BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang, penulis memiliki peranan sebagai *software developer (front end)*. Kerja magang dilakukan dalam tim *project* Timesheet and Attendance, di bawah bimbingan Bapak Hans Permana sebagai supervisi tim. *Project* dijalankan dengan dibagi-bagi menjadi periode yang disebut dengan *sprint.* Setiap 2 minggu sekali, dilakukan *sprint planning* yang bertujuan untuk membagi *task* yang akan dikerjakan selama periode *sprint* tersebut. *Task* atau *backlog* disusun oleh *product owner* yang selanjutnya ditentukan bersama-sama apa saja yang akan diambil dalam *sprint* tersebut dan berapa estimasi poin dari setiap *task*. Selain melakukan *planning,* pada hari yang sama dilakukan juga *showcase* yang bertujuan untuk menunjukkan hasil kerja selama periode *sprint* oleh setiap member tim, serta *retrospective* untuk mendiskusikan kendala ataupun hal yang berjalan dengan baik selama *sprint* tersebut.

Setiap harinya, dilakukan *stand up* yang bertujuan untuk melaporkan apa yang dilakukan pada hari sebelumnya, dan yang akan dikerjakan selama hari ini. Selama *sprint* berjalan, akan dikoordinasi oleh *scrum master*. Koordinasi tugas dilakukan melalui Taiga Project Management. Untuk komunikasi baik secara personal, *team project*, atau keseluruhan karyawan digunakan Discord dan Slack. Sedangkan untuk pertemuan atau diskusi yang memerlukan *share screen,* digunakan Google Meet.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama pelaksanaan kerja magang di PT Sumber Inovasi Informatika yaitu mengembangkan sistem presensi berbasis *website*. Untuk tampilan menggunakan *framework* ReactJS dan dengan komponen SemanticUI. *Logic frontend* dibangun menggunakan ReactJS, sedangkan untuk *backend* menggunakan *framework* Frappe yang berbasis Python.

Selain itu, tanggung jawab yang diberikan selama kegiatan kerja magang adalah sebagai berikut.

1. Mempelajari *framework* ReactJS, komponen Semantic, serta *request* ke *backend*
2. Ikut serta dalam *sprint planning* dan demo aplikasi setiap dua minggu sekali
3. Mengimplementasikan ReactJS pada *frontend website* TAP
4. Membuat tampilan *website* sesuai dengan *mockup* yang disediakan bagian *design* dengan mengimplementasikan komponen Semantic
5. Melakukan *request (PUSH, GET)* informasi dari *backend* untuk ditampilkan

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Aktivitas Kerja Magang

Kerja magang dilasanakan selama empat belas minggu dengan *timeline* kerja tercantum pada Tabel 3.1

Tabel 3. 1 Uraian Pelaksanaan Kerja Magang

Minggu	Uraian Pekerjaan
1 – 2	• Mempelajari *framework* ReactJS • Instalasi NodeJS, npm, dan *tools* lainnya • Melakukan *setup account* • Mempelajari *source code project* yang sudah ada • Melakukan *bug fix*
3 – 4	• Mengikuti *Sprint Planning* • Membuat komponen UI yang dibutuhkan • Membuat tampilan Attendance berdasarkan waktu saat ini • Menyusun *automated test* • Melakukan *bug fix* • Mencari *sprint issue* untuk dibenarkan
5 – 6	• Mengikuti *Sprint Planning* • Mempelajari cara melakukan *request* • Memperbarui tampilan menjadi tampilan terbaru sesuai *mockup* • Menyusun *automated test*
7 – 8	• Mengikuti *Sprint Planning* • Migrasi *request* menggunakan Axios • Membuat fitur Take a break • Membuat halaman Leave Application (Daftar pengajuan cuti)

Tabel 3. 1 Uraian Pelaksanaan Kerja Magang (Lanjutan)

	• Membuat form *Leave Application* • Membuat *request (GET, PUSH)* ke *backend*
9 – 10	• Mengikuti *Sprint Planning* • Membuat *request (GET, PUSH)* ke *backend* • Membuat struktur data JSON • Menampilkan informasi yang diperoleh dari hasil *request*
11 – 12	• Mengikuti *Sprint Planning* • Membuat struktur data JSON • Memperbarui tampilan sesuai dengan *mockup* • Membuat fitur *Timesheet* • Melakukan *request* ke *backend (GET, PUSH)*
13 – 14	• Mengikuti *Sprint Planning* • Membuat fitur *Timesheet* • Melakukan *request* ke *backend (GET, PUSH)* • Menerapkan sistem poin baru • Menambahkan *Achievement* baru

Pada minggu pertama dan kedua, penulis melakukan instalasi *tools* dan aplikasi yang dibutuhkan, yaitu Slack, Gitlab, NodeJS, OwnCloud, WebStorm, Postman, dan Zohomail. Selain itu, penulis juga melakukan *setup account* yang dibutuhkan, seperti Zoho, Taiga, dan *account* perusahaan. Pada *sprint* ini penulis

mulai mempelajari *framework* dan *environment* yang ada, penulis juga mulai mengerjakan *bug fix*.

Pada minggu ketiga dan keempat, penulis mulai mengikuti *sprint planning*. Penulis juga membuat komponen UI yang dibutuhkan, seperti indikator status *employee*, menampilkan Attendance Page sesuai dengan waktu saat halaman dibuka, membuat *modal*, dan memperbaiki beberapa *bug* yang dijumpai.

Pada minggu kelima dan keenam, penulis berfokus untuk mengubah tampilan *website* menjadi tampilan yang terbaru yang telah dibuat oleh bagian *design* dan menyambungkannya dengan setiap fungsionalitas yang ada. Penulis juga berpartisipasi dalam *sprint planning.* Pada *sprint* ini juga dilakukan migrasi *request* dari *fetch* seperti biasa menjadi menggunakan Axios. Dengan menggunakan Axios, maka *request* menjadi lebih mudah dipakai dan jauh lebih singkat.

Pada minggu ketujuh dan kedelapan, penulis mulai membuat 2 fitur baru, yaitu Take a Break yang dapat digunakan saat *employee* beristirahat (makan siang atau keperluan lain). Fitur kedua yaitu Leave Application yang berguna untuk mengajukan cuti. Penulis juga mulai menyusun *request* baru terhadap *endpoint* yang disediakan, sehingga diperoleh informasi untuk ditampilkan pada UI. Penulis juga mulai menyediakan struktur data sehingga *backend* dapat menyediakan *endpoint* yang sesuai.

Minggu kesembilan dan kesepuluh, penulis melanjutkan fitur Leave Application, di mana pada *sprint* ini pula fitur Leave Application sudah bisa menyimpan dan mengambil data sesungguhnya. Penulis juga mulai membuat

struktur data JSON untuk fitur selanjutnya. Selain data *leave*, ditambahkan juga tampilan sisa cuti dari *employee*, serta status dari pengajuan.

Minggu kesebelas dan kedua belas, penulis menambahkan fitur baru, yaitu Timesheet. Fitur ini berfungsi untuk *employee* melaporkan apa yang dikerjakan setiap harinya. Setiap *task* yang dikerjakan harus dilaporkan melalui aplikasi, beserta nama *project* dan durasinya. Tampilan *website* juga mengalami penambahan sesuai dengan *mockup* yang ada.

Pada minggu ketiga belas dan keempat belas, penulis melanjutkan fitur Timesheet hingga rampung. Selain itu, karena pada awalnya sistem presensi ini dibangun berdasarkan gamifikasi, maka terjadi penyesuaian terhadap sistem poin, sehingga fitur yang baru juga turut menjadi faktor pemberian poin. Misalnya, jika *employee* rutin mengisi Timesheet dalam satu minggu, maka akan mendapat tambahan poin. Selain itu, ditambahkan juga beberapa *achievement* baru.

Selain dari tugas di atas, selama kegiatan magang berlangsung, penulis juga membuat *Automated Test.* Tujuannya agar *code* yang dihasilkan dapat dites terlebih dahulu secara otomatis, sehingga mempersingkat waktu *review*.

3.3.2 Framework dan Alur Kerja yang Digunakan

Timesheet and Attendance dibangun menggunakan *framework* ReactJS sebagai *frontend*-nya. Untuk basis bahasa pemrogramannya menggunakan Typescript. Sedangkan untuk *framework user interface* digunakan SemanticUI yang selanjutnya dikostumisasi melalui CSS. Pengambilan dan penyimpanan data digunakan menggunakan Axios terhadap *endpoint* yang ada.

Alur kerja yang diterapkan selama kerja magang ini yaitu menggunakan metode Agile, yang terdiri dari *sprint planning*, *showcase, retrospective, dan standup. Standup* dilakukan setiap hari, di mana setiap anggota membahas apa yang dikerjakan pada hari sebelumnya dan yang akan dikerjakan hari ini. Sedangkan untuk sisanya dilakukan setiap dua minggu sekali, yaitu pada hari Selasa. *Showcase* berfungsi untuk menampilkan dan menjelaskan kepada seluruh anggota tim mengenai apa saja yang telah dikerjakan selama satu *sprint.* Lalu dilanjut dengan *retrospective* yang bertujuan untuk membahas segala hal yang terjadi selama satu *sprint*, dapat berupa hal-hal yang mendukung, menghambat, ataupun sekedar melakukan klarifikasi terhadap suatu isu. Sedangkan *sprint planning* berguna untuk menentukan *task* apa saja yang akan dikerjakan pada *sprint* selanjutnya, beserta dengan melakukan estimasi poin dari setiap *task* berdasarkan tingkat kompleksitas dan kerumitannya.

3.3.3 Perancangan Sistem

Perancangan sistem ditampilkan dalam bentuk *user requirement, entitiy relationship diagram* dan *flowchart*. Selanjutnya diimplementasikan dan ditampilkan dalam bentuk *screenshot* aplikasi.

A. User Requirement

User requirement merupakan ekspetasi yang diharapkan saat aplikasi berjalan. *User requirement* dibuat oleh *product owner* dan didiskusikan saat *sprint planning. Requirement* biasanya disusun untuk setiap *task* dan mungkin terjadi perubahan selama masa *development*. Pada bagian ini, penulis merangkum setiap

user requirement yang ada menjadi per halaman dan mengikuti *requirement* yang paling baru dan yang digunakan dalam applikasi.

A.1 Halaman Home

Berikut ini adalah *requirement* untuk halaman Home:

1. Menampilkan informasi *user* berupa nama, avatar sesuai inisial nama, *role*, dan *icon* sesuai dengan *role*
2. Menampilkan hari dan tanggal hari ini
3. Menampilkan salam untuk *user* sesuai dengan waktu saat ini, yaitu:
    a. 06.00 – 10.59 : "Good morning"
    b. 11.00 – 14.59 : "Good day"
    c. 15.00 – 16.59 : "Good afternoon"
    d. 17.00 – 18.59 : "Good evening"
    e. 19.00 – 23.59 : "Good night"
    f. 00.00 - 05.59 : "What's up"
4. Menampilkan tiga *icon menu* yaitu Attendance, Timesheet, dan Leave Application yang jika diklik akan membawa *user* ke halaman bersangkutan
5. Memiliki tombol *easy* yang dapat diklik oleh *user* untuk melakukan *check in* dan *check out*. Tombol *easy check* in apabila user belum melakukan *check* in atau tombol *easy check* out apabila user telah melakukan *check* in, namun *attendance* belum ter-*submit.* Apabila *attendance* pada hari itu telah ter-*submit*, maka tombol *easy* tidak ditampilkan
6. Menampilkan *timelogs* yang telah di-*submit* pada hari ini, serta tombol "+" untuk membuka *modal* Timesheet dan mengirimkan tanggal hari ini sebagai tanggal aktif

A.2 Halaman Attendance

Berikut ini adalah *requirement* untuk halaman Attendance:

1. Menampilkan *background* dan *style* sesuai dengan waktu saat ini, yaitu:
    a. 04.00 – 05.59 : *dawn style*
    b. 06.00 – 10.59 : *morning style*
    c. 11.00 – 13.59 : *noon style*
    d. 14.00 – 15.59 : *afternoon style*
    e. 16.00 – 18.59 : *evening style*
    f. 19.00 – 03.59 : *night style*
2. Menampilkan jam saat ini
3. Menampilkan tombol *check in* jika *user* belum melakukan *check in* pada hari tersebut, atau menampilkan tombol *check out* apabila *user* telah melakukan *check in* dan *attendance* belum ter-*submit*. Saat tombol diklik, maka akan menjalankan *action* sesuai dengan tombolnya. Apabila *attendance* telah ter-*submit*, maka tidak ada tombol yang ditampilkan
4. Menampilkan tombol *take a break* apabila *user* telah melakukan *check in* dan *attendance* belum ter-*submit*. Apabila diklik *user* akan dalam kondisi *break* dan tombol berganti menjadi *back to work*. Saat tombol *back to work* diklik, *user* akan kembali ke kondisi *online.*
5. Menampilkan tombol *other date* untuk beralih ke halaman Other Date.

A.3 Halaman Leaderboard

Berikut ini adalah *requirement* untuk halaman Leaderboard:

1. Menampilkan daftar *employee* dan poinnya, berurutan mulai dari *employee* dengan poin yang paling tinggi ke rendah
2. Menampilkan indikator status *employee*, ditandai warna sesuai dengan status *employee* saat ini, antara lain:
    a. *Online* : hijau
    b. *Offline* : abu-abu
    c. *Break* : kuning

A.4 Halaman Timesheet

Berikut ini adalah *requirement* untuk halaman Timesheet:

1. Menampilkan tulisan hari dan tanggal yang sedang aktif (untuk *default*-nya tanggal hari ini)
2. Terdapat tombol panah di kanan dan kiri tulisan hari, tanggal aktif. Apabila diklik akan beralih ke tanggal sebelum (tombol kiri) atau tanggal sesudahnya (tombol kanan)
3. Menampilkan *list timelogs* yang di-*submit* pada tanggal yang sedang aktif
4. Menampilkan tombol "+" yang apabila diklik membuka *modal* Timesheet dan mengirimkan tanggal sesuai dengan tanggal aktif

A.5 Modal Timesheet

Berikut ini adalah *requirement* untuk *modal* Timesheet:

1. *Modal* akan terbuka, apabila *user* menekan tombol "+" baik di halaman Home atau halaman Timesheet
2. Terdapat tombol "x" yang apabila diklik maka akan menutup *modal* Timesheet
3. Menampilkan hari dan tanggal yang diterima dari halaman pemanggil
4. *User* dapat memilih *project name, activity,* serta *duration* dalam satuan jam dan menit
5. Terdapat tombol *submit* yang jika diklik akan melakukan *action submit timelogs* dan menutup *modal*

A.6 Halaman Leave Application

Berikut ini adalah *requirement* untuk halaman Leave Application:

1. Menampilkan sisa jatah cuti dalam bentuk *icon* hati penuh dan jatah cuti yang sudah terpakai dalam bentuk *icon* hati kosong apabila *user* merupakan *employee full timer*. Apabila tidak, maka ditampilkan tulisan "*You have not been allocated annual leaves*"
2. Menampilkan daftar *leave* yang telah diajukan, yang berisi jenis *leave*, tanggal mulai *leave*, serta durasinya (dalam hitungan hari). Setiap *leave* ditampilkan dalam bentuk *card* dengan warna dan *icon* sesuai dengan statusnya, antara lain:
   a. *Approved* : Hijau, *icon checklist*

b. *Rejected* : Merah, *icon* silang

c. *Pending* : Kuning, *icon* jam pasir

3. Terdapat dua tombol untuk membuka *modal* Leave Application sesuai dengan jenisnya, yaitu *sick* dan *casual*

A.6 Modal Leave Application

Berikut ini adalah *requirement* untuk *modal* Leave Application:

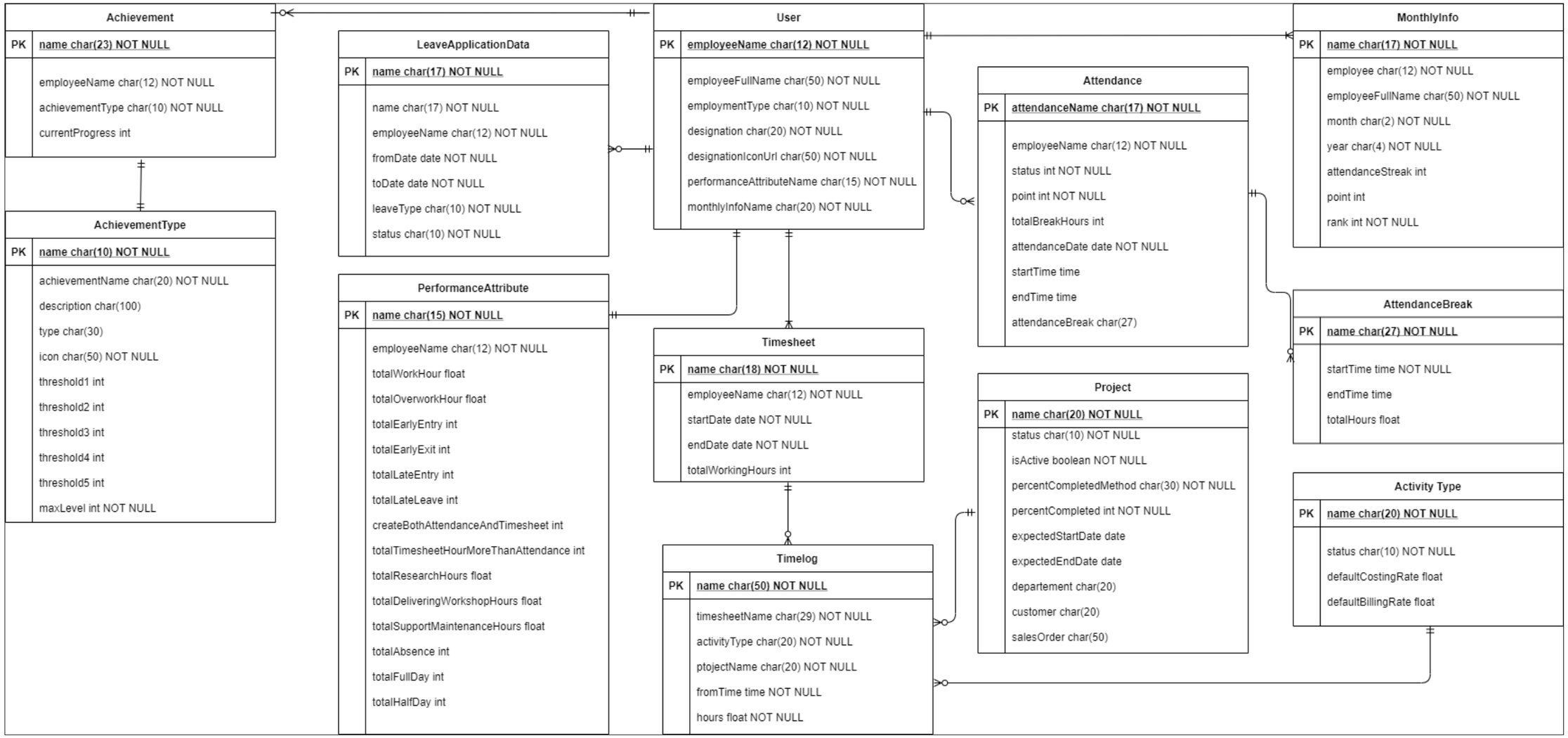
1. Menampilkan ilustrasi sesuai dengan jenis *leave*
2. Memiliki tombol "x" yang apabila ditekan akan menutup *modal*
3. Menampilkan tipe *leave*
4. *User* dapat memilih tanggal mulai dan berakhir *leave*
5. Memiliki tombol *submit* yang jika diklik akan melakukan *submit leave* dan menutup *modal*

A.6 Success Modal

Berikut ini adalah *requirement* untuk *modal* Success:

1. *Modal* akan muncul apabila dipanggil oleh *event* tertentu
2. Memiliki tombol "x" yang apabila ditekan akan menutup *modal*
3. Menampilkan judul *success* serta *detail*-nya

B. Entity Relationship Diagram



Gambar 3. 1 Entitiy Relationship Diagram Timesheet and Attendance

Entitas pada aplikasi ini terbagi menjadi *user*, *attendance, attendance break, timesheet, leave application, dan monthly info.* Sedangkan *timelog* merupakan bagian dari *timesheet* yang juga memiliki referensi terhadap *activity* dan *project.* Hubungan setiap entitas secara lengkap yang ada di *project* Timesheet and Attendance dapat dilihat seperti dalam Gambar 3.1.

C. Implementasi

Implementasi disajikan dalam bentuk *flowchart* dan tampilan aplikasi. *Flowchart* menampilkan alur kerja fitur serta proses yang ada dalam pengambilan maupun penyimpanan data ke *database*. Tampilan dari Timesheet and Attendance dibuat berdasarkan *mockup* yang ada. *Mockup* telah disediakan oleh tim *design* dan dibuat menggunakan Adobe XD. Meskipun berbasis *website*, tampilan aplikasi dirancang untuk ukuran *handphone*, karena ke depannya akan dikembangkan menjadi aplikasi *mobile* juga, serta mempertimbangkan mayoritas interaksi terhadap aplikasi akan terjadi melalui *handphone* karyawan.

C.1 Halaman Home

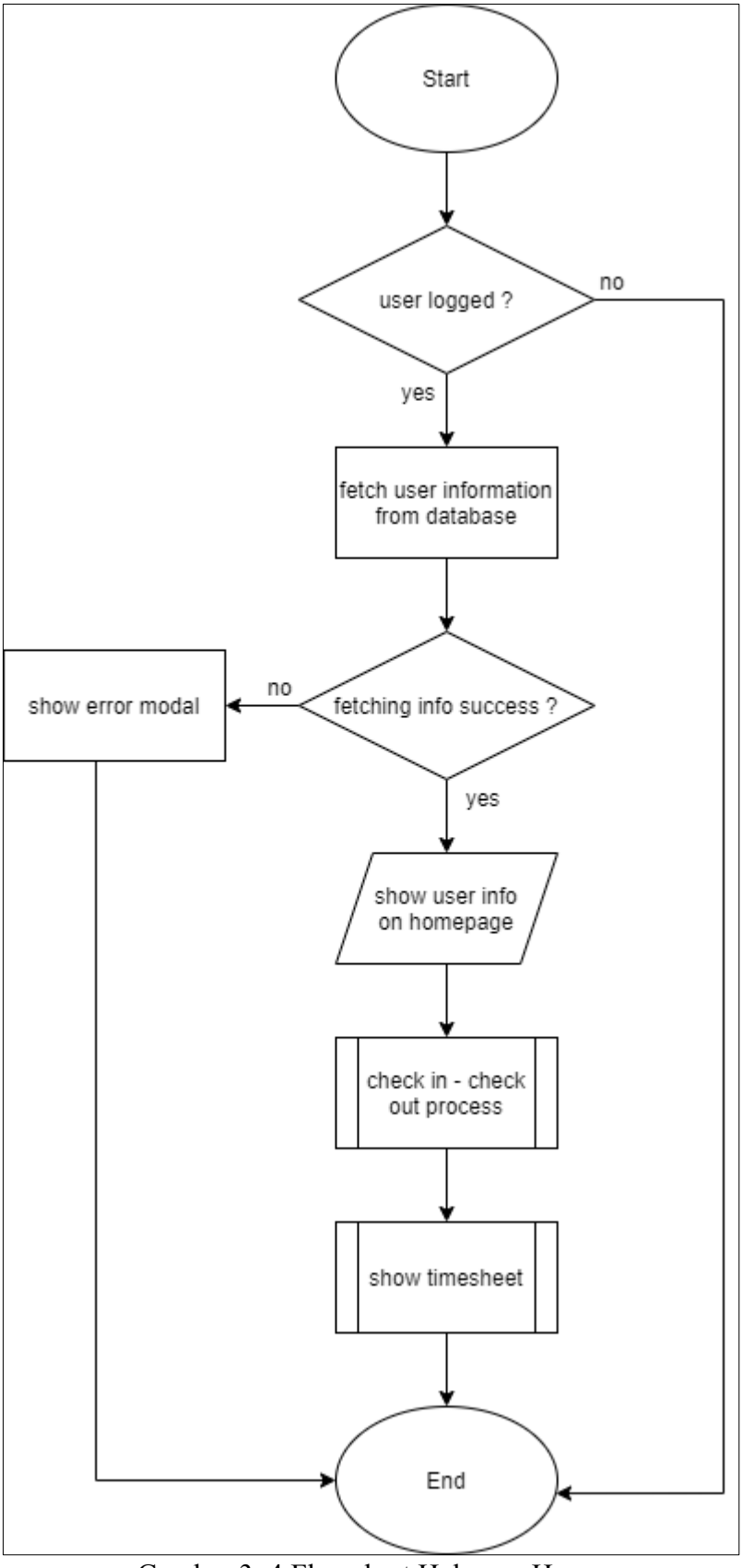
Halaman Home menampilkan informasi umum mengenai karyawan, seperti nama lengkap, jabatan, serta *timelog* yang sudah di-*submit* pada hari ini. Halaman Home didesain dengan tombol dan *icon* yang akan membawa *user* ke halaman lainnya. Tersedia juga tombol *easy* untuk *check in* dan *check out*, sehingga menaikkan *user experience* karena cukup dengan satu klik, maka fitur paling dasar dari aplikasi ini, yaitu presensi, dapat dikerjakan. Tidak hanya kegiatan presensi, fitur Timesheet yang berguna untuk melaporkan apa yang dikerjakan *user* setiap

harinya juga bisa langsung dikerjakan dari halaman ini. Dengan mengklik *icon* "+" di sebelah list timelog, maka akan terbuka *modal* Timesheet. Penulis juga bertugas untuk memperbarui tampilan Home serta menambahkan komponen-komponen baru. Berikut adalah tampilan dari Home sebelum dan sesudah terjadi perubahan.

Gambar 3. 2 Halaman Home sebelum perubahan

Gambar 3. 3 Halaman Home sesudah perubahan

Untuk proses yang terjadi pada halaman Home diuraikan seperti pada Gambar 3.4. Dimulai dari pengecekan apakah *user* sudah *login* dan valid. Jika benar, maka proses dilanjutkan, jika tidak maka proses berhenti. Selanjutnya, dari data autentikasi, dilakukan pengambilan informasi terkait *user* dari *database*. Apabila berhasil, maka informasi ditampilkan, jika gagal maka akan muncul *modal error*. Selanjutnya untuk proses *check in* dan *check out* serta Timesheet akan dijelaskan dalam modul terpisah yaitu pada subbab C.3 dan C.5.



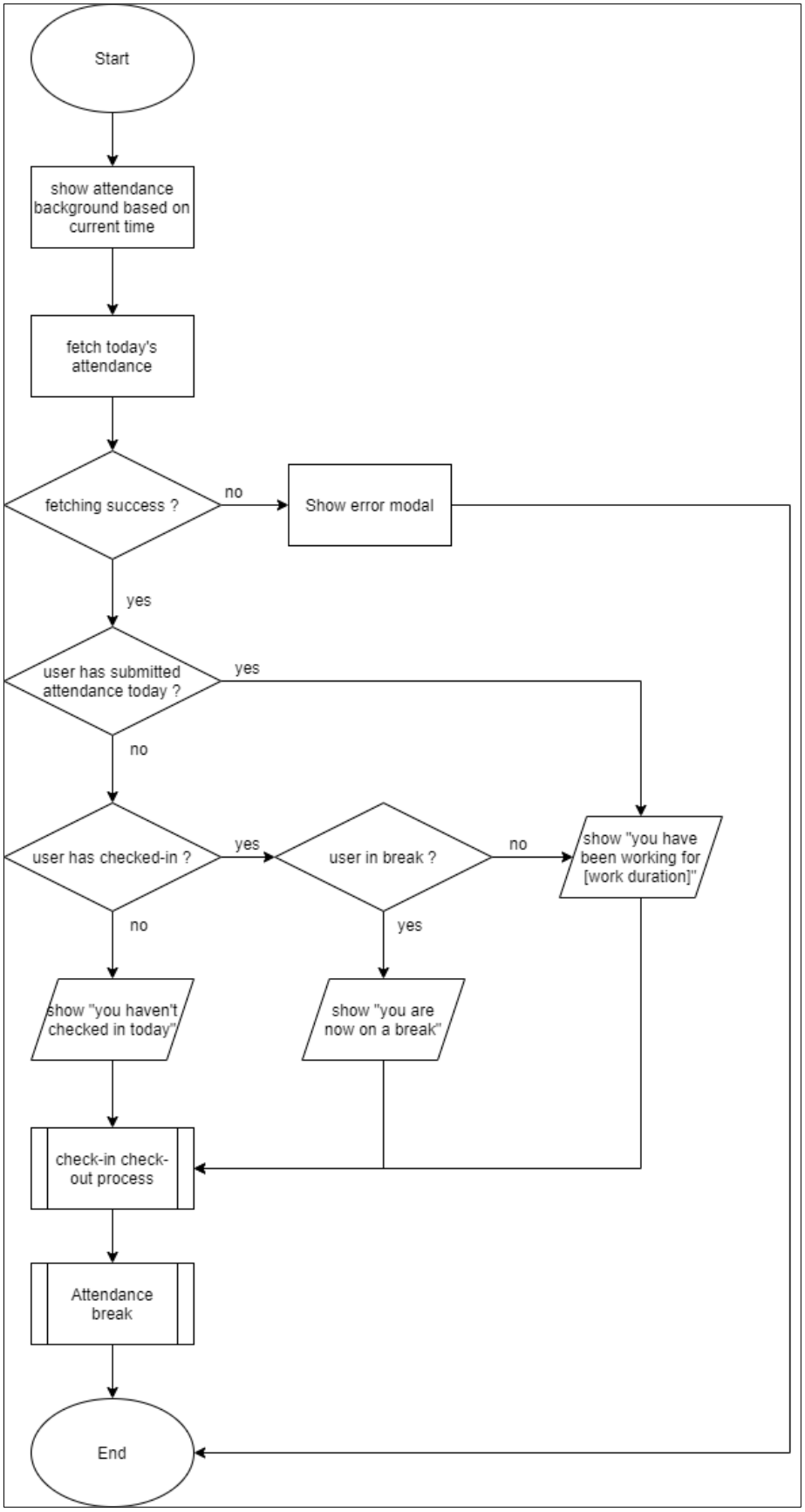
Gambar 3. 4 Flowchart Halaman Home

C.2 Halaman Attendance

Halaman Attendance menampilkan jam saat ini dan status presensi. Apabila *user* belum melakukan absen masuk pada hari ini, maka akan ditampilkan tulisan "*You haven't checked in today*", apabila *user* sudah *check in* dan sedang bekerja, atau bahkan sudah *check out* maka akan ditampilkan durasi kerja *user.* Sedangkan apabila *user* dalam posisi sedang *break* akan ditampilkan tulisan "*You are now on a break*". Terdapat juga 2 tombol yang berfungsi untuk *check in - check out* atau untuk *take a break*, serta 1 tombol yang jika diklik menuju ke halaman Other Date Page yang berfungsi untuk mengisi absen pada tanggal lainnya. *Background* dari halaman ini juga akan berubah-ubah sesuai dengan waktu saat dibuka, serta memiliki animasi. Untuk animasi, serta komponen *background* dikerjakan oleh tim desain, sedangkan penulis hanya mengatur *logic*-nya saja. Berikut ini adalah tampilan halaman Attendance.

Gambar 3. 5 Halaman Attendance Berdasarkan Waktu

Untuk proses yang terjadi pada halaman ini dijabarkan pada Gambar 3.6, yaitu mengambil informasi *attendance* dari *user.* Lalu menampilkan pesan dan tombol sesuai dengan status *attendance*-nya. Sedangkan untuk proses *attendance break* dan *check in – check out* akan dijelaskan pada subbab C.3 dan C.4.

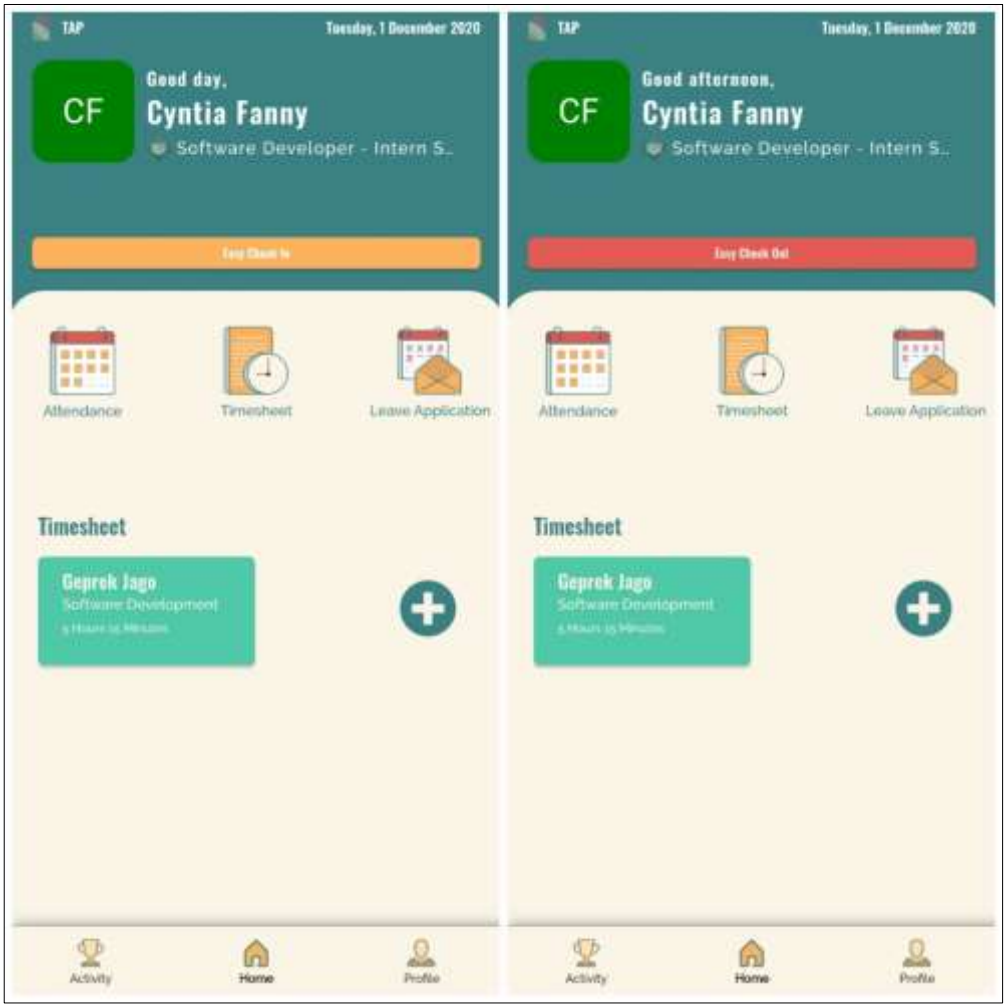


Gambar 3. 6 Flowchart Halaman Attendance

C.3 Check In dan Check out

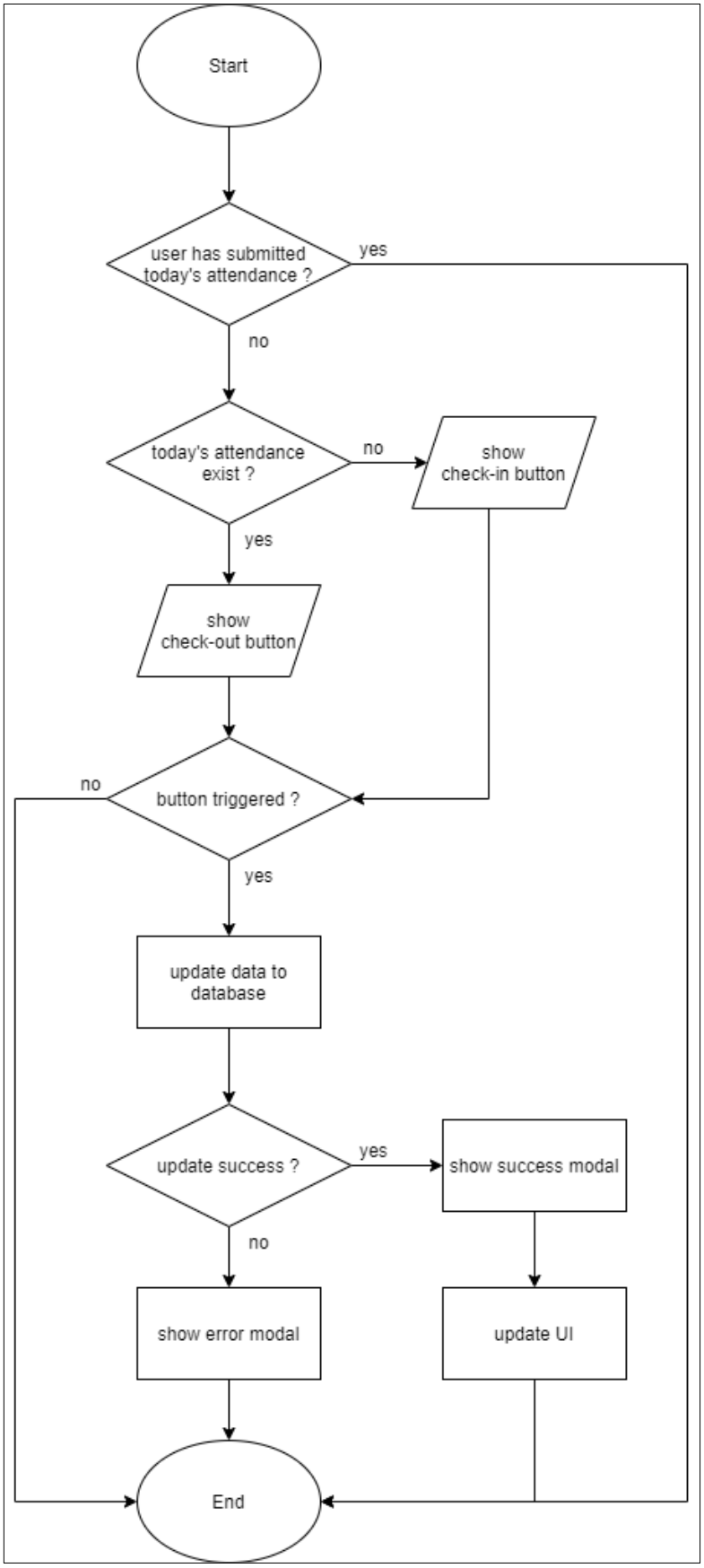
Pada sistem presensi TAP, fitur paling utama adalah untuk mencatat presensi atau kehadiran karyawan, beserta informasi seperti waktunya. Presensi dapat dilakukan dengan *check in* pada saat memulai pekerjaan dan melakukan *check out* saat selesai bekerja, yang sekaligus akan men-*submit attendance* pada hari itu. Fitur ini dapat diakses menggunakan tombol yang terdapat di halaman Attendance dan tombol *easy* di halaman Home. Tombol akan memiliki warna dan tulisan sesuai dengan status karyawan saat itu. Apabila karyawan belum *check in*, maka tombolnya akan berwarna kuning dan bertuliskan "*check in*". Saat tombol *check in* diklik, maka *attendance* hari itu akan terbuat. Sedangkan jika sudah ada *attendance* hari tesrsebut, tombol akan berwarna merah bertuliskan "*check out"*. Apabila karyawan telah selesai bekerja, dan *attendance* ter-*submit* maka tombol tersebut akan menghilang dari tampilan. Tampilan tersebut dapat dilihat pada Gambar 3.7 dan Gambar 3.8.

Gambar 3. 7 Tombol Check In dan Check Out pada Halaman Attendance



Gambar 3. 8 Tombol Check In dan Check Out pada Halaman Home

Proses *check in* dan *check out* digambarkan melalui *flowchart* pada Gambar 3.9. Saat *attendance* terbuat dan ter-*submit* terjadi proses penyimpanan data ke *database*. Apabila berhasil maka *success modal* akan ditampilkan, sebaliknya apabila terjadi kegagalan, maka *error modal* akan ditampilkan.



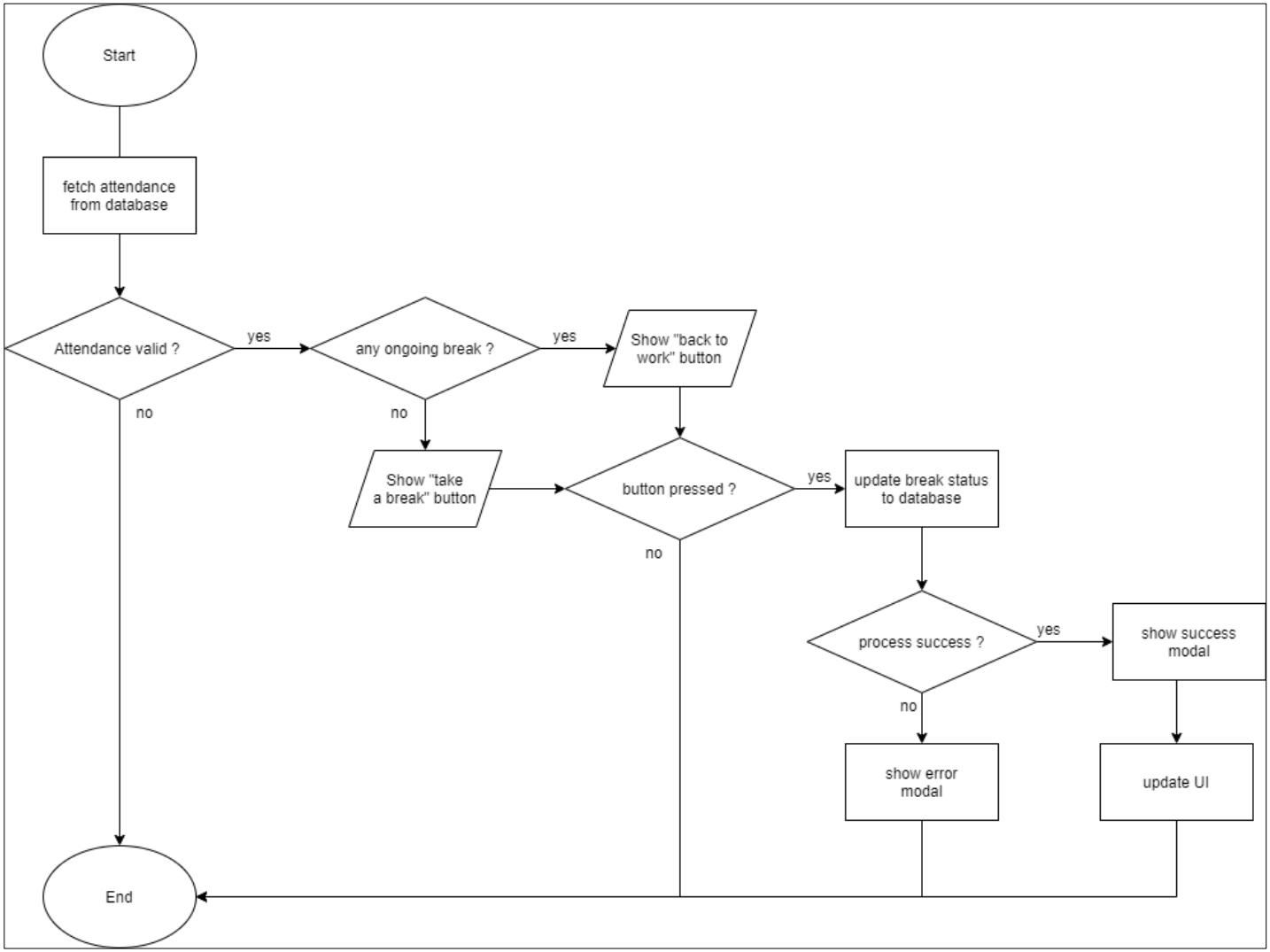
Gambar 3. 9 Flowchart Check In -Check Out Process

C.4 Attendance Break

Attendance Break merupakan fitur yang berguna bagi karyawan untuk beristirahat selama jam kerja. Tidak ada batasan dalam mengambil *break*, hanya saja semuanya aakan tercatat dan turut mengurangi durasi kerja. Attendance Break dapat dilakukan melalui tombol yang tersedia pada halaman Attendance. Tombol tersebut hanya akan muncul jika *user* dalam posisi sudah *check in* namun belum *check out*. Tombol akan berubah tulisannya sesuai dengan status *user* pada saat tersebut. Apabila *user* sedang tidak *break*, maka tulisannya "*Take a break*", sebaliknya apabila *user* sudah dalam kondisi *break,* maka tulisannya "*Back to Work*". Tampilan tersebut dapat dilihat pada Gambar 3.10.

Gambar 3. 10 Tombol Break pada Halaman Attendance

Proses *break* terjadi dimulai dari mengambil informasi *attendance* dari *database*, apabila terdapat *attendance* yang valid, baru akan dicek apakah memiliki *child attendance break*. Apabila ada, baru dari setiap *break* kembali dicek apakah ada yang *duration*-nya masih 0. Apabila ada, maka status *user* saat itu dalam kondisi *break.* Sebaliknya, jika tidak ada, maka *user* tidak dalam kondisi *break.* Proses secara rinci dapat dilihat pada Gambar 3.11.



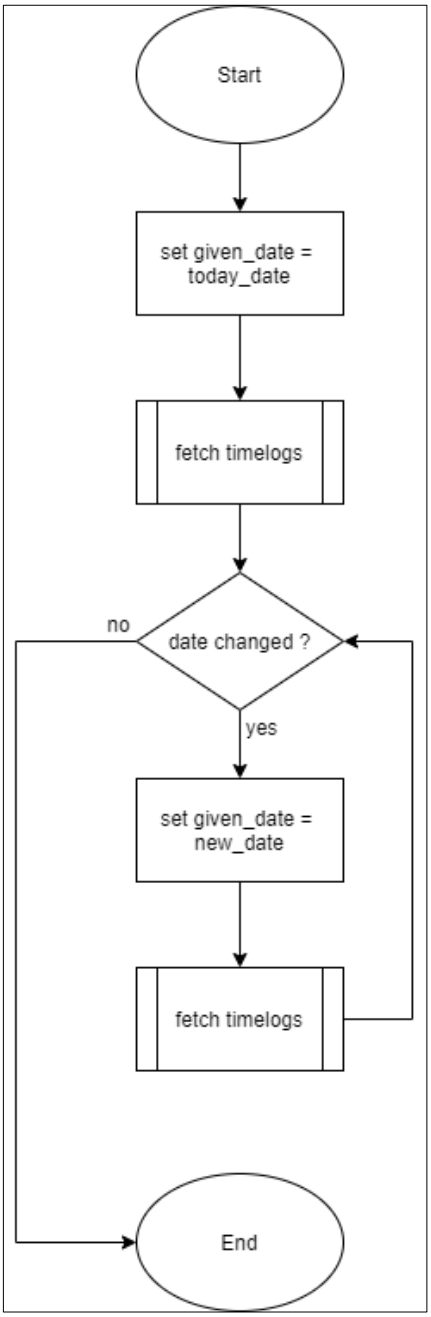
Gambar 3. 11 Flowchart Attendance Break

C.5 Halaman Timesheet

Halaman Timesheet menampilkan list pekerjaan (atau disebut sebagai *timelogs)* yang dilakukan oleh karyawan dalam satu hari. Pekerjaan tersebut direkap setiap harinya, dan juga dicantumkan *project* apa yang sedang dikerjakan, jenis aktivitasnya, serta durasinya. Pada saat halaman dibuka, maka akan ditunjukkan tanggal hari ini, namun *user* dapat mengklik tombol panah di sebelah kanan dan kiri tanggal untuk melihat tanggal lainnya. Terdapat juga tombol "+" di bawah *list* yang berguna untuk membuka Timesheet *modal* untuk membuat *timelog* baru. Tampilan dari halaman Timesheet dapat dilihat pada Gambar 3.12.

Gambar 3. 12 Halaman Timesheet

Untuk proses pengambilan informasi untuk halaman ini dapat dilihat melalui *flowchart* Gambar 3.13. Pada awalnya, tanggal yang digunakan untuk mengambil data *timelog* merupakan tanggal hari ini. Saat *user* mengubah tanggal, maka dilakukan pengambilan lagi dengan tanggal yang sedang dipilih. Untuk proses pengambilan *timelog* dipisah menjadi *function* tersendiri yaitu Timelog yang dapat dilihat pada subbab C.6.



Gambar 3. 13 Flowchart Halaman Timesheet

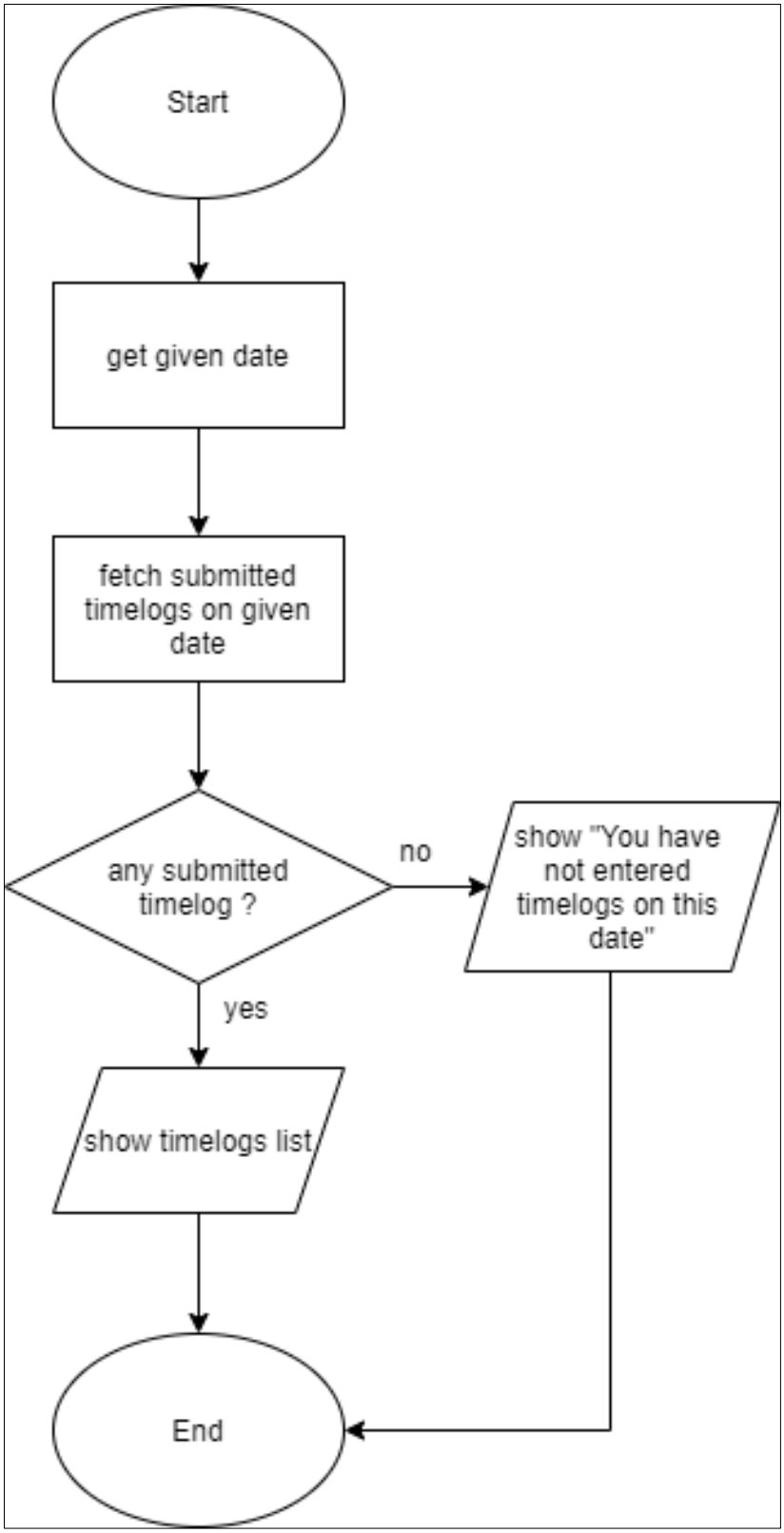
C.6 Timelog

Timelog merupakan satuan *task* yang dilakukan. Kumpulan *timelog* dalam satu hari disimpan dalam sebuah *timesheet*. Satu *timesheet* khusus berlaku hanya untuk 1 *user* dan pada 1 tanggal juga. Timelog ditampilkan pada halaman Timesheet dan halaman Home. Khusus pada halaman *Home,* timelog yang ditampilkan hanya *timelog* yang di-*submit* pada hari tersebut. Tampilan *timelog* pada halaman Home dan halaman Timesheet dapat dilihat pada Gambar 3.14.

Gambar 3. 14 Timelog pada Halaman Timesheet dan Halaman Home

Proses pengambilan *timelog* dapat dilihat pada Gambar 3.15. Dimulai dari memperoleh *given_date* yang dikirimkan dari halaman pemanggilnya. Untuk *timelog* yang ditampilkan di halaman Home, maka *given_date*-nya adalah tanggal

hari ini. Sedangkan untuk *timelog* pada halaman Timesheet diambil sesuai tanggal yang dikirimkan. Apabila tidak ada *timelog* pada tanggal tersebut, maka akan muncul tulisan "*You have not entered timelogs in this date*". Apabila proses berhasil, maka *timelog* akan ditampilkan, namun apabila gagal maka akan memunculkan *error modal.*



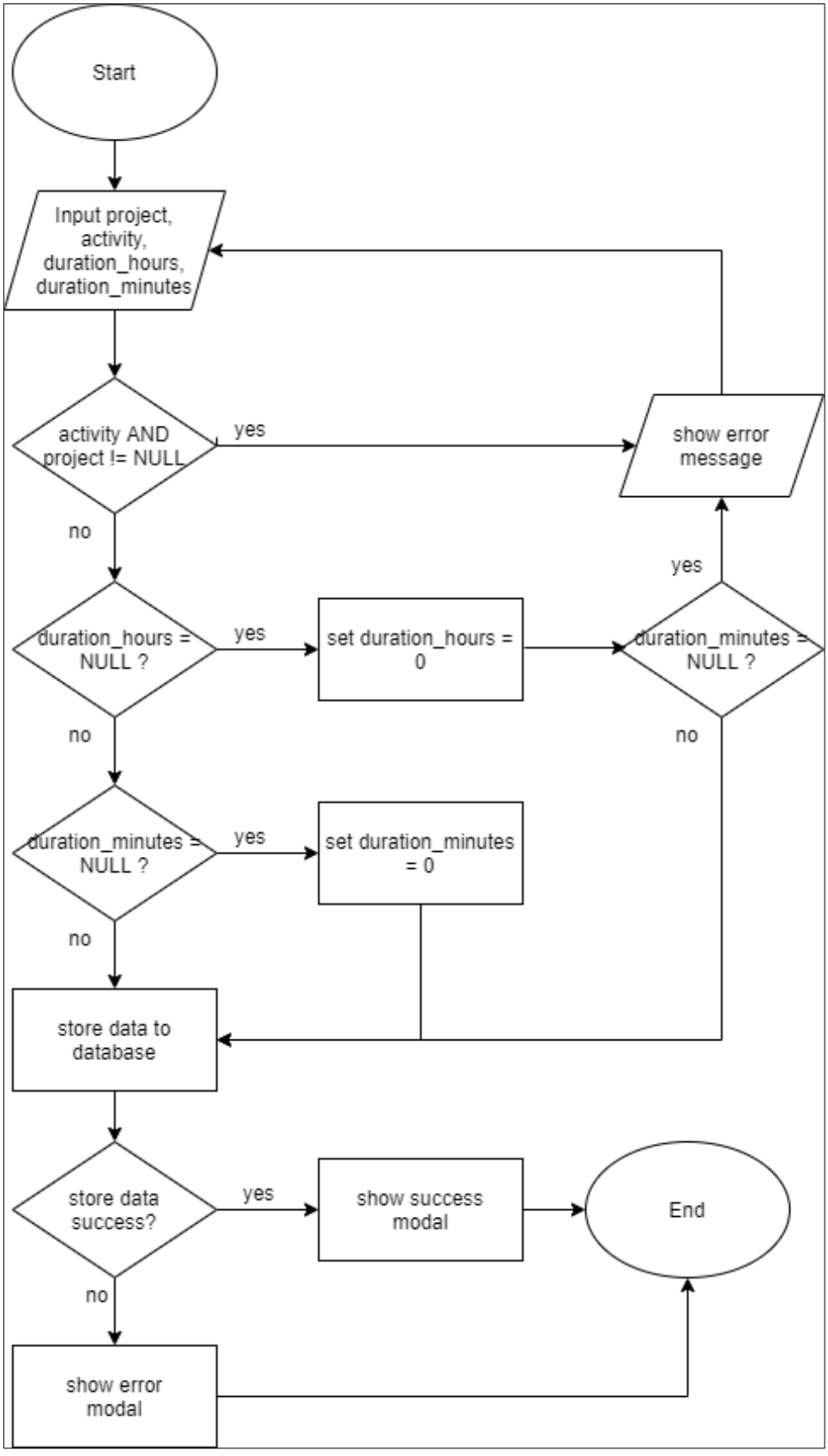
Gambar 3. 15 Flowchart Fetch Timelogs

C.7 Timesheet Modal

Timesheet Modal akan muncul apabila tombol "+" pada halaman Home atau halaman Timesheet diklik, Pada *modal* ini, *user* dapat membuat *timelog* yang berisi *project* apa yang dikerjakan, aktivitasnya, serta durasinya. Untuk tanggal sudah di-*set* sesuai dengan tanggal *user* mengklik tanda "+", sedangkan jika *modal* terbuka dari halaman Home, maka akan digunakan tanggal hari ini. Saat *user* mengklik tombol *Submit*, maka timelog akan tersimpan. Tampilan dari Timesheet Modal dapat dilihat pada Gambar 3.16.

Gambar 3. 16 Tampilan Timesheet Modal

Proses pengisian data dan penyimpanan *timelog* dapat dilihat pada Gambar 3.17. Semua data yang dibutuhkan harus terisi, apabila ada yang kosong, maka akan muncul tulisan warning dan proses penyimpanan tidak dapat dilakukan. Jika semua data valid, barulah *user* dapat men-*submit timelog.* Apabila *timelog* berhasil disimpan, maka akan muncul *success modal* dan data *timelog* yang ditampilkan akan diperbarui. Apabila terjadi kegagalan, maka akan muncul *error modal.*



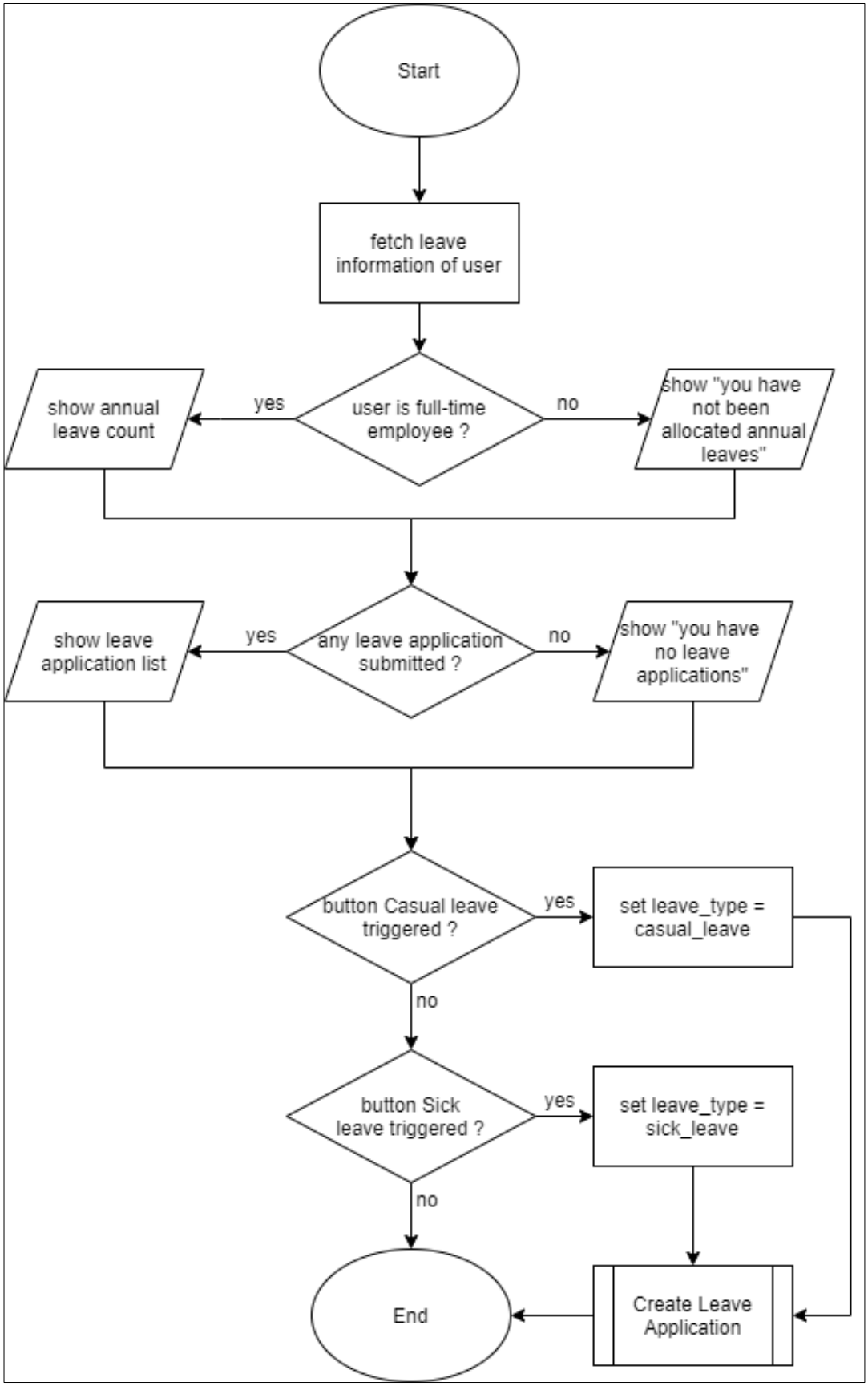
Gambar 3. 17 Flowchart Create Timelog

C.8 Halaman Leave Application

Halaman Leave Application menampilkan jumlah jatah cuti yang tersisa milik *user* serta *list* pengajuan cuti yang sudah *user* buat. Pada halaman ini juga *user* dapat melihat status dari pengajuannya, apakah diterima, ditolak, atau masih *pending*. Apabila *leave* diterima, maka akan berwarna hijau dan memiliki *icon checklist*. Apabila ditolak, maka berwarna merah dan memiliki *icon* silang. Sedangkan apabila masih *pending* maka akan berwarna kuning dengan *icon* jam. Di bagian bawah *list* juga terdapat dua tombol yang jika diklik akan memunculkan *modal* Leave Application dengan *leave type* sesuai dengan tombol yang diklik. Tampilan dari halaman ini dapat dilihat pada Gambar 3.18.

Gambar 3. 18 Tampilan Halaman Leave Application

Untuk proses pengambilan data *leave* dijabarkan pada Gambar 3.19. Proses dimulai dengan pengambilan data *user*. Karena jatah cuti atau *leave count* hanya berlaku untuk *Full-time employee*, maka dilakukan pengecekan terhadap status karyawan. Apabila *Full time*, maka akan ditampilkan sisa jatah (digambarkan oleh hati merah) dan jatah yang sudah terpakai (digambarkan hati abu-abu). Selanjutnya terjadi pengambilan data *leave* dari *database*. Apabila *user* belum pernah mengajukan cuti, maka akan ditampilkan tulisan "*You have no leave applications*", jika sudah pernah maka setiap data *leave* ditampilkan dalam bentuk *card* dengan warna dan *icon* sesuai dengan statusnya. Apabila *button* diklik, maka akan terbuka Leave Application Modal dengan *leave type* yang di-*parsing* dari halaman ini sesuai dengan *button* yang diklik.



Gambar 3. 19 Flowchart Halaman Leave Application

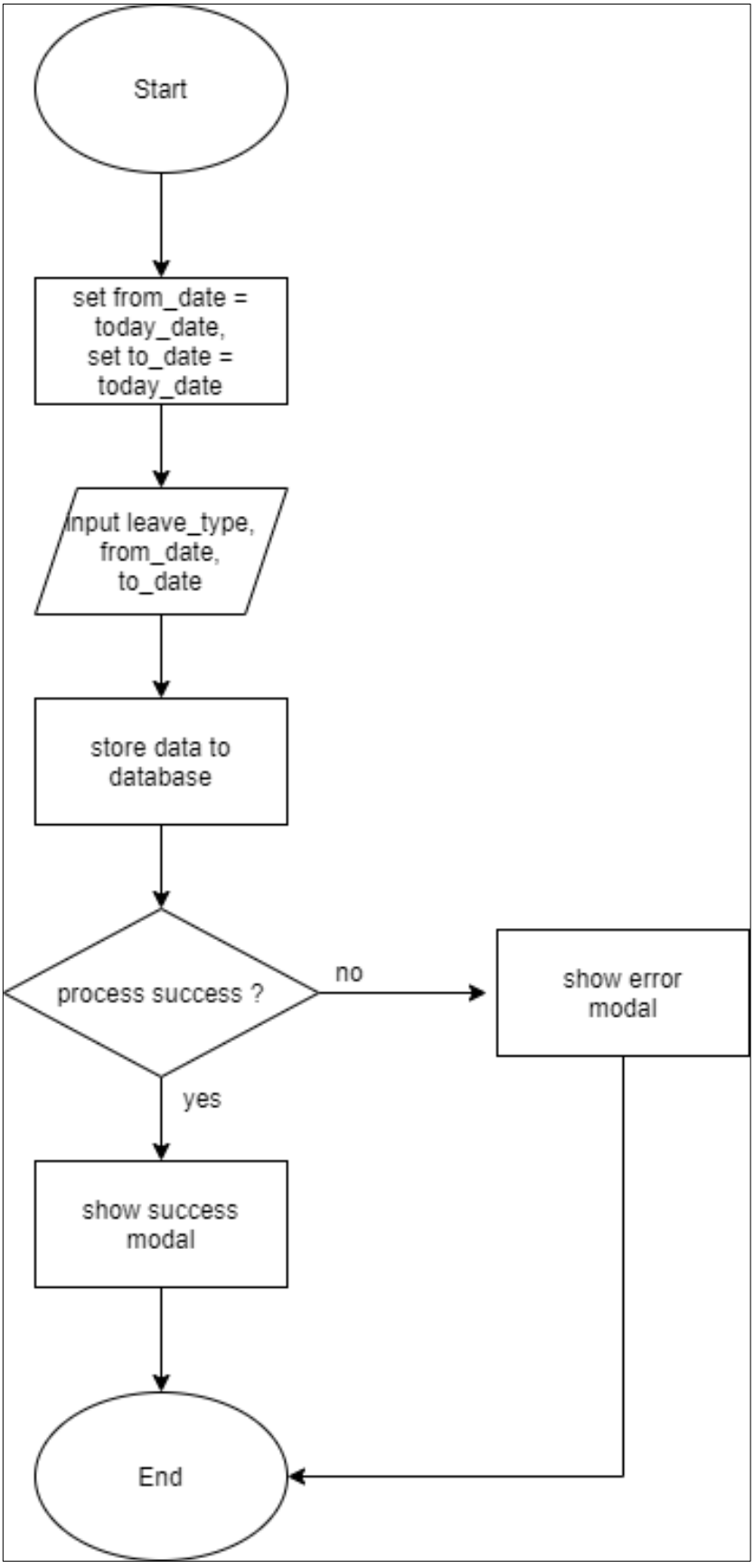
C.9 Leave Application Modal

Leave Application Modal akan muncul apabila tombol pada halaman Leave Application diklik. Modal berisi ilustrasi yang sesuai dengan *leave type* yang dipilih. *User* harus mengisi tanggal mulai dan tanggal berakhir cuti yang diajukan. Secara *default*, kedua tanggal tersebut diisi tanggal hari ini. Tampilan dari *modal* ini dapat dilihat pada Gambar 3.20 dan Gambar 3.21.

Gambar 3. 20 Leave Application Modal (Casual Leave)

Gambar 3. 21 Leave Application Modal (Sick Leave)

Proses yang terjadi pada modal ini dimulai dengan *from date* dan *to date* yang secara *default* diisi tanggal hari ini. Selanjutnya *leave type* yang dimunculkan sesuai dengan *input user* berdasarkan *button* yang diklik dari halaman Leave Application. *User* harus mengisi *from date* dan *to date* dari cuti yang diajukan. Saat *user* men-*submit leave*, maka akan terjadi proses penyimpanan ke *database*. Apabila proses berhasil, maka akan muncul *success modal* dan apabila gagal akan muncul *error modal.* Proses yang terjadi digambarkan melalui Gambar 3.22.



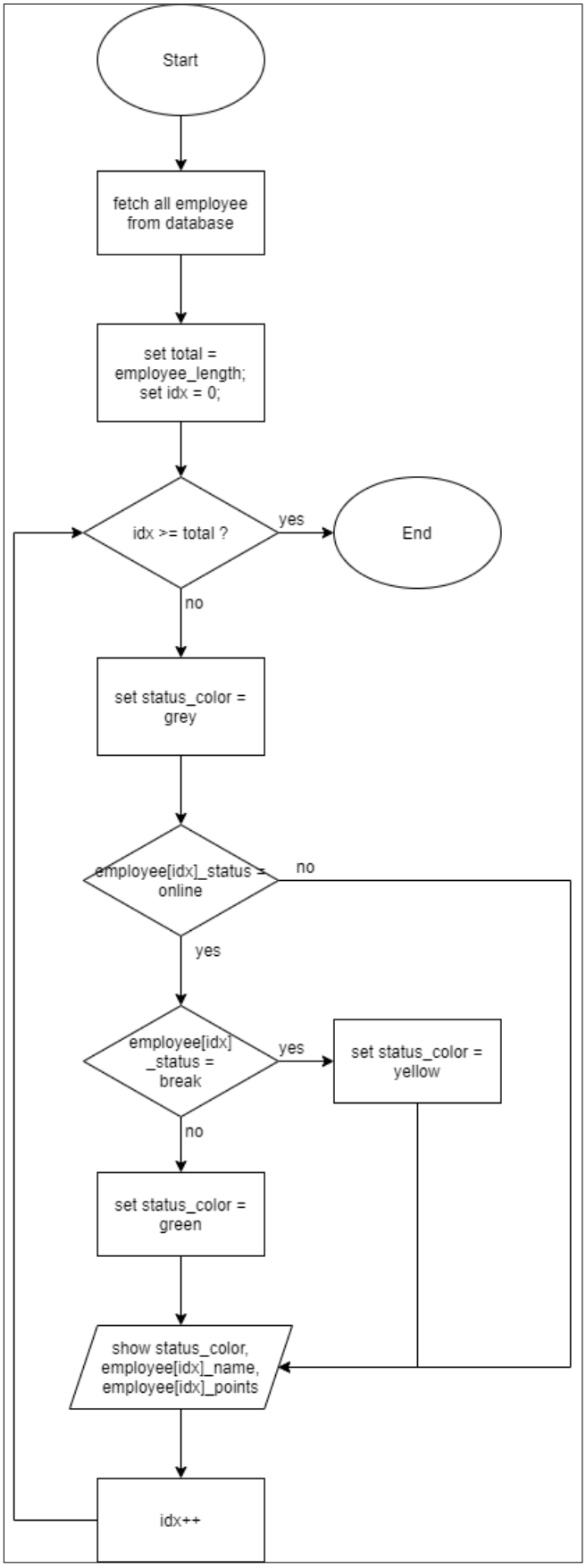
Gambar 3. 22 Flowchart Create Leave Application

C.10 Halaman Leaderboard

Halaman Leaderboard berisi *list* nama *employee* dan *point* yang dimilikinya, serta status ketersediaan dari *employee* tersebut. Adapun *employee* diurutkan berdasarkan jumlah *point*-nya, dari yang paling besar. Apabila *employee* tersebut sedang *online* (dalam posisi *check in*), maka indikator status akan berwarna hijau, apabila sedang *break*, indikator status berwarna kuning, sedangkan apabila sedang *offline* maka indikator status berwarna abu-abu. Tujuan dari adanya indikator status yaitu agar *employee* dapat mengetahui status *employee* lainnya. Tampilan dari halaman ini dapat dilihat pada Gambar 3.23.

Gambar 3. 23 Halaman Leaderboard

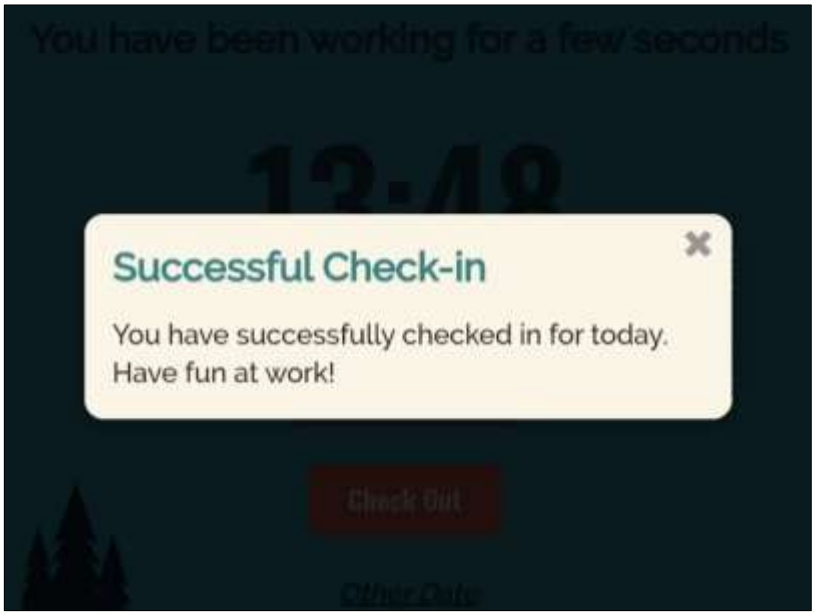
Proses yang terjadi pada halaman ini dimulai dengan pengambilan informasi seperti nama, *point*, dan status dari semua *employee*. List yang dikirim dari database sudah urut berdasarkan *point* dari yang paling tinggi ke rendah. Selanjutnya dari setiap data yang ada, dicek statusnya, dan *class* dari *style* diberi warna sesuai dengan status. Lalu data ditampilkan dalam bentuk baris. Proses ini digambarkan melalui Gambar 3.24.



Gambar 3. 24 Flowchart Halaman Leaderboard

C.11 Success Modal

Success Modal merupakan komponen yang akan muncul apabila suatu proses berhasil dijalankan. Tidak semua proses yang berhasil akan menghasilkan munculnya Success Modal, biasanya terjadi pada proses yang bersifat *user* memberikan inputan. Sedangkan untuk proses *fetching* tidak men-*trigger* munculnya Success Modal. *Title* dan isi dari Success Modal tergantung dari proses apa yang memunculkannya. *State* yang menyatakan apakah Success Modal sedang terbuka atau tidak disimpan dalam *global variable* sehingga dapat diakses di manapun. Tampilan dari Success Modal dapat dilihat pada Gambar 3.25.



Gambar 3. 25 Tampilan Success Modal

3.3.4 Proses Development dan Maintenance

Proses *development* dari aplikasi Timesheet and Attendance sudah berlangsung dari November 2019, dimulai dengan perancangan sistem awal, *literature review,* membuat *business workflow*, menyusun *wireframe* dan *mockup,*

baru masuk ke tahap pengimplementasian dalam bentuk *code*. Total keseluruhan *sprint* yang telah berjalan yaitu 23 *sprint* dengan durasi setiap *sprint*-nya adalah dua minggu. Setiap selesai sebuah *sprint* maka dilakukan *deployment*. *Deployment* dilakukan pada hari Rabu pagi, sebelum jam kerja dimulai. Proses *development* aplikasi ini dilengkapi dengan *version control system*, yaitu pencatatan setiap pembaruan yang terjadi. *Version* yang digunakan terdiri atas 3 angka dengan format vX.X.X, di mana digit pertama mewakili versi perubahan yang bersifat *major*, digit kedua mewakili versi perubahan *minor*, serta versi ketiga mewakili *patch.*

Version untuk aplikasi TAP versi *website* ini dimulai dari v1.0.0. Sedangkan untuk v2.0.0 akan digunakan saat TAP sudah dalam bentuk *mobile application*. Saat memasuki *sprint* baru, maka version akan ditandai dengan penggunaan ".dev" di akhir. Setiap terjadi *deployment*, *version* di-*upgrade* dengan mengubah nilai kedua dari *version* sebelumnya. Hal ini dilakukan karena perubahan yang terjadi setiap proses *deployment* merupakan perubahan minor. Misalnya apabila *sprint* lima terjadi *deployment* v1.2.0, maka *sprint* enam akan dimulai dengan *version* v.1.3.0.dev, dan akan berubah menjadi v.1.3.0 saat dilakukan *deployment* pada akhir *sprint*.

Setelah aplikasi selesai dari tahap *development*, sudah tidak ada *sprint* khusus yang diperuntukkan bagi aplikasi ini dan aplikasi memasuki masa *maintenance*. Namun apabila terdapat *bug* maka akan diselesaikan dan untuk *version*-nya tidak dibuatkan *version* baru lagi. Melainkan hanya ditambah akhiran "b[bug ke berapa]". Contohnya yaitu v.1.9.0 b12.

3.3.5 Kendala yang Ditemukan

Selama pelaksanaan kerja magang, penulis menghadapi beberapa kendala, antara lain sebagai berikut:

1. Belum mempunyai pemahaman terhadap React dan Typescript.
2. Kondisi *Work from Home* cukup menyulitkan dalam komunikasi. Komunikasi hanya dapat dilakukan secara daring dan terkadang terkendala koneksi internet atau *noise* yang ada. Selain itu, dalam *resolve* masalah pun lebih sulit karena hanya melalui daring.
3. Sempat adanya *bottleneck*, dikarenakan *backend* yang memegang lebih dari 1 *project*, sehingga *task* tidak dapat selesai karena *endpoint* yang belum siap.
4. Terjadi perubahan batas terhadap fitur yang ada.
5. Adanya *deadline* yang singkat untuk *project* ini sehingga harus mengejarnya.

3.3.6 Solusi Atas Kendala yang Ditemukan

Agar dapat melancarkan jalannya kerja magang, maka setiap kendala di atas harus memiliki solusi. Berikut adalah solusi yang penulis lakukan

1. Melakukan pembelajaran React dan Typescript dari internet serta diskusi dengan *frontend* lain.
2. Beberapa kali diadakan kerja ke kantor, walaupun tidak dapat full semua *employee*, namun per *project.*

3. Melakukan komunikasi dengan *backend* saat *sprint planning*, dan memperbanyak *task* selain *backend* apabila saat itu *backend* benar-benar sedang sibuk pada *project* lain.
4. Melakukan diskusi dengan *Product Owner* dan *backend* sehingga jelas batasannya dan mana yang lebih mungkin dilakukan.
5. Memaksimalkan hari kerja yang ada serta memebuat beberapa fitur menjadi lebih *simple*.