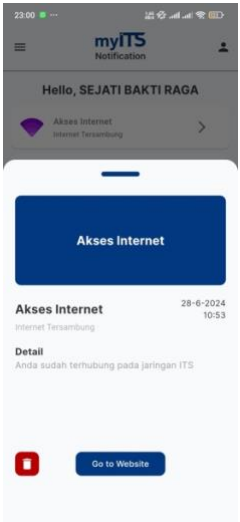
Gambar 4. 4. Light Theme - ProfileScreen



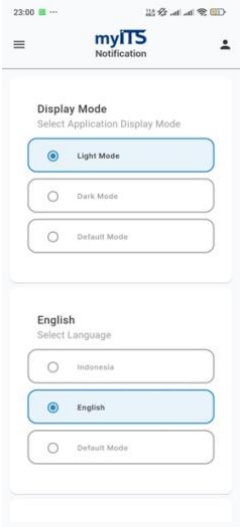
Gambar 4. 5. Light Theme – AppsScreen



Gambar 4. 6. Light Theme – NotifsScreen



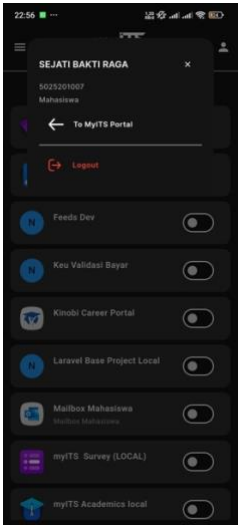
Gambar 4. 7. Light Theme - DetailNotifScreen



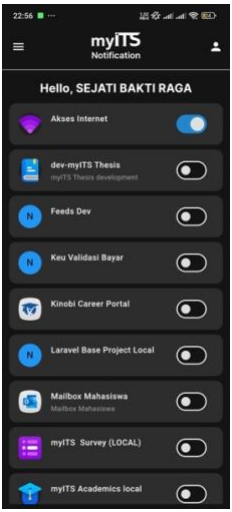
Gambar 4. 8. Light Theme - SettingsScreen

Gambar 4. 9. Dark Theme - SplashScreen

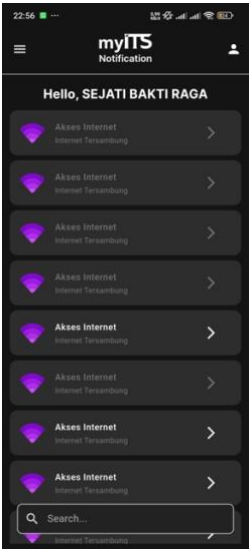
Gambar 4. 10. Dark Theme - LoginScreen



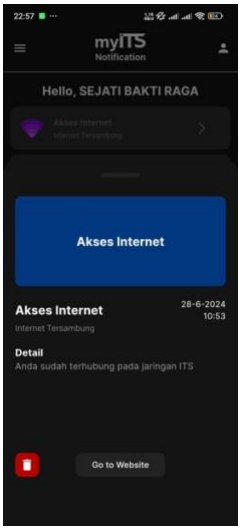
Gambar 4. 11. Dark Theme - ProfileScreen



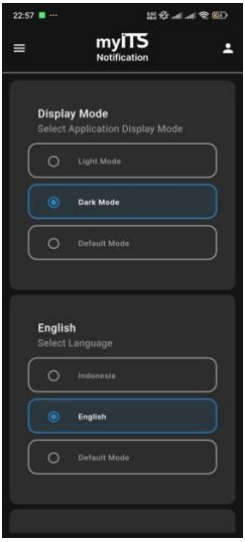
Gambar 4. 12. Dark Theme – AppsScreen



Gambar 4. 13.Dark Theme – NotifsScreen



Gambar 4. 14. Dark Theme - DetailNotifScreen



Gambar 4. 15. Dark Theme - SettingsScreen

4.2.2. *BackEnd*

Backend dari myITS Notification menggunakan Firebase sebagai infrastruktur utama. Firebase Cloud Functions berperan sebagai *serverless backend* yang menerima HTTP *request* dari aplikasi-aplikasi ITS. Fungsi ini memproses data notifikasi yang dikirimkan, menyimpan informasi tersebut ke dalam Firebase Firestore sebagai database NoSQL. Setelah data notifikasi tersimpan, Firebase Cloud Functions menggunakan Firebase Cloud Messaging (FCM) untuk mengirimkan notifikasi push secara real-time ke perangkat pengguna yang terkait.

4.2.3. Integrasi dengan Aplikasi-aplikasi ITS

Sistem myITS Notification berintegrasi dengan berbagai aplikasi ITS seperti sistem myITS Academic, myITS Presensi, dan sebagainya. Aplikasi-aplikasi ini dapat mengirimkan HTTP *request* ke *endpoint* yang di-*host* oleh Firebase Cloud Functions. Setelah menerima *request*, Firebase Cloud Functions memproses data notifikasi, menyimpannya di Firebase Firestore, dan mengirimkan notifikasi push ke perangkat pengguna melalui FCM. Integrasi ini memastikan bahwa pengguna dari berbagai aplikasi ITS dapat menerima notifikasi penting dengan cepat dan efisien.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang kami buat. Implementasi ini akan dibagi ke dalam dua bagian, yaitu bagian implementasi *FrontEnd* dan implementasi *BackEnd*.

5.1. Implementasi *FrontEnd*

Implementasi *frontend* berfokus pada pengembangan antarmuka pengguna menggunakan Flutter. Adapun rincian implementasi *frontend* sebagai berikut:

5.1.1. Authentication

Pada implementasi ini, kami menggunakan Firebase Authentication untuk mengelola autentikasi pengguna. Kode Sumber 5.1 menunjukkan konfigurasi Firebase Authentication dalam aplikasi Flutter.

```
u class AuthRemoteDataSourceImpl implements AuthRemoteDataSource {
  const AuthRemoteDataSourceImpl({
    required SharedPreferences sharedPreferences,
    required Dio dio,
    required API api,
    required FirebaseAuth firebaseAuth,
    required FirebaseMessaging firebaseMessaging,
    required FirebaseFirestore firebaseFirestore,
  })  : _sharedPreferences = sharedPreferences,
        _dio = dio,
        _api = api,
        _firebaseAuth = firebaseAuth,
        _firebaseMessaging = firebaseMessaging,
        _firebaseFirestore = firebaseFirestore;

  final SharedPreferences _sharedPreferences;
```

```
final Dio _dio;
final API _api;
final FirebaseAuth _firebaseAuth;
final FirebaseMessaging _firebaseMessaging;
final FirebaseFirestore _firebaseFirestore;

@override
Future<LocalUserModel> signIn() async {
  try {
    final provider = OAuthProvider(kProviderIDFirebase);

    provider.setScopes([
      'email',
      'group',
      'phone',
      'profile',
      'resource',
      'role',
      'openid',
    ]);

    final UserCredential result =
        await _firebaseAuth.signInWithProvider(provider);

    final accessToken = result.credential!.accessToken!;

    await _sharedPreferences.setString(kAccessToken, accessToken);

    final user = await getUserInfo(accessToken: accessToken);
    return LocalUserModel.fromMap(user);
  } on ServerException {
    rethrow;
  } on DioException catch (e) {
    throw ServerException(message: e.toString(), statusCode: e.type);
  } on FirebaseAuthException catch (e) {
    throw ServerException(message: e.toString(), statusCode: 505);
  } catch (e) {
    throw ServerException(message: e.toString(), statusCode: 505);
  }
}
```

```
@override
Future<void> signOut() async {
  try {

    final accessToken = _sharedPreferences.getString(kAccessToken);
    await _firebaseAuth.signOut();
    await _dio.get(
      _api.auth.logout,
      options: Options(
        headers: ApiHeaders.getHeaders(
          token: kAccessToken,
        ).headers,
      ),
    );
    final fcmtoken = await _firebaseMessaging.getToken();
    final email = _sharedPreferences.getString(kEmail);
    final userRef =
       _firebaseFirestore.collection('users').doc(email);
    final userDoc = await userRef.get();

    if (userDoc.exists) {
      final userDocData = userDoc.data();
      if (userDocData != null) {
        final fcmList = userDocData['fcmtoken'] as List<dynamic>;
        if (fcmList.contains(fcmtoken)) {
          fcmList.remove(fcmtoken);
          await userRef.update({
            'fcmtoken': fcmList,
          });
        }
      }
    }
    await _sharedPreferences.remove(kAccessToken);
    await _sharedPreferences.remove(kEmail);
    await _sharedPreferences.remove(kSub);
    await _sharedPreferences.remove(kRole);
  } on ServerException {
    rethrow;
  } on DioException catch (e) {
    throw ServerException(message: e.toString(), statusCode: e.type);
  } catch (e) {
```

```
      throw ServerException(message: e.toString(), statusCode: 505);
    }
  }

  @override
  Future<LocalUserModel> signInWithCredential() async {
    try {
      final accessToken = _sharedPreferences.getString(kAccessToken);
      final user = await getUserInfo(accessToken: accessToken);
      return LocalUserModel.fromMap(user);
    } on ServerException {
      rethrow;
    } on DioException catch (e) {
      throw ServerException(message: e.toString(), statusCode: e.type);
    } catch (e) {
      throw ServerException(message: e.toString(), statusCode: 505);
    }
  }

  @override
  Future<DataMap> getUserInfo({
    required String accessToken,
  }) async {
    try {
      final fcmtoken = await _firebaseMessaging.getToken();
      final userInfo = await _dio
          .get(
        _api.auth.userInfo,
        options: Options(
          headers: ApiHeaders.getHeaders(token: accessToken).headers,
        ),
      );
      DataMap user = {
        "nrp": userInfo.data['reg_id'],
        "name": userInfo.data['name'],
        "email": userInfo.data['email'],
        "role": userInfo.data['group'][1]['group_name'],
        "sub": userInfo.data['sub'],
      };
      final userRef =
          _firebaseFirestore.collection('users').doc(user['sub']);
```

```
      final userDoc = await userRef.get();
      if (userDoc.exists) {
        final userDocData = userDoc.data();
        if (userDocData != null) {
          final fcmList = userDocData['fcmtoken'];
          if (!fcmList.contains(fcmtoken)) {
            fcmList.add(fcmtoken);
            await userRef.update({
              'fcmtoken': fcmList,
            });
          }
        }
      } else {
        await userRef.set({
          'nrp': user['nrp'],
          'name': user['name'],
          'email': user['email'],
          'role': user['role'],
          'fcmtoken': [fcmtoken],
          'sub': user['sub'],
        });
      }

      await _sharedPreferences.setString(kEmail, user['email']);
      await _sharedPreferences.setString(kSub, user['sub']);
      await _sharedPreferences.setString(
          kRole, (user['role'] as String? ?? '').toLowerCase());

      return user;
    } on DioException catch (e) {
      throw ServerException(message: e.toString(), statusCode: e.type);
    } catch (e) {
      throw ServerException(message: e.toString(), statusCode: 505);
    }
  }
}
```

Kode Sumber 5. 1. Konfigurasi Firebase Authentication

5.1.2. Notifications

Bagian ini menjelaskan implementasi penerimaan notifikasi push menggunakan Firebase Cloud Messaging (FCM) di aplikasi Flutter. Konfigurasi penerimaan notifikasi dapat dilihat pada Kode Sumber 5.2.

```
class FirebaseAPIImpl implements FirebaseAPI {
  const FirebaseAPIImpl({
    required FirebaseMessaging firebaseMessaging,
  }) : _firebaseMessaging = firebaseMessaging;

  final FirebaseMessaging _firebaseMessaging;

  @override
  Future<void> initNotification() async {
    await _firebaseMessaging.requestPermission(
      alert: true,
      announcement: true,
      badge: true,
      carPlay: false,
      criticalAlert: true,
      provisional: false,
      sound: true,
    );
    await getToken();
    initPushNotification();
  }

  static void handleMessages({required RemoteMessage? message}) {
    if (message == null) return;
    CoreUtils.showNotification(message);
  }

  @override
  Future<void> initPushNotification() async {
    _firebaseMessaging.getInitialMessage().then((message) {
      handleMessages(message: message);
    });
```

```
    FirebaseMessaging.onMessageOpenedApp.listen((message) {
      handleMessages(message: message);
    });


    FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundH
    andler);

    FirebaseMessaging.onMessage.listen((message) {
      handleMessages(message: message);
    });

    _firebaseMessaging.onTokenRefresh.listen((token) {
      // handle token refresh
    });
  }

  @override
  Future<String> getToken() async {
    try {
      final token = await _firebaseMessaging.getToken();
      return token ?? '';
    } catch (e) {
      return '';
    }
  }

  static Future<void> _firebaseMessagingBackgroundHandler(
      RemoteMessage? message) async {
    if (message != null) {
      // handle background message
    }
  }
}
```

Kode Sumber 5. 2. Konfigurasi penerimaan notifikasi

5.1.3. Applications

Implementasi ini mencakup pengelolaan daftar aplikasi ITS yang terintegrasi dengan MyITS Notification. Pengguna dapat memilih aplikasi yang notifikasinya ingin mereka terima. Detail implementasi dapat dilihat pada Kode Sumber 5.3.

```
class ApplicationRemoteDataSourceImpl implements
ApplicationRemoteDataSource {
  const ApplicationRemoteDataSourceImpl({
    required SharedPreferences sharedPreferences,
    required FirebaseFirestore firebaseFirestore,
  })  : _sharedPreferences = sharedPreferences,
        _firebaseFirestore = firebaseFirestore;

  final SharedPreferences _sharedPreferences;
  final FirebaseFirestore _firebaseFirestore;

  @override
  Future<List<ApplicationModel>> getApplication() async {
    try {
      final accessToken = _sharedPreferences.getString(kAccessToken);
      final sub = _sharedPreferences.getString(kSub);

      QuerySnapshot appQuery =
          await _firebaseFirestore.collection('applications').get();

      final List<ApplicationModel> appList = appQuery.docs
          .map(
            (e) => ApplicationModel(
              id: e.get('id'),
              name: e.get('name'),
              description: e.get('description'),
              status: e.get('list_sub').contains(sub),
            ),
          )
          .toList();
```

```
    appList.sort((a, b) {
      if (a.status && !b.status) {
        return -1;
      } else if (!a.status && b.status) {
        return 1;
      } else {
        return a.name.toLowerCase().compareTo(b.name.toLowerCase());
      }
    });
    return appList;
  } on ServerException {
    rethrow;
  } on DioException catch (e) {
    debugPrint(e.toString());
    throw ServerException(
        message: e.toString(), statusCode: e.response?.statusCode);
  } catch (e, s) {
    debugPrintStack(stackTrace: s);
    throw ServerException(message: e.toString(), statusCode: 505);
  }
}

@override
Future<List<ApplicationModel>> activateNotification({
  required String id,
  required List<dynamic> applications,
}) async {
  try {
    final accessToken = _sharedPreferences.getString(kAccessToken);
    final sub = _sharedPreferences.getString(kSub);
    List<ApplicationModel> apps = List<ApplicationModel>.from(
      applications
          .map(
            (e) => ApplicationModel(
              id: e.id,
              name: e.name,
              description: e.description,
              status: e.status,
              appImg: e.appImg,
              url: e.url,
            ),
```

```
          )
          .toList(),
    );

    apps = apps.map((e) {
      if (e.id == id) {
        return e.copyWith(status: true);
      }
      return e;
    }).toList();

    apps.sort((a, b) {
      if (a.status && !b.status) {
        return -1;
      } else if (!a.status && b.status) {
        return 1;
      } else {
        return 0;
      }
    });

    await _firebaseFirestore
        .collection('applications')
        .doc(id)
        .update({
          'list_sub': FieldValue.arrayUnion([sub])
        });

    return apps;
  } on ServerException {
    rethrow;
  } on DioException catch (e) {
    debugPrint(e.toString());
    throw ServerException(
        message: e.toString(), statusCode: e.response?.statusCode);
  } catch (e, s) {
    debugPrintStack(stackTrace: s);
    throw ServerException(message: e.toString(), statusCode: 505);
  }
}
```

```
@override
Future<List<ApplicationModel>> deactivateNotification({
  required String id,
  required List<dynamic> applications,
}) async {
  try {
    final accessToken = _sharedPreferences.getString(kAccessToken);
    final sub = _sharedPreferences.getString(kSub);

    List<ApplicationModel> apps = List<ApplicationModel>.from(
      applications
          .map(
            (e) => ApplicationModel(
              id: e.id,
              name: e.name,
              description: e.description,
              status: e.status,
              appImg: e.appImg,
              url: e.url,
            ),
          )
          .toList(),
    );

    apps = apps.map((e) {
      if (e.id == id) {
        return e.copyWith(status: false);
      }
      return e;
    }).toList();

    apps.sort((a, b) {
      if (a.status && !b.status) {
        return -1;
      } else if (!a.status && b.status) {
        return 1;
      } else {
        return 0;
      }
    });
```

```
    await _firebaseFirestore
        .collection('applications')
        .doc(id)
        .update({
          'list_sub': FieldValue.arrayRemove([sub])
        });

    return apps;
  } on ServerException {
    rethrow;
  } on DioException catch (e) {
    debugPrint(e.toString());
    throw ServerException(
        message: e.toString(), statusCode: e.response?.statusCode);
  } catch (e, s) {
    debugPrintStack(stackTrace: s);
    throw ServerException(message: e.toString(), statusCode: 505);
  }
}

@override
Stream<List<ApplicationModel>> getApplicationsStream() {
  try {
    final sub = _sharedPreferences.getString(kSub);

    return _firebaseFirestore.collection('applications')
      .snapshots()
      .map(
      (event) {
        final appList = event.docs
            .map(
              (e) => ApplicationModel(
                id: e.id,
                name: e.get('name'),
                description: e.get('description'),
                status: e.get('list_sub').contains(sub),
                appImg: e.get('appImg'),
                url: e.get('url'),
              ),
            )
            .toList();
```

```
          appList.sort((a, b) {
            if (a.status && !b.status) {
              return -1;
            } else if (!a.status && b.status) {
              return 1;
            } else {
              return a.name.toLowerCase()
                     .compareTo(b.name.toLowerCase());
            }
          });
          return appList;
        },
      );
    } on ServerException {
      rethrow;
    } on CacheException {
      rethrow;
    } catch (e, s) {
      debugPrintStack(stackTrace: s);
      throw ServerException(message: e.toString(), statusCode: 505);
    }
  }
}
```

Kode Sumber 5. 3. Implementasi screen list aplikasi

5.1.4. Settings

Bagian ini menjelaskan implementasi pengaturan notifikasi di aplikasi Flutter, di mana pengguna dapat mengatur preferensi notifikasi mereka. Kode Sumber 5.4 menunjukkan detail konfigurasi pengaturan ini.

```
class _SettingsScreenState extends State<SettingsScreen> {
  @override
  Widget build(BuildContext context) {
    return Consumer<InsightProvider>(
      builder: (_, insightProvider, __) {
        return BlocConsumer<SettingsBloc, SettingsState>(
          listener: (context, state) {
```

```
            if (state is SettingsError) {
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text(state.message),
                  backgroundColor: Colors.red,
                ),
              );
            } else if (state is LanguageChanged) {
              if (state.language == "id") {
                insightProvider.setToIndonesian();
              } else if (state.language == "en") {
                insightProvider.setToEnglish();
              } else if (state.language == "default") {
                insightProvider.setToDefaultLanguage();
              }
            } else if (state is ThemeChanged) {
              if (state.theme == "Light") {
                insightProvider.setToLightMode();
              } else if (state.theme == "Dark") {
                insightProvider.setToDarkMode();
              } else if (state.theme == "Default") {
                insightProvider.setToDefaultMode();
              }
            }
          },
          builder: (context, state) {
            return GradientBackground(
              color: context.theme.colorScheme.background,
              child: SingleChildScrollView(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    SettingsContainer(
                      title: AppLocalizations.of(context)!
                             .displayMode,
                      description: AppLocalizations.of(context)!
                                   .selectDisplayMode,
                      content: [
                        SettingsButton(
                          title: AppLocalizations.of(context)!
                                 .lightMode,
```

```
                    isSelected: insightProvider.isLightMode(),
                    onTap: () {
                      context.read<SettingsBloc>()
                          .add(const ChangeThemeEvent("Light"));
                    },
                    iconColor: insightProvider.isLightMode() ?
                         Colours.secondaryColour :
                         Colours.fontSecondaryColour,
                    backgroundColor:
                         insightProvider.isLightMode() ?
                         Colours.secondaryColour.withOpacity(0.1
                         ) : Colors.transparent,
                    borderColor: insightProvider.isLightMode() ?
                           Colours.secondaryColour :
                           Colours.fontSecondaryColour,
                  ),
                  const SizedBox(height: 10),

                  SettingsButton(
                    title: AppLocalizations.of(context)!
                           .darkMode,
                    isSelected: insightProvider.isDarkMode(),
                    onTap: () {
                      context.read<SettingsBloc>().add(
                        const ChangeThemeEvent("Dark")
                      );
                    },
                    iconColor: insightProvider.isDarkMode() ?
                         Colours.secondaryColour :
                         Colours.fontSecondaryColour,
                    backgroundColor:
                        insightProvider.isDarkMode() ?
                        Colours.secondaryColour.withOpacity(0.1)
                        : Colors.transparent,
                    borderColor: insightProvider.isDarkMode() ?
                        Colours.secondaryColour :
                        Colours.fontSecondaryColour,
                  ),
                  const SizedBox(height: 10),

                  SettingsButton(
```

```
                        title: AppLocalizations.of(context)!
                               .defaultMode,
                        isSelected: insightProvider.themeData ==
                                    null,
                        onTap: () {
                          context.read<SettingsBloc>().add(
                            const ChangeThemeEvent("Default")
                          );
                        },
                        iconColor: insightProvider.themeData ==
                             null ? Colours.secondaryColour :
                             Colours.fontSecondaryColour,
                        backgroundColor: insightProvider.themeData
                             == null ?
                             Colours.secondaryColour.withOpacity(0.1
                             ) : Colors.transparent,
                        borderColor: insightProvider.themeData ==
                             null ? Colours.secondaryColour :
                             Colours.fontSecondaryColour,
                      ),
                    ],
                  ),

                  SettingsContainer(
                    title: AppLocalizations.of(context)!.language,
                    description: AppLocalizations.of(context)!
                                 .selectLanguage,
                    content: [
                      SettingsButton(
                        title: AppLocalizations.of(context)!
                               .indonesia,
                        isSelected: insightProvider.isIndonesian(),
                        onTap: () {
                          context.read<SettingsBloc>().add(
                            const ChangeLanguageEvent("id")
                          );
                        },
                        iconColor: insightProvider.isIndonesian() ?
                             Colours.secondaryColour :
                             Colours.fontSecondaryColour,
                        backgroundColor:
```

```
                    insightProvider.isIndonesian() ?
                    Colours.secondaryColour.withOpacity(0.1
                    ) : Colors.transparent,
                borderColor:
                    insightProvider.isIndonesian() ?
                    Colours.secondaryColour :
                    Colours.fontSecondaryColour,
              ),
              const SizedBox(height: 10),

              SettingsButton(
                title: AppLocalizations.of(context)!
                      .english,
                isSelected: insightProvider.isEnglish(),
                onTap: () {
                  context.read<SettingsBloc>().add(
                    const ChangeLanguageEvent("en")
                  );
                },
                iconColor: insightProvider.isEnglish() ?
                    Colours.secondaryColour :
                    Colours.fontSecondaryColour,
                backgroundColor:
                    insightProvider.isEnglish() ?
                    Colours.secondaryColour.withOpacity(0.1)
                    : Colors.transparent,
                borderColor: insightProvider.isEnglish() ?
                    Colours.secondaryColour :
                    Colours.fontSecondaryColour,
              ),
              const SizedBox(height: 10),

              SettingsButton(
                title: AppLocalizations.of(context)!
                      .defaultMode,
                isSelected: insightProvider
                            .isDefaultLanguage(),
                onTap: () {
                  context.read<SettingsBloc>().add(
                    const ChangeLanguageEvent("default")
                  );
```

```
                        },
                        iconColor:
                            insightProvider.isDefaultLanguage() ?
                            Colours.secondaryColour :
                            Colours.fontSecondaryColour,
                        backgroundColor:
                            insightProvider.isDefaultLanguage() ?
                            Colours.secondaryColour.withOpacity(0.1)
                            : Colors.transparent,
                        borderColor:
                            insightProvider.isDefaultLanguage() ?
                            Colours.secondaryColour :
                            Colours.fontSecondaryColour,
                      ),
                    ],
                  ),
                ],
              ),
            ),
          );
        },
      );
    },
  );
  }
}
```

Kode Sumber 5. 4. Implementasi screen settings

5.2. Implementasi *BackEnd*

Implementasi backend berfokus pada pengelolaan notifikasi dan penyimpanan data menggunakan Firebase. Adapun rincian implementasi backend sebagai berikut:

5.2.1. Firebase Authentication

Pada implementasi ini, Firebase Authentication digunakan untuk mengelola autentikasi dan otorisasi pengguna. Konfigurasi Firebase Authentication dapat dilihat pada Kode Sumber 5.5.

```
{
  "project_info": {
    "project_number": "323445684354",
    "project_id": "myits-housing",
    "storage_bucket": "myits-housing.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id":
            "1:323445684354:android:1b4664e8558e0f8fd3dc9b",
        "android_client_info": {
          "package_name": "id.ac.its.my.notification"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": "xxxx"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Kode Sumber 5. 5. Konfigurasi Firebase Authentication

5.2.2. Firebase Functions

Firebase Cloud Functions digunakan untuk mengelola dan memproses notifikasi yang masuk dari berbagai aplikasi ITS. Kode Sumber 5.6 menunjukkan konfigurasi Firebase Functions untuk menangani request notifikasi.

```
exports.sendNotification = functions.region("asia-southeast2")
    .https.onCall(async (data, context) => {
        const target = data["target"];
        const tokens = [];
        const appId = data["idApp"];
        const applications = await admin.firestore()
            .collection("applications").doc(appId).get();
        const application = applications.data();
        const subscriptions = application["list_sub"];
        // store notification to firestore
        const notifData = {
            idApp: data["idApp"],
            appName: application["name"],
            title: data["title"],
            body: data["body"],
            read: [],
            deleted: [],
            img: data["img"],
            imgApp: application["appImg"],
            url: application["url"],
            date: Timestamp.now(),
            target: data["target"].map((element) => {
                return element.toLowerCase();
            }),
        };
        let notifRef = null;
        try {
            notifRef = await admin.firestore()
                .collection("notifications").add(notifData);
        } catch (error) {
            return {
                message: error,
                data: notifData,
            };
        }
        const dataSend = {
            id: notifRef.id,
            idApp: notifData["idApp"],
            appName: notifData["appName"],
            title: notifData["title"],
            body: notifData["body"],
```

```
            img: notifData["img"],
            imgApp: notifData["imgApp"],
            url: notifData["url"],
            date: notifData["date"].toDate().toISOString(),
        };
        const notifSend = {
            title: data["title"],
            body: data["body"],
        };
        const androidSend = {
            notification: {
                sound: "beep",
                priority: "high",
                imageUrl: data["img"],
                channelId: application["name"],
            },
        };
        if (target.includes("all")) {
            const users = await admin.firestore()
                .collection("users").get();
            users.forEach(async (user) => {
                if (subscriptions.includes(user.data()["sub"])) {
                    const userFcmTokens = user.data()["fcmtoken"];
                    userFcmTokens.forEach(async (token) => {
                        tokens.push(token);
                    });
                }
            });
        } else {
            const users = await admin.firestore()
                .collection("users").get();
            users.forEach(async (user) => {
                if (subscriptions.includes(user.data()["email"])) {
                    const userRole = user.data()["role"];
                    const userSub = user.data()["sub"];
                    if (target.includes(userRole) ||
                       target.includes(userSub)) {
                        const userFcmTokens = user
                            .data()["fcmtoken"];
                        userFcmTokens.forEach(async (token) => {
                            tokens.push(token);
```

```
                        });
                    }
                }
            });
        }
        const payload = {
            data: dataSend,
            notification: notifSend,
            android: androidSend,
            tokens: tokens,
        };
        try {
            return await admin.messaging()
                .sendEachForMulticast(payload);
        } catch (error) {
            return {
                message: error,
                tokens: tokens,
            };
        }
    });
// Http Call
POST::host/sendNotification
{
    "data": {
        "idApp": "0E2679E3-6CC3-4F20-8DCD-AEA449C41875",
        "title": "Internet Tersambung",
        "body": "Anda sudah terhubung pada jaringan ITS",
        "img": "https://portal.its.ac.id/images/icon-app/Akses-
                Internet.png",
        "target": [
            "all"
        ]
    }
}
```

Kode Sumber 5. 6. Konfigurasi Firebase Function

5.2.3. Firebase Firestore Database

Firebase Firestore digunakan sebagai basis data untuk menyimpan informasi notifikasi. Kode Sumber

5.7 menunjukkan konfigurasi Firebase Firestore untuk penyimpanan data notifikasi.

```
Future<void> addUserToFirestore({
  required String email,
  required List<String> fcmToken,
  required String name,
  required String nrp,
  required String role,
  required String sub,
}) async {
  try {
    // Reference to Firestore collection
    CollectionReference users =
FirebaseFirestore.instance.collection('users');

    // Set document with 'sub' as document ID
    await users.doc(sub).set({
      'email': email,
      'fcmtoken': fcmToken,
      'name': name,
      'nrp': nrp,
      'role': role,
      'sub': sub,
    });

    print('User added/updated successfully');
  } catch (e) {
    print('Error adding/updating user: $e');
  }
}

Future<void> addNotificationToFirestore({
  required String appName,
  required String body,
  required Timestamp date,
  required List<String> deleted,
  required String idApp,
  required String img,
  required String imgApp,
  required List<String> read,
```

```
  required List<String> target,
  required String title,
  required String url,
}) async {
  try {
    // Reference to Firestore collection
    CollectionReference notifications =
        FirebaseFirestore.instance.collection('notifications');

    // Add document with a random ID
    await notifications.add({
      'appName': appName,
      'body': body,
      'date': date,
      'deleted': deleted,
      'idApp': idApp,
      'img': img,
      'imgApp': imgApp,
      'read': read,
      'target': target,
      'title': title,
      'url': url,
    });

    print('Notification added successfully');
  } catch (e) {
    print('Error adding notification: $e');
  }
}

Future<void> addApplicationToFirestore({
  required String idApp,
  required String appImg,
  required String description,
  required List<String> listSub,
  required String name,
  required String url,
}) async {
  try {
    // Reference to Firestore collection
    DocumentReference application =
```

```
      FirebaseFirestore.instance.collection('applications').doc(idA
      pp);

    // Set document data
    await application.set({
      'appImg': appImg,
      'description': description,
      'list_sub': listSub,
      'name': name,
      'url': url,
    });

    print('Application added successfully');
  } catch (e) {
    print('Error adding application: $e');
  }
}
```

Kode Sumber 5. 7. Konfigurasi Firebase Function

5.2.4. Firebase Messaging

Firebase Cloud Messaging (FCM) digunakan untuk mengirimkan notifikasi push ke perangkat pengguna. Kode Sumber 5.8 menunjukkan konfigurasi FCM untuk mengirimkan notifikasi secara real-time.

```
const dataSend = {
  id: notifRef.id,
  idApp: notifData["idApp"],
  appName: notifData["appName"],
  title: notifData["title"],
  body: notifData["body"],
  img: notifData["img"],
  imgApp: notifData["imgApp"],
  url: notifData["url"],
  date: notifData["date"]
    .toDate()
    .toISOString(),
};
```

```
const notifSend = {
  title: data["title"],
  body: data["body"],
};

const androidSend = {
  notification: {
    sound: "beep",
    priority: "high",
    imageUrl: data["img"],
    channelId: application["name"],
  },
};

const payload = {
  data: dataSend,
  notification: notifSend,
  android: androidSend,
  tokens: tokens,
};

try {
  return await admin
    .messaging()
    .sendEachForMulticast(payload);
} catch (error) {
  return {
    message: error,
    tokens: tokens,
  };
}
```

Kode Sumber 5. 8. Konfigurasi FCM

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Aplikasi myITS Notification. Pengujian dilakukan untuk memastikan fungsionalitas dan kesesuaian hasil implementasi arsitektur dengan analisis dan perancangan arsitektur.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap Aplikasi myITS Notification guna menguji kemampuan arsitektur dalam melayani permintaan sistem aplikasi.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut :

a. Kemampuan arsitektur untuk melayani tampilan aplikasi.
b. Kemampuan arsitektur untuk menambahkan notifikasi melalui API *call.*
c. Kemampuan arsitektur untuk melakukan autentikasi menggunakan akun ITS.
d. Kemampuan arsitektur untuk mengaktifkan dan menonaktifkan aplikasi yang ingin di-*subscribe.*
e. Kemampuan arsitektur untuk melihat notifikasi yang baru ditambahkan
f. Kemampuan arsitektur untuk melihat riwayat notifikasi.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai user yang akan menjalankan fitur-fitur.

Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut :

1. Admin dapat menambahkan notifikasi baru melalui API *call.*
2. User dapat melakukan autentikasi menggunakan akun ITS.
3. User dapat mengaktifkan dan menonaktifkan aplikasi yang ingin di-*subscribe.*
4. User dapat melihat notifikasi yang baru ditambahkan pada *bar* notifikasi *device*.
5. User dapat melihat riwayat notifikasi.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem aplikasi PPDB terhadap kasus skenario uji coba. Tabel 6.1 di bawah ini menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

Tabel 6. 1. Hasil Evaluasi Pengujian

Kriteria Pengujian	**Hasil Pengujian**
Sistem dapat melayani tampilan aplikasi	Terpenuhi
Sistem dapat menambahkan notifikasi melalui API *call*	Terpenuhi
Ssitem dapat melakukan autentikasi menggunakan akun ITS	Terpenuhi
Sistem dapat menampilkan aplikasi yang ingin dan tidak	Terpenuhi

ingin di-*subscribe*	
Sistem dapat menampilkan notifikasi yang baru ditambahkan	Terpenuhi
Sistem dapat melihat riwayat notifikasi	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan perancangan arsitektur sistem aplikasi myITS Notification adalah sebagai berikut :

a. Arsitektur sistem yang dibangun telah sesuai dengan permintaan.
b. Dengan adanya aplikasi myITS Notification, DPTSI ITS dapat dengan mudah mengirimkan notifikasi ke user sesuai dengan aplikasi-aplikasi yang diinginkan.

7.2. Saran

Saran untuk perancangan arsitektur sistem aplikasi myITS Notification adalah dilakukan pengembangan lebih lanjut untuk perangkat IOS.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] Guru99, “Guru 99,” [Online]. Available: https://www.guru99.com/mobile-testing.html. [Diakses 1 June 2024].

[2] Google Developer, “Android Developer,” [Online]. Available: https://developer.android.com/?hl=id. [Diakses 1 June 2024].

[3] Google Developer, “Apple Developer,” [Online]. Available: https://developer.apple.com/ios/. [Diakses 1 June 2024].

[4] Dart, “Dart,” [Online]. Available: https://dart.dev/. [Diakses 1 June 2024].

[5] Flutter, “Flutter,” [Online]. Available: https://flutter.dev/. [Diakses 1 June 2024].

[6] Firebase, “Firebase,” [Online]. Available: https://firebase.google.com/. [Diakses 1 June 2024].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS I

Nama	: Jabalnur
Tempat, Tanggal Lahir	: Pangkep, 29 Juni 2000
Jenis Kelamin	: Laki-laki
Telepon	: +6285342582944
Email	: jabalnur.it@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2020
Semester	: 8 (Delapan)

BIODATA PENULIS II

Nama	: Sejati Bakti Raga
Tempat, Tanggal Lahir	: Lombok Timur, 17 Februari 2003
Jenis Kelamin	: Laki-laki
Telepon	: +6283117305743
Email	: sejatibaktiraga@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2020
Semester	: 8 (Delapan)