BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama melaksanakan magang di PT. Gajah Terbang Kreatif, sebagai *Web Developer Intern* yang bertugas mengembangkan aplikasi *GodPlan*. Pekerjaan dilakukan dibawah bimbingan dan pengawasan langsung dari Bapak Vincent Lim sebagai Direktur perusahaan.

Tugas utamanya adalah membuat versi paling dasar (*MVP*) dari aplikasi *GodPlan* yang berfokus pada sistem absensi terlebih dahulu. Rencana pengembangannya bertahap, dimana setelah sistem absensi berjalan baik, akan dilanjutkan dengan modul *CRM* untuk mengelola pelanggan dan modul tugas, yang nantinya semua akan disatukan dalam satu *login* saja.

Proses pengerjaan dilakukan secara mandiri, dimulai dari analisis kebutuhan, perancangan sistem, pemrograman, pengujian, hingga *deployment*. Komunikasi dengan pembimbing dilakukan secara rutin baik secara langsung (*offline*) maupun melalui *Google Meet*, dengan pertemuan mingguan untuk melaporkan perkembangan dan berdiskusi mengenai kendala yang dihadapi.

Pekerjaan dilakukan dengan sistem *hybrid*, dapat bekerja dari kantor atau dari rumah setelah melakukan koordinasi terlebih dahulu. Seluruh kode program dikembangkan dan diverifikasi secara mandiri dengan melakukan *testing* yang komprehensif untuk memastikan kualitas sesuai dengan kebutuhan sistem.

3.1.1 Alur Koordinasi Web Developer di PT. Gajah Terbang Kreatif

Dalam pelaksanaan tugas teknis yang lebih kompleks, koordinasi dilakukan langsung dengan Bapak Vincent Lim selaku Direktur, terutama untuk keperluan diskusi pengembangan lanjutan dan persetujuan fitur-fitur yang akan diimplementasikan pada proyek *GodPlan*.

Sistem kerja yang diterapkan bersifat *hybrid* dengan fleksibilitas tinggi. Meskipun memiliki *basecamp* kerja utama di kantor, perusahaan memberlakukan kebijakan *Work from Anywhere* (WFA) yang memungkinkan pelaksanaan tugas secara *online* dengan izin dan koordinasi sebelumnya. Komunikasi dan koordinasi proyek dilakukan secara intensif melalui pertemuan langsung dan *Google Meet*, dengan pertemuan mingguan untuk melaporkan progres dan mengatur prioritas

tugas berikutnya.

Alur koordinasi dalam diagram tersebut menggambarkan proses kerja sistematis di bawah koordinasi Direktur. Proses dimulai dari tahap penerimaan tugas yang diberikan langsung oleh Bapak Vincent Lim sebagai Direktur, kemudian dilanjutkan dengan analisis kebutuhan serta studi dokumentasi untuk memahami ruang lingkup pekerjaan yang akan dilaksanakan. Tahap ini memastikan untuk memahami dengan jelas apa yang dibutuhkan, baik dari sisi teknis maupun fungsional, sebelum masuk ke tahap implementasi fitur.

Selanjutnya, implementasi dilakukan dalam bentuk pengembangan fitur menggunakan teknologi seperti *Next.js* dan *React* untuk *frontend*, serta *Go language* untuk *backend* yang terintegrasi dengan *database PostgreSQL* pada *Aiven*. Setelah fitur dikembangkan, dilakukan *testing* lokal sebelum dilakukan diskusi dan presentasi hasil kepada Direktur. Jika ditemukan kekurangan atau ketidaksesuaian, maka proses revisi akan dilakukan. Proses ini bisa berulang hingga hasil implementasi dinyatakan sesuai, setelah itu barulah dilakukan *deployment* ke *Vercel*. Alur ini mencerminkan pola kerja yang terstruktur yang melibatkan komunikasi aktif dengan pimpinan guna menjamin kualitas dan ketepatan implementasi aplikasi *GodPlan*.

3.1.2 Alur Kerja Web Developer di PT. Gajah Terbang Kreatif

Sebagai *web developer* pada tim pengembangan *GodPlan*, alur koordinasi kerja dimulai dari penerimaan *task* yang diberikan langsung oleh Bapak Vincent Lim sebagai Direktur. Setelah menerima *task*, langkah selanjutnya adalah melakukan analisis kebutuhan dan studi dokumentasi untuk memahami konteks dan spesifikasi teknis fitur yang akan dikembangkan. Proses ini penting agar implementasi sesuai dengan kebutuhan sistem yang telah dirancang.

Setelah tahap analisis selesai, pengerjaan implementasi fitur melalui proses *coding*, baik pada sisi *frontend* menggunakan *Next.js* dan *React*, maupun *backend* menggunakan *Go language* yang terintegrasi dengan *database PostgreSQL* pada *Aiven*. Setelah fitur selesai dikembangkan, masuk tahap *testing* lokal sebelum mempresentasikan hasilnya kepada Direktur.

Jika terdapat masukan atau revisi dari hasil diskusi dengan Direktur, maka alur kerja kembali ke tahap implementasi untuk penyempurnaan fitur. Proses ini terus berulang hingga fitur dinyatakan sesuai dan selesai, kemudian dilakukan *deployment* ke *Vercel*. Alur kerja ini menunjukkan bahwa koordinasi

yang dilakukan bersifat iteratif, *developer* dapat mendengarkan kebutuhan, mengimplementasikan solusi, serta merespons umpan balik secara cepat demi mendukung kelancaran pengembangan aplikasi *GodPlan*.

3.2 Tugas yang Dilakukan

Selama melaksanakan kegiatan magang di PT. Gajah Terbang Kreatif selama 4 bulan, tugas utama difokuskan pada pengembangan aplikasi *web* progresif *GodPlan* yang mencakup sistem absensi. Pengembangan dilakukan menggunakan *tech stack* modern yang meliputi *Next.js* untuk *frontend*, *Go language* untuk *backend*, *PostgreSQL* sebagai *database*, serta *Docker* untuk *containerization*.

Tahapan pekerjaan dimulai dengan melakukan analisis kebutuhan sistem melalui studi mendalam mengenai *workflow* perusahaan dan kebutuhan pengguna. Proses ini meliputi identifikasi fitur-fitur esensial untuk modul absensi sebagai prioritas pengembangan pertama. Lingkungan pengembangan disiapkan menggunakan *Docker container* untuk memastikan konsistensi lingkungan antara *development* dan *production*.

Implementasi teknis diawali dengan pembuatan *API documentation* menggunakan *Swagger* untuk memastikan spesifikasi yang jelas antara *frontend* dan *backend*. Pada sisi *frontend*, dikembangkan *user interface* yang responsif menggunakan *Next.js* dan *React* dengan pendekatan *component-based architecture*. Sementara pada sisi *backend*, dibangun *RESTful API* menggunakan *Go language* dengan integrasi *database PostgreSQL* yang dihosting pada *Aiven*.

Seluruh fitur diuji secara komprehensif dalam lingkungan lokal sebelum dilakukan *deployment* ke *Vercel*. Untuk memastikan kualitas kode, dilakukan *testing* secara berkala serta penerapan *continuous integration* untuk otomasi *testing*.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kegiatan kerja magang dilaksanakan di PT. Gajah Terbang Kreatif sebagai *Web Developer Intern* dengan durasi 4 bulan. Proyek pengembangan aplikasi *GodPlan* dilakukan secara mandiri dengan koordinasi langsung kepada Direktur, dengan fokus utama pada implementasi MVP sistem absensi.

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

Minggu Ke-	**Pekerjaan yang Dilakukan**
1	Memahami kebutuhan sistem absensi dan melakukan analisis *requirements* untuk aplikasi *GodPlan*, serta mempelajari *tech stack* yang digunakan (*Next.js, Go, PostgreSQL*)
2	Menerapkan desain *UI/UX* pada halaman autentikasi (*login, register*) dan *dashboard* utama menggunakan *Next.js* dan *Tailwind CSS*
3	Menerapkan desain *UI/UX* untuk modul absensi meliputi halaman *check-in, check-out*, dan riwayat presensi karyawan
4	Melakukan *setup environment development* dengan *Docker container* dan konfigurasi *database PostgreSQL* pada *Aiven*
5	Membuat *API documentation* menggunakan *Swagger* dan implementasi *endpoint* autentikasi (*login, register*) dengan *JWT*
6	Implementasi *backend service* untuk modul absensi dengan *Go language* termasuk fitur *geolocation tracking*
7	Integrasi *Google Maps API* untuk validasi lokasi absensi dan implementasi validasi radius kantor
8	Pengembangan fitur riwayat absensi dan *dashboard* laporan kehadiran karyawan
9	Implementasi sistem notifikasi untuk konfirmasi absensi berhasil/gagal
10	Membuat modul *reporting* untuk *generate* laporan absensi harian, mingguan, dan bulanan
11	Implementasi *role-based access control* untuk pembagian akses administrator dan karyawan
12	Optimasi performa aplikasi dengan *database indexing* dan *query optimization*
13	Melakukan *comprehensive testing* pada sistem absensi meliputi *unit test* dan *integration test*
14	*Deployment* aplikasi ke *Vercel* dan konfigurasi *environment* produksi

Minggu Ke-	Pekerjaan yang Dilakukan
15	Pembuatan dokumentasi teknis dan *user manual* untuk modul absensi *GodPlan*
16	Finalisasi fitur absensi dan persiapan presentasi hasil kerja magang

3.3.1 Analisis Kebutuhan Sistem

Tahap awal pengembangan aplikasi *GodPlan* dimulai dengan melakukan analisis kebutuhan sistem yang komprehensif. Proses ini bertujuan untuk mengidentifikasi dan memahami kebutuhan pengguna secara mendalam, sehingga fitur dan fungsionalitas yang dikembangkan dapat sesuai dengan ekspektasi dan alur kerja perusahaan [1].

Analisis dilakukan melalui diskusi langsung dengan Direktur perusahaan dan observasi terhadap proses bisnis yang berjalan. Dari hasil analisis tersebut, teridentifikasi beberapa kebutuhan pengguna utama yang menjadi fokus pengembangan.

Pertama, administrator membutuhkan kemampuan untuk mengelola data karyawan secara lengkap, mulai dari proses penambahan, pengeditan, hingga penghapusan data. Selain itu, administrator juga perlu dapat memantau presensi karyawan secara *real-time* [2], menghasilkan berbagai laporan absensi sesuai periode yang diinginkan, serta mengkonfigurasi pengaturan sistem sesuai kebutuhan perusahaan.

Kedua, dari sisi karyawan, terdapat kebutuhan untuk dapat melakukan absensi *check-in* dan *check-out* dengan mudah dan akurat menggunakan teknologi *Progressive Web App* yang memungkinkan akses melalui perangkat mobile [3]. Karyawan juga perlu dapat melihat riwayat presensi pribadi serta mengelola tugas harian melalui fitur *to-do list* yang terintegrasi dalam sistem aplikasi [4].

Ketiga, manajer memerlukan akses untuk memantau kinerja tim yang menjadi tanggung jawabnya. Fitur yang dibutuhkan meliputi kemampuan untuk menyetujui atau menolak pengajuan cuti dari anggota tim, serta mengelola penugasan pekerjaan dengan efektif melalui sistem yang terpusat [5].

3.3.2 Studi Literatur dan Teknologi

Sebelum memulai implementasi, dilakukan studi literatur dan teknologi untuk menentukan *tech stack* yang paling sesuai dengan kebutuhan proyek. Studi ini mencakup penelitian terhadap berbagai teknologi modern yang dapat mendukung pengembangan aplikasi *web* progresif yang *scalable* dan *maintainable* [6].

Untuk *frontend*, dipilih *Next.js* dengan *React* karena kemampuannya dalam menyediakan *server-side rendering* yang dapat meningkatkan performa dan optimasi *SEO* [7]. Framework ini telah terbukti efektif dalam pengembangan aplikasi web modern yang membutuhkan responsivitas tinggi [8]. *Tailwind CSS* dipilih sebagai *framework CSS* karena utilitasnya yang lengkap dan kemudahan dalam melakukan kustomisasi desain.

Pada sisi *backend*, *Go language* dengan *framework Gin* dipilih karena performanya yang tinggi dalam menangani *concurrent request*, serta kemudahan dalam pengembangan *API* [9]. Bahasa *Go* juga dikenal dengan kesederhanaan sintaksis dan kemampuan *memory management* yang efisien.

Untuk penyimpanan data, *PostgreSQL* dipilih sebagai *database* utama karena kemampuannya dalam menangani data yang terstruktur dengan baik dan fitur *ACID compliance* yang menjamin konsistensi data. Pemilihan teknologi ini didasarkan pada kebutuhan sistem presensi yang memerlukan keandalan tinggi dalam pengelolaan data [10].

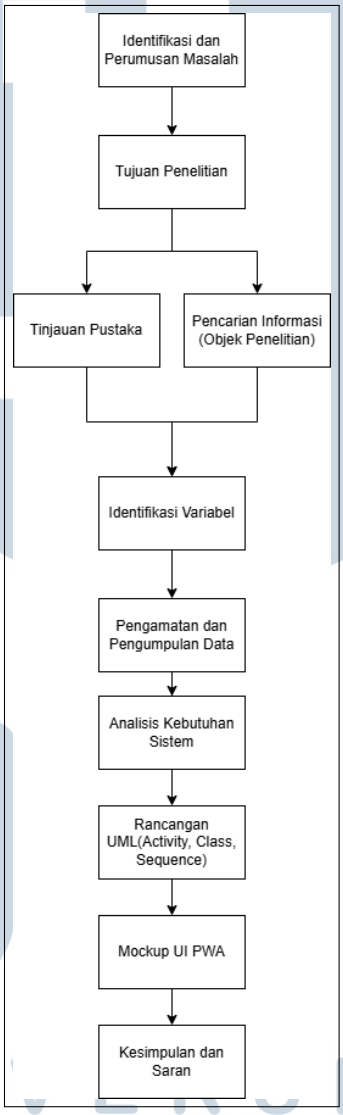
3.3.3 Perancangan Arsitektur Sistem

Aplikasi *GodPlan* dirancang dengan menggunakan pendekatan arsitektur *three-tier* yang membagi sistem menjadi tiga lapisan utama. Pemilihan arsitektur ini didasarkan pada pertimbangan skalabilitas, *maintainability*, dan keamanan sistem. Untuk memperjelas desain sistem, berikut adalah diagram UML yang menggambarkan rancangan sistem.

A *Research Metodology*

Gambar 3.1 merupakan diagram metodologi penelitian ini menggambarkan alur sistematis yang dimulai dari identifikasi masalah serta penetapan tujuan penelitian, yang kemudian didukung oleh tinjauan pustaka dan pencarian informasi objek penelitian sebagai landasan teori. Proses dilanjutkan dengan identifikasi variabel, pengamatan, serta pengumpulan data untuk melakukan analisis kebutuhan

sistem guna merancang aplikasi berbasis *Progressive Web App* (PWA). Berdasarkan analisis tersebut, dilakukan perancangan menggunakan *UML (Activity, Class, Sequence Diagram)* untuk memodelkan logika *backend* Golang dan *frontend* Next.js, pembuatan *mockup UI*, hingga diakhiri dengan penarikan kesimpulan serta saran.

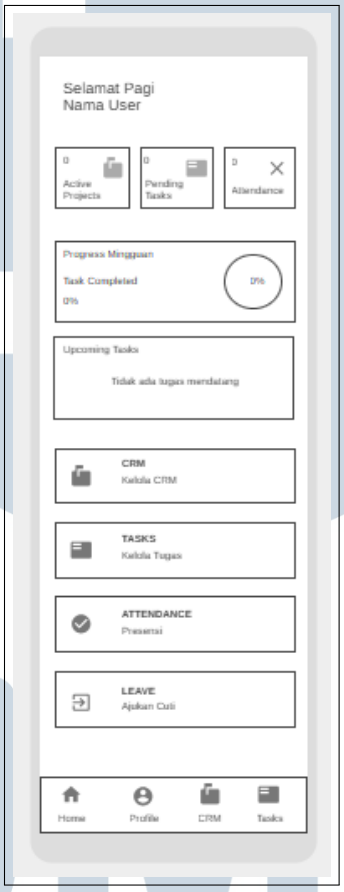


Gambar 3.1. *Research Metodology*

B *Wireframe*

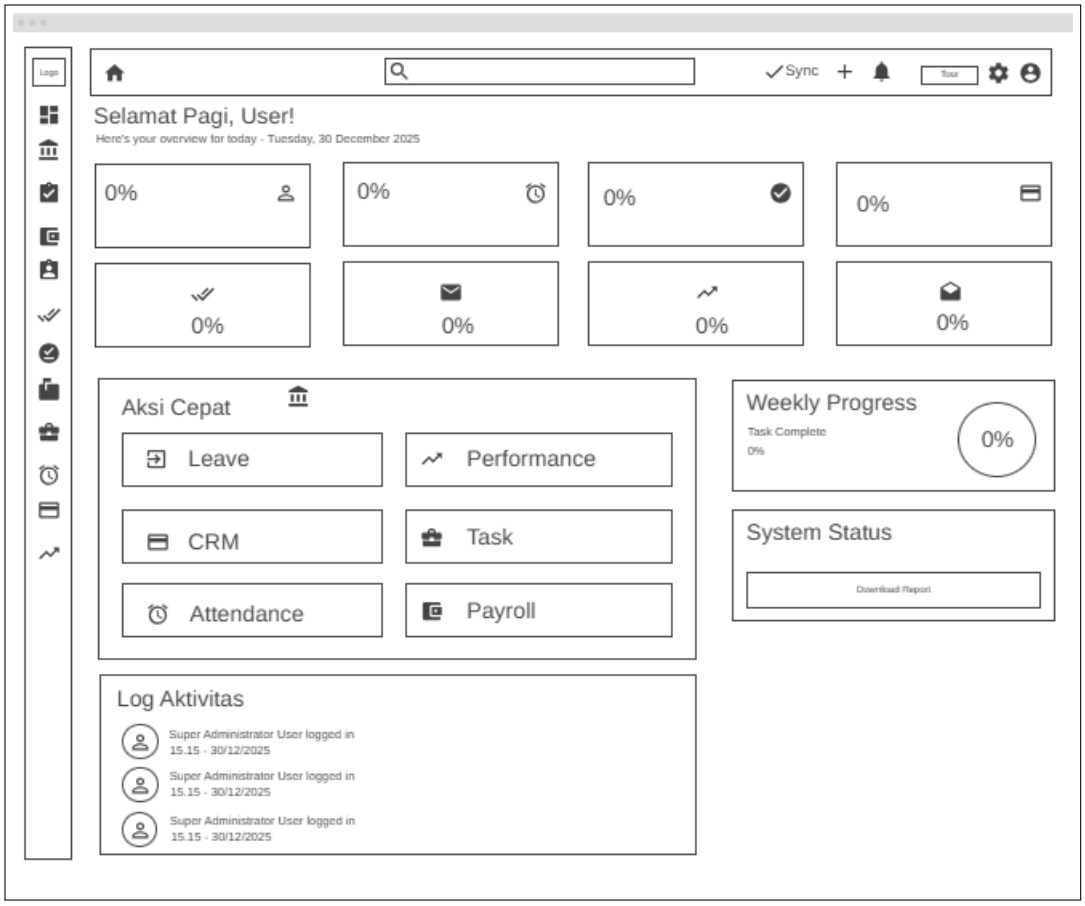
Gambar 3.2 merupakan *Dashboard* versi *Mobile* yang mengadaptasi *main feature* ke dalam *responsive layout* vertikal untuk penggunaan di perangkat *smartphone*. Bagian atas menonjolkan ringkasan *project status* dan *attendance*,

diikuti oleh grafik *weekly progress* yang memberikan informasi *visual* cepat mengenai penyelesaian *task*. Navigasi pada versi *mobile* ini disederhanakan melalui *bottom navigation bar* yang mencakup *menu Home*, *Profile*, *CRM*, dan *Tasks*, serta *menu card* berukuran besar di tengah layar yang memudahkan *user* melakukan *tap* saat mengelola *CRM*, *task*, *attendance*, maupun pengajuan *leave* di layar yang lebih kecil.



Gambar 3.2. *Wireframe Mobile*

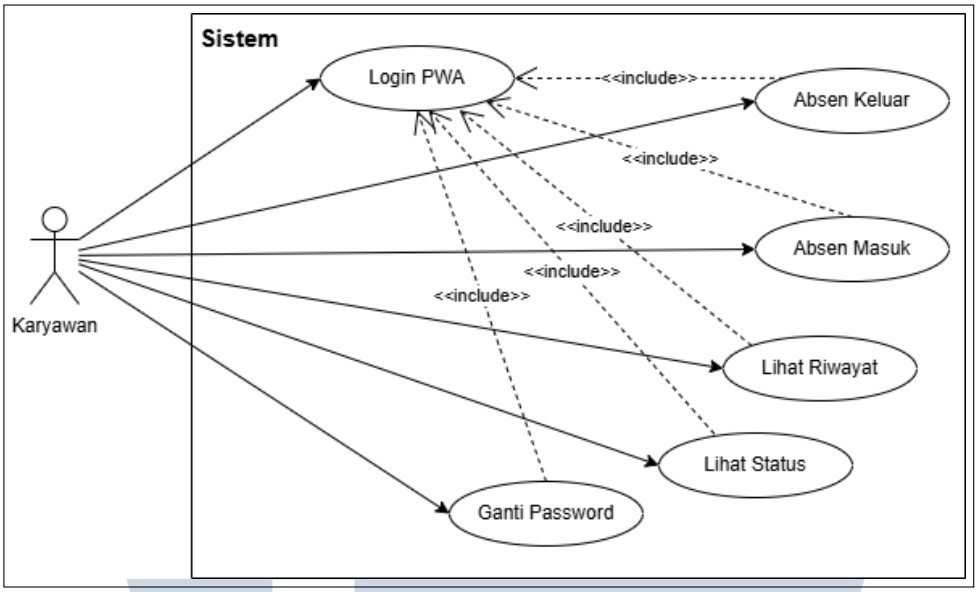
Gambar 3.3 menampilkan *interface Dashboard* versi *Desktop* yang dirancang sebagai pusat kendali *management* kerja bagi *user*. Di bagian atas, terdapat statistik ringkas dalam bentuk *scorecards* yang menunjukkan data persentase secara *real-time*, sementara di sisi kiri terdapat *navigation bar* (*sidebar*) yang padat dengan berbagai ikon *feature* untuk *quick access*. Area utama *dashboard* mencakup bagian *"Quick Action"* yang memungkinkan *user* melakukan navigasi instan ke *module* penting seperti *Leave*, *Attendance*, dan *Payroll*, serta kolom *"Activity Log"* di bagian bawah yang berfungsi sebagai *audit trail* untuk memantau riwayat *login user*.



Gambar 3.3. *Wireframe Dekstop*

C *Use Case* Diagram Karyawan

Diagram *Use Case* Karyawan ini memodelkan interaksi *Employee* dengan *System*. Sebelum *Employee* dapat menggunakan empat fungsi utama *Clock In*, *Clock Out*, *View History*, atau *View Status* mereka diwajibkan untuk melalui proses *Authentication* yang diwakili oleh *PWA Login*. Kewajiban ini berarti *PWA Login* adalah langkah yang selalu harus dieksekusi pertama kali sebagai prasyarat teknis untuk memastikan identitas pengguna*Employee* tidak akan bisa mengakses fungsi-fungsi tersebut jika belum berhasil *login*. Selain empat fungsi tersebut, *Employee* juga memiliki akses untuk *Change Password* setelah berhasil melakukan *authentication*.



Gambar 3.4. Use Case Diagram Karyawan

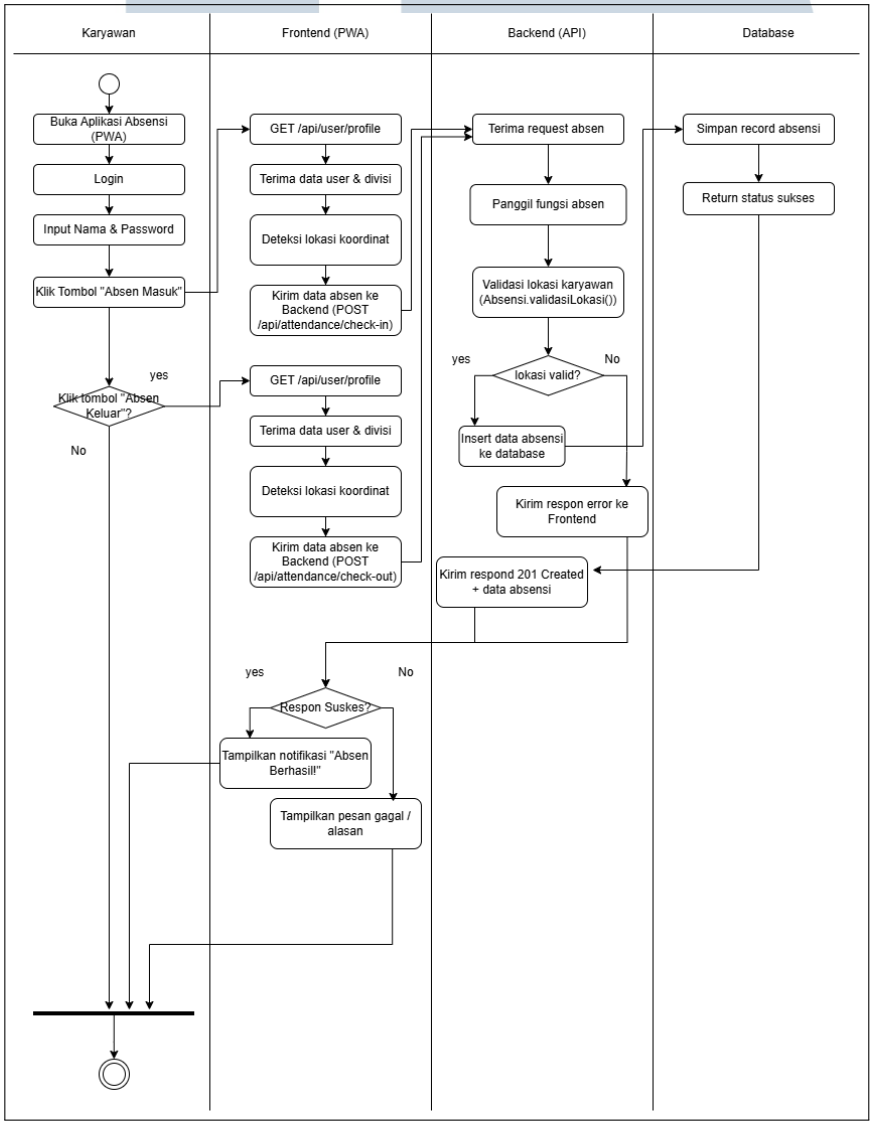
D Activity Diagram Absen Masuk

Gambar 3.5 merupakan proses absen yang prosesnya dimulai ketika Karyawan membuka aplikasi *Frontend (PWA)*, memasukkan kredensial untuk *login*, dan menekan tombol ”Absen Masuk”. Terdapat juga percabangan keputusan di mana jika pengguna memilih tombol lain, alur akan beralih ke proses ”Absen Keluar”. Setelah tombol ditekan, sistem *Frontend* secara otomatis melakukan *request* ke *endpoint GET /api/user/profile* untuk mengambil data profil dan divisi pengguna, lalu dilanjutkan dengan proses deteksi koordinat lokasi terkini. Setelah semua informasi terkumpul, *Frontend* mengirimkan data tersebut ke *Backend* menggunakan metode *POST* ke alamat *api/attendance/check-in* (untuk masuk) atau *api/attendance/check-out* (untuk keluar).

Di sisi *Backend (API)*, sistem menerima *request* absensi tersebut dan segera memanggil fungsi eksekusi absen. Tahap paling kritis di sini adalah validasi keamanan lokasi menggunakan fungsi *Absensi.validasiLokasi()*. Sistem akan mengecek apakah koordinat yang dikirim valid; jika kondisi *lokasi valid* tidak terpenuhi (bernilai *No*), *Backend* akan langsung mengirimkan respon *error* ke *Frontend* tanpa menyimpan data. Sebaliknya, jika lokasi valid (bernilai *yes*), *Backend* akan meneruskan perintah ke *Database* untuk melakukan *insert data absensi* guna pencatatan permanen.

Pada *swimlane Database*, sistem menyimpan *record* absensi dan

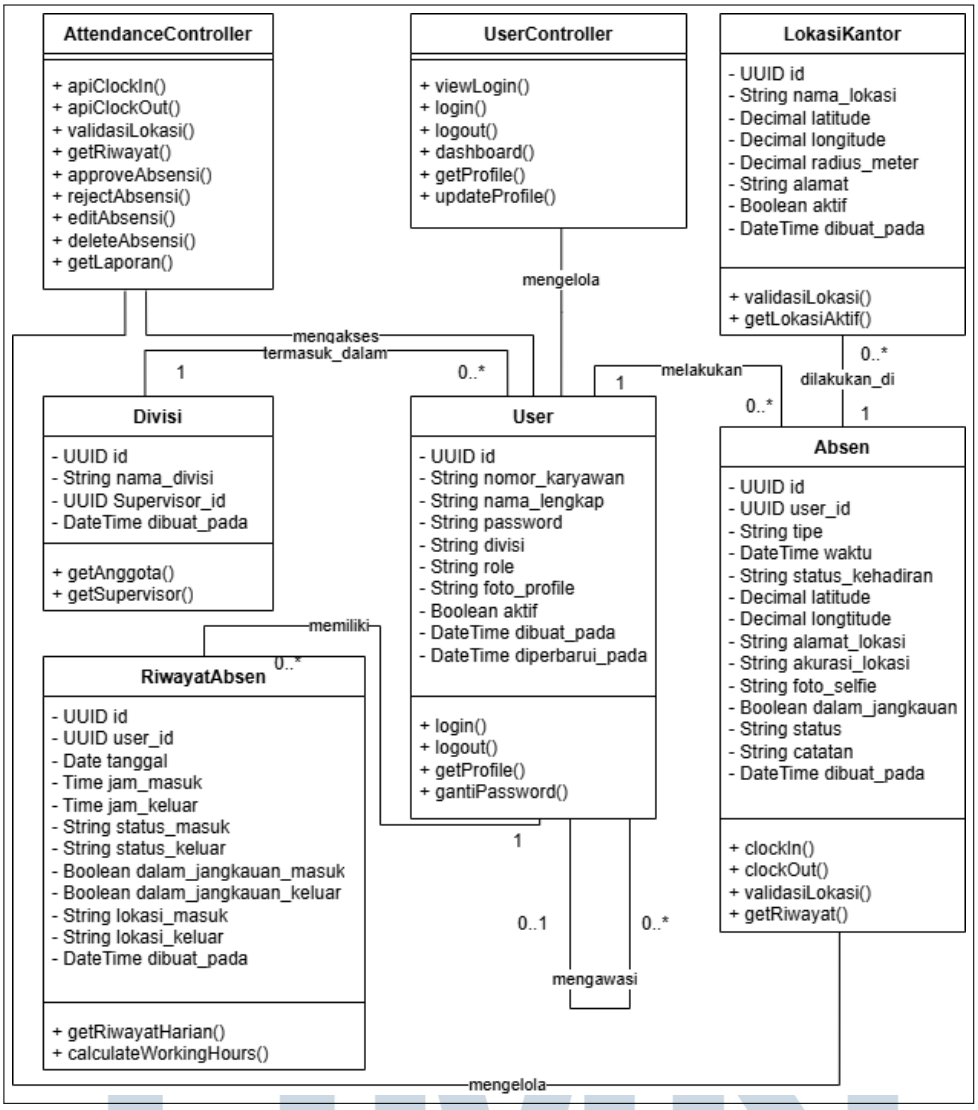
mengembalikan status sukses ke server. Setelah menerima konfirmasi dari *Database*, *Backend* mengirimkan respon final berupa status *HTTP 201 Created* beserta data absensi kembali ke *Frontend*. Aplikasi kemudian mengevaluasi respon tersebut, jika *Respon Sukses*, layar akan menampilkan notifikasi ”Absen Berhasil!”, namun jika gagal, sistem akan menampilkan pesan kesalahan atau alasannya kepada Karyawan. Seluruh rangkaian aktivitas ini kemudian berakhir pada *final node*, menandakan proses absensi telah selesai.



Gambar 3.5. Activity Diagram Absen

E Class Diagram Sistem

Diagram Kelas ini memodelkan alur data sistem absensi menggunakan tujuh entitas utama yang terstruktur secara spesifik. Entitas pusat adalah *User*, yang menyimpan identitas, peran (*role*), dan *password*, serta memiliki relasi satu-ke-banyak dengan *Divisi* untuk pengelompokan. Setiap *User* menghasilkan banyak catatan *Absen* yang merekam detail *real-time* (*tipe* dan lokasi) dan banyak *RiwayatAbsen* yang merangkum data absensi per hari, termasuk *calculateWorkingHours()*. Pencatatan *Absen* dan *RiwayatAbsen* ini dilakukan di lokasi yang divalidasi oleh entitas *LokasiKantor*, yang menyimpan koordinat dan radius yang sah. Seluruh interaksi fungsionalitas sistem dikendalikan oleh dua pengontrol: *UserController* mengelola proses otentikasi (*login()*) dan pembaruan profil (*updateProfile()*) untuk entitas *User* sementara *AttendanceController* bertanggung jawab penuh atas semua proses absensi yang sensitif, seperti memproses masukan absensi (*apiClockIn()*), memverifikasi lokasi (*validasiLokasi()*), memberikan persetujuan (*approveAbsensi()*), hingga menghasilkan laporan (*getLaporan()*).

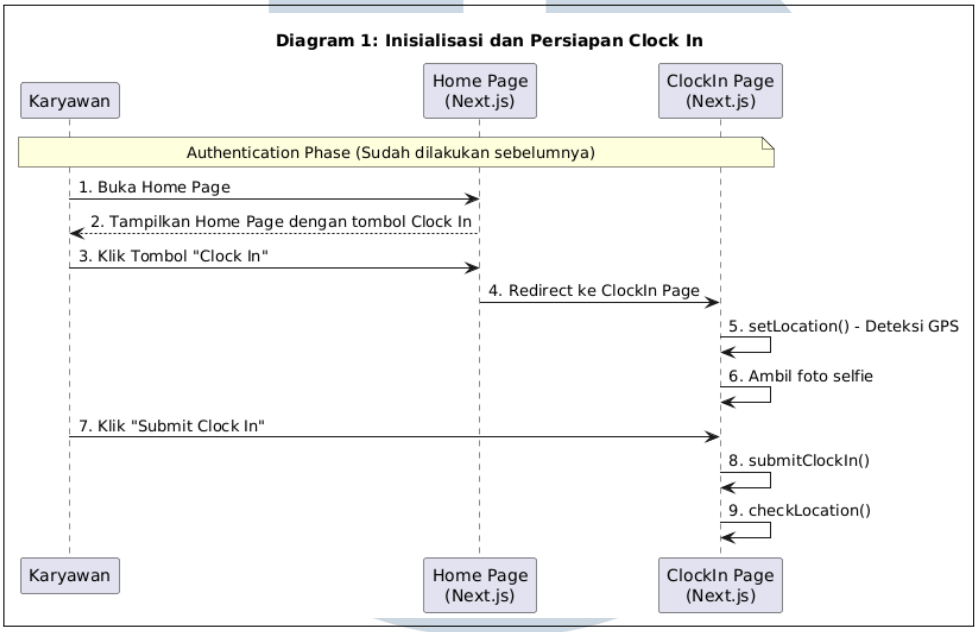


Gambar 3.6. Class Diagram Sistem Absensi GodPlan

F Sequence Diagram 1 Inisialisasi dan Persiapan Clock In

Diagram urutan ini memvisualisasikan tahapan inisiasi fungsionalitas *Clock In* yang dilakukan oleh *Karyawan* setelah berhasil melewati fase otentikasi. Proses diawali dengan akses *Karyawan* terhadap *Home Page*, yang menampilkan opsi untuk memulai absensi. Respon terhadap *trigger* Klik Tombol *Clock In* pada langkah 3 adalah pengalihan *client* menuju *ClockIn Page* pada langkah 4. Halaman tujuan segera memulai proses persiapan data dengan memanggil fungsi *setLocation()* untuk deteksi lokasi *GPS* dan menginstruksikan *Karyawan* untuk mengambil foto *selfie* sebagai validasi biometrik. Setelah data

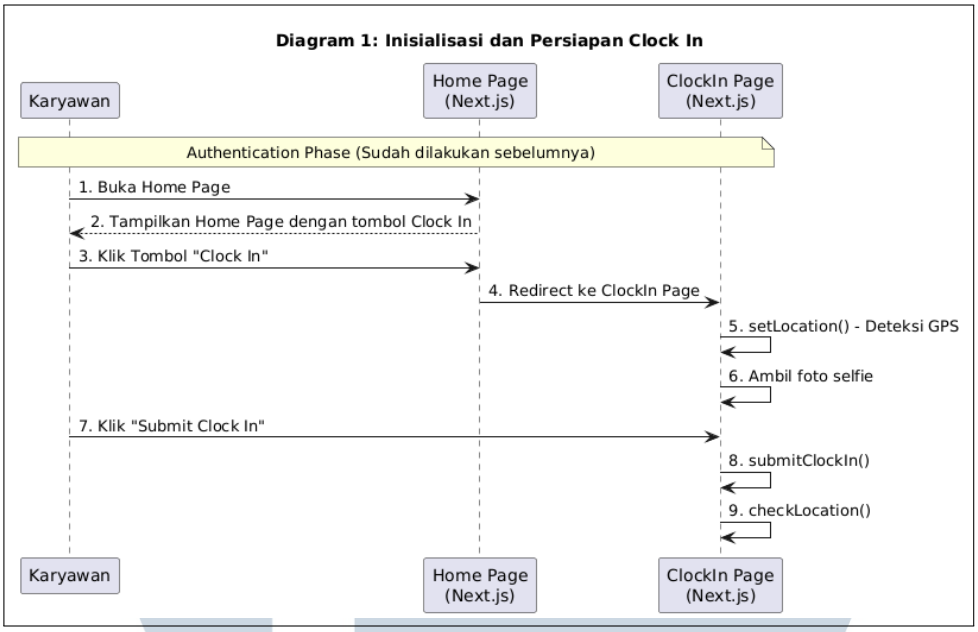
pendukung terpenuhi, *trigger* Klik *Submit Clock In* akan memicu eksekusi metode *submitClockIn()* dan dilanjutkan dengan *checkLocation()*, yang krusial untuk memverifikasi keabsahan geografis posisi *Karyawan* terhadap titik koordinat kantor yang telah ditetapkan sebelum *log* absensi dicatatkan.



Gambar 3.7. Sequence Diagram 1 Inisialisasi dan Persiapan Clock In

G Sequence Diagram 2 Validasi Lokasi

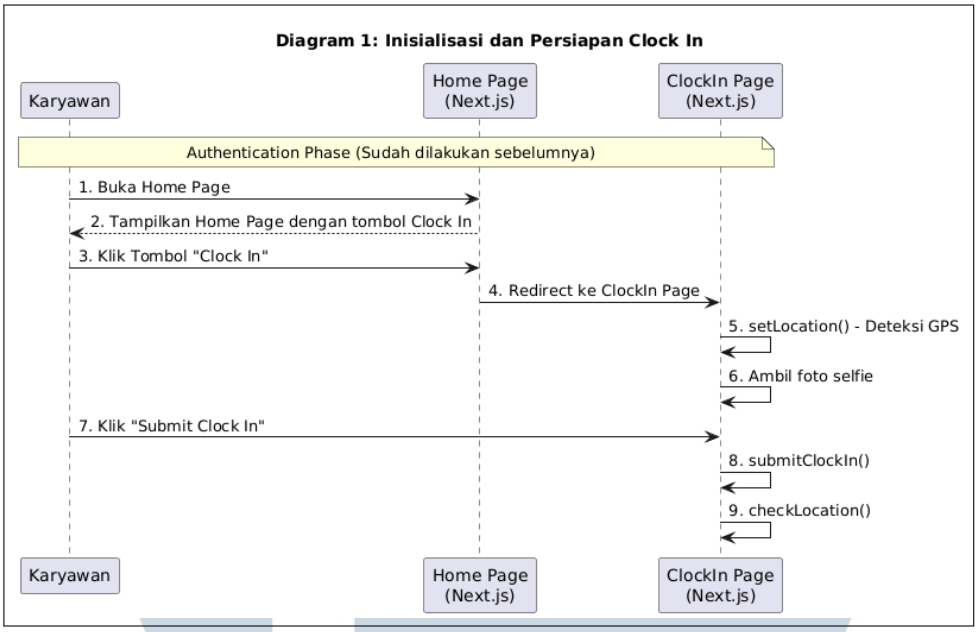
Diagram urutan ini menguraikan proses krusial *Validasi Lokasi* yang terjadi setelah *Karyawan* mengajukan permintaan *Clock In*. Proses dimulai ketika *ClockIn Page* mengirimkan permintaan *POST* dengan data *latitude*, *longitude*, dan *user_id* ke *AttendanceController* (langkah 1). *AttendanceController* kemudian memproses permintaan ini dengan memanggil metode *validasiLokasi()* pada entitas *LokasiKantor* (langkah 2), yang memberikan *response* status lokasi (*Within Range* atau *Out of Range*) kembali ke *AttendanceController* (langkah 3). Berdasarkan hasil validasi tersebut, diterapkan *fragment alt*. Jika lokasi teridentifikasi *Within Range*, *AttendanceController* merespons dengan *200 OK* dan status *Lokasi Valid*, dan proses dilanjutkan ke *Diagram 3A*. Sebaliknya, jika lokasi *Out of Range*, *AttendanceController* merespons dengan *200 OK* dan pesan *Butuh Konfirmasi*, yang memicu *ClockIn Page* untuk menampilkan *Confirmation Modal* kepada *Karyawan* (langkah 5), dan proses diteruskan ke *Diagram 3B* untuk penanganan lebih lanjut.



Gambar 3.8. Sequence Diagram 2 Validasi Lokasi

H Sequence Diagram 3A Clock In Berhasil (Dalam Lokasi)

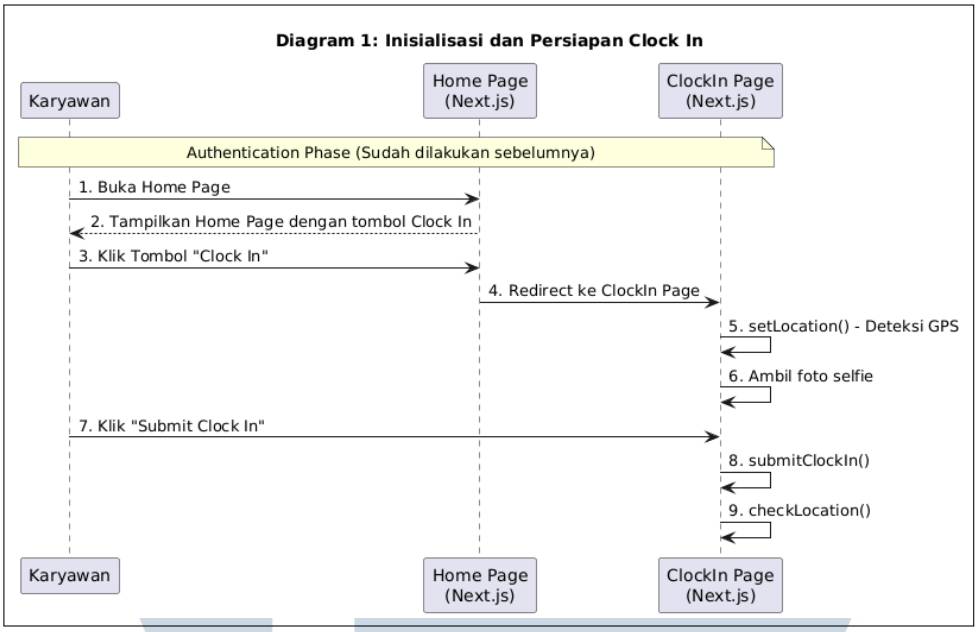
Diagram urutan ini menguraikan tahapan penyelesaian proses *Clock In* dalam kondisi sukses, yaitu ketika hasil validasi dari *Diagram 2* menunjukkan status *Within Range*. Alur dimulai ketika *ClockIn Page* mengirimkan permintaan *POST* ke *endpoint* `/api/attendance/clock-in` pada *AttendanceController* (langkah 1). Permintaan ini memuat seluruh data absensi yang diperlukan, termasuk *user_id*, *latitude*, *longitude*, *foto_selfie*, dan *timestamp*. *AttendanceController* kemudian menjalankan operasi `INSERT INTO absensi` untuk menyimpan data tersebut ke dalam *Database* (langkah 2), yang memberikan konfirmasi *Success* (langkah 3). Setelah perekaman berhasil, *AttendanceController* merespons *ClockIn Page* dengan kode status *201 Created* disertai data absensi yang baru dicatat (langkah 4). Proses diakhiri dengan pengalihan (*redirect*) *client* ke *Home Page* (langkah 5), di mana *Karyawan* menerima notifikasi akhir *Absen Berhasil* (langkah 6).



Gambar 3.9. Sequence Diagram 3A Clock In Berhasil (Dalam Lokasi)

I Sequence Diagram 3b Clock In dengan Konfirmasi (Luar Lokasi)

Diagram urutan ini menggambarkan prosedur penyelesaian *Clock In* yang melibatkan konfirmasi manual, yang diaktifkan ketika hasil validasi dari *Diagram 2* menunjukkan status *Out of Range*. Proses ini diatur oleh *fragment alt* yang menangani dua skenario keputusan *Karyawan*. Skenario pertama (Klik ”Ya”) terjadi ketika *Karyawan* mengklik tombol ”*Ya*” sebagai persetujuan untuk melanjutkan absensi (langkah 1). *ClockIn Page* kemudian mengirimkan permintaan *POST* ke *endpoint* `/api/attendance/clock-in`, menyertakan seluruh data absensi ditambah parameter *force: true* untuk memaksa perekaman meskipun di luar lokasi (langkah 2). *AttendanceController* memproses permintaan ini dengan operasi `INSERT INTO absensi` ke *Database* (langkah 3), menerima konfirmasi *Success* (langkah 4), dan merespons *ClockIn Page* dengan kode status *201 Created* dan data absensi (langkah 5). Proses ini diselesaikan dengan pengalihan (*redirect*) ke *Home Page* (langkah 6) dan menampilkan notifikasi *Absen Berhasil (Luar Lokasi)* (langkah 7). Skenario kedua (Klik Batal) adalah penolakan oleh *Karyawan* dengan mengklik tombol *Batal* (langkah 1), yang hanya menyebabkan penutupan *modal* konfirmasi, sementara *Karyawan* tetap berada di *ClockIn Page* tanpa mencatatkan absensi (langkah 2).



Gambar 3.10. Sequence Diagram 3b Clock In dengan Konfirmasi (Luar Lokasi)

3.3.4 Pseudocode

Pseudocode merupakan representasi algoritma dalam bentuk semi-formal yang menjelaskan alur logika program tanpa terikat sintaks bahasa pemrograman tertentu. Representasi ini digunakan untuk mendokumentasikan tiga fungsi utama sistem autentikasi: *Register* (pendaftaran pengguna), *Login* (autentikasi), dan *RefreshToken* (pembaruan sesi).

A Fungsi Login

Kode 3.1 menjelaskan alur utama untuk autentikasi pengguna. Proses dimulai dengan pemeriksaan kesehatan *database* untuk memastikan layanan *backend* beroperasi sebelum melanjutkan. Sistem kemudian mengurai dan memvalidasi kredensial login pengguna (email dan kata sandi) dari *request* JSON. Setelah validasi berhasil, sebuah kueri *database* yang aman mengambil profil pengguna yang aktif menggunakan alamat email yang diberikan. Jika email tidak ditemukan atau kata sandi tidak cocok dengan hash yang tersimpan, sistem mengembalikan pesan kesalahan umum untuk alasan keamanan.

Jika kredensial benar, sistem menghasilkan dua token: sebuah *access token* untuk otorisasi permintaan *API* berikutnya dan sebuah *refresh token* untuk mendapatkan token akses baru nanti. Kata sandi dihapus dari objek pengguna

sebelum dikembalikan, dan respons sukses yang berisi kedua token beserta data profil pengguna (tanpa kata sandi) dikirimkan. Keseluruhan proses dibatasi waktu untuk mencegah permintaan yang menggantung dan memastikan pengalaman pengguna yang responsif.

```
FUNCTION Login(c *gin.Context):
// 1. Cek koneksi database
IF database.HealthCheck() error:
RESPONSE 503: service unavailable
RETURN

// 2. Parse dan validasi request
DECLARE credentials sebagai LoginRequest
IF c.ShouldBindJSON(credentials) error:
RESPONSE 400: validation error dengan pesan format
RETURN

// 3. Persiapkan context dengan timeout
ctx = context.WithTimeout(10 detik)
defer cancel()

// 4. Query database untuk mencari user berdasarkan email
DECLARE user sebagai models.User
QUERY database:
SELECT id, tenant_id, username, email, password, role,
full_name, phone, avatar_url, is_active, created_at, updated_at
FROM godplan.users
WHERE email = credentials.Email AND is_active = true

IF query error atau user tidak ditemukan:
RESPONSE 401: "Invalid email or password" (pesan generik)
RETURN

// 5. Verifikasi password menggunakan bcrypt
IF bcrypt.CompareHashAndPassword(user.Password, credentials.
    Password) error:
RESPONSE 401: "Invalid email or password"
RETURN

// 6. Generate Access Token
accessToken = jwtUtil.GenerateToken(user.ID, user.Email, user.Role
    , user.TenantID)
IF error:
```

```
RESPONSE 500: "Failed to generate session"
RETURN

// 7. Generate Refresh Token
refreshToken = jwtUtil.GenerateRefreshToken(user.ID, user.TenantID
    )
IF error:
RESPONSE 500: "Failed to generate session"
RETURN

// 8. Hapus password dari response untuk keamanan
user.Password = ""

// 9. Response sukses
RESPONSE 200:
success: true
message: "Login successful"
data: {
token: accessToken,
refresh_token: refreshToken,
user: user (tanpa password)
}
```

Kode 3.1: Pseudocode Fungsi Login

B Fungsi Register

Kode 3.2 merinci prosedur pendaftaran akun pengguna baru. Fungsi ini memvalidasi input pengguna secara menyeluruh, termasuk pemeriksaan panjang kata sandi minimum dan keunikan *username* serta email. Kata sandi langsung di-hash menggunakan algoritma bcrypt sebelum penyimpanan untuk menjamin keamanan data sensitif. Peran (*role*) pengguna default diatur sebagai ”karyawan” jika tidak ditentukan.

Data pengguna baru kemudian dimasukkan ke dalam *database* dengan ID *tenant* default. Jika pendaftaran untuk peran karyawan, sebuah catatan tambahan secara otomatis dibuat di tabel karyawan dengan ID yang dihasilkan. Setelah penyisipan berhasil, data pengguna yang baru dibuat diambil kembali untuk memastikan keakuratannya. Sebagai langkah akhir, token akses dan penyegar dibuat untuk sesi langsung pengguna baru, dan respons konfirmasi dikirimkan beserta token dan profil pengguna, sehingga memungkinkan akses langsung ke

sistem setelah pendaftaran.

```
FUNCTION Register(c *gin.Context):
  // 1. Cek koneksi database
  IF database.HealthCheck() error:
    RESPONSE 503: service unavailable
    RETURN

  // 2. Parse dan validasi request
  DECLARE req sebagai UserRegistrationRequest
  IF c.ShouldBindJSON(req) error:
    RESPONSE 400: validation error dengan pesan format
    RETURN

  // 3. Validasi tambahan
  IF req.Password length < 8:
    RESPONSE 400: password minimal 8 karakter
    RETURN

  // 4. Hash password
  hashedPassword = bcrypt.GenerateFromPassword(req.Password)
  IF error:
    RESPONSE 500: internal error
    RETURN

  // 5. Set default role
  IF req.Role == "":
    req.Role = "employee"

  // 6. Persiapkan context dengan timeout
  ctx = context.WithTimeout(10 detik)
  defer cancel()

  // 7. Siapkan default tenant ID
  defaultTenantID = uuid.Parse("
    00000000-0000-0000-0000-000000000001")

  // 8. Insert user ke database
  EXECUTE database query:
    INSERT INTO users (tenant_id, username, email, password, role,
                       full_name, phone, avatar_url, is_active,
                       created_at, updated_at)
    VALUES (...)
    RETURNING id
```

```
42
43  IF error:
44    IF error mengandung "unique" atau "duplicate":
45      RESPONSE 409: username/email sudah ada
46    ELSE:
47      RESPONSE 500: gagal membuat akun
48    RETURN
49
50  // 9. Jika role employee, buat record employee
51  IF req.Role == "employee":
52    employeeID = "EMP-" + userID[:8]
53    EXECUTE database query:
54      INSERT INTO employees (tenant_id, user_id, employee_id,
55                             join_date, created_at, updated_at)
56      VALUES (...)
57
58  // 10. Ambil data user yang baru dibuat
59  QUERY database: SELECT ... FROM users WHERE id = userID
60  SET createdUser dari hasil query
61
62  // 11. Generate tokens
63  accessToken = jwtUtil.GenerateToken(userID, email, role,
     tenantID)
64  refreshToken = jwtUtil.GenerateRefreshToken(userID, tenantID)
65
66  // 12. Response sukses
67  RESPONSE 201:
68    success: true
69    message: "User registered successfully"
70    data: {
71      token: accessToken,
72      refresh_token: refreshToken,
73      user: createdUser (tanpa password)
74    }
```

Kode 3.2: Pseudocode Fungsi Register

C Fungsi RefreshToken

Kode 3.3 menggambarkan mekanisme untuk memperbarui sesi pengguna tanpa kredensial login. Fungsi ini menerima sebuah *refresh token* yang masih berlaku, yang diperoleh pengguna saat login atau registrasi awal. Token ini

divalidasi terlebih dahulu untuk memastikan keaslian dan masa berlakunya belum habis.

Setelah token divalidasi, sistem mengekstrak identitas pengguna yang terkandung di dalamnya dan melakukan kueri ke *database* untuk memverifikasi bahwa akun tersebut masih aktif. Verifikasi tambahan ini mencegah penerbitan token baru untuk akun yang telah dinonaktifkan. Jika semua pemeriksaan lolos, sebuah *access token* baru yang segar dibuat dan dikirim kembali ke pengguna, memperpanjang sesi akses mereka ke *API* dengan aman.

```
FUNCTION RefreshToken(c *gin.Context):
// 1. Parse request body
DECLARE body dengan struktur: { refresh_token: string }
IF c.ShouldBindJSON(body) error:
RESPONSE 400: "refresh_token is required"
RETURN

// 2. Validasi Refresh Token
claims = jwtUtil.ValidateRefreshToken(body.RefreshToken)
IF error:
RESPONSE 401: "Invalid or expired refresh token"
RETURN

// 3. Verifikasi user masih aktif di database
ctx = context.WithTimeout(5 detik)
defer cancel()

DECLARE user sebagai models.User
QUERY database:
SELECT email, role
FROM godplan.users
WHERE id = claims.UserID AND is_active = true

IF query error atau user tidak ditemukan:
RESPONSE 401: "User no longer active"
RETURN

// 4. Generate Access Token baru
newAccessToken = jwtUtil.GenerateToken(
claims.UserID,
user.Email,
user.Role,
claims.TenantID
```

```
)

IF error:
RESPONSE 500: "Failed to generate token"
RETURN

// 5. Response sukses dengan token baru
RESPONSE 200:
success: true
data: {
token: newAccessToken
}
```

Kode 3.3: Pseudocode Fungsi RefreshToken

D Fungsi CheckLocation

Kode 3.4 merupakan fungsi *CheckLocation* bertugas untuk memvalidasi apakah lokasi pengguna berada dalam jangkauan kantor yang ditentukan sebelum melakukan absensi. Fungsi ini menerima koordinat latitude dan longitude dari pengguna melalui *request* JSON, kemudian memvalidasi format data input. Setelah data divalidasi, sistem akan menghitung jarak antara lokasi pengguna dengan titik pusat kantor yang telah ditentukan dalam konfigurasi menggunakan algoritma haversine.

Hasil validasi dikembalikan dalam bentuk respons yang mencakup status apakah lokasi berada dalam jangkauan (*in_range*), pesan deskriptif, indikasi apakah perlu menggunakan fitur *force*, dan jarak aktual dalam meter. Fungsi ini membantu memberikan umpan balik langsung kepada pengguna mengenai kelayakan lokasi mereka untuk melakukan absensi sebelum melanjutkan proses clock-in atau clock-out.

```
FUNCTION CheckLocation(c *gin.Context):
  // 1. Log awal proses di mode development
  IF config.IsDevelopment():
    PRINT "CheckLocation started"

  // 2. Setup panic recovery untuk error handling
  DEFER fungsi recovery:
    IF terjadi panic:
      IF config.IsDevelopment():
        PRINT panic details
```

```
    RESPONSE 500: "Location check failed"

// 3. Parse dan validasi request body
DECLARE req sebagai LocationCheckRequest
IF c.ShouldBindJSON(req) error:
  IF config.IsDevelopment():
    PRINT bind error details
  RESPONSE 400: "Invalid request body"
  RETURN

// 4. Validasi lokasi menggunakan utility function
validation = utils.ValidateLocation(req.Latitude, req.Longitude)

// 5. Siapkan response berdasarkan hasil validasi
response = MAP:
  "in_range": validation.InRange,
  "message": validation.Message,
  "need_force": validation.NeedForce,
  "distance": validation.Distance

// 6. Kirim response sukses
RESPONSE 200:
  success: true
  message: "Location validation successful"
  data: response
```

Kode 3.4: Pseudocode Fungsi CheckLocation

E Fungsi ClockIn

Kode 3.5 merupakan fungsi *ClockIn* menangani proses absensi masuk (clock-in) pengguna dengan melakukan validasi lokasi dan mencatat waktu kedatangan. Fungsi ini dimulai dengan memverifikasi identitas pengguna dari token autentikasi dan mengekstrak *userID* serta *tenantID*. Sistem kemudian memvalidasi apakah lokasi pengguna berada dalam radius yang diizinkan menggunakan koordinat yang dikirimkan bersama *request*.

Jika lokasi berada di luar jangkauan dan pengguna tidak menggunakan opsi *force*, sistem akan menolak proses clock-in dengan memberikan informasi jarak aktual dan radius yang diizinkan. Sebelum mencatat absensi, sistem memeriksa apakah pengguna sudah melakukan clock-in pada hari yang sama untuk mencegah duplikasi. Jika semua validasi berhasil, catatan absensi baru dibuat di database

dengan status ”approved” untuk lokasi dalam jangkauan, atau ”pending_forced” jika menggunakan opsi *force* di luar jangkauan.

```
FUNCTION ClockIn(c *gin.Context):
  // 1. Log awal proses
  IF config.IsDevelopment():
    PRINT "ClockIn started"

  // 2. Setup panic recovery
  DEFER fungsi recovery:
    IF terjadi panic:
      RESPONSE 500: "Clock in failed"

  // 3. Parse request body
  DECLARE req sebagai ClockInRequest
  IF c.ShouldBindJSON(req) error:
    RESPONSE 400: "Invalid request body"
    RETURN

  // 4. Ambil userID dan tenantID dari context
  userIDVal = c.Get("userID")
  IF userIDVal tidak exists:
    RESPONSE 401: "User not authenticated"
    RETURN

  tenantIDStr = c.GetString("tenant_id")
  tenantID = uuid.Parse(tenantIDStr)

  // 5. Konversi userID ke UUID
  SWITCH tipe userIDVal:
    CASE uuid.UUID: userID = userIDVal
    CASE string: userID = uuid.Parse(userIDVal)
    DEFAULT: RESPONSE 500: "Invalid user ID type"

  // 6. Validasi lokasi
  inRange, distance = utils.IsWithinOfficeRange(req.Latitude, req.
   Longitude)
  cfg = config.Load()

  // 7. Cek jika lokasi di luar jangkauan
  IF NOT inRange AND NOT req.Force:
    RESPONSE 400: "Lokasi di luar jangkauan kantor"
    RETURN

```

```
// 8. Tentukan status berdasarkan lokasi
status = "approved"
IF NOT inRange AND req.Force:
  status = "pending_forced"

// 9. Cek apakah sudah clock-in hari ini
QUERY database: SELECT id FROM attendances
  WHERE user_id = userID AND tenant_id = tenantID
  AND attendance_date = CURRENT_DATE

IF record ditemukan:
  RESPONSE 400: "Sudah melakukan Clock In hari ini"
  RETURN

// 10. Insert record clock-in baru
now = time.Now()
EXECUTE database query:
  INSERT INTO attendances (tenant_id, user_id, type, status,
    check_in_time, check_in_lat, check_in_lng, check_in_photo,
    attendance_date, in_range, force_attendance, created_at)
  VALUES (...)
  RETURNING id

// 11. Siapkan response
response = AttendanceResponse:
  ID: attendanceID,
  UserID: userID,
  Type: "CheckIn",
  Status: status,
  Latitude: req.Latitude,
  Longitude: req.Longitude,
  PhotoSelfie: req.PhotoSelfie,
  InRange: inRange,
  ForceAttendance: req.Force,
  CreatedAt: now,
  Distance: distance,
  MaxRadius: cfg.AttendanceRadiusMeters

// 12. Kirim response sukses
RESPONSE 201:
  success: true
  message: "Clock in successful"
```

```
    data: response
```

Kode 3.5: Pseudocode Fungsi ClockIn

F Fungsi ClockOut

Kode 3.5 merupakan fungsi *ClockOut* mengelola proses absensi keluar (clock-out) dengan melengkapi catatan absensi yang telah dimulai pada clock-in. Fungsi ini memastikan bahwa pengguna telah melakukan clock-in sebelumnya pada hari yang sama sebelum mengizinkan clock-out. Validasi lokasi juga dilakukan untuk memastikan konsistensi dengan kebijakan absensi berbasis lokasi.

Sistem menghitung total jam kerja berdasarkan selisih waktu antara clock-in dan clock-out, kemudian memperbarui catatan absensi yang ada dengan informasi waktu keluar, lokasi, dan foto selfie. Jika clock-out dilakukan di luar jangkauan kantor tanpa opsi *force*, sistem akan menolak permintaan tersebut. Fungsi ini memberikan fleksibilitas dengan opsi *force* untuk situasi darurat, namun menandai absensi tersebut dengan status ”pending_forced” untuk memerlukan verifikasi manual.

```
FUNCTION ClockOut(c *gin.Context):
  // 1. Log awal proses
  IF config.IsDevelopment():
    PRINT "ClockOut started"

  // 2. Setup panic recovery
  DEFER fungsi recovery:
    IF terjadi panic:
      RESPONSE 500: "Clock out failed"

  // 3. Parse request body
  DECLARE req sebagai ClockOutRequest
  IF c.ShouldBindJSON(req) error:
    RESPONSE 400: "Invalid request body"
    RETURN

  // 4. Ambil userID dan tenantID dari context
  userIDVal = c.Get("userID")
  IF userIDVal tidak exists:
    RESPONSE 401: "User not authenticated"
    RETURN

```

```
tenantIDStr = c.GetString("tenant_id")
tenantID = uuid.Parse(tenantIDStr)

// 5. Konversi userID ke UUID
SWITCH tipe userIDVal:
  CASE uuid.UUID: userID = userIDVal
  CASE string: userID = uuid.Parse(userIDVal)
  DEFAULT: RESPONSE 500: "Invalid user ID type"

// 6. Validasi lokasi
inRange, distance = utils.IsWithinOfficeRange(req.Latitude, req.
 Longitude)
cfg = config.Load()

// 7. Cek jika lokasi di luar jangkauan
IF NOT inRange AND NOT req.Force:
  RESPONSE 400: "Lokasi di luar jangkauan kantor"
  RETURN

// 8. Cari record clock-in untuk hari ini
QUERY database: SELECT id, check_in_time FROM attendances
  WHERE user_id = userID AND tenant_id = tenantID
  AND attendance_date = CURRENT_DATE AND check_out_time IS NULL

IF record tidak ditemukan:
  RESPONSE 400: "Belum melakukan Clock In hari ini atau sudah
 Clock Out"
  RETURN

// 9. Hitung total jam kerja
now = time.Now()
totalHours = now.Sub(checkInTime).Hours()

// 10. Tentukan status berdasarkan lokasi
status = "approved"
IF NOT inRange AND req.Force:
  status = "pending_forced"

// 11. Update record dengan data clock-out
EXECUTE database query:
  UPDATE attendances SET
    check_out_time = now,
    check_out_lat = req.Latitude,
```

```
      check_out_lng = req.Longitude,
      check_out_photo = req.PhotoSelfie,
      total_hours = totalHours,
      type = 'CheckOut',
      status = status,
      updated_at = now
    WHERE id = attendanceID AND tenant_id = tenantID

  // 12. Siapkan response
  response = AttendanceResponse:
    ID: attendanceID,
    UserID: userID,
    Type: "CheckOut",
    Status: status,
    Latitude: req.Latitude,
    Longitude: req.Longitude,
    PhotoSelfie: req.PhotoSelfie,
    InRange: inRange,
    ForceAttendance: req.Force,
    CreatedAt: now,
    Distance: distance,
    MaxRadius: cfg.AttendanceRadiusMeters

  // 13. Kirim response sukses
  RESPONSE 200:
    success: true
    message: "Clock out successful"
    data: response
```

Kode 3.6: Pseudocode Fungsi ClockOut

G Fungsi GetAttendance

Fungsi *GetAttendance* menyediakan akses kepada pengguna untuk melihat riwayat absensi mereka dengan berbagai opsi penyaringan. Fungsi ini mendukung filter berdasarkan tanggal spesifik atau pembatasan jumlah record yang ditampilkan. Data yang diambil dari database termasuk informasi lengkap setiap sesi absensi seperti jenis (clock-in/clock-out), status, tanggal, waktu, lokasi, dan foto selfie.

Sistem mengonversi format waktu dari database menjadi format yang lebih mudah dibaca untuk keperluan tampilan di aplikasi mobile. Fungsi ini juga menambahkan informasi lokasi nama default (”Kantor Pusat Godplan”) untuk

memberikan konteks lebih baik pada data yang ditampilkan. Dengan struktur respons yang konsisten, fungsi ini memudahkan integrasi dengan berbagai platform klien yang membutuhkan data riwayat absensi.

```
FUNCTION GetAttendance(c *gin.Context):
  // 1. Log awal proses
  IF config.IsDevelopment():
    PRINT "GetAttendance started"

  // 2. Setup panic recovery
  DEFER fungsi recovery:
    IF terjadi panic:
      RESPONSE 500: "Failed to get attendance"

  // 3. Ambil userID dan tenantID dari context
  userIDVal = c.Get("userID")
  IF userIDVal tidak exists:
    RESPONSE 401: "User not authenticated"
    RETURN

  tenantIDStr = c.GetString("tenant_id")
  tenantID = uuid.Parse(tenantIDStr)

  // 4. Konversi userID ke UUID
  SWITCH tipe userIDVal:
    CASE uuid.UUID: userID = userIDVal
    CASE string: userID = uuid.Parse(userIDVal)
    DEFAULT: RESPONSE 500: "Invalid user ID type"

  // 5. Parse query parameters
  dateFilter = c.Query("date")
  limitStr = c.Query("limit")

  // 6. Tentukan limit default atau dari parameter
  IF limitStr == "":
    limit = 30
  ELSE:
    limit = strconv.Atoi(limitStr)
    IF limit <= 0: limit = 30

  // 7. Query database dengan filter yang sesuai
  IF dateFilter != "":
    QUERY database: SELECT ... FROM attendances
```

```
    WHERE user_id = userID AND tenant_id = tenantID
    AND attendance_date = dateFilter
    ORDER BY created_at DESC LIMIT limit
ELSE:
  QUERY database: SELECT ... FROM attendances
    WHERE user_id = userID AND tenant_id = tenantID
    ORDER BY created_at DESC LIMIT limit

// 8. Iterasi hasil query dan bangun response
DECLARE attendances sebagai array AttendanceResponse
FOR setiap row dalam hasil query:
  attendance = AttendanceResponse:
    ID: att.ID,
    UserID: att.UserID,
    Type: att.Type,
    Status: att.Status,
    Date: attendanceDate,
    Time: attendanceTime,
    LocationName: "Kantor Pusat Godplan",
    Latitude: att.Latitude,
    Longitude: att.Longitude,
    PhotoSelfie: att.PhotoSelfie,
    InRange: att.InRange,
    ForceAttendance: att.ForceAttendance,
    CreatedAt: att.CreatedAt
  APPEND attendance ke attendances

// 9. Kirim response sukses
RESPONSE 200:
  success: true
  message: "Attendance records retrieved"
  data: attendances
```

Kode 3.7: Pseudocode Fungsi GetAttendance

3.3.5 Hasil dan Implementasi Godplan Mobile

Pada subbab ini dijelaskan hasil pengembangan serta proses implementasi *Godplan Mobile* sebagai aplikasi *Progressive Web App (PWA)* yang terintegrasi dengan sistem desktop. Penjelasan mencakup fitur-fitur utama yang telah berhasil diterapkan, alur kerja aplikasi, serta bagaimana aplikasi tersebut berinteraksi dengan platform pendukung lainnya.

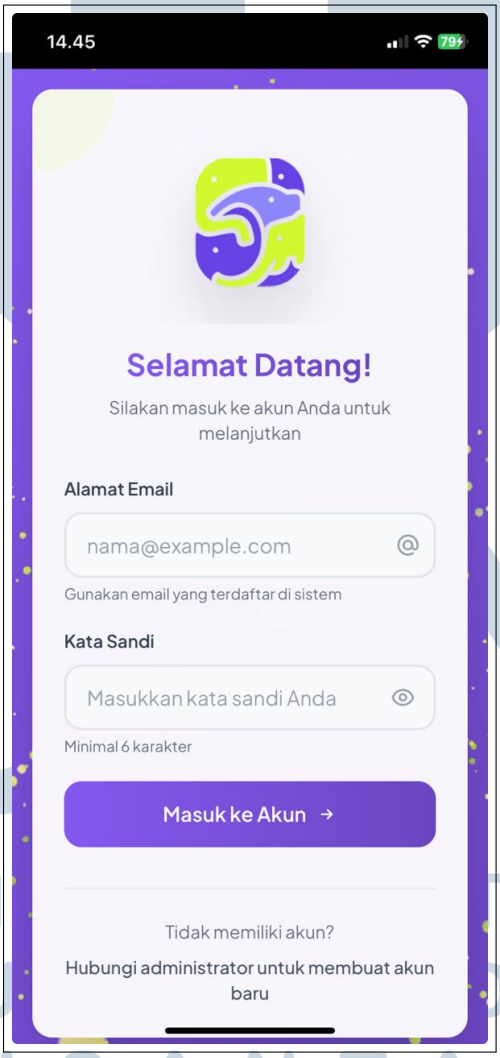
A Halaman Loading Awal Godplan Mobile

Gambar 3.10 menampilkan Halaman *Loading* Awal (*Splash Screen*) pada aplikasi *Godplan Mobile*. Antarmuka ini merupakan tampilan pertama yang diakses pengguna saat membuka aplikasi, yang berfungsi untuk menjembatani proses inisialisasi sistem sebelum masuk ke menu utama. Desain halaman dibuat minimalis dengan latar belakang ungu yang menjadi identitas visual aplikasi, serta menampilkan logo dan nama '*Godplan App*' secara sentral untuk memperkuat *branding*.

Gambar 3.11. Halaman Loading Awal

B Halaman Login Godplan Mobile

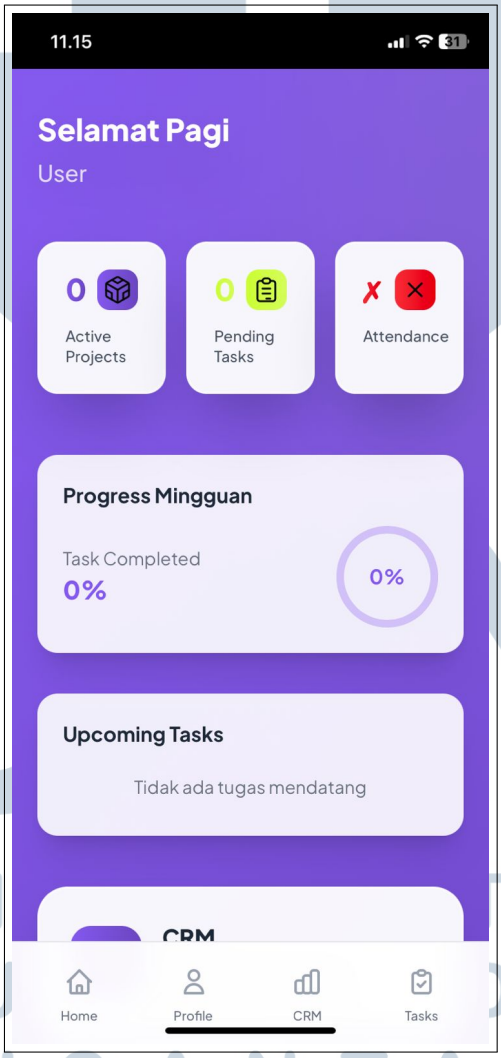
Gambar 3.11 menampilkan Halaman *Login* pada aplikasi *Godplan Mobile* yang berfungsi sebagai gerbang otentikasi pengguna. Pada antarmuka ini, pengguna diminta untuk memasukkan kredensial akses berupa alamat *email* dan kata sandi (*password*) pada kolom yang tersedia untuk memverifikasi identitas mereka. Halaman ini juga menyertakan informasi navigasi bagi pengguna yang belum memiliki akun untuk menghubungi administrator, serta tombol aksi utama untuk memproses data *login* ke dalam sistem.



Gambar 3.12. Halaman Login

C Halaman Home Godplan Mobile

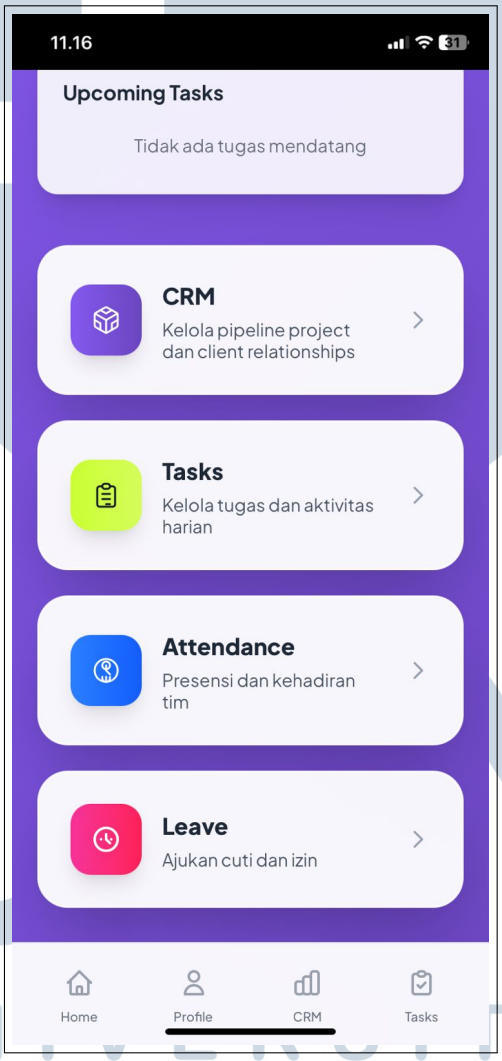
Gambar 3.12 menampilkan Halaman *Home* pada aplikasi *Godplan Mobile* yang berfungsi sebagai *dashboard* utama bagi pengguna. Antarmuka ini dirancang untuk menyajikan ringkasan produktivitas secara *real-time*, yang mencakup informasi mengenai jumlah *Active Projects*, *Pending Tasks*, serta status *Attendance*. Selain itu, halaman ini dilengkapi dengan fitur visualisasi *Progress* Mingguan untuk memantau persentase penyelesaian tugas, daftar *Upcoming Tasks*, serta *navigation bar* di bagian bawah untuk akses cepat ke menu *Profile*, *CRM*, dan *Tasks*.



Gambar 3.13. Halaman Home

Gambar 3.13 menampilkan kelanjutan dari Halaman *Home* pada aplikasi *Godplan Mobile*, yang memuat daftar menu navigasi ke fitur-fitur inti. Bagian

ini menyediakan akses cepat (*shortcut*) bagi pengguna untuk masuk ke modul operasional utama, yaitu *CRM* untuk pengelolaan *pipeline project* dan hubungan klien, *Tasks* untuk manajemen tugas harian, *Attendance* untuk melakukan presensi kehadiran tim, serta fitur *Leave* untuk pengajuan permohonan cuti. Tampilan disusun dalam format daftar (*list view*) vertikal dengan ikon representatif untuk meningkatkan *user experience* dan kemudahan akses.

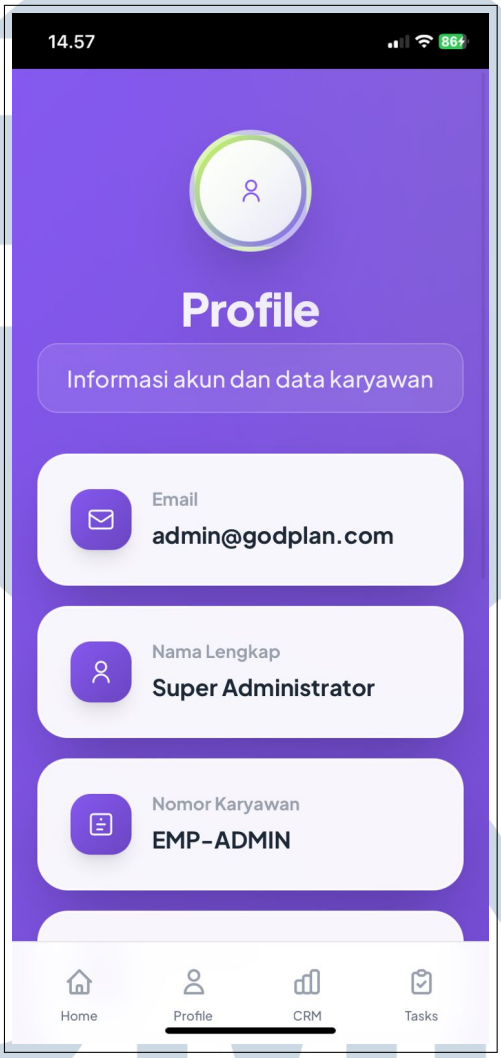


Gambar 3.14. Halaman Home

D Halaman Profile Godplan Mobile

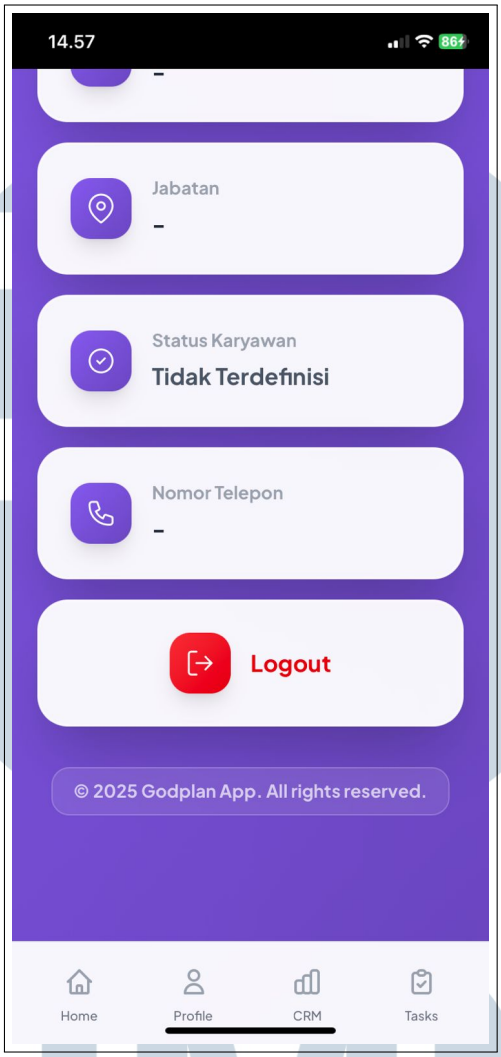
Gambar 3.14 menampilkan Halaman *Profile* pada aplikasi *Godplan Mobile*, yang didedikasikan untuk menyajikan informasi akun dan data karyawan. Pada antarmuka ini, pengguna dapat meninjau rincian identitas mereka yang terdaftar

dalam sistem, meliputi alamat *email*, nama lengkap pengguna, serta nomor identitas karyawan. Halaman ini berfungsi sebagai pusat informasi personal untuk memastikan validitas data pengguna yang sedang *login* dalam ekosistem kerja.



Gambar 3.15. Halaman Profile

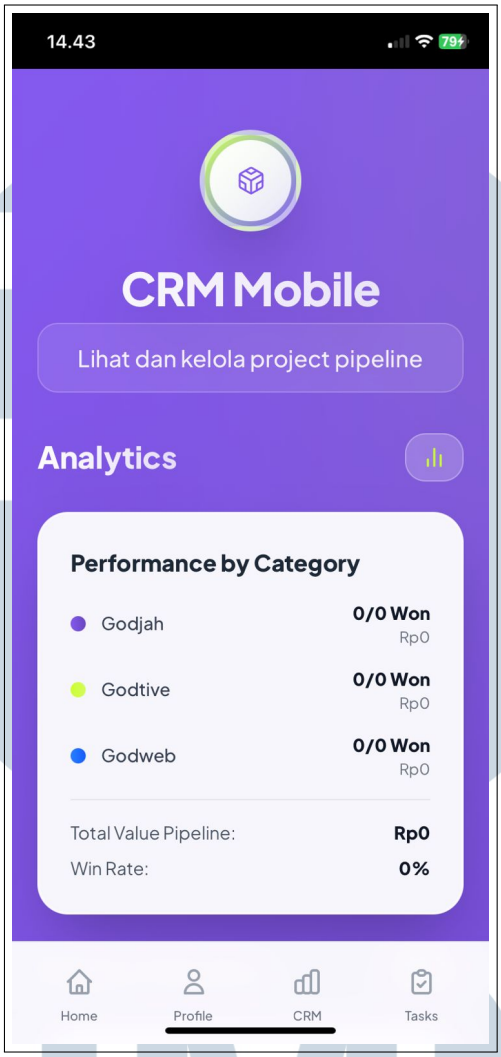
Gambar 3.15 merepresentasikan bagian lanjutan dari Halaman *Profile* pada aplikasi *Godplan Mobile*. Antarmuka ini menampilkan rincian data administratif tambahan pengguna, seperti informasi Jabatan, Status Karyawan, dan Nomor Telepon. Selain berfungsi sebagai media informasi, halaman ini menyediakan fitur keamanan berupa tombol *Logout* yang digunakan untuk mengakhiri sesi aktif pengguna di dalam sistem. Pada bagian paling bawah (*footer*), turut dicantumkan atribusi hak cipta (*copyright*) tahun 2025 sebagai identitas kepemilikan aplikasi.



Gambar 3.16. Halaman Profile

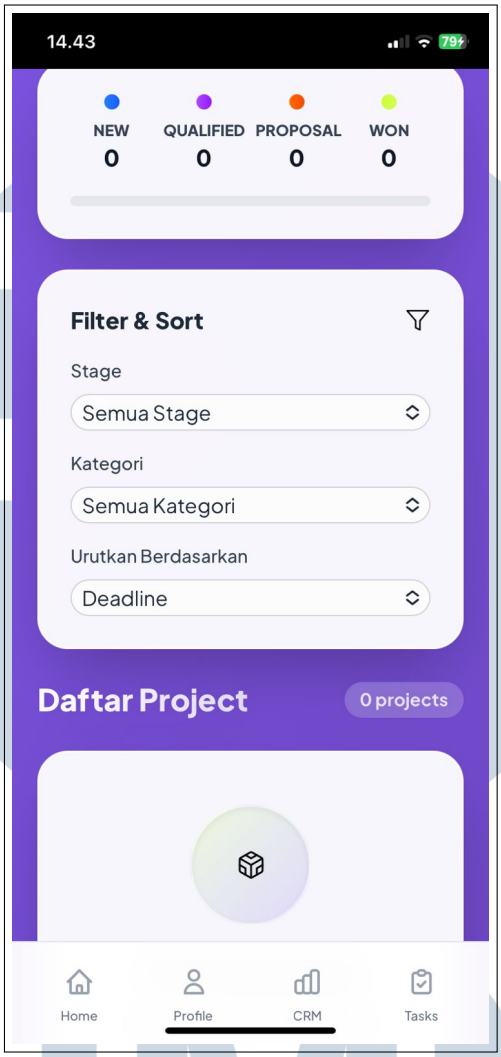
E Halaman CRM Godplan Mobile

Gambar 3.16 menampilkan Halaman *CRM* (*Customer Relationship Management*) pada aplikasi *Godplan Mobile*. Antarmuka ini dirancang untuk memfasilitasi pemantauan kinerja bisnis secara terpusat melalui fitur *Analytics*. Pada halaman ini, pengguna dapat melihat ringkasan visual *Performance by Category* untuk memantau status proyek di berbagai lini seperti Godjah, Godtive, dan Godweb. Informasi yang disajikan mencakup jumlah kesepakatan yang berhasil (*Won*), estimasi nilai pendapatan, serta indikator kunci lainnya seperti *Total Value Pipeline* dan persentase *Win Rate* guna mengevaluasi efektivitas strategi penjualan.



Gambar 3.17. Halaman CRM

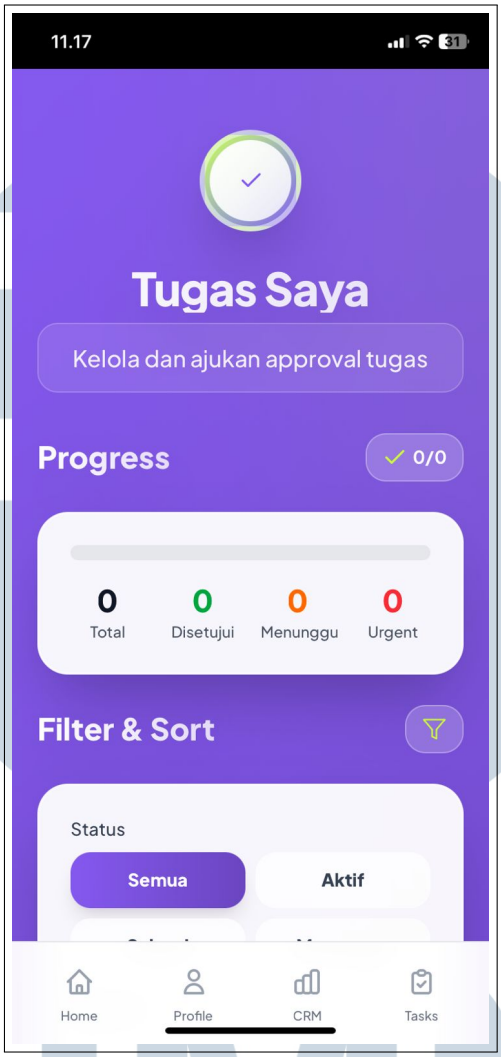
Gambar 3.17 merepresentasikan antarmuka manajemen daftar proyek dalam modul *CRM* pada aplikasi *Godplan Mobile*. Halaman ini dilengkapi dengan fitur *Filter & Sort* yang memungkinkan pengguna untuk melakukan penyaringan data proyek berdasarkan kriteria *Stage* dan kategori, serta mengurutkan daftar berdasarkan tenggat waktu (*Deadline*). Bagian atas halaman menyajikan ringkasan status *pipeline* secara horizontal, mencakup indikator *New*, *Qualified*, *Proposal*, dan *Won*. Di bawah panel filter, terdapat area *Daftar Project* yang berfungsi menampilkan kartu informasi proyek sesuai dengan parameter pencarian yang aktif.



Gambar 3.18. Halaman CRM

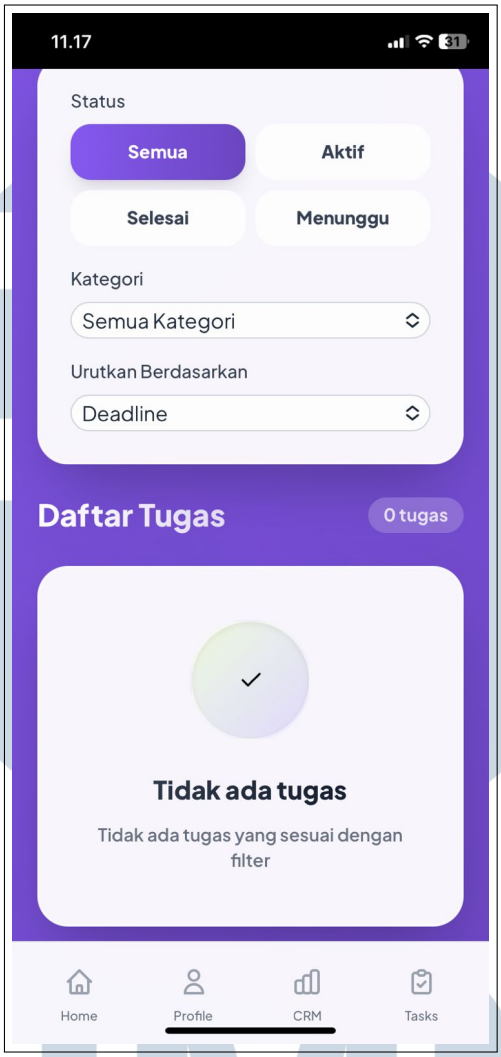
F Halaman Task Godplan Mobile

Gambar 3.18 menampilkan Halaman *Task* pada aplikasi *Godplan Mobile*, yang berfungsi sebagai pusat pengelolaan tugas harian pengguna. Antarmuka ini menyediakan panel *Progress* untuk memantau status penyelesaian pekerjaan secara kuantitatif, yang terbagi ke dalam indikator *Total*, Disetujui, Menunggu, dan *Urgent*. Selain itu, terdapat fitur *Filter & Sort* yang memudahkan pengguna untuk mengelompokkan dan mencari tugas berdasarkan kategori status tertentu, serta mendukung proses pengajuan persetujuan (*approval*) tugas kepada atasan.



Gambar 3.19. Halaman Task

Gambar 3.19 menampilkan bagian daftar tugas pada halaman *Task* di aplikasi *Godplan Mobile*. Antarmuka ini dilengkapi dengan fitur penyaringan (*filter*) interaktif yang memungkinkan pengguna mengelompokkan tugas berdasarkan *Status* (Semua, Aktif, Selesai, Menunggu), *Kategori*, serta opsi pengurutan (*sorting*) berdasarkan tenggat waktu atau *Deadline*. Jika tidak ada data yang sesuai dengan kriteria filter, sistem akan menampilkan indikator *empty state* dengan pesan "Tidak ada tugas" untuk memberikan umpan balik visual yang jelas kepada pengguna.



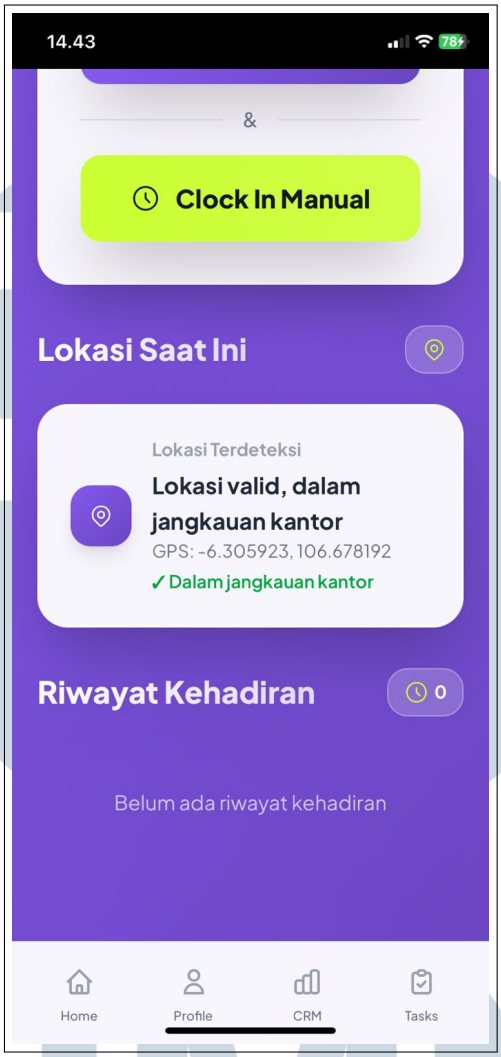
Gambar 3.20. Halaman Task

G Halaman Attendance Godplan Mobile

Gambar 3.20 menampilkan Halaman *Attendance* pada aplikasi *Godplan Mobile*, yang berfungsi sebagai antarmuka pencatatan presensi harian. Halaman ini menyajikan informasi waktu aktual (*real-time*) berupa jam dan tanggal untuk menjadi acuan ketepatan waktu masuk. Fitur utama yang disediakan adalah tombol *Clock In dengan Foto*, yang mengharuskan pengguna melakukan verifikasi visual dan lokasi saat mencatatkan kehadiran mereka, sehingga meningkatkan validitas data absensi karyawan.

Gambar 3.21. Halaman Attendance

Gambar 3.21 merepresentasikan fitur lanjutan pada Halaman *Attendance* di aplikasi *Godplan Mobile*. Bagian ini menyediakan metode presensi alternatif melalui tombol *Clock In Manual* dan menampilkan panel *Lokasi Saat Ini* yang memvalidasi koordinat *GPS* pengguna untuk memastikan status "Dalam jangkauan kantor". Selain itu, terdapat modul *Riwayat Kehadiran* di bagian bawah yang berfungsi sebagai log aktivitas untuk merekam dan menampilkan sejarah presensi pengguna.

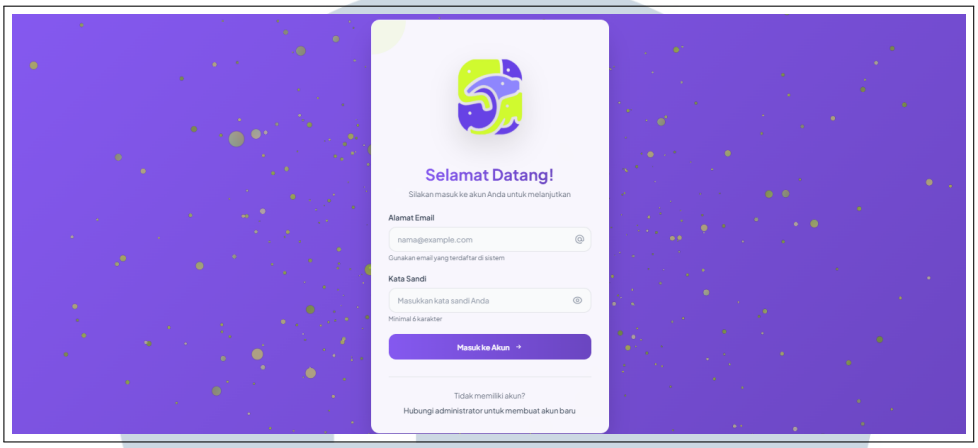


Gambar 3.22. Halaman Attendance

3.3.6 Hasil dan Implementasi Godplan Desktop

Pada subbab ini dipaparkan hasil pengembangan serta implementasi *Godplan Desktop* sebagai komponen utama yang berfungsi melengkapi dan mengelola ekosistem *Godplan Mobile*. Penjelasan difokuskan pada penerapan fitur inti, perancangan alur kerja sistem, serta integrasi data dua arah yang memastikan sinkronisasi informasi antara aplikasi desktop dan platform mobile.

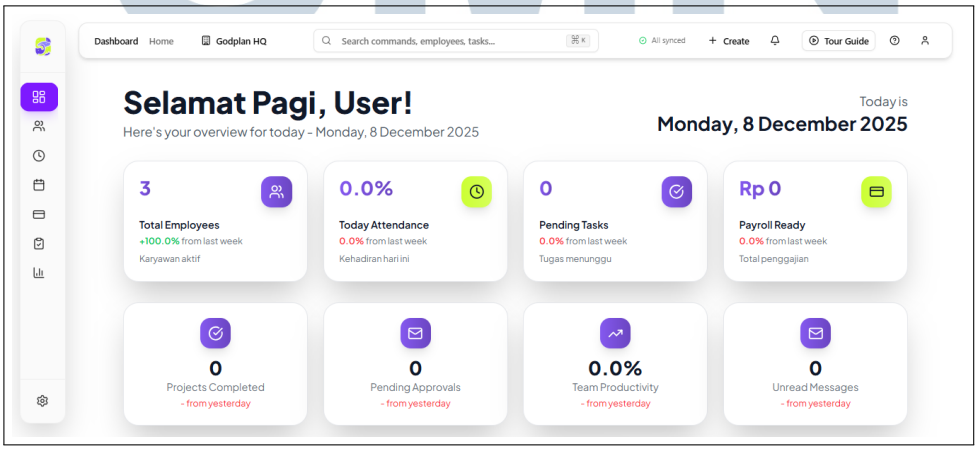
A Halaman Login Godplan Mobile



Gambar 3.23. Halaman Login Desktop Godplan

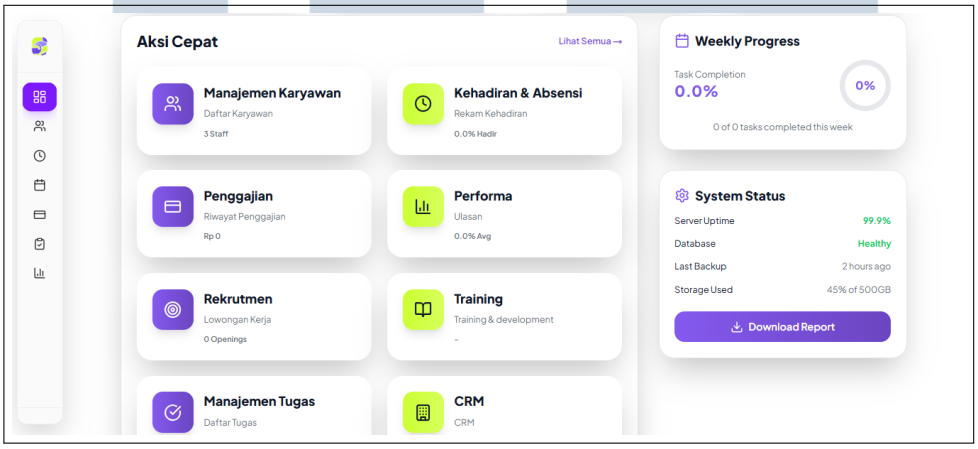
B Halaman Home Godplan Mobile

Gambar 3.23 menampilkan antarmuka Halaman *Home* versi *Desktop* pada sistem *Godplan*, yang berfungsi sebagai *dashboard* administratif terpusat. Halaman ini menyajikan ringkasan operasional harian secara komprehensif, dimulai dengan sapaan pengguna dan informasi tanggal terkini di bagian atas. Bagian inti *dashboard* memuat kartu statistik (*stat cards*) yang menampilkan metrik kunci seperti *Total Employees*, *Today Attendance*, *Pending Tasks*, dan status *Payroll Ready*. Selain itu, baris bawah menyediakan indikator kinerja tambahan meliputi *Projects Completed*, *Pending Approvals*, *Score Productivity*, serta notifikasi *Unread Messages* untuk memudahkan pemantauan aktivitas perusahaan secara *real-time*.



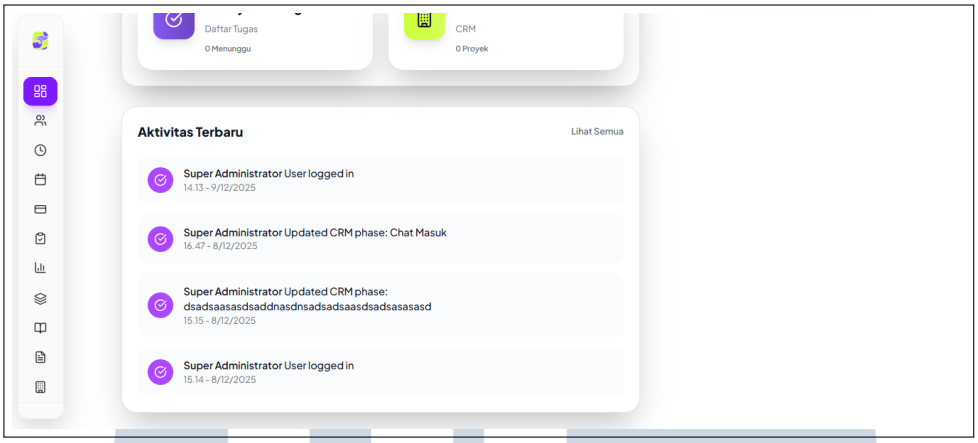
Gambar 3.24. Halaman Home Desktop Godplan

Gambar 3.24 menampilkan panel "Aksi Cepat" pada halaman *Home Desktop* aplikasi *Godplan*, yang dirancang untuk mempercepat navigasi ke modul-modul inti. Antarmuka ini menyajikan menu operasional dalam format *grid*, mencakup Manajemen Karyawan, Penggajian, *CRM*, hingga *Training*. Di bagian sisi kanan (*sidebar*), terdapat widget *Weekly Progress* untuk memantau persentase penyelesaian tugas mingguan, serta panel *System Status* yang menampilkan indikator kesehatan teknis sistem seperti *Server Uptime*, status *Database*, dan kapasitas penyimpanan (*Storage Used*). Tombol *Download Report* juga disediakan untuk memfasilitasi pengunduhan laporan manajerial secara instan.



Gambar 3.25. Halaman Home Desktop Godplan

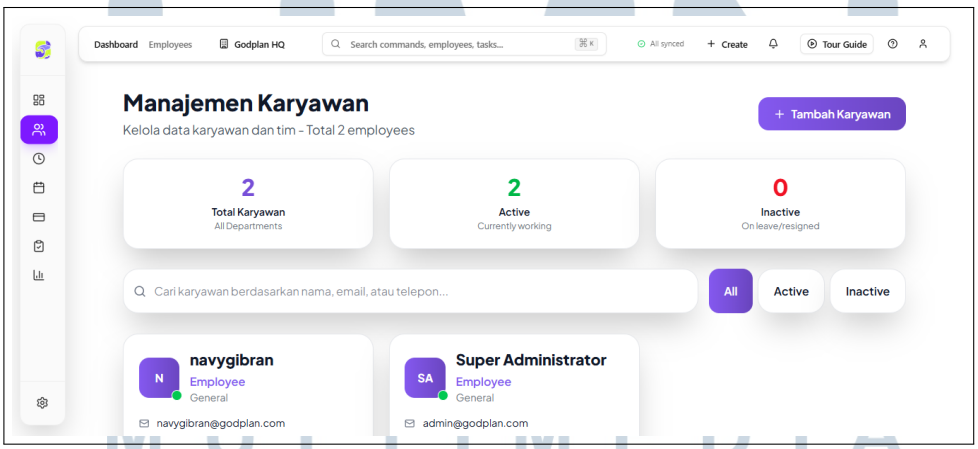
Gambar 3.25 menampilkan panel *Aktivitas Terbaru* pada halaman *Home Desktop* aplikasi *Godplan*, yang terletak di bagian bawah antarmuka utama. Fitur ini berfungsi sebagai *log* aktivitas sistem yang mencatat tindakan pengguna secara kronologis, seperti riwayat *login* dan pembaruan status pada modul *CRM*. Setiap entri aktivitas dilengkapi dengan informasi detail mengenai aktor pelaksana (misalnya *Super Administrator*) serta penanda waktu (*timestamp*) yang spesifik untuk mendukung transparansi dan pengawasan operasional.



Gambar 3.26. Halaman Home Desktop Godplan

C Halaman Employees Godplan Mobile

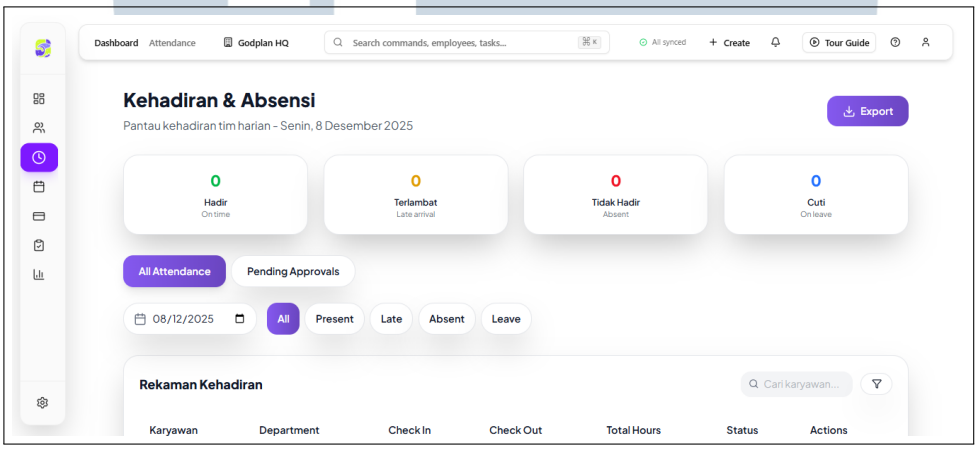
Gambar 3.26 menampilkan Halaman *Employee* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi sebagai pusat administrasi sumber daya manusia (*HR*). Antarmuka ini menyajikan ringkasan status kepegawaian melalui kartu statistik (*stats card*) yang mencakup jumlah *Total Karyawan*, personel *Active*, dan yang berstatus *Inactive*. Halaman ini juga dilengkapi dengan fitur pencarian (*search bar*) dan tombol filter (All, Active, Inactive) untuk mempermudah navigasi data, serta tombol aksi *Tambah Karyawan* untuk mendaftarkan personel baru ke dalam sistem.



Gambar 3.27. Halaman Employee Desktop Godplan

D Halaman Attendance Godplan Mobile

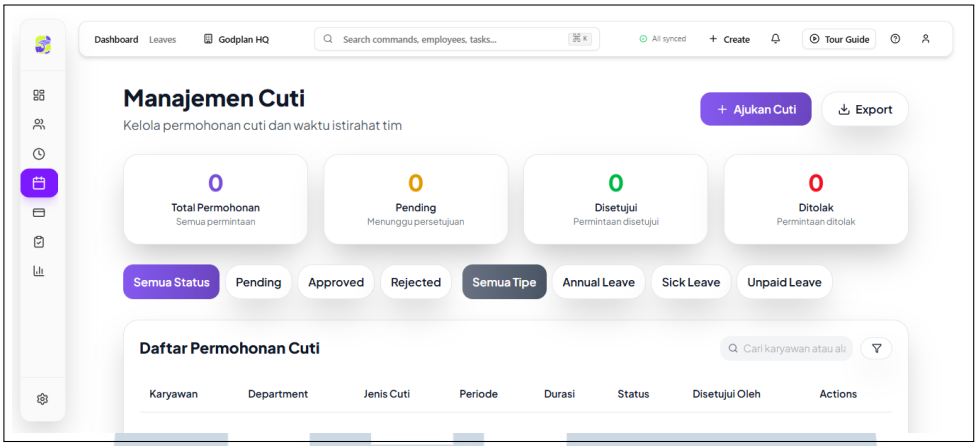
Gambar 3.27 menampilkan Halaman *Attendance* pada versi *Desktop* aplikasi *Godplan*, yang dirancang untuk memantau kedisiplinan tim secara harian. Antarmuka ini menyajikan ringkasan statistik kehadiran melalui kartu metrik yang mengelompokkan status karyawan menjadi *Hadir* (*On time*), *Terlambat* (*Late arrival*), *Tidak Hadir* (*Absent*), dan *Cuti* (*On leave*). Di bagian bawah, terdapat tabel *Rekaman Kehadiran* yang memuat rincian aktivitas presensi setiap karyawan, mencakup waktu *Check In*, *Check Out*, serta *Total Hours* kerja, lengkap dengan fitur filter tanggal dan status untuk memudahkan rekapitulasi data.



Gambar 3.28. Halaman Attendance Desktop Godplan

E Halaman Leaves Godplan Mobile

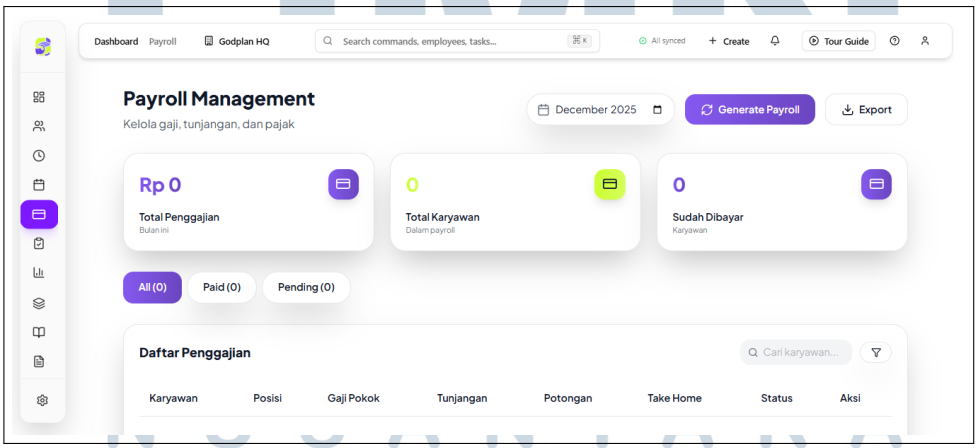
Gambar 3.28 menampilkan Halaman *Leaves* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi sebagai pusat pengelolaan cuti dan waktu istirahat tim. Antarmuka ini menyajikan ringkasan status permohonan melalui kartu statistik yang mencakup kategori Total Permohonan, *Pending*, Disetujui, dan Ditolak. Selain tombol aksi *Ajukan Cuti* dan fitur *Export*, halaman ini memuat tabel *Daftar Permohonan Cuti* yang dilengkapi filter kategori seperti *Annual Leave*, *Sick Leave*, dan *Unpaid Leave* untuk mempermudah klasifikasi data.



Gambar 3.29. Halaman Leaves Desktop Godplan

F Halaman Payroll Godplan Mobile

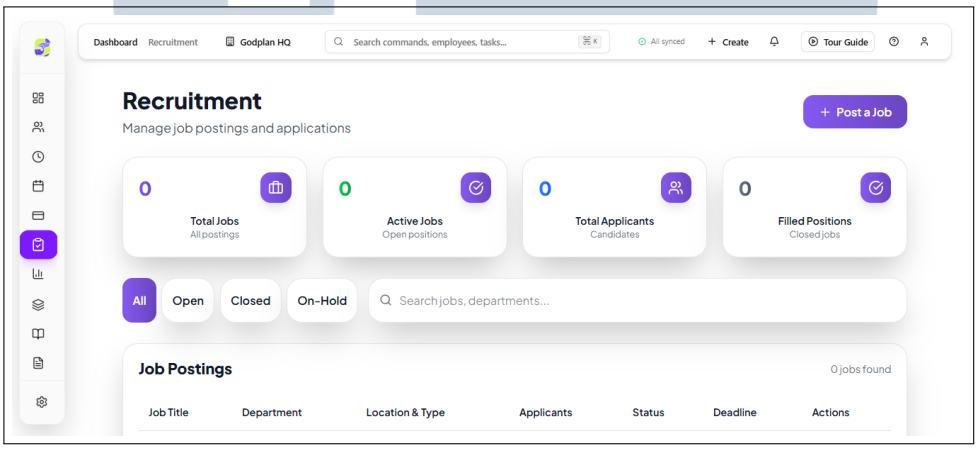
Gambar 3.29 menampilkan Halaman *Payroll* pada versi *Desktop* aplikasi *Godplan*, yang didedikasikan untuk manajemen penggajian, tunjangan, dan pajak secara terpusat. Antarmuka ini menyediakan fitur *Generate Payroll* untuk mengotomatisasi perhitungan gaji bulanan, serta menyajikan ringkasan finansial melalui kartu statistik yang mencakup *Total Penggajian*, *Total Karyawan* dalam periode penggajian, dan status pembayaran (*Sudah Dibayar*). Di bagian bawah, terdapat tabel *Daftar Penggajian* yang merinci komponen pendapatan setiap karyawan, mulai dari *Gaji Pokok*, *Tunjangan*, *Potongan*, hingga nominal *Take Home* dan *Status* pembayaran untuk memastikan transparansi keuangan.



Gambar 3.30. Halaman Payroll Desktop Godplan

G Halaman Recruitment Godplan Mobile

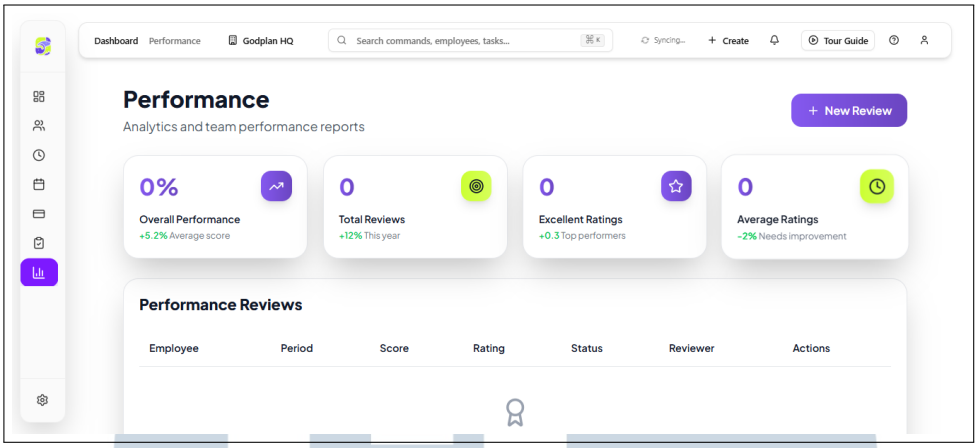
Gambar 3.30 menampilkan Halaman *Recruitment* pada versi *Desktop* aplikasi *Godplan*, yang dirancang untuk mengelola proses rekrutmen dan data pelamar kerja secara terintegrasi. Antarmuka ini menyajikan ringkasan statistik lowongan melalui kartu metrik yang mencakup *Total Jobs*, *Active Jobs*, *Total Applicants*, dan *Filled Positions*. Halaman ini juga dilengkapi dengan tombol aksi *Post a Job* untuk mempublikasikan lowongan baru, serta tabel *Job Postings* yang dapat disaring berdasarkan status (*Open*, *Closed*, *On-Hold*) untuk mempermudah pemantauan kandidat di setiap posisi.



Gambar 3.31. Halaman Recruitment Desktop Godplan

H Halaman Performance Godplan Mobile

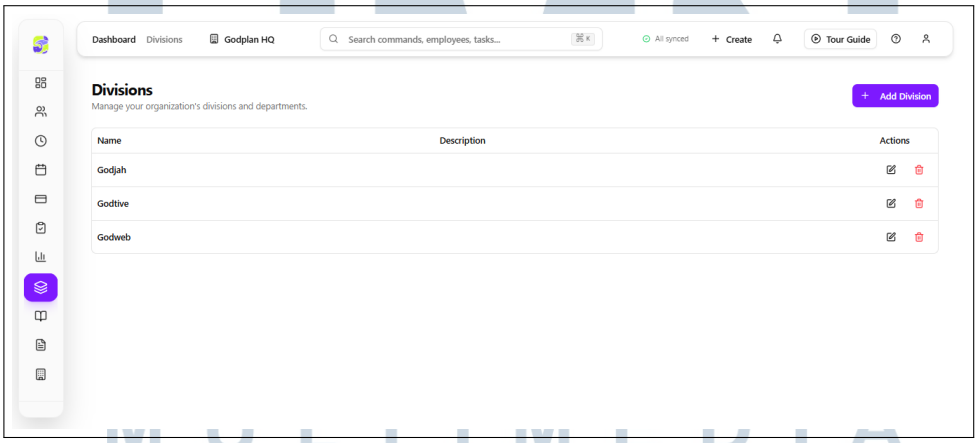
Gambar 3.31 menampilkan Halaman *Performance* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi sebagai pusat analitik dan laporan evaluasi kinerja tim. Antarmuka ini menyajikan indikator kualitas kerja melalui kartu statistik utama yang meliputi *Overall Performance*, *Total Reviews*, *Excellent Ratings*, dan *Average Ratings* untuk memberikan gambaran umum produktivitas perusahaan. Fitur ini juga memfasilitasi penilaian karyawan secara mendetail melalui tabel *Performance Reviews* yang memuat informasi skor, peringkat (*Rating*), dan status evaluasi, serta menyediakan tombol *New Review* untuk memulai sesi penilaian baru.



Gambar 3.32. Halaman Performance Desktop Godplan

I Halaman Division Godplan Mobile

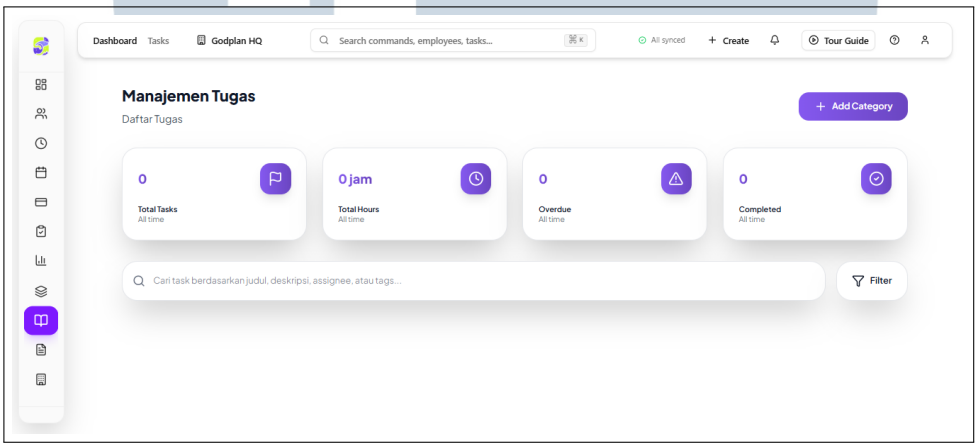
Gambar 3.32 menampilkan Halaman *Division* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi untuk mengelola struktur organisasi perusahaan. Halaman ini memuat tabel daftar divisi yang terdiri dari kolom *Name*, *Description*, dan *Actions*, dengan contoh entri data seperti Godjah, Godtive, dan Godweb. Pengguna dengan hak akses administratif dapat menggunakan tombol *Add Division* untuk menambah unit kerja baru, serta menggunakan ikon pada kolom aksi untuk menyunting (*edit*) atau menghapus (*delete*) data divisi yang ada.



Gambar 3.33. Halaman Division Desktop Godplan

J Halaman Tasks Godplan Mobile

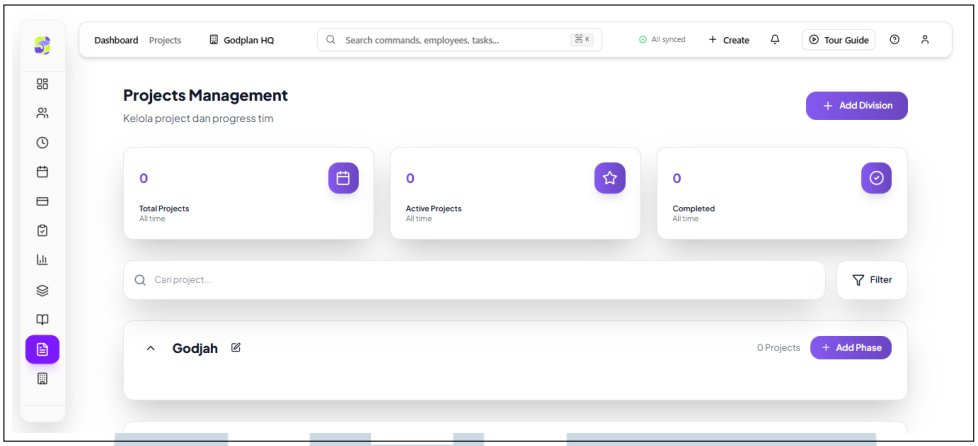
Gambar 3.33 menampilkan Halaman *Tasks* pada versi *Desktop* aplikasi *Godplan*, yang dirancang untuk memfasilitasi manajemen tugas secara komprehensif. Antarmuka ini menyajikan ringkasan produktivitas melalui kartu statistik yang menampilkan jumlah *Total Tasks*, akumulasi waktu kerja (*Total Hours*), tugas yang melewati tenggat waktu (*Overdue*), serta tugas yang telah diselesaikan (*Completed*). Selain itu, halaman ini menyediakan fitur pencarian (*search bar*) yang mendetail serta tombol aksi *Add Category* untuk membantu pengguna mengorganisasi struktur tugas dengan lebih efisien.



Gambar 3.34. Halaman Tasks Desktop Godplan

K Halaman Projects Godplan Mobile

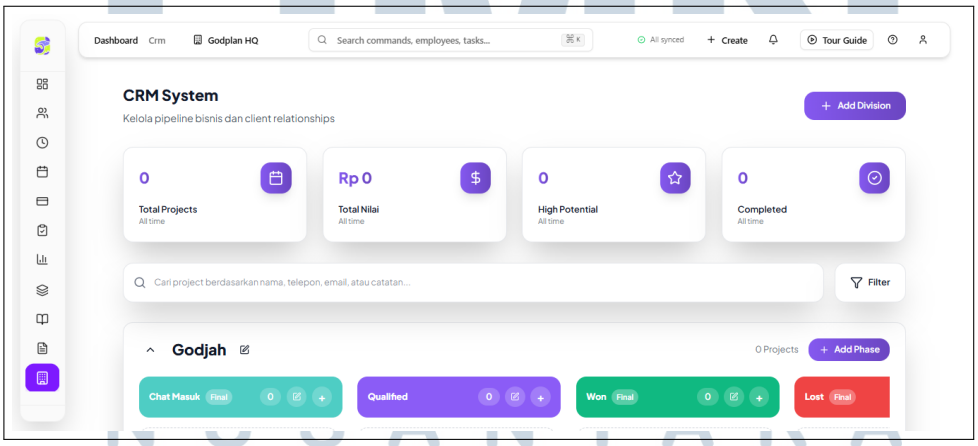
Gambar 3.34 menampilkan Halaman *Projects* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi sebagai pusat manajemen proyek dan pemantauan kemajuan tim. Antarmuka ini menyediakan ringkasan kuantitatif melalui kartu statistik yang mencakup *Total Projects*, *Active Projects*, dan proyek yang telah selesai (*Completed*). Selain fitur pencarian dan penyaringan data (*Filter*), halaman ini memungkinkan administrator untuk mengelola struktur organisasi proyek melalui tombol *Add Division* serta menambahkan tahapan kerja baru secara spesifik menggunakan tombol *Add Phase*.



Gambar 3.35. Halaman Project Desktop Godplan

L Halaman CRM Godplan Mobile

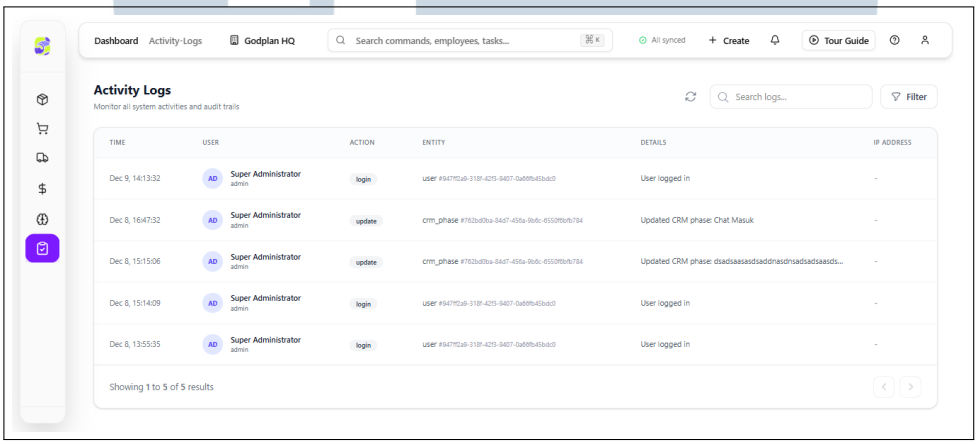
Gambar 3.35 menampilkan Halaman *CRM System* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi untuk mengelola hubungan klien dan *pipeline* bisnis secara visual. Antarmuka ini menyajikan ringkasan kinerja melalui kartu statistik yang mencakup *Total Projects*, *Total Nilai* pendapatan, potensi proyek tinggi (*High Potential*), serta proyek yang telah selesai (*Completed*). Di bagian bawah, terdapat daftar proyek yang dikelompokkan per divisi (contoh: Godjah) dengan visualisasi tahapan status berwarna seperti *Chat Masuk*, *Qualified*, *Won*, dan *Lost*, serta dilengkapi tombol *Add Division* dan *Add Phase* untuk kustomisasi alur kerja.



Gambar 3.36. Halaman CRM Desktop Godplan

M Halaman *Activity Log* Godplan Mobile

Gambar 3.36 menampilkan Halaman *Activity Logs* pada versi *Desktop* aplikasi *Godplan*, yang berfungsi sebagai sistem audit untuk memantau jejak aktivitas pengguna secara rinci. Antarmuka ini menyajikan tabel riwayat tindakan yang mencakup informasi waktu (*Time*), identitas pengguna (*User*), jenis aksi (*Action*) seperti *login* atau *update*, serta entitas sistem yang terdampak. Selain kolom rincian aktivitas (*Details*), halaman ini juga menyediakan fitur pencarian (*Search logs*) dan penyaringan (*Filter*) untuk memudahkan administrator dalam melakukan penelusuran keamanan atau pemecahan masalah teknis.



Gambar 3.37. Halaman *Activity Log Desktop* Godplan

3.3.7 Blackbox Testing API Login

Pengujian blackbox pada endpoint Login API aplikasi mobile GodPlan dilaksanakan pada tanggal 25 Desember 2025 menggunakan *automated testing script*. Pengujian dilakukan terhadap endpoint `http://localhost:8080/api/v1/auth/login` dengan total 10 skenario yang mencakup alur fungsional, validasi, keamanan, dan *edge cases*.

A Hasil Keseluruhan

Dari 10 *test cases* yang dijalankan, semua berhasil PASS (100%). Tidak ditemukan *critical issues* atau *minor issues* yang tersisa. Validasi format email yang sebelumnya bermasalah telah diperbaiki.

B Analisis Hasil Pengujian

1. Keamanan: *Excellent* - Perlindungan terhadap *SQL Injection* berfungsi dengan baik, hashing password menggunakan *Bcrypt* valid, dan pesan error bersifat generik sehingga tidak membocorkan informasi sensitif.
2. Autentikasi: *Working perfectly* - *JWT token* dan *refresh token* berhasil digenerate dengan benar. Alur login dengan kredensial valid berjalan sesuai ekspektasi.
3. Validasi Input: *Excellent* - Semua validasi berfungsi dengan baik termasuk validasi format email yang sebelumnya bermasalah.
4. Kinerja: *Excellent* - Waktu respons rata-rata di bawah 300ms, memenuhi standar untuk aplikasi mobile.

C Tabel Hasil Testing Login API

Berikut adalah hasil detail dari pengujian Login API yang mencakup berbagai skenario testing:

Tabel 3.2. Hasil Blackbox Testing - Login Mobile API GodPlan

ID	Skenario Testing	Input Data	E. Status	A. Status	Result	Notes
001	Login dengan kredensial valid	Email: admin@godplan.com, Password: password123	200	200	PASS	*Token* & *refresh token* berhasil digenerate
002	Login dengan email tidak terdaftar	Email: notfound@example.com, Password: password123	401	401	PASS	Error message: ”Invalid email or password”
003	Login dengan password salah	Email: admin@godplan.com, Password: wrongpassword	401	401	PASS	Error message: ”Invalid email or password”

ID	Skenario Testing	Input Data	E. Status	A. Status	Result	Notes
004	Login dengan email kosong	Email: ””, Password: password123	400	400	PASS	*Validation error triggered*
005	Login dengan password kosong	Email: admin@godplan.com, Password: ””	400	400	PASS	*Validation error triggered*
006	Login dengan kedua field kosong	Email: ””, Password: ””	400	400	PASS	*Validation error triggered*
007	Login dengan format email tidak valid	Email: invalidemail, Password: password123	400	400	PASS	Error: ”email must be a valid email”
008	Login dengan spasi di email	Email: ” admin@godplan.com ”, Password: password123	401	401	PASS	Error: ”Invalid email or password”
009	Login dengan email *uppercase*	Email: ADMIN@GODPLAN.COM, Password: password123	401	401	PASS	Email adalah *case-sensitive*
010	Login dengan *SQL Injection attempt*	Email: admin@godplan.com’ OR ’1’=’1, Password: anything	401	401	PASS	*SQL Injection blocked successfully*

D Bug yang Ditemukan dan Diperbaiki

Terdapat satu bug minor yang berhasil diidentifikasi dan diperbaiki selama proses testing:

1. BUG-002: Email Format Validation Missing
   Severity: Minor
   Status: FIXED

Deskripsi: Validasi format email pada layer input validation sebelumnya tidak berfungsi. Email tanpa @ symbol masih diterima dan diproses hingga *database query*.

Solusi: Menambahkan validasi *email* pada *struct tag LoginRequest*.

```
// File: pkg/models/auth.go
type LoginRequest struct {
    Email    string `json:"email" binding:"required,email"`
    // Added ,email
    Password string `json:"password" binding:"required"`
}

```

Kode 3.8: Perbaikan Validasi Email pada LoginRequest

E Statistik Hasil Testing Login

Statistik keseluruhan hasil pengujian Login API dapat dilihat pada tabel berikut:

Tabel 3.3. Statistik Hasil Testing Login API

Metric	**Value**
Total Tests	10
Passed	10
Failed	0
Pass Rate	100%
Critical Issues	0
Minor Issues	0
Average Response Time	300ms

3.3.8 Blackbox Testing API Attendance

Pengujian blackbox pada endpoint Attendance API aplikasi mobile GodPlan dilaksanakan pada tanggal 26 Desember 2025 menggunakan *automated testing script*. Pengujian dilakukan terhadap endpoint `http://localhost:8080/api/v1/attendance` dengan total 20 skenario yang mencakup empat fungsi utama: *Check Location, Clock In, Clock Out, dan Get Attendance History*.

A Hasil Keseluruhan

Dari 20 *test cases* yang dijalankan, semua berhasil PASS (100%). *Critical bug* pada endpoint *Get Attendance* yang sebelumnya menyebabkan *error 500* telah berhasil diperbaiki.

B Ringkasan Hasil per Section

Ringkasan hasil pengujian per section dapat dilihat pada tabel berikut:

Tabel 3.4. Ringkasan Hasil Testing per Section Attendance API

Section	Total Tests	Passed	Failed	Pass Rate
Check Location	4	4	0	100%
Clock In	6	6	0	100%
Clock Out	4	4	0	100%
Get Attendance	6	6	0	100%
TOTAL	**20**	**20**	**0**	**100%**

C Analisis Hasil Pengujian

1. *Check Location* (100% PASS): *Endpoint* berfungsi sempurna. Validasi lokasi dan perhitungan jarak bekerja dengan baik.
2. *Clock In* (100% PASS): Semua fungsionalitas berjalan sesuai ekspektasi termasuk validasi lokasi, *duplicate check*, dan opsi foto *selfie*.
3. *Clock Out* (100% PASS): Endpoint berfungsi sempurna. Semua validasi dan perhitungan *total hours* bekerja sesuai ekspektasi.
4. *Get Attendance* (100% PASS): *Critical bug* telah diperbaiki. Semua *request* GET berhasil dengan status code 200.

D Tabel Hasil Testing Attendance API

Berikut adalah hasil *detail* dari pengujian *Attendance* API yang mencakup semua fungsi utama:

Tabel 3.5. Hasil Blackbox Testing - Attendance Mobile API GodPlan

ID	Skenario Testing	Input Data	E. Status	A. Status	Hasil	Keterangan
BAGIAN 1: CHECK LOCATION ENDPOINT						
LOC-001	Check lokasi dalam radius kantor	Lat: -6.301286, Lng: 106.650010	200	200	LULUS	*in_range: true*, *distance: 0m*
LOC-002	Check lokasi luar radius kantor	Lat: -6.3100, Lng: 106.6600	200	200	LULUS	*in_range: false*, *distance: 1469m*
LOC-003	Check lokasi tanpa autentikasi	No auth header	401	401	LULUS	*Auth required*
LOC-004	Check lokasi koordinat invalid	Lat: 999, Lng: 999	200	200	LULUS	*Distance calculated*
BAGIAN 2: CLOCK IN ENDPOINT						
CLKIN-001	Clock dalam radius kantor	Lat: -6.301286, Lng: 106.650010, Photo: base64	201	201	LULUS	*Clock in successful*
CLKIN-002	Clock luar radius tanpa force	Lat: -6.3100, Lng: 106.6600, Force: false	400	400	LULUS	*Error: "Lokasi di luar jangkauan"*
CLKIN-003	Clock dua kali hari sama	Sama seperti CLKIN-001	400	400	LULUS	*Error: "Sudah melakukan Clock In"*

ID	Skenario Testing	Input Data	E. Status	A. Status	Hasil	Keterangan
CLKIN-004	Clock tanpa foto selfie	Photo: "", Field valid	201	201	LULUS	*Attendance created without photo*
CLKIN-005	Clock tanpa autentikasi	No auth header	401	401	LULUS	*Auth required*
CLKIN-006	Clock dengan koordinat kosong	No lat/lng	400	400	LULUS	*Error: "Lokasi di luar jangkauan"*
BAGIAN 3: CLOCK OUT ENDPOINT						
CLOUT-001	Clock out dalam radius kantor	Lat: -6.301286, Lng: 106.650010, Photo: base64	200	200	LULUS	*Clock out successful*
CLOUT-002	Clock out luar radius tanpa force	Lat: -6.3100, Lng: 106.6600, Force: false	400	400	LULUS	*Error: "Lokasi di luar jangkauan"*
CLOUT-003	Clock out dua kali hari sama	Sama seperti CLOUT-001	400	400	LULUS	*Error: "Belum Clock In atau sudah Clock Out"*
CLOUT-004	Clock out tanpa autentikasi	No auth header	401	401	LULUS	*Auth required*
BAGIAN 4: GET ATTENDANCE HISTORY						
GETH-001	Get attendance tanpa filter	No query params	200	200	LULUS	*Array of attendance records*

ID	Skenario Testing	Input Data	E. Status	A. Status	Hasil	Keterangan
GETH-002	Get attendance dengan limit	?limit=10	200	200	LULUS	*Limit respected*
GETH-003	Get attendance filter date	?date=2025-12-30	200	200	LULUS	*Filter by date working*
GETH-004	Get attendance tanpa autentikasi	No auth header	401	401	LULUS	*Auth required*
GETH-005	Get attendance date invalid	?date=invalid-date	200	200	LULUS	*Gracefully handled*

E Bug yang Ditemukan dan Diperbaiki

Terdapat satu *bug critical* yang berhasil diidentifikasi dan diperbaiki selama proses *testing*:

1. BUG-001: *Get Attendance Endpoint Failing*
   Severity: Critical
   Status: FIXED
   Deskripsi: Semua *request* ke *endpoint* `/attendance (GET)` menghasilkan *error 500 Internal Server Error* karena *mismatch* nama kolom antara *query* dan skema database.
   Solusi: Memperbaiki *query* dengan mengganti nama kolom sesuai skema database aktual.

```
// File: pkg/handlers/attendance.go line 456-467
// BEFORE (BROKEN):
SELECT latitude, longitude, photo_selfie

// AFTER (FIXED):
SELECT check_in_lat as latitude, check_in_lng as longitude,
       check_in_photo as photo_selfie
```

```
8
```

Kode 3.9: Perbaikan Query Database pada Attendance Endpoint

F Statistik Hasil Testing Attendance

Statistik keseluruhan hasil pengujian *Attendance* API dapat dilihat pada tabel berikut:

Tabel 3.6. Statistik Hasil Testing Attendance API

Metric	Value
Total Tests	20
Passed	20
Failed	0
Pass Rate	100%
Critical Issues	0
Minor Issues	0
Average Response Time	150ms

G Kesimpulan Akhir

API *Login* dan *Attendance* untuk aplikasi mobile GodPlan telah berhasil melewati semua *test cases* dengan *pass rate* 100%. Kedua *bug* yang ditemukan (validasi format email dan *column mismatch* pada *query Get Attendance*) telah berhasil diperbaiki.

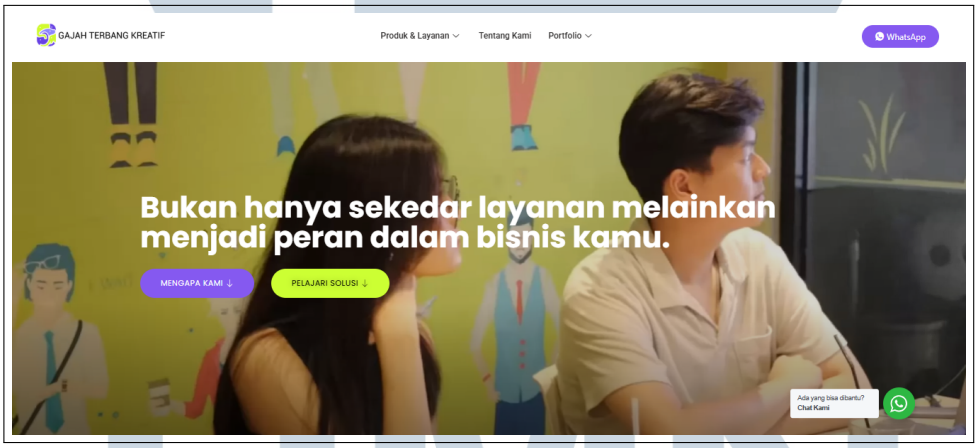
Status: *PRODUCTION READY*. Sistem memenuhi semua kriteria untuk *deployment* ke lingkungan produksi:

1. Total 30 *test cases* (*Login*: 10, *Attendance*: 20) semua PASS
2. Tidak ada *bug* yang tersisa
3. Semua *security tests passed*
4. Performa excellent (semua *endpoint* 500ms)
5. Validasi input bekerja dengan sempurna

Test Execution Time: ∼45 detik (mencakup semua 30 *test cases*)

3.3.9 Website PT. Gajah Terbang Kreatif

Gambar 3.40 menunjukan tentang website *company profile* PT. Gajah Terbang Kreatif dengan menggunakan *domaingodjahstudio.com* dirancang dengan pendekatan *modern* dan *professional* guna merepresentasikan identitas perusahaan sebagai agensi kreatif yang dinamis. Dari sisi desain, website ini menggunakan *hero image* yang menampilkan interaksi manusia untuk memberikan kesan *warm* dan *collaborative*, dipadukan dengan tipografi putih yang kontras dan tegas. Pesan utama yang ingin disampaikan menekankan pada nilai kemitraan strategis, di mana perusahaan tidak hanya memberikan layanan tetapi mengambil peran dalam bisnis klien. Navigasi situs dibuat intuitif dengan menu yang terstruktur, serta dilengkapi dengan fitur *call to action* (CTA) yang mencolok dan integrasi tombol WhatsApp untuk memudahkan konversi serta komunikasi langsung dengan calon pelanggan.



Gambar 3.38. Website PT. Gajah Terbang Kreatif

3.3.10 Website Client

Pada subbab ini dijelaskan hasil pengembangan berbagai situs web untuk klien, baik yang sepenuhnya dibangun menggunakan *WordPress* maupun yang dikembangkan dengan pendekatan *hybrid* yang menggunakan *WordPress*, *PHP*, *HTML*, dan *CSS*. Pembahasan meliputi implementasi fitur dasar hingga pengembangan fitur khusus sesuai kebutuhan klien, termasuk penyesuaian fungsional dan visual agar setiap situs dapat berjalan optimal dan memenuhi standar yang telah ditetapkan.

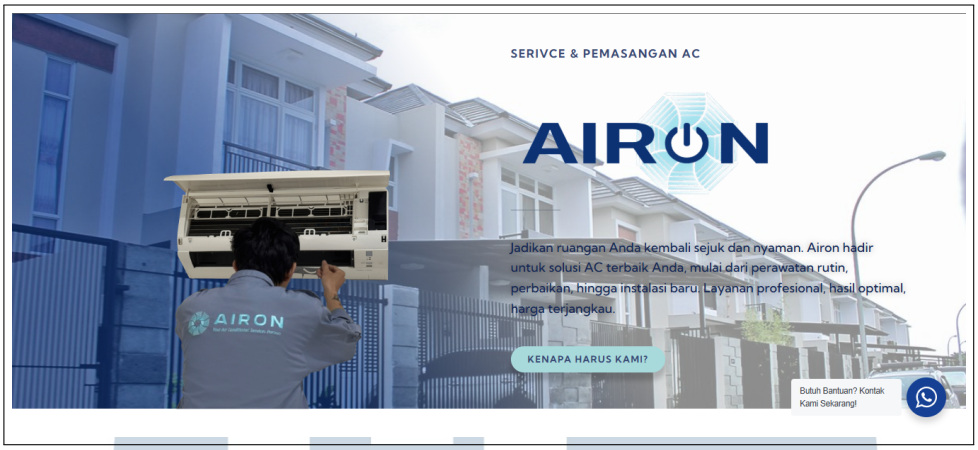
A Website Ramein

Situs web Rameinaja.com dibangun sebagai platform sosial dan direktori acara yang ditujukan bagi kalangan muda untuk menemukan event klub malam, venue nongkrong, dan komunitas hangout. Pengembangan situs ini menekankan pada antarmuka yang dinamis dan *youth-oriented*, dengan fitur utama seperti pencarian dan promosi event, sistem ”Ramein” untuk mengumpulkan peserta acara, serta pembangunan komunitas sosial. Penyesuaian fungsional difokuskan pada kemudahan penggunaan, yang ditunjukkan melalui proses empat langkah yang sederhana untuk bergabung atau membuat event, guna mendorong interaksi dan partisipasi aktif pengguna.

Gambar 3.39. Website Ramein

B Website Airon

Airon.site dikembangkan sebagai situs web *landing pag*e layanan jasa yang mempromosikan bisnis perawatan dan perbaikan AC (*Air Conditioner*). Situs ini dirancang dengan struktur informasi yang sangat jelas dan transparan, menampilkan detail layanan seperti cuci AC, isi freon, dan pemasangan beserta daftar harga, syarat, serta FAQ. Pendekatan pengembangan bertujuan untuk menjawab langsung pertanyaan calon pelanggan dan membangun kepercayaan melalui informasi yang lengkap serta testimoni, dengan *call-to-action* (CTA) yang menonjol seperti nomor kontak untuk memudahkan konversi menjadi pesanan jasa.



Gambar 3.40. Website Airon

C Website NfOptical

Nfopticalofficial.com berfungsi sebagai situs web perusahaan dan katalog digital untuk optik yang menyediakan layanan kesehatan mata serta penjualan produk seperti kacamata dan softlens. Pengembangan situs ini berfokus pada penyajian katalog produk yang profesional dengan galeri foto berkualitas tinggi dan deskripsi detail untuk berbagai frame dan lensa. Fitur seperti artikel edukasi dan penekanan pada kredibilitas sertifikasi optik diimplementasikan untuk mengedukasi pengunjung dan memperkuat posisi bisnis sebagai penyedia layanan yang terpercaya di industri kesehatan mata.

Gambar 3.41. Website NfOptical

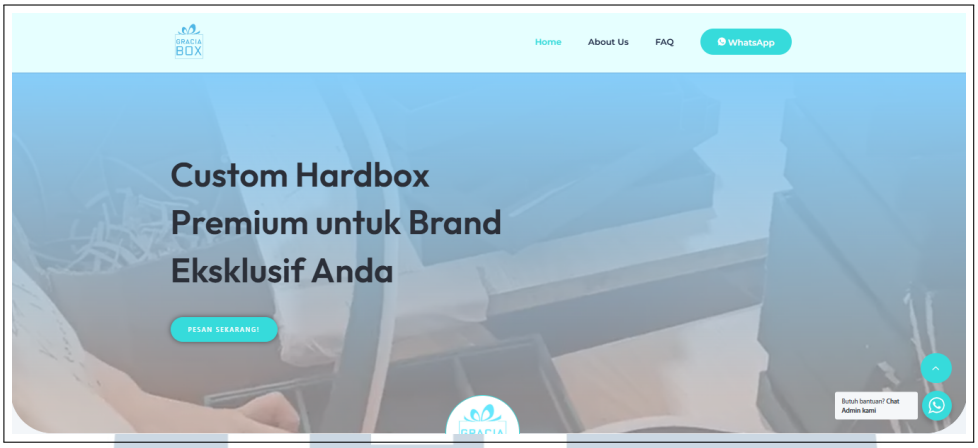
D Website Miniaturkapal

Miniaturkapal.com merupakan situs web portofolio sekaligus toko online yang memamerkan dan memasarkan karya miniatur kapal buatan tangan dengan detail tinggi. Pengembangan visual situs ini mengutamakan estetika yang elegan dan klasik untuk mencerminkan nilai artistik produk, dilengkapi dengan galeri foto yang memungkinkan *zoom in* untuk menampilkan kerumitan dan ketelitian pengerjaan. Selain sebagai galeri, situs juga dilengkapi dengan fitur pemesanan, baik untuk model standar maupun *custom*, serta konten yang menjelaskan proses pembuatan, sehingga berfungsi efektif sebagai platform pemasaran sekaligus edukasi bagi penggemar dan calon pembeli.

Gambar 3.42. Website Miniaturkapal

E Website Graciabox

Graciabox.id adalah situs web perusahaan yang menawarkan *hardbox* kemasan *premium* dan *custom* untuk kebutuhan *retail* maupun korporat (B2B). Pengembangannya menekankan pada presentasi visual yang mewah dan profesional, menampilkan berbagai model kemasan dengan kemungkinan konfigurasi 3D. Fitur kunci yang diimplementasikan adalah sistem ”*Create Your Own Box*” yang memungkinkan kustomisasi ukuran dan desain secara *user-friendly*, didukung oleh informasi korporat yang lengkap seperti visi-misi, profil perusahaan, dan FAQ detail untuk melayani kebutuhan klien bisnis yang menuntut kejelasan dan fleksibilitas.



Gambar 3.43. Website Graciabox

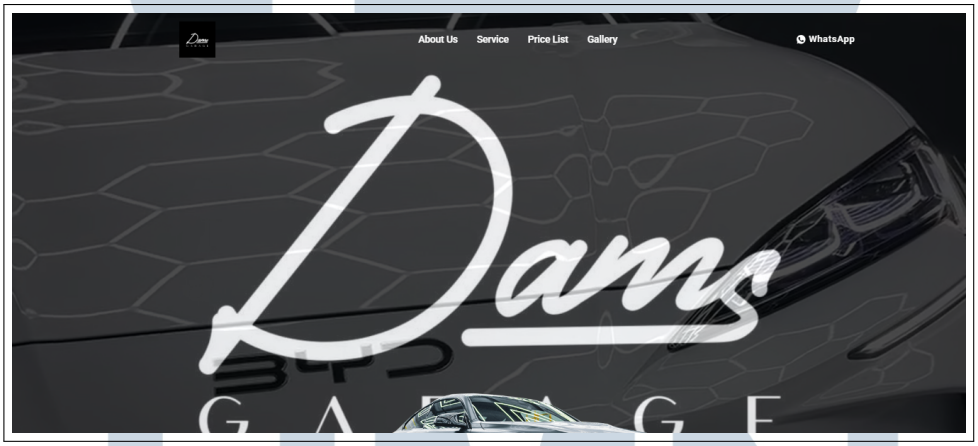
F Website Recon

Reconstruction.id adalah situs web yang dikembangkan untuk perusahaan *property* dengan fokus pada layanan *rekonstruksi* dan renovasi bangunan. Situs ini berfungsi sebagai platform untuk memamerkan portofolio proyek-proyek konstruksi sebelumnya, menjelaskan layanan spesialis seperti perbaikan struktural dan desain ulang, serta membangun kredibilitas di industri properti. Pengembangannya menekankan pada presentasi visual yang kuat melalui galeri foto dan studi kasus proyek, dilengkapi dengan informasi layanan yang jelas untuk menarik klien potensial yang membutuhkan jasa konstruksi dan renovasi yang andal.

Gambar 3.44. Website Recon

G Website Damsgarage

Damsgarage.id adalah situs web yang dikembangkan untuk penyedia jasa perawatan otomotif profesional dengan fokus utama pada layanan *detailing*, *coating*, dan *car wash*. Situs ini berfungsi sebagai platform digital untuk memamerkan kualitas hasil pengerjaan estetika kendaraan, menjelaskan berbagai paket layanan spesialis mulai dari pembersihan mendalam hingga perlindungan cat tingkat lanjut, serta membangun kepercayaan pelanggan di industri perawatan mobil. Pengembangannya menekankan pada aspek visual yang menarik guna menampilkan detail kemilau kendaraan setelah perawatan, dilengkapi dengan informasi layanan yang komprehensif untuk memudahkan pemilik kendaraan dalam memilih jasa perawatan yang paling sesuai dengan kebutuhan otomotif mereka.



Gambar 3.45. Website Damsgarage

3.4 Kendala dan Solusi yang Ditemukan

Selama pengembangan *frontend* sistem *GodPlan*, terdapat tiga kendala teknis utama:

1. Pengguna dapat memanipulasi koordinat *GPS* melalui *developer tools browser*, yang memungkinkan proses absen dilakukan dari luar lokasi kantor yang telah ditentukan.
2. Terjadi penurunan performa (*lag*) saat me-*render* grafik statistik (*Payroll/Attendance*) dan tabel data dalam jumlah besar secara bersamaan di halaman *Dashboard*, terutama pada perangkat dengan spesifikasi rendah.

3. Terdapat risiko kegagalan pengiriman data absensi (foto & lokasi) saat pengguna melakukan proses *Clock-in* di area dengan koneksi *internet* lambat atau terputus, yang menyebabkan aplikasi *stuck* atau data hilang.

Solusi yang diterapkan untuk mengatasi kendala tersebut adalah:

1. Implementasi validasi *integrity* data dilakukan dengan *hashing* koordinat *GPS* dan *timestamp*, serta penambahan validasi *side-by-side* dengan data *WiFi fingerprint*.
2. Penerapan teknik *Lazy Loading* pada komponen grafik dan *Virtualization* (misalnya *react-window*) pada tabel data, agar *DOM* hanya melakukan *rendering* elemen yang terlihat di layar saja untuk menjaga performa tetap ringan.
3. Implementasi *Optimistic UI* dan mekanisme *Offline-First* menggunakan *Local Storage*, di mana aplikasi akan menyimpan data absensi secara lokal terlebih dahulu dan menyinkronkannya ke *server* secara otomatis (*background sync*) saat koneksi kembali stabil.