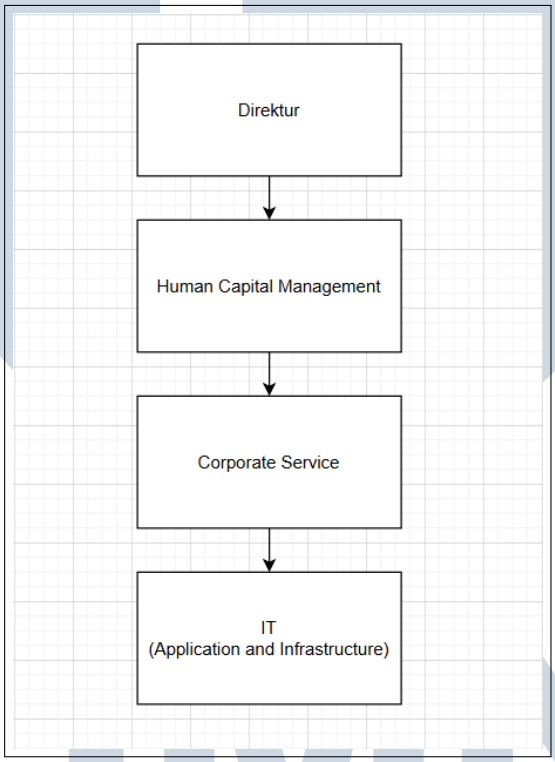
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Dibawah ini akan diberikan gambar dan juga penjelasan tentang posisi selama menjalankan praktik magang di PT Karya Solusi Prima Sejahtera.



Gambar 3.1. Struktur kedudukan untuk pelaksanaan kerja magang pada PT Karya Solusi Prima Sejahtera

Gambar 3.1 merupakan struktur pelaksanaan praktik kerja magang pada PT Karya Solusi Prima Sejahtera sebagai *staff IT*, yang berada di bawah pengawasan *Corporate Service*. Dalam menjalankan praktik kerja magang di PT Karya Solusi Prima Sejahtera, terkhususnya untuk pelaksanaan pengerjaan proyek HRIS akan disupervisi oleh Bapak Ulil Albab.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama melaksanakan praktik kerja magang di PT Karya Solusi Prima Sejahtera dapat di rincikan sebagai berikut:

1. Pengenalan HRIS

   Pengenalan HRIS bermaksud agar pada saat proyek telah diberikan, pengerjaan ataupun pengembangannya tidak mengalami masalah karena kurangnya pemahaman tentang apa itu HRIS.

2. Melakukan pengembangan fitur HRIS sesuai *task*

   Dalam praktik magang kali ini, untuk pelaksanaan pengerjaan proyek HRIS terkhususnya pada pengembangannya akan dilakukan sesuai dengan task yang diberikan oleh supervisi yang selanjutnya akan diserahkan atau *push* lagi melalui *branch* masing-masing di git, dan akan dilakukan pengecekan oleh supervisi. *Task* ini akan diberikan secara berkala oleh supervisi.

3. Memperbaiki *error* pada *codingan*

   Tugas lain yang dilakukan adalah revisi pada *codingan*, dimana codingan akan dikembalikan oleh supervisi jika terjadi *error* atau ditemukan *bug* pada *codingan*. Maka dari itu tugas selanjutnya yang dilakukan adalah memperbaiki *codingan* jika terjadi kesalahan dalam pengerjaan yang telah di*push* ke *branch* pada git.

4. *Testing* aplikasi

   Setelah pengerjaan akan dilakukan testing untuk mencoba serta mencari *error*, namun untuk *testing* sendiri jarang dilakukan, melainkan supervisi sendiri yang akan langsung melakukan *testing*.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang yang dimulai pada bulan September 2024 dapat diuraikan seperti pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mempelajari apa itu HRIS, melakukan *installasi* untuk Laravel-docker, mempelajari alur database dan *flowchart*
2	Menambahkan fitur pada halaman *Home*
3	Pengintegrasian *front end* dan *back end* untuk fitur *check in* dan *check out*
4	Menambahkan fitur absensi (absen, koreksi, dan *mapping* absensi
5	Menambahkan fitur shift (shift dan tukar shift)
6	Menambahkan fitur Izin
7	Menambahkan fitur Lembur
8	Menambahkan fitur Karyawan
9	Mengembangkan fitur absensi khususnya bagian koreksi
10	Melanjutkan pengembangan fitur pada page absensi
11	Mengembangkan fitur tukar shift untuk menu shift

3.3.1 Tools, Proyek, Flowchart dan Implementasi

Pelaksanaan magang di PT Karya Solusi Prima Sejahtera diawali dengan pelatihan dan pengenalan lingkungan kerja sebagai *IT Developer*. Praktik kerja magang berlanjut dengan diberikannya proyek oleh supervisi untuk melanjutkan proyek rancang bangung aplikasi HRIS, untuk kepentingan absensi, pengurusan izin, pembayaran gaji, dan urusan administrasi lainnya. Dalam pengerjaan proyek ini, digunakan beberapa *tools* untuk membantu pengerjaan. Berikut *tools* dan penjelasan tentang proyek yang digunakan dan dilaksanakan:

A. *Tools*

- Ubuntu

  Ubuntu atau Linux Ubuntu adalah *Operating System* yang berasal dari keluarga Linux. Ubuntu diambil dari bahasa Zulu dan Xhosa dari Afrika

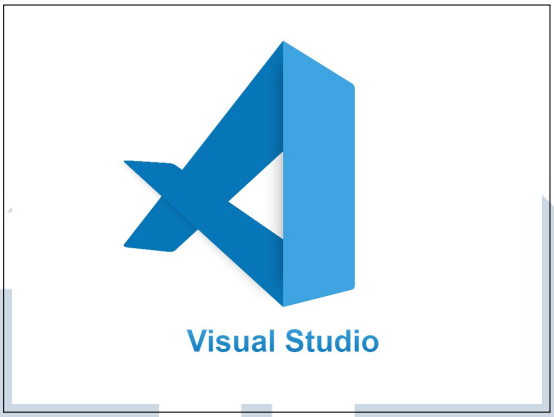
Selatan yang memiliki arti ”kemanusiaan terhadap sesama”. Ubuntu sendiri salah satu sistem operasi Linux yang populer di dunia. Ubuntu bertujuan untuk membantu pengguna yang belum terbiasa menggunakan sistem operasi Linux, karena Ubuntu lebih ramah pengguna. Ubuntu menjadi sistem operasi Linux yang memberikan pengalaman pengguna yang sederhana dan mudah dioperasikan [3].

Gambar 3.2. Logo Ubuntu
Sumber: [4]

- Visual Studio Code

  Visual Studio Code adalah sebuah *code editor* yang dimana di dalamnya menggabungkan kesederhanaan *editor* kode sumber dengan *tools* pengembang yang canggih [5]. Visual Studio Code atau VSCode adalah *code editor* yang memiliki fitur untuk *cross platform* dengan *interface* yang bersih dan ringan, serta sudah memiliki banyak bahasa pemrograman dan ekstensi yang dapat dikustomisasi, selain itu juga VSCode juga mendukung pengguna untuk berkolaborasi secara *real-time* menggunakan ekstensinya [6].

  VSCode juga memiliki berbagai fitur yang mendukung produktivitas seorang *developer*. Fitur seperti *auto-completion* membantu *developer* dengan menampilkan saran kode saat mereka sedang mengetik kode, adapun fitur *refactoring* memungkinkan *developer* untuk mengubah struktur kode dengan aman [6].



Gambar 3.3. Logo Visual Studio Code

Sumber: [7]

- PHP dan Laravel

  Pada pengerjaan proyek HRIS atau *Human Resource Information System*, pengerjaannya menggunakan bahasa pemrograman PHP dan *framework* Laravel. Laravel sendiri adalah sebuah *framework* yang dirancang untuk mempermudah dan mempercepat proses pengembangan aplikasi berbasi *website*. Laravel merupakan salah satu *framework* yang banyak digunakan dikarenakan fitur yang diberikan dan juga komunitasnya yang aktif [8]. Berikut adalah beberapa fitur pada Laravel yang juga turut membantu pengerjaan proyek HRIS:

  1. *Eloquent ORM*

     Eloquent ORM adalah sistem ORM *Object-Relational Mapping* yang disediakan oleh Laravel untuk membantu *developer* dalam menghubungkan *database* menggunakan model.

  2. *Routing*

     Laravel juga menyediakan sistem *routing* untuk mengelola *URL* aplikasi secara terstruktur, fleksibel, dan mudah digunakan.

  3. *Blade Templating*

     Blade Templating adalah sebuah mesin *template* dari laravel yang ringan namun juga kuat untuk membuat sebuah tampilan yang dinamis.

  4. *Middleware*

     Middleware memungkinkan *developer* untuk memfilter *HTTP request* yang masuk ke aplikasi. Middleware sangat berguna untuk otentikasi

dan validasi input. Dengan kata lain *middleware* berfungsi untuk mengatur logika sebelum *request* diproses atau setelah respon diberikan.

5. *Artisan CLI*

   Artisan CLI

   Artisan CLI atau *Command Line Interface* bawaan pada Laravel dapat mempermudah *developer* untuk menjalankan tugas seperti migrasi *database*, pembuatan *controller*, dan model ataupun komponen-komponen lainnya

Gambar 3.4. Logo PHP

Gambar 3.5. Logo Laravel

Sumber: [9]

- Docker

  Docker adalah layanan yang di dalamnya menyediakan fungsi untuk mengemas dan menjalankan sebuah aplikasi dalam sebuah *container*. Dengan adanya isolasi ini dan keamanan yang memadai memungkinkan *developer* untuk menjalankan banyak container di waktu yang bersamaan pada host tertentu. Salah satu fitur yang sering digunakan pada proyek HRIS ini adalah docker *compose*. Docker *Compose* docker *compose* sendiri digunakan untuk menjalankan beberapa *container* sekaligus untuk menghemat banyak waktu

[10], dalam kasus ini docker *compose* digunakan untuk menjalankan *Local Host* dan juga *PHP My Admin*.

Sehingga pada proyek rancang bangun aplikasi HRIS menggunakan PHP Laravel, dan Docker. Hal ini dikarenakan Laravel sering dijalankan menggunakan Docker untuk memudahkan dalam pengelolaan lingkungan pengembangan dan produksi. Dengan adanya Docker, *developer* dapat menjalankan Laravel tanpa perlu mengatur server secara manual.

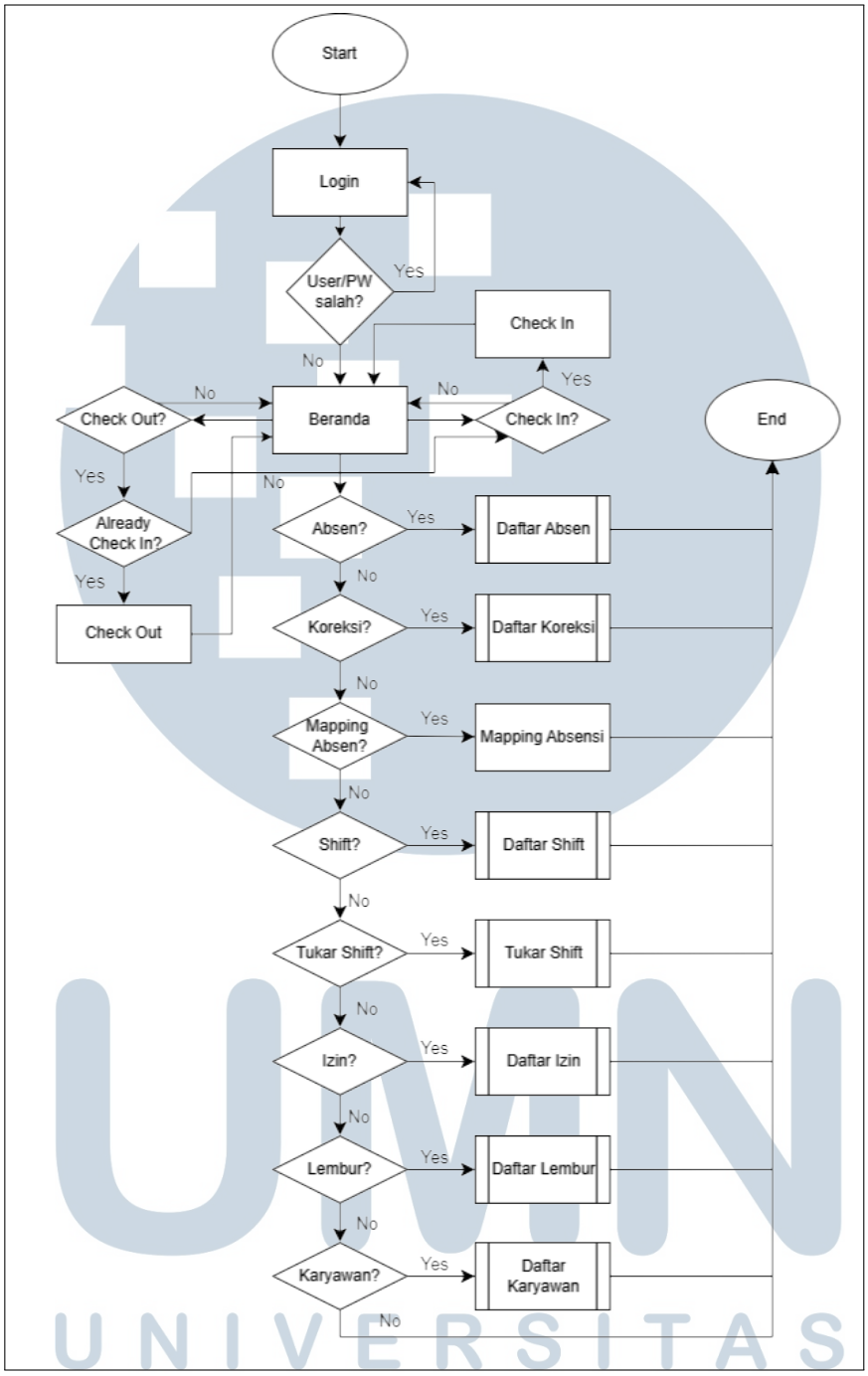
3.3.2 Flowchart, DFD, Database Schema, Fitur HRIS

A. *Flowchart* aplikasi HRIS

Pada Gambar 3.6 menampilkan *flowchart* atau alur penggunaan aplikasi HRIS secara keseluruhan, sebelum masuk pada bagian implementasi. Dengan adanya *flowchart* akan membantu pengguna untuk lebih memahami alur penggunaan aplikasi HRIS. Berikut akan dijelaskan mengenai penjabaran alur pada *flowchart* HRIS secara keseluruhan.

Seperti pada Gambar 3.6, hal pertama kali yang dilakukan adalah karyawan harus melakukan *login*. [ada saat karyawna melakukan *login*, karyawan diminta untuk memasukkan *username* dan *password* yang telah diberikan. Pada proses ini sistem akan mengecek atau memvalidasi apakah *username* dan *password* karyawan valid, jika valid maka karyawan atau *user* akan diarahkan ke halaman beranda, namun jika pada saat proses validasi terdapat kesalahan maka karyawan akan diarahkan ke halaman login kembali.

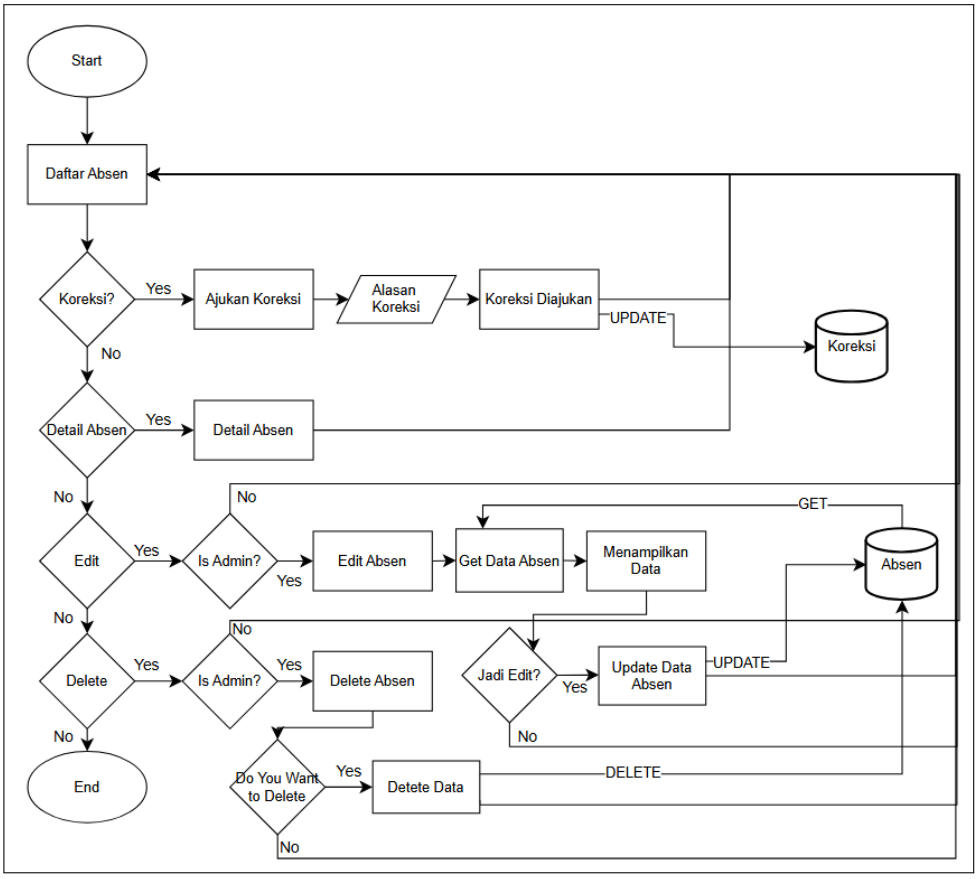
User yang telah berhasil *login*, akan diarahkan ke *landing page* HRIS yakni beranda, dimana pada halaman beranda ini *user* dapat melakukan *check in* ataupun *check out* jika pada hari itu terdapat jam kerja, namun jika tidak *check in* dan *check out* bersifat opsional dan karyawan dapat memilih untuk mengakses menu yang diinginkan. Menu-menu yang dapat diakses oleh pengguna adalah, absen, shift, izin, lembur, atau menu karyawan. Di dalam absen terdapat submenu lain yakni koreksi, dan *mapping* absensi. Koreksi bertujuan untuk mengajukan koreksi kepada admin jika pada saat *check in* atau *check out* terdapat kesalahan ataupun ingin memperbaiki absensi, sedangkan *mapping* absensi adalah fitur untuk melihat absensi karyawan pada perusahaan tersebut secara keseluruhan. Pada fitur *shift* terdapat juga submenu untuk melihat daftar shift dan juga melakukan pengajuan tukar shift dengan karyawan lain.



Gambar 3.6. Tampilan *flowchart* dari aplikasi HRIS

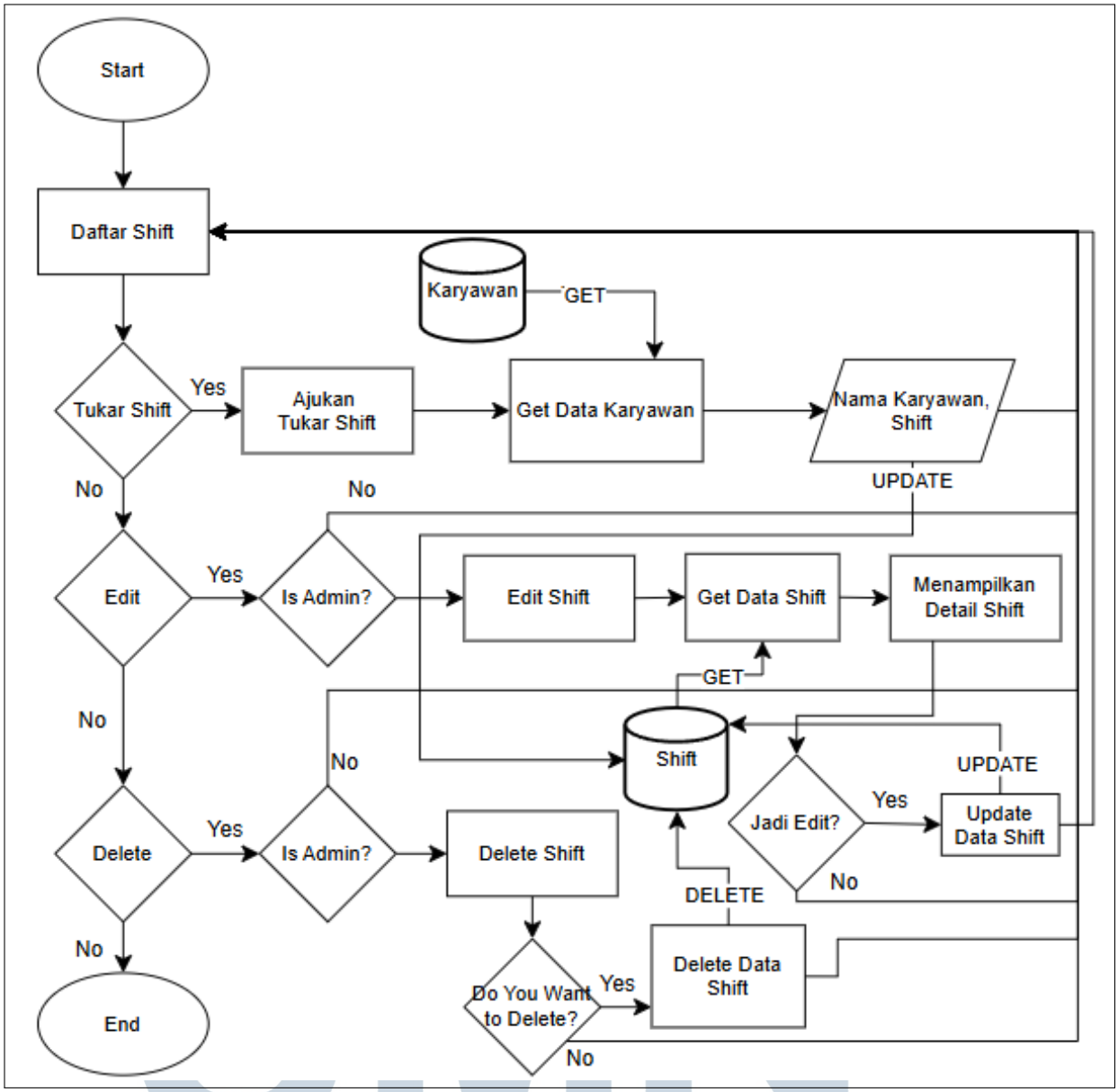
Selanjutnya pada Gambar 3.7 proses mengenai menu absen, dimana karyawan akan diarahkan kepada tampilan yang berisikan daftar absen. Pada daftar absen karyawan dapat memilih untuk melakukan koreksi absen, melihat detail absen. Pada menu absen juga terdapat *edit* dan *delete* absen, namun yang dapat *edit* dan juga *delete* hanya admin saja, dikarenakan di dalam aplikasi HRIS terdapat

role untuk membatasi apa yang bisa diakses oleh karyawan.



Gambar 3.7. Tampilan *flowchart* menu absen dari aplikasi HRIS

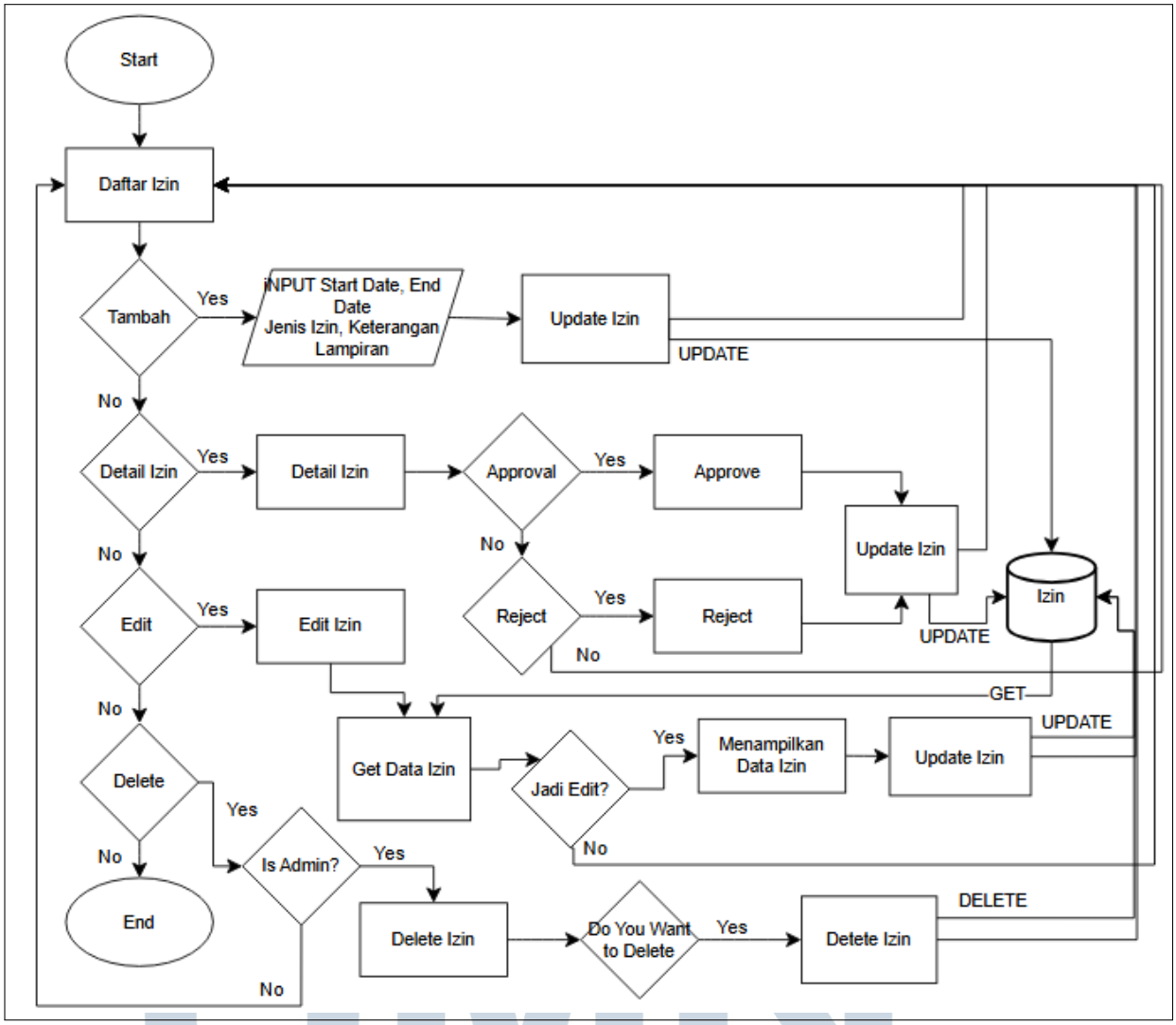
Selanjutnya pada Gambar 3.8 proses mengenai menu shift, dimana karyawan akan diarahkan kepada tampilan yang berisikan daftar shift karyawan-karyawan yang berada pada perusahaan yang sama, dikarenakan adanya *function* pada *index* shift untuk melakukan pengurutan karyawan yang berada pada perusahaan yang sama dengan *user* yang login jika rolenya bukan admin. Pada daftar shift karyawan dapat memilih untuk melakukan tukar shift dengan karyawan yang lain, ataupun melakukan *edit* dan *delete* shift, namun yang dapat *edit* dan juga *delete* hanya admin saja, dikarenakan di dalam aplikasi HRIS terdapat role untuk membatasi apa yang bisa diakses oleh karyawan.



Gambar 3.8. Tampilan *flowchart* menu shift dari aplikasi HRIS

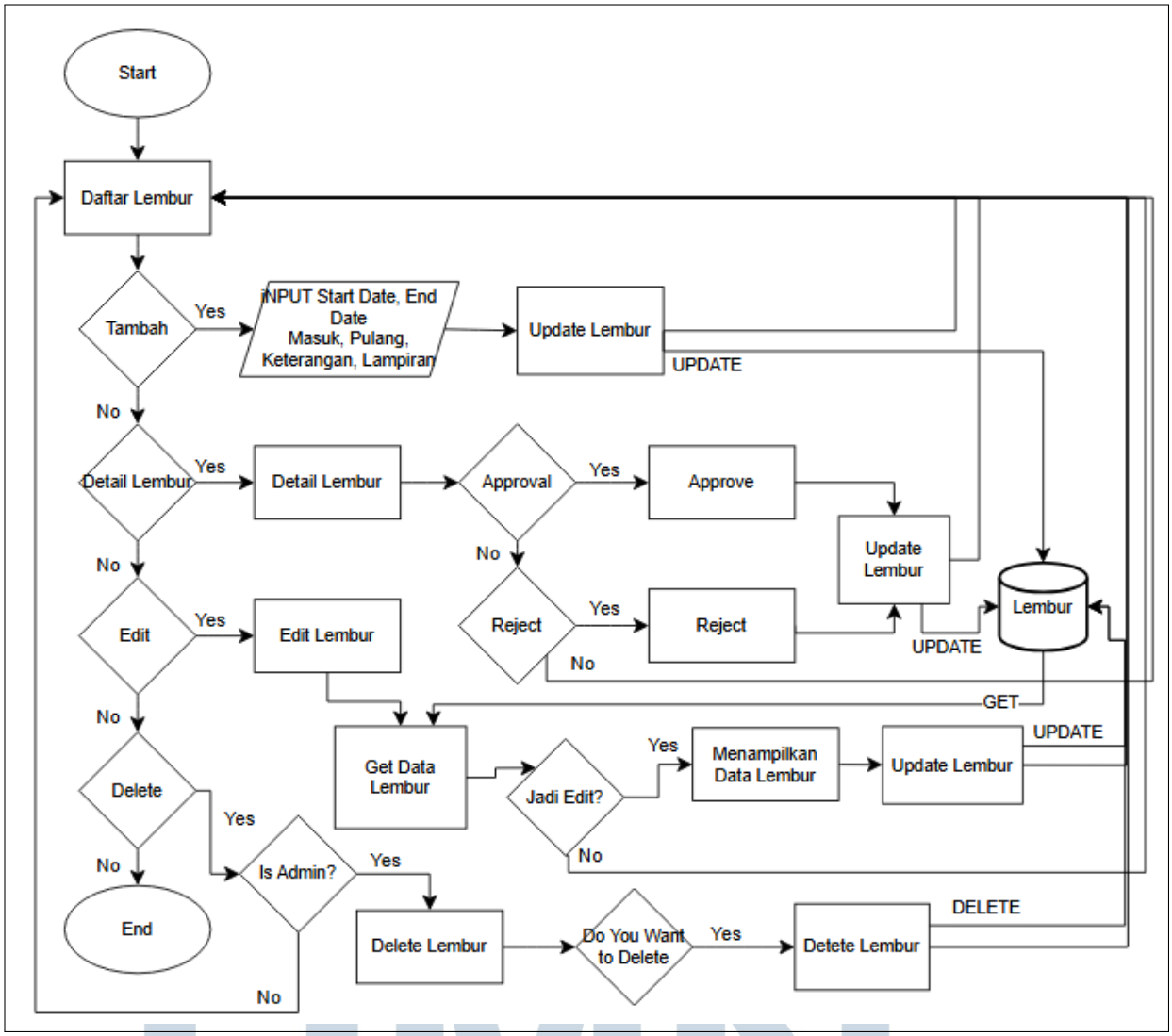
Selanjutnya pada Gambar 3.9 proses mengenai menu izin, dimana karyawan akan diarahkan kepada tampilan yang berisikan daftar izin. Pada menu ini karyawan dapat melakukan pengajuan izin atau cuti, dengan menekan tambah izin lalu melakukan input pada form yang diminta, setelah itu melakukan save dan izin berhasil diajukan. Pada menu izin juga karyawan dapat melakukan *edit* izin yang bertujuan untuk melakukan perbaikan jika jenis izin yang dipilih terdapat kesalahan ataupun memperbaiki jumlah hari izin. Selain itu pada menu izin terdapat detail izin untuk melakukan *approval* atau *reject* izin yang diajukan, adapun tombol delete pada menu izin berguna untuk melakukan penghapusan *record* izin yang diajukan

oleh karyawan, namun hanya admin saja yang dapat menghapus izin yang diajukan.



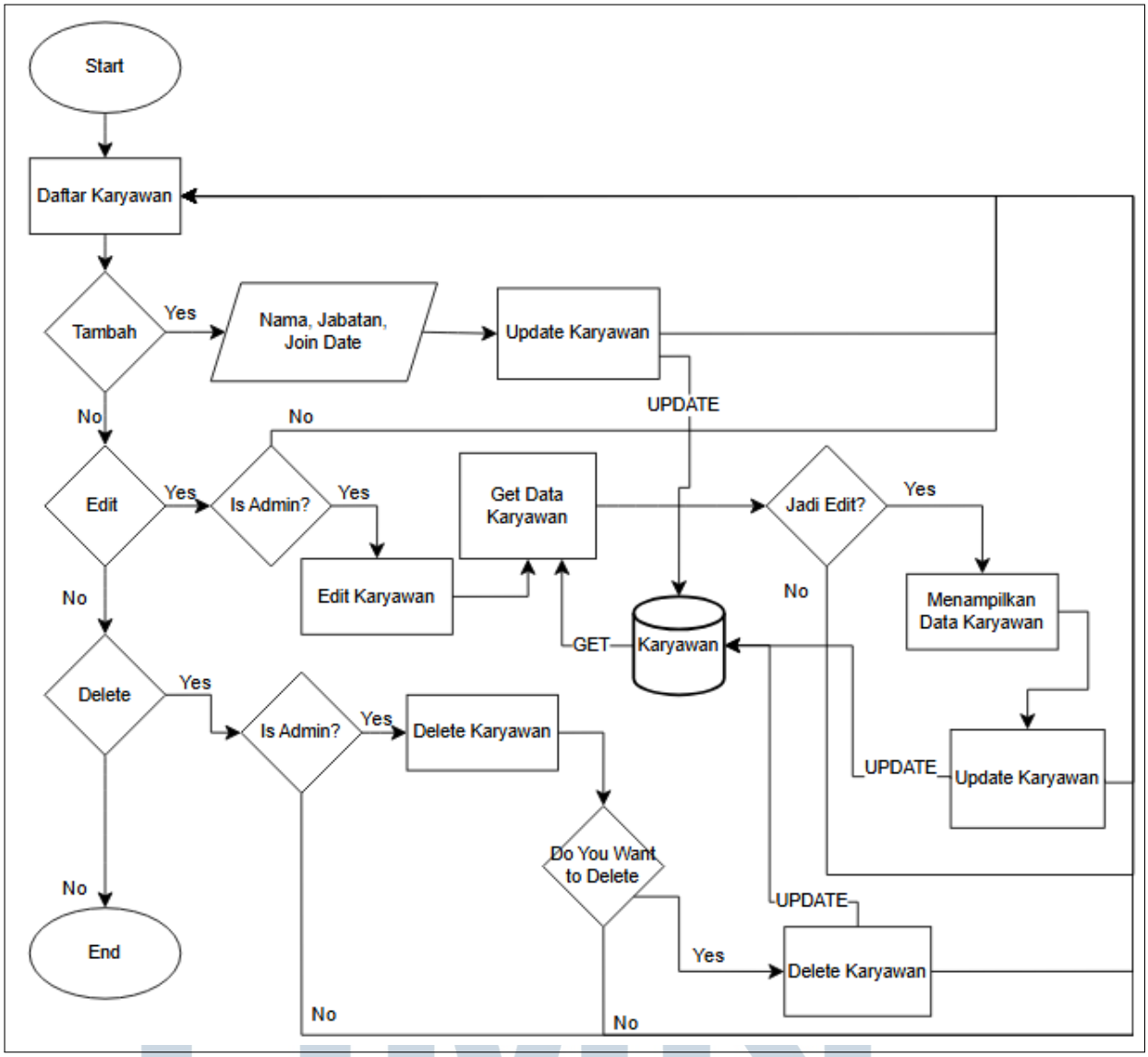
Gambar 3.9. Tampilan *flowchart* menu izin dari aplikasi HRIS

Selanjutnya pada Gambar 3.10 proses mengenai menu lembur, pada menu lembur sendiri memiliki kesamaan dengan menu izin. Perbedaan yang terdapat hanya pada penggunaannya saja, dimana lembur digunakan jika karyawan yang masuk pada hari itu ternyata harus bekerja lembur sehingga mereka harus mengajukan lembur pada menu lembur.



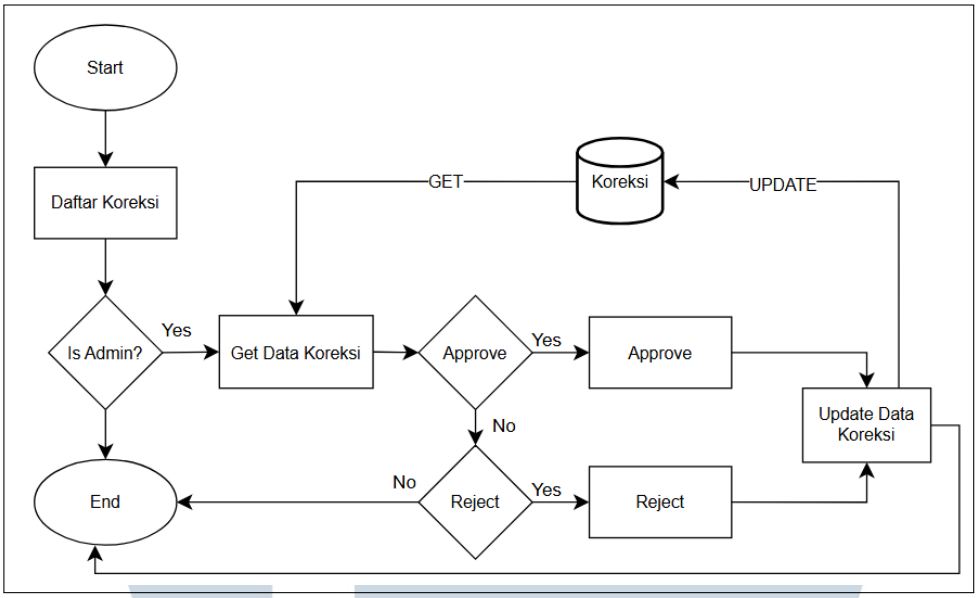
Gambar 3.10. Tampilan *flowchart* menu lembur dari aplikasi HRIS

Selanjutnya pada Gambar 3.11 proses mengenai menu karyawan, pada menu karyawan bertujuan untuk melakukan penambahan karyawan pada perusahaan. Pada menu ini, jika ingin melakukan penambahan karyawan, user diminta untuk mengisi form yang berisikan nama karyawan, jabatan, dan join date. Pada menu ini juga edit dan delete hanya bisa dilakukan oleh admin saja.



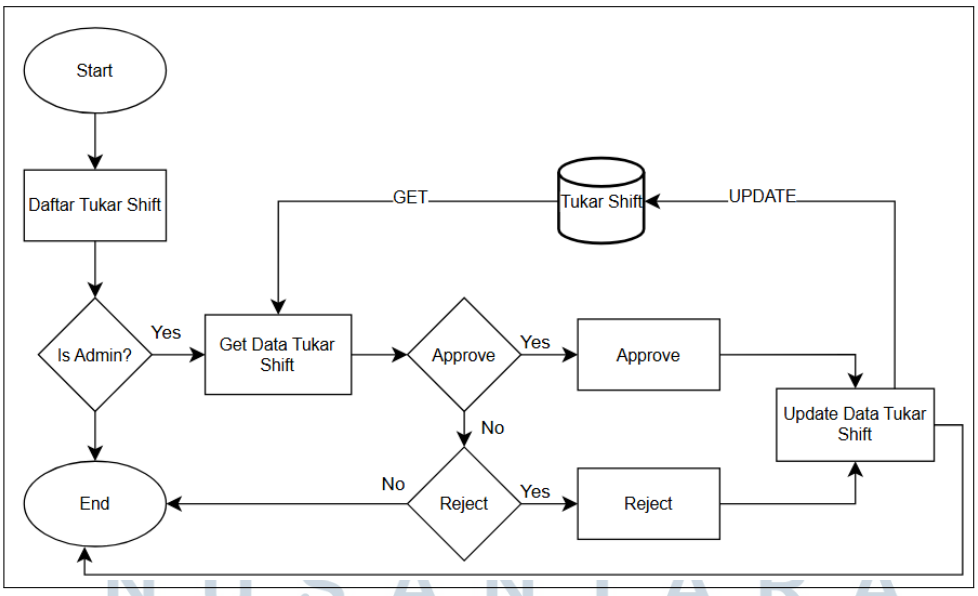
Gambar 3.11. Tampilan *flowchart* menu karyawan dari aplikasi HRIS

Selanjutnya pada Gambar 3.12 proses mengenai submenu absen yakni menu daftar koreksi. Pada menu daftar koreksi bertujuan untuk melakukan *approval* atau *reject* yang dilakukan oleh admin terkait koreksi yang sebelumnya sudah diajukan oleh karyawan pada menu absen.



Gambar 3.12. Tampilan *flowchart* menu daftar koreksi dari aplikasi HRIS

Selanjutnya pada Gambar 3.13 proses mengenai submenu absen yakni menu tukar shift. Pada menu tukar shift bertujuan untuk melakukan *approval* atau *reject* yang dilakukan oleh admin terkait pengajuan tukar shift karyawan yang mengajukan dengan karyawan yang diinginkan, yang sebelumnya juga sudah diajukan oleh karyawan pada menu shift.



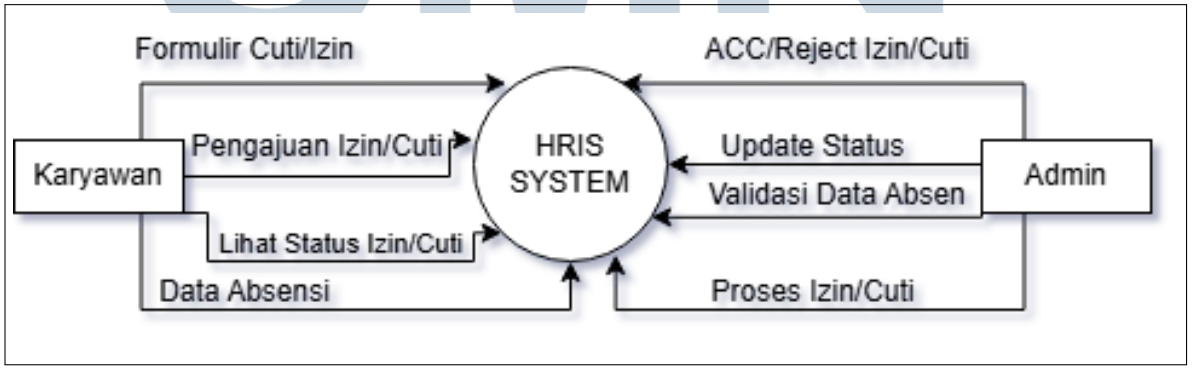
Gambar 3.13. Tampilan *flowchart* menu daftar koreksi dari aplikasi HRIS

B. DFD HRIS

DFD atau *data flow diagram* adalah sebuah gambaran tentang bagaimana alur sistem dari aplikasi yang dirancang bekerja [11]. DFD sendiri memiliki level dari nol hingga level dua, dimana setiap levelnya memiliki fungsi atau representasi masing-masing. DFD level 0 digunakan untuk merepresentasikan kegunaan secara global dari sebuah sistem yang dibangun. DFD level 1 digunakan jika proses-proses pada level 0 masih terlalu kompleks, misalnya pada HRIS terdapat ”validasi waktu kerja” yang di dalamnya masih terdapat subproses seperti validasi *shift*, validasi lokasi kerja, dan lainnya. Untuk DFD level 2 hanya dibutuhkan pada saat subproses pada level 1 masih terlalu kompleks. Dibawah ini akan dibahas lebih lanjut untuk DFD level 0 dan level 1 pada HRIS:

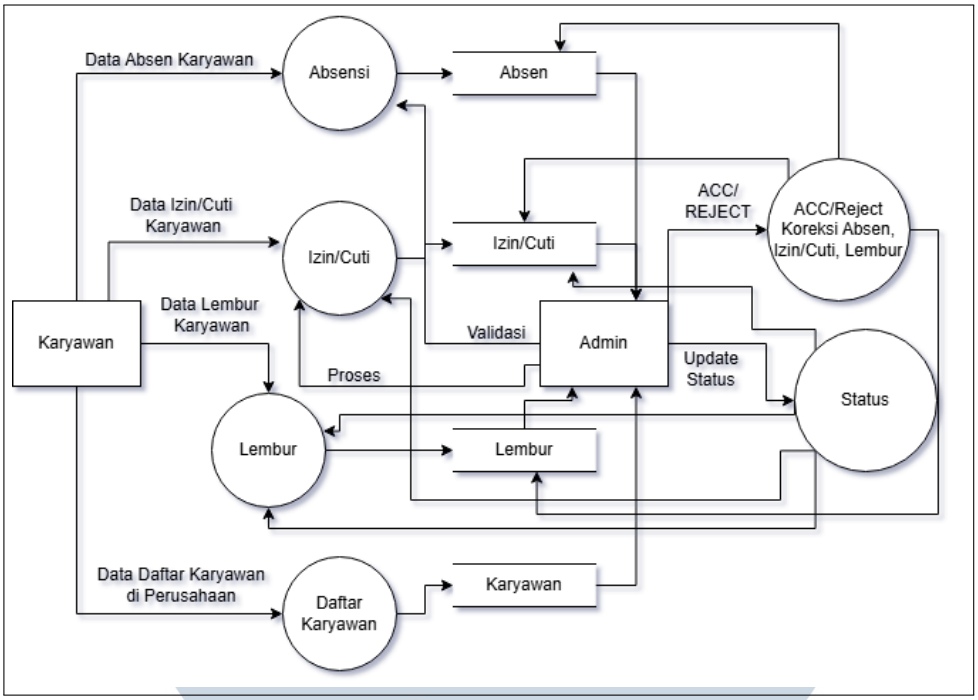
- DFD Level 0

  Pada Gambar 3.14 menunjukkan *data flow diagram* pada level 0 yang dimana DFD level 0 memiliki komponen utama yakni karyawan, dan admin. DFD level 0 juga menunjukkan proses-proses di dalam HRIS yaitu proses pengolahan data absensi, proses pengajuan cuti/izin. Alur utama pada DFD level 0 HRIS meliputi input data absensi, cuti, atau izin, yang akan digunakan untuk berbagai proses pada aplikasi nantinya, lalu akan diproses oleh sistem HRIS, setelah data berhasil diproses maka data tersebut akan dikirimkan ke admin. Admin akan bertugas sebagai komponen untuk melakukan *approve/reject*, validasi data absen, melihat proses pengajuan izin/cuti dan terakhir data tersebut akan dikirim kembali kepada admin ataupun karyawan sehingga bisa melihat status izin/cuti yang diproses.



Gambar 3.14. *Data Flow Diagram* level 0 dari aplikasi HRIS

- DFD Level 1

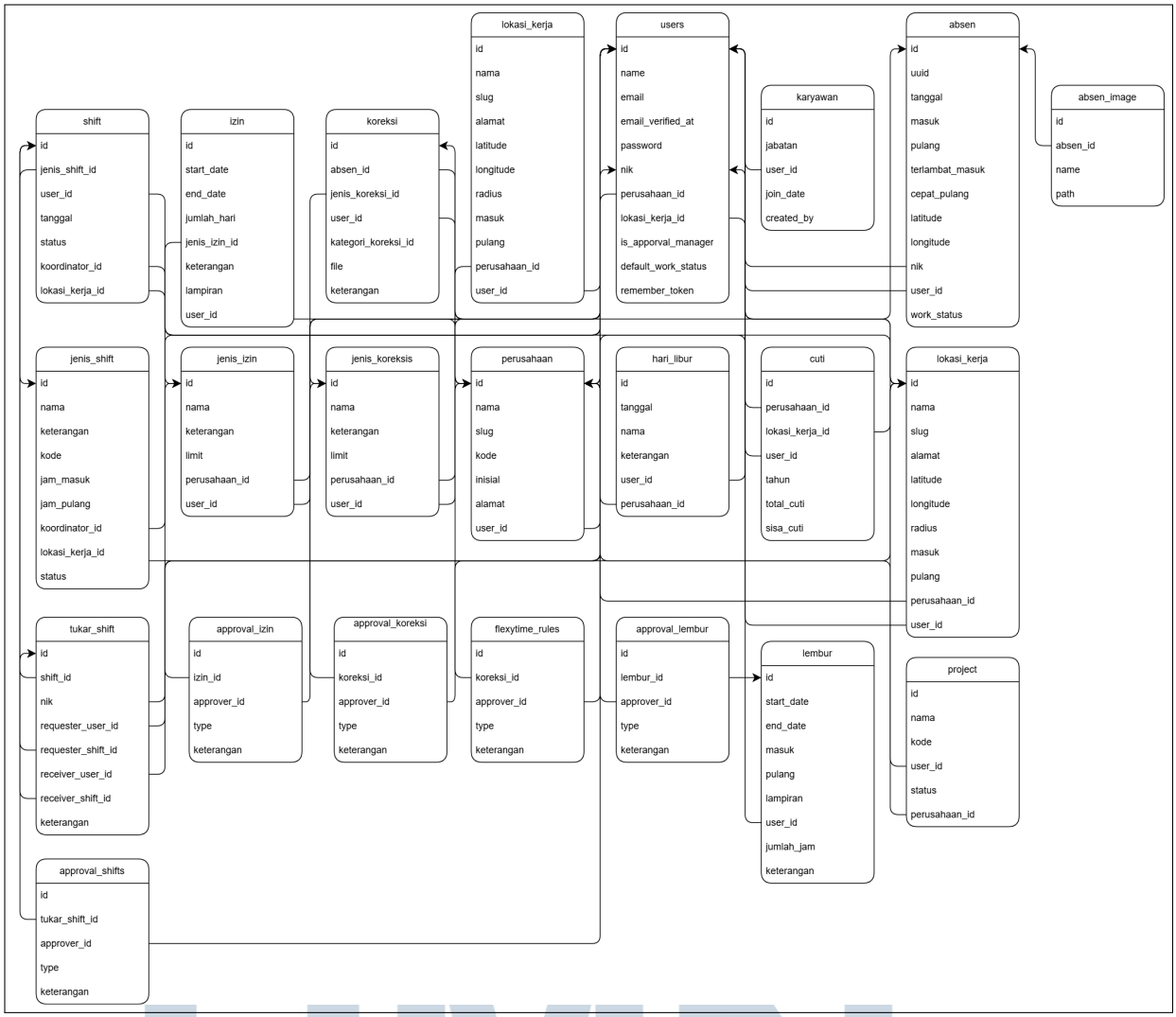


Gambar 3.15. *Data Flow Diagram* level 1 dari aplikasi HRIS

Pada Gambar 3.15 menunjukkan *data flow diagram* pada level 1 yang menunjukkan penggambaran lebih rinci mengenai proses-proses yang ada pada HRIS. DFD Level 1 ini menunjukkan bahwa karyawan akan bisa melihat data absensi, data pengajuan izin/cuti, data lembur, dan juga data karyawan yang berada di perusahaan yang sama. Pada sisi admin, admin akan menerima semua data yang disimpan di database masing-masing tabel yang akan digunakan untuk melakukan *approval/reject* terkait koreksi absen yang diajukan, izin/cuti yang diajukan, dan juga lembur yang diajukan karyawan. Dengan kata lain, admin bertindak sebagai validasi/pengontrol. Selain itu, admin juga dapat melakukan CRUD (*Create, Read, Edit, Update* data yang dipilih dari proses sebelumnya.

C. *Database Scheme*

Dibawah ini ini merupakan *database scheme* dari aplikasi HRIS

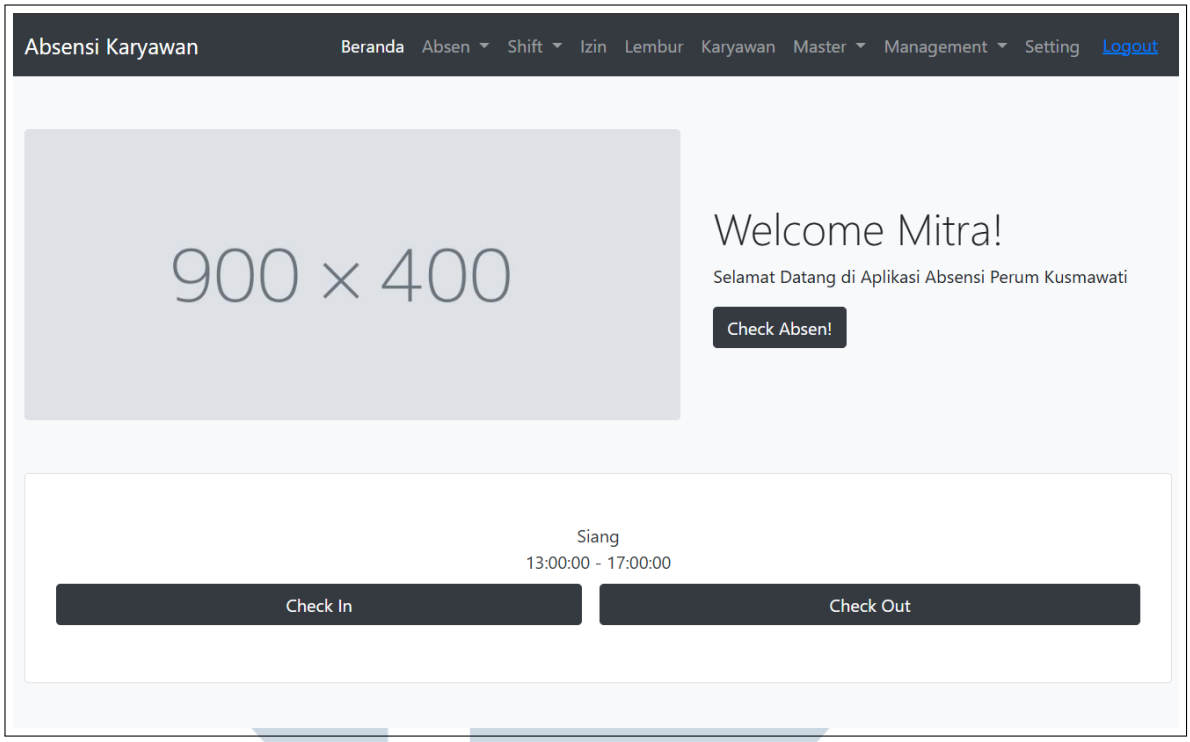


Gambar 3.16. *Database Scheme* dari aplikasi HRIS

Gambar 3.16 menunjukkan skema *database* pada aplikasi HRIS yang dikembangkan, yang menunjukkan relasi atau hubungan antar tabel pada *database*.

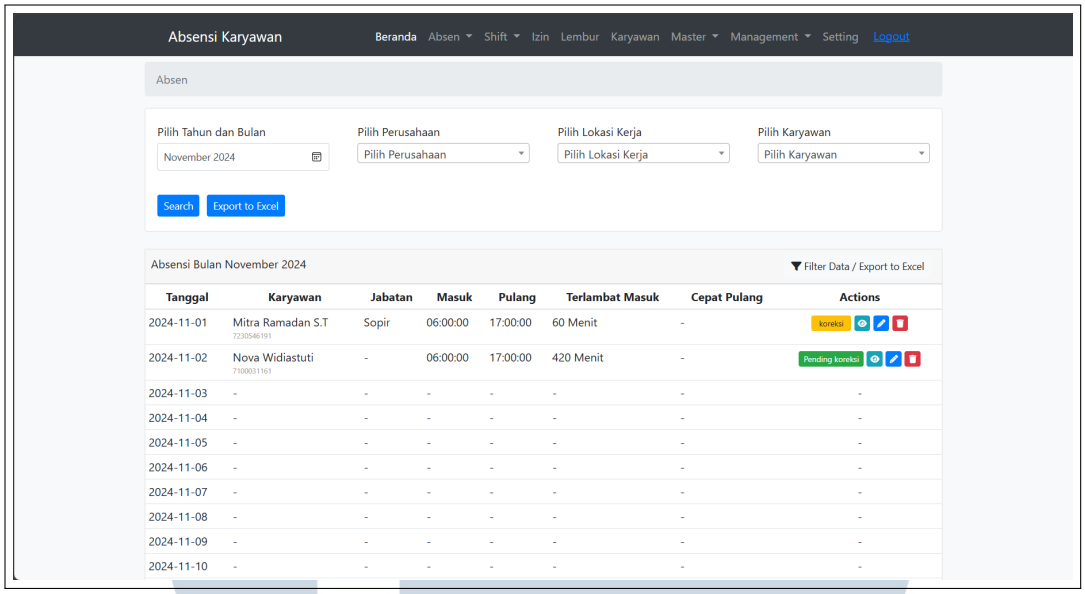
D. Fitur pada Proyek HRIS

Pada Gambar 3.17 merupakan tampilan dari *home* aplikasi HRIS, dimana pengguna akan melakukan *check in* sebelum melanjutkan aktivitas, dan melakukan *check out* setelah selesai jam kerja. Pada halaman *home* terdapat juga jadwal *shifting* pada hari itu. Selain itu di halaman *home* terdapat juga fitur pengkalkulasian persentasi kehadiran di bulan saat ini, dan terdapat juga jumlah total telat masuk dalam menit bulan ini.



Gambar 3.17. Tampilan *Page Home* dari aplikasi HRIS

Pada Gambar 3.18 menunjukkan fitur absensi yang dimana absensi karyawan akan diurutkan berdasarkan bulan saat ini dan juga jumlah tanggal yang sudah disesuaikan berdasarkan bulan saat ini. Di menu absensi ini terdapat fitur untuk melakukan pengajuan koreksi jika karyawan terlambat masuk karena alasan tertentu, ataupun pengajuan koreksi absen untuk perizinan cepat pulang. Selain itu pada Gambar 3.18 juga terdapat action bergambar mata yang berguna untuk melihat detail absen, gambar pensil untuk melakukan *edit* absen, dan gambar seperti kotak sampah untuk fitur *delete* absen. Pada menu absensi juga terdapat *sub-menu* untuk melakukan *approval* dan mapping absensi.

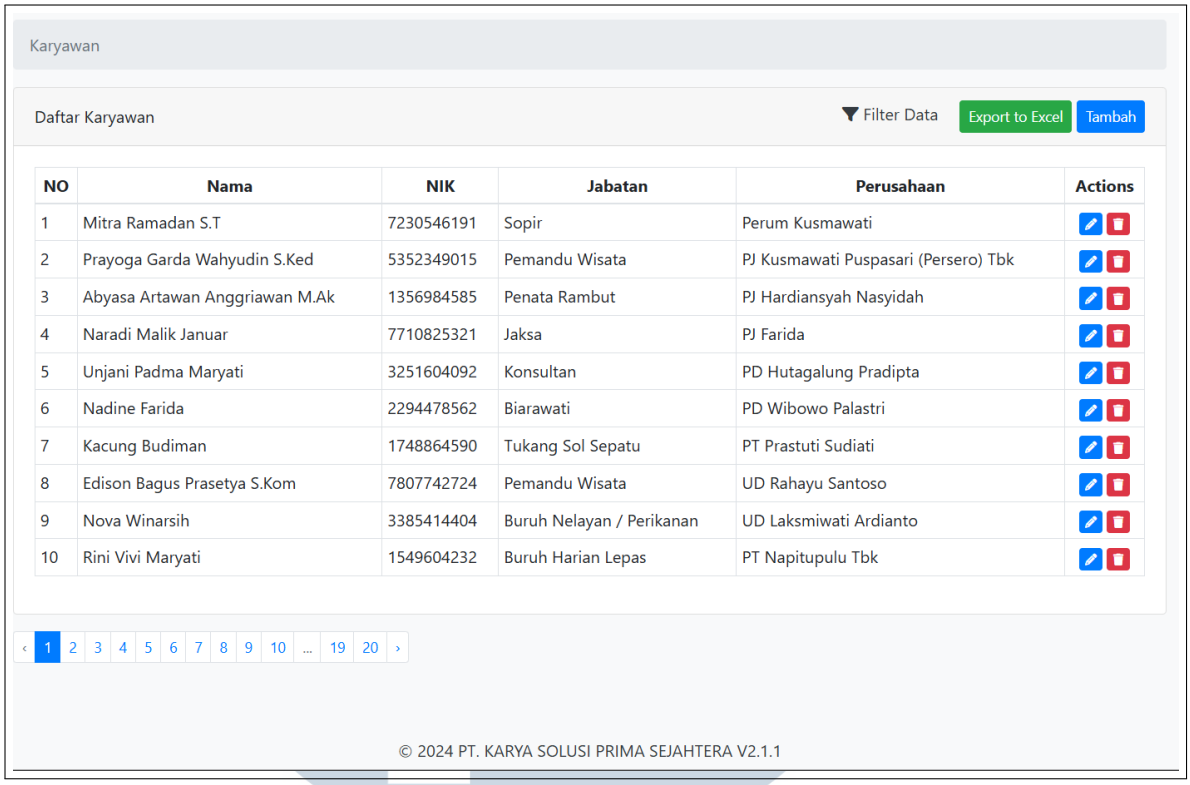


Gambar 3.18. Tampilan *Page* Absensi dari aplikasi HRIS

Pada Gambar 3.19 adalah menu dimana karyawan dapat melakukan pengajuan izin baik itu cuti tahunan, ataupun izin yang lainnya. Pada menu ini juga supervisi atau *approver* dapat melakukan *approval* terkait izin yang diajukan dengan menekan action bergambar mata. Karyawan juga dapat melakukan *edit* terkait izin yang diajukan dengan menekan action bergambar pensil.

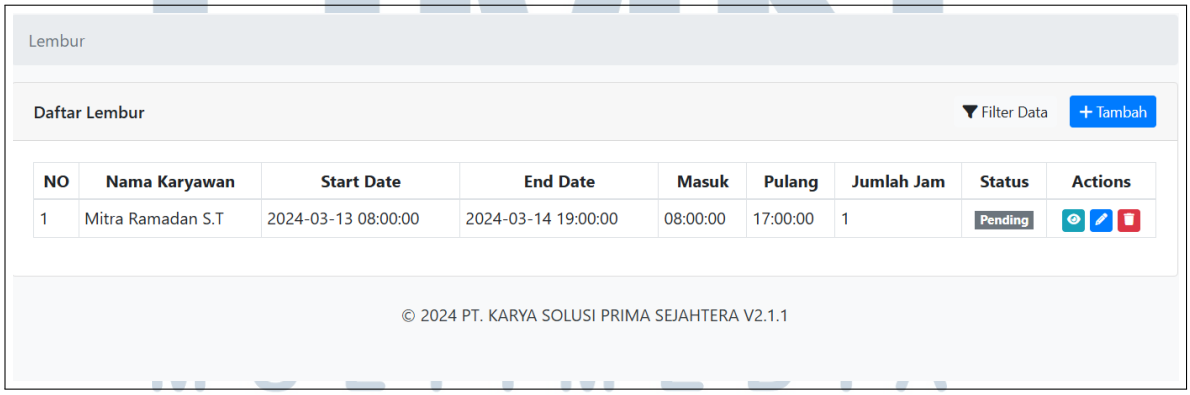
Gambar 3.19. *Page* Izin

Pada Gambar 3.20 yang merupakan tampilan dari menu karyawan bertujuan untuk menambah list karyawan baru dan juga untuk melakukan pengecekan nama-nama karyawan pada perusahaan.



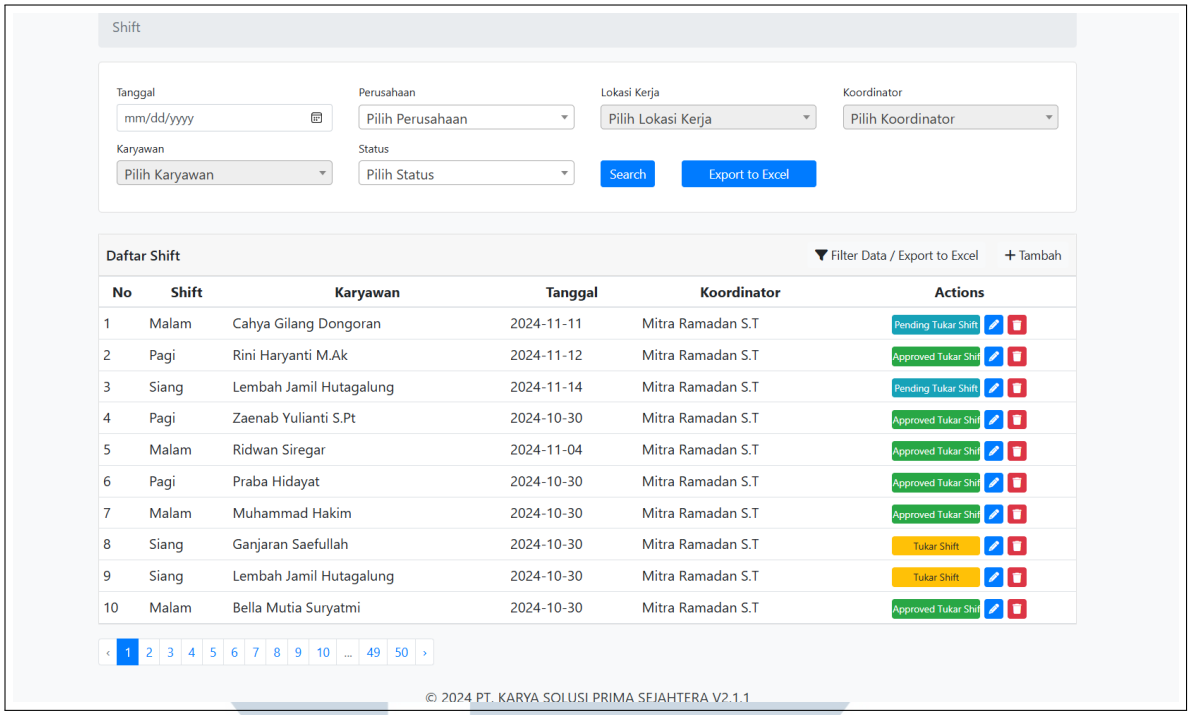
Gambar 3.20. Tampilan *Page Karyawan* dari aplikasi HRIS

Pada Gambar 3.21 adalah tampilan dari menu lembur yang bertujuan untuk melihat karyawan yang mengajukan lembur dari sisi admin, supervisi, dan *approver*. Dari sisi karyawan menu ini berguna untuk pengajuan lembur, jika karyawan pada hari kerja terdapat lembur.



Gambar 3.21. Tampilan *Page* Lembur dari aplikasi HRIS

Pada Gambar 3.22 merupakan tampilan dari menu shift dimana admin bisa melihat karyawan siapa saja yang memiliki shift pada pagi, siang, ataupun malam. Karyawan juga bisa mengajukan tukar shift dari *sub-menu* dalam menu shift, yang nantinya admin akan melakukan *approval* dari action ”pending tukar shift”.



Gambar 3.22. Tampilan *Page Shifting* dari aplikasi HRIS

3.3.3 Implementasi

A. Implementasi *Codingan* pada bagian *Index, Store* Menu Absen

Untuk implementasi index absensi dapat dilihat dari potongan Kode 4.1, yang dimana di *index* terdapat filter untuk melakukan pencarian perusahaan, lokasi kerja, dan karyawan yang diambil dari tabel masing-masing dengan melakukan inisiasi variabel ”perusahaan”, ”lokasi_kerja”, dan ”karyawan” dengan mengambil ID dari karyawan yang akan di *filter*. Pada potongan kode 4.1 terdapat $*query* yang digunakan untuk mengambil data absensi berdasarkan tahun dan bulan yang telah dipilih. Setelah melakukan inisialisasi menggunakan $*query* barulah tiga variabel pertama untuk melakukan filter dijalankan menggunakan function ”*if*”. Setelah itu data yang ada akan masuk ke dalam proses *fecth* dan diformat sesuai filter, termasuk relasi users, dan mengelompokkan data berdasarkan tanggal dan di akhir akan dikembalikan menggunakan fungsi ”*return view*”.

Pada potongan Kode 4.2 pertama kali dilakukan adalah untuk memastikan data yang diterima dari pengguna memiliki format atau atribut yang benar sebelum diproses lebih lanjut menggunakan *”request validate”*, yang selanjutnya akan dilakukan *testing* melalui fungsi *try & catch* untuk melakukan pengecekan terkait apakah *user* memiliki shift pada hari itu, mengecek jam kerja berdasarkan lokasi

kerja, menghitung radius pada saat melakukan *check in*. Langkah terakhir dari metode *store* adalah menyimpan ke dalam *database* yang bisa dilihat pada potongan Kode 4.2 bagian $data. Setelah melalui proses ini akan dilakukan *testing* kembali menggunakan metode *catch* untuk melihat apakah ada kesalahan selama proses penyimpanan data dan jika tidak ada akan dikembalikan ke tampilan absen *index*.

B. Implementasi *Codingan* pada bagian *Index, Store* Menu Izin

Pada potongan Kode 4.3 ditampilkan *script* untuk *index* izin, yang di dalamnya sama seperti *index* untuk absen, dimana akan dilakukan inisialisasi awal untuk variabel filter yang selanjutnya akan dijalankan setelah adanya query untuk mengambil data izin dan akan diproses menggunakan fungsi "*if*". Berbeda dari *index* absen, pada *index* izin terdapat perintah untuk melakukan *pagination* yang dibuat agar tampilan pada izin dibagi 10 *item* per halaman. Pada akhir potongan kode data akan diambil dan dimasukkan kedalam variabel yang telah dibuat dan akan dikembalikan ketampilan melalui *return view* .

Pada potongan Kode 4.4 untuk metode *store* izin memiliki fungsi yang sama seperti metode *store* pada menu absen, dimana pertama kali yang dilakukan adalah validasi, lalu akan dilanjutkan menggunakan metode *try & catch*. Di dalam metode ini akan dilakukan pengecekan untuk limit izin. Limit izin digunakan untuk validasi saat *user* melakukan *request* izin, jika limit sudah mencapai batas maka *user* tidak bisa melakukan pengajuan izin. Pada bagian ini juga akan dilakukan pengecekan melalui variabel $pendingIzinCount untuk mengecek apakah *user* memiliki izin yang masih pending, jika user masih memiliki izin yang pending maka user bisa melakukan *request* izin kembali. Namun jika *user* memiliki *request* yang sudah diterima dan sudah mencapai limit maka user tidak bisa melakukan *request* untuk izin dengan jenis izin tersebut, hal ini bisa dilihat dari potongan Kode 4.4 pada variabel $approvedIzinCount, yang selanjutnya akan ditotalkan melalui variabel $totalIzinCount dan akan dicek kembali melalui fungsi *if*. Setelah semua metode dilakukan maka izin akan disimpan ke dalam *database* dan akan dikembalikan ke *index* izin.

C. Implementasi *Codingan* pada bagian *Index, Store* Menu Karyawan

Pada potongan Kode 4.5 ditampilkan potongan kode bagian *index* karyawan yang memiliki fungsi yang sama dengan index pada dua menu sebelumnya.

Potongan kode diatas menjelaskan bagaimana filter dapat digunakan, namun yang berbeda pada potongan kode diatas adalah di dalamnya terdapat *whereHas* yang berguna untuk melakukan filter berdasarkan karyawan yang memiliki hubungan dengan perusahaan tertentu berdasarkan relasi. Pada potongan kode 4.5 juga menggunakan *slug* yang berguna untuk mengidentifikasi perusahaan karena *slug* berperan sebagai *unique column* dalam format *URL-friendly*.

Pada potongan Kode 4.6 menampilkan fungsi untuk melakukan *store* data pada menu karyawan. Memiliki fungsi yang sama dengan dua menu sebelumnya dimana akan dilakukan validasi terlebih dahulu, lalu menyimpan data ke dalam tabel karyawan dan akan dikembalikan ketampilan *index* karyawan.

D. Implementasi *Codingan* pada bagian *Index, Store* Menu Lembur

Pada potongan Kode 4.7 bagian *index* lembur akan menampilkan semua data dari *database* lembur ke dalam *index*. Fungsi *controller index* pada menu lembur memiliki kegunaan yang sama dengan *controller* lainnya yang telah dijelaskan, dimana *controller* akan melakukan inisialisasi *query* terlebih dahulu untuk melakukan pemrosesan untuk filter data, lalu data yang ada akan ditampilkan dengan menambahkan paginasi di *view*. *Query* yang dilakukan menggunakan *join* dan *left join* untuk menggabungkan data dari dua tabel, *join* berguna Jika tidak ada kecocokan pada data, maka baris tersebut akan dihilangkan dari hasil query. Sedangkan *left join* sama-sama berguna untuk menggabungkan dua data dari tabel, namun akan menampilkan semua baris dari tabel kiri atau tabel pertama, meskipun tidak ada kecocokan di tabel kanan atau tabel kedua. Jika tidak ada kecocokan di tabel kanan, maka kolom dari tabel kanan akan berisi *NULL*.

E. Implementasi *Codingan* pada bagian *Index, Store* Menu Shift

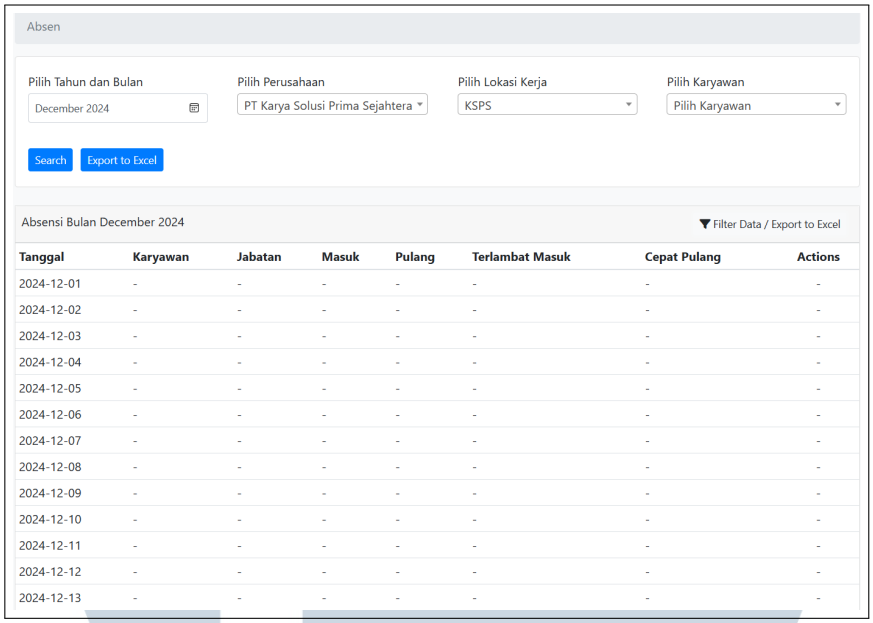
Pada potongan Kode 4.8 menampilkan baris kode untuk *index shift*, dimana *index shift* juga memiliki kegunaan untuk melakukan filter seperti *index* pada menu lainnya. Ada sedikit perbedaan pada inisialisasi awal untuk melakukan *query*, dimana *query* menggunakan *with* yang bertujuan untuk menggunakan *eager loading* dengan relasi yang didefinisikan pada model. *Eager loading* sendiri adalah suatu cara yang digunakan untuk memuat data yang didefinisikan di model relasi dengan data utama dalam satu *query SQL*, sehingga dapat mengurangi jumlah dari *query* yang dieksekusi selama aplikasi itu berjalan. Setelah data dilakukan *query* barulah

masuk ke dalam fungsi untuk melakukan filter dengan menggunakan metode *if* untuk mencari karyawan, perusahaan, lokasi kerja, dan koordinator. Data yang didapatkan, akan di fetch dan akan dikembalikan ketampilan menggunakan paginasi yang dibatasi 10 *item* per halaman.

Pada potongan Kode 4.9 menampilkan metode untuk *store* data ke dalam *database shift* dengan melakukan validasi terlebih dahulu. Data yang sudah divalidasi akan dicek menggunakan metode *try & catch*, jika tidak ada masalah maka data akan langsung tersimpan ke dalam *database*, fungsi *try & catch* pada semua *controller* digunakan untuk menangkap dan menangani keselahan yang mungkin terjadi selama eksekusi berjalan. Jika ada kesalahan, maka paramater *catch* akan menangkap dan menghentikan proses lalu memunculkan kesalahan menggunakan *"dd"* atau *dump and die*, yang merupakan fungsi bawaan dari Laravel untuk melakukan *debugging*. Jika selama proses tidak ada kesalahan yang terjadi, maka data akan tersimpan ke dalam tabel *shift* dan *user* akan diarahkan kembali ke *index shift*.

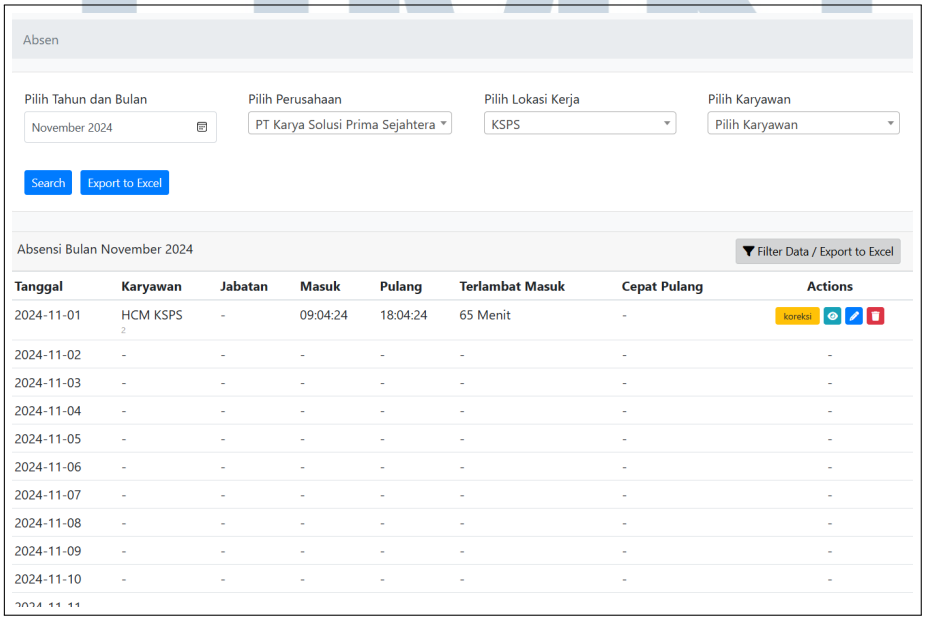
F. Contoh Pengujian Filter di Blade

Pengujian yang dilakukan adalah pada *filter*, dikarenakan aplikasi yang dikembangkan belum selesai dan masih terdapat beberapa perubahan. Sehingga untuk melampirkan tentang uji coba aplikasi HRIS masih belum bisa dilampirkan, namun terdapat *filter* yang sudah pasti, sehingga bisa dilampirkan sebagain pengujian aplikasi. Berikut potongan Kode 4.10 yang menunjukkan potongan kode untuk implementasi filter dari *controller* untuk *blade* atau tampilan, menunjukkan *filter* akan dibungkus menggunakan *div* dengan "*class collapse in*" yang bertujuan bahwa, filter merupakan elemen yang dapat diperluas atau dikurangi dan dalam kondisi terbuka. Selanjutnya filter akan dipanggil melalui metode "*GET*", melalui *route* absen *index*. Pada filter, agar filter bekerja maka diperlukan inisiasi awal di *index* yang nantinya akan dipanggil di dalam *blade* menggunakan "*name*" sesuai dengan nama variabel yang telah dibuat di *index controller*. Tujuan adanya filter ini untuk mempermudah pengguna jika ingin mencari data tertentu, sehingga pengguna tidak perlu mencari satu per satu data yang ada. Dibawah merupakan gambar dari hasil pengujian filter yang telah dibuat untuk memfilter absen karyawan.



Gambar 3.23. Tampilan hasil filter dari aplikasi HRIS

Pada Gambar 3.23 menampilkan hasil pengujian filter pada menu absen karyawan, jika filter digunakan untuk mencari data absensi karyawan berdasarkan bulan Desember pada tahun 2024, dengan perusahaan PT Karya Solusi Prima Sejahtera, dan lokasi Kerja KSPS. Dari Gambar 3.23 data tidak mendapatkan hasil karyawan yang melakukan absensi, hal ini dikarenakan *database* pada bulan desember tidak memiliki data karyawan yang tercatat melakukan absensi dan berikut adalah contoh hasil uji filter absensi jika terdapat *record* absen karyawan.



Gambar 3.24. Tampilan hasil filter jika terdapat data

Pada Gambar 3.24 menunjukkan, filter data karyawan pada bulan November 2024, dengan perusahaan PT Karya Solusi Prima Sejahtera, dan lokasi kerja KSPS. Dimana filter mendapatkan data absen karyawan pada bulan itu, dikarenakan terdapat data absensi karyawan di dalam *database* absensi pada bulan November 2024.

3.4 Kendala dan Solusi yang Ditemukan

Kendala selama pembuatan menu absen, izin, karyawan, lembur, dan *shift* untuk proyek HRIS ini adalah pada bagian *controller* dan juga *blade*. Dimana diperlukannya integrasi dari *front end* ke *back end* yang cukup kompleks. Pada saat pengembangan aplikasi juga terdapat beberapa error lainnya seperti kesalahan pada model untuk relasi dengan tabel lainnya, kesalahan pada logika di *controller* yang menyebabkan *fetch* data ke *blade* menjadi terhambat.

Selain itu masalah atau kendala lainnya adalah *request* yang diberikan oleh atasan atau supervisi terkait dengan tampilan yang terus berubah-ubah, sehingga sedikit menghambat pengerjaan proyek. Adapun masalah lainnya adalah terkait dengan *database* atau *migration file* yang tidak sesuai dengan model, sehingga diperlukannya migrasi ulang agar sesuai dengan model. Kendala lainnya adalah, *route* yang dibuat tidak sesuai, sehingga *URL* tidak ditemukan dan terdapat beberapa perubahan lainnya seperti alur atau menu yang dihilangkan atau disatukan setelah dibuat.

Solusi yang bisa diberikan untuk *controller* dan juga *blade* terkait integrasi *front end* dan *back end* adalah dengan memperhatikan lebih detail variabel yang ada, selain itu untuk *developer* lebih memperhatikan logika pada *controller* dikarenakan pada php Laravel, *controller* mempunyai peran penting untuk melakukan logika pemrograman. Solusi untuk masalah model dan *migration table* adalah dengan membangun sebuah model dan juga *migration table* yang pasti, sehingga untuk membangun sebuah menu tidak mengalami kendala karena model dan *migration table* sudah pasti. Terkait solusi untuk tampilan yang berubah-ubah, solusi yang bisa diberikan adalah merancang terlebih dahulu tampilan yang diinginkan seperti apa, sehingga pengerjaan tidak terhambat ataupun berulang kali mengerjakan hal yang sama untuk mengganti atau mengubah tampilan yang sudah dikerjakan sebelumnya.