SKRIPSI

PENGEMBANGAN SISTEM PRESENSI UNTUK *WORK FROM HOME* (WFH) DAN *WORK FROM OFFICE* (WFO) SELAMA PANDEMI COVID-19

Disusun Oleh:

N a m a : Fajar Pratama Purwantoro Putra

NIM : 17523094

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN SISTEM PRESENSI UNTUK *WORK FROM HOME* (WFH) DAN *WORK FROM OFFICE* (WFO) SELAMA PANDEMI COVID-19

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Fajar Pratama Purwantoro Putra

NIM : 17523094

Yogyakarta, 26 Desember 2021

Pembimbing,

(Moh. Idris, S.Kom., M.Kom)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN SISTEM PRESENSI UNTUK *WORK FROM HOME* (WFH) DAN *WORK FROM OFFICE* (WFO) SELAMA PANDEMI COVID-19

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 10 Januari 2022

Tim Penguji

Ketua Penguji

Moh. Idris, S.Kom., M.Kom

Anggota 1

Lizda Iswari, S.T., M.Sc.

Anggota 2

Elyza Gustri Wahyuni, S.T., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Fajar Pratama Purwantoro Putra
NIM	:	17523094

Tugas akhir dengan judul:

PENGEMBANGAN SISTEM PRESENSI UNTUK *WORK FROM HOME* (WFH) DAN *WORK FROM OFFICE* (WFO) SELAMA PANDEMI COVID-19

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 26 Desember 2021

(Fajar Pratama Purwantoro Putra)

HALAMAN PERSEMBAHAN

Puji syukur atas segala rahmat dan karunia yang Allah SWT berikan kepada penulis sehingga penulis dapat menyelesaikan Laporan Tugas Akhir dengan maksimal. Laporan Tugas Akhir ini adalah salah satu kewajiban yang penulis tunaikan dan persembahkan kepada jurusan, diri sendiri, kedua orang tua, dan teman-teman yang selalu memberikan motivasi, dorongan, dan dukungan kepada penulis.

HALAMAN MOTO

"To live is to suffer, to survive is to find some meaning in the suffering"

— Friedrich Nietzsche

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya"

— QS. Al-Baqarah:286

"You can never understand everything. But, you should push yourself to understand the system"

— Ryan Dahl

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

Alhamdulillah, segala puji dan syukur penulis panjatkan kepada Allah SWT, yang telah memberikan rahmat dan hidayah-Nya sehingga penulis mampu menyelesaikan Laporan Tugas Akhir yang berjudul "Pengembangan Sistem Presensi untuk *Work From Home* (WFH) dan *Work From Office* (WFO) Selama Pandemi COVID-19".

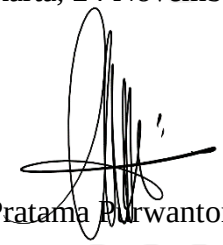
Laporan ini disusun sebagai bukti pelaksanaan magang dan syarat untuk menyelesaikan pendidikan pada jenjang strata 1 (S1) pada jurusan Informatika Universitas Islam Indonesia. Laporan ini tidak akan dapat diselesaikan dengan cepat dan maksimal tanpa adanya dukungan serta motivasi dari berbagai pihak. Oleh karena itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, atas limpahan rahmat dan hidayah-Nya yang telah memberikan rezeki, kesehatan, keselamatan, dan kelancaran dalam menyelesaikan Laporan ini.
2. Bapak Purwantoro dan Almarhumah Ibu Suratmi, selaku orang tua penulis yang senantiasa memberikan doa, dukungan, dan selalu bersedia menjadi tempat bercerita bagi penulis.
3. Bapak Moh. Idris, S.Kom., M.Kom, selaku dosen pembimbing Tugas Akhir yang telah memberikan waktu, ilmu, arahan, dan bimbingan kepada penulis dalam menyelesaikan Laporan Tugas Akhir ini.
4. Bapak Cahyo Dwi Raharjo, Ibu Ira Anggraini Siregar, Ibu Anisah Puji, Mba Icha dan segenap karyawan lain di Refactory yang telah memberikan kesempatan, ilmu, dan bantuan kepada penulis selama proses magang berlangsung.
5. Bapak dan Ibu dosen Program Studi Informatika, yang telah memberikan ilmu yang bermanfaat kepada penulis selama masa perkuliahan.
6. Aliffia Marsha Nadhira yang selalu memberikan nasihat, dukungan, dan bantuan kepada penulis untuk dapat menyelesaikan Laporan ini.
7. Teman-teman Kontrakan BRI yang tidak bisa disebutkan satu persatu selaku teman dekat penulis yang selalu memberikan semangat dan mewarnai kehidupan perkuliahan penulis. Semoga kalian selalu diberikan kesehatan dan lindungan oleh Allah SWT.
8. Teman-teman PIXEL, Informatika angkatan 2017.
9. Seluruh pihak yang tidak bisa penulis sebutkan satu persatu tetapi senantiasa mendukung penulis baik secara langsung maupun tidak langsung.

Semoga segala bantuan, ilmu, dukungan, dan bimbingan yang telah diberikan kepada penulis mendapatkan kebaikan dari Allah SWT. Penulis menyadari Laporan Tugas Akhir ini masih jauh dari kata sempurna, sehingga penulis dengan senang hati akan selalu menerima segala bentuk kritik dan saran yang membangun demi kesempurnaan Laporan Tugas Akhir ini. Akhir kata, semoga Laporan Tugas Akhir ini dapat memberikan manfaat bagi semua pihak.

Wassalamu'alaikum Waramahtullahi Wabarakatuh

Yogyakarta, 24 November 2021



(Fajar Pratama Purwantoro Putra)

SARI

Pandemi COVID-19 yang belum menunjukkan penurunan membuat beberapa perusahaan menerapkan kebijakan *work from home* atau bekerja dari rumah bagi sebagian karyawannya. Kebijakan ini diambil dengan tujuan agar karyawan tidak perlu datang ke kantor untuk bekerja dan meminimalisir potensi penyebaran virus COVID-19. Akan tetapi, tidak sedikit juga perusahaan yang menerapkan sistem *hybrid*, yaitu mempekerjakan sebagian karyawannya dari kantor dan dari rumah sebagai penerapan dari *new normal*. Salah satu kendala yang dihadapi oleh perusahaan yang menerapkan *work from office* sekaligus *work from home* adalah data kehadiran karyawan yang tidak terintegrasi dalam sebuah sistem. Selain itu, transparansi pekerjaan juga menjadi salah satu kendala yang ditemui. Untuk mengatasi kendala tersebut, dibutuhkan sebuah sistem presensi yang dapat digunakan baik dari rumah ataupun dari kantor untuk mempermudah proses pengajuan dan pendataan kehadiran karyawan. Sistem ini berbentuk sebuah aplikasi berbasis web yang dikembangkan menggunakan Laravel. Selain itu, sebuah *Rest API* juga dikembangkan agar sistem ini dapat digunakan pada aplikasi *mobile*. Penelitian ini membahas fitur-fitur apa saja yang dapat diimplementasikan ke dalam sebuah sistem presensi. Dari hasil penelitian dan pengujian yang dilakukan, diketahui bahwa sistem presensi dapat mempermudah proses pengajuan, pendataan, dan manajemen kehadiran karyawan karena sistem dapat digunakan pada web *browser* dan aplikasi *mobile*. Selain itu, sistem ini juga memiliki fitur-fitur yang memenuhi kebutuhan manajemen kehadiran karyawan.

Kata kunci: sistem presensi, *Rest API*, *work from office*, *work from home*.

GLOSARIUM

Agile	salah satu metodologi pengembangan perangkat lunak
Debug	langkah untuk menelusuri kesalahan kode program
Package	kumpulan *file* yang dibutuhkan untuk mengembangkan sebuah perangkat lunak
QR Code	sebuah *barcode* yang berbentuk dua dimensi
Rest API	salah satu arsitektur pengembangan dari sebuah *API*
Scrum	salah satu kerangka kerja yang menggunakan metodologi *agile*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kondisi pandemi virus COVID-19 yang semakin meluas menyebabkan banyak perusahaan mengambil keputusan untuk mempekerjakan karyawannya dari rumah atau *work from home* (WFH). Kebijakan ini diambil dengan tujuan agar karyawan tidak perlu datang ke kantor untuk bekerja dan meminimalisir potensi penularan virus COVID-19. Akan tetapi, tidak sedikit juga perusahaan yang masih membutuhkan karyawan untuk bekerja dari kantor ataupun yang menerapkan sistem *hybrid*, yaitu kebijakan untuk mempekerjakan sebagian karyawan dari rumah dan dari kantor sebagai penerapan dari *new normal*.

Kehadiran adalah salah satu komponen penting sebagai indikator kedisiplinan dan produktivitas karyawan di dalam sebuah perusahaan. Selain itu, kehadiran juga digunakan untuk menentukan besaran gaji karyawan. Pada umumnya, perusahaan menggunakan *fingerprint* untuk mendata kehadiran karyawannya (Fitria, 2020). Akan tetapi, pendataan kehadiran karyawan menggunakan metode ini hanya dapat digunakan jika karyawan bekerja di kantor dan tidak dapat digunakan oleh karyawan yang bekerja dari rumah. Hal ini tentu saja dapat memperbesar resiko penularan virus COVID-19 karena fasilitas tersebut digunakan secara bersama. Selain itu, presensi menggunakan *fingerprint* tentu saja akan mempersulit pendataan kehadiran karyawan yang bekerja dari rumah atau WFH.

Selama penerapan WFH atau bekerja dari rumah, pendataan kehadiran karyawan menjadi salah satu kendala yang ditemui oleh perusahaan. Salah satu kendala lain yang ditemui perusahaan yang menerapkan WFO dan WFH adalah perusahaan tidak tahu apa yang sedang dikerjakan karyawan pada saat itu dan apakah karyawan melakukan presensi dari rumah atau kantor atau sedang dalam perjalanan ke suatu tempat. Oleh karena itu, untuk mengatasi kendala tersebut dibutuhkan sebuah sistem presensi yang dapat digunakan baik dari rumah maupun dari kantor untuk mempermudah proses pengajuan presensi dan pendataan kehadiran karyawan. Sistem ini juga dibutuhkan untuk memberikan transparansi kerja karyawan agar karyawan dapat tetap produktif meskipun harus bekerja dari rumah.

Sistem presensi adalah sebuah sistem yang digunakan untuk manajemen kehadiran karyawan pada sebuah lembaga atau instansi. Sistem ini mencatat kehadiran karyawan secara otomatis dan dapat menggunakan data kehadiran sebagai sumber laporan untuk kebutuhan

manajemen karyawan. Sistem presensi juga dinilai lebih efektif, efisien, dan akurat karena sistem presensi dapat dilengkapi dengan pengambilan lokasi menggunakan *GPS*.

Sistem ini adalah sebuah proyek yang dikembangkan selama melaksanakan magang di Refactory. Proyek sistem presensi adalah salah satu dari beberapa proyek yang diberikan kepada pemagang. Metode pengembangan yang digunakan untuk mengembangkan sistem ini adalah *agile* dengan *scrum* sebagai kerangka kerjanya. *Scrum* dipilih oleh Refactory sebagai kerangka kerja karena dapat membantu individu, tim, ataupun organisasi dalam menghasilkan solusi untuk menyelesaikan masalah kompleks yang adaptif. Pengembangan sistem menggunakan *scrum* dilakukan dalam sebuah iterasi singkat yang disebut dengan *sprint*, dimana di setiap akhir *sprint*, klien akan memberikan *feedback* terkait penambahan dan perubahan fitur yang ada (Schwaber Ken & Sutherland Jeff, 2020).

Pada laporan tugas akhir jalur magang ini akan dibahas fitur-fitur apa saja yang dapat diimplementasikan ke dalam sebuah sistem presensi untuk mempermudah proses pengajuan presensi, pendataan kehadiran, dan manajemen kerja karyawan selama kebijakan bekerja dari rumah atau WFO diterapkan.

1.2 Ruang Lingkup Magang

Pelaksanaan magang di Refactory berlangsung selama 6 bulan dari tanggal 4 Januari 2021 sampai dengan tanggal 9 Juli 2021. Refactory merupakan sebuah perusahaan pengembang perangkat lunak yang sedang berkembang di Indonesia yang telah melayani beberapa klien dalam skala nasional maupun internasional. Dengan tim yang terdiri dari *programmer* handal, Refactory sudah banyak membantu perusahaan-perusahaan untuk mewujudkan transformasi digital yang diinginkan. Selain itu, Refactory juga membantu banyak *programmer* untuk meningkatkan kemampuan mereka agar dapat bersaing serta sebagai solusi atas kurangnya sumber daya manusia. Beberapa layanan yang diberikan Refactory adalah *Joint Development, DevOps Service, Online Bootcamp/Custom Training on Site,* dan *Refactory Course.*

Selama melaksanakan magang di Refactory, penulis ditempatkan di bagian/divisi *training* yang dipimpin oleh Maulana Primbadi sebagai Direktur Divisi Training, dan dibimbing oleh soerang *supervisor* yang bernama Cahyo Dwi Raharjo yang kemudian diganti oleh Ira Anggraini Siregar.

Refactory berlokasi di Jalan Palagan Tentara Pelajar, KM. 9,8 Ngaglik, Kabupaten Sleman, Daerah Istimewa Yogyakarta, 55581 yang dapat dilihat pada Gambar 1.1

Gambar 1.1 Tangkapan layar *GPS* lokasi Refactory

Selama melaksanakan magang, penulis mengerjakan beberapa tugas seperti *Skill Test RSP—Room Booking Application, Skill Test RSP—Notes Application,* dan sebuah proyek yaitu Sistem Presensi. Secara keseluruhan, aktivitas magang yang dilakukan ditunjukkan pada Tabel 1.1.

Tabel 1.1 Tabel Aktivitas Magang

No	Aktivitas	Waktu	Lokasi
1	*Training Fundamental Course*	4—22 Januari 2021	*WFH*
2	Pengerjaan *Skill Test RSP—Room Booking Application*	25 Januari—1 Februari 2021	*WFH*
3	Pembagian *Project*	19 Februari 2021	*WFH*
4	Proses Pengerjaan Sistem Presensi	22 Februari—25 Juni 2021	*WFH*
5	Pengerjaan *Skill Test RSP—Notes Application*	28 Juni—5 Juli 2021	*WFH*

1.3 Tujuan

Tujuan dari pelaksanaan magang ini adalah untuk mengembangkan sebuah sistem presensi yang dapat digunakan baik dari rumah ataupun dari kantor.

1.4 Manfaat

Manfaat yang diperoleh dari pengembangan sistem presensi yang dapat digunakan baik dari rumah ataupun dari kantor adalah sebagai berikut:

a. Mempermudah proses pengajuan kehadiran dari rumah.
b. Mempermudah pendataan kehadiran karyawan yang melakukan presensi baik dari rumah maupun dari kantor.
c. Mempermudah proses manajemen kehadiran karyawan.
d. Memberikan transparansi kerja karyawan kepada atasan.

1.5 Sistematika Penulisan

Sistematika penulisan laporan ini dibuat dengan tujuan untuk mempermudah pembaca dalam memahami isi laporan secara menyeluruh. Berikut adalah susunan dari laporan ini.

BAB I PENDAHULUAN

Bab ini membahas latar belakang pengembangan sistem presensi, ruang lingkup magang, tujuan dari penulisan laporan, manfaat sistem presensi, dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini membahas beberapa teori yang berkaitan dengan pengembangan sistem presensi untuk karyawan yang menjalankan WFH dan WFO.

BAB III PELAKSANAAN MAGANG

Bab ini membahas alur pelaksanaan magang untuk megembangkan sistem presensi dan fitur-fitur apa saja yang dibutuhkan untuk mempermudah proses pengajuan dan pendataan kehadiran karyawan serta hasil dan pembahasan pengembangan sistem presensi.

BAB IV REFLEKSI PELAKSANAAN MAGANG

Bab ini membahas tentang refleksi diri dari pelaksanaan magang di Refactory baik dari segi teknis dan non-teknis.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari pengembangan sistem presensi dan saran yang diberikan untuk pihak-pihak terkait.

BAB II
DASAR TEORI

2.1 Sistem Presensi

Sistem presensi adalah sistem manajemen kehadiran yang digunakan oleh suatu lembaga atau instansi yang dapat secara otomatis mencatat data kehadiran yang dapat digunakan sebagai sumber laporan untuk kebutuhan manajemen karyawan (Khoiriyah et al., 2018). Sistem presensi sangat penting untuk mengetahui kehadiran karyawan dalam sebuah perusahaan. Perkembangan sistem presensi saat ini telah berkembang seiring dengan perkembangan teknologi seperti internet, komputer, maupun *smartphone.* Dimulai dari presensi menggunakan kertas, *barcode*, *fingerprint* dan biometrik lainnya, hingga sistem presensi yang dapat digunakan melalui aplikasi berbasis web maupun *smartphone* yang mendukung sistem untuk digunakan baik di kantor maupun di rumah (Husain et al., 2017).

Sistem presensi menggunakan aplikasi berbasis web maupun perangkat *mobile* merupakan bentuk digitalisasi dari pengajuan presensi konvensional yang menggunakan kertas atau *fingerprint*. Dengan menggunakan sistem presensi, karyawan tidak perlu mendatangi alat presensi dan dapat mengajukan presensi menggunakan komputer, laptop, *smartphone* atau perangkat *mobile* lainnya. Sistem presensi juga ada yang dikembangkan menggunakan *internet of things* karena sistem presensi dapat digunakan untuk mengidentifikasi, menemukan, melacak, memantau, dan memicu *event* terkait secara otomatis dan di waktu yang sebenarnya (Ferdika & Nasution, 2020).

Selama pandemi virus COVID-19 kegiatan karyawan yang bekerja dari rumah harus dipantau dan diawasi agar tidak ada karyawan yang melakukan kecurangan seperti berkeliaran di saat jam kerja atau lalai dengan pekerjaan yang diberikan. Hal tersebut perlu diperhatikan untuk memberikan transparansi kerja karyawan agar karyawan dapat tetap produktif walaupun harus bekerja dari rumah. Sistem presensi sangat diperlukan dalam pengawasan kegiatan kerja karyawan yang bekerja dari rumah maupun karyawan yang bekerja dari kantor.

Hal lain yang perlu diperhatikan dalam pengajuan presensi selama pandemi adalah penerapan protokol kesehatan. Pengajuan presensi menggunakan biometrik seperti *fingerprint* dapat membuat antrean karyawan yang hendak melakukan presensi di kantor. Hal ini dapat meningkatkan resiko penularan virus COVID-19. Dengan adanya sistem presensi, karyawan dapat mengajukan presensi menggunakan gawai pribadi seperti *laptop* ataupun *smartphone*, sehingga karyawan tidak perlu mengantre untuk mengajukan presensi. Sistem presensi juga diperlukan oleh

karyawan yang bekerja dari rumah sebagai alternatif dari alat *fingerprint* yang hanya tersedia di kantor.

Selain itu, kemudahan proses manajemen data kehadiran karyawan juga menjadi salah satu alasan mengapa sistem presensi dibutuhkan selama pandemi COVID-19. Penerapan WFO dan WFH secara bersamaan akan berarti kehadiran karyawan akan diajukan dari dua tempat yang berbeda, yaitu dari rumah dan dari kantor. Penerapan kebijakan WFH tentu akan mempersulit proses pengajuan kehadiran bagi karyawan yang bekerja dari rumah karena tidak bisa memiliki akses terhadap fasilitas presensi *fingerprint* yang ada di kantor. Hal ini juga akan mempersulit kebutuhan manajemen karyawan karena data kehadiran karyawan yang tidak terintegrasi. Dengan kata lain, kehadiran karyawan dari rumah dan dari kantor akan didata menggunakan cara yang berbeda. Oleh karena itu, diperlukan sebuah sistem agar data kehadiran karyawan dapat terintegrasi. Selain untuk memfasilitasi karyawan untuk mengajukan kehadiran dengan mudah, sistem presensi juga akan berperan untuk menyimpan data kehadiran semua karyawan, terlepas dari penerapan kebijakan WFO dan WFH.

2.2 Laravel

Laravel adalah sebuah *framework* pengembangan aplikasi berbasis web yang dikembangkan oleh Taylor Otwell. Laravel dibuat dengan arsitektur *Model-View-Controller* (MVC) yang menggunakan bahasa pemrograman PHP. Laravel didesain untuk meningkatkan kualitas aplikasi dengan meminimalisir biaya pengembangan dan perbaikan serta mempermudah pekerjaan dengan sintaks yang ekspresif, mudah dipahami, dan kerangka aplikasi yang dapat mengurangi waktu pengembangan untuk meningkatkan produktivitas (Purnomo, 2016).

Laravel menyediakan sebuah *tools* untuk berinteraksi dengan *database* yang disebut dengan *migration*. Dengan menggunakan *migration*, *developer* dapat dengan mudah mendesain dan mengubah *database* pada sebuah *platform* secara independen karena implementasi skema *database* yang direpresentasikan dalam sebuah *class*. *Migration* dapat digunakan pada beberapa *database* yang telah didukung Laravel seperti MySQL, PostgreSQL, MSSQL, dan SQLite.

Laravel menggunakan *object-relational mapper* (ORM) yang dinamakan Eloquent untuk berinteraksi dengan *database* yang digunakan pada sebuah aplikasi. Eloquent mempermudah *developer* dalam melakukan transaksi ke *database* seperti proses *Create, Read, Update,* dan *Delete* (CRUD) hanya dengan menggunakan sintaks PHP dan tidak perlu membuat kode *SQL* yang kompleks. Selain itu, Eloquent juga mempermudah *developer* dalam mengelola *relationship* antar model dan membuat halaman secara otomatis.

Laravel juga menyediakan sebuah *command-line interface tool* yang disebut dengan Artisan. Dengan menggunakan Artisan, *developer* dapat dengan mudah berinteraksi dengan aplikasi yang dikembangkan menggunakan *terminal* untuk menjalankan *migration*, menjalankan *unit test*, menjalankan *scheduled task*, membuat *controller*, dan masih banyak lagi. *Developer* juga dapat dengan bebas membuat sebuah *command* Artisan baru sesuai dengan kebutuhan aplikasi yang dikembangkan. Selain itu, Laravel juga hadir dengan *routing system* yang mudah untuk dikelola yang dapat memberikan kemudahan bagi *developer* untuk mengatur *URL* yang ada pada aplikasi. Laravel menggunakan *Blade template engine* yang memberikan estetika dan kebersihan kode pada *view* secara parsial (McCool, 2012).

2.3 *Rest API*

REST API atau yang juga dikenal sebagai *RESTful API* adalah sebuah *Application Programming Interface* (*API* atau *web API*) yang mengimplementasikan prinsip-prinsip arsitektur *REST*. *REST* atau *Representasional State Transfer* adalah seperangkat prinsip arsitektur dimana data dapat ditransmisikan melalui sebuah *interface* yang terstandarisasi seperti *HTTP* dan ditemukan oleh Roy Fielding (Dhingra, 2016).

Application Programming Interface atau *API* sendiri adalah sekumpulan definisi dan protokol yang digunakan untuk mengembangkan dan mengintegrasikan perangkat lunak (Akbar, 2018). *API* juga dapat diartikan sebagai penghubung antara pengguna atau *client* dengan sumber data atau layanan *web* yang diinginkan (Akbar, 2018).

Seorang pengembang *API* dapat mengimplementasikan *REST* dalam berbagai macam cara. Beberapa *HTTP method* yang digunakan *REST* adalah *GET, POST, PUT, PATCH,* dan *DELETE* untuk mendefinisikan berbagai operasi CRUD yang berbeda. Selain itu, *response* yang diberikan *REST* dapat berupa *XML* atau *JSON* (Pranajaya, 2018).

2.4 *QR Code*

QR Code atau singkatan untuk *Quick Response Code* adalah tipe *barcode* yang berisi matriks titik yang pertama kali didesain untuk industri otomotif di Jepang. Akan tetapi, sekarang *QR Code* menjadi populer dan banyak digunakan di luar industri otomotif karena kemampuan *QR Code* yang dapat dibaca dengan cepat dan memiliki kapasitas penyimpanan yang lebih besar dibandingkan dengan *barcode* pada umumnya (Masalha & Hirzallah, 2014).

QR Code dapat dipindai menggunakan *QR Scanner* atau *smartphone*. Ketika *QR Code* dipindai, *software* yang ada di perangkat akan mengkonversi titik-titik di dalam kode menjadi sebuah rangkaian karakter atau angka (Masalha & Hirzallah, 2014).

QR Code seperti yang ditunjukkan pada Gambar 2.1 dibaca menggunakan perangkat pencitraan seperti kamera dan diformat menggunakan algoritma oleh perangkat yang menggunakan *Reed-Solomon error correction* hingga *QR Code* dapat diinterpretasikan dengan tepat. Data kemudian didapatkan dari pola yang ada baik dari komponen vertikal dan horizontal yang ada pada *QR Code* (Masalha & Hirzallah, 2014).

Gambar 2.1 *QR Code (Quick Response Code)*

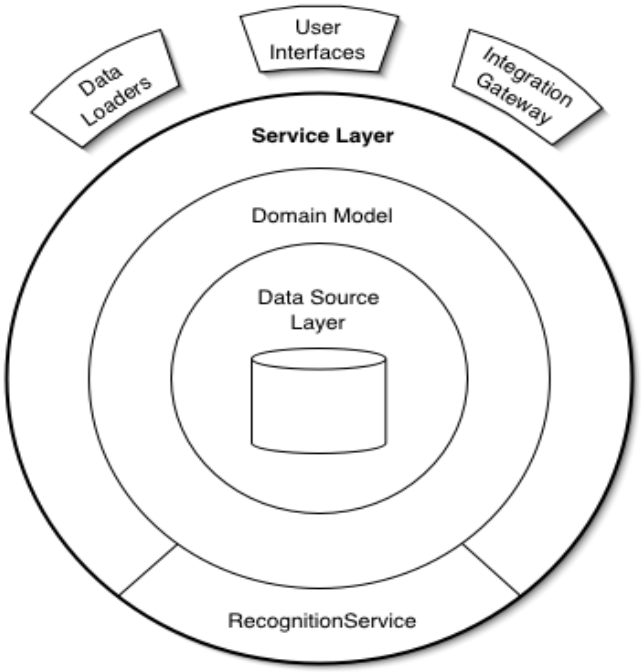
QR Code berbentuk kotak dan memiliki beberapa kotak di sudut bawah kiri, sudut atas kiri, dan sudut atas kanan. Kotak-kotak tersebut mendefinisikan orientasi dari *QR Code*. Titik-titik yang ada pada *QR Code* berisi format dan versi informasi, serta konten yang ada dienkripsi menggunakan *QR Code*. *QR Code* juga menggunakan beberapa tingkat *error correction*, yang didefinisikan sebagai L, M, Q, dan H. *QR Code* dengan *error correction* yang rendah (L) membuat *QR Code* dapat menyimpan lebih banyak konten di dalamnya, sedangkan tingkatan *error correction* yang lebih tinggi (H) membuat *QR Code* lebih mudah untuk dipindai (Christensson, 2015).

2.5 *Service Layer Pattern*

Sebuah aplikasi pada umumnya memerlukan beberapa bentuk interaksi yang berbeda untuk data yang akan disimpan dan logika yang diimplementasikan, seperti *data loaders, user interfaces,* dan *integration gateways*. Terlepas dari tujuan yang berbeda, *interface* tersebut terkadang

membutuhkan interaksi yang sama kepada aplikasi untuk mengakses dan memanipulasi data serta untuk menjalankan proses bisnisnya. Interaksi tersebut bisa saja kompleks, melibatkan transaksi di berbagai sumber daya dan koordinasi dari beberapa respons untuk sebuah operasi. Pengkodean logika yang digunakan interaksi secara terpisah di setiap *interface* akan mengakibatkan banyak duplikasi (Fowler et al., 2002).

Service layer mendefinisikan batasan-batasan dari sebuah aplikasi dengan operasi yang tersedia. *Service layer* mengenkapsulasi *business logic* yang terdapat pada sebuah aplikasi, mengontrol transaksi, dan mengkoordinasi *response* yang diberikan dari implementasi sebuah operasi (Kosasi & Liauw, 2013). Struktur dari *service layer* ditunjukkan pada Gambar 2.2.



Gambar 2.2 Struktur *Service Layer*

Berdasarkan penjelasan di atas, dapat disimpulkan bahwa *service layer* adalah sebuah pola arsitektur yang digunakan dalam pengembangan sebuah perangkat lunak yang bertujuan untuk mengorganisir logika bisnis dari sebuah perangkat lunak tersebut. Dalam penerapannya terutama pada pengembangan sistem presensi ini, *service layer* digunakan untuk memisahkan logika bisnis yang digunakan dari *web controller* dan *API controller.* Penggunaan *service layer* pada sistem ini bertujuan untuk membuat modifikasi logika bisnis menjadi lebih mudah, mencegah duplikasi diantara *web controller* dan *API controller*, dan membuat kode di *controller* menjadi lebih bersih dan mudah untuk dibaca.

2.6 Laravel Queue and Jobs

Sebuah aplikasi berbasis web memiliki sebuah *task* penting untuk dikerjakan di *background*. *Task* tersebut dikerjakan diluar proses utama aplikasi yang dapat dilihat oleh pengguna, dan tetap memastikan *task* yang penting tetap dapat dijalankan agar aplikasi web dapat tetap berjalan dengan mulus. Dengan mengerjakan *task* yang berat pada *background*, pengguna dapat tetap menggunakan aplikasi tanpa harus menunggu aplikasi menyelesaikan sebuah proses yang dapat memakan waktu yang lama.

Untuk meningkatkan pengalaman pengguna, kecepatan sebuah halaman dimuat menjadi salah satu hal penting bagi sebuah aplikasi web. Data perlu ditampilkan di layar, *request* harus diselesaikan, dan pengguna perlu merasa memegang kendali atas aplikasi web yang sedang digunakan. *Background jobs* dibuat untuk menangani *task* yang membutuhkan waktu untuk diselesaikan atau *task* yang tidak terlalu penting untuk ditampilkan hasilnya di layar. Untuk *query* apapun yang mungkin memerlukan waktu lebih dari 1 (satu) detik untuk diproses, pengembang aplikasi harus mempertimbangkan untuk menjalankan *query* tersebut di *background* sehingga aplikasi web dapat memberikan respon dengan cepat dan memproses *request* lainnya.

Task yang mengandalkan layanan eksternal juga cocok untuk dikerjakan di *background*. Mengirim *email* konfirmasi, menyimpan foto, membuat *thumbnail*, atau membuat *post* di media sosial adalah beberapa contoh dari *task* yang tidak perlu dijalankan pada *foreground* aplikasi karena akan membutuhkan waktu yang lama untuk diselesaikan. *Controller* yang ada pada aplikasi dapat menempatkan *email*, pemrosesan gambar, atau *post* media sosial ke dalam sebuah *delayed job queue*, yang kemudian akan mengembalikan kontrol aplikasi kepada pengguna secepat mungkin (Stamat, 2020).

Laravel menyediakan sebuah fitur yang dapat digunakan untuk memasukkan sebuah *task* ke dalam *queue* dan mengerjakan *task* atau *job* tersebut pada *background* aplikasi. Berdasarkan dokumentasi Laravel, Laravel Queue adalah sebuah fitur antrian untuk pemrosesan *task* atau *job* yang akan dikerjakan pada *background* menggunakan Amazon SQS, Redis, atau *database* relasional seperti MySQL dan PostgreSQL.

Queue dan *jobs* pada pengembangan sistem presensi diimplementasikan untuk proses pengiriman *email* notifikasi kepada atasan dan pengiriman *email* untuk *reset password*. Kedua fitur ini perlu untuk diproses pada *background* aplikasi karena pengiriman *email* yang membutuhkan waktu untuk diselesaikan dan juga karena respon dari pengiriman *email* yang tidak terlalu penting untuk ditampilkan kepada pengguna.

2.7 Laravel Cache

Cache adalah sebuah komponen yang digunakan untuk menyimpan data, sehingga *request* data di masa yang akan datang dapat diberikan dengan cepat. Data yang disimpan di dalam *cache* bisa jadi adalah hasil dari komputasi yang pernah dilakukan atau duplikasi dari data yang disimpan dalam media penyimpanan seperti *database*. Penggunaan *cache* membuat aplikasi tidak perlu melakukan komputasi yang sama berulang kali (Ekandem, 2021).

Beberapa proses pengambilan data yang dilakukan oleh sebuah aplikasi dapat membutuhkan banyak CPU atau membutuhkan waktu hingga beberapa detik untuk selesai diproses. Jika hal ini terjadi, biasanya data yang dihasilkan akan disimpan di dalam *cache* untuk sementara waktu agar data tersebut dapat diakses dengan cepat pada *request* dengan hasil data yang sama. Pada umumnya, data yang telah di-*cache* akan disimpan pada media penyimpanan yang dapat diakses dengan cepat seperti Memcached dan Redis (Laravel, n.d.-a).

Laravel telah menyediakan sebuah fitur yang dapat digunakan untuk menyimpan data ke dalam *cache*, dengan beberapa pilihan *driver* seperti Memcached, Redis, dan DynamoDB. Pada pengembangan sistem presensi ini, *cache* akan disimpan ke dalam Redis. Redis sendiri adalah sebuah struktur penyimpanan data dalam memori yang dapat digunakan sebagai *database*, *cache*, dan *message broker* (Redis, n.d.). Data-data yang akan di-*cache* dan disimpan ke dalam Redis pada sistem presensi ini adalah data-data seperti daftar kantor, daftar karyawan, dan daftar presensi karyawan. Redis dipilih sebagai media penyimpanan sementara pada pengembangan sistem presensi ini karena Redis adalah sebuah media penyimpanan pada memori yang sedang populer, mampu membaca data lebih cepat dibandingkan dengan *database* relasional, dan mudah untuk digunakan.

2.8 Laravel Debugbar

Pada tahap pengembangan sebuah aplikasi web, proses *query* ke *database* adalah hal yang pasti dilakukan. Akan tetapi, seringkali *query* yang terdapat pada sebuah proses aplikasi menjadi kompleks dan mempengaruhi *response time* aplikasi. Untuk mengetahui *query* pada proses apa yang memperberat jalannya aplikasi, perlu untuk dilakukan *debug*. *Debug* dilakukan dengan tujuan agar pengembang aplikasi dapat melakukan optimasi *query*, sehingga aplikasi dapat berjalan dengan cepat dan memberikan pengalaman pengguna yang baik. Proses *debugging* akan dilakukan selama aplikasi berada dalam tahap *development* atau pengembangan.

Dalam pengembangan aplikasi web menggunakan Laravel, *developer* atau pengembang menggunakan sebuah *package* untuk melakukan *debugging* yang bernama Laravel Debugbar.

Laravel Debugbar adalah sebuah *package* yang mengintegrasikan *PHP Debug Bar* dengan Laravel. *Package* ini mengimplementasikan beberapa *custom DataCollectors* yang dibuat khusus untuk Laravel. Adapun beberapa *custom collectors* yang digunakan oleh *package* ini adalah sebagai berikut (Heuvel, n.d.).

a. *QueryCollector,* menampilkan semua *query* termasuk waktu yang dibutuhkan untuk menyelesaikan *query* tersebut.
b. *RouteCollector,* menampilkan informasi terkait *Route* saat ini.
c. *ViewCollector,* menampilkan informasi terkait *views* yang sedang dimuat.
d. *EventsCollector,* menampilkan semua *events.*
e. *LaravelCollector,* menampilkan informasi terkait versi Laravel dan *environment* yang digunakan.
f. *SymfonyRequestCollector,* sebagai pengganti dari *RequestCollector* dengan informasi yang lebih banyak terkait dengan *request* dan *response.*
g. *LogsCollector,* menampilkan entri log terbaru dari log penyimpanan (dinonaktifkan secara *default*).
h. *FilesCollector*, menampilkan *file* yang disertakan atau yang dibutuhkan oleh PHP (dinonaktifkan secara *default*).
i. *ConfigCollector,* menampilan semua *values* yang terdapat pada *file* konfigurasi (dinonaktifkan secara *default*).
j. *CacheCollector,* digunakan untuk menampilkan semua *cache events* (dinonaktifkan secara *default*).

Penggunaan *package* ini dapat membantu pengembang aplikasi web mendapatkan informasi yang relevan dengan halaman yang diakses. Informasi yang diberikan dapat digunakan oleh pengembang untuk mengoptimasi aplikasi sehingga membuat kinerja aplikasi menjadi lebih baik. Selain itu, Laravel Debugbar mudah dipahami dan mudah untuk digunakan.

2.9 Laravel Sanctum

Dalam pengembangan sebuah *API,* autentikasi pengguna penting untuk dilakukan, hal ini bertujuan untuk mengetahui apakah seorang pengguna memiliki hak akses untuk mengakses atau memanipulasi sebuah *resource* atau data tertentu. Untuk mencapai tujuan ini pengembang *API* pada umumnya menggunakan *authorization request header*. *Request header* sendiri adalah sebuah *HTTP header* yang digunakan dalam *HTTP request* untuk memberikan informasi terkait konteks dari *request* yang dilakukan, sehingga *server* dapat memberikan respon yang sesuai. Salah satu

header yang ada pada *HTTP request header* adalah *authorization* yang dapat digunakan untuk mengautentikasi pengguna (Mozilla, n.d.).

Pada pengembangan *API* menggunakan Laravel, Laravel menyediakan sebuah *package* yang dapat digunakan untuk mengautentikasi pengguna yang dinamakan Laravel Sanctum. Berdasarkan dokumentasi, Laravel Sanctum adalah sebuah *package* yang dapat memberikan sistem autentikasi untuk *SPA (single page application)* dan aplikasi *mobile.* Laravel Sanctum dapat digunakan untuk membuat *API* token untuk pengguna tanpa penggunaan *OAuth* yang kompleks. Laravel Sanctum terinspirasi dari *GitHub* dan aplikasi-aplikasi lainnya yang menggunakan *personal access tokens* untuk autentikasi penggunanya. Token yang dibuat biasanya memiliki waktu habis atau *expiration time* yang sangat lama, namun pengguna aplikasi dapat mencabut (*revoke)* token secara manual kapan pun (Laravel, n.d.-b).

BAB III
PELAKSANAAN MAGANG

3.1 Pengembangan Proyek Sistem Presensi

3.1.1 Pendefinisian Proyek

Sistem presensi merupakan sebuah proyek yang diberikan kepada penulis pada tanggal 19 Februari 2021. Sistem presensi adalah sebuah aplikasi berbasis *web* yang digunakan untuk melakukan pengajuan kehadiran dan manajemen kehadiran karyawan baik dari rumah ataupun dari kantor. Sistem ini dibuat dengan tujuan untuk mengintegrasi data kehadiran karyawan yang bekerja dari kantor dan karyawan yang bekerja dari rumah agar pendataan dan manajemen kehadiran karyawan menjadi lebih mudah.

3.1.2 Inisialisasi Proyek

Pada tahap inisialisasi, spesifikasi sistem presensi ditentukan agar sistem dapat dikembangkan sesuai dengan harapan dan tujuan sistem dapat tercapai. Beberapa teknologi yang digunakan dalam pengembangan sistem presensi dapat dilihat pada Tabel 3.1. Setelah spesifikasi teknologi telah ditentukan, tahap selanjutnya adalah membuat sebuah *git repository* baru dan membuat proyek menggunakan *framework* yang telah ditentukan.

Tabel 3.1 Spesifikasi Teknologi Sistem Presensi

Aspek	**Teknologi**
Bahasa Pemrograman	PHP 7.3
Framework	*Laravel 8*
Database	*MySQL*
Cache Driver	*Redis*
Source Code Version Control	*GitHub*
SMTP Mail Server	*Mailtrap*
Deployment Platform	*Heroku*

Selain itu, pengembangan proyek sistem presensi juga terbagi dalam beberapa peran, di antaranya adalah:

a. *Project Manager,* bertanggung jawab atas pengelolaan proyek secara keseluruhan. Selain itu, *project manager* juga berperan sebagai otoritas tertinggi untuk konsultasi apabila dalam proses pengembangan ditemui kendala. Pada pengembangan sistem presensi, *supervisor* magang yang bernama Cahyo Dwi Raharjo berperan sebagai *project manager* yang kemudian digantikan oleh Ira Anggraini Siregar.
b. *Programmer*, bertanggung jawab untuk mengimplementasikan dan mengeksekusi rancangan kebutuhan sistem dalam bentuk kode program, membuat algoritma, dan mengimplementasikan teknologi yang sesuai dengan kebutuhan sistem untuk mencapai tujuan sistem yang sebelumnya telah dibuat. Terdapat 2 orang *programmer*, yaitu *web developer* dan *mobile developer* termasuk penulis. Selama pengembangan sistem, penulis bertugas untuk mengembangkan aplikasi berbasis web dan *Rest API* yang akan digunakan oleh aplikasi *mobile.*

3.1.3 Perencanaan Proyek

Sistem presensi sebagai sebuah sistem yang dapat digunakan untuk melakukan pengajuan dan manajemen kehadiran karyawan juga dapat digunakan untuk keperluan manajemen kantor, divisi, dan pengaturan jam kerja karyawan. Sistem ini direncanakan untuk memiliki pengguna dengan masing-masing fitur yang berbeda berdasarkan hak akses dan jabatan dari pengguna tersebut. Selain itu, sistem ini juga direncanakan dapat digunakan pada perangkat *mobile*. Oleh karena itu, pada sistem ini juga dikembangkan sebuah *Rest API* yang akan digunakan oleh aplikasi *mobile* yang dikembangkan oleh pemagang lain.

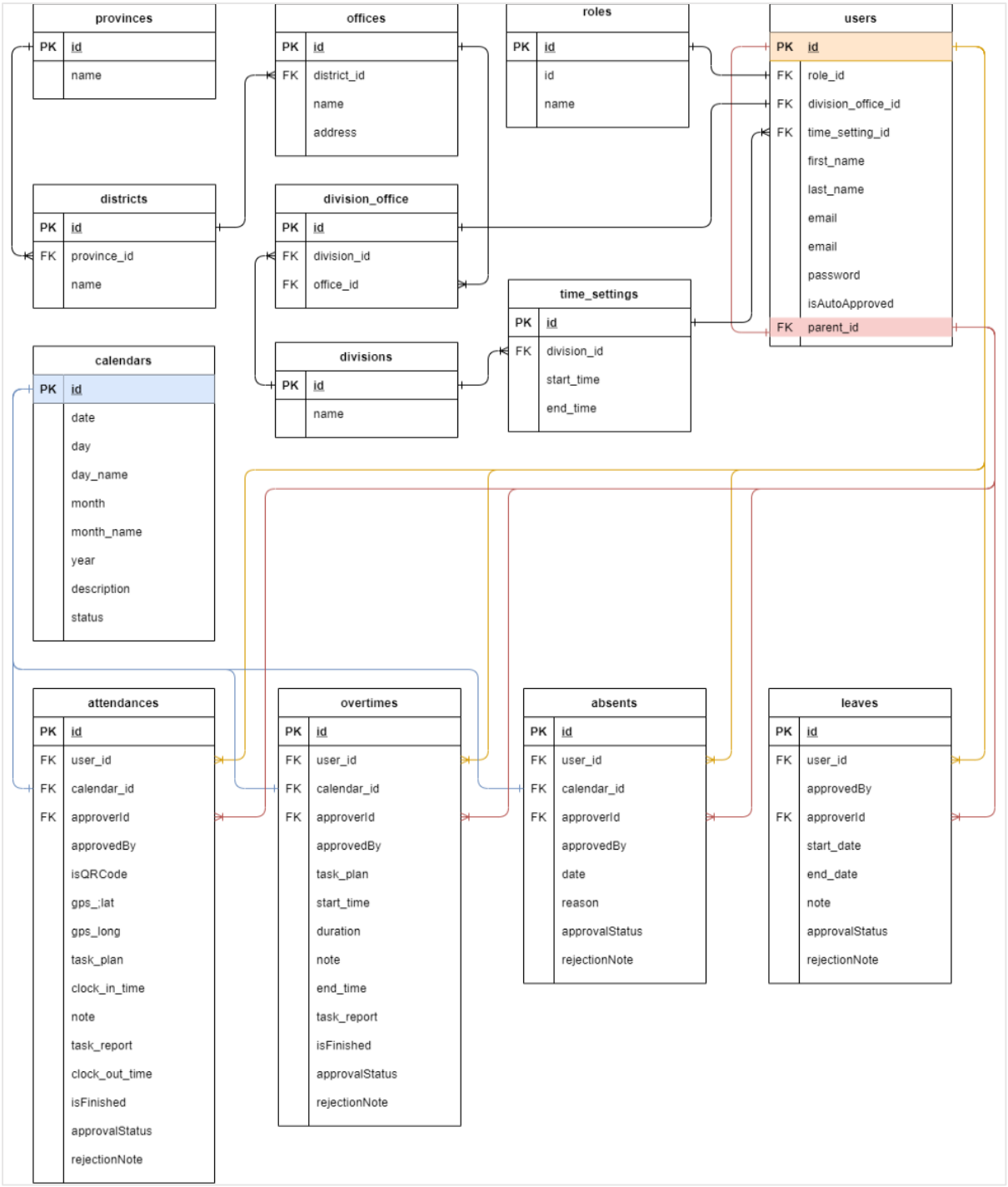
Sistem presensi dikembangkan menggunakan metode *agile* dengan kerangka kerja *scrum* dengan *weekly sprint planning* sebagai media untuk pelaporan *progress* dan konsultasi terkait kendala yang dihadapi selama proses pengembangan sistem presensi. Pengembangan sistem ini dijadwalkan selama 4 bulan, terhitung dari tanggal 22 Februari 2021 sampai dengan tanggal 1 Juli 2021 atau selama penulis melaksanakan magang.

3.1.4 Pelaksanaan Proyek

a. Analisis *requirement* dari Sistem Presensi

Proses pengembangan sistem presensi dimulai dengan menganalisa *requirement* sistem dan siapa saja pengguna sistem. Proses ini dilakukan pada tanggal 22—24 Februari 2021 oleh penulis

sebagai *web developer*, rekan pemagang sebagai *mobile developer*, dan *supervisor* yang berperan sebagai *project manager* sistem presensi. Dalam proses ini, dilakukan sesi curah ide atau *brainstorming* dimana peserta yang terlibat menyampaikan ide dan pendapat yang dimiliki. Hasil dari analisis ini adalah sebuah *database diagram* yang dapat dilihat pada Gambar 3.1 dan fitur-fitur berdasarkan pengguna sistem.



Gambar 3.1 *Database Diagram* Sistem Presensi

Selain itu, karena *Rest API* akan digunakan oleh aplikasi *mobile, endpoint* untuk keperluan autentikasi pengguna perlu dibuat secara manual. Untuk menangani autentikasi pengguna pada *Rest API,* sistem ini menggunakan sebuah *package* bernama Laravel Sanctum. Sistem ini juga menggunakan sebuah *package* bernama *Laravolt-Indonesia*, yaitu sebuah *package* yang berisi data provinsi, kabupaten/kota, dan kecamatan/desa di seluruh Indonesia yang akan digunakan untuk lokasi dari kantor yang ada di dalam sistem. Untuk mempermudah proses pengembangan sistem presensi, *migration* dan *seeder* yang berisi data *dummy* juga perlu untuk dibuat.

b. Analisis fitur-fitur berdasarkan hak akses pengguna

Setelah analisis rancangan dan *requirement* sistem selesai dilakukan, tahap selanjutnya adalah menganalisis fitur-fitur berdasarkan hak akses pengguna. Tahap analisis fitur berdasarkan hak akses pengguna dilakukan pada tanggal 25—26 Februari 2021. Ada beberapa *roles* yang terdapat pada sistem ini, salah satunya adalah *administrator* yang memiliki akses penuh terhadap sistem. Selain menggunakan *roles,* hak akses pengguna pada sistem ini juga akan ditentukan dari jabatan dan divisi pengguna tersebut. Sistem ini dirancang agar setiap karyawan memiliki atasan, kecuali karyawan yang memiliki *role* direktur ataupun *CEO* atau pimpinan perusahaan. Pada sistem ini atasan akan berperan sebagai pemberi persetujuan atau *approval* terhadap presensi yang dibuat dengan tujuan agar karyawan dapat bekerja secara efektif dan transparan.

Berdasarkan hak akses, pengguna sistem akan terbagi menjadi dua, yaitu *administrator* dan pengguna yang memiliki *role* selain *administrator* atau karyawan. Selain itu, fitur-fitur admin hanya dapat diakses menggunakan web *browser* karena aplikasi *mobile* hanya dapat digunakan oleh karyawan untuk melakukan pengajuan dan manajemen kehadiran. Beberapa fitur yang dapat digunakan oleh *administrator* adalah sebagai berikut:

a. Manajemen Karyawan
b. Manajemen *Roles*
c. Manajemen Kantor
d. Manajemen Divisi
e. Manajemen Kalender
f. Manajemen *Time Setting* untuk Divisi
g. Membuat *QR Code*
h. Membuat Laporan Kehadiran Karyawan
i. Melakukan *Reset Password* Karyawan Melalui *Email*

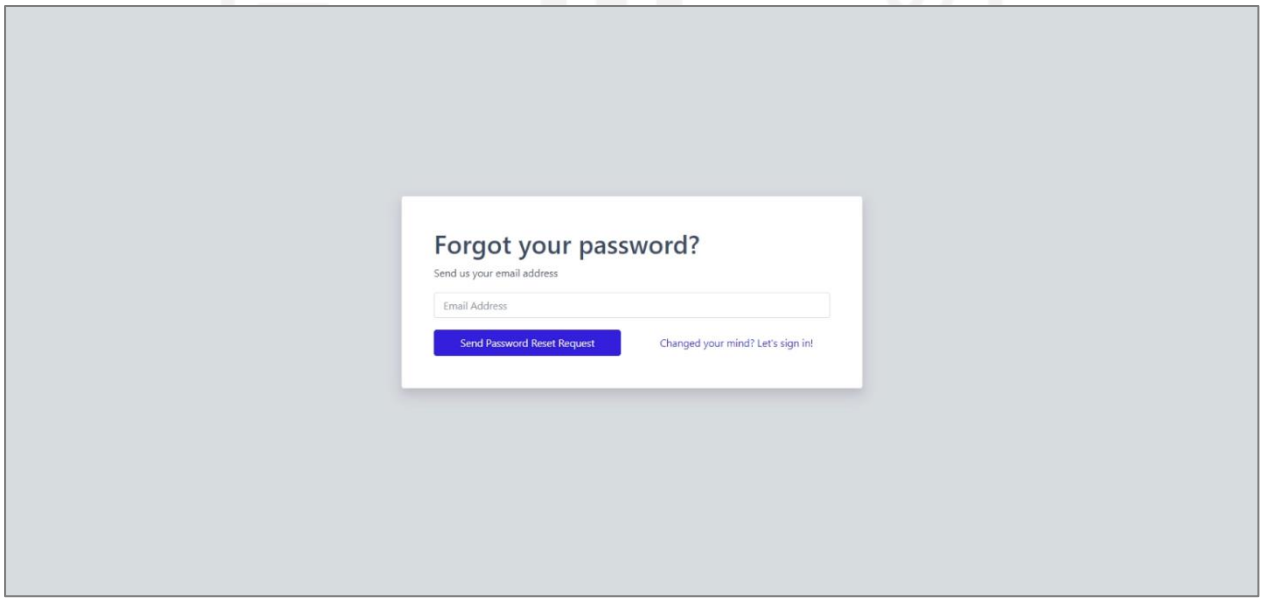
Pengguna yang memiliki *role* karyawan, terlepas dari jabatan dan divisi yang dimilikinya secara umum dapat menggunakan beberapa fitur seperti:

a. Pengajuan Presensi
b. Pengajuan Lembur
c. Pengajuan Izin
d. Pengajuan Cuti

Karyawan yang berperan sebagai *parent* atau atasan dari beberapa karyawan lain dapat menggunakan beberapa fitur tambahan seperti pemberian persetujuan atau *approval* atas kehadiran karyawan bawahannya. Sementara itu, karyawan yang bekerja di divisi *Human Resource* dapat menggunakan fitur manajemen kalender sama seperti *administrator*.

c. Pembuatan *custom password reset*

Berdasarkan hasil *brainstorming* pada proses analisis fitur, *supervisor* menyarankan untuk merubah proses *reset password* pengguna. Proses pengembangan fitur ini dilakukan pada tanggal 12 April 2021. Secara *default*, proses *reset password* dari Laravel diawali dengan pengguna mengisi *email* pada *form forgot password* yang ditunjukkan pada Gambar 3.2, kemudian *link* untuk mengubah *password* akan dikirimkan oleh aplikasi kepada pengguna tersebut melalui *email*. Ketika *link* tersebut diakses, pengguna akan dialihkan ke halaman yang berisi *form* untuk mengganti *password*.

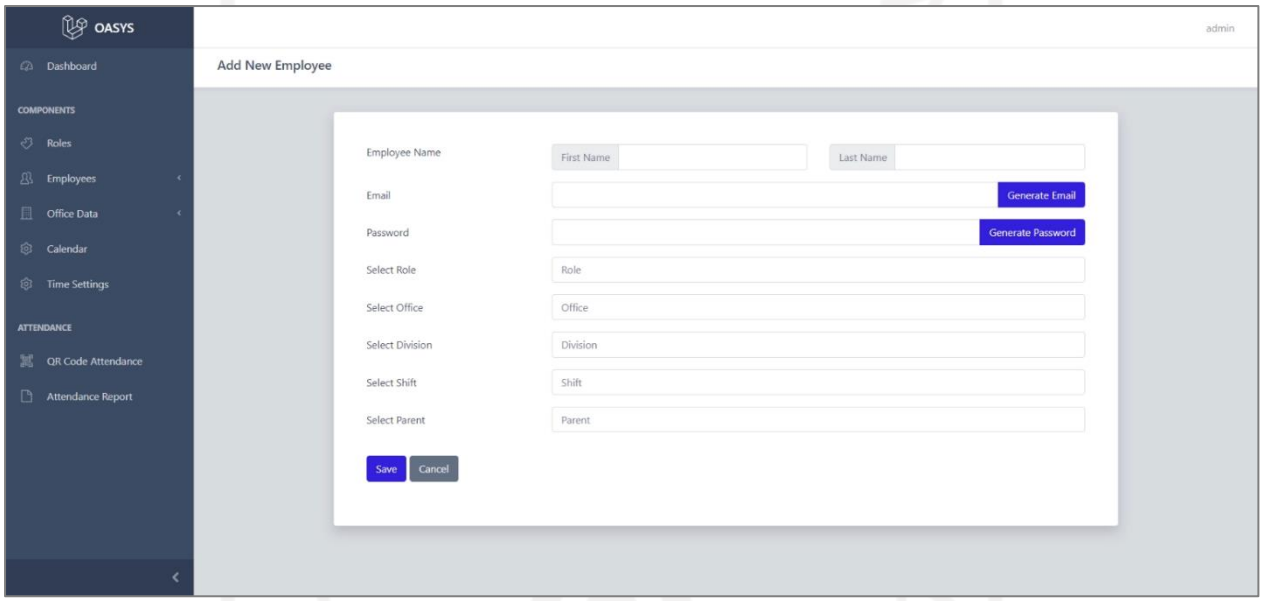


Gambar 3.2 *Form Forgot Password*

Sementara itu pada fitur *reset password* yang diubah *link* akan dikirimkan kepada *administrator* melalui *email*, kemudian *administrator* dapat mengubah *password* pengguna menggunakan *link* tersebut, sehingga pengguna tidak perlu mengubah *password* secara manual.

d. Pengembangan fitur manajemen karyawan untuk *administrator*

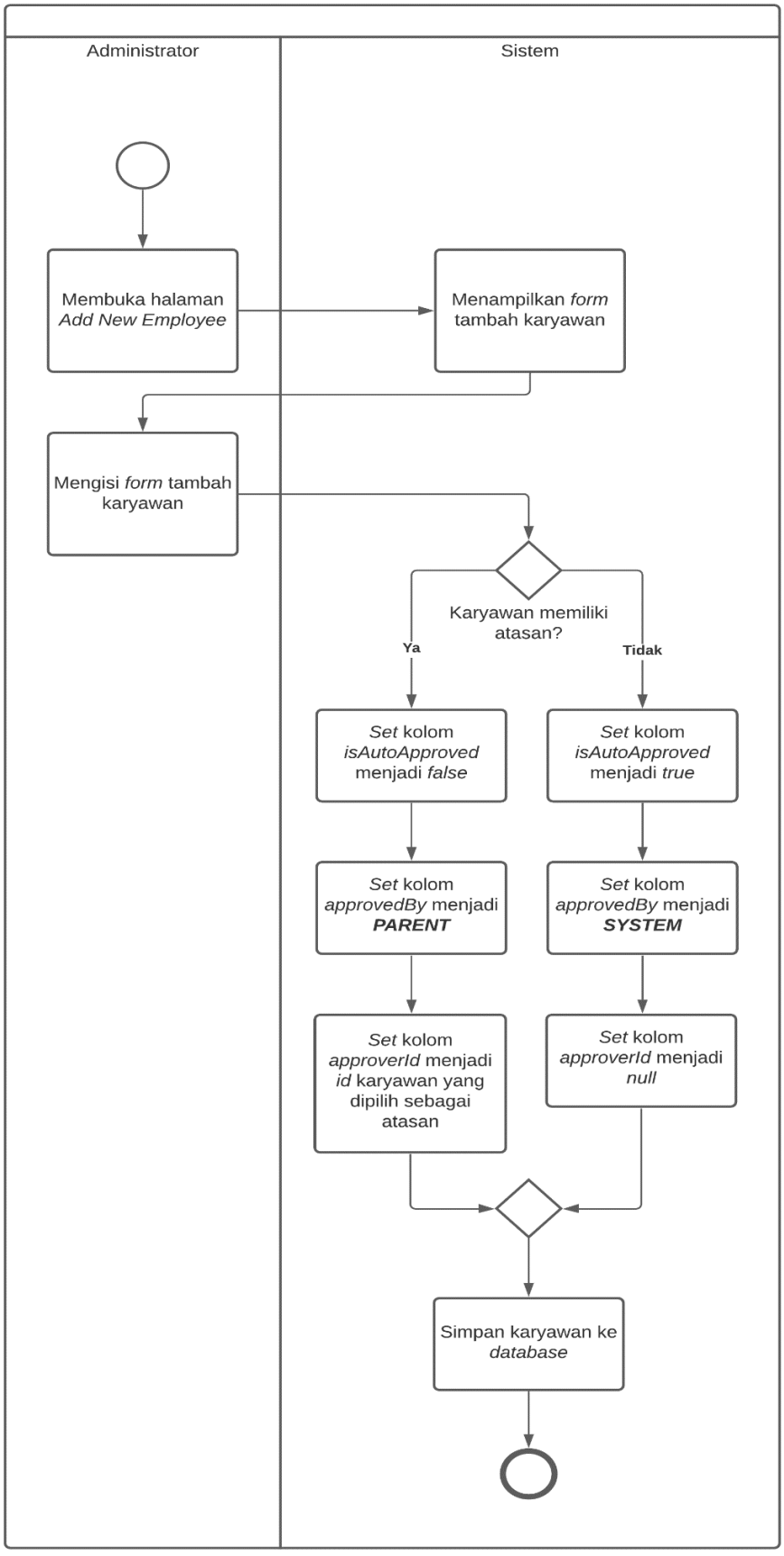
Salah satu fitur untuk *administrator* adalah fitur manajemen karyawan. Fitur ini dikembangkan selama 2 hari pada tanggal 11—12 Maret 2021. Pada fitur manajemen karyawan, *administrator* dapat mendaftarkan karyawan baru, mengubah data karyawan, menonaktifkan karyawan dan menghapus karyawan. Untuk mendaftarkan karyawan baru, *administrator* dapat mengakses *form* pada halaman *Add New Employee* yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 Halaman *Add New Employee*

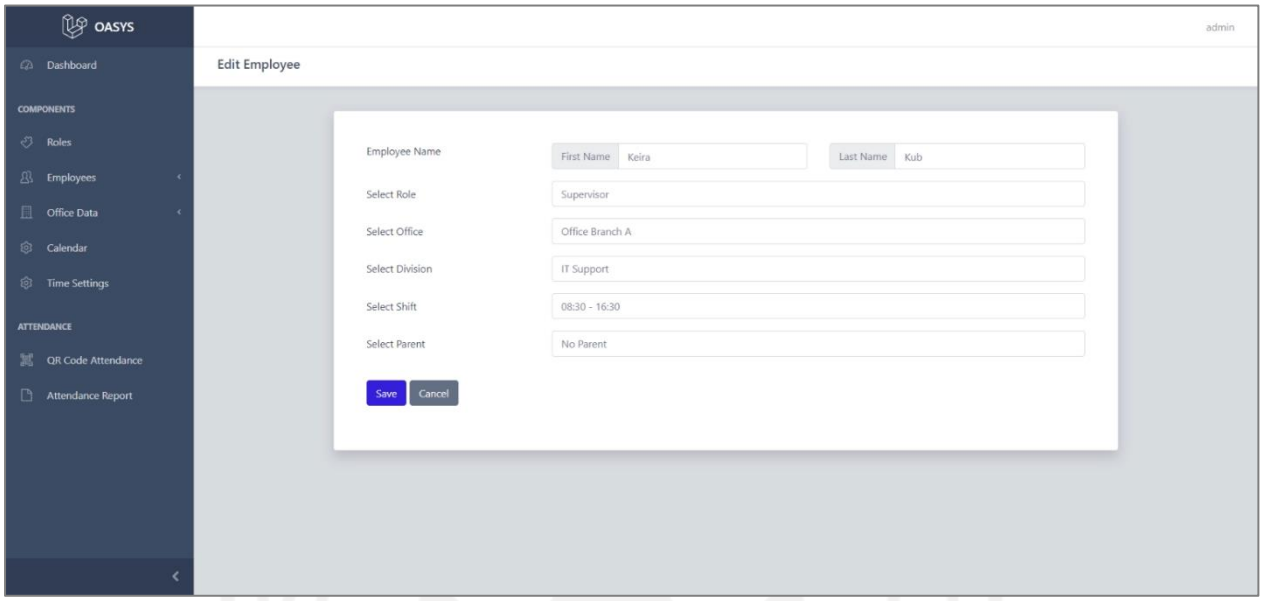
Untuk menambahkan karyawan baru, *administrator* perlu mengisi beberapa *field* pada *form* tambah karyawan seperti nama awal, nama akhir, *email*, *password*, *role* atau jabatan, kantor, divisi, jam kerja, dan *parent* atau atasan. *Administrator* dapat membuat *email* dan *password* secara otomatis atau secara manual. Selain itu, *dependent dropdown* digunakan untuk mempermudah pengisian *form* pada *field* kantor, divisi, dan jam kerja. *Administrator* dapat mengatur siapa *parent* atau atasan dari karyawan yang akan didaftarkan. Jika karyawan tersebut tidak memiliki atasan, maka sistem akan secara otomatis mengatur kolom *isAutoApproved* pada tabel *users* menjadi *true*, agar setiap kehadiran yang diajukan oleh karyawan tersebut secara otomatis disetujui oleh sistem.

Flowchart untuk proses tambah data karyawan baru oleh *administrator* dapat dilihat pada Gambar 3.4 di bawah ini.



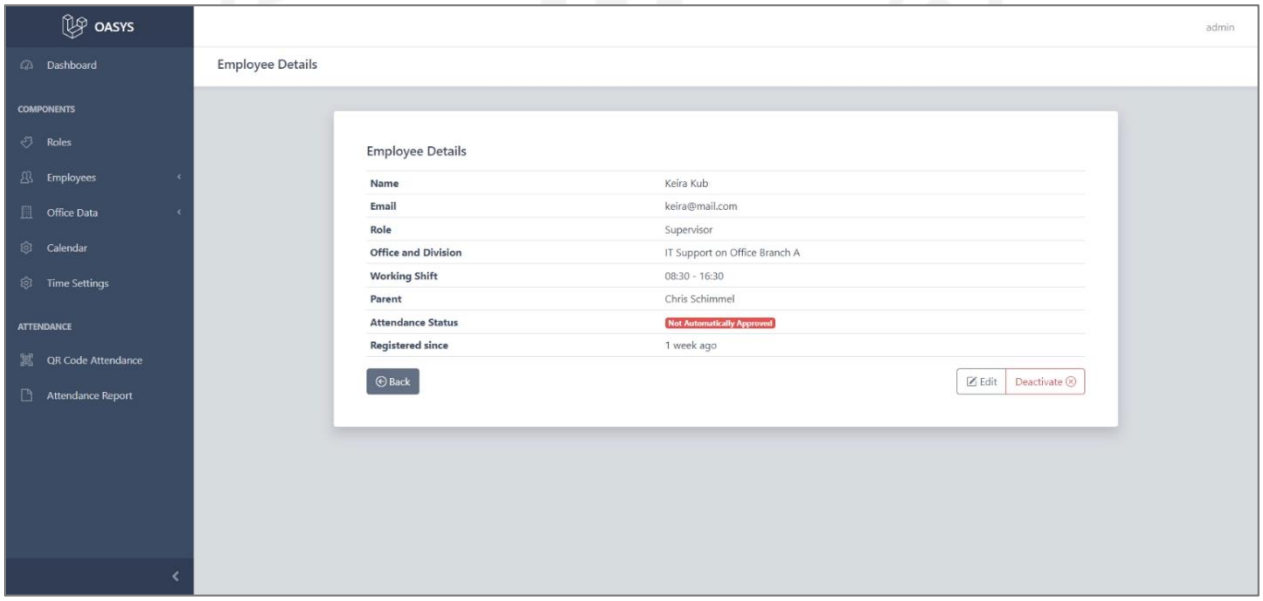
Gambar 3.4 *Flowchart* proses tambah karyawan

Untuk memperbarui data karyawan, *administrator* dapat mengakses *form* pada halaman *Edit Employee* seperti yang ditunjukkan pada Gambar 3.5. Proses dari penyuntingan data karyawan kurang lebih sama seperti proses pendaftaran karyawan pada Gambar 3.3.



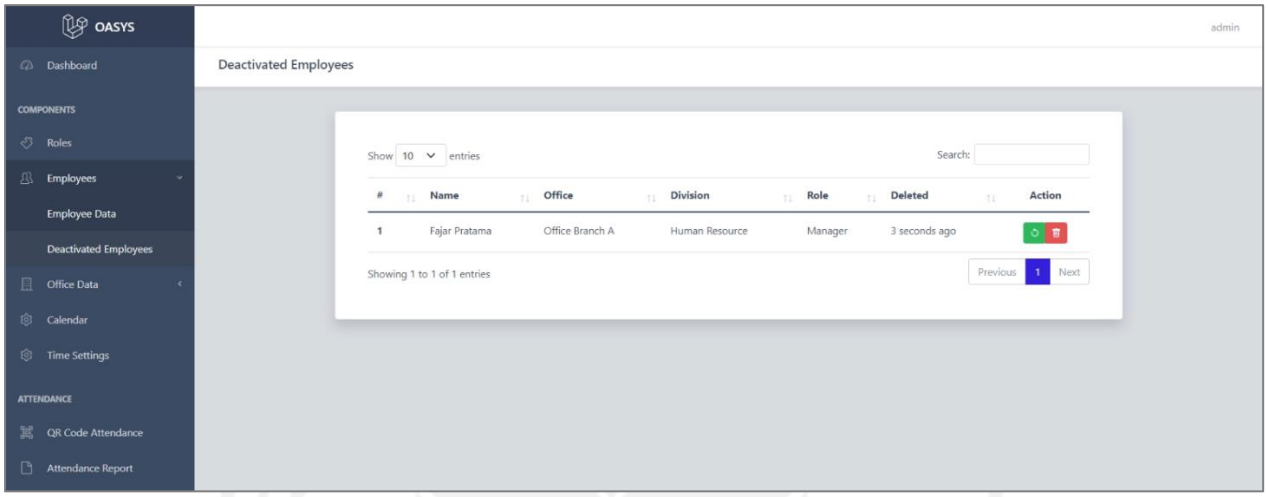
Gambar 3.5 *Form Edit Employee*

Selain menambah dan menyunting data karyawan, *administrator* dapat menonaktifkan karyawan melalui halaman *Employee Details* yang dapat dilihat pada Gambar 3.6.



Gambar 3.6 Halaman *Employee Details*

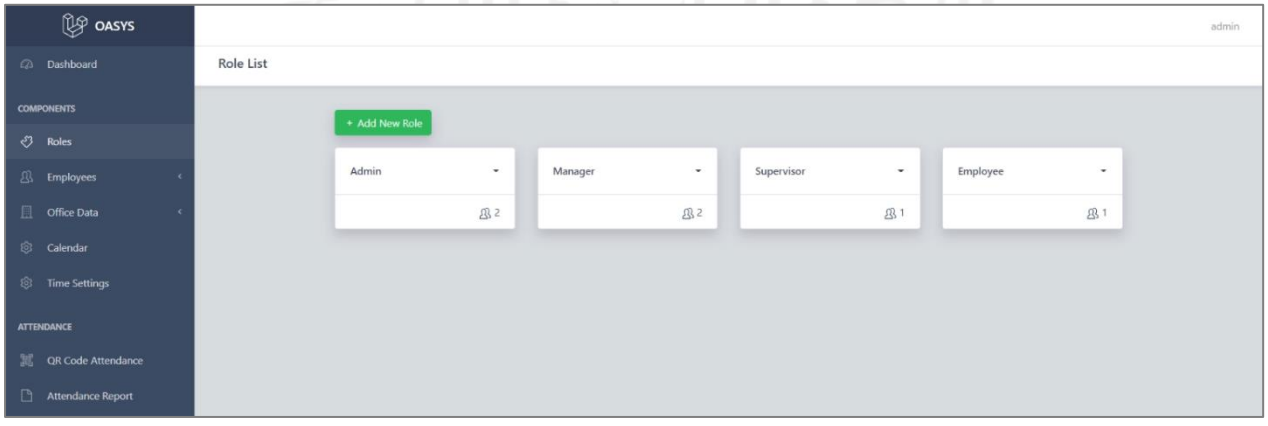
Sebelum menonaktifkan karyawan, sebuah pesan peringatan akan ditampilkan dalam bentuk *modal*. Fitur ini menerapkan *soft delete*, yang berarti data karyawan tidak benar-benar dihapus dari tabel *users*, tetapi hanya mengisi kolom *deletedAt* yang terdapat pada tabel tersebut sehingga data tidak ditampilkan kecuali menggunakan *method withTrashed* saat memuat data menggunakan *Eloquent. Administrator* dapat menghapus data karyawan secara permanen atau mengembalikan data tersebut melalui halaman *Deactivated Employees* yang dapat dilihat pada Gambar 3.7.



Gambar 3.7 Halaman *Deactivated Employees*

e. Pengembangan fitur *CRUD roles* untuk *administrator*

Pada sistem ini, *roles* atau jabatan berfungsi untuk mengatur hak akses pengguna dan dapat diatur secara dinamis oleh *administrator*. Fitur ini dikembangkan pada tanggal 13 Maret 2021. Gambar 3.8 menunjukkan data semua *roles* yang ada pada sistem beserta pengguna yang memiliki *role* tersebut. Pada fitur ini, *administrator* dapat menambahkan *role* baru, menyunting *role* yang ada, dan menghapus *role*.

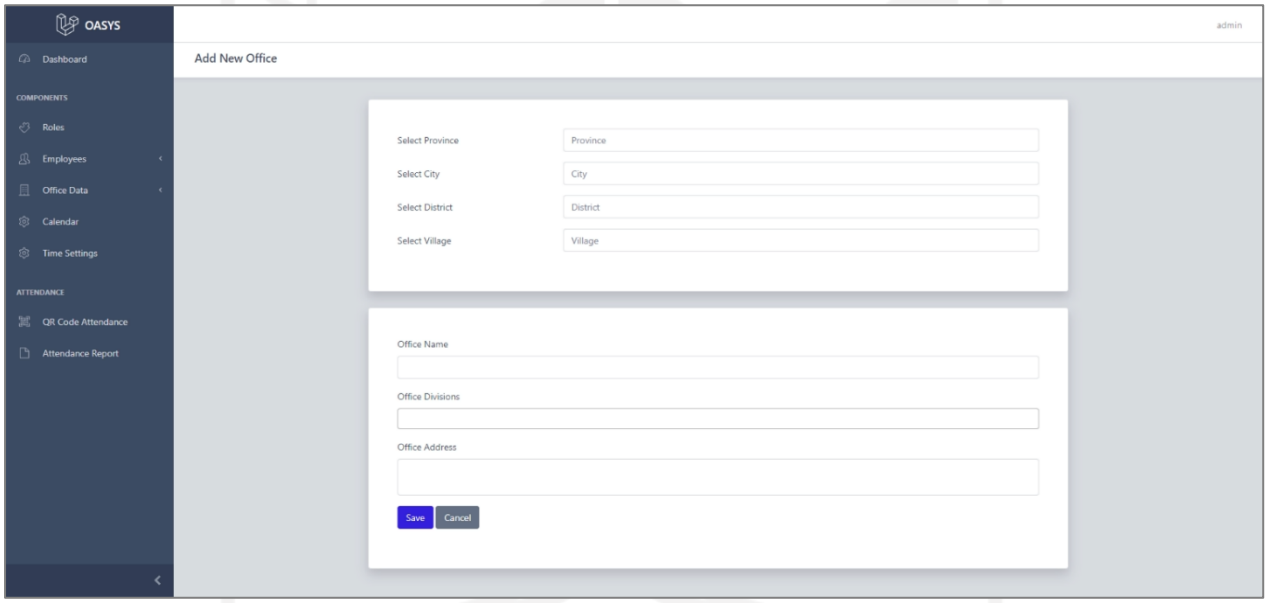


Gambar 3.8 Halaman *Roles List*

f. Pengembangan fitur *CRUD* kantor untuk *administrator*

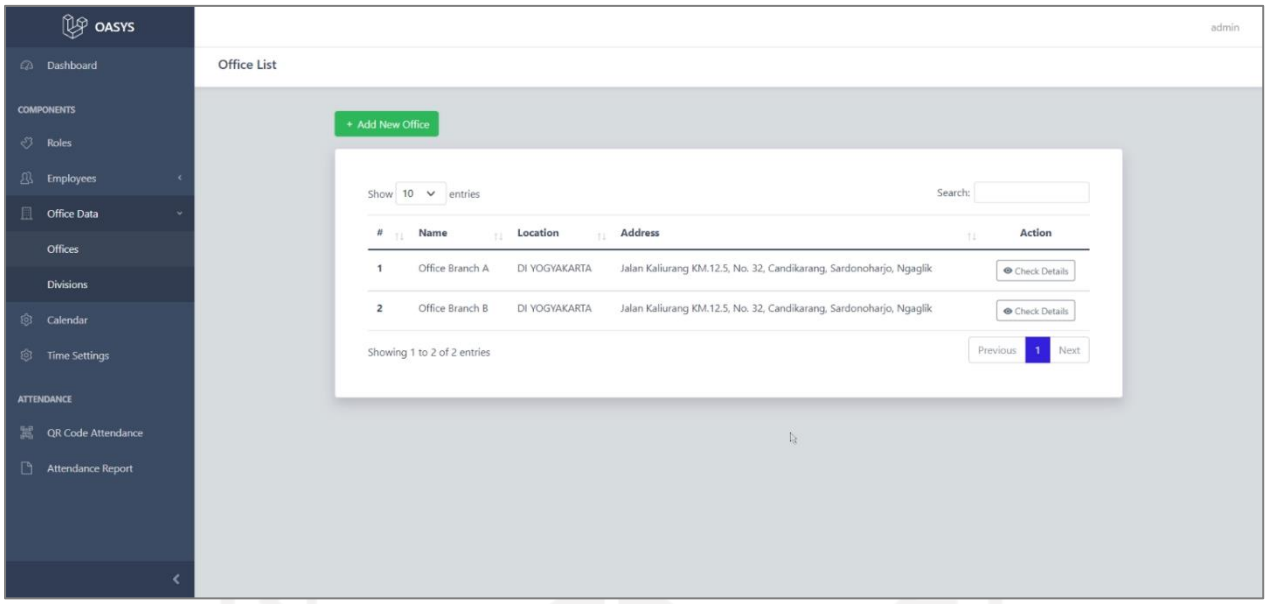
Fitur *administrator* selanjutnya adalah fitur untuk melakukan operasi *Create, Read, Update,* dan *Delete* (CRUD) pada data kantor yang dikembangkan pada tanggal 10 Maret 2021. Data kantor pada sistem ini digunakan untuk mengetahui lokasi di mana kantor tersebut berada, divisi apa saja dan berapa karyawan yang ada pada kantor tersebut. Gambar 3.9 menunjukkan halaman untuk menambah data kantor yang dapat diakses oleh *administrator.*

Untuk menambah data kantor, *administrator* perlu mengisi beberapa *field* yang ada pada *form*, seperti lokasi kantor yang dapat dipilih menggunakan *dependent dropdown,* nama kantor, divisi-divisi yang ada pada kantor, dan alamat kantor. Relasi yang menghubungkan kantor dengan divisi adalah *many to many,* karena satu kantor dapat memiliki banyak divisi dan satu divisi dapat berada di beberapa kantor.



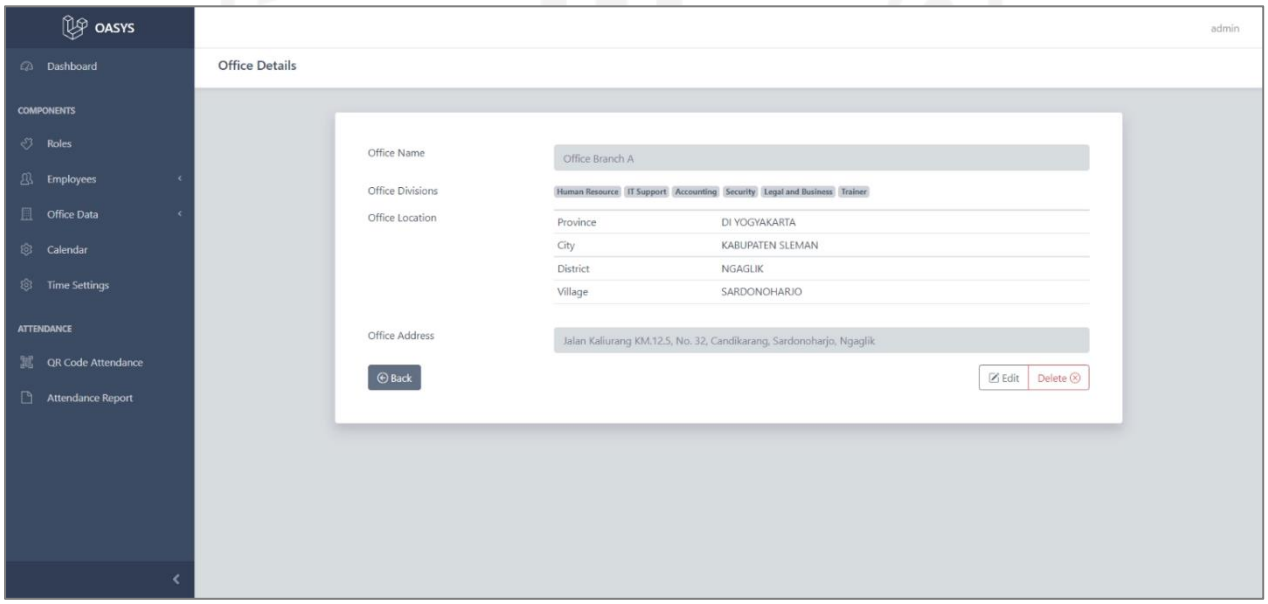
Gambar 3.9 Halaman *Add New Office*

Setelah menambahkan data kantor baru, *administrator* akan dialihkan menuju halaman *Offices List* seperti yang ditunjukkan Gambar 3.10 dimana semua data kantor akan ditampilkan. Data kantor yang ditampilkan pada halaman *Offices List* adalah nama kantor, lokasi kantor, dan alamat tempat kantor tersebut berada.



Gambar 3.10 Halaman *Offices List*

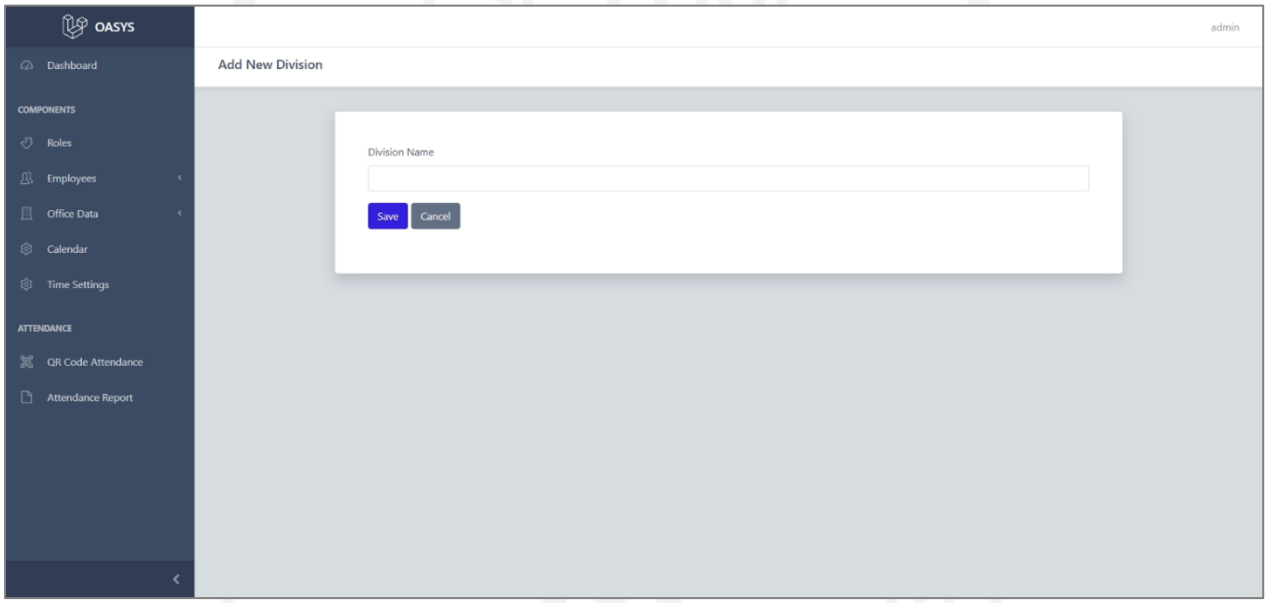
Administrator dapat melihat detail data kantor melalui halaman *Office Details* yang dapat dilihat pada Gambar 3.11. Saat mengakses halaman *Office Details*, *administrator* dapat memilih untuk melakukan operasi *update* atau *delete* terhadap data kantor yang dipilih. Sebelum *administrator* menghapus data kantor, sebuah pesan peringatan akan ditampilkan dalam bentuk *modal.*



Gambar 3.11 Halaman *Office Details*

g. Pengembangan fitur *CRUD* divisi untuk *administrator*

Seperti yang telah disebutkan sebelumnya bahwa data divisi diperlukan *administrator* untuk menambahkan data kantor, maka fitur *Create, Read, Update,* dan *Delete* (CRUD) untuk data divisi perlu untuk dikembangkan. Untuk menambahkan data divisi, *administrator* dapat mengakses halaman *Add New Division* seperti yang ditunjukkan Gambar 3.12 dan mengisi *field* nama untuk divisi. Setelah menambahkan data divisi, data tersebut disimpan ke dalam tabel *division* yang kemudian dapat ditampilkan pada *form* tambah kantor dalam bentuk *dropdown*.



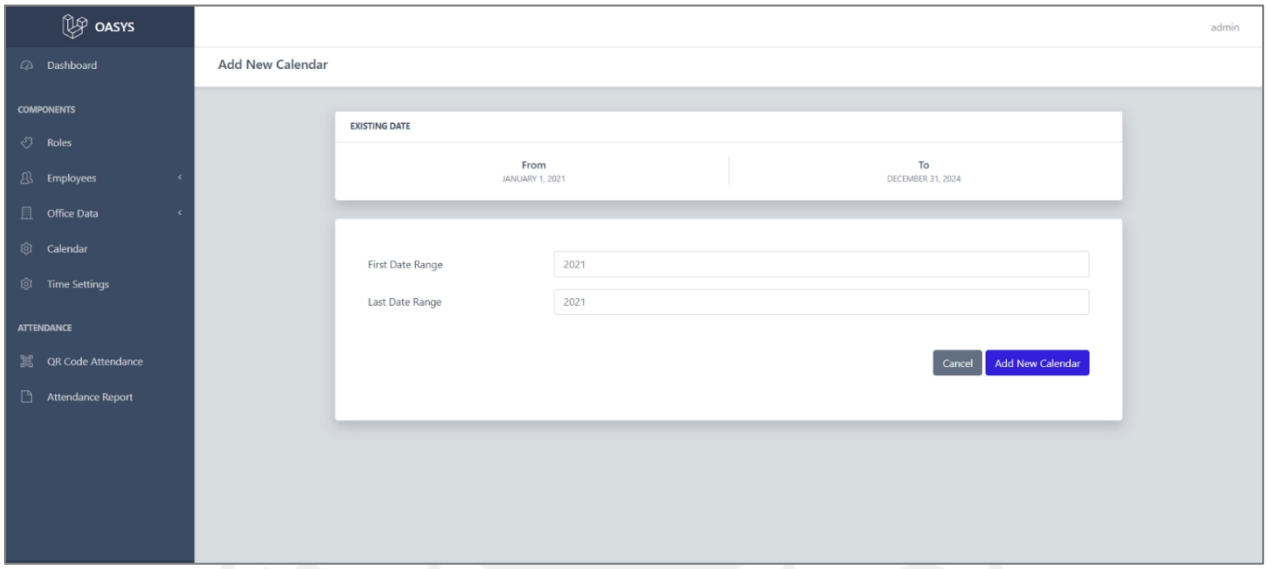
Gambar 3.12 Halaman *Add New Division*

h. Pengembangan fitur manajemen kalender untuk *administrator* dan karyawan divisi *human resource*

Fitur manajemen kalender akan digunakan untuk memanipulasi data kalender. Fitur ini dikembangkan selama 5 hari dari tanggal 15—20 April 2021. Data kalender sendiri akan digunakan untuk mengetahui status hari dan menampilkan status tersebut pada saat karyawan mengajukan kehadiran. Berdasarkan hasil analisis fitur pengguna, fitur manajemen kalender dapat diakses oleh *administrator* dan karyawan yang bekerja pada divisi *human resource.* Proses dari manajemen kalender dimulai dengan pengguna menambahkan data kalender baru menggunakan *form* yang dapat diakses melalui halaman *Add New Calendar* seperti yang ditunjukkan pada Gambar 3.13.

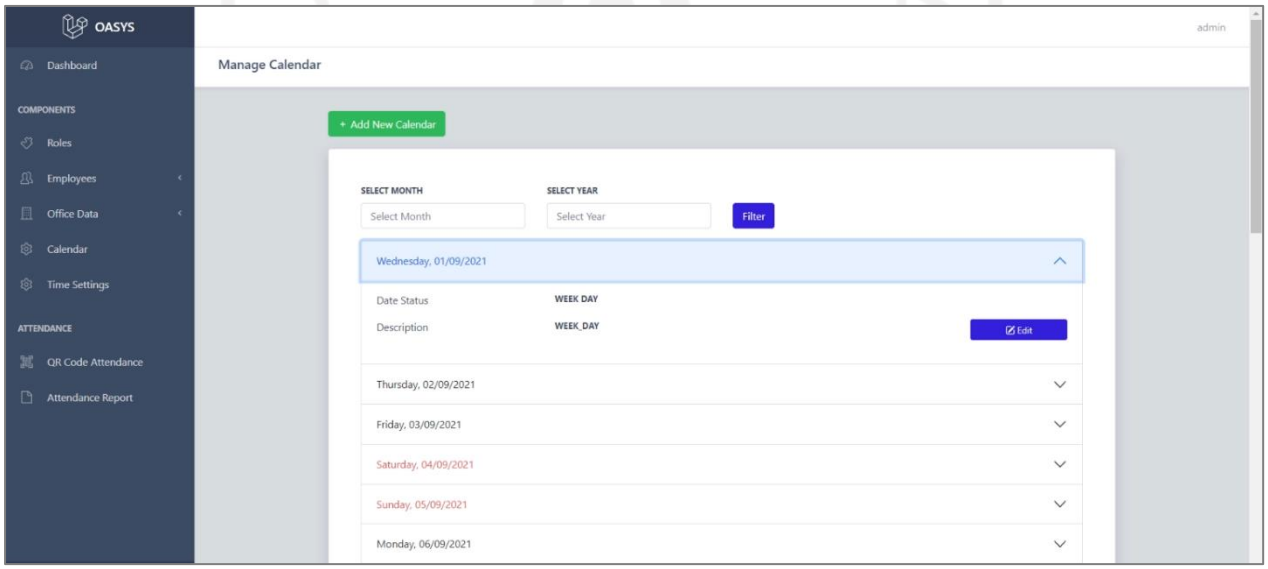
Pada halaman tersebut akan ditampilkan *existing date range* atau rentang tanggal yang sudah ada pada *database* dan sebuah *form* yang berisi dua buah *dropdown* untuk memilih periode tahun

yang diinginkan. *Dropdown* tersebut berisi *option* atau pilihan yang berisi tahun sekarang ditambah lima (5) tahun yang akan datang, misalnya tahun sekarang adalah 2021, maka pilihan yang ditampilkan pada *dropdown* tersebut adalah 2021, 2022, 2023, 2024, 2025, dan 2026.



Gambar 3.13 *Form* pada halaman *Add New Calendar*

Pengguna dapat memperbarui data kalender menggunakan *form* tersebut. Perlu diketahui bahwa saat pengguna memperbarui data kalender, maka data kalender akan bertambah pada *database*, tidak menghapus data kalender yang sudah ada. Proses manajemen kalender kemudian akan dilanjutkan dengan pengguna memperbarui status hari pada data kalender yang dapat diakses melalui halaman *Manage Calendar* seperti yang ditunjukkan pada Gambar 3.14.

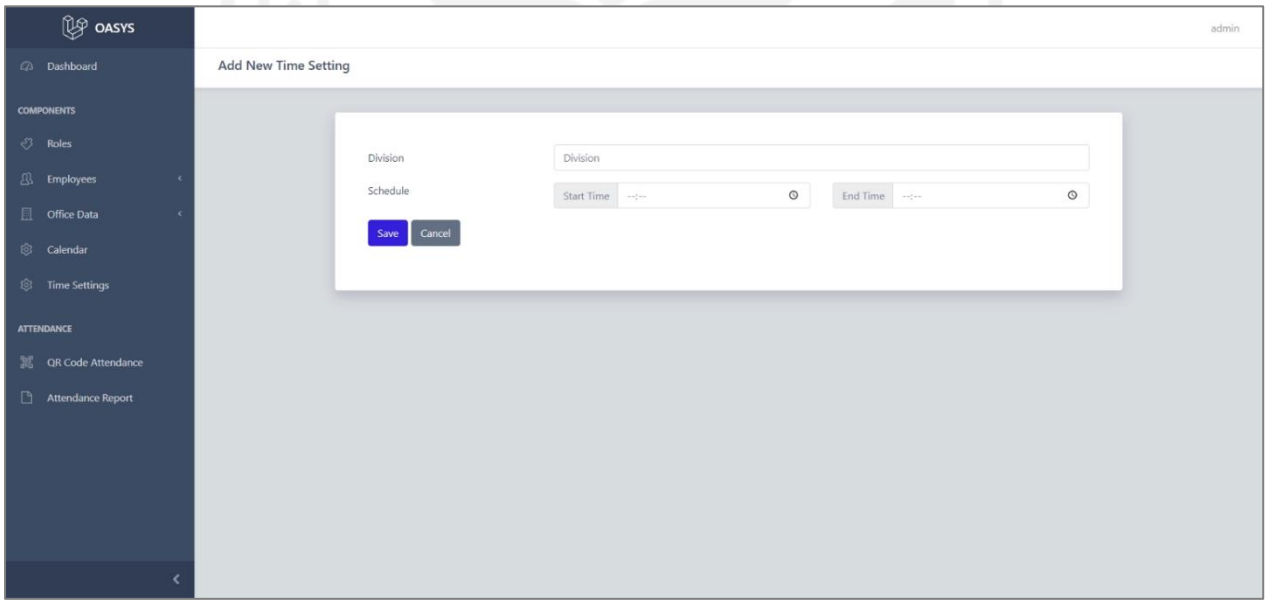


Gambar 3.14 Halaman *Manage Calendar*

Pada halaman ini data kalender ditampilkan dalam bentuk *accordion* dan pengguna dapat memilih bulan dan tahun menggunakan *filter*. *Administrator* atau karyawan *human resource* dapat memperbarui status hari menggunakan tombol *edit* yang kemudian akan menampilkan sebuah *modal* yang berisi *form* untuk memperbarui status hari pada tanggal tersebut. *Form* yang ada pada *modal* memiliki sebuah *input text* untuk deskripsi status hari dan sebuah *dropdown* yang berisi pilihan status hari, yaitu *WEEK_DAY, WEEK_END,* dan *HOLIDAY.*

i. Pengembangan fitur *CRUD time settings* untuk divisi

Fitur *administrator* selanjutnya adalah fitur untuk melakukan operasi CRUD terhadap data *time settings* atau jam kerja divisi. Fitur CRUD *time settings* untuk divisi dikembangkan pada tanggal 12 Maret 2021. Pada tahap analisis fitur pengguna yang telah dilakukan, jam kerja divisi akan digunakan untuk mengetahui apakah pada saat karyawan mengajukan presensi ada di dalam jam kerja atau tidak. Jika presensi diajukan tidak pada jam kerja, maka pada *form* pengajuan presensi akan ditampilkan pesan bahwa karyawan sedang tidak berada di dalam jam kerja. *Administrator* dapat menambahkan data jam kerja baru pada halaman *Add New Time Setting* seperti yang ditunjukkan pada Gambar 3.15.

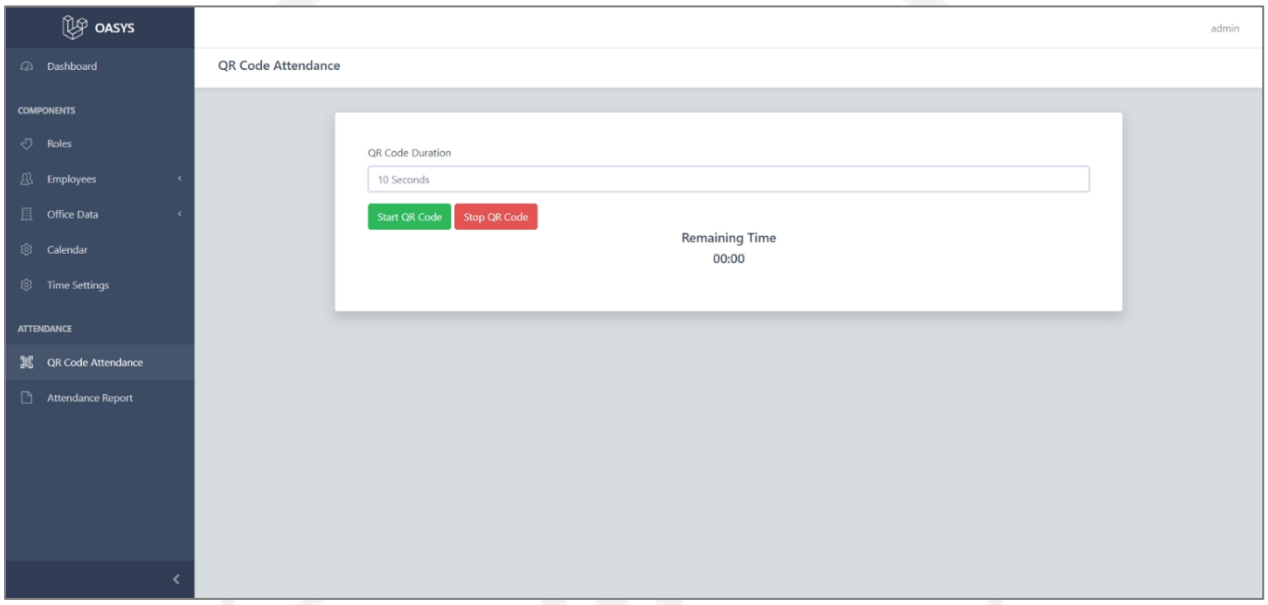


Gambar 3.15 Halaman *Add New Time Setting*

Setelah berhasil menambahkan jam kerja baru, *administrator* akan dialihkan ke halaman *Time Settings List* yang menampilkan semua data jam kerja yang digunakan pada divisi-divisi yang ada di dalam sistem. Selain itu, data tersebut dapat digunakan pada *dependent dropdown* yang ada pada *form* tambah karyawan baru.

j. Pengembangan fitur *generate QR Code* untuk *administrator*

QR Code pada sistem ini akan digunakan karyawan untuk mengajukan presensi melalui perangkat *mobile*. Fitur ini dikembangkan pada tanggal 21—25 April 2021. Bagi pengguna dengan *role administrator*, fitur untuk membuat *QR Code* akan tersedia pada halaman *QR Code Attendance* seperti yang ditunjukkan pada Gambar 3.16. *Administrator* dapat memilih durasi waktu *QR Code* (dalam satuan detik) menggunakan *dropdown* yang ada pada halaman tersebut dan memulai *QR Code* menggunakan tombol *Start QR Code.*



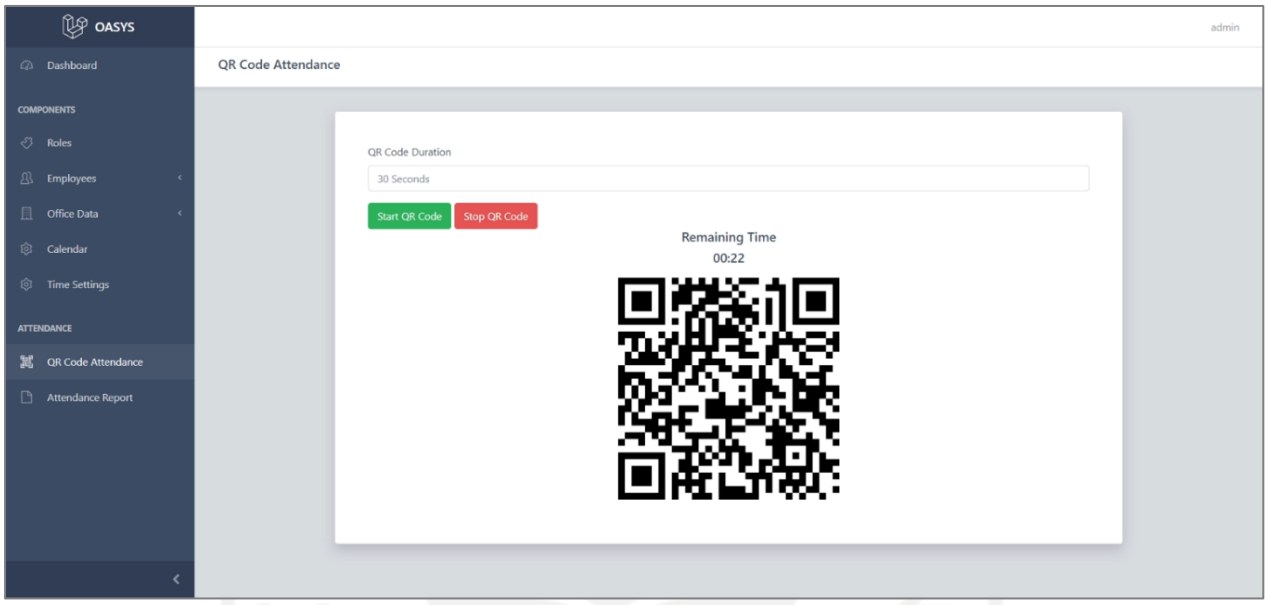
Gambar 3.16 Halaman *QR Code Attendance*

Pada saat tombol *Start QR Code* diklik sistem akan membuat sebuah *QR Code* yang berisi token yang terdiri dari lima huruf acak menggunakan kode yang dapat dilihat pada Gambar 3.17. Token tersebut kemudian akan disimpan ke dalam tabel *qr_tokens* dan dapat digunakan untuk mengecek validitas token pada *QR Code* yang dipindai oleh karyawan.

```
$token = '';
for ($i = 0; $i < 5; $i++) {
    $token .= chr(rand(ord('a'), ord('z')));
}
```

Gambar 3.17 Kode yang digunakan untuk membuat token

Selain itu, sebuah *countdown timer* akan berjalan sesuai dengan durasi waktu yang telah dipilih dan *QR Code* akan dibuat ulang secara otomatis setiap *timer* menyentuh angka 0. Proses ini akan terus dilakukan berulang-ulang sampai *administrator* menekan tombol *Stop QR Code* yang akan mengosongkan tabel *qr_tokens*. Pada Gambar 3.18, dapat dilihat *QR Code* yang sedang berjalan.

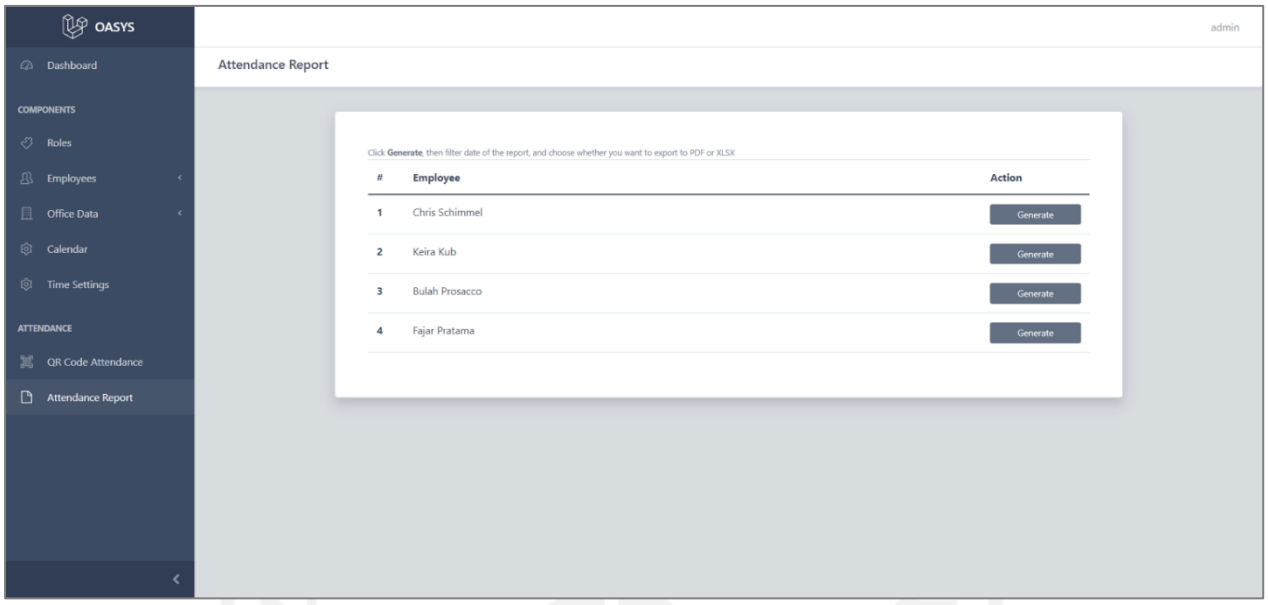


Gambar 3.18 Tampilan *QR Code* yang sedang berjalan

k. Pengembangan fitur *generate report* kehadiran karyawan

Berdasarkan hasil analisis fitur yang telah dilakukan, fitur *administrator* yang terakhir adalah fitur untuk membuat laporan bulanan kehadiran karyawan. Pengembangan fitur ini dilaksanakan dari tanggal 29 April 2021 sampai dengan tanggal 7 Mei 2021. *Administrator* dapat membuat laporan dengan mengakses halaman *Attendance Report* yang dapat dilihat pada Gambar 3.19. Proses pembuatan laporan bulanan dimulai dengan *administrator* memilih karyawan dari daftar karyawan kemudian menekan tombol *Generate.*

Pada saat tombol tersebut ditekan, sebuah *modal* yang berisi pilihan bulan, tahun, dan format laporan akan ditampilkan. *Administrator* dapat menggunakan *filter* bulan dan tahun untuk mendapatkan laporan bulanan yang diinginkan serta format laporan yang sesuai dengan kebutuhan. Jika format yang dipilih adalah *PDF*, maka laporan akan ditampilkan pada *browser*. Sedangkan jika format yang dipilih adalah *XLSX* atau *spreadsheet*, laporan akan langsung diunduh ke dalam komputer.



Gambar 3.19 Tampilan halaman *Attendance Report*

Data-data yang ditampilkan pada laporan kehadiran karyawan di antaranya adalah data karyawan seperti nama karyawan, posisi atau jabatan karyawan, divisi dan kantor tempat karyawan tersebut bekerja, bulan dari laporan tersebut, dan *monthly attendance count* atau total kehadiran karyawan selama bulan tersebut. Laporan kehadiran karyawan memiliki dua buah tabel, yaitu tabel *attendance report* dan tabel *overtimes report.* Tabel *attendance report* akan menampilkan detail kehadiran karyawan pada setiap hari kerja di bulan tersebut. Data-data yang akan ditampilkan pada tabel ini di antaranya adalah tanggal, *task plan* atau tugas yang dikerjakan pada tanggal tersebut, catatan kehadiran, dan tipe kehadiran. Tabel *attendance report* dikhususkan untuk mendata presensi, izin, dan cuti karyawan pada bulan tersebut. Kolom *task plan* akan dikosongkan jika pada tanggal tersebut karyawan mengajukan izin atau cuti.

Tabel kedua, yaitu tabel *overtimes report* akan menampilkan data-data terkait pengajuan lembur yang dilakukan oleh karyawan, data-data tersebut di antaranya adalah tanggal lembur, *task plan* atau tugas yang dikerjakan pada saat lembur, jam mulai dan jam selesai, serta durasi lembur yang diajukan karyawan. Total durasi lembur karyawan kemudian akan ditampilkan pada bagian *monthly attendance count* yang telah disebutkan sebelumnya. Contoh dari laporan bulanan kehadiran karyawan dapat dilihat pada Gambar 3.20.

Employee Name: Fajar Pratama
Position: Manager
Division: Human Resource
Office: Office Branch B
Report Month: **September**

Monthly Attendance Count
Attendance: 6 Days
Absent: 1 Day
Overtime: 3 Hours
Leave: 5 Days

Attendance Report

No	Date	Task Plan	Note	Type
1	Wed, 01-09-2021	add dynamic field for task plan	late attendance	Attend
2	Thu, 02-09-2021	test new attendance approval flow fetch user's geolocation	early clock-out	Attend
3	Fri, 03-09-2021	Lorem ipsum dolor sit amet	consectetur adipiscing elit	Attend
4	Saturday			
5	Sunday			
6	Mon, 06-09-2021		sick	Absent
7	Tue, 07-09-2021	sunt in culpa qui officia deserunt mollit anim id est laborum	none	Attend
8	Wed, 08-09-2021	eaque ipsa quae	none	Attend
9	Thu, 09-09-2021		sick leave	Leave
10	Fri, 10-09-2021		sick leave	Leave
11	Saturday		sick leave	Leave
12	Sunday		sick leave	Leave
13	Mon, 13-09-2021		sick leave	Leave
14	Tue, 14-09-2021	At vero eos et accusamus	et iusto odio dignissimos ducimus	Attend
15	Wed, 15-09-2021			
16	Thu, 16-09-2021			
17	Fri, 17-09-2021			
18	Saturday			
19	Sunday			
20	Mon, 20-09-2021			
21	Tue, 21-09-2021			
22	Wed, 22-09-2021			
23	Thu, 23-09-2021			
24	Fri, 24-09-2021			
25	Saturday			
26	Sunday			
27	Mon, 27-09-2021			
28	Tue, 28-09-2021			
29	Wed, 29-09-2021			
30	Thu, 30-09-2021			

Overtimes Report

No	Date	Task Plan	Start Time	End Time	Duration (Hours)
1	Sat, 04-09-2021	Ut enim ad minim veniam	08:45:00	10:45:00	2
2	Sun, 05-09-2021	Excepteur sint occaecat cupidatat non proident	08:45:00	09:45:00	1

Gambar 3.20 Contoh laporan dalam format *PDF*

l. Pembuatan *service class* untuk *controller* dan *API controller*

Sistem ini memiliki banyak *controller* dan *API controller* yang berfungsi untuk menangkap *request* dan memberikan *response* yang sesuai kepada pengguna. Agar *controller* mudah dibaca, proses *business logic* sistem seharusnya tidak dilakukan pada *controller*. Untuk meningkatkan *code readibility* dan *code reusability*, sebuah *service class* yang bertugas untuk memproses *business logic* sistem dibuat untuk setiap *controller*. Pembuatan *service class* dan modifikasi *controller* dilakukan pada tanggal 30 Maret 2021, ketika *API* mulai dikembangkan dan *method* yang ada pada *controller* mulai menjadi kompleks. Laravel tidak memiliki *command* khusus untuk membuat *service class*, sehingga *service class* tidak dapat dibuat menggunakan *artisan command.* Tidak ada aturan khusus terkait penempatan *service class* pada Laravel, pada sistem ini *service class* disimpan pada folder app\Http\Services. Salah satu *method* pada *service class* yang digunakan pada sistem ini adalah *method getAttendance* pada *AttendanceService* yang dapat dilihat pada Gambar 3.21.

```
public function getAttendance()
{
    $userId = auth()->id();
    return Attendance::query()
        ->where(user_id, $userId)
        ->latest()
        ->get();
}
```

Gambar 3.21 Contoh *method getAttendance* pada *AttendanceService*

Service class dapat digunakan oleh *controller* menggunakan *dependency injection.* Contoh penggunaan *service class* pada sistem ini adalah penggunaan *method getAttendance* pada *method index* yang terdapat di *AttendendanceController* dan *API/AttendanceController* yang dapat dilihat pada Gambar 3.22 dan Gambar 3.23.

```
public function index()
{
    $attendances = $this->attendanceService->getAttendance();
    return view('user.attendance.index', compact('attendances'));
}
```

Gambar 3.22 *Method index* pada *AttendanceController*

```
public function index()
{
    return response()->json($this->attendanceService->getAttendance(), 200);
}
```

Gambar 3.23 *Method index* pada *API/AttendanceController*

Dapat dilihat pada Gambar 3.22 dan Gambar 3.23 *method getAttendance* digunakan oleh dua buah *controller*. Penggunaan *service class* membuat kode yang sama dapat digunakan oleh beberapa *controller* sekaligus sehingga kode yang sama tidak perlu ditulis berulang-ulang. Penggunaan *service class* juga mempermudah proses pengembangan sistem jika terdapat perubahan kode karena tidak perlu merubah kode yang sama pada tempat yang berbeda.

m. Implementasi Laravel Sanctum untuk sistem autentikasi *Rest API*

Laravel Sanctum adalah sebuah *package* Laravel yang digunakan untuk mengautentikasi seorang pengguna yang mengakses *API*. Laravel Sanctum pada sistem ini digunakan sebagai sistem autentikasi *Rest API* yang akan digunakan oleh aplikasi *mobile*. *Package* ini diimplementasikan ke dalam sistem pada tanggal 31 Maret 2021.

Langkah pertama yang dilakukan pada proses implementasi Laravel Sanctum adalah instalasi yang dapat dilakukan dengan menggunakan perintah *"composer require laravel/sanctum"*. Setelah instalasi, Laravel Sanctum akan membuat sebuah *file migration* baru yang akan digunakan untuk membuat tabel *personal_access_tokens*. Tabel ini akan digunakan untuk menyimpan *access token* milik pengguna. Agar Laravel Sanctum dapat digunakan untuk membuat *access token*, model *user* perlu dimodifikasi menggunakan kode seperti yang ditunjukkan oleh Gambar 3.24.

```
use Laravel/Sanctum/HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
}
```

Gambar 3.24 Penambahan kode pada model *user*

Ketika pengguna melakukan proses *login*, maka Laravel Sanctum akan membuat sebuah token dan menyimpannya ke dalam tabel *personal_access_tokens*. Tampilan dari tabel *personal_access_tokens* dapat dilihat pada Gambar 3.25.

Gambar 3.25 Tampilan tabel *personal_access_tokens*

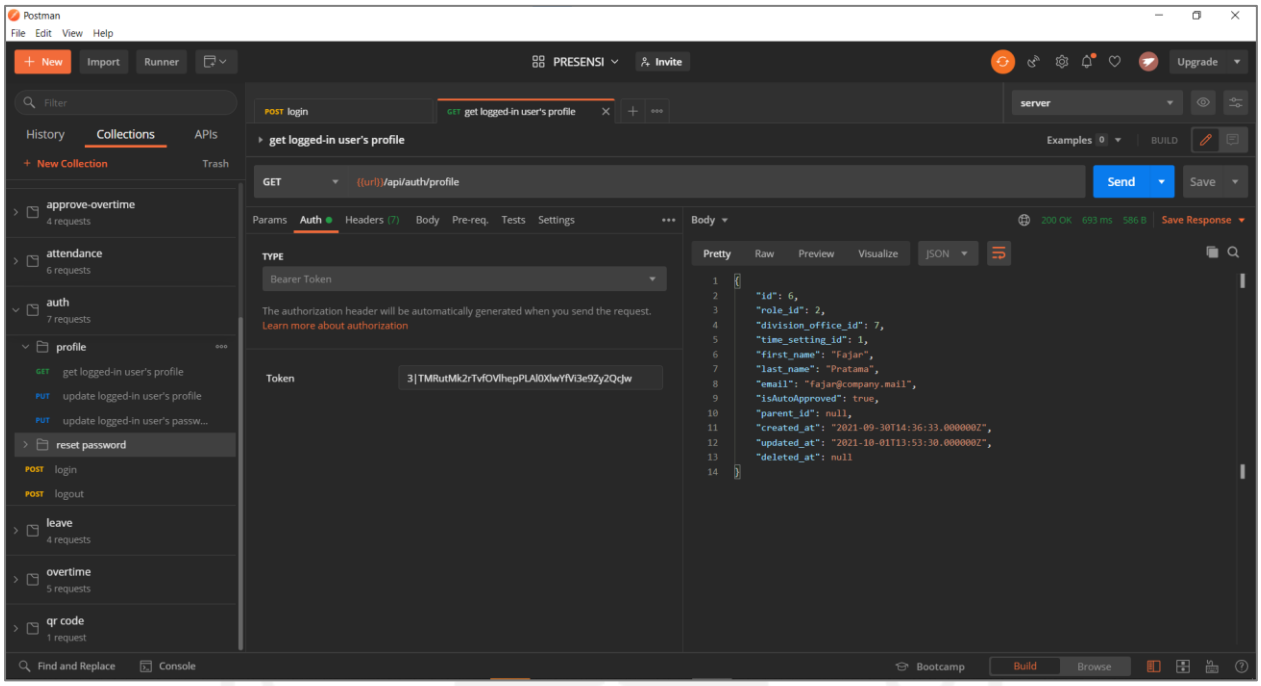
Tabel *personal_access_tokens* memiliki beberapa kolom, di antaranya adalah *tokenable_type, tokenable_id, name, token,* dan *last_used_at.* Kolom *tokenable_type* pada tabel tersebut berisi model apa pada aplikasi yang menggunakan Laravel Sanctum, pada sistem ini kolom tersebut berisi model *user*. Sedangkan kolom *tokenable_id* adalah *id* dari *user* atau pengguna yang bersangkutan. Sebagai contoh, dapat dilihat pada Gambar 3.24 di atas *model* yang menggunakan Laravel Sanctum adalah model *user,* dengan pengguna yang memiliki *id* 6. Kolom *name* pada tabel akan berisi nama token yang digunakan pada saat proses pembuatan token. Tidak ada aturan khusus terkait penamaan token pada dokumentasi Laravel Sanctum, maka dari itu pada sistem ini nama token yang digunakan adalah *api_token*. Kode untuk membuat token digunakan pada *method login* yang ditunjukkan oleh Gambar 3.26

```
public function login(Request $request)
{
    $data = $request->validate([
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string', 'min:8']
    ]);

    if (!Auth::attempt($data)) {
        return response()->json([
            'message' => 'Credentials does not match!'
        ], 401);
    }

    return response()->json([
        'message' => 'Logged In Successfully',
        'token' => auth()->user()->createToken('api_token')->plainTextToken
    ]);
}
```

Gambar 3.26 Kode yang digunakan untuk membuat token

Kolom lain yang terdapat pada tabel *personal_access_tokens* adalah kolom token yang akan digunakan untuk menyimpan token yang dienkripsi menggunakan SHA-256. Saat pengguna melakukan *API request* kepada aplikasi, token ini akan digunakan pada *Authorization Header* sebagai *Bearer Token* untuk mengautentikasi pengguna tersebut. Kolom *last_used_at* yang terdapat pada tabel *personal_access_tokens* akan digunakan untuk menyimpan *timestamp* kapan token tersebut terakhir kali digunakan.

Setelah token tersimpan, pengguna dapat menggunakan token tersebut untuk mengakses *resource* atau data yang dilindungi oleh *middleware auth:sanctum.* Pada Gambar 3.27, pengujian autentikasi pengguna pada *Rest API* sistem presensi dilakukan dengan mengakses *url* 127.0.0.1:8000/api/auth/profile menggunakan *Postman.*



Gambar 3.27 Pengujian autentikasi pengguna menggunakan *Postman*

Saat pengguna selesai melakukan *request* terhadap *resource* yang ada pada sistem, pengguna dapat melakukan *logout.* Ketika pengguna melakukan *logout*, maka token yang dimilikinya akan dihapus dari tabel *personal_access_tokens.* Kode yang digunakan untuk menangani proses *logout* dapat dilihat pada Gambar 3.28.

```
public function logout()
{
    auth()->user()->tokens()->delete();
    return response()->json([
        'message' => 'Token Revoked!'
    ]);
}
```

Gambar 3.28 Kode yang digunakan untuk menangani proses *logout*

n. Pengembangan fitur *CRUD* kehadiran untuk karyawan

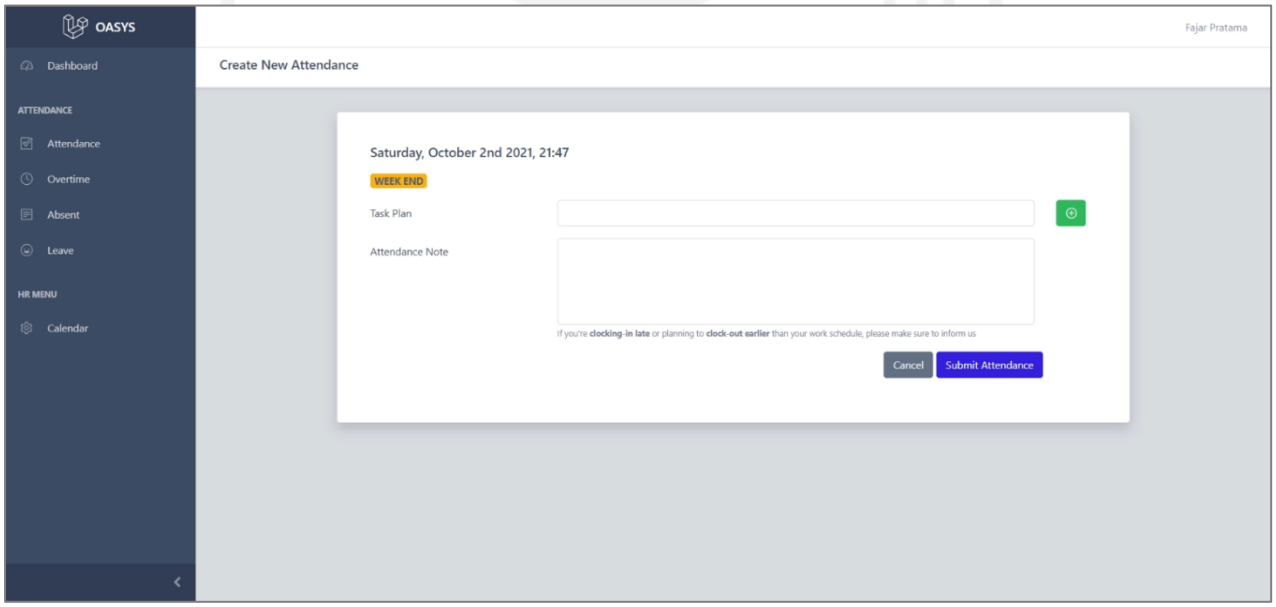
Setelah fitur-fitur *administrator* selesai dikembangkan, proses pengembangan sistem dilanjutkan dengan mengembangkan fitur-fitur untuk karyawan. Fitur-fitur yang dikembangkan untuk karyawan di antaranya adalah pengajuan presensi, lembur, izin, dan cuti. Fitur-fitur ini dapat digunakan baik pada aplikasi berbasis *web* ataupun perangkat *mobile.*

Ada beberapa jenis kehadiran pada sistem ini, yang pertama adalah *attendance*/presensi. Presensi merupakan kehadiran harian yang diajukan oleh karyawan pada hari dan jam kerja. Jenis kehadiran yang kedua adalah *overtime*/lembur yang diajukan oleh karyawan di luar hari dan jam

kerja. Berbeda dengan presensi, untuk mengajukan lembur karyawan perlu memilih durasi waktu lembur di antara 1 sampai dengan 3 jam. Selain presensi dan lembur, pengajuan izin seperti izin sakit dan cuti juga dapat diajukan menggunakan sistem ini. Akan tetapi, pengajuan izin sakit dan izin karena keperluan mendesak akan dibedakan dengan izin cuti, karena pada sistem ini izin akan disebut sebagai *absent*, sedangkan cuti akan disebut sebagai *leave*.

Fitur pengajuan presensi adalah salah satu fitur untuk karyawan yang dikembangkan pada tanggal 16—18 Maret 2021. Selama pengembangan sistem presensi, fitur ini beberapa kali mengalami perubahan sesuai dengan hasil konsultasi yang dilakukan dengan *project manager*. Perubahan pertama pada pengembangan fitur ini dilaksanakan pada tanggal 24-26 Maret 2021, sedangkan perubahan kedua dilaksanakan pada tanggal 1 April 2021.

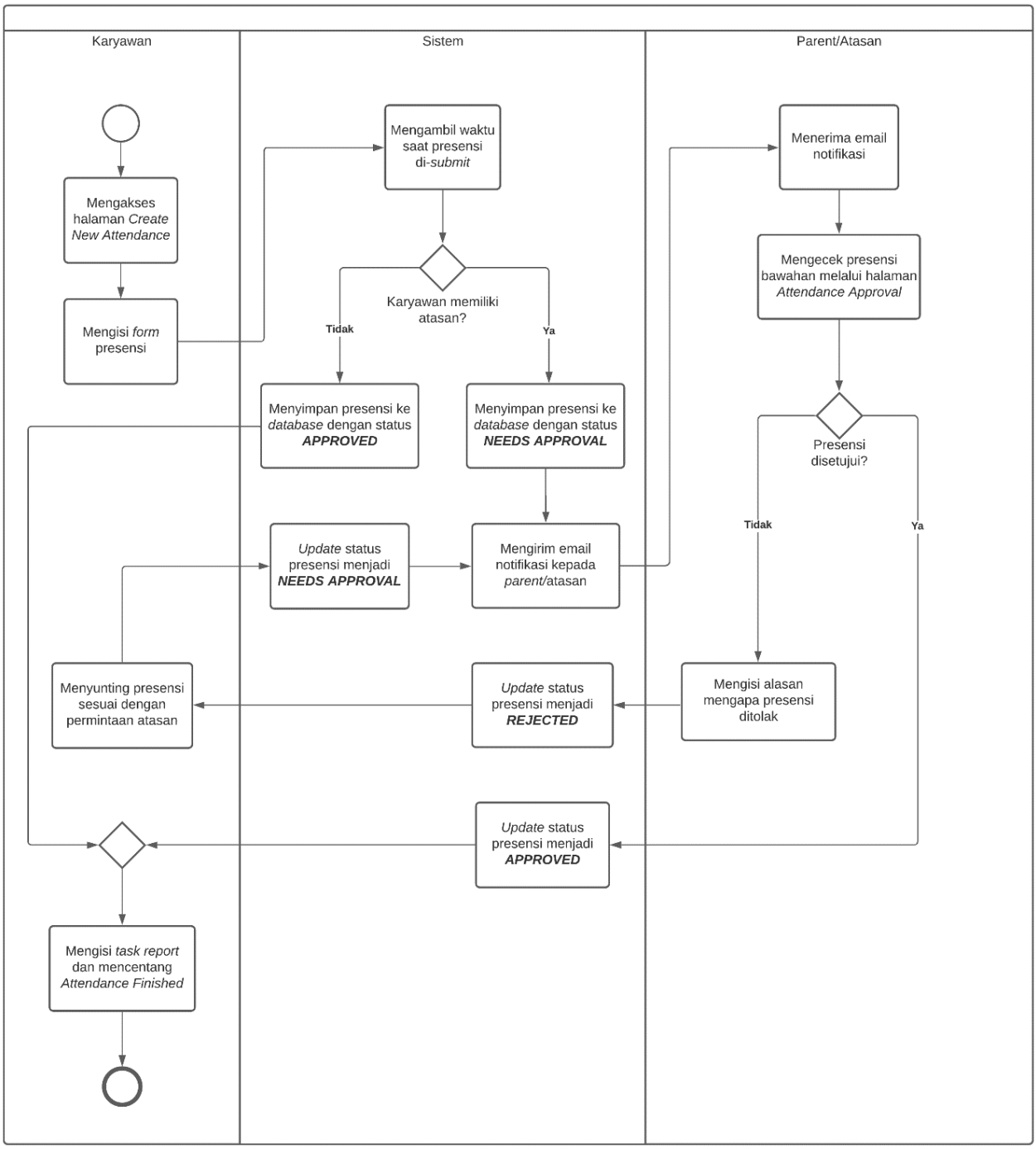
Pada fitur ini, karyawan dapat mengakses *form* pengajuan presensi pada halaman *Create New Attendance* yang dapat dilihat pada Gambar 3.29. *Form* pengajuan presensi memiliki dua *input field*, yang pertama adalah *field task plan* yang dibuat menggunakan *dynamic form*, sehingga karyawan dapat menambahkan beberapa *task plan* sekaligus dalam sebuah presensi. *Field* yang kedua adalah *attendance note* yang dapat diisi oleh karyawan mengenai catatan kehadiran, seperti jika karyawan hadir *clock-in* atau hadir terlambat maupun *clock-out* atau pulang lebih awal. Selain itu, keterangan tanggal dan jam kerja akan ditampilkan pada *form* tersebut.



Gambar 3.29 Tampilan *form* pada halaman *Create New Attendance*

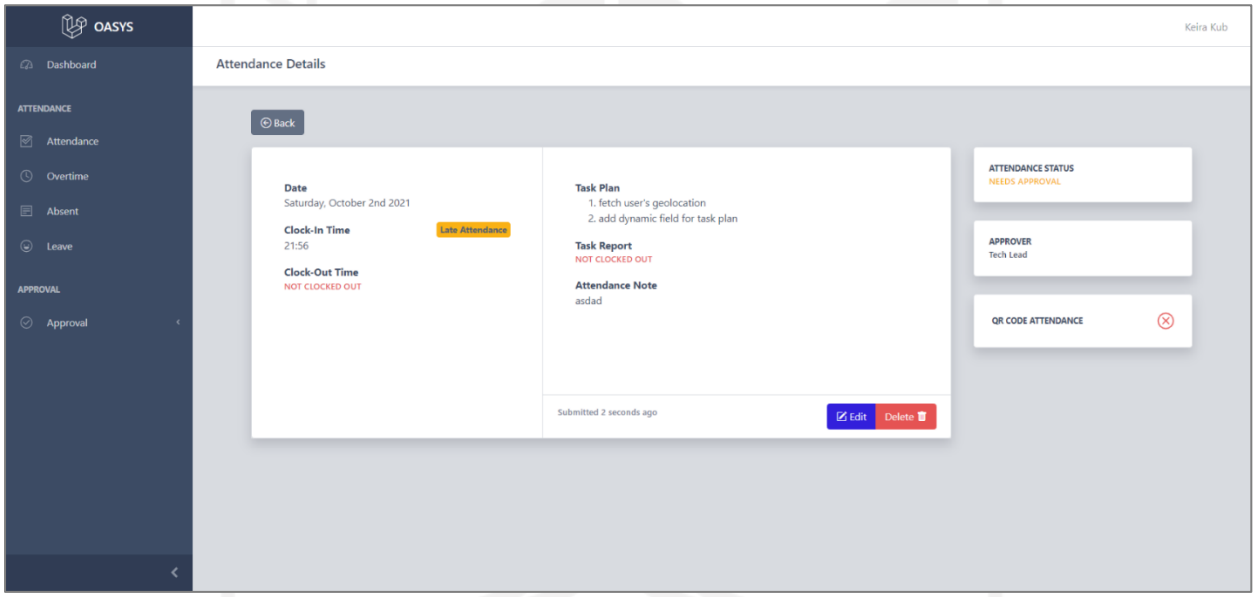
Pada saat karyawan berhasil mengajukan presensi, sebuah *email* akan dikirimkan kepada *parent* atau atasan jika karyawan tersebut memiliki atasan. Presensi tersebut juga memiliki status *needs approval*, yang berarti presensi tersebut perlu untuk disetujui oleh atasan. Untuk mendukung

penjelasan terkait proses pengajuan dan *approval* presensi, sebuah *flowchart* yang menggambarkan proses tersebut dapat dilihat pada Gambar 3.30 di bawah ini.



Gambar 3.30 *Flowchart* proses pengajuan dan *approval* presensi

Setelah mengajukan presensi, karyawan dapat melihat detail presensi yang diajukan pada halaman *Attendance Details* yang dapat dilihat pada Gambar 3.31. Detail presensi yang terdapat pada halaman *Attendance Details* di antaranya adalah tanggal, *clock-in time, clock-out time, task plan* atau tugas yang dikerjakan, *task report* atau laporan pengerjaan tugas, catatan kehadiran, *approval status*, *parent* atau atasan karyawan tersebut, dan status apakah presensi tersebut merupakan presensi yang diajukan menggunakan *QR Code* atau tidak. Presensi dinilai tepat waktu apabila diajukan pada jam mulai kerja yang sesuai dengan toleransi keterlambatan 15 menit. Sedangkan jika presensi diajukan melewati waktu tersebut, maka sebuah teks yang memberitahu bahwa presensi tersebut terlambat akan ditampilkan. Selain itu, jika presensi tidak mendapatkan persetujuan dari atasan, maka *rejection note* atau alasan mengapa presensi tersebut ditolak akan ditampilkan pada halaman tersebut.

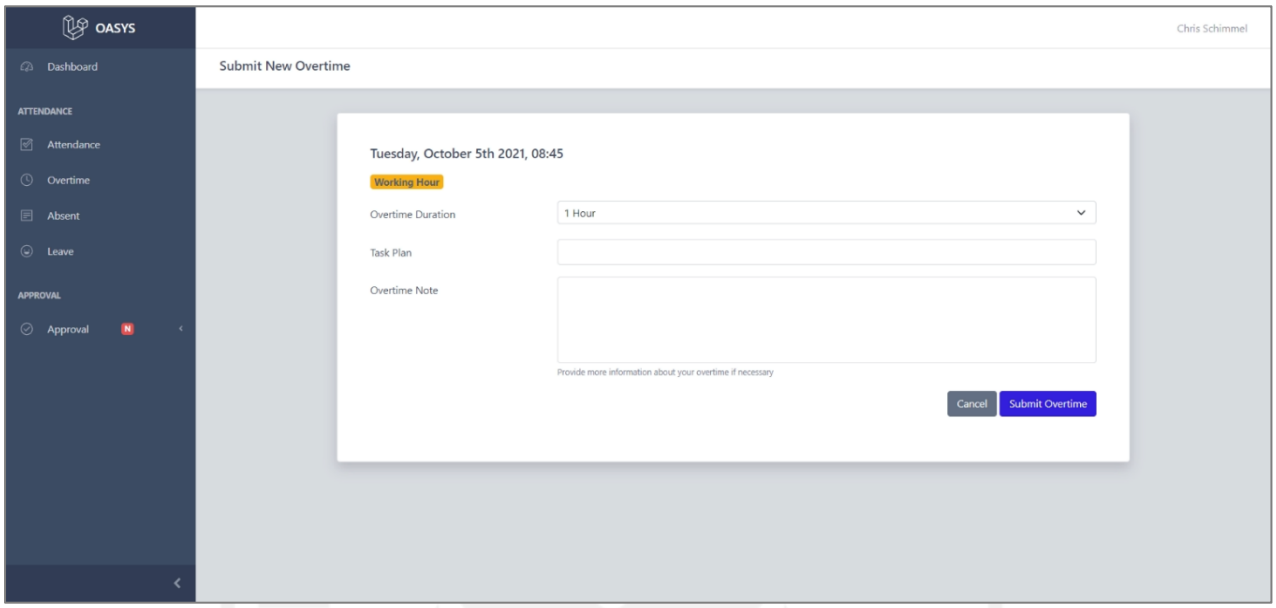


Gambar 3.31 Halaman *Attendance Details*

Fitur karyawan yang kedua adalah pengajuan lembur yang dikembangkan pada tanggal 29 Maret 2021 dan dilanjutkan pada tanggal 5—6 April 2021. Untuk mengajukan lembur, karyawan dapat mengakses halaman *Submit New Overtime. Form* yang terdapat pada halaman *Submit New Overtime* memiliki sebuah *dropdown* yang berisi pilihan durasi lembur, *task plan* atau tugas yang akan dikerjakan, dan catatan lembur yang bersifat opsional (Gambar 3.32).

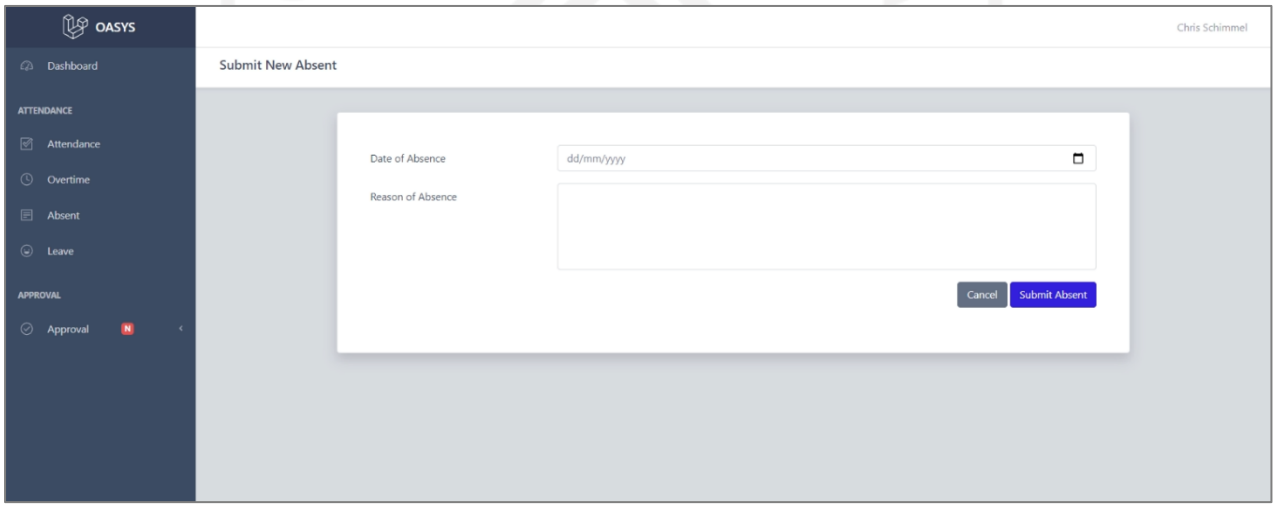
Perlu diketahui bahwa lembur hanya boleh diajukan di luar jam kerja. Jika karyawan mengakses halaman *Submit New Overtime* pada jam kerja, sebuah *label* yang bertuliskan *Working Hour* akan ditampilkan. Proses pengajuan lembur kurang lebih sama seperti pengajuan presensi, lembur yang diajukan kemudian akan membutuhkan persetujuan dari atasan karyawan yang

bersangkutan untuk mengetahui apakah *task* tersebut perlu untuk dikerjakan pada waktu lembur atau tidak.



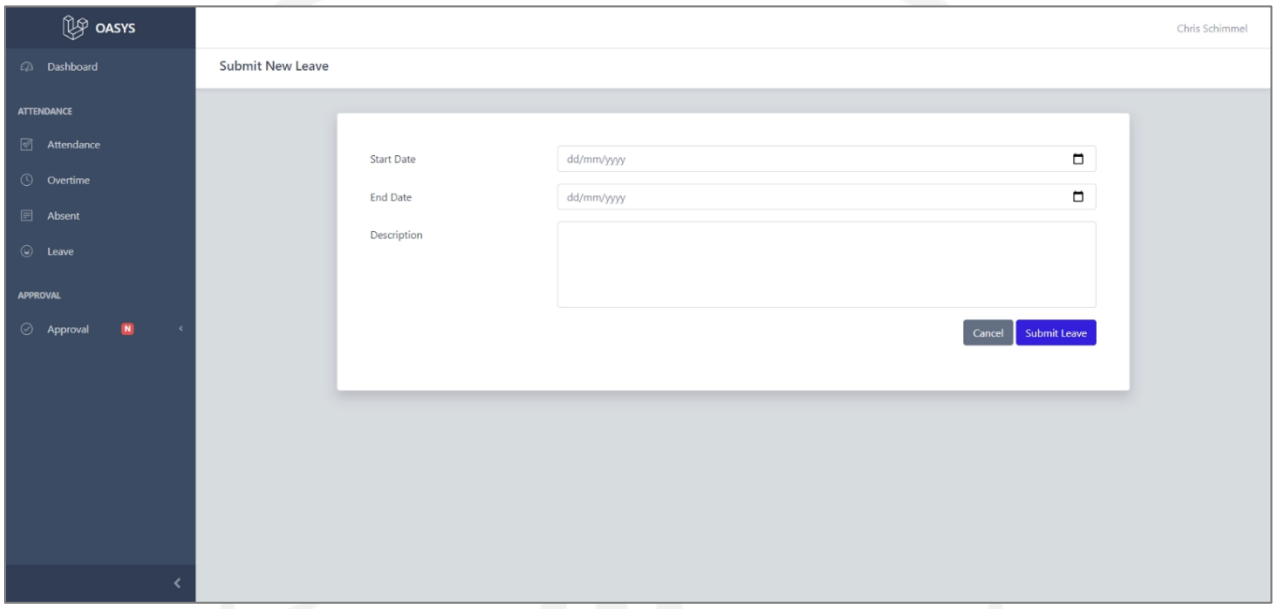
Gambar 3.32 *Form* pada halaman *Submit New Overtime*

Fitur karyawan yang selanjutnya adalah fitur untuk mengajukan izin. Izin diajukan saat karyawan tidak dapat hadir pada hari kerja karena sakit atau jika ada keadaan yang mendesak. Fitur ini dikembangkan selama 2 hari terhitung dari tanggal 7—8 April 2021. Untuk mengajukan izin, karyawan dapat mengakses *form* pengajuan izin yang terdapat pada halaman *Submit New Absent* dan mengisi beberapa *field* yang ada seperti tanggal pengajuan izin dan alasan izin yang dapat dilihat pada Gambar 3.33.



Gambar 3.33 *Form* pada halaman *Submit New Absent*

Pengembangan fitur CRUD untuk karyawan yang terakhir adalah fitur untuk mengajukan cuti. Pengembangan fitur ini juga dilaksanakan selama 2 hari dari tanggal 8—9 April 2021. Karyawan dapat mengajukan cuti dengan mengakses *form* yang terdapat pada halaman *Submit New Leave* yang ditunjukkan Gambar 3.34. *Form* pengajuan cuti memiliki beberapa *input field* seperti tanggal mulai, tanggal selesai, dan deskripsi untuk memperjelas pengajuan cuti. Seperti pengajuan kehadiran yang lainnya, cuti juga memerlukan persetujuan jika karyawan yang mengajukan cuti memiliki *parent* atau atasan.



Gambar 3.34 Halaman *Submit New Leave*

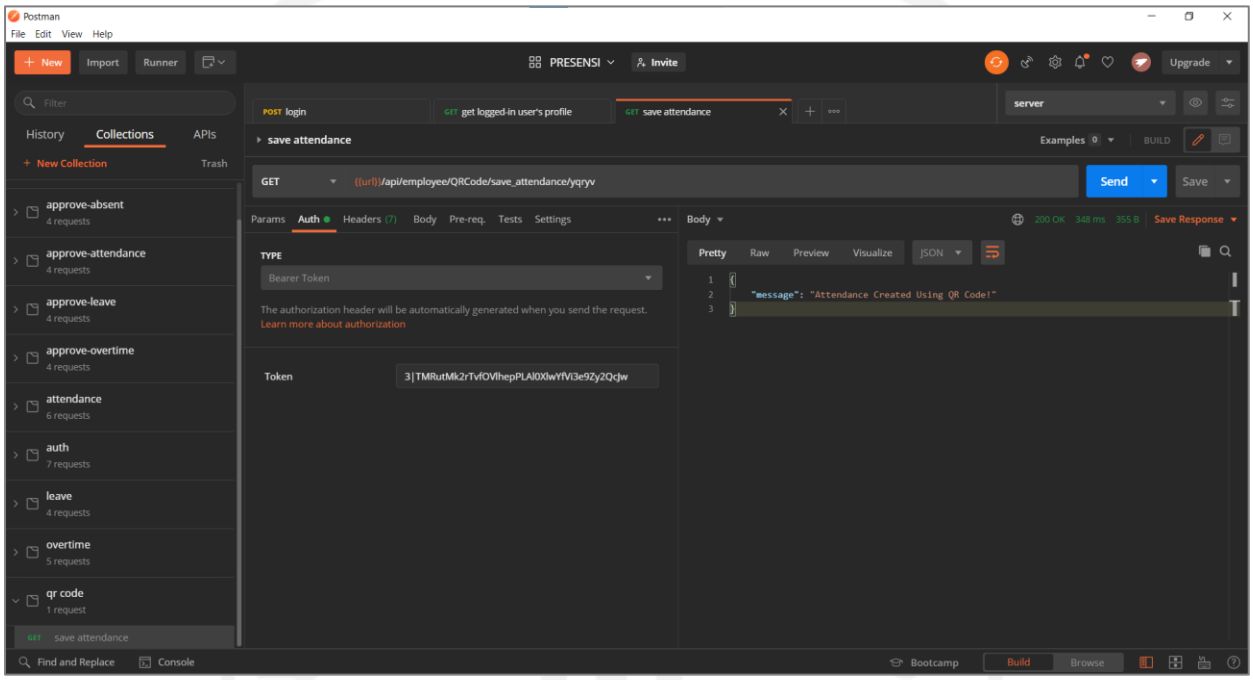
o. Pengembangan fitur presensi menggunakan *QR Code*

Selain fitur-fitur sebelumnya, fitur lain yang dapat digunakan karyawan adalah pengajuan presensi menggunakan *QR Code.* Fitur ini ditujukan kepada karyawan yang bekerja dari kantor atau *on site.* Fitur pengajuan presensi menggunakan *QR Code* dikembangkan selama 3 hari dari tanggal 26—28 April 2021.

Seperti yang telah dijelaskan sebelumnya, *administrator* dapat mengakses fitur untuk membuat *QR Code* dimana *QR Code* tersebut dapat dipindai oleh karyawan menggunakan aplikasi pada perangkat *mobile* yang akan dikembangkan oleh pemagang lain. Proses yang terjadi ketika *QR Code* dipindai oleh aplikasi *mobile* adalah sistem akan mengalihkan karyawan menuju alamat atau *url* yang akan mengecek apakah token yang ada pada *QR Code* tersebut sama dengan token yang ada pada *database.*

Ketika presensi menggunakan *QR Code* berhasil dibuat, sistem akan mengambil lokasi karyawan menggunakan *latitude* dan *longitude* dari *ip address* karyawan, serta mengirim *email*

notifikasi kepada *parent* atau atasan karyawan untuk meminta persetujuan. Selain itu, karena presensi dibuat menggunakan *QR Code* karyawan tidak bisa menentukan *task plan* dan catatan kehadiran pada saat mengajukan presensi, maka dari itu *task plan* dan catatan kehadiran pada presensi yang menggunakan *QR Code* harus diperbarui oleh karyawan yang bersangkutan. Pengajuan presensi menggunakan *QR Code* diuji menggunakan *Postman* melalui alamat atau *url* 127.0.0.1:8000/api/employee/QRCode/save_attendance/{token} (Gambar 3.35), dimana token pada *url* tersebut adalah token pada *QR Code* yang dipindai oleh karyawan.



Gambar 3.35 Pengujian presensi *QR Code* menggunakan *Postman*

p. Pengembangan fitur *approval* untuk atasan/*parent*

Salah satu fitur yang terdapat pada sistem ini adalah fitur bagi *parent* atau atasan untuk memberikan persetujuan atau *approval* kepada kehadiran karyawan. Fitur ini dikembangkan selama 4 hari terhitung dari tanggal 1—4 April 2021. Pada sistem ini *approval* dapat diberikan kepada semua jenis kehadiran seperti presensi, lembur, izin, dan cuti. Kehadiran pada sistem ini memiliki 3 *approval* status, yaitu:

a. *Needs Approval*

Kehadiran dengan status *needs approval* adalah kehadiran yang memerlukan persetujuan dari atasan. Kehadiran karyawan akan memiliki status *needs approval* ketika pertama kali diajukan atau ketika kehadiran tersebut disunting setelah ditolak oleh *parent* atau atasan karyawan yang bersangkutan.

b. *Approved*

Kehadiran dengan status *approved* adalah kehadiran yang telah mendapatkan persetujuan dari *parent* atau atasan. Kehadiran karyawan akan dianggap valid saat kehadiran tersebut diajukan tepat waktu atau saat *task* yang akan dikerjakan oleh karyawan dianggap sesuai oleh atasan dari karyawan yang bersangkutan.

c. *Rejected*

Kehadiran dengan status *rejected* adalah kehadiran yang tidak mendapatkan persetujuan dari *parent* atau atasan. Kehadiran karyawan akan dianggap tidak valid atau tidak mendapatkan persetujuan atasan saat karyawan terlambat mengajukan kehadiran atau saat *task* yang akan dikerjakan dianggap tidak sesuai oleh atasan dari karyawan tersebut. Untuk menolak kehadiran karyawan bawahannya, seorang atasan diwajibkan untuk mengisi *rejection note* atau alasan mengapa kehadiran tersebut ditolak.

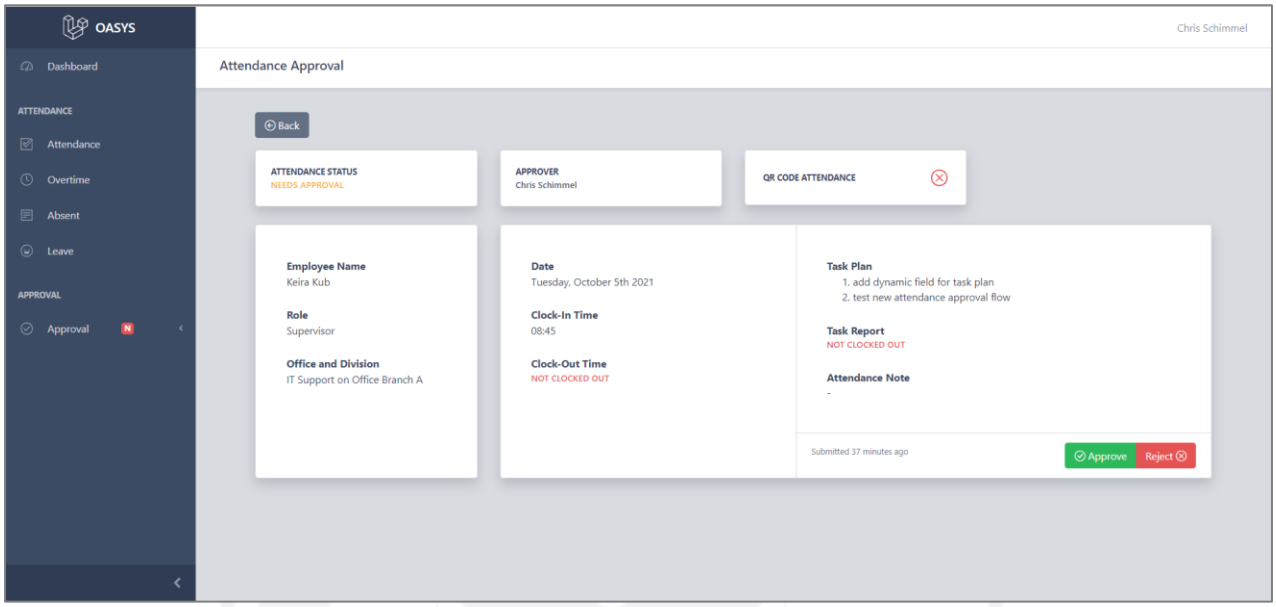
Sistem ini memberikan hak akses kepada atasan untuk mengakses halaman *approval* bagi setiap jenis kehadiran menggunakan *gate* pada *class AuthServiceProvider* dengan nama *approve-attendances*. Kemudian, *gate* tersebut akan didaftarkan pada file kernel.php agar dapat digunakan sebagai *middleware* pada *routes*. Kode yang digunakan untuk membuat *gate* bagi atasan dapat dilihat pada Gambar 3.36.

```
public function boot()
{
    Gate::define('approve-attendances', function() {
        $user = User::query()->find(auth()->id());
        if ($user->children()->exists()) {
            return true;
        }
        return false;
    });
{
```

Gambar 3.36 Kode yang digunakan untuk membuat *gate* bagi atasan

Fitur pemberian persetujuan kepada kehadiran karyawan baik presensi, lembur, izin, ataupun cuti pada dasarnya memiliki proses yang sama. Sebagai contoh, ketika ada seorang karyawan yang mengajukan presensi, maka sebuah *email* notifikasi akan dikirimkan kepada atasannya yang memberitahukan bahwa ada sebuah presensi yang perlu dicek dan disetujui. Untuk memberikan persetujuan atas sebuah presensi, atasan dapat mengakses halaman *Attendance Approval* yang dapat dilihat pada Gambar 3.37. Data-data yang ditampilkan pada halaman tersebut pada dasarnya sama dengan halaman *Attendance Details* milik karyawan, dengan penambahan data karyawan

yang mengajukan presensi tersebut. Pada halaman tersebut juga terdapat dua buah tombol untuk menyetujui atau menolak presensi.



Gambar 3.37 Tampilan halaman *Attendance Approval*

Saat atasan menekan tombol *approve*, maka sistem akan memperbarui status presensi menjadi *approved*, dan ketika atasan memilih untuk menolak presensi maka sebuah *modal* yang berisi *input field* untuk *rejection note* atau alasan mengapa presensi tersebut ditolak akan ditampilkan dan sistem akan memperbarui status presensi menjadi *rejected*. Pada saat presensi seorang karyawan memiliki status *rejected*, karyawan yang bersangkutan dapat memperbarui detail dari presensi tersebut dan status dari presensi akan berubah kembali menjadi *needs approval*.

q. Implementasi *cache* untuk optimasi sistem

Pengembangan sistem kemudian dilanjutkan dengan optimasi *request duration* ke *database* saat halaman sistem pada *web* dimuat. *Task* ini dikerjakan selama 5 hari terhitung dari tanggal 17—25 Mei 2021. Optimasi dilakukan dengan tujuan agar sistem tidak perlu memakan waktu yang banyak untuk melakukan *request* data ke *database* yang akan mempengaruhi durasi waktu saat sebuah halaman dimuat. Untuk mencapai tujuan ini data yang telah diambil dari *database* disimpan ke dalam *Redis*. *Redis* atau singkatan dari *Remote Dictionary Server* adalah sebuah media penyimpanan pada memori yang dapat diakses dengan cepat dan dapat digunakan sebagai *database*, *cache*, dan *message broker*. Secara *default*, Laravel sudah menyediakan fitur untuk *caching* ke dalam *Redis* sebagai media penyimpanan sementara.

Saat seorang karyawan mengakses halaman *index* presensi, sistem akan mengecek apakah data yang akan ditampilkan disimpan di dalam *Redis* atau tidak. Jika data tersebut belum disimpan di dalam *Redis,* maka sistem akan melakukan *request* ke *database* untuk mendapatkan data tersebut dan menyimpannya ke dalam *Redis.* Sehingga saat karyawan mengakses halaman *index* presensi lagi sistem tidak perlu mengambil data dari *database*, tetapi mengambil data tersebut dari *Redis* yang akan mempercepat durasi halaman tersebut dimuat. Kode untuk menyimpan data ke dalam *Redis* digunakan pada *method index* yang dapat dilihat pada Gambar 3.38.

```
public function index()
{
    if(Cache::has('attendance.all')) {
        $attendances = Cache::get('attendance.all');
    } else {
        $attendances = Cache::remember('attendance.all', 60, function() {
            return $this->attendanceService->getAttendance();
        });
    }
    return view('user.attendance.index', compact('attendances'));
}
```

Gambar 3.38 Kode yang digunakan untuk menyimpan data ke dalam *Redis*

Untuk melakukan *debug* dan melihat berapa lama sebuah halaman dimuat serta jumlah *query* yang dijalankan, sistem ini menggunakan sebuah *package* bernama Laravel Debugbar. Pada Gambar 3.39 menunjukkan saat *cache* belum diimplementasikan pada halaman *index* presensi. Dapat dilihat waktu yang dibutuhkan untuk memuat halaman *index* presensi adalah 403ms dengan *query* yang dijalankan sebanyak 16 buah.

Gambar 3.39 Halaman *index* presensi sebelum implementasi *cache*

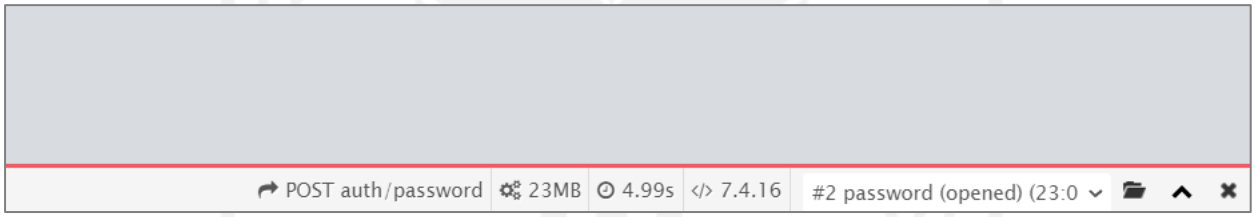
Sedangkan pada Gambar 3.40, *cache* telah diimplementasikan pada halaman *index* presensi dan mempercepat waktu halaman dimuat menjadi hanya 296ms dengan *query* yang dijalankan sebanyak 15 buah.

Gambar 3.40 Halaman *index* presensi setelah implementasi *cache*

r. Implementasi *queue and jobs* untuk mengirim *email* notifikasi

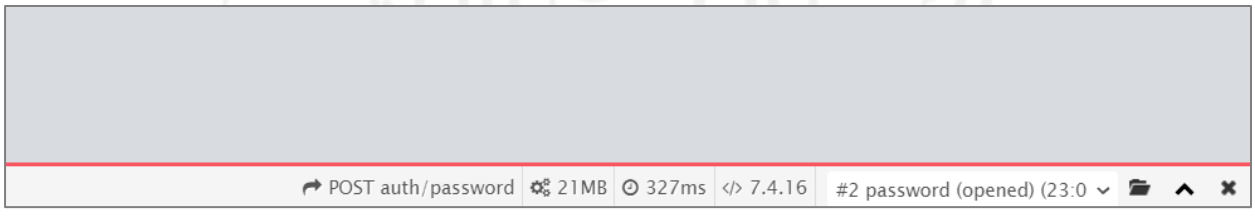
Selain menggunakan *cache* ke dalam *Redis*, sistem ini juga menggunakan *background task* untuk membuat sistem menjadi semakin optimal. Implementasi *queue and jobs* pada sistem presensi dilakukan pada tanggal 8—13 Juni 2021. *Background task* akan digunakan untuk mengirim *email* notifikasi kepada *parent* atau atasan saat ada sebuah presensi yang memerlukan persetujuan dan pengiriman *email forgot password* kepada *administrator*. Proses seperti pengiriman *email* dapat memakan waktu yang cukup lama dan akan memberikan pengalaman pengguna yang kurang baik. Tujuan dari implementasi *background task* pada proses pengiriman *email* adalah agar pengiriman *email* dapat diproses pada *background*, sehingga pengguna tidak perlu menunggu proses selesai untuk dapat menggunakan sistem kembali. Untuk mengetahui perbedaan durasi waktu yang dibutuhkan untuk proses pengiriman *email*, Laravel Debugbar akan kembali digunakan.

Dapat dilihat pada Gambar 3.41 proses pengiriman *email forgot password* kepada *administrator* memakan waktu sebanyak 4.99s atau hampir lima detik sebelum *background task* diimplementasikan.



Gambar 3.41 Durasi pengiriman *email* sebelum implementasi *background task*

Sedangkan setelah *background task* diimplementasikan, durasi yang dibutuhkan untuk mengirim *email* kepada *administrator* berkurang drastis menjadi hanya 327ms seperti yang ditunjukkan pada Gambar 3.42.

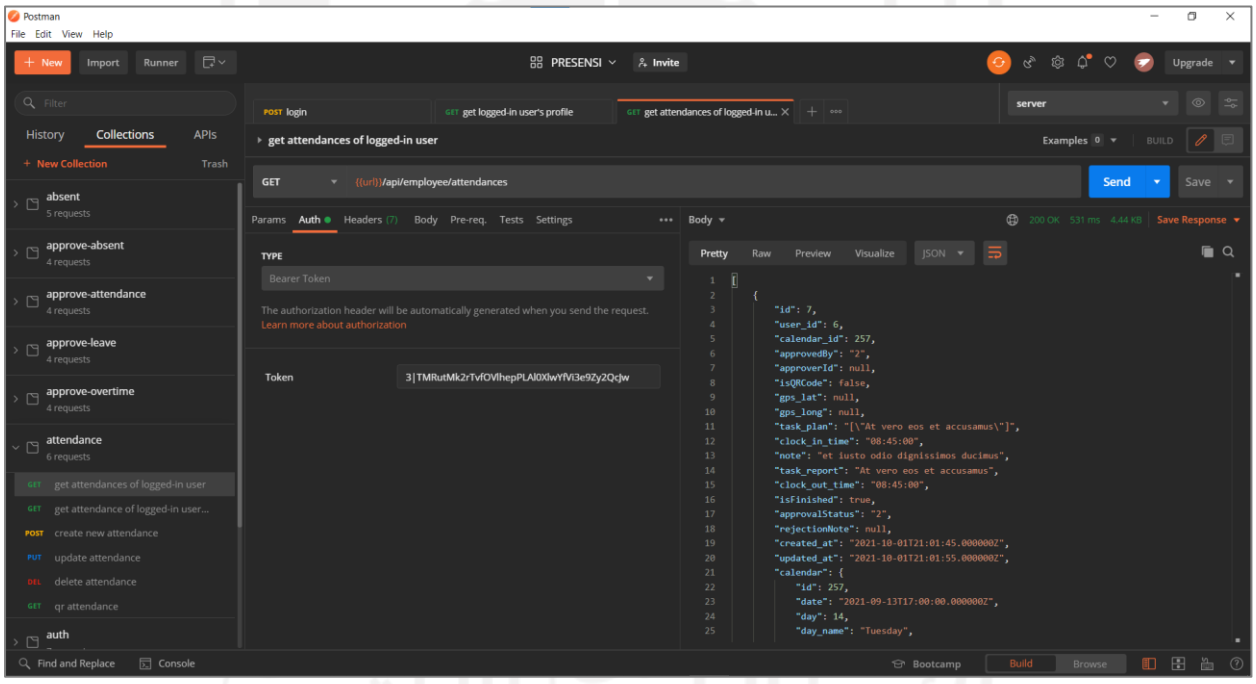


Gambar 3.42 Durasi pengiriman *email* setelah implementasi *background task*

s. Pengujian *endpoint Rest API* menggunakan *Postman*

Telah dijelaskan sebelumnya bahwa sistem ini adalah sebuah sistem yang dapat digunakan pada *web* dan perangkat *mobile*, maka dari itu sebuah *Rest API* juga dikembangkan agar karyawan yang bekerja dari kantor atau *work from office* dapat menggunakan sistem ini pada sebuah aplikasi *mobile* yang dikembangkan oleh rekan pemagang lain. Untuk memastikan *Rest API* dapat berjalan dengan baik, setiap *endpoint* harus diuji. Pengujian *endpoint Rest API* menggunakan *Postman* dilakukan sejak pengembangan *API* dimulai atau pada tanggal 30 Maret 2021.

Pengujian *endpoint Rest API* dilakukan menggunakan *Postman*. Selain digunakan untuk menguji *endpoint Rest API, Postman* juga digunakan untuk membuat dokumentasi *API* bagi rekan pemagang yang akan mengembangkan aplikasi *mobile* berbasis *Flutter*. Salah satu pengujian *endpoint Rest API* menggunakan *Postman* adalah pengujian *endpoint index* presensi yang diakses melalui *url* 127.0.0.1:8000/api/employee/attendances yang dapat dilihat pada Gambar 3.43.



Gambar 3.43 Pengujian *endpoint index* presensi menggunakan *Postman*

3.1.5 Pemantauan dan Pengendalian Proyek

Tahap pemantauan dan pengendalian proyek dilakukan agar proses pengembangan sistem dapat berjalan dengan baik dan memenuhi fungsi sebagaimana yang dibutuhkan. Proses pemantauan dan pengendalian proyek dilakukan melalui *weekly sprint planning* yang dilakukan dengan *supervisor* sebagai *project manager* menggunakan *Google Meet.*

Pada *weekly sprint planning*, pemagang melaporkan *progress* pengembangan sistem, mengkonsultasikan kendala yang dihadapi pada pengembangan fitur selama satu minggu sebelumnya, dan merencanakan *task* atau fitur apa yang akan dikembangkan untuk minggu berikutnya. Pelaporan *progress* pengembangan sistem presensi, konsultasi terkait kendala, tanggapan *supervisor* dan *task* yang dikerjakan pada *sprint* berikutnya ditunjukkan oleh Tabel 3.2 di bawah ini.

Tabel 3.2 Tabel Kegiatan *Weekly Sprint Planning*

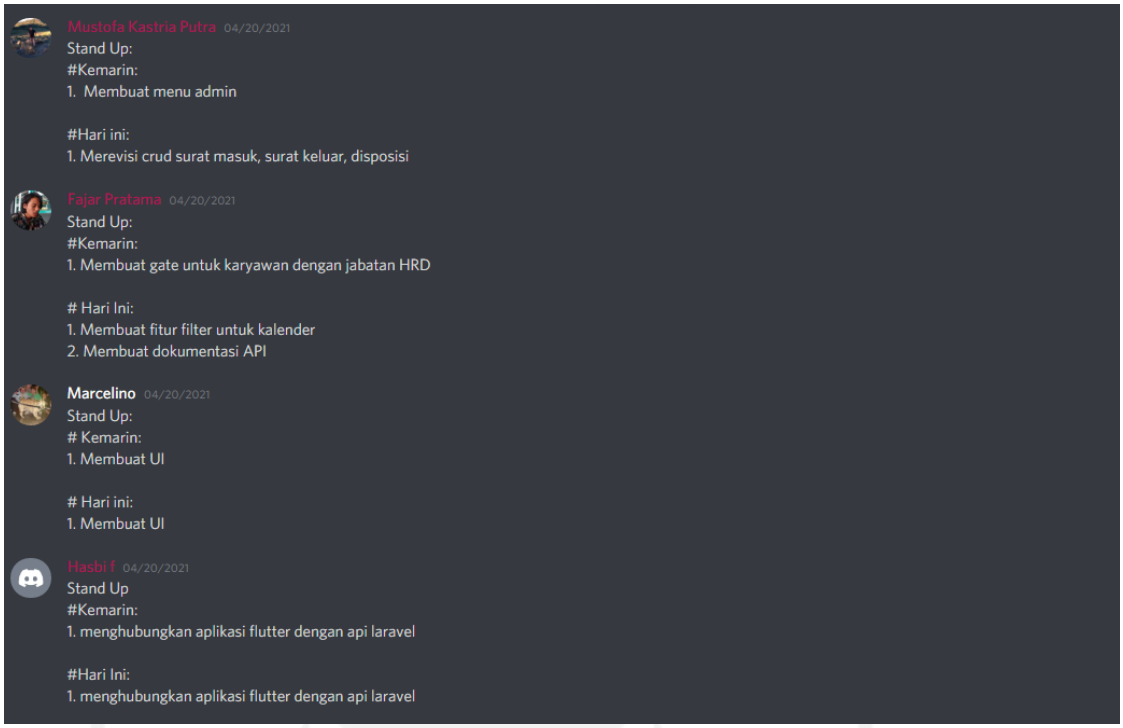
No	**Tanggal**	**Kendala**	**Tanggapan dari *supervisor***	***Task* pada *sprint* selanjutnya**
1	19-02-2021	-	-	Membuat *flowchart, ERD,* dan *mockup* tampilan aplikasi.
2	27-02-2021	*Flow* dan *ERD* aplikasi yang dirasa kurang tepat.	*Brainstorming* bersama anggota tim sistem presensi pada tanggal 1 Maret 2021.	Memperbaiki sebagian *flow* dan *ERD,* dan melanjutkan *mockup* tampilan.
3	01-03-2021	*Flow* dan *ERD* aplikasi yang dirasa kurang tepat.	Saran terkait perubahan *flow* dan *ERD* serta fitur-fitur pengguna.	Merancang ulang *flow* dan *ERD,* membuat sebagian *migration* dan *seeder,* serta mengembangkan sebagian fitur.
4	26-03-2021	Pertanyaan terkait apakah karyawan tetap bisa mengajukan presensi di luar jam dan hari kerja.	Karyawan tetap bisa mengajukan presensi di luar jam dan hari kerja untuk mengantisipasi kejadian dimana karyawan lupa mengajukan presensi.	Memperbaiki fitur pengajuan presensi (*web* dan *API*), mengembangkan fitur pengajuan lembur, dan mengembangkan fitur-fitur *administrator.*

5	12-04-2021	Pertanyaan terkait fitur manajemen kalender.	Fitur dapat diakses oleh *administrator* dan karyawan *human resource*, serta tampilkan status hari pada *form* pengajuan presensi.	Memperbaiki fitur manajemen kalender, mengubah fitur *password reset*, dan mengembangkan *API*.
6	16-04-2021	Mengalami kesulitan saat pengembangan fitur presensi *QR Code*.	Pelajari kembali konsep terkait *QR Code* dan penggunaannya.	Melanjutkan pengembangan fitur presensi *QR Code*, mengembangkan fitur *approval* dan membuat dokumentasi *API*.
7	23-04-2021	Selalu mendapatkan *ip address localhost* saat mengambil *geolocation* ketika *QR Code* dipindai.	Sistem masih berada pada *local environment*, berikan saja *field* untuk mengisi *latitude* dan *longitude*.	Melanjutkan pengambilan *geolocation*, melanjutkan pengembangan *API* dan membuat laporan kehadiran.
8	30-04-2021	Pertanyaan terkait data apa saja yang ditampilkan pada laporan kehadiran.	Saran terkait pembuatan laporan kehadiran bulanan karyawan.	Memperbaiki *query* dan membuat fitur *filter* untuk laporan kehadiran.
9	07-05-2021	Mengalami kesulitan saat implementasi *cache* ke dalam *Redis*.	Pelajari kembali konsep terkait *Redis* dan penggunaannya serta saran data apa saja yang sebaiknya disimpan ke dalam *Redis*.	Mempelajari *Redis* dan menerapkan *caching* ke dalam *Redis* untuk data yang akan diakses baik *administrator* maupun karyawan.

10	21-05-2021	Penerapan fitur *password reset* yang dirasa masih kurang tepat.	Saran terkait perbaikan fitur *password reset.*	Melanjutkan penerapan *Redis,* memperbaiki fitur *password reset,* dan membuat *dashboard*
11	28-05-2021	Pertanyaan terkait data apa saja yang ditampilkan pada *dashboard administrator* dan karyawan.	Saran untuk menampilkan jumlah kehadiran karyawan pada satu hari untuk *administrator*.	Memperbaiki *dashboard* untuk *administrator* dan karyawan.
12	04-06-2021	Pertanyaan terkait implementasi *background task.*	Saran untuk mengimplementasikan *background task* pada proses pengiriman *email* notifikasi.	Mempelajari *background task,* menerapkan *background task,* dan membuat *email* notifikasi.
13	17-06-2021	-	-	Membuat *dynamic form* untuk *task plan* pada pengajuan presensi sesuai dengan saran *supervisor.*
14	24-06-2021	-	-	Melengkapi dokumentasi *API,* mencari *bug,* memperbaiki *script timer QR Code,* dan presentasi final *project.*

Selama magang, ada beberapa *tools* yang digunakan untuk memantau dan mengendalikan proyek, di antaranya adalah:

a. *Discord,* merupakan sebuah *platform* komunikasi yang digunakan di Refactory. Aktivitas pemagang seperti *stand up* dan diskusi secara umum dilakukan pada *channel* magang yang dapat dilihat pada Gambar 3.44.



Gambar 3.44 Tangkapan layar *channel* magang Refactory

b. *Google Meet*, digunakan sebagai media untuk melaporkan *progress* dari sistem yang sedang dikembangkan dan sarana komunikasi dalam bentuk *tele-conference*.

3.1.6 Penutupan dan Pengujian Proyek

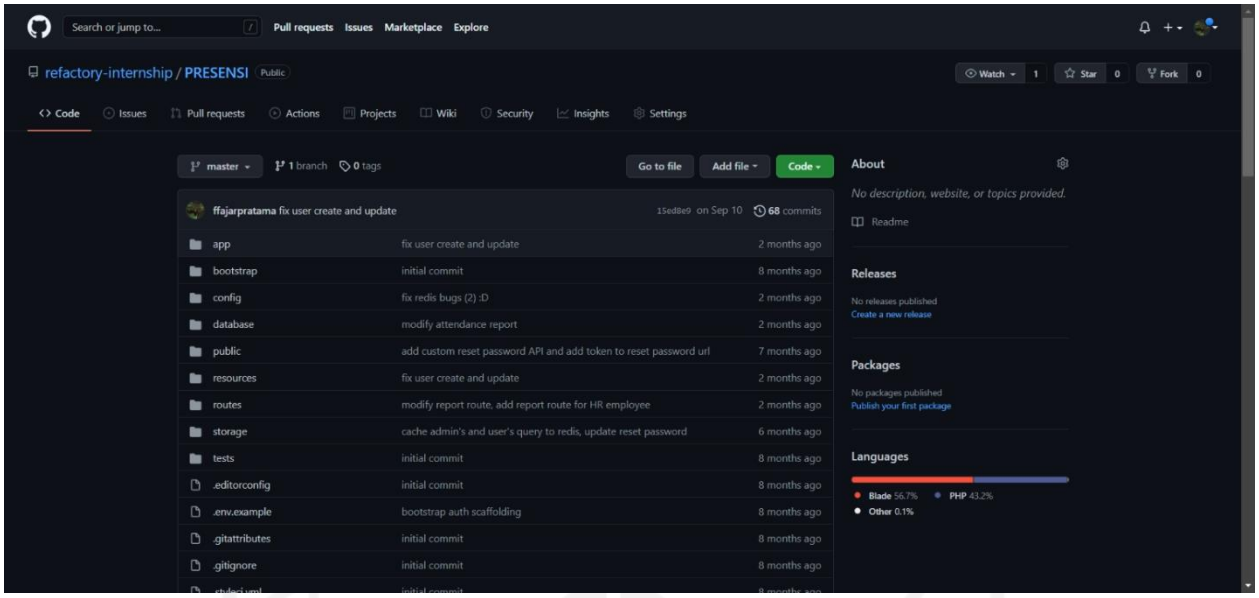
Pengembangan proyek sistem presensi dilaksanakan selama kurang lebih 4 bulan terhitung dari tanggal 22 Februari 2021 sampai dengan tanggal 1 Juli 2021, dimana semua kebutuhan sistem telah dikembangkan dan berjalan dengan baik.

Hasil dari pelaksanaan magang ini adalah sebuah sistem presensi yang dapat digunakan baik dari kantor ataupun dari rumah bagi perusahaan yang menerapkan WFO dan WFH atau sistem *hybrid*. Sistem presensi dikembangkan menggunakan beberapa teknologi seperti yang telah didefinisikan pada proses analisis *requirement* sistem. Selain itu, fitur-fitur pada sistem presensi juga telah dikembangkan sesuai dengan hasil analisis fitur berdasarkan hak akses pengguna pada proses perencanaan proyek. Fitur-fitur yang telah dikembangkan pada sistem presensi ditunjukkan oleh Tabel 3.3 di bawah ini.

Tabel 3.3 Fitur-Fitur Sistem Presensi yang Telah Dikembangkan

No	Pengguna fitur	Fitur yang dikerjakan
1	Semua pengguna	*Login*
2		Meminta pengaturan ulang *password*
3		*Edit profile*
4	*Administrator*	Pengaturan ulang *password* karyawan
5		Manajemen karyawan
6		Manajemen *roles*
7		Manajemen kantor
8		Manajemen divisi
9		Manajemen *time settings* untuk divisi
10		Pembuatan *QR Code*
11		Pembuatan laporan kehadiran karyawan
12	*Administrator* dan karyawan *Human Resource*	Manajemen kalender
13	Semua karyawan	Pengajuan presensi
14		Pengajuan presensi menggunakan *QR Code*
15		Pengajuan lembur
16		Pengajuan izin
17		Pengajuan cuti
18	Karyawan (*parent*/atasan)	*Approval* presensi
19		*Approval* lembur
20		*Approval* izin
21		*Approval* cuti

Proyek sistem presensi ditutup pada tanggal 9 Juli 2021, dilakukan bersamaan dengan selesainya pengembangan aplikasi *mobile* oleh pemagang lain. Penutupan aktivitas pengembangan sistem presensi juga dilakukan pada hari terakhir penulis melaksanakan magang di Refactory. Pengembangan sistem diakhiri dengan presentasi final dan menyertakan dokumentasi *API* pada *repository* proyek di *Github* yang dapat dilihat pada Gambar 3.45.



Gambar 3.45 *Repository* sistem presensi pada *Github*

Untuk memastikan sistem yang dikembangkan berjalan dengan baik dan telah memenuhi requirement sebagaimana yang telah didefinisikan, pengujian dilakukan kepada calon pengguna yang kemudian dilanjutkan dengan pengisian kuesioner. Pengujian sistem presensi akan dilakukan oleh pengguna dengan cara berinteraksi secara langsung dengan sistem presensi, terutama pada fitur pengajuan presensi dan *approval* presensi. Sebelum melakukan pengujian, penulis menentukan kriteria pengguna agar mendapatkan hasil pengujian yang optimal. Kriteria pengguna adalah karyawan yang memiliki atasan dan/atau bawahan. Penjelasan mengenai tingkatan penilaian yang akan diberikan oleh pengguna dapat dilihat pada Tabel 3.4.

Tabel 3.4 Tabel Tingkat Penilaian Pengujian

Keterangan	**Jawaban**
Tidak Setuju	1
Kurang Setuju	2
Cukup	3
Setuju	4
Sangat Setuju	5

Kemudian, daftar pertanyaan yang diajukan kepada pengguna dan penilaian yang diberikan oleh 10 orang pengguna ditunjukkan oleh Tabel 3.5. Selain pertanyaan, penulis juga meminta saran agar sistem presensi ini dapat menjadi lebih baik lagi kedepannya.

Tabel 3.5 Tabel Daftar Pertanyaan dan Penilaian Pada Pengujian

No	**Pertanyaan**	**Penilaian yang Diberikan**					**Total**
		1	**2**	**3**	**4**	**5**	
1	Apakah menurut Anda sistem ini dapat mempermudah proses pengajuan presensi dari rumah dan dari kantor?				3	7	47
2	Apakah menurut Anda sistem ini dapat mempermudah proses pendataan kehadiran karyawan terlepas dari kebijakan *work from home* dan *work from office?*				3	7	47
3	Apakah proses pengajuan presensi pada sistem ini cukup mudah?				5	5	45
4	Apakah menurut Anda proses *clock-out* presensi pada sistem ini cukup mudah?			2	4	4	42
5	Apakah menurut Anda fitur *approval presensi* pada sistem ini dapat memberikan transparansi kerja karyawan kepada atasan?				4	6	46
6	Apakah menurut Anda pelaporan *task* kepada atasan dapat membantu karyawan bekerja secara efektif?			2	3	5	43
7	Apakah menurut Anda proses *approval* presensi pada sistem ini cukup mudah?				5	5	45
8	Apakah menurut Anda sistem presensi yang dapat digunakan dari rumah dan dari kantor mempermudah proses presensi pada perusahaan yang mempekerjakan karyawannya secara *hybrid?*				3	7	47
Total							**362**

Kesimpulan dari pengujian dapat diambil dengan menentukan bobot penilaian dari keseluruhan nilai yang sudah diambil. Bobot penilaian dari pengujian sistem dapat dilihat pada Tabel 3.6.

Tabel 3.6 Tabel Bobot Penilaian

Kesimpulan Pengujian	**Bobot**
Sangat Tidak Memuaskan	0% - 19%
Tidak Memuaskan	20% - 39%
Cukup Memuaskan	40% - 59%
Memuaskan	60% - 79%
Sangat Memuaskan	80% - 100 %

Berdasarkan penilaian yang telah dikumpulkan, persentase keseluruhan penilaian didapatkan dari perhitungan menggunakan rumus (3.1) di bawah ini.

$$persentase\ penilaian = \frac{x}{z} \times 100\%$$

$$x = \sum_{i=1}^{y} n_i = n_1 + n_2 + \cdots + n_i$$

$$x = \sum_{i=1}^{10} n_i = 37_1 + 39_2 + 36_3 + 36_4 + 35_5 + 31_6 + 33_7 + 40_8 + 35_9 + 40_{10} \quad (3.1)$$

$$x = 362$$

$$z = (bobot\ nilai\ tertinggi \times jumlah\ pertanyaan) \times jumlah\ penguji$$

$$z = (5 \times 8) \times 10 = 400$$

$$persentase\ penilaian = \frac{362}{400} \times 100\% = 91\%$$

Dari perhitungan persentase penilaian di atas, penilaian pengguna terhadap fitur pengajuan dan *approval* presensi adalah 91%, yang memiliki bobot penilaian sangat memuaskan.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Teknis

Selama melaksanakan magang di Refactory, penulis diberikan sebuah studi kasus sebagai proyek untuk mengembangkan sebuah sistem presensi. Dalam proyek tersebut, penulis berperan sebagai *full stack developer* yang mengembangkan aplikasi berbasis web dan mengembangkan sebuah *Rest API* untuk sebuah aplikasi pada perangkat *mobile* yang dikembangkan oleh pemagang lain. Sistem presensi yang dikembangkan adalah sebuah sistem presensi yang dapat digunakan baik dari kantor ataupun dari rumah, mengingat banyak perusahaan yang mempekerjakan karyawannya dari kantor dan dari rumah selama pandemi COVID-19. Ada banyak kesulitan yang dihadapi selama proses pengembangan sistem berlangsung, namun sistem ini dapat dikembangkan sesuai dengan ekspektasi dan *requirement* yang telah ditentukan oleh *supervisor* sebagai *project manager* sistem presensi.

Selama magang ada banyak pengetahuan baru yang didapat seperti apa itu *Rest API,* bagaimana cara agar *API* dapat memberikan *response* yang mudah dimengerti, bagaimana cara membuat sebuah *Rest API* yang baik, dan bagaimana cara membuat sistem yang dikembangkan menjadi lebih optimal menggunakan *cache* dan *backgrond task*. Ada banyak kesulitan yang dihadapi saat magang karena selama ini penulis belum pernah membuat *Rest API* atau belum benar-benar paham terkait konsep *caching* dan *background task.* Kesulitan ini berakibat pada *task* yang dikerjakan tidak dapat diselesaikan dalam satu *sprint*. Meskipun demikian, dari kesulitan tersebut penulis mendapatkan motivasi untuk belajar lebih tekun agar dapat menyelesaikan *task* yang diberikan sesuai dengan tenggat waktu yang ditentukan.

Selain itu, penulis juga banyak mempelajari teknologi dan *tools* baru selama melaksanakan magang di Refactory seperti PhpStorm sebagai *IDE* untuk mengembangkan aplikasi berbasis PHP yang menggunakan *framework* Laravel, Visual Studio Code untuk mengembangkan aplikasi yang menggunakan Express.js sebagai *framework,* Postman sebagai *rest client* untuk pengujian *endpoint Rest API* dan membuat dokumentasi *API,* DataGrip sebagai *tools* untuk manajemen *database,* SourceTree sebagai *tools* yang mempermudah *version control* pada proyek, dan Redis sebagai media penyimpanan sementara.

4.2 Non Teknis

4.2.1 Manfaat Magang

Melaksanakan magang di Refactory selama 6 bulan sebagai *back-end developer* membuat penulis mendapatkan banyak sekali pengalaman dan pengetahuan baru. Penulis mendapatkan pengetahuan tentang teknologi apa saja yang saat ini banyak digunakan untuk mengembangkan sebuah aplikasi berbasis web dan sedang dibutuhkan oleh banyak perusahaan. Penulis mendapatkan pengalaman untuk menjadi bagian dari Refactory dan mengetahui bagaimana lingkungan dan aturan kerja yang ada. Pengalaman tersebut memberikan penulis gambaran bagaimana industri teknologi dan informasi yang ada sekarang. Selain itu, penulis juga mendapatkan pengetahuan bagaimana mengerjakan sebuah proyek menggunakan metode *agile* dengan kerangka kerja *scrum* yang belum pernah penulis lakukan selama kuliah.

Selama melaksanakan magang di Refactory penulis juga mendapatkan kesempatan untuk mengikuti kegiatan rutin yang ada seperti *daily stand-up*, *core time*, dan *weekly sprint planning*. Selain itu, *supervisor* juga memberikan banyak *task* selama kegiatan magang berlangsung dimana setiap *task* memiliki tenggat waktunya masing-masing. Melaksanakan magang di Refactory mengajarkan penulis agar dapat memiliki disiplin dan mengatur waktu dengan baik karena jika terlambat mengikuti salah satu kegiatan tersebut, ada sanksi yang akan diberikan.

Sebelum melaksanakan magang di Refactory, penulis sempat mengalami kegagalan saat melaksanakan magang di tempat lain. Beberapa faktor yang menyebabkan kegagalan tersebut adalah penulis belum memiliki kemampuan yang cukup dan membutuhkan waktu yang terlalu lama untuk menyelesaikan *task* yang diberikan. Akan tetapi, kegagalan tersebut tidak membuat penulis patah semangat dan tetap memilih jalur magang untuk menyelesaikan Tugas Akhir. Kegagalan tersebut membuat penulis semakin terpacu untuk belajar lebih keras dan giat. Penulis percaya, dengan jalur magang penulis dapat memperoleh pengalaman dan wawasan baru terkait industri teknologi dan informasi yang ada pada saat ini. Selama melaksanakan magang di Refactory, penulis juga mengalami beberapa kesulitan teknis dan non teknis. Akan tetapi, penulis berusaha untuk belajar dari kegagalan yang pernah terjadi sehingga penulis dapat menyelesaikan magang di Refactory dengan hasil yang semaksimal mungkin.

Dengan melaksanakan magang di Refactory penulis juga dapat menemukan *passion* pada bidang *back-end web development.* Sebelum melaksanakan magang, penulis belum tahu pasti akan berkarir sebagai apa pada bidang teknologi dan informasi. Akan tetapi, magang di Refactory memberikan penulis pengalaman dan pengetahuan baru sebagai *back-end developer.* Saat awal magang pun penulis juga sempat mengalami kebingungan akan magang sebagai *front-end*

developer atau *back-end developer*, sehingga harus mengambil *test* untuk menentukan posisi magang yang bisa membuat penulis berkontribusi dan meningkatkan kemampuan yang dimiliki. Selama melaksanakan magang di Refactory, penulis merasa menjadi *back-end developer* adalah posisi yang paling cocok. Memiliki kesempatan untuk magang di Refactory membuat penulis memiliki kepercayaan diri dan keinginan untuk menjadi seorang *back-end developer* yang handal. Untuk mencapai tujuan tersebut, penulis harus mencari pengalaman dan pengetahuan sebanyak-banyaknya serta belajar dari kesalahan yang pernah dilakukan untuk membantu karier penulis di masa yang akan datang.

4.2.2 Hambatan dan Tantangan Selama Magang

Selama magang ada beberapa hambatan dan tantangan yang ditemui, seperti kesulitan untuk beradaptasi dengan lingkungan kerja saat awal magang, tugas-tugas yang cukup sulit untuk dikerjakan, dan kesulitan untuk menyelesaikan tugas dalam tenggat waktu yang telah ditentukan. Selain itu kondisi pandemi yang mengharuskan kegiatan magang dilakukan dari rumah disertai dengan koneksi internet yang kurang stabil juga membuat komunikasi kurang maksimal.

Sistem operasi juga sempat menjadi hambatan dalam pengembangan sistem presensi karena konfigurasi *cron* yang tidak tersedia untuk sistem operasi *Windows* sehingga fitur *daily mail* tidak dapat diimplementasikan. Hal ini juga disebabkan karena instalasi *Linux* yang selalu menemui *error.* Selain sistem operasi, beberapa hambatan dan tantangan lain yang dirasakan pada pengembangan Sistem Presensi adalah ketika mengembangkan fitur presensi menggunakan *QR Code*, membuat *dependent dropdown* untuk mendaftarkan karyawan baru, dan *dynamic form* untuk *task plan.*

Selain hambatan dan tantangan pada pengembangan sistem presensi, penulis juga mendapatkan hambatan dan tantangan lain seperti pada *task* membuat *Rest API* menggunakan Express.js. Pada awalnya, penulis tidak tahu bagaimana cara mengembangkan *Rest API* menggunakan *framework* dari *node.js* tersebut. Hal ini disebabkan karena selama masa kuliah, penulis hanya belajar mengembangkan aplikasi berbasis *web* menggunakan *framework* PHP, yaitu Laravel. Selain itu, penulis juga belum pernah mengembangkan sebuah aplikasi dengan berbagai fitur yang cukup kompleks seperti pada saat melaksanakan magang. Akan tetapi, penulis bersyukur bisa mendapatkan hambatan dan tantangan tersebut, karena penulis bisa mendapatkan banyak pengetahuan dan kemampuan baru dalam menyelesaikan masalah yang ada pada pengembangan aplikasi berbasis web.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah kegiatan pengembangan sistem presensi dan pengujian kepada pengguna menggunakan kuesioner dilakukan, dapat disimpulkan bahwa implementasi fitur-fitur yang telah dijelaskan di atas dapat mempermudah proses pengajuan kehadiran karyawan karena karyawan dapat dengan mudah mengajukan kehadiran seperti presensi, lembur, izin, dan cuti menggunakan sistem yang dapat digunakan baik dari *web browser* ataupun aplikasi *mobile.*

Selain itu, dengan dikembangkannya sistem ini akan mempermudah proses manajemen kehadiran karyawan karena data kehadiran karyawan terintegrasi terlepas dari kebijakan *work from office* (WFO) dan *work from home* (WFH). Sistem ini juga memberikan transparansi kerja karyawan dengan proses *approval* yang diberikan oleh atasan sehingga karyawan dapat bekerja dengan akurat dan tetap produktif meskipun harus bekerja dari rumah.

Pengembangan sistem ini dibantu oleh beragam *package* dan fitur-fitur pada Laravel sehingga sistem dapat dikembangkan sesuai dengan *requirement* dan ekspektasi dari *project manager*. Sebuah *Rest API* juga dikembangkan agar sistem ini dapat digunakan pada aplikasi *mobile* oleh karyawan yang bekerja dari kantor dengan cara memindai *QR Code* yang dibuat *administrator*.

Ada beberapa kendala yang dihadapi selama proses pengembangan sistem berlangsung seperti penulis yang masih kurang terbiasa mengembangkan *Rest API* beserta sistem autentikasinya dan belum benar-benar paham terkait konsep dari *caching* ke dalam *Redis* serta *background task.* Akan tetapi, kendala tersebut dapat diatasi dan pengembangan sistem dapat diselesaikan sesuai dengan tenggat waktu yang telah ditentukan.

Selain itu, dari kegiatan magang yang telah dilaksanakan selama 6 bulan di Refactory dapat disimpulkan bahwa ada banyak pengalaman dan ilmu yang penulis dapat yang sebelumnya tidak diajarkan di perkuliahan. Selain pengalaman dan ilmu baru terkait teknis pengembangan *web,* manfaat lain yang didapatkan penulis adalah pengalaman bekerja di dalam sebuah perusahaan dan manajemen waktu serta *task* yang sangat berguna untuk memberikan gambaran awal bagaimana bekerja dalam sebuah perusahaan. Walaupun ada banyak kendala dan hambatan pada saat pelaksanaan magang, magang dapat memberikan banyak manfaat. Dengan melaksanakan magang, penulis dapat memperoleh banyak wawasan dan pengetahuan baru serta kemampuan untuk beradaptasi dengan lingkungan perusahaan.

5.2 Saran

Berdasarkan hasil pengujian, diperoleh beberapa saran agar sistem dapat menjadi lebih baik kedepannya. Beberapa saran tersebut adalah:

1. Pelaporan *task* tidak perlu dilakukan saat presensi karena akan terasa rumit. Karyawan harus memikirkan *task* apa yang akan dikerjakan ketika hendak mengajukan presensi.
2. *Parent*/atasan tidak mengetahui apakah karyawan mengajukan presensi dari rumah atau dari kantor karena tidak ada keterangan yang menjelaskan hal tersebut pada sistem.
3. *Parent*/atasan tidak perlu melakukan *approval* satu per satu karyawannya karena akan merepotkan jika karyawannya banyak. Sebaiknya *approval* dilakukan secara otomatis dan atasan mendapatkan *report* seperti *dashboard* sehingga dapat melakukan analisa.
4. Sebaiknya pada fitur pengajuan cuti karyawan bisa melihat jatah cuti untuk memastikan apakah karyawan masih bisa mengajukan cuti pada periode waktu tertentu.
5. Sebaiknya fitur *clock-out* memiliki *button* sendiri, tidak di dalam *form* edit.
6. *UX Design* aplikasi perlu diperbaiki, karena beberapa komponen masih sulit dipahami dan perlu waktu bagi pengguna untuk mempelajari.
7. Sistem yang diakses menggunakan *web browser* sebaiknya dibuat *responsive*.

Selain saran yang didapat dari pengujian, penulis juga mendapat saran dari *supervisor* saat melaksanakan magang di Refactory. Saran yang diberikan adalah agar fitur *daily mail* menggunakan *task scheduling* yang masih belum dapat diimplementasikan karena terkendala konfigurasi *cron* yang tidak tersedia bagi sistem operasi *Windows* agar dapat dikembangkan menggunakan sistem operasi lain seperti *Linux*. Saran selanjutnya adalah pengambilan lokasi karyawan menggunakan *latitude* dan *longitude* saat karyawan mengajukan presensi menggunakan *QR Code* agar lebih akurat, karena saat pengembangan *ip address* karyawan adalah 127.0.0.1 atau *localhost*.

Selain saran terhadap pengembangan sistem, saran yang dapat diberikan adalah saran kepada Program Studi Informatika Universitas Islam Indonesia dan mahasiswa yang akan mengambil jalur magang di masa yang akan datang. Berdasarkan pengalaman dan observasi yang dilakukan saat magang, dapat diketahui bahwa sebagian besar industri teknologi dan informasi pada saat ini terutama pada *web development* sudah menggunakan teknologi *microservice* dengan *stack* MERN (MongoDB, Express, React, Node.js). Selain itu, bahasa pemrograman yang sedang banyak dipakai saat ini adalah Javascript dan Go, sehingga membuat *framework* berbasis PHP seperti Laravel atau Code Igniter menjadi kurang relevan.

Berdasarkan kedua hal tersebut, maka saran yang dapat diberikan kepada program studi adalah untuk mulai memperkenalkan teknologi tersebut terutama pada mata kuliah Pengembangan Aplikasi Berbasis Web. Penulis dapat memberi saran seperti itu karena selama kegiatan magang berlangsung, penulis merasa kesulitan memahami konsep dari *Rest API.*

Sedangkan saran yang dapat diberikan kepada mahasiswa yang akan mengambil jalur magang di masa yang akan datang adalah cari informasi dan detail perusahaan tempat magang, baik dari mitra kampus ataupun bukan. Tingkatkan kemampuan *problem solving* sebagai bekal untuk *coding test* yang akan dilakukan pada masa *interview*, pelajari dan gunakan *Git* semaksimal mungkin karena *Git* adalah pengetahuan dasar yang wajib dimiliki ketika memasuki industri teknologi dan informasi. Selain itu, perbanyak pengetahuan dan pengalaman berdasarkan posisi magang yang diinginkan.

DAFTAR PUSTAKA

Akbar, M. (2018). *PENGEMBANGAN RESTFUL API UNTUK APPLICATION SPECIFIC HIGH LEVEL LOCATION SERVICE* [Universitas Islam Indonesia]. https://dspace.uii.ac.id/handle/123456789/9836

Christensson, P. (2015, March 5). *QR Code Definition*. https://techterms.com/definition/qr_code

Dhingra, S. (2016). *REST vs. SOAP: Choosing the best web service*. https://searchapparchitecture.techtarget.com/tip/REST-vs-SOAP-Choosing-the-best-web-service

Ekandem, K. (2021, May 31). *Using Redis as a Cache in Laravel*. https://www.honeybadger.io/blog/laravel-caching-redis/

Ferdika, R., & Nasution, R. D. (2020). Perubahan Orientasi Motivasi Pegawai pada Penerapan E-Absensi di Kabupaten Ponorogo. *JURNAL PENELITIAN KOMUNIKASI DAN OPINI PUBLIK*, *24*(1), 71–84. https://doi.org/10.33299/jpkop.24.1.2439

Fitria, N. J. L. (2020). Penerapan Work From Home Dan Work From Office Dengan Absensi Online Sebagai Implikasi E-Government Di Masa New Normal. *Civil Service*, *14*(1), 69–84.

Fowler, M., Rice, D., Foemmel, M., Hieeatt, E., Mee, R., & Stafford, R. (2002). *Patterns of Enterprise Application Architecture*.

Heuvel, B. vd. (n.d.). *GitHub - barryvdh/laravel-debugbar: Laravel Debugbar*. Retrieved August 24, 2021, from https://github.com/barryvdh/laravel-debugbar

Husain, A., Prastian, A. H. A., & Ramadhan, A. (2017). Perancangan Sistem Absensi Online Menggunakan Android Guna Mempercepat Proses Kehadiran Karyawan Pada PT. Sintech Berkah Abadi. *Technomedia Journal*, *2*(1), 105–116. https://doi.org/10.33050/tmj.v2i1.319

Khoiriyah, N. L., Marisa, F., & Wijaya, I. D. (2018). Rancang Bangun Sistem Presensi Online Berbasis Granted Validitas Data. *J I M P - Jurnal Informatika Merdeka Pasuruan*, *3*(1), 53–61. https://doi.org/10.37438/jimp.v3i1.89

Kosasi, S., & Liauw, D. (2013). Penerapan Design Pattern Dalam Perancangan Web Order. *Jurnal Teknologi*, *6*(1), 1–9.

Laravel. (n.d.-a). *Cache - Laravel - The PHP Framework For Web Artisans*. Retrieved August 24, 2021, from https://laravel.com/docs/8.x/cache

Laravel. (n.d.-b). *Laravel Sanctum - Laravel - The PHP Framework For Web Artisans*. Retrieved August 25, 2021, from https://laravel.com/docs/8.x/sanctum

Masalha, F., & Hirzallah, N. (2014). A Students Attendance System Using QR Code. *International*

Journal of Advanced Computer Science and Applications, *5*(3), 75–79.

McCool, S. (2012). *Laravel Starter*.

Mozilla. (n.d.). *Request header - MDN Web Docs Glossary: Definitions of Web-related terms*. Retrieved August 25, 2021, from https://developer.mozilla.org/en-US/docs/Glossary/Request_header

Pranajaya, M. H. (2018). *PERANCANGAN APLIKASI WEB DAN REST API UNTUK SISTEM KEHADIRAN MAHASISWA BERBASIS QR CODE DAN LOKASI* [Universitas Sumatera Utara]. https://repositori.usu.ac.id/handle/123456789/11503

Purnomo, H. (2016). Perancangan Aplikasi Pencarian Layanan Kesehatan Berbasis Html 5 Geolocation. *Jurnal Sistem Komputer*, *6*(1), 44–51.

Redis. (n.d.). *Introduction to Redis*. Retrieved August 24, 2021, from https://redis.io/topics/introduction

Schwaber Ken, & Sutherland Jeff. (2020). Panduan Definitif untuk Scrum: Aturan Permainan. *Scrum.Org*, *November*, 1–17.

Stamat, D. (2020, October 1). *Every Web Application Needs a Background Processing Queue*. https://blog.iron.io/every-web-application-needs-background/

LAMPIRAN