BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan program kerja magang di PT. AIRFAST Indonesia, penempatan posisi dilakukan pada departemen MIS (*Management Information System*) dengan kedudukan sebagai *Mobile App Developer*. Penempatan ini sesuai dengan latar belakang pendidikan di bidang Informatika. Dalam posisi tersebut, tanggung jawab utama adalah merancang dan mengembangkan aplikasi internal karyawan untuk platform Android dan iOS menggunakan bahasa pemrograman Kotlin dan Swift.

Selama program magang, supervisi dilakukan langsung oleh Bapak Galih Suyoga, yang berperan sebagai supervisor dan pembimbing lapangan. Sebagai supervisor, Bapak Galih Suyoga bertanggung jawab untuk memberikan arahan proyek, menyediakan *tools* dan sumber daya yang diperlukan untuk pengembangan aplikasi, serta memberikan evaluasi dan umpan balik terhadap pekerjaan yang dilakukan.

Alur koordinasi dalam pengerjaan proyek bersifat langsung, di mana supervisor memberikan penugasan proyek pengembangan aplikasi internal karyawan, kemudian dilakukan seluruh tahapan pengembangan dari awal. Tahapan tersebut dimulai dari penyusunan konsep awal dan tata letak antarmuka pengguna (UI), yang kemudian langsung diimplementasikan ke dalam kode XML di Android Studio. Selanjutnya, dilakukan implementasi desain tersebut ke dalam bentuk kode, hingga pengembangan fungsi dan logika bisnis di balik tampilan aplikasi." Selama proses pengembangan, terdapat kebebasan untuk berkonsultasi dengan supervisor jika menemui kendala atau membutuhkan arahan lebih lanjut. Selain konsultasi langsung, pencarian solusi juga dapat dilakukan secara mandiri melalui berbagai sumber di internet.

Mekanisme pelaporan dan koordinasi dilakukan melalui laporan progres harian. Setiap hari, wajib dibuat dokumentasi kemajuan pengerjaan proyek dan dilaporkan kepada supervisor. Pelaporan ini bertujuan untuk memastikan bahwa pengembangan aplikasi berjalan sesuai dengan *timeline* yang telah ditentukan dan memungkinkan supervisor untuk memberikan umpan balik secara tepat waktu.

Dalam pelaksanaan tugas, koordinasi hanya dilakukan dengan anggota

departemen MIS. Hal ini memungkinkan untuk fokus pada pengembangan aplikasi tanpa terdistraksi oleh koordinasi lintas departemen yang kompleks. Meskipun demikian, perhatian tetap diberikan pada kebutuhan pengguna dari berbagai departemen dalam merancang aplikasi, berdasarkan informasi dan arahan yang diberikan oleh supervisor.

Melalui struktur koordinasi langsung dan efisien ini, pekerjaan dapat dilakukan secara mandiri dengan tetap memperoleh bimbingan yang memadai, sehingga memungkinkan penyelesaian proyek pengembangan aplikasi internal karyawan sesuai dengan target dan kualitas yang diharapkan oleh perusahaan.

3.2 Tugas yang Dilakukan

Pada masa magang di PT. AIRFAST Indonesia, berbagai tugas dan tanggung jawab yang berkaitan dengan pengembangan aplikasi *mobile* diberikan untuk meningkatkan efisiensi karyawan. Tugas yang dilakukan mencakup pengembangan dan implementasi berbagai fitur yang esensial dalam aplikasi, seperti desain antarmuka pengguna (UI/UX), integrasi API (*Application Programming Interface*), sistem autentikasi dan keamanan, serta fitur-fitur seperti absensi, *helpdesk*, tanda tangan digital, pengumuman, dan notifikasi. Setiap fitur dirancang dan diimplementasikan dengan mempertimbangkan kebutuhan perusahaan dan pengguna akhir, serta memastikan bahwa aplikasi dapat berjalan secara efisien dan aman di platform Android dan iOS. Adapun detail tugas yang dilakukan selama magang ini akan dijelaskan pada bagian berikutnya.

1. Pengembangan UI/UX Aplikasi
    - Mengimplementasikan keseluruhan UI/UX aplikasi
    - Mengembangkan komponen UI seperti *bottom navigation*, *tab layout*, dan dialog
2. Integrasi API
    - Mengintegrasikan API *backend* menggunakan Retrofit dan OkHttp untuk platform Android dan Alamofire untuk platform iOS
    - Menerapkan *converter data* untuk transformasi JSON ke model objek
    - Mengimplementasikan *Shared Preferences* untuk penyimpanan data lokal dan *Session Manager*

3. Sistem Autentikasi dan Keamanan
    - Mengembangkan sistem *login*
    - Mengimplementasikan *Session Manager* untuk pengelolaan data pengguna dan token autentikasi
    - Menerapkan *Time-based One-Time Password* (TOTP) untuk keamanan multi-faktor
    - Mengembangkan mekanisme penanganan token kedaluwarsa dan *auto logout*
    - Menerapkan *Proguard Rules* untuk *obfuscation* kode pada *build release*
4. Fitur Absensi
    - Mengimplementasikan fungsionalitas *check-in* dan *check-out* dengan verifikasi lokasi
    - Mengembangkan opsi absensi khusus (*work from home*, *external activity*, dll.)
    - Menerapkan perhitungan waktu kerja dan visualisasi waktu tersisa
5. Fitur *Helpdesk*
    - Mengembangkan sistem manajemen tiket bantuan
    - Menerapkan alur kerja untuk pengambilan dan penyelesaian tiket
6. Fitur *Sign Document*
    - Merancang sistem persetujuan dokumen dan tanda tangan digital
    - Mengembangkan komponen kanvas untuk pembuatan tanda tangan
    - Mengimplementasikan *rendering* PDF dengan dukungan *zoom* dan navigasi
    - Menerapkan integrasi tanda tangan ke dokumen PDF
7. Fitur *Announcement*
    - Mengimplementasikan sistem kategori pengumuman dengan tab navigasi
    - Merancang *PDF viewer* untuk dokumen pengumuman menggunakan *Web View*

- Mengintegrasikan penanganan API untuk mengambil data pengumuman dari server

8. Fitur Notifikasi
    - Mengintegrasikan *Firebase Cloud Messaging* (FCM) untuk *push notification*
    - Mengembangkan *background service* untuk pengambilan notifikasi secara periodik
    - Menerapkan *bottom sheet dialog* untuk menampilkan detail notifikasi
9. Profile Page
    - Merancang tampilan profil pengguna dengan data detail (nama, email, jabatan, divisi)
    - Mengimplementasikan penggunaan data respons API *login* untuk mengisi tampilan profil pengguna,
10. *Error Handling*
    - Menerapkan penanganan kesalahan jaringan dengan pesan informatif
    - Mengembangkan sistem *logging* untuk pelacakan *error*
11. *Web View* dan *Quick Links*
    - Mengimplementasikan *Web View* untuk halaman yang membutuhkan tampilan web.
    - Mengembangkan render PDF melalui Google Docs Viewer
    - Mengimplementasikan *quick links* pada halaman *Home* ke aplikasi perusahaan lainnya

3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan kerja magang di PT. AIRFAST Indonesia, tugas yang dilakukan berfokus pada pengembangan aplikasi internal karyawan. Peran ini mencakup seluruh siklus pengembangan, mulai dari pembuatan desain UI/UX untuk memastikan pengalaman pengguna yang optimal hingga melakukan integrasi API untuk mendukung fungsionalitas utama aplikasi serta mengimplementasikan

autentikasi TOTP *(Time-based One-Time Password)* guna meningkatkan keamanan. Tugas yang dilakukan setiap minggu dapat dilihat pada Tabel 3.1

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Perkenalan perusahaan, pengarahan deskripsi pekerjaan, mempelajari Android *development* dan Kotlin, dan membuat UI/UX aplikasi.
2	Membuat desain UI/UX langsung ke dalam bentuk kode dan integrasi API ke dalam aplikasi.
3	Mengerjakan halaman *Home*, *Announcement*, *Notification*, *Profile*, dan *Helpdesk*.
4	Menyelesaikan fitur *helpdesk* dan mulai pengerjaan fitur absensi.
5	Mengerjakan fitur absensi, mencoba menerapkan fitur *push notification*, dan mengerjakan tugas sampingan yaitu membuat *front end* untuk halaman situs *Bulletin* perusahaan.
6	Melakukan perbaikan fitur absensi, pengujian fitur *push notification*, menambahkan *splash screen* aplikasi, dan menerapkan TOTP untuk autentikasi aplikasi.
7	Finalisasi fitur absensi beserta pengujian, mengerjakan fitur *Sign Document*, memperbaiki *bug* pada fitur *Helpdesk*, dan melakukan *obfuscate* pada aplikasi.
8	Mengerjakan tugas lain yaitu membuat halaman Admin untuk situs *bulletin* perusahaan menggunakan Flask.
9	Mulai pengerjaan untuk aplikasi iOS, mengimplementasikan UI ke dalam bentuk kode, dan memperbaiki fitur absensi pada Android.
10	Mengerjakan UI untuk aplikasi iOS, dan mempelajari integrasi API, *network* dan TOTP di Swift.
11	Mulai mengintegrasikan API, menerapkan TOTP, mengerjakan beberapa halaman, dan melakukan setup untuk *push notification* menggunakan APNs dan Firebase.
12	Mengerjakan fitur absensi serta melakukan pengujian fitur ketika sudah selesai.

Minggu Ke -	Pekerjaan yang dilakukan
13	Mengerjakan fitur *helpdesk*: Integrasi API tampilan *helpdesk*, *ticket detail*, dan *request ticket*.
14	Melanjutkan pengerjaan fitur *Helpdesk* dan mengerjakan fitur *Sign Document*.
15	Memperbaiki beberapa *bug* pada fitur absensi dan melakukan pengujian fitur yang sudah dikerjakan.
16	Melakukan pembelajaran mandiri.
17	Melakukan pembelajaran mandiri.
18	Melakukan *build file* untuk aplikasi iOS dan memperbaiki *bug* pada aplikasi Android

3.3.1 Metodologi dan Alur Kerja Pengembangan

Selama proses pengembangan aplikasi internal karyawan di PT. AIRFAST Indonesia, tidak ada metodologi manajemen proyek yang secara resmi diterapkan. Tugas-tugas diberikan oleh supervisor yang kemudian diselesaikan sesuai dengan prioritas dan kebutuhan perusahaan. Untuk memantau kemajuan, dilakukan laporan harian yang berisi progres yang telah dicapai dan masalah yang dihadapi dalam pengerjaan aplikasi. Laporan ini disampaikan langsung kepada supervisor yang kemudian memberikan umpan balik atau arahan untuk melanjutkan pengembangan lebih lanjut. Dengan adanya laporan harian ini, pengawasan terhadap progres pekerjaan dapat dilakukan dengan efektif dan memastikan aplikasi dapat dikembangkan sesuai dengan kebutuhan.

Alur kerja pengembangan aplikasi mengikuti pendekatan bertahap, dimulai dengan pemahaman kebutuhan aplikasi, pembuatan desain awal, pengembangan UI/UX, hingga implementasi fitur dan pengujian. Berikut adalah penjelasan rinci mengenai alur kerja pengembangan aplikasi yang dilakukan selama magang:

1. Diskusi Awal dan Pemahaman Kebutuhan Aplikasi

   Tahap awal pengembangan aplikasi Android dimulai dengan sesi diskusi bersama supervisor, Bapak Galih Suyoga, yang bertujuan untuk menerima penugasan resmi. Dalam diskusi ini, dilakukan penggalian informasi mendalam guna memperoleh pemahaman komprehensif mengenai keseluruhan alur fungsional aplikasi, fitur-fitur utama yang esensial,

serta kebutuhan spesifik yang diharapkan oleh pengguna di lingkungan perusahaan.

2. Pembuatan *Mockup* Sederhana

   Setelah pemahaman kebutuhan terbentuk, langkah selanjutnya adalah perancangan awal antarmuka pengguna dengan pembuatan *mockup* atau sketsa desain sederhana secara mandiri di kertas. Mockup ini berfungsi sebagai panduan visual awal untuk tata letak dan alur navigasi aplikasi, tanpa melalui proses perancangan UI/UX formal menggunakan perangkat lunak khusus seperti Figma.

3. Implementasi UI/UX

   Berdasarkan *mockup* sederhana yang telah dibuat, dilakukan implementasi desain antarmuka pengguna (UI) dan pengalaman pengguna (UX) langsung ke dalam kode untuk platform Android. Implementasi ini menggunakan bahasa *markup* XML untuk mendefinisikan tata letak dan komponen visual, serta bahasa pemrograman Kotlin untuk logika interaksi antarmuka.

4. Integrasi API Berdasarkan Fitur

   Setelah keseluruhan kerangka antarmuka pengguna utama aplikasi Android selesai diimplementasikan, supervisor menyediakan akses beserta dokumentasi *Application Programming Interface (API)* yang krusial untuk mengaktifkan seluruh fungsionalitas aplikasi. Proses integrasi API ini dijalankan secara modular dan bertahap, dengan urutan prioritas berdasarkan fitur yang dikerjakan. Sebagai contoh, API untuk fitur "*Announcement*" diintegrasikan terlebih dahulu. Setelah halaman "*Announcement*" selesai, kemudian diikuti dengan integrasi API untuk fitur lainnya dan pola serupa diterapkan untuk fitur-fitur lainnya. Setiap keberhasilan integrasi API selalu diiringi dengan pengujian fungsionalitas dan pemeriksaan cermat terhadap log aplikasi. Langkah ini bertujuan untuk memverifikasi kelancaran transmisi data serta mendeteksi dan mengatasi potensi masalah sedini mungkin.

5. Implementasi *Push Notification* dan TOTP

   Menyusul integrasi API untuk fitur-fitur inti, pengembangan aplikasi Android dilanjutkan dengan implementasi fungsionalitas *push notification*. Fitur ini bertujuan untuk memungkinkan penyampaian informasi penting secara real-time kepada pengguna. Selanjutnya, untuk meningkatkan aspek

keamanan aplikasi, diimplementasikan mekanisme autentikasi multi-faktor menggunakan *Time-based One-Time Password (TOTP)*.

6. Pengujian Aplikasi dan *Obfuscation*

   Setelah seluruh fitur berhasil diimplementasikan pada aplikasi Android, dilakukan tahap pengujian menyeluruh terhadap semua fungsionalitas. Pengujian ini bertujuan untuk memastikan stabilitas dan kesesuaian aplikasi dengan kebutuhan yang telah ditetapkan. Sebagai langkah pengamanan kode sumber pada versi rilis, diterapkan pula teknik *code obfuscation*.

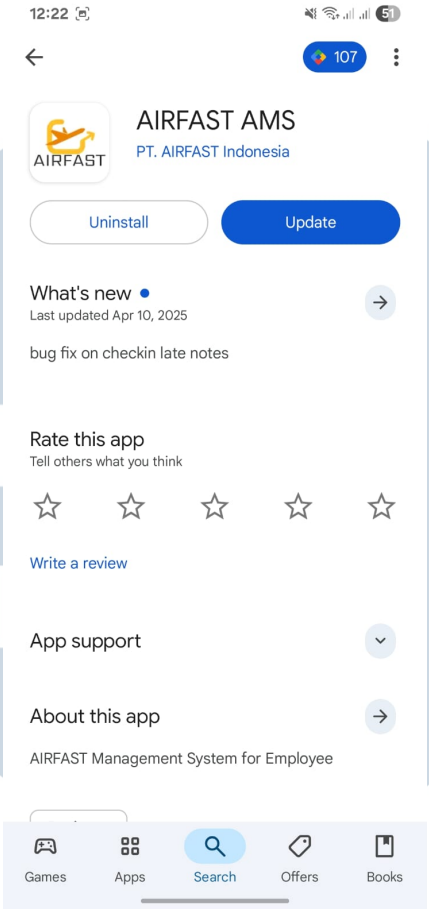
7. *Review*, Revisi, dan Penerbitan

   Versi aplikasi Android yang telah melalui pengujian dan *obfuscation* kemudian diunggah ke repositori GitLab perusahaan. Tahap ini bertujuan untuk proses peninjauan oleh supervisor. Jika terdapat temuan masalah atau permintaan revisi dari supervisor, perbaikan dilakukan segera. Setelah aplikasi dinyatakan stabil, aman, dan telah disetujui, aplikasi internal karyawan versi Android tersebut dipublikasikan untuk digunakan oleh seluruh karyawan perusahaan dan didistribusikan melalui Google Play Store.

8. Pembuatan Versi iOS

   Menyusul keberhasilan peluncuran versi Android, penugasan dilanjutkan dengan pengembangan aplikasi serupa untuk platform iOS menggunakan bahasa pemrograman Swift. Alur kerja pengembangan aplikasi iOS pada dasarnya mengikuti langkah-langkah yang sama dengan versi Android. Perbedaan utama terletak pada perancangan antarmuka, di mana desain UI untuk aplikasi iOS secara langsung mengacu pada tampilan akhir aplikasi Android yang telah selesai dikembangkan, sehingga mempercepat proses visualisasi dan implementasi UI.

Sebagai bukti dari tahap akhir penerbitan, aplikasi telah berhasil diunggah dan tersedia untuk distribusi melalui Google Play Store, seperti yang ditunjukkan pada gambar 3.1.



Gambar 3.1. Bukti Publikasi Aplikasi di Google Play Store

3.3.2 Pengujian Sistem

Setelah melakukan serangkaian pengujian terhadap berbagai fitur dan komponen aplikasi, hasil yang diperoleh menunjukkan bahwa semua skenario pengujian berhasil dijalankan dengan baik. Setiap skenario pengujian telah memberikan hasil yang sesuai dengan perkiraan, yang menandakan bahwa sistem berfungsi dengan baik, stabil, dan siap untuk digunakan. Berikut adalah tabel-tabel hasil pengujian yang telah dilakukan:

Tabel 3.2 menunjukkan hasil pengujian inisialisasi aplikasi dan splash screen pada aplikasi yang dikembangkan.

Tabel 3.2. Tabel pengujian inisialisasi aplikasi dan *Splash Screen*

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Buka aplikasi untuk pertama kali (belum login)	*Splash Screen* muncul selama 2 detik. Firebase diinisialisasi. Token FCM diambil dan disimpan. RetrofitClient diinisialisasi. *Channel* notifikasi dibuat. Pengguna diarahkan ke halaman *Login*.	Sesuai
Buka aplikasi (sudah login)	*Splash Screen* muncul selama 2 detik. Firebase diinisialisasi. Token FCM diambil/diperbarui dan disimpan. RetrofitClient diinisialisasi. Channel notifikasi dibuat. Pengguna diarahkan ke halaman *Login*.	Sesuai

Tabel 3.3 menunjukkan hasil pengujian pada halaman *Home*. Pengujian ini memastikan bahwa halaman *Home* ditampilkan dengan benar, serta memeriksa navigasi antara berbagai halaman di aplikasi.

Tabel 3.3. Tabel pengujian halaman *Home*

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Menampilkan halaman *Home* setelah *login* berhasil	Data *home* dimuat dari API. Status absensi (*check-in/check-out*) ditampilkan dengan benar. Tombol navigasi *home* bawah aktif dan terseleksi.	Sesuai
Klik card *Attendance*, *Helpdesk*, *Sign Document*, *Announcement*	Halaman *Attendance*, *Helpdesk*, *Sign Document*, *Announcement* terbuka.	Sesuai
Klik card ”*Technical Log*” dan ”*Aircraft Reports*”	Halaman *Web View* terbuka, URL sesuai dengan penambahan *keyword* pengguna.	Sesuai
Klik link aplikasi eksternal)	*Browser* eksternal terbuka dengan URL yang sesuai.	Sesuai
Navigasi ke halaman *Notification* dari *bottom navigation*	Halaman *Notification* ditampilkan.	Sesuai
Navigasi ke halaman *Profile* dari *bottom navigation*	Halaman *Profile* ditampilkan.	Sesuai

Tabel 3.4 menunjukkan hasil pengujian terkait fitur absensi aplikasi. Pengujian ini mencakup pengujian status *check-in* dan *check-out* dan penanganan izin lokasi dan *error* yang terjadi selama proses absensi.

Tabel 3.4. Tabel pengujian absensi)

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Buka halaman *Attendance*	Tanggal saat ini, status absensi, waktu *check-in*, dan jam kerja ditampilkan sesuai data terakhir.	Sesuai
Melakukan "*Regular Check-in*" (lokasi didapat dan tidak terlambat)	Lokasi diambil. API *check-in* dipanggil tanpa *notes* dan pesan "*Check-in successful*!" ditampilkan. Status di UI diperbarui (waktu *check-in*, status "*Present*", jam kerja mulai menghitung). Tombol berubah menjadi "*CHECK OUT*".	Sesuai
Melakukan "*Regular Check-in*" (lokasi didapat dan terlambat)	API *check-in* dipanggil. *View Model* mendeteksi kebutuhan *notes* dan dialog alasan telat muncul. Setelah input *notes* valid, *API check-in* dipanggil lagi. Pesan sukses ditampilkan dan UI diperbarui.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Melakukan ”*Special Occasion Check-in*” (contohnya ”*Work From Home*”)	Setelah melakukan pemilihan tipe absen *Special Occasion* kemudian muncul tipe *occasion* tersebut. Setelah dipilih akan memunculkan *notes*. Lokasi diambil dan API dipanggil, kemudian pesan ”*Special check-in successful*!” ditampilkan. UI diperbarui.	Sesuai
Melakukan ”*Special Occasion Check-i*n” tanpa mengisi *notes*	Pesan error ”*Notes must have more than 7 letters*” atau ”*Notes is required*” (sesuai validasi) ditampilkan. API tidak dipanggil.	Sesuai
Melakukan ”*check-out*”	Lokasi berhasil diambil. API *check-out* dipanggil. Pesan ”*Check-out successful*!” ditampilkan. Status di UI diperbarui menjadi nonaktif ”*Already Checkout*”.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Gagal *check-in* karena *error* API	Pesan *error* dari API ditampilkan. Status UI tidak berubah.	Sesuai
Izin lokasi belum diberikan saat melakukan absensi	Prompt izin lokasi muncul. Jika ditolak, pesan "*Location permission required*" ditampilkan.	Sesuai
Layanan lokasi mati saat melakukan absensi	Pesan "*Please enable location services*" ditampilkan. Absensi tidak dapat dilanjutkan.	Sesuai
Gagal mendapatkan lokasi saat absensi	Pesan "*Failed to retrieve location. Please try again.*" atau "*Location not available. Please try again.*" ditampilkan. Absensi tidak dapat dilanjutkan.	Sesuai

Tabel 3.5 menunjukkan hasil pengujian autentikasi pengguna, mencakup pengujian *login* dengan kredensial yang valid maupun tidak valid, serta pengujian proses *logout*. Pengujian ini memastikan bahwa aplikasi dapat mengelola sesi pengguna dengan benar dan memberikan pesan *error* yang sesuai ketika kredensial tidak valid.

Tabel 3.5. Tabel pengujian autentikasi (*login* dan *logout*)

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Login dengan kredensial valid	*Loading indicator* muncul. Permintaan API *login* dikirim dengan *username*, *password*, dan token FCM. *Session Manager* menyimpan data pengguna dan *API key*. Pengguna diarahkan ke halaman *Home*.	Sesuai
Login dengan *username* atau *password* kosong	*Error message* muncul di bawah kolom *username* dan *password*.	Sesuai
Login dengan *username* atau *password* salah	*Loading indicator* muncul lalu hilang. Pesan *error* dari API ditampilkan. Pengguna tetap berada di halaman *Login*.	Sesuai
Proses *Logout*	Dialog konfirmasi *logout* muncul. Setelah konfirmasi "*Yes*", API *logout* dipanggil. Pengguna diarahkan ke halaman *Login*.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Logout dari halaman *Profile* saat status absensi *checked in*	Dialog "*Warning*!" dengan pesan pemberitahuan dan pengguna tidak bisa *logout*.	Sesuai

Tabel 3.6 menunjukkan hasil pengujian notifikasi pada aplikasi. Pengujian ini mencakup pengujian notifikasi yang diterima saat aplikasi berada di *foreground* maupun *background*, serta interaksi dengan notifikasi yang muncul pada *status bar* dan navigasi melalui aplikasi.

Tabel 3.6. Tabel pengujian notifikasi

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Menerima notifikasi saat aplikasi di *foreground*	*Firebase Messaging Service* dipanggil. Notifikasi muncul di *status bar*.	Sesuai
Menerima notifikasi saat aplikasi di *background/killed*	*Firebase Messaging Service* dipanggil. Notifikasi muncul di *status bar*.	Sesuai
Klik item *push notification*	Aplikasi terbuka dan pengguna diarahkan ke halaman *Notification*.	Sesuai
Halaman *Notification* dibuka	Notifikasi diambil dari API. Daftar notifikasi ditampilkan menggunakan *Notification Adapter*.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Halaman *Notification* menampilkan kondisi tidak ada notifikasi	Pesan "*empty state*" ditampilkan dan *Recycler View* disembunyikan.	Sesuai
Klik item notifikasi di Halaman *Notification*	*Bottom sheet detail* dari notifikasi muncul menampilkan judul, isi, dan waktu.	Sesuai

Tabel 3.7 menunjukkan hasil pengujian fitur profil pengguna. Pengujian ini memastikan bahwa data pengguna yang disimpan dalam *Session Manager* ditampilkan dengan benar di halaman profil, serta menguji navigasi antar halaman terkait profil.

Tabel 3.7. Tabel pengujian profil pengguna

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Menampilkan halaman *Profile*	Nama pengguna dari *Session Manager* ditampilkan dan opsi menu terlihat.	Sesuai
Klik opsi "*Profile Information*"	Halaman *Profile Information* terbuka dan menampilkan nama, email, jabatan, dan divisi.	Sesuai

Skenario Pengujian	Perkiraan Hasil	Hasil
Halaman *Profile Information* menampilkan data pengguna	Nama, email, jabatan, dan divisi pengguna dari *Session Manager* ditampilkan dengan benar.	Sesuai
Kembali dari Halaman *Profile Information*	Kembali ke halaman *Profile*.	Sesuai

Tabel 3.8 menunjukkan hasil pengujian fitur *helpdesk*. Pengujian ini memastikan bahwa data tiket *helpdesk* ditampilkan dengan benar, serta memverifikasi fungsionalitas pembuatan tiket baru, perubahan status tiket, dan navigasi antar kategori status.

Tabel 3.8. Tabel pengujian *helpdesk*

Skenario Pengujian	Perkiraan Hasil	Hasil
Buka halaman *Helpdesk*	Data tiket *helpdesk* ditampilkan.	Sesuai
Pindah antar kategori (*Open, In Progress, Escalated, Closed*)	Konten berubah sesuai tab yang dipilih, menampilkan daftar tiket yang relevan.	Sesuai
Klik *Floating Button* di halaman *Helpdesk*	Halaman untuk *Add Request* terbuka.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Membuat permintaan tiket baru dengan *input* valid	*Loading indicator* muncul. Setelah sukses, pesan ”*Request successfully submitted*” ditampilkan. halaman *Add Request* tertutup dan kembali ke halaman *Helpdesk*. Tiket baru muncul di tab ”*Open*”.	Sesuai
Membuat permintaan baru namun masih memiliki tiket yang belum diselesaikan	Pesan *error* dari API ditampilkan. Pengguna tetap di berada di halaman *Add Reques*t.	Sesuai
Klik item tiket di halaman *Helpdesk*	Halaman detail tiket akan terbuka dengan menampilkan judul, status, lokasi, *requester*, *timestamp* dibuat/diperbarui, dan *timeline action* ditampilkan. Tombol aksi (*Take Job/Finish*) muncul sesuai kondisi.	Sesuai

Skenario Pengujian	Perkiraan Hasil	Hasil
Klik "*Take Job*" dan konfirmasi "*Yes*"	API *take job* dipanggil. Tiket pindah ke tab "*In Progress*".	Sesuai
Klik "*Finish*", pilih opsi "*Close Job*"	Dialog muncul. Pesan "*Successfully closed the ticket*" ditampilkan. Tiket pindah ke tab "*Closed*".	Sesuai
Klik "*Finish*", pilih opsi "*Escalate Job*"	Pesan "*Successfully escalated the ticket*" ditampilkan. Tiket pindah ke tab "*Escalated*".	Sesuai
Klik "*Finish*", pilih opsi "*Cancel Job*"	Pesan "*Successfully cancelled the ticket*" ditampilkan. Tiket pindah ke tab "*In Progress*".	Sesuai

Tabel 3.9 menunjukkan hasil pengujian fitur tanda tangan digital pada dokumen. Pengujian ini mencakup pengujian penandatanganan dokumen, penghapusan tanda tangan, dan pengiriman dokumen setelah tanda tangan dilakukan, serta fitur *zoom* pada PDF.

Tabel 3.9. Tabel pengujian *sign document*

Skenario Pengujian	Perkiraan Hasil	Hasil
Buka halaman *Sign Document*	Daftar dokumen dengan status "*Pending*" dimuat dan ditampilkan.	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Halaman *Sign Document* menampilkan kondisi kosong	Pesan ”*No pending documents to sign*.” ditampilkan. *Recycler View* disembunyikan.	Sesuai
Klik item dokumen di halaman *Sign Document*	Halaman detail *sign document* terbuka dengan judul, PDF dokumen yang sesuai, dan tempat untuk tanda tangan.	Sesuai
Menggambar tanda tangan	Tanda tangan muncul di area tempat tanda tangan.	Sesuai
Klik tombol ”*Clear*” di halaman detail *Sign Document* (mode gambar)	Tanda tangan terhapus.	Sesuai
Klik tombol ”*Submit*”	Dialog *input* catatan muncul. Ketika dialog sudah diisi maka akan muncul pesan berhasil dan kembali ke halaman *Sign Document*	Sesuai

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Input catatan dan klik "*Preview*" di dialog	Catatan disimpan. Tanda tangan di *encode* ke *Base64*. Halaman detail *sign document* masuk ke mode *preview*: PDF di *render* ulang dengan tanda tangan.	Sesuai
Zoom in/out pada PDF	Gestur *pinch-to-zoom* berfungsi dengan baik. *Double tap* untuk *reset/zoom in*.	Sesuai

Tabel 3.10 menunjukkan hasil pengujian fitur pengumuman dalam aplikasi. Pengujian ini mencakup pengujian tampilan pengumuman, memo, dan panduan, serta memverifikasi fungsionalitas pemilihan item pengumuman yang menampilkan dokumen PDF terkait.

Tabel 3.10. Tabel pengujian *announcement*

Skenario Pengujian	**Perkiraan Hasil**	**Hasil**
Buka halaman *Announcement*	*Tab* dengan kategori "*Announcement*", "*Memo*", "*Guide*" dan data pengumuman berhasil ditampilkan.	Sesuai

Skenario Pengujian	Perkiraan Hasil	Hasil
Pindah antar kategori (*Announcement, Memo, Guide*)	Berhasil menampilkan daftar item berdasarkan kategori *annoucement*.	Sesuai
Klik item pengumuman	Halaman *PDF Viewer* terbuka dan memuat PDF dari *announcement*.	Sesuai

Tabel 3.11 menunjukkan hasil pengujian terkait implementasi TOTP (*Time-Based One-Time Password*). Pengujian ini memastikan bahwa *interceptor* TOTP pada RetrofitClient berhasil menambahkan *header* ”otp” pada semua *request* API yang memerlukannya, kecuali pada *request login*.

Tabel 3.11. Tabel pengujian TOTP

Skenario Pengujian	Perkiraan Hasil	Hasil
Interceptor di RetrofitClient menambahkan *header* ”otp” ke semua *request* API kecuali login	Semua *request* API selain yang URL-nya mengandung ”*/login*” dan memiliki header ”otp” yang berisi kode TOTP.	Sesuai

Tangerang, 10 Juli 2025

Galih Suyoga

Supervisor

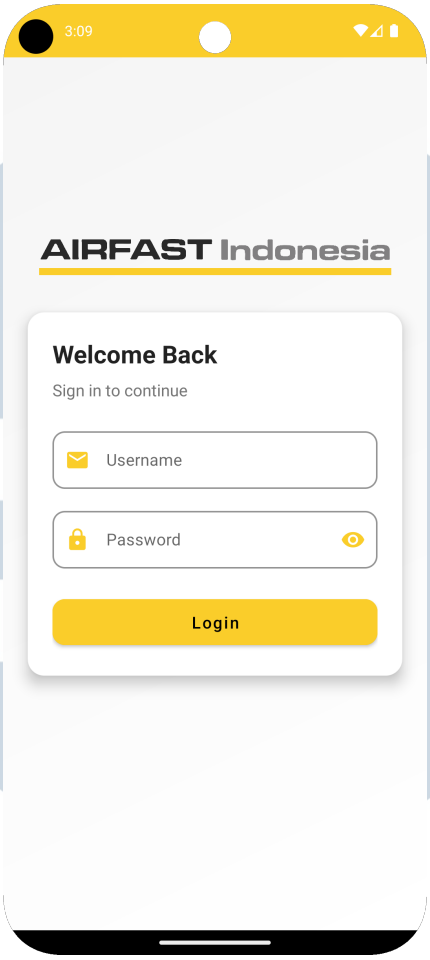
3.3.3 Hasil Akhir

1. Halaman *Login*

   Halaman *Login* merupakan gerbang utama bagi pengguna untuk mengakses fungsionalitas aplikasi. Pada halaman ini, pengguna diwajibkan untuk memasukkan *username* dan *password* yang telah didaftarkan oleh perusahaan untuk proses autentikasi. Sistem tidak menyediakan fitur registrasi mandiri karena manajemen akun dilakukan secara terpusat oleh perusahaan.

   Secara teknis, saat pengguna menekan tombol *login*, aplikasi akan mengirimkan permintaan ke API *backend* melalui *library* Retrofit dan OkHttp, yang berisi kredensial pengguna serta *token* Firebase Cloud Messaging (FCM) untuk keperluan notifikasi. Jika autentikasi berhasil, server akan memberikan respons berisi data pengguna dan sebuah *API key*. *Session Manager* pada aplikasi kemudian menyimpan *API key* ini di penyimpanan lokal menggunakan *Shared Preferences* untuk mengamankan sesi pengguna pada permintaan-permintaan berikutnya.

   Setelah itu, mekanisme keamanan *Time-based One-Time Password* (TOTP) akan diaktifkan untuk menambah lapisan keamanan pada setiap transaksi data setelah *login*. Jika seluruh proses ini berhasil, pengguna akan diarahkan ke Halaman *Home*, namun jika gagal, sebuah pesan kesalahan akan ditampilkan. Tampilan antarmuka Halaman *Login* dapat dilihat pada gambar 3.2.



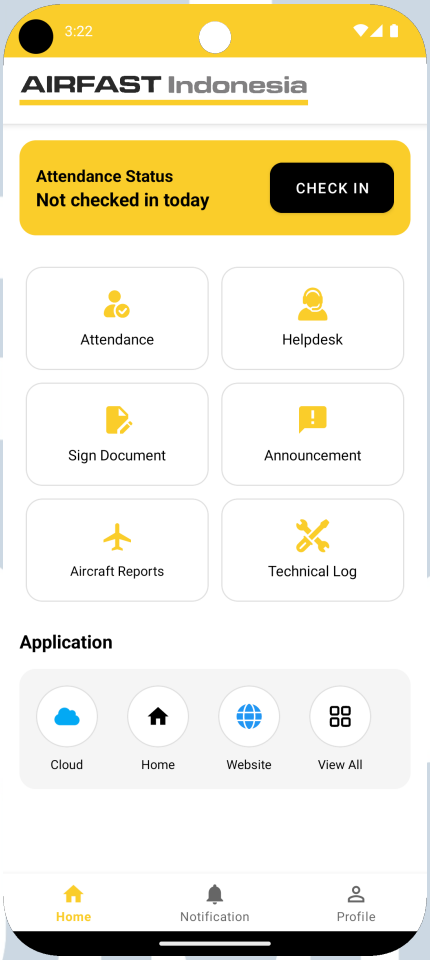
Gambar 3.2. Halaman Login

2. Halaman *Home*

Halaman utama atau Halaman *Home* berfungsi sebagai *dashboard* utama yang menyajikan informasi penting dan menyediakan akses cepat ke berbagai fitur aplikasi. Saat halaman ini dimuat, aplikasi secara otomatis melakukan pemanggilan API ke server untuk mengambil data status absensi terakhir pengguna, yang kemudian ditampilkan di bagian atas layar.

Di bawah informasi absensi, terdapat beberapa menu utama dalam bentuk (*card*) yang berfungsi sebagai navigasi. Menu seperti *Attendance*, *Helpdesk*, dan *Sign Document* akan mengarahkan pengguna ke halaman internal aplikasi yang sesuai. Sementara itu, menu lain seperti *"Technical Log"* dan *"Aircraft Reports"* dirancang untuk membuka halaman web internal melalui komponen *Web View* di dalam aplikasi, di mana URL yang dituju dapat ditambahkan *keyword* spesifik pengguna.

Selain itu, pada bagian bawah terdapat menu *"Application"* yang berisi tautan cepat (*quick links*) ke aplikasi atau situs web perusahaan lainnya, yang akan dibuka melalui *browser* eksternal perangkat pengguna. Tampilan halaman *Home* dapat dilihat pada gambar 3.3.

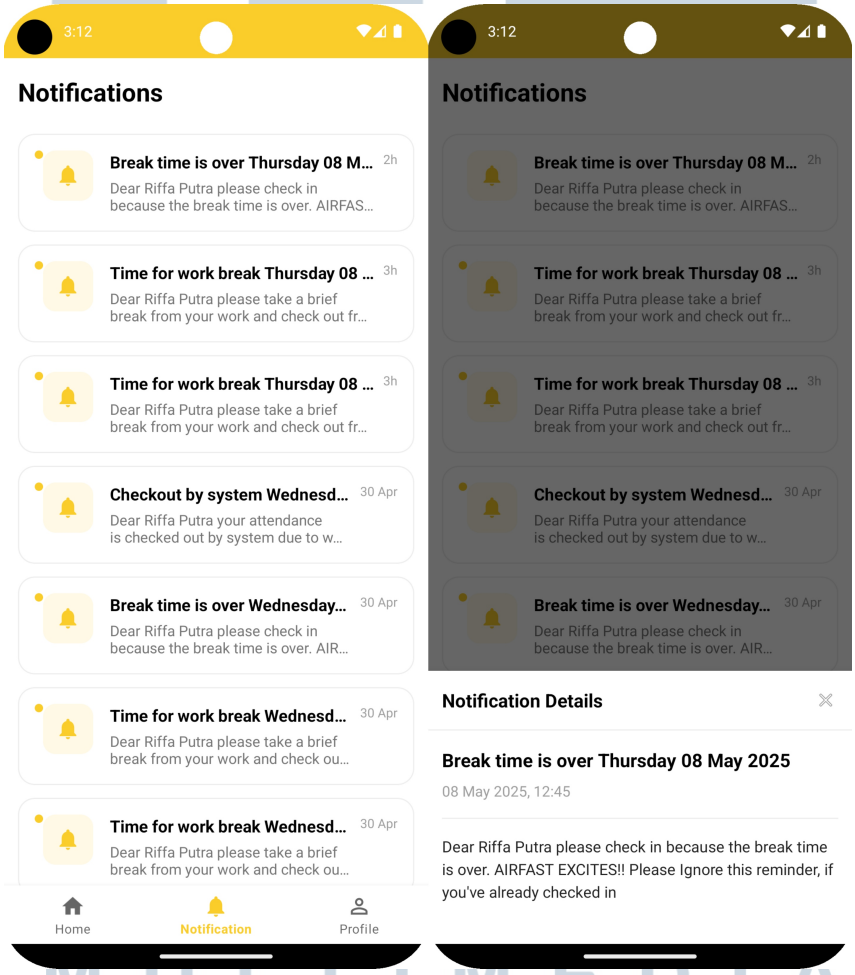


Gambar 3.3. Halaman Home

3. Halaman *Notification*

Halaman *Notification* berfungsi sebagai pusat riwayat notifikasi bagi pengguna. Sistem notifikasi pada aplikasi ini dirancang dengan dua mekanisme utama untuk memastikan pengiriman informasi yang baik. Pertama, menggunakan *push notification* melalui *Firebase Cloud Messaging (FCM)*, yang memungkinkan server mengirimkan pemberitahuan secara *real-time* ke perangkat pengguna, baik saat aplikasi sedang dibuka maupun ditutup.

Kedua, terdapat sebuah *background service* yang secara periodik mengambil data notifikasi dari API sebagai mekanisme sinkronisasi dan cadangan. Ketika pengguna membuka halaman *Notification*, aplikasi akan melakukan pemanggilan API baru untuk memuat seluruh riwayat notifikasi dan menampilkannya dalam bentuk daftar. Jika pengguna menekan salah satu item notifikasi pada daftar tersebut, sebuah *dialog bottom sheet* akan muncul dari bawah layar, menampilkan detail lengkap dari notifikasi yang dipilih, seperti judul, isi pesan, dan waktu pengiriman. Tampilan halaman *Notification* dan detail notifikasi dapat dilihat pada gambar 3.4.



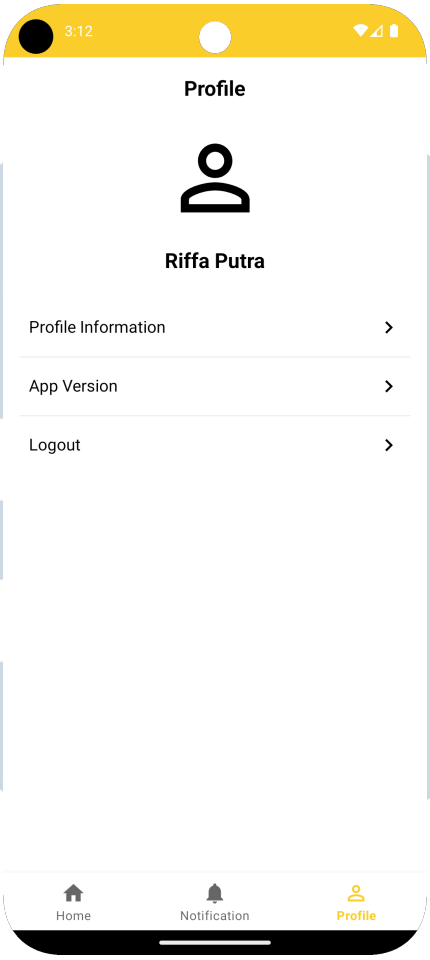
Gambar 3.4. Halaman *Notification*

4. Halaman Profil Pengguna

Halaman Profil Pengguna berfungsi sebagai pusat informasi personal dan manajemen sesi aplikasi. Berbeda dengan halaman lain yang dinamis,

halaman ini tidak melakukan pemanggilan API baru saat dimuat. Sebaliknya, informasi pengguna seperti nama, email, jabatan, dan divisi diambil langsung dari data yang telah tersimpan di *Session Manager* saat proses *login* awal. Pendekatan ini membuat pemuatan halaman menjadi lebih efisien. Halaman ini juga menyediakan beberapa menu, seperti ”Profile Information” yang akan menampilkan detail lengkap pengguna, dan ”App Version” untuk melihat versi aplikasi yang terpasang.

Fungsi vital pada halaman ini adalah tombol *logout*, yang memiliki logika bisnis spesifik. Sebelum proses *logout* dapat dilakukan, sistem akan memeriksa status absensi pengguna. Jika pengguna masih dalam status *”checked in”*, sistem akan menolak permintaan *logout* dan menampilkan dialog peringatan yang menginstruksikan pengguna untuk melakukan *check-out* terlebih dahulu. Jika tidak, proses *logout* akan dilanjutkan dengan menampilkan dialog konfirmasi, menghapus data sesi dari perangkat, dan mengarahkan pengguna kembali ke Halaman *Login*. Tampilan halaman Profil Pengguna dapat dilihat pada gambar 3.5.



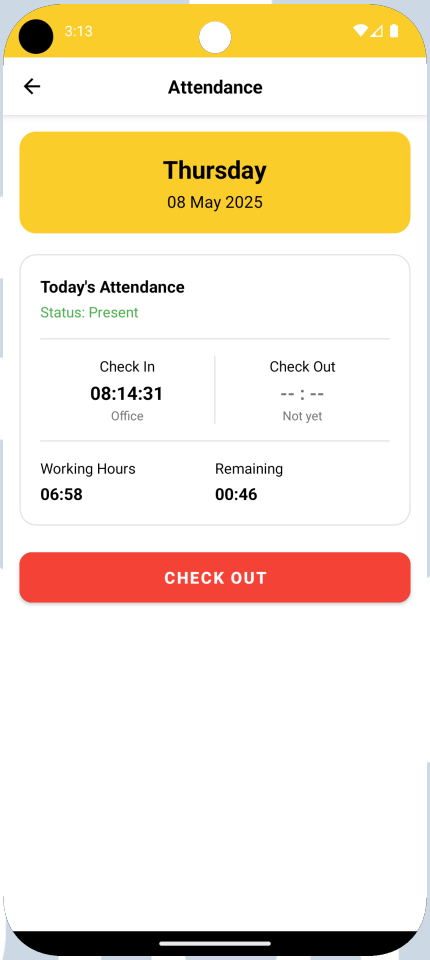
Gambar 3.5. Halaman Profil Pengguna

5. Halaman Absensi Pengguna

Halaman Absensi Pengguna merupakan fitur inti yang dirancang untuk melacak kehadiran karyawan secara akurat. Saat halaman ini diakses, aplikasi akan menampilkan status absensi terakhir pengguna, termasuk waktu *check-in* dan total jam kerja, berdasarkan data terbaru dari server. Proses absensi, baik *check-in* maupun *check-out*, mewajibkan akses lokasi perangkat untuk verifikasi.

Sebelum mengirim data ke API, sistem akan memastikan layanan lokasi aktif dan telah mendapatkan izin dari pengguna. Terdapat dua jenis alur *check-in*, yaitu *regular* untuk kehadiran di kantor dan *special occasion* untuk aktivitas seperti *work from home* atau dinas luar. Untuk *check-in regular* yang terlambat atau *check-in special occasion*, pengguna diwajibkan untuk mengisi kolom catatan sebagai justifikasi sebelum data dapat dikirim.

Setelah tindakan berhasil dilakukan, antarmuka akan diperbarui secara *real-time* untuk menampilkan status baru, mengubah tombol menjadi ”Check Out”, serta memulai atau menghentikan kalkulator jam kerja. Tampilan halaman Absensi Pengguna dapat dilihat pada gambar 3.6.



Gambar 3.6. Halaman Absensi Pengguna

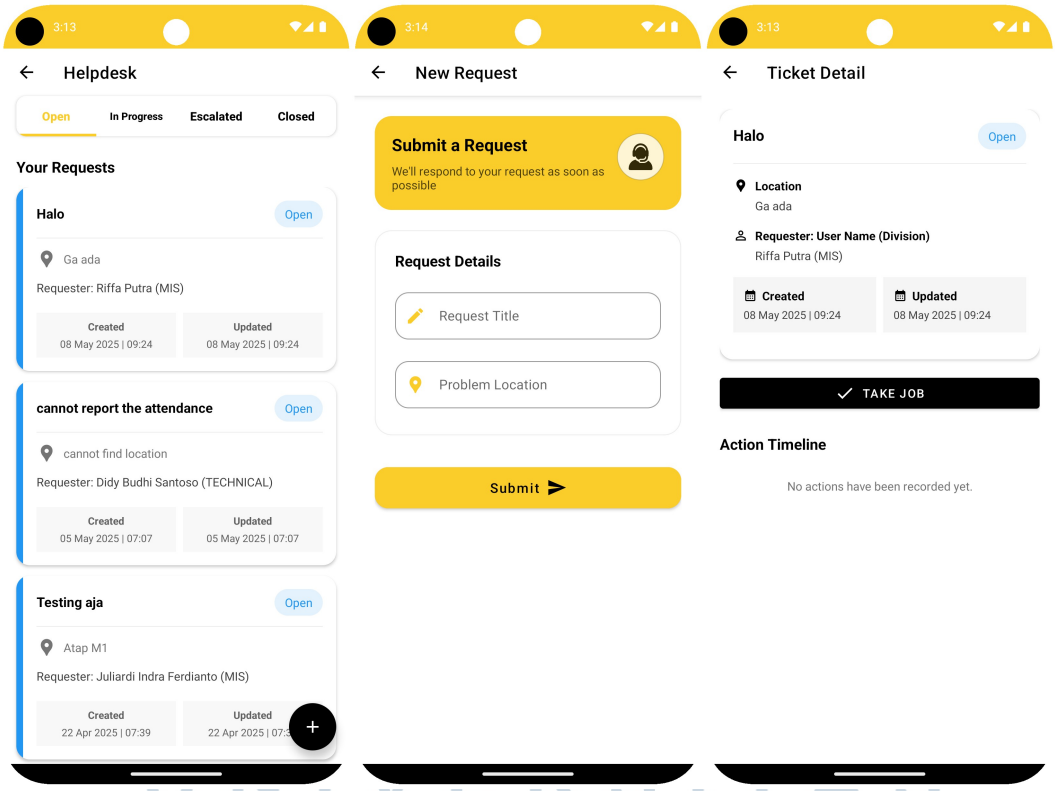
6. Halaman *Helpdesk*

Halaman *Helpdesk* dirancang sebagai sistem manajemen tiket terpusat untuk menangani permintaan dukungan teknis internal. Saat halaman ini diakses, aplikasi memanggil API untuk mengambil daftar tiket yang kemudian dapat disaring oleh pengguna berdasarkan statusnya melalui beberapa tab, yaitu *Open, In Progress, Escalated,* dan *Closed*.

Pengguna dapat membuat tiket bantuan baru melalui sebuah *floating action button*, namun sistem menerapkan aturan bisnis di mana seorang

pengguna tidak dapat membuat tiket baru jika masih memiliki tiket lain yang belum terselesaikan. Proses pengelolaan tiket memiliki alur kerja yang jelas. Petugas pendukung dapat melihat detail tiket, lalu mengambil pekerjaan dengan menekan tombol *"Take Job"*, yang akan mengubah status tiket menjadi *"In Progress"*.

Setelah pekerjaan selesai, petugas dapat memilih beberapa tindakan penyelesaian, seperti *"Close Job"* untuk menutup tiket, *"Escalate Job"* jika masalah memerlukan penanganan lebih lanjut. Setiap tindakan ini memanggil API yang sesuai untuk memperbarui status tiket secara *real-time* dan mencatatnya dalam riwayat aktivitas tiket. Tampilan halaman *Helpdesk* dapat dilihat pada gambar 3.7.



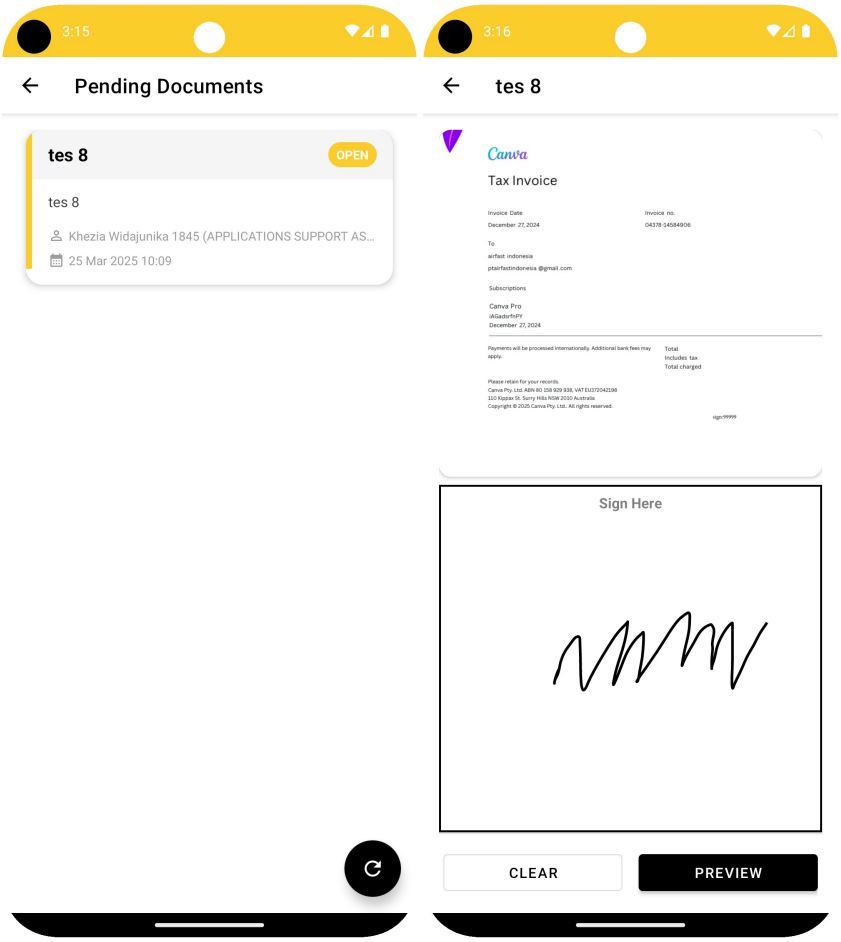
Gambar 3.7. Halaman Helpdesk

7. Halaman *Sign Document*

Halaman *Sign Document* menyediakan fungsionalitas bagi pengguna untuk menandatangani dokumen secara digital. Saat halaman ini diakses, aplikasi memanggil API untuk mendapatkan daftar dokumen yang memerlukan persetujuan dengan status *"Pending"*. Ketika pengguna memilih

salah satu dokumen, aplikasi akan membuka antarmuka penandatanganan yang terdiri dari dua komponen utama, yaitu sebuah *PDF viewer* di bagian atas yang mendukung fungsionalitas *zoom* dan sebuah area kanvas di bagian bawah untuk menggambar tanda tangan. Setelah pengguna membubuhkan tanda tangan, sistem menyediakan fitur *preview*.

Pada tahap ini, data gambar dari tanda tangan tersebut akan di-enkode menjadi format teks Base64. Proses enkoding ini sangat penting karena mengubah data biner gambar menjadi format teks yang aman dan standar untuk ditransmisikan melalui API ke server. Setelah di-enkode, gambar tanda tangan dari data Base64 tersebut akan ditampilkan di atas dokumen PDF sesuai posisinya agar pengguna dapat melihat hasil akhir sebelum melakukan submisi. Untuk menyelesaikan proses, pengguna menekan tombol *submit*, yang akan mengirimkan data tanda tangan dalam format Base64 beserta catatan ke server. Tampilan halaman *Sign Document* dapat dilihat pada gambar 3.8.



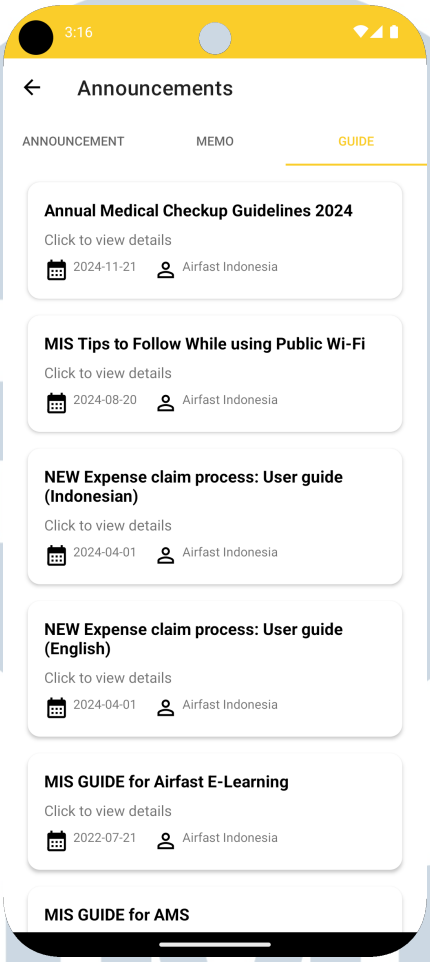
Gambar 3.8. Halaman Persetujuan Dokumen

8. Halaman *Announcement*

Halaman *Announcement* berfungsi sebagai papan pengumuman digital perusahaan. Saat halaman ini diakses, aplikasi melakukan pemanggilan API untuk mengambil daftar pengumuman, yang kemudian ditampilkan dan dikelompokkan ke dalam beberapa kategori seperti *"Announcement"*, *"Memo"*, dan *"Guide"* menggunakan navigasi tab. Pengguna dapat dengan mudah berpindah antar kategori untuk melihat informasi yang relevan.

Ketika pengguna memilih salah satu item pengumuman, aplikasi akan membuka sebuah *PDF viewer* untuk menampilkan dokumen terkait. Secara teknis, *viewer* ini diimplementasikan menggunakan komponen *Web View* yang memuat dokumen PDF melalui layanan eksternal Google Docs Viewer. Pendekatan ini memastikan dokumen dapat ditampilkan secara konsisten tanpa perlu mengintegrasikan pustaka *rendering* PDF asli yang kompleks ke

dalam aplikasi. Tampilan halaman *Announcement* dapat dilihat pada gambar 3.9.



Gambar 3.9. Halaman Announcement

3.3.4 Perangkat dan Dokumentasi

Dalam proses pengembangan aplikasi *mobile* untuk PT. AIRFAST Indonesia, digunakan serangkaian perangkat keras, perangkat lunak, serta dokumentasi resmi untuk mendukung kelancaran seluruh tahapan kerja. Berikut adalah rincian perangkat dan dokumentasi yang digunakan.

A Perangkat Keras (*Hardware*)

Perangkat keras yang digunakan terbagi menjadi dua sistem utama untuk mengakomodasi pengembangan pada platform Android dan iOS.

- Laptop HP Pavilion Gaming 15, digunakan sebagai mesin utama untuk pengembangan aplikasi versi Android dengan spesifikasi sebagai berikut:
  - Prosesor: AMD Ryzen 5 4500H
  - GPU: NVIDIA GeForce GTX 1650
  - RAM: 16 GB DDR4
  - Penyimpanan: 512 GB SSD
- Mac Mini M1, digunakan secara khusus untuk pengembangan, kompilasi, dan pengujian aplikasi versi iOS.

B Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan selama proses pengembangan adalah sebagai berikut:

- Android Studio Meerkat Feature Drop 2024.3.2: *Integrated Development Environment* (IDE) resmi dari Google yang digunakan untuk seluruh siklus pengembangan aplikasi Android, mulai dari perancangan antarmuka, penulisan kode, hingga proses *debugging* dan *build*.
- Xcode 16.0: *Integrated Development Environment* (IDE) resmi dari Apple yang digunakan untuk seluruh siklus pengembangan aplikasi iOS, mulai dari penulisan kode hingga proses *build* dan *debugging*.
- Postman: Platform untuk pengujian API (*Application Programming Interface*). Alat ini sangat penting untuk memastikan setiap *endpoint* dari server berfungsi sesuai ekspektasi sebelum diintegrasikan ke dalam aplikasi.
- GitLab: Platform berbasis web yang digunakan untuk manajemen repositori Git dan kontrol versi (*version control*). Seluruh kode sumber proyek disimpan, dikelola, dan dikolaborasikan melalui GitLab sesuai alur kerja perusahaan.

C Dokumentasi

Berikut adalah tautan dokumentasi resmi yang menjadi acuan utama dalam pengembangan aplikasi:

- Dokumentasi Kotlin: https://kotlinlang.org/docs/home.html
- Dokumentasi Swift: https://www.swift.org/documentation/
- Dokumentasi SwiftUI: https://developer.apple.com/documentation/swiftui
- Dokumentasi Retrofit: https://square.github.io/retrofit/

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Pada saat pengembangan aplikasi, beberapa kendala teknis dan non-teknis muncul yang mempengaruhi kelancaran operasional aplikasi. Kendala-kendala ini memerlukan perhatian khusus agar pengembangan dapat dilanjutkan dan aplikasi dapat berfungsi dengan baik di semua platform yang digunakan. Berikut adalah beberapa kendala yang ditemukan selama pengembangan aplikasi:

1. Terdapat pengguna yang mengeluhkan bahwa setiap kali *login*, akun seketika *logout* secara otomatis. Hal ini menyebabkan ketidaknyamanan bagi pengguna karena mereka tidak dapat mengakses aplikasi setelah *login*.
2. Menerapkan fitur *Time-based One-Time Password* (TOTP) untuk keamanan *multi*-faktor pada aplikasi iOS dan Android menghadapi kendala dalam proses verifikasi kode. Pada awalnya, kode TOTP yang dihasilkan tidak sesuai dengan ekspektasi, yang menyebabkan kegagalan dalam proses autentikasi.
3. Ketika integrasi API dilakukan, sering kali terjadi kesalahan sehingga data tidak berhasil diambil. Hal ini mengakibatkan beberapa fitur tidak berfungsi sebagaimana mestinya, karena aplikasi tidak dapat mengambil data yang dibutuhkan dari server.

3.4.2 Solusi

Setiap kendala yang ditemukan selama pengembangan aplikasi memerlukan solusi yang tepat untuk memastikan kelancaran proses pengembangan dan operasional aplikasi. Solusi yang diterapkan bertujuan untuk mengatasi masalah yang ada dan memperbaiki kualitas aplikasi secara keseluruhan. Berikut adalah solusi yang diterapkan untuk mengatasi kendala yang ditemukan:

1. Pengaturan *Date and Time* di device yang digunakan oleh pengguna tidak diatur secara otomatis, sehingga terjadi perbedaan waktu antara perangkat dan server. Perbedaan waktu ini menyebabkan TOTP tidak bekerja dengan baik. Solusinya adalah memastikan bahwa pengaturan *Date and Time* di perangkat pengguna diaktifkan untuk disesuaikan secara otomatis, yang akan memperbaiki sinkronisasi waktu antara perangkat dan server.

2. Memeriksa kembali implementasi algoritma HMAC-SHA512 yang digunakan dalam pembuatan kode TOTP dan memastikan bahwa pengaturan waktu pada perangkat *mobile* selaras dengan waktu server. Selain itu, dilakukan *debugging* untuk memastikan bahwa parameter yang benar diterapkan dalam pembuatan kode TOTP, sehingga proses autentikasi berjalan dengan lancar dan kode yang dihasilkan sesuai dengan ekspektasi.

3. Memeriksa kembali parameter dan tipe data yang diterima dari API. Penyamaan parameter dan tipe data yang digunakan dalam API juga dilakukan untuk memastikan bahwa data yang dikirim dan diterima sesuai format yang diharapkan. Dengan demikian, integrasi API dapat dilakukan dengan sukses dan aplikasi dapat mengambil data dengan benar.