BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Magang dilaksanakan di PT. Winnicode Garuda Teknologi, dengan topik yang terpilih adalah proyek perancangan *Smart Attendance System*. Magang dengan posisi *fullstack developer* berada langsung di bawah naungan ketua pelaksana program magang sekaligus *supervisor* magang. Untuk mendukung kelancaran kegiatan magang, *supervisor* melakukan bimbingan secara berkala setiap akhir minggu.

3.2 Tugas yang Dilakukan

Tanggung jawab dalam proyek termasuk perencanaan serta implementasi proyek perancangan dan pembangunan *Smart Attendance System* untuk keperluan internal perusahaan. Tugas yang diberikan bermula dari proposal proyek, desain sistem dengan pembuatan diagram *flowchart*, ERD serta prototipe, sampai tahap pembangunan dengan menggunakan *framework* React serta Laravel. Tabel 3.1 menunjukkan *job description* yang dilakukan selama magang:

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

No	Tugas
1	Melakukan perancangan desain alur sistem pada proyek *Smart Attendance System*.
2	Melakukan perancangan desain UI/UX sistem pada proyek *Smart Attendance System*.
3	Membangun *Smart Attendance System* berbasis *website* berdasarkan desain yang telah dirancang.
4	Melakukan optimalisasi *Smart Attendance System* dari segi performa, efisiensi, serta keamanan.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.2.

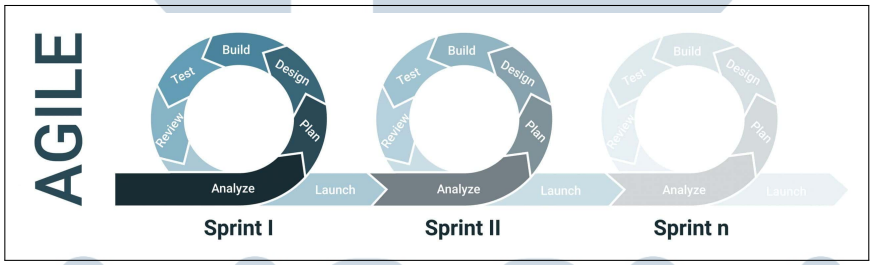
Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Memulai tahap perencanaan proyek dan melakukan penyusunan proposal.
2	Melakukan perancangan *flow* sistem dalam bentuk *sitemap*, *flowchart*, serta *entity relationship diagram* (ERD).
3	Merancang desain UI/UX untuk fitur utama sistem, termasuk menu *sign in/out*, menu *application*, serta menu *history*.
4	Memulai perancangan desain UI/UX untuk tampilan admin.
5	Melakukan penyempurnaan desain serta menyelesaikan perancangan prototipe sistem.
6	Membangun tampilan *login* pengguna serta menyusun *database* dengan menggunakan PostgreSQL.
7	Membangun tampilan menu *sign in/out*, *application*, dan *history*.
8	Merancang arsitektur *server-side* untuk menangani pemrosesan data pada setiap menu.
9	Mengimplementasikan koneksi *server-side* dengan *client-side*.
10	Memulai pembangunan tampilan halaman *admin*, termasuk menu *dashboard*, *manage employee*, *manage attendance*, dan *manage shift*.
11	Mengintegrasikan tampilan halaman *admin* dengan *server-side*.
12	Melakukan pengujian secara bertahap, serta melakukan *debugging*.
13	Melakukan penyempurnaan fitur dan optimasi tampilan, serta menyusun dokumentasi teknis dan dokumentasi pengguna, serta melakukan *deployment*.
14	Menyelesaikan tahap *deployment website* di *server lokal* perusahaan.

3.3.1 Perencanaan Proyek

Metode *Agile* digunakan dalam proses pembangunan sistem berbasis *website*. Penerapan metode ini dimulai dari tahap identifikasi masalah dan kebutuhan sistem (*analyze*), dilanjutkan dengan tahap perencanaan (*plan*), perancangan (*design*), pembangunan (*build*), pengujian (*test*), serta peninjauan hasil (*review*). Pendekatan ini bersifat iteratif, sehingga memungkinkan adanya penyesuaian terhadap perubahan kebutuhan atau pengembangan fitur selama proyek berlangsung.

Proyek dilaksanakan dengan membaginya ke dalam beberapa iterasi atau *sprint*. Setiap *sprint* difokuskan pada penyelesaian sejumlah tugas atau fitur tertentu dalam kurun waktu satu minggu. Di akhir setiap minggu, dilakukan sesi *review* untuk mengevaluasi kesesuaian hasil dengan kriteria yang telah ditentukan. Iterasi berikutnya akan dilakukan apabila diperlukan perbaikan atau penambahan fitur baru.



Gambar 3.1. Metode *Agile*

Sumber: [6]

Tahap perencanaan pengembangan *Smart Attendance System* berbasis *website* dimulai dengan diskusi bersama *user* untuk menentukan kebutuhan fitur, desain tampilan, serta pemilihan teknologi yang tepat guna mendukung kelancaran proyek. Berdasarkan hasil pertimbangan, teknologi *software* yang digunakan dalam pengembangan sistem meliputi:

- **Laravel,** adalah *framework* PHP yang digunakan dalam pembangunan *back-end*. *Laravel* menyediakan berbagai fitur seperti *routing, middleware,* dan *ORM (Eloquent)* yang memudahkan *developer* dalam memahami alur pemrosesan pada *server-side* serta dalam merumuskan relasi antar atribut di dalam *database*.

- **React,** yaitu *framework* untuk pengembangan bagian *front-end* berbasis

JavaScript. React menggunakan *virtual DOM* dan mendukung definisi komponen yang dapat digunakan kembali.

- **Tailwind,** merupakan *framework* CSS dengan pendekatan *utility-first* yang mempermudah proses *styling*. Dengan demikian, proses desain dapat dilaksanakan dengan lebih cepat tanpa perlu menuliskan setiap *style* secara manual.
- **Flowbite React,** merupakan *library* berbasis Tailwind CSS yang menyediakan berbagai komponen berbasis ReactJS untuk mempermudah perancangan antarmuka *website*, tanpa perlu merancang ulang setiap komponen pada website.
- **Recharts,** merupakan *library* visualisasi data berbasis ReactJS yang digunakan untuk menampilkan komponen grafik dan diagram interaktif, seperti *pie chart* dan *bar chart*.
- **TensorFlow.js,** merupakan *library* pembelajaran mesin berbasis JavaScript yang memungkinkan pengembangan dan penerapan model pembelajaran mesin pada aplikasi berbasis JavaScript.
- **PostgreSQL,** merupakan sistem basis data yang berfungsi menyimpan dan mengelola data aplikasi. *PostgreSQL* digunakan karena mendukung banyak jenis tipe data yang spesifik.

3.3.2 Perancangan Sistem

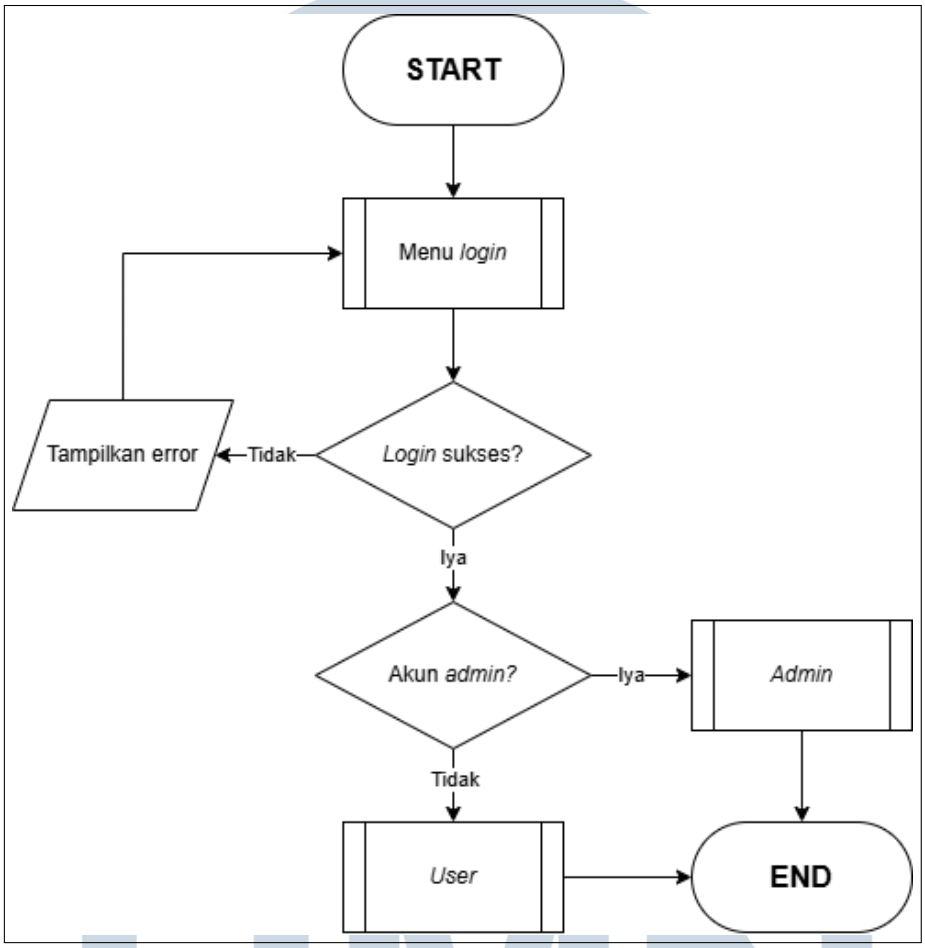
A Flowchart

Flowchart menggambarkan cara kerja dan alur pemrosesan dari *Smart Attendance System*, mulai dari ketika pengguna mengakses *website* hingga selesai. *Flowchart* juga memperlihatkan kapan sistem menangani proses komunikasi dengan basis data.

A.1 Flowchart Utama

Flowchart pada Gambar 3.2 menggambarkan alur proses utama sistem bermula dari saat pengguna pertama kali mengakses *website* hingga pengguna

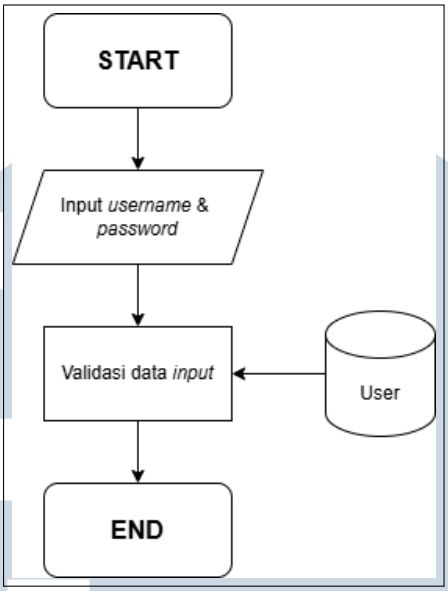
keluar dari *website*. Pada *flowchart*, terlihat bahwa tampilan website dipengaruhi oleh tipe akun yang *login*.



Gambar 3.2. *Flowchart Utama*

A.2 Flowchart Menu Login

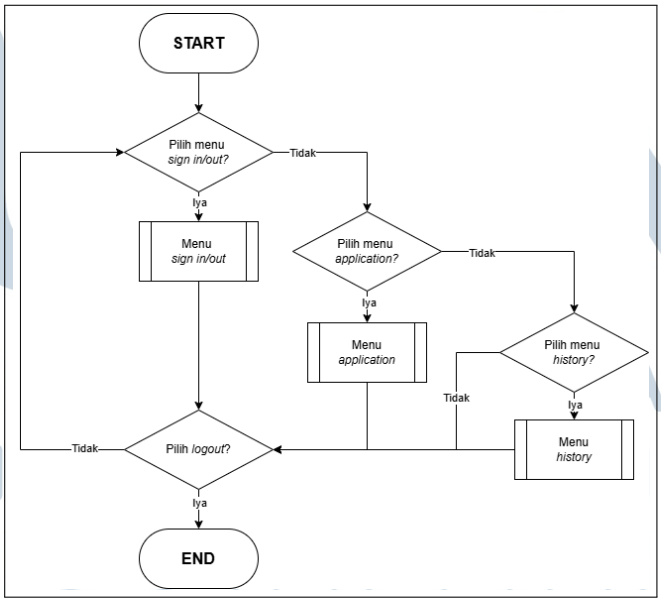
Flowchart pada Gambar 3.3 menggambarkan alur proses login yang merupakan langkah pertama saat pengguna mengakses *website*. Pengguna yang sudah memiliki akun dapat login menggunakan kredensial akun yang terkait. Seperti terlihat pada Gambar 3.2, jika akun yang masuk berupa admin, pengguna akan dialihkan ke menu admin. Selain itu, pengguna dialihkan ke menu *user*.



Gambar 3.3. *Flowchart* Menu *Login*

A.3 Flowchart User

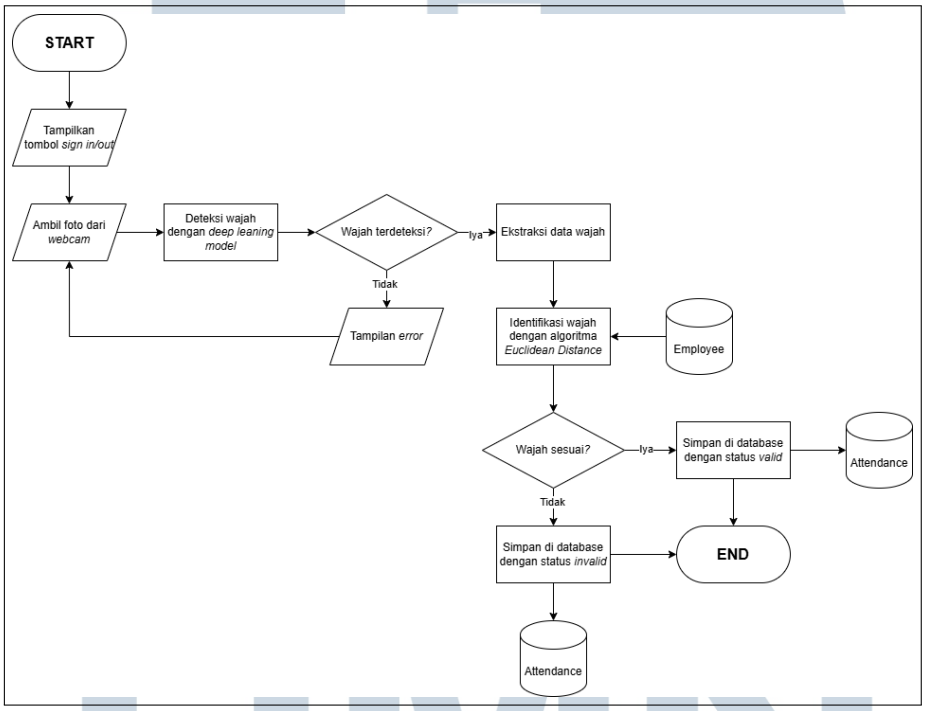
Flowchart pada Gambar 3.4 menggambarkan alur proses pada menu pengguna. Pengguna memiliki beberapa menu yang dapat diakses mulai dari menu *sign in/out*, *application*, serta *history*, masing-masing dengan fungsi tersendiri.



Gambar 3.4. *Flowchart User*

A.3.1 Flowchart Menu Sign in/out

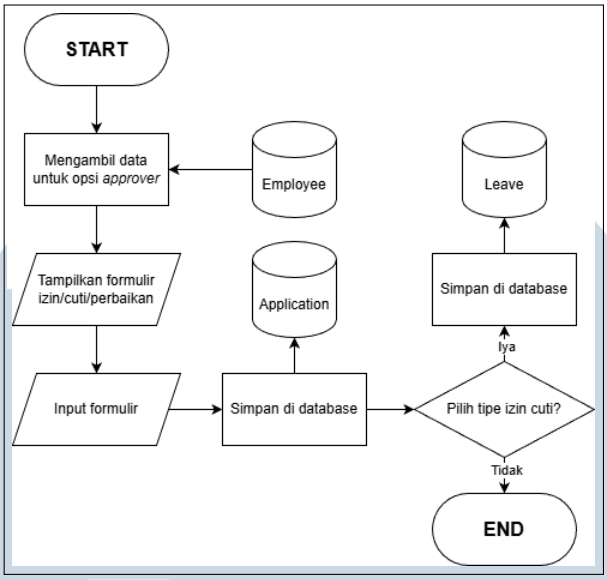
Flowchart pada Gambar 3.5 menggambarkan alur proses ketika pengguna mengakses menu *sign in/out*. Pada menu ini, pengguna dapat melakukan presensi dengan mengambil foto. Foto yang diambil akan diproses menggunakan *deep learning model* untuk fitur *face detection* dan *extraction* serta algoritma *Euclidean Distance* sebagai alat identifikasi wajah. Setelah proses validasi, data akan disimpan ke *database*.



Gambar 3.5. *Flowchart* Menu *Sign in/out*

A.3.2 Flowchart Menu Application

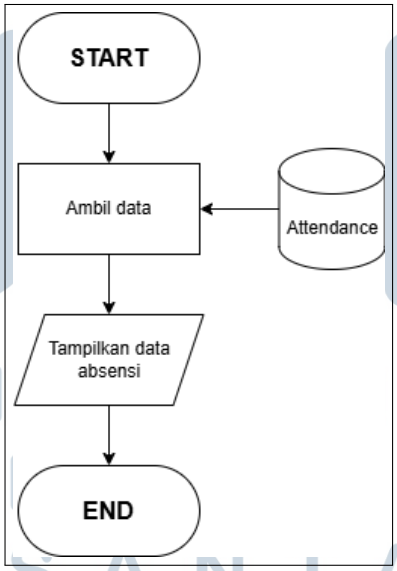
Flowchart pada Gambar 3.6 menggambarkan alur proses dari menu *application*. Pada menu ini, pengguna dapat mengisi formulir berisi pengajuan untuk izin, cuti, serta perbaikan catatan absensi. Pada formulir, tertera pilihan dimana pengguna perlu memilih akun *admin* yang perlu melakukan *approval*. Formulir yang telah dikirim akan disimpan ke *database*.



Gambar 3.6. *Flowchart* Menu *Application*

A.3.3 Flowchart Menu History

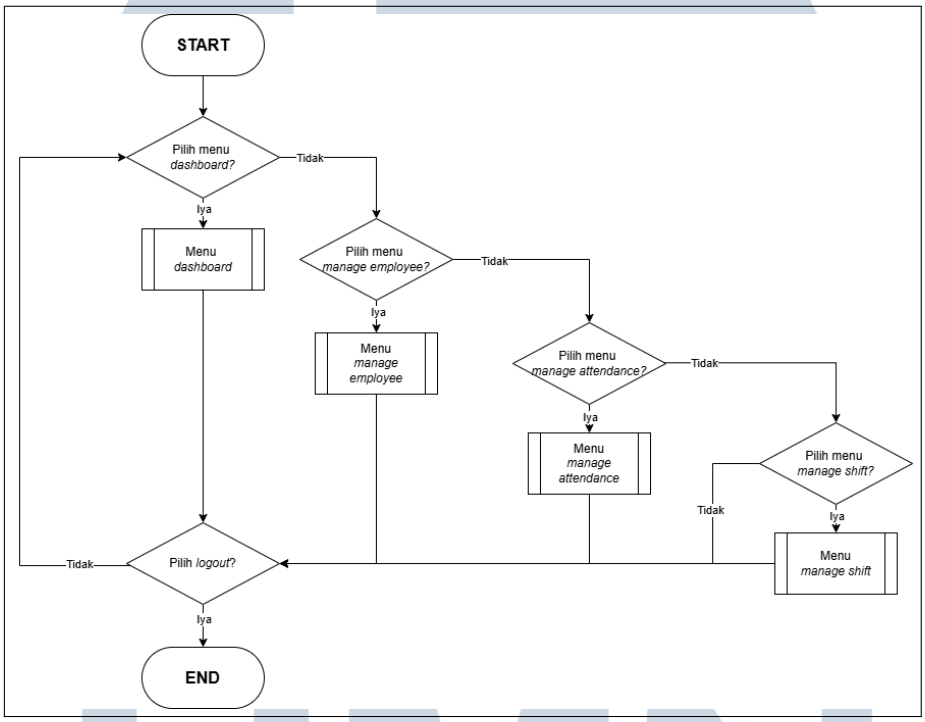
Flowchart pada Gambar 3.7 menggambarkan tampilan menu *history*, dimana pengguna dapat memeriksa catatan kehadiran serta formulir permohonan yang telah dikirim beserta statusnya.



Gambar 3.7. *Flowchart* Menu *History*

A.4 Flowchart Admin

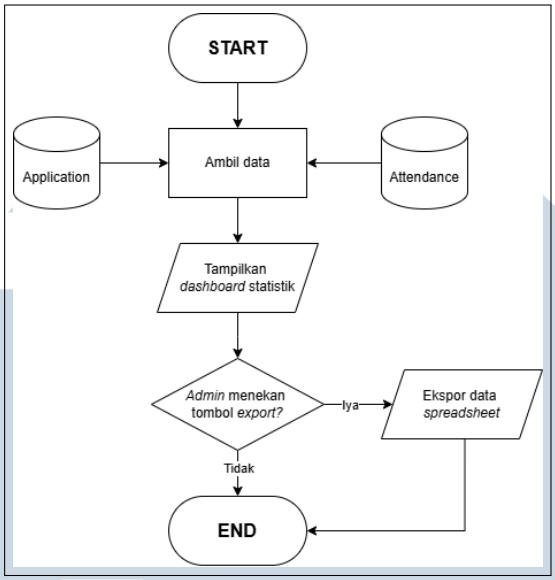
Flowchart pada Gambar 3.8 menggambarkan alur proses pada menu khusus pengguna admin yang berperan sebagai pengelola data sistem. Pengguna memiliki beberapa menu yang dapat diakses mulai dari menu *dashboard*, *manage employee*, *manage attendance*, serta *manage shift*.



Gambar 3.8. *Flowchart Admin*

A.4.1 Flowchart Menu Dashboard

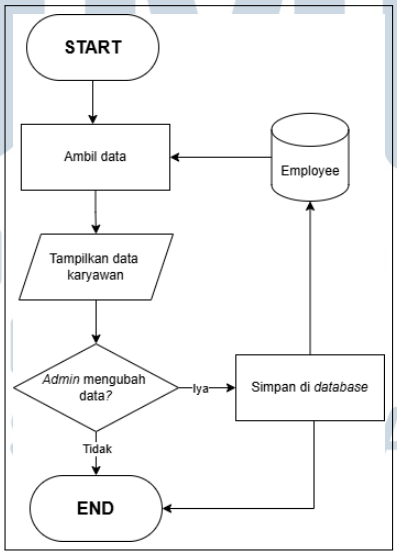
Flowchart pada Gambar 3.9 menggambarkan tampilan menu *dashboard*. Ketika pengguna mengakses *dashboard*, pengguna dapat melakukan ekspor data *spreadsheet* yang berupa data rekapan dari presensi karyawan.



Gambar 3.9. *Flowchart* Menu *Dashboard*

A.4.2 Flowchart Menu Manage Employee

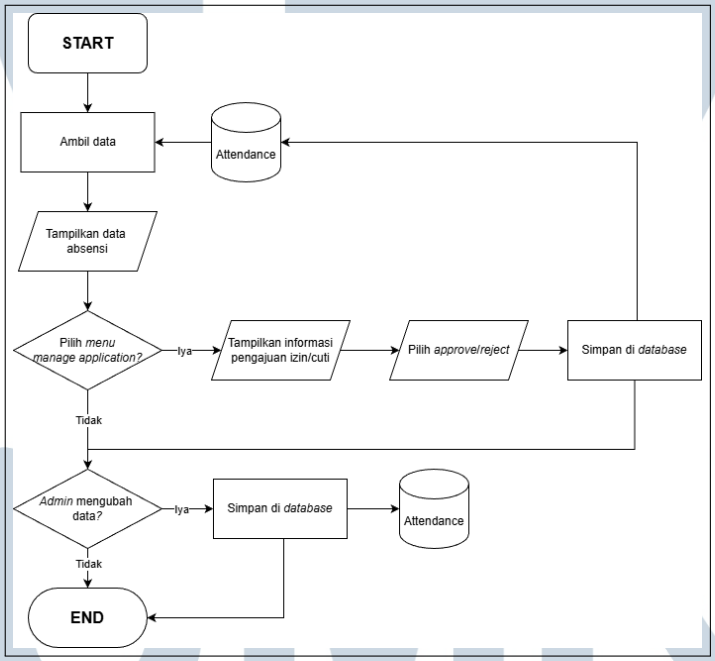
Flowchart pada Gambar 3.10 menunjukkan alur proses pada halaman *manage employee*, dimana *admin* mengelola data karyawan dalam sistem. Halaman menu menampilkan seluruh data karyawan yang tersimpan. Selain itu, sistem juga memberikan opsi kepada *admin* untuk melakukan perubahan data. Jika *admin* melakukan perubahan data, sistem akan memproses dan menyimpan perubahan tersebut ke dalam *database*.



Gambar 3.10. *Flowchart* Menu *Manage Employee*

A.4.3 Flowchart Menu Manage Attendance

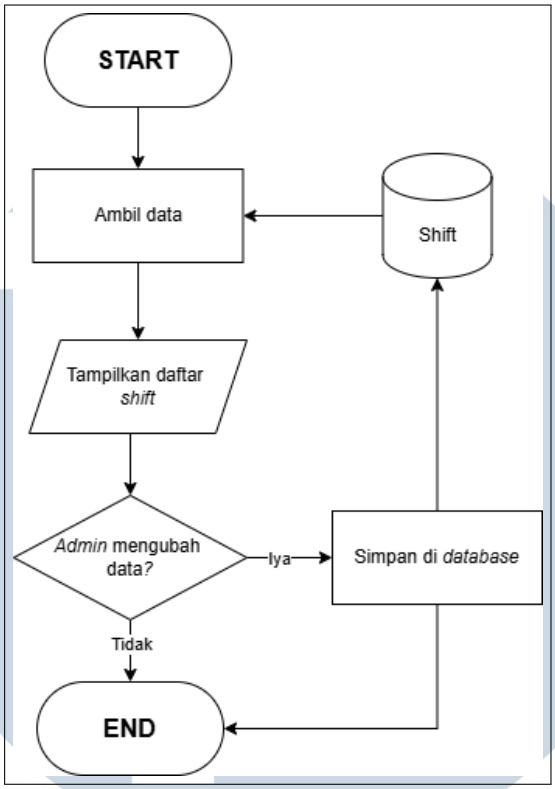
Flowchart pada Gambar 3.11 menggambarkan alur proses pada halaman *manage attendance*, dimana *admin* dapat mengelola data presensi karyawan. Pada halaman tersebut, sistem menampilkan seluruh data presensi yang tersedia. *Admin* dapat membuka menu *manage application* untuk melihat pengajuan izin atau cuti. Jika menu tersebut dipilih, sistem akan menampilkan informasi pengajuan dan *admin* dapat memberi keputusan berupa *approve* atau *reject*. Keputusan tersebut kemudian disimpan ke dalam *database*. Selain itu, admin juga dapat memilih untuk melakukan perubahan data. Sistem akan memproses dan menyimpan perubahan tersebut ke dalam *database*.



Gambar 3.11. *Flowchart* Menu *Manage Attendance*

A.4.4 Flowchart Menu Manage Shift

Flowchart pada Gambar 3.12 menunjukkan alur proses pada halaman *manage shift*, dimana *admin* dapat mengelola data jadwal *shift* karyawan. Pada halaman tersebut, sistem menampilkan daftar *shift* yang tersedia. *Admin* kemudian dapat memilih untuk melakukan perubahan data *shift*, termasuk penyuntingan dan penambahan *shift* baru. Jika ada perubahan yang dilakukan, sistem akan memprosesnya dan menyimpan data yang telah diperbarui ke dalam *database*.



Gambar 3.12. *Flowchart* Menu *Manage Attendance*

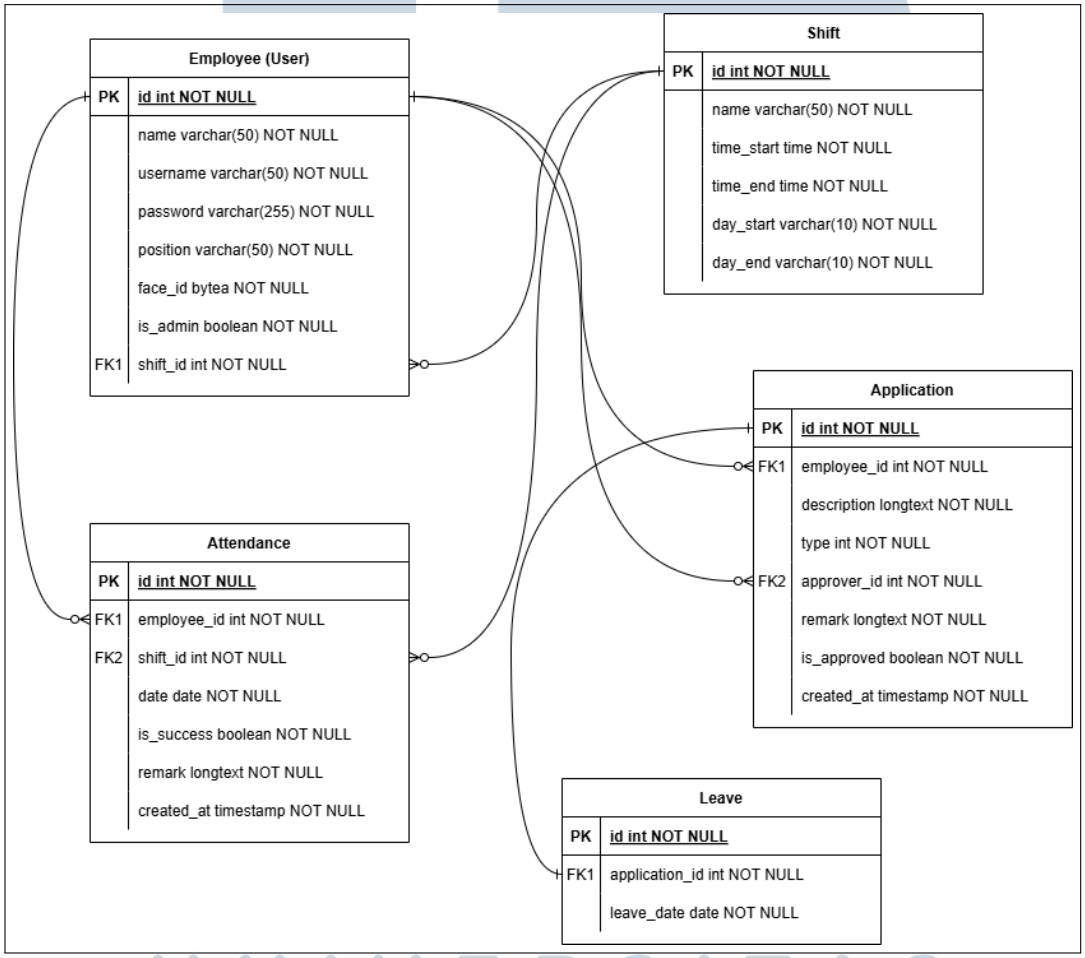
B Entity Relationship Diagram

Gambar 3.13 menunjukkan tabel ERD atau *Entity Relational Diagram* yang menggambarkan struktur basis data, termasuk atribut dan hubungan setiap entitas untuk sistem presensi yang dibangun. Terdapat beberapa tabel yang dirumuskan, antara lain tabel *Employee* (*User*), *Shift*, *Attendance*, *Application*, dan *Leave*.

Tabel *Employee* merupakan tabel yang digunakan untuk menyimpan data akun karyawan/pengguna beserta data kredensial yang digunakan untuk *login* dalam sistem. Tabel *Employee* juga memiliki atribut *is_admin* yang berfungsi untuk menentukan akun pengguna yang berupa *admin*. Akun *admin* memiliki tampilan yang berbeda dengan pengguna biasa. Untuk melakukan *login*, pengguna perlu melakukan *input* kredensial berupa *username* dan *password* yang telah terdaftar.

Fungsi utama sistem terpaku pada tabel *Attendance*, yang berfungsi untuk menyimpan setiap upaya karyawan saat melakukan presensi setiap harinya, termasuk data *invalid*. Tabel *Attendance* memiliki atribut *is_success* sebagai konfirmasi presensi yang *valid*. Data *invalid* hanya digunakan untuk mencatat rekaman pada tampilan *history*.

Kemudian, setiap permohonan yang diajukan oleh pengguna pada menu *Application* akan tersimpan pada tabel *Application*. Tipe permohonan diatur pada atribut *type*. Secara teknis, setiap *user* hanya dapat mengajukan permohonan dengan jenis yang sama sebanyak satu kali, kecuali jika permohonan telah ditolak oleh *approver*. Untuk permohonan tipe *leave*, data termasuk tanggal yang diajukan oleh karyawan untuk cuti akan tersimpan pada tabel *Leave*. Terakhir, atribut *is_approved* berperan untuk mengetahui jika permohonan di *approve* atau tidak.



Gambar 3.13. Entity Relationship Diagram

B.1 Struktur Tabel

Setiap tabel digunakan untuk mendukung setiap fitur dan fungsi yang terdapat di dalam *website*.

1. Struktur Tabel Employee

Tabel 3.3 menunjukkan struktur tabel *employee* untuk menyimpan data karyawan.

Nama tabel : *Employee*
Fungsi : Menyimpan data pengguna untuk autentikasi serta validasi.
Primary key : *id*
Foreign key : *shift_id*

Tabel 3.3. Struktur tabel *Employee*

Nama *Field*	Tipe	*Size*	Deskripsi
id	int		ID tabel *Employee*
name	varchar	50	Nama pengguna
username	varchar	50	*Username* akun pengguna
password	varchar	255	Password akun pengguna
position	varchar	50	Posisi atau jabatan pengguna akun
face_id	bytea		Data *binary* identifikasi wajah
is_admin	boolean		Peran akun (TRUE: *admin*/FALSE: pengguna)
shift_id	int		ID *shift* yang disandingkan

2. Struktur Tabel Attendance

Tabel 3.4 menunjukkan struktur tabel *attendance* untuk menyimpan data presensi karyawan.

Nama tabel : *Attendance*
Fungsi : Menyimpan data upaya presensi karyawan.
Primary key : *id*
Foreign key : *employee_id*, *shift_id*

Tabel 3.4. Struktur tabel *Attendance*

Nama *Field*	Tipe	*Size*	Deskripsi
id	int		ID tabel *Attendance*
employee_id	int		ID *employee* yang disandingkan
Lanjut pada halaman berikutnya			

Tabel 3.4 Struktur tabel *Attendance* (lanjutan)

Nama *Field*	**Tipe**	***Size***	**Deskripsi**
shift_id	int		ID *shift* yang disandingkan
date	date		Tanggal presensi
is_success	boolean		Pernyataan validasi upaya presensi
remark	longtext		Catatan tambahan
created_at	timestamp		Waktu upaya presensi dilakukan

3. Struktur Tabel Shift

Tabel 3.5 menunjukkan struktur tabel *shift* untuk menyimpan data *shift* yang tersedia di perusahaan.

Nama tabel : *Shift*
Fungsi : Menyimpan data *shift* karyawan.
Primary key : *id*
Foreign key : -

Tabel 3.5. Struktur tabel *Shift*

Nama *Field*	**Tipe**	***Size***	**Deskripsi**
id	int		ID tabel *Shift*
name	varchar	50	Nama *shift*
time_start	time		Jam masuk *shift*
time_end	time		Jam selesai *shift*
day_start	varchar	10	Hari pertama *shift*
day_end	varchar	10	Hari terakhir *shift*

4. Struktur Tabel Application

Tabel 3.6 menunjukkan struktur tabel *application* untuk menyimpan data permohonan karyawan.

Nama tabel : *Application*
Fungsi : Menyimpan data permohonan karyawan untuk izin/cuti/perbaikan data.
Primary key : *id*
Foreign key : *employee_id*, *approver_id*

Tabel 3.6. Struktur tabel *Application*

Nama *Field*	Tipe	*Size*	Deskripsi
id	int		ID tabel *Application*
employee_id	int		ID *employee* yang disandingkan
description	longtext		Deskripsi detail permohonan
type	int		Tipe permohonan
approver_id	int		ID *employee* sebagai *approver* yang disandingkan
remark	longtext		Catatan tambahan
is_approved	boolean		Pernyataan permohonan *approved* atau *rejected*
created_at	timestamp		Waktu permohonan dikirimkan

5. Struktur Tabel Leave

Tabel 3.7 menunjukkan struktur tabel *leave* untuk menyimpan data perizinan cuti.

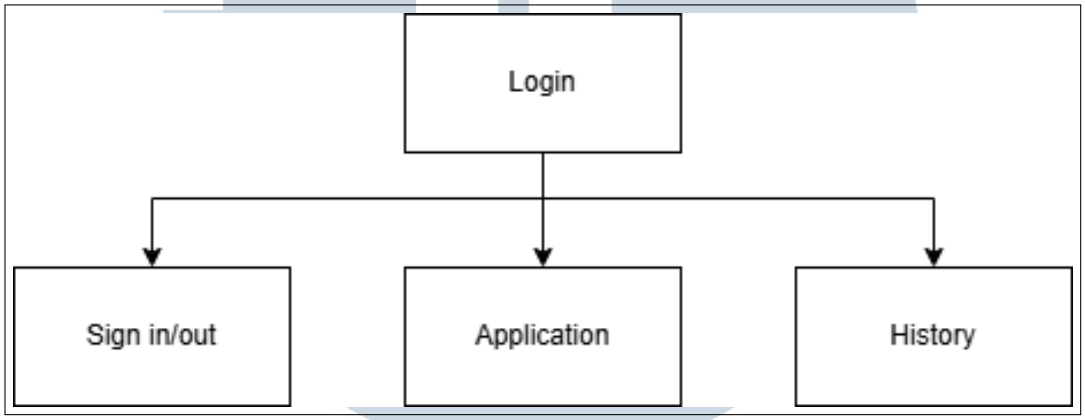
Nama tabel : *Leave*
Fungsi : Menyimpan data izin/cuti.
Primary key : *id*
Foreign key : *application_id*

Tabel 3.7. Struktur tabel *Leave*

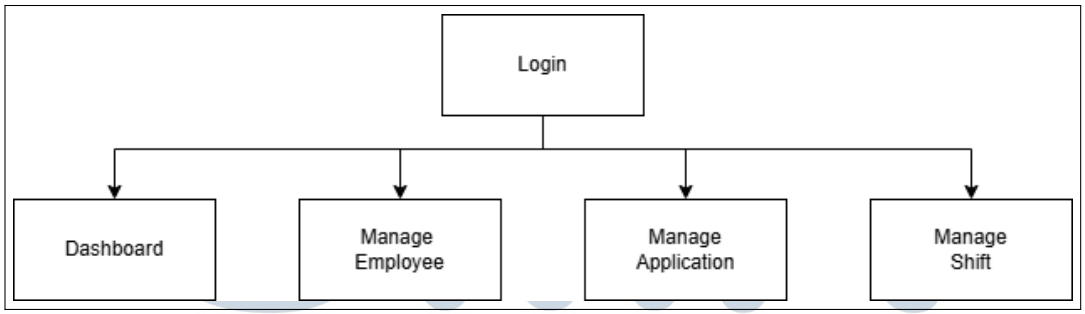
Nama *Field*	Tipe	*Size*	Deskripsi
id	int		ID tabel *Leave*
application_id	int		ID permohonan yang disandingkan
leave_date	date		Tanggal izin/cuti

C Sitemap

Sitemap merupakan representasi visual dari struktur *website Smart Attendance System*. Gambar 3.14 dan Gambar 3.15 masing-masing menunjukkan struktur *website* yang ditampilkan kepada pengguna biasa dan *admin*. Tampilan umum *website* terbagi menjadi tiga bagian dasar, yakni *sign in/out*, *application* dan *history*. Sedangkan, tampilan *admin* terbagi menjadi empat bagian, yakni *dashboard*, *manage employee*, *manage application* , dan *manage shift*.



Gambar 3.14. *Sitemap User*



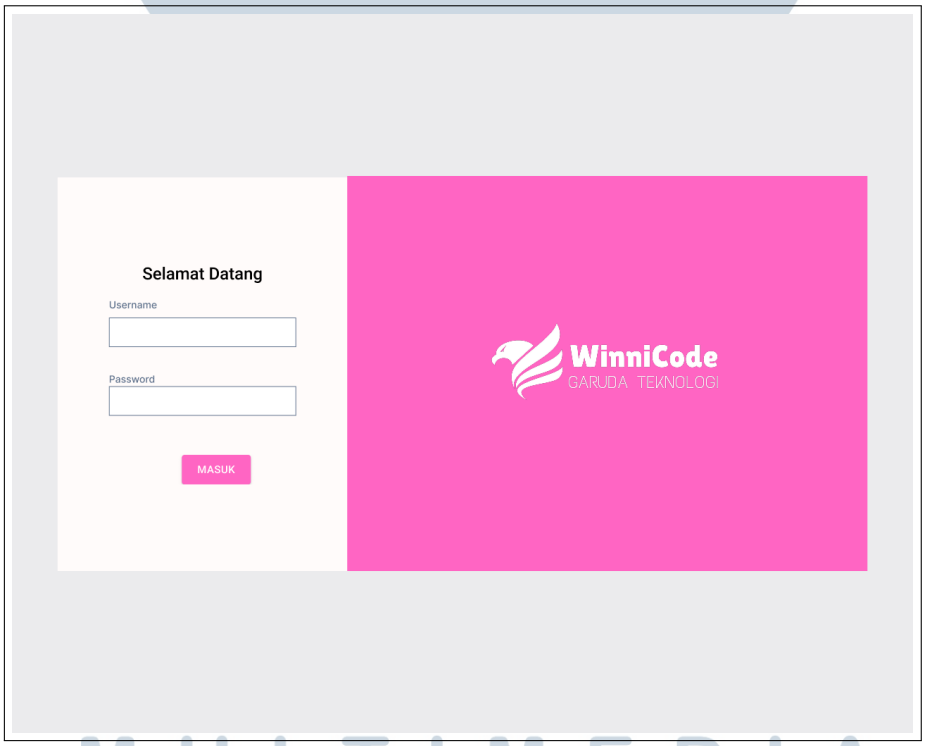
Gambar 3.15. *Sitemap Admin*

D Wireframe

Wireframe adalah representasi visual awal dari tampilan antarmuka *website Smart Attendance System*. Perancangan *wireframe* berfungsi sebagai kerangka dasar yang menunjukkan susunan elemen, penempatan komponen, serta alur interaksi pengguna sebelum tahap pembangunan.

D.1 Wireframe Login

Halaman *login* yang ditunjukkan pada Gambar 3.16 menampilkan antarmuka sederhana dengan fitur utama yakni form autentikasi pengguna berupa *input username* dan *password* disertai logo serta identitas perusahaan Winnicode Garuda Teknologi dengan warna pink, yang merupakan tema warna perusahaan sebagai elemen utama.



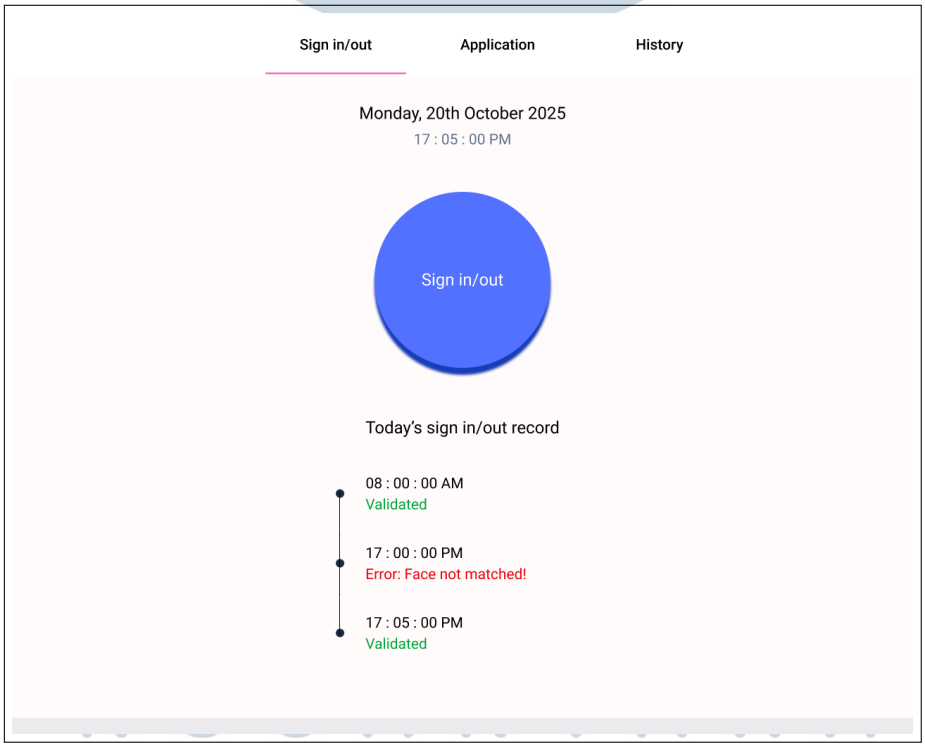
Gambar 3.16. *Wireframe Login*

D.2 Wireframe User

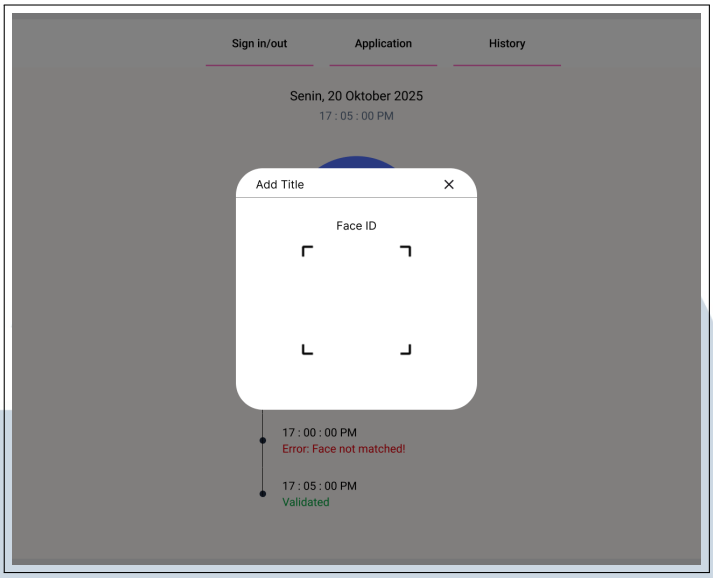
Wireframe user menggambarkan rancangan awal antarmuka yang digunakan oleh pengguna akhir dalam berinteraksi dengan sistem. *Wireframe* user terbagi menjadi tiga bagian, yakni *wireframe sign in/out*, *wireframe application*, *wireframe history*.

D.2.1 Wireframe Sign In/Out

Gambar 3.17 menunjukkan halaman *sign in/out* yang berfungsi sebagai tempat utama bagi karyawan untuk melakukan presensi dengan identifikasi wajah. Tampilan halaman menampilkan tombol yang digunakan saat pengguna ingin melakukan presensi, serta daftar riwayat presensi hari itu dalam bentuk garis waktu yang menunjukkan jam masuk, jam keluar, serta status validasi setiap upaya presensi. Tampilan pada Gambar 3.18 akan tampil ketika tombol *sign in/out* ditekan. *Modal* ini berfungsi untuk menampilkan dan mengaktifkan *webcam* untuk proses presensi.



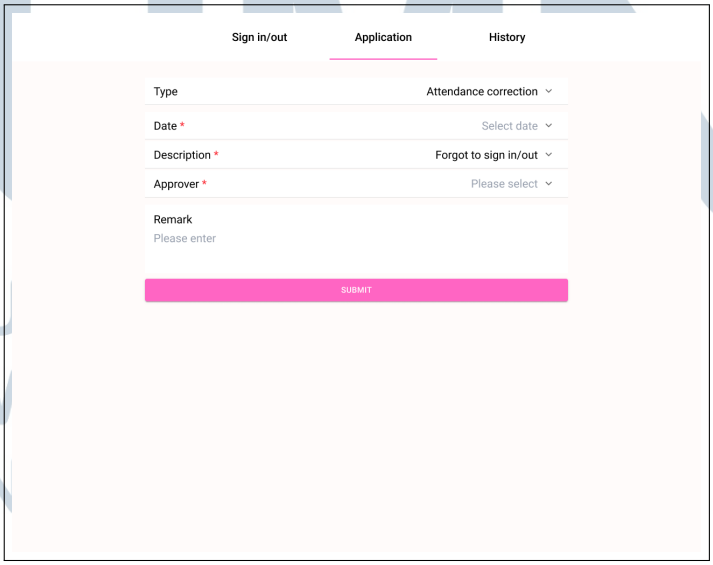
Gambar 3.17. *Wireframe Sign in/out (1)*



Gambar 3.18. *Wireframe Sign in/out (2)*

D.2.2 Wireframe Application

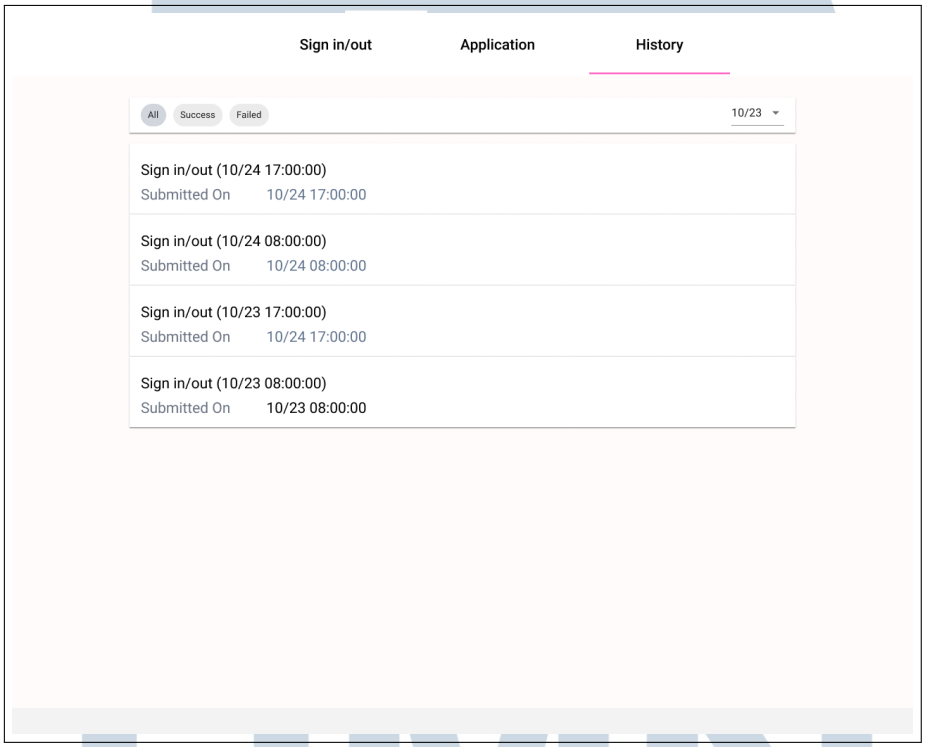
Pada Gambar 3.19, ditunjukkan halaman *application* yang digunakan untuk pengajuan koreksi absensi atau permohonan terkait perizinan. Halaman ini terdiri atas formulir yang terdiri dari pilihan tipe pengajuan, waktu kejadian, deskripsi alasan, pilihan *approver*, serta kolom *remark* untuk menambah keterangan tambahan.



Gambar 3.19. *Wireframe Application*

D.2.3 Wireframe History

Gambar 3.20 menampilkan halaman *history* yang berisi riwayat lengkap upaya presensi pengguna dalam bentuk daftar kronologis. Pengguna dapat melihat setiap catatan *sign in/out* beserta waktu pengiriman datanya, serta melakukan *filter* data. Selain itu, terdapat *dropdown* untuk memilih tanggal tertentu agar pencarian data lebih spesifik.



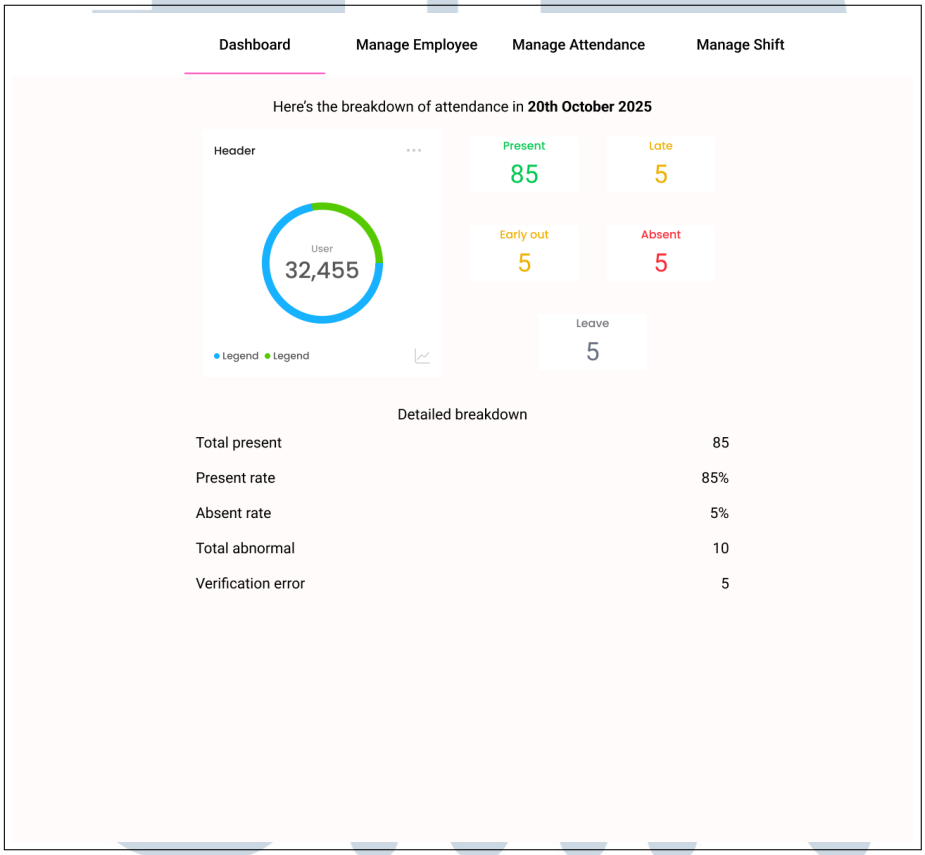
Gambar 3.20. *Wireframe History*

D.3 Wireframe Admin

Wireframe admin menggambarkan rancangan awal antarmuka yang digunakan oleh akun yang termasuk dalam kategori administrator. *Wireframe admin* berfokus pada penyajian informasi dan fungsi yang mendukung proses manajemen data secara efisien. *Wireframe admin* terbagi menjadi tiga bagian, yakni *wireframe dashboard*, *wireframe manage employee*, *wireframe manage attendance*, dan *wireframe manage shift*.

D.3.1 Wireframe Dashboard

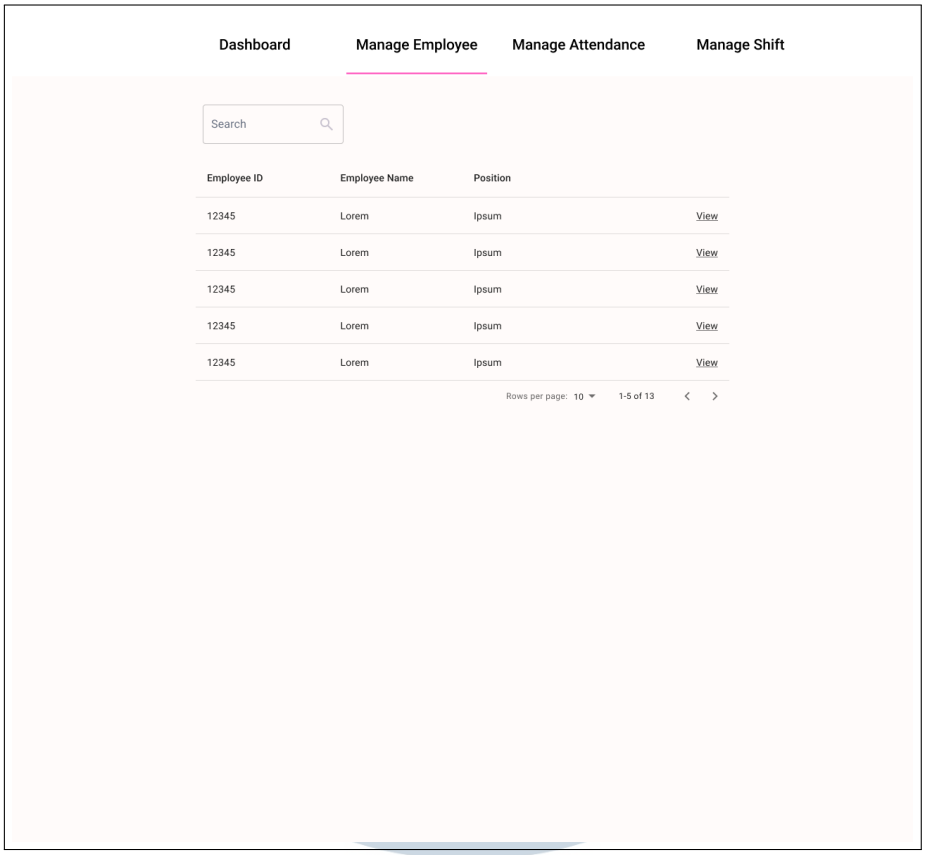
Gambar 3.21 menunjukkan halaman *dashboard* menampilkan ringkasan kehadiran karyawan dalam bentuk *visual* yang sederhana dan informatif. Tampilan utamanya berfokus pada grafik lingkaran dan beberapa kartu informasi yang menyoroti data karyawan yang hadir, terlambat, pulang lebih awal, absen, dan izin cuti pada hari tersebut. Di bagian bawah terdapat tabel ringkas yang menampilkan rekapan statistik dari data kehadiran serta tombol ekspor *spreadsheet*.



Gambar 3.21. *Wireframe Dashboard*

D.3.2 Wireframe Manage Employee

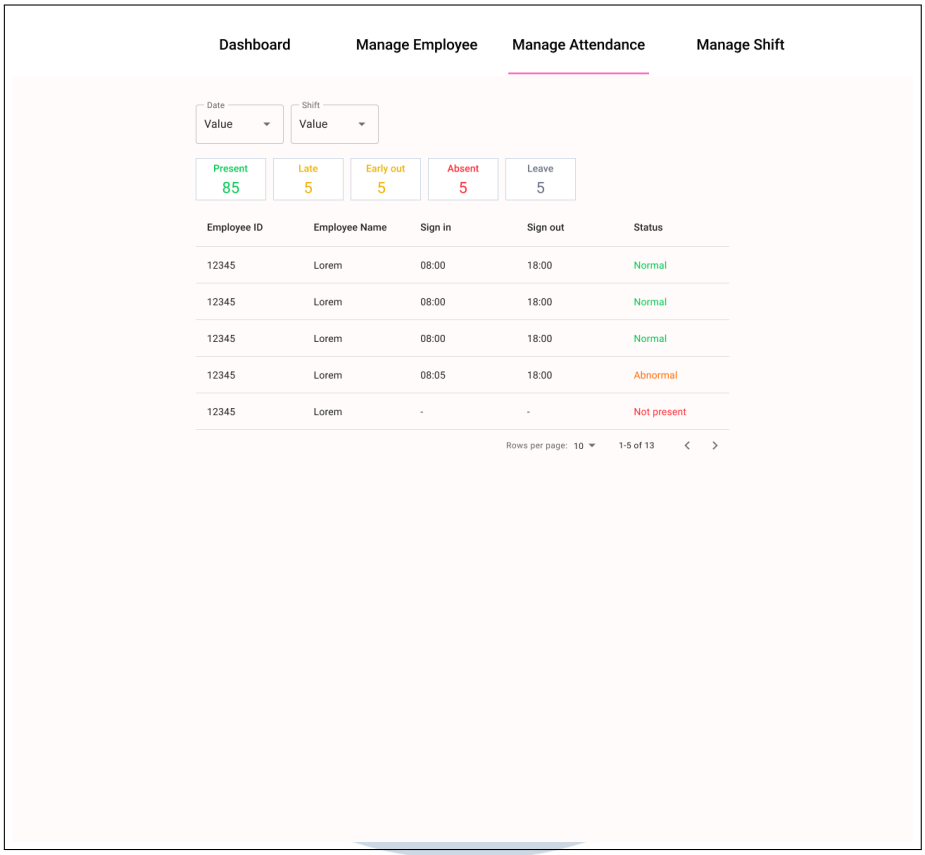
Pada Gambar 3.22, ditunjukkan halaman *manage employee* yang menampilkan tabel sederhana berisi daftar karyawan. Di bagian atas terdapat kolom pencarian yang memudahkan pengguna dalam menemukan data karyawan tertentu dengan melakukan *input* nama.



Gambar 3.22. *Wireframe Manage Employee*

D.3.3 Wireframe Manage Attendance

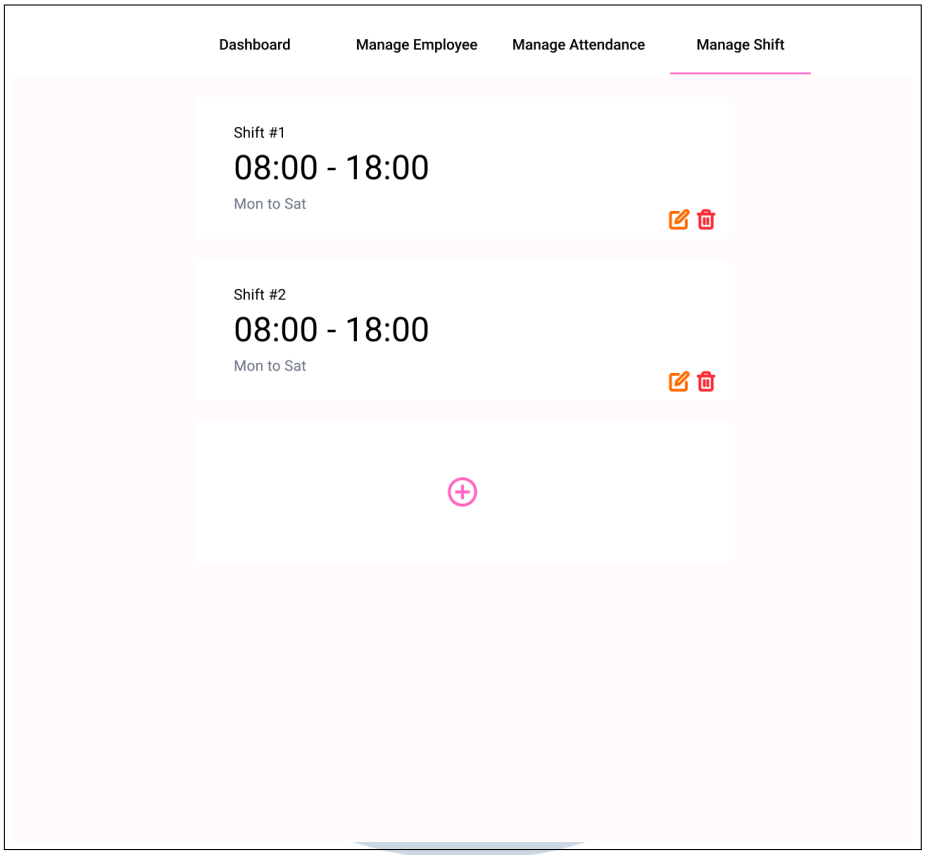
Halaman *manage attendance* pada Gambar 3.23 menampilkan data kehadiran karyawan dalam format tabel yang rapi dan mudah dipahami. Di bagian atas terdapat pilihan *dropdown* untuk menyaring data berdasarkan tanggal dan *shift*, diikuti dengan *card* yang berisi informasi jumlah karyawan dengan status hadir, terlambat, pulang lebih awal, absen, dan izin cuti pada tanggal yang terpilih.



Gambar 3.23. *Wireframe Manage Attendance*

D.3.4 Wireframe Manage Shift

Halaman *manage shift* pada Gambar 3.24 berfungsi menampilkan daftar jadwal kerja atau *shift* yang ada. Setiap *card* berisi informasi mengenai masing-masing shift yang telah dibuat, termasuk jam kerja mulai hingga selesai, serta hari kerja yang berlaku. Di sisi kanan setiap kartu terdapat pilihan untuk menghapus atau mengedit *shift* yang sudah ada. Pada bagian bawah, terdapat pilihan untuk menambahkan *shift* baru.



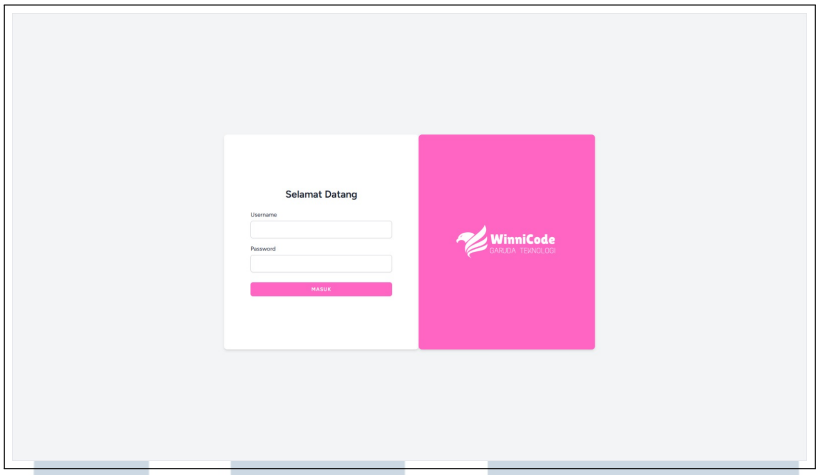
Gambar 3.24. *Wireframe Manage Shift*

E Implementasi

Hasil akhir dari pembangunan aplikasi *Smart Attendance System* berbasis *website* dapat dilihat pada Gambar 3.25 sampai 3.37

E.1 Login

Gambar 3.25 menunjukkan tampilan saat seseorang mengakses *website* ketika belum melakukan *login*. Pada halaman ini, pengguna diarahkan untuk melakukan *login* akun yang telah didaftarkan dalam sistem oleh pihak *admin*.



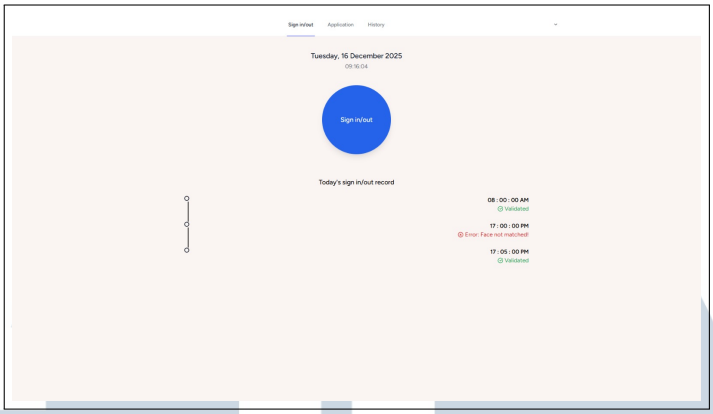
Gambar 3.25. Tampilan *login*

E.2 Halaman User

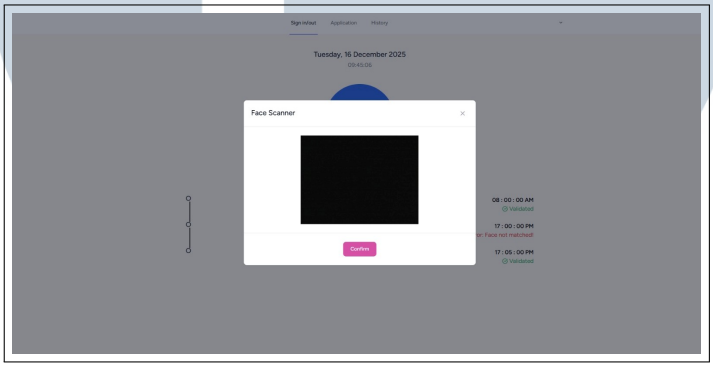
Setelah proses *login*, akun *default* akan diarahkan ke menu *user*, dengan halaman utama yakni menu *sign in/out* yang terlihat pada Gambar 3.26.

E.2.1 Sign in/out

Setelah akun pengguna berhasil masuk dan disimpan dalam *session website*, pengguna akan diarahkan ke menu *user sign in/out* yang terlihat pada Gambar 3.26. Ketika pengguna berinteraksi dengan tombol *sign in/out*, sebuah *modal* akan muncul yang mengaktifkan dan menampilkan *webcam*, dimana pengguna dapat melakukan pengambilan foto untuk presensi, seperti terlihat pada Gambar 3.27. *Library* TensorFlow.js digunakan untuk melakukan pemrosesan foto, termasuk pemanfaatan *deep learning model FaceNet* sebagai alat pendeteksi dan ekstraksi data wajah, serta algoritma sebagai alat identifikasi wajah yang bekerja langsung pada *client-side*.



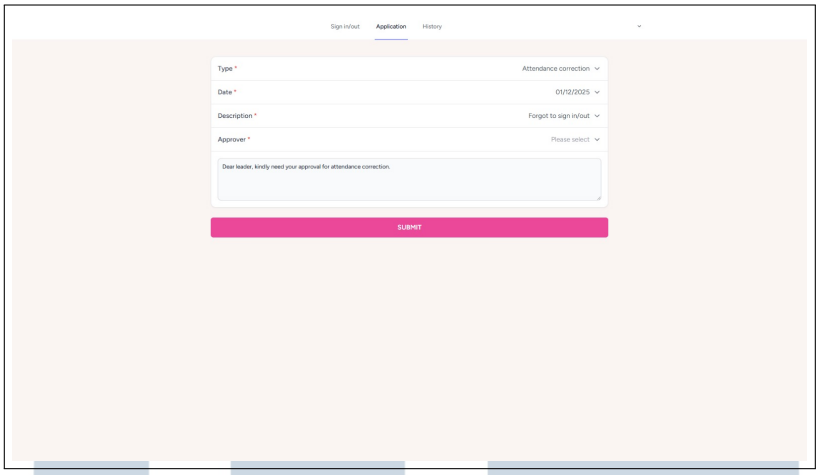
Gambar 3.26. Tampilan menu *sign in/out* (1)



Gambar 3.27. Tampilan menu *sign in/out* (2)

E.2.2 Application

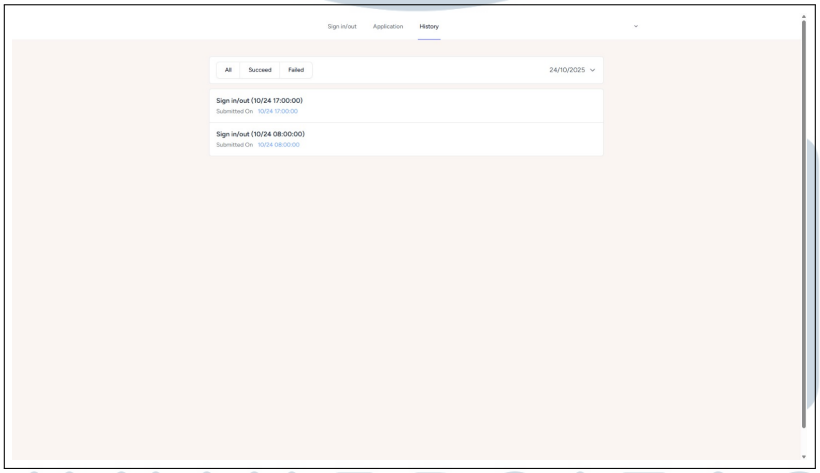
Gambar 3.28 menunjukkan menu *application* yang telah diimplementasikan, berisi formulir dengan komponen berupa *Select*, *TextArea*, serta *Button* untuk melakukan *submit*.



Gambar 3.28. Tampilan menu *application*

E.2.3 History

Gambar 3.29 menunjukkan menu *history*, yang berhasil menampilkan seluruh riwayat presensi karyawan berdasarkan tanggal yang dipilih. Data yang ditampilkan disesuaikan dengan *filter* pada sisi *client*.



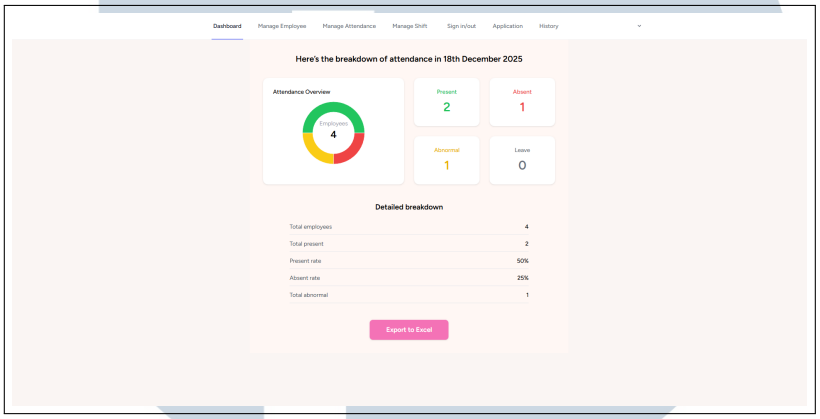
Gambar 3.29. Tampilan menu *history*

E.3 Halaman Admin

Setelah proses *login*, akun dengan kategori *admin* akan diarahkan ke halaman utama, yakni menu *dashboard*.

E.3.1 Dashboard

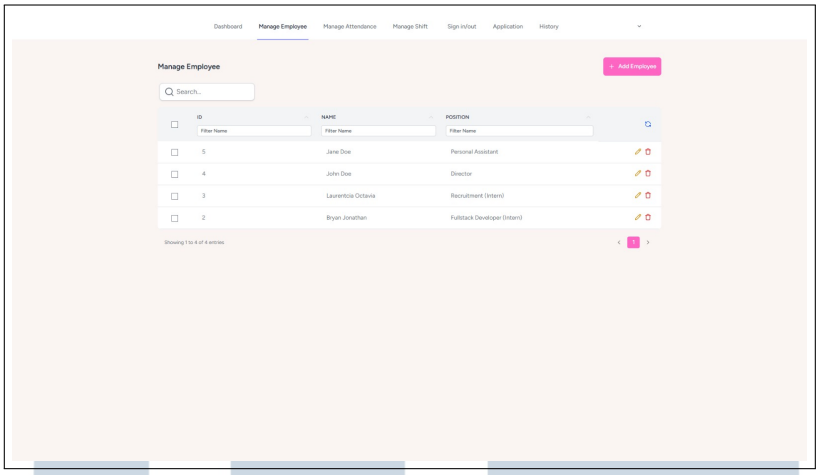
Halaman *dashboard* yang ditampilkan pada Gambar 3.30 merupakan tampilan utama bagi *admin* yang berisi rekapan data kehadiran karyawan secara keseluruhan. Visualisasi data dengan komponen *pie chart* dibangun menggunakan *library* Recharts. Halaman ini juga berhasil menyediakan fitur ekspor data dalam bentuk *spreadsheet* yang diproses pada *server-side*.



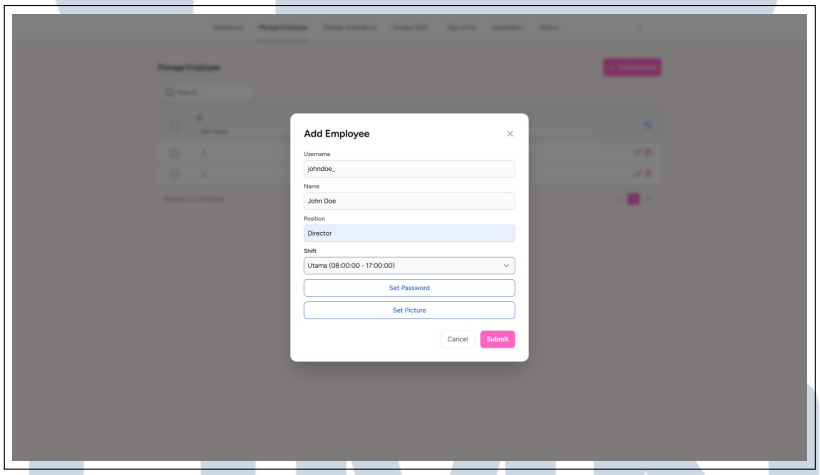
Gambar 3.30. Tampilan menu *dashboard*

E.3.2 Manage Employee

Halaman *manage employee* yang ditampilkan pada Gambar 3.31 digunakan untuk mengelola data karyawan pada sistem. Di bagian atas tersedia komponen *search bar* dan *filter* yang memudahkan *admin* mencari karyawan tertentu secara cepat. Ketika *admin* menekan tombol *edit*, akan tampil *modal* yang berisi data karyawan untuk dikelola, seperti terlihat pada Gambar 3.32.



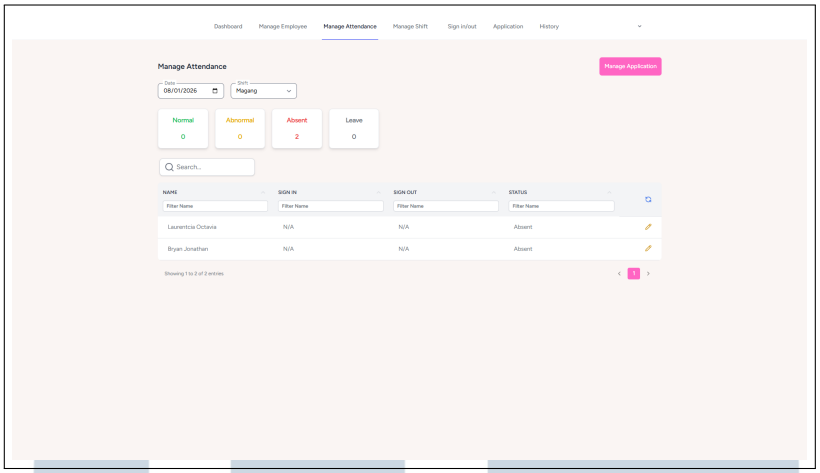
Gambar 3.31. Tampilan menu *employee* (1)



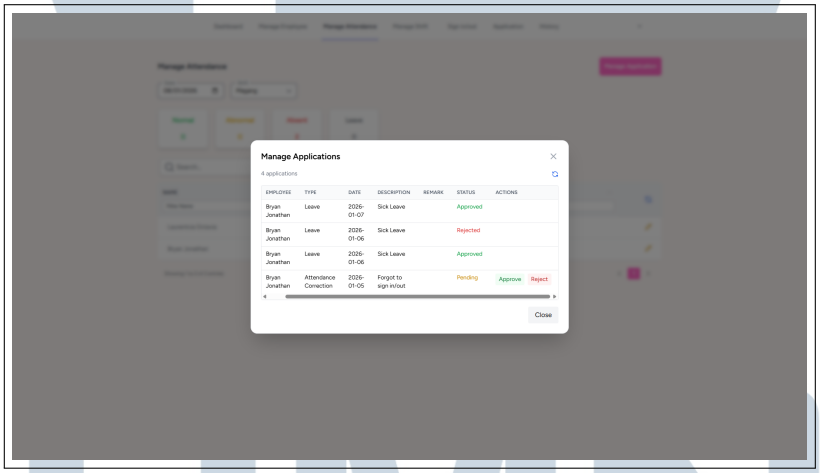
Gambar 3.32. Tampilan menu *employee* (2)

E.3.3 Manage Attendance

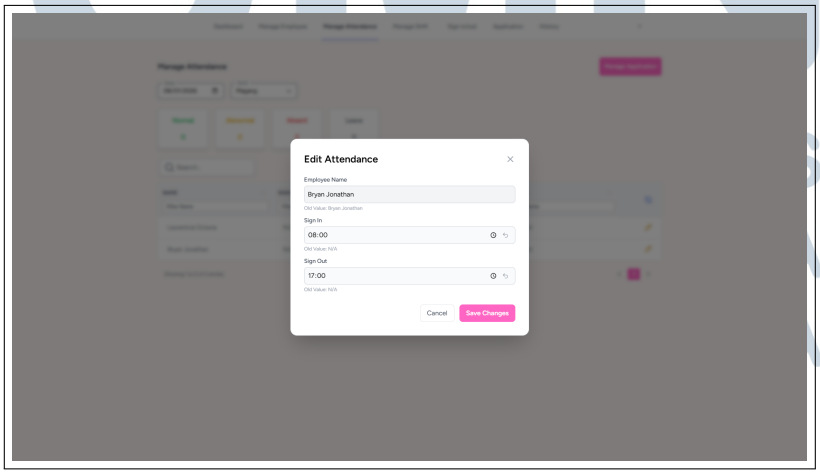
Halaman *manage attendance* yang ditampilkan pada Gambar 3.33 digunakan untuk mengelola data presensi karyawan. *Admin* dapat melakukan *filter* berdasarkan *shift* maupun tanggal untuk menampilkan catatan presensi. Jika admin menekan tombol *manage application*, *admin* dapat melihat daftar pengajuan karyawan dan memilih untuk melakukan *approve/reject*, seperti terlihat pada Gambar 3.34. Jika *admin* menekan tombol *edit*, akan tampil *modal* yang berguna hika *admin* ingin mengubah data presensi secara manual, seperti terlihat pada Gambar 3.35.



Gambar 3.33. Tampilan menu *attendance* (1)



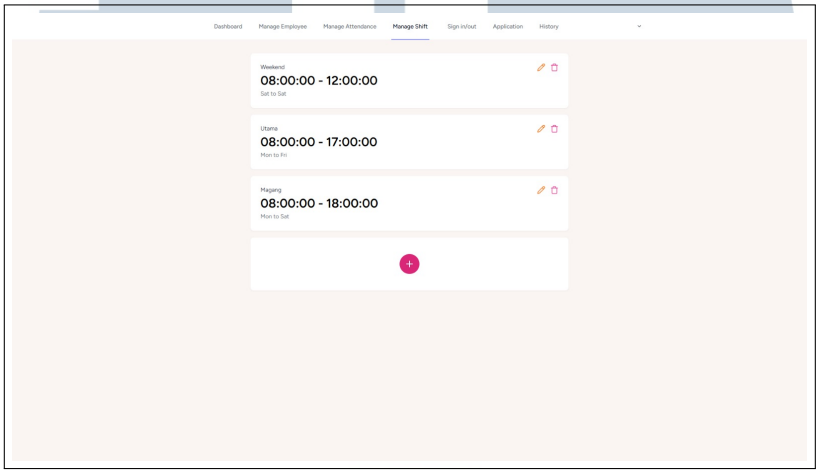
Gambar 3.34. Tampilan menu *attendance* (2)



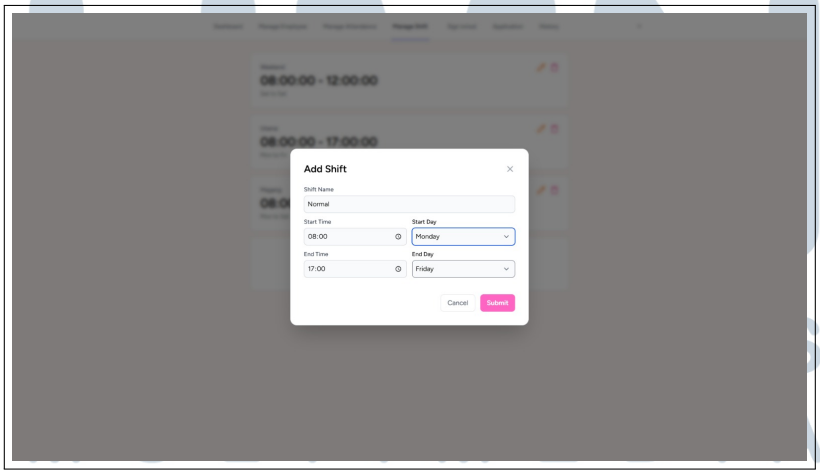
Gambar 3.35. Tampilan menu *attendance* (3)

E.3.4 Manage Shift

Gambar 3.36 menampilkan halaman *manage shift* yang digunakan untuk mengelola *shift* atau jadwal kerja karyawan. *Shift* yang tersedia akan ditampilkan dalam bentuk *card*, masing-masing dilengkapi dengan tombol *edit* serta *delete*. Pada *card* terakhir, terdapat tombol yang berfungsi untuk menambahkan *shift* baru. Gambar 3.37 memperlihatkan tampilan *modal* ketika pengguna melakukan *add* atau *edit shift*.



Gambar 3.36. Tampilan menu *shift* (1)



Gambar 3.37. Tampilan menu *shift* (2)

3.4 Pengujian

Pengujian *website Smart Attendance System* dilakukan menggunakan metode *black box* untuk memastikan bahwa setiap fitur berjalan sesuai dengan fungsionalitas yang telah dirancang. Tabel 3.8 menyajikan hasil pengujian terhadap beberapa fitur utama menggunakan pendekatan *black box*.

Tabel 3.8. Hasil Pengujian *Black Box*

No	**Halaman**	**Skenario**	**Ekspektasi**	**Hasil**
1	Halaman *Login*	*Input* kredensial yang benar	Pengguna berhasil masuk ke halaman utama sesuai dengan kategori akun (admin/user)	Berhasil
2	Halaman *Login*	*Input* kredensial yang salah	Tampilan pesan *error*	Berhasil
3	Halaman *Sign in/out*	*Presensi* dengan autentikasi wajah yang sesuai	Data tersimpan dengan status *valid* dan muncul di daftar presensi	Berhasil
4	Halaman *Sign in/out*	*Presensi* dengan autentikasi wajah yang tidak sesuai	Data tersimpan dengan status *invalid*	Berhasil
5	Halaman *Sign in/out*	Foto yang tidak jelas atau tidak mengandung wajah saat *presensi*	Tampilkan pesan *error*	Berhasil
6	Halaman *Application*	*Input* data pada formulir lengkap	Data tersimpan pada tabel *Application*	Berhasil
Lanjut pada halaman berikutnya				

Tabel 3.8: *Hasil Pengujian Black Box* (lanjutan)

No	Halaman	Skenario	Ekspektasi	Hasil
7	Halaman *Application*	*Input* data pada formulir tidak lengkap	Tampilkan pesan *error*	Berhasil
8	Halaman *History*	Melakukan *filter* data	Data presensi ditampilkan berdasarkan *filter*	Berhasil
9	Halaman *Dashbord*	Melakukan ekspor data dalam bentuk *spreadsheet*	Data berhasil dikonversi dan diunduh dalam bentuk *spreadsheet*	Berhasil
10	Halaman *Manage Employee*	Melakukan *filter* pada tabel data	Data ditampilkan berdasarkan filter	Berhasil
11	Halaman *Manage Employee*	*Input* data pada formulir *add/edit employee* lengkap	Data karyawan tersimpan dalam sistem	Berhasil
12	Halaman *Manage Employee*	*Input* data pada formulir *add/edit employee* tidak lengkap	Tampilkan pesan *error*	Berhasil
13	Halaman *Manage Attendance*	Melakukan *filter* pada tabel data	Data ditampilkan berdasarkan filter	Berhasil
14	Halaman *Manage Attendance*	*Input* data pada formulir *edit attendance* lengkap	Data presensi tersimpan dalam sistem	Berhasil
15	Halaman *Manage Attendance*	*Input* data pada formulir *edit attendance* tidak lengkap	Tampilkan pesan *error*	Berhasil

Lanjut pada halaman berikutnya

Tabel 3.8: *Hasil Pengujian Black Box* (lanjutan)

No	Halaman	Skenario	Ekspektasi	Hasil
16	Halaman *Manage Attendance*	Melakukan *approve/reject* pengajuan karyawan	Data berhasil tersimpan dalam sistem	Berhasil
17	Halaman *Manage Shift*	*Input* data pada formulir *add/edit shift* lengkap	Data *shift* tersimpan dalam sistem	Berhasil
18	Halaman *Manage Shift*	*Input* data pada formulir *add/edit shift* tidak lengkap	Tampilkan pesan *error*	Berhasil

3.5 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan magang dan pengembangan sistem di PT. Winnicode Garuda Teknologi, terdapat beberapa kendala teknis yang memengaruhi proses pengembangan sistem. Berikut adalah kendala yang ditemukan:

1) Tidak tersedianya *Application Programming Interface* (API) pada sistem utama perusahaan menyebabkan data karyawan dan presensi tidak dapat diintegrasikan secara langsung.
2) Tantangan dalam melakukan sinkronisasi data antara sisi *frontend* dan *backend*, khususnya terkait pembaruan data secara *real-time*.
3) Model *face identification* tidak selalu mampu mengidentifikasi wajah secara akurat, terutama pada kondisi tertentu seperti pencahayaan yang kurang optimal atau adanya wajah dengan karakteristik yang identik.

Sebagai upaya untuk mengatasi kendala-kendala tersebut, berikut adalah solusi yang diterapkan:

1) Mengembangkan mekanisme integrasi data secara tidak langsung namun berkala, seperti melalui proses impor dan ekspor data dalam bentuk *spreadsheet*.

2) Menerapkan mekanisme sinkronisasi data yang lebih terkontrol dengan menambahkan validasi respons server, serta penggunaan indikator seperti (*loading state*) pada sisi *frontend* sebelum ada respon dari *backend*.

3) Menambahkan fungsi validasi data untuk memastikan resolusi foto yang terdaftar dalam sistem telah memenuhi standar minimum yang telah ditetapkan, demi meningkatkan akurasi pengenalan wajah oleh model *face identification*.