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

Fajar Pratama Purwantoro Putra
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta, Indonesia
17523094@students.uii.ac.id

ABSTRAKSI

Kebijakan bekerja dari rumah atau *work from home* adalah salah satu kebijakan yang diambil oleh perusahaan-perusahaan sebagai cara untuk beradaptasi dengan pandemi COVID-19 yang belum menunjukkan penurunan. Kebijakan ini diterapkan dengan tujuan agar karyawan tidak perlu datang ke kantor untuk bekerja dengan mengerjakan pekerjaannya secara *remote* dari rumah. Akan tetapi, seiring berjalannya waktu banyak perusahaan yang menerapkan sistem *hybrid*, yaitu mempekerjakan sebagian karyawannya dari rumah dan dari kantor sebagai penerapan dari *new normal*. Salah satu kendala yang ditemui oleh perusahaan yang menerapkan *work from office* sekaligus *work from home* adalah data kehadiran karyawan yang tidak terintegrasi dalam sebuah sistem. 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 aplikasi berbasis *web* yang dikembangkan menggunakan *Laravel.* Selain itu, sebuah *Rest API* juga dikembangkan agar sistem dapat digunakan pada aplikasi *mobile.* Makalah ini membahas fitur-fitur apa saja yang dapat diimplementasikan ke dalam sebuah sistem presensi sebagai solusi dari kendala tersebut. Berdasarkan hasil penelitian, diketahui bahwa fitur-fitur yang terdapat pada sistem presensi dapat mempermudah proses pengajuan, pendataan, dan manajemen kehadiran karyawan karena sistem dapat dengan mudah digunakan pada *web browser* dan aplikasi *mobile*.

Kata Kunci

Sistem Presensi; *Rest API;* Presensi *Online; Work from Office; Work from Home;*

1. PENDAHULUAN

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 [1]. 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* akan mempersulit pendataan kehadiran karyawan yang bekerja dari rumah atau *work from home*.

Selama penerapan *work from home* atau bekerja dari rumah, pendataan kehadiran karyawan menjadi salah satu kendala yang ditemui oleh perusahaan. Kendala lain yang ditemui perusahaan dengan penerapan *work from office* dan *work from home* 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, 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 melakukan 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.*

Metode pengembangan yang digunakan untuk mengembangkan sistem ini adalah *agile* dengan *scrum* sebagai kerangka kerjanya. *Scrum* dipilih 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 [2].

Makalah ini menyajikan pengembangan sebuah sistem presensi yang dapat digunakan baik dari rumah maupun dari kantor. Penelitian perlu dilakukan untuk mengetahui fitur-fitur apa saja yang dapat mempermudah proses pengajuan, pendataan, dan manajemen kehadiran karyawan yang bekerja dari kantor *(work from office)* atau yang bekerja dari rumah *(work from home)* selama pandemi COVID-19.

2. KAJIAN PUSTAKA

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 [3]. Perkembangan sistem presensi saat ini telah berkembang seiring dengan perkembangan teknologi seperti internet, komputer, maupun *smartphone.* Dimulai dari presensi yang 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 [4].

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 [5].

Selama pandemi virus COVID-19 kegiatan karyawan yang bekerja dari rumah atau *work from home* harus dipantau dan diawasi agar tidak ada karyawan yang melakukan kecurangan seperti berkeliaran di saat jam kerja, tidak lalai dengan pekerjaan yang diberikan, dan memberikan transparansi kerja karyawan. Sistem presensi sangat diperlukan dalam pengawasan kegiatan kerja karyawan, baik dalam mendata kehadiran karyawan yang bekerja dari rumah maupun karyawan yang bekerja dari kantor.

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 [6].

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 [7].

Application Programming Interface atau *API* sendiri adalah sekumpulan definisi dan protokol yang digunakan untuk mengembangkan dan mengintegrasikan perangkat lunak [8]. *API* juga dapat diartikan sebagai penghubung antara pengguna atau *client* dengan sumber data atau layanan *web* yang diinginkan [8].

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* [9].

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 [10].

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 [10].

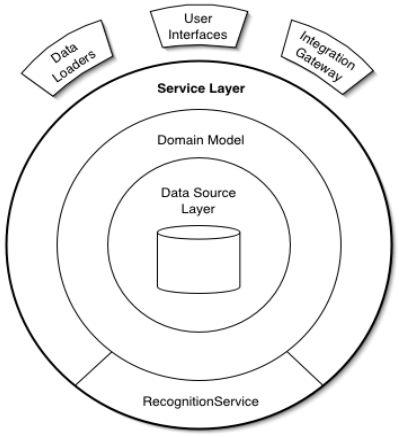
QR Code seperti pada Gambar 1 dibaca menggunakan perangkat pencitraan seperti kamera dan diformat menggunakan algoritma oleh perangkat yang menggunakan *Reed-Solomon error correction* hingga *QR Code* dapat diinterpretasikan dengan tepat. Data kemudian didapatkan dari pola yang ada baik dari komponen vertikal dan horizontal yang ada pada *QR Code* [10].

Gambar 1. *QR Code (Quick Response Code)*

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 gateway*s. 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 [11]. *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 [12]. Struktur dari *service layer* ditunjukkan pada Gambar 2.



Gambar 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 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.

3. METODOLOGI

Pengembangan sistem presensi dikerjakan bersama dalam sebuah tim yang terdiri atas *project manager, web developer,* dan *mobile developer*. Pengembangan sistem dilakukan dalam beberapa tahapan, yaitu:

a. Inisialisasi, yaitu tahap pertama dengan mengumpulkan informasi terkait teknologi apa saja yang akan digunakan pada proses pengembangan sistem presensi.
b. Analisis, yaitu tahap kedua dengan melakukan diskusi dan wawancara bersama *project manager* untuk mendefinisikan *requirement* dari sistem presensi.
c. Pengembangan, yaitu tahap ketiga dengan memulai proses pengembangan sistem dan mengimplementasikan fitur-fitur yang telah dirancang

Sistem presensi dikembangkan menggunakan metode *agile* dengan kerangka kerja *scrum* dan *weekly sprint planning* sebagai media untuk pelaporan *progress* dan konsultasi terkait kendala yang dihadapi selama proses pengembangan sistem presensi

4. HASIL DAN PEMBAHASAN

4.1 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 1. Setelah spesifikasi teknologi telah ditentukan, tahap selanjutnya adalah membuat sebuah *git repository* baru dan membuat proyek menggunakan *framework* yang telah ditentukan.

Tabel 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* yang bertanggung jawab atas pengelolaan proyek secara keseluruhan. Selain itu, project manager juga berperan sebagai otoritas tertinggi untuk konsultasi apabila dalam proses pengembangan ditemui kendala.
b. *Programmer* yang bertanggung jawab untuk mengimplementasikan dan mengeksekusi rancangan kebutuhan sistem dalam bentuk kode program, membuat algoritma. Selain itu, *programmer* juga mengimplementasikan teknologi yang sesuai dengan kebutuhan sistem untuk mencapai tujuan sistem yang sebelumnya telah dibuat.

4.2 Analisis Requirement Sistem

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. Selain itu, pada sistem ini juga dikembangkan sebuah *Rest API* yang dapat digunakan pada aplikasi *mobile* yang akan dikembangkan oleh rekan tim proyek sistem presensi.

Berdasarkan hak akses, pengguna sistem ini akan terbagi menjadi dua, yaitu *administrator* dan karyawan. 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. Pembuatan *QR Code*
h. Pembuatan laporan kehadiran karyawan
i. Pengaturan ulang atau *reset password* karyawan

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

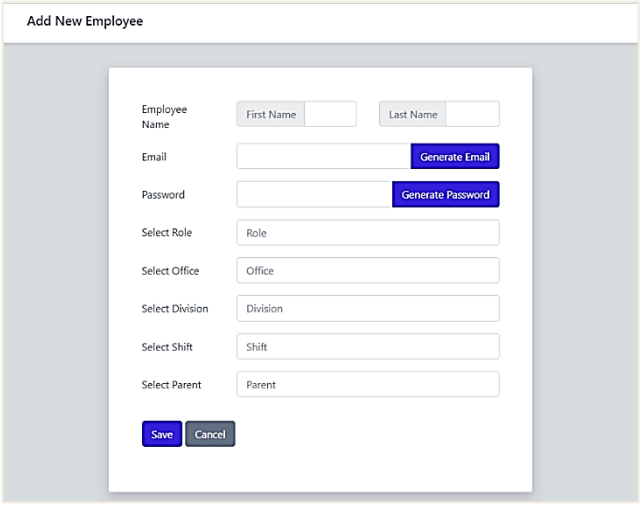
a. Pengajuan presensi
b. Pengajuan lembur
c. Pengajuan izin
d. Pengajuan cuti

4.3 Pengembangan Sistem Presensi

4.3.1 Fitur Administrator

4.3.1.1 Manajemen Karyawan

Salah satu fitur *administrator* yang terdapat pada sistem ini adalah manajemen karyawan. Pada fitur ini *administrator* dapat melakukan operasi CRUD terhadap data karyawan. Untuk menambahkan atau mendaftarkan karyawan baru, *administrator* dapat mengakses halaman *Add New Employee* seperti yang ditunjukkan pada Gambar 3.

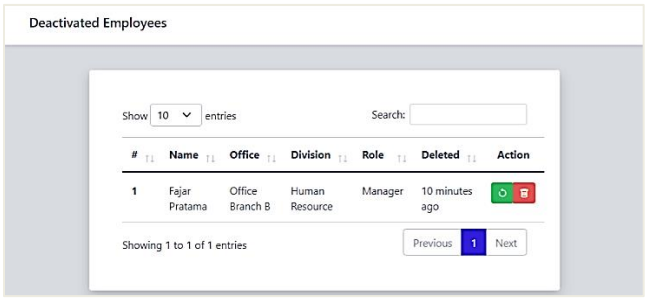


Gambar 3. Halaman *Add New Employee*

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 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 jabatan, 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.

Setelah menambah data karyawan baru, *administrator* dapat mengakses halaman *Employee Details* yang akan menampilkan data-data karyawan. *Administrator* juga dapat menyunting dan menonaktifkan karyawan melalui halaman ini.

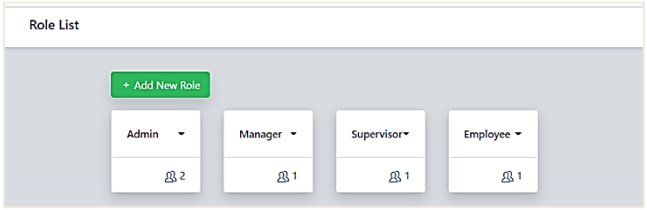
Sebelum menonaktifkan karyawan, sebuah pesan peringatan akan ditampilkan dalam bentuk *modal.* Fitur ini menerapkan *soft delete,* yang berarti data karyawan tidak benar-benar dihapus dari tabel *users,* tetapi hanya mengisi kolom *deletedAt* yang terdapat pada tabel sehingga data tidak ditampilkan kecuali menggunakan *method wtihTrashed* saat memuat data menggunakan *Eloquent. Administrator* dapat menghapus data karyawan secara permanen atau mengembalikan data tersebut melalui halaman *Deactivated Employees* seperti yang ditunjukkan pada Gambar 4.



Gambar 4. Halaman *Deactivated Employees*

4.3.1.2 Manajemen Roles

Pada sistem ini, *role* atau jabatan berfungsi untuk mengatur hak akses pengguna dan dapat diatur secara dinamis oleh *administrator*. Gambar 5 menunjukkan data semua *roles* yang ada pada sistem ini beserta jumlah pengguna yang memiliki *role* tersebut. Pada fitur ini, *administrator* dapat menambahkan *role* baru, menyunting *role* yang ada, dan menghapus *role.*

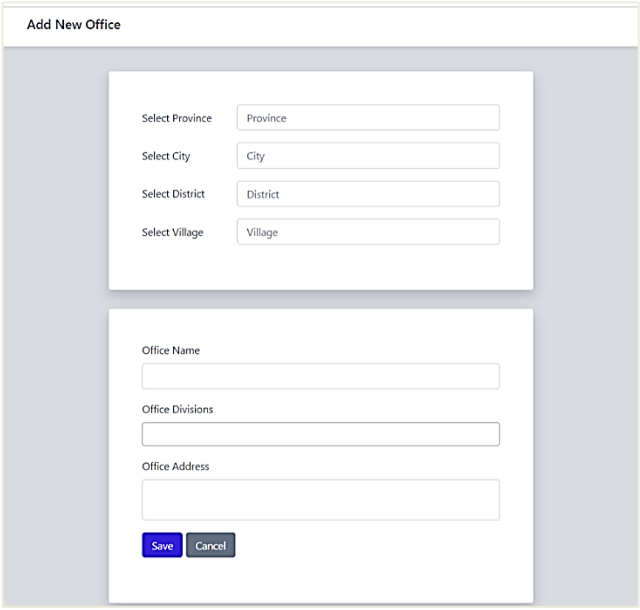


Gambar 5. Halaman *Roles List*

4.3.1.3 Manajemen Kantor

Selain fitur manajemen karyawan dan *roles*, salah satu fitur CRUD bagi *administrator* yang lainnya adalah fitur CRUD untuk data kantor. Data kantor pada sistem ini digunakan untuk mengetahui lokasi kantor, divisi pekerjaan dan jumlah karyawan yang ada pada kantor tersebut.

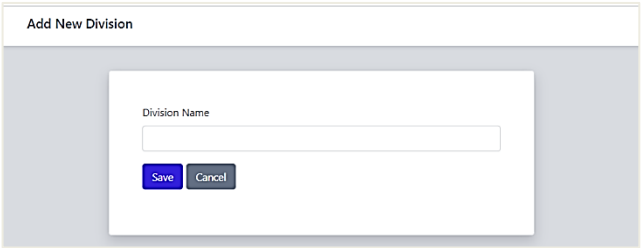
Gambar 6 menunjukkan halaman untuk menambah data kantor yang dapat diakses oleh *administrator*. Untuk menambah data kantor, *administrator* perlu mengisi beberapa *field* yang ada pada *form*, seperti lokasi kantor yang dapat dipilih menggunakan *dependent dropdown*, nama kantor, divisi-divisi yang 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 6. Halaman *Add New Office*

4.3.1.4 Manajemen Divisi

Untuk menambahkan data divisi, *administrator* dapat mengakses halaman *Add New Division* seperti yang ditunjukkan pada Gambar 7 dan mengisi *field* nama untuk divisi. Setelah menambahkan data divisi, data tersebut disimpan ke dalam tabel *divisions* yang kemudian dapat ditampilkan pada *form* tambah kantor dalam bentuk *dropdown.*



Gambar 7. Halaman *Add New Division*

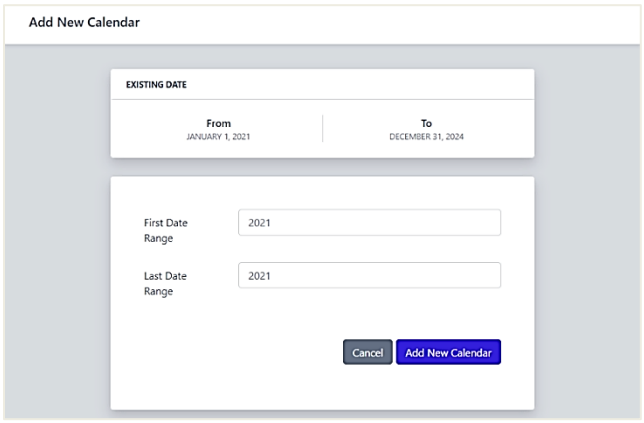
4.3.1.5 Manajemen Data Kalender

Fitur *administrator* selanjutnya adalah fitur manajemen data kalender yang digunakan untuk memanipulasi data kalender. Data kalender sendiri 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

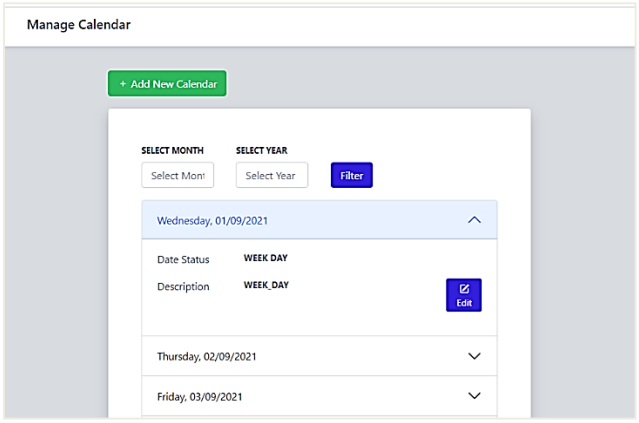
baru menggunakan *form* yang dapat diakses melalui halaman *Add New Calendar* seperti yang ditunjukkan pada Gambar 8.

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.



Gambar 8. *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 data kalender kemudian dilanjutkan dengan pengguna memperbarui status hari pada data kalender yang dapat diakses melalui halaman *Manage Calendar* seperti yang ditunjukkan pada Gambar 9.

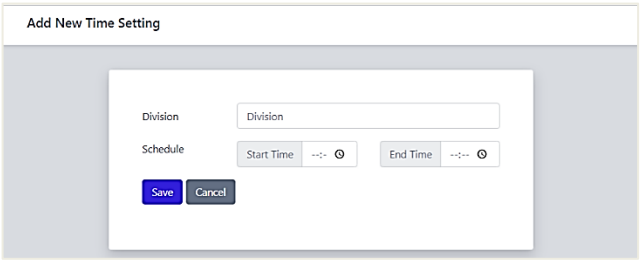


Gambar 9. Halaman *Manage Calendar*

Pada halaman ini data kalender ditampilkan dalam bentuk *accordion* dan pengguna dapat memilih bulan dan tahun menggunakan *filter*. Pengguna kemudian dapat memperbarui status hari menggunakan tombol *edit* yang 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.*

4.3.1.6 Manajemen Time Setting untuk Divisi

Fitur *administrator* selanjutnya adalah fitur untuk melakukan operasi CRUD terhadap data *time settings* atau jam kerja divisi. 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 10.



Gambar 10. Halaman *Add New Time Setting*

Untuk menambahkan jam kerja baru, *administrator* perlu mengisi *form* yang berisi sebuah *dropdown* yang akan digunakan untuk memilih divisi dan dua buah *timepicker* untuk memilih *start time* dan *end time.* Setelah berhasil menambahkan jam kerja baru, *administrator* akan dialihkan ke halaman *Time Settings List* yang menampilkan semua 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.

4.3.1.7 Pembuatan QR Code

Salah satu fitur *administrator* yang selanjutnya adalah fitur membuat *QR Code* yang akan digunakan oleh karyawan untuk mengajukan presensi menggunakan aplikasi *mobile. Administrator* dapat memilih durasi waktu *QR Code* (dalam satuan detik) menggunakan *dropdown* yang tersedia dan memulai *QR Code* menggunakan tombol *Start QR Code.*

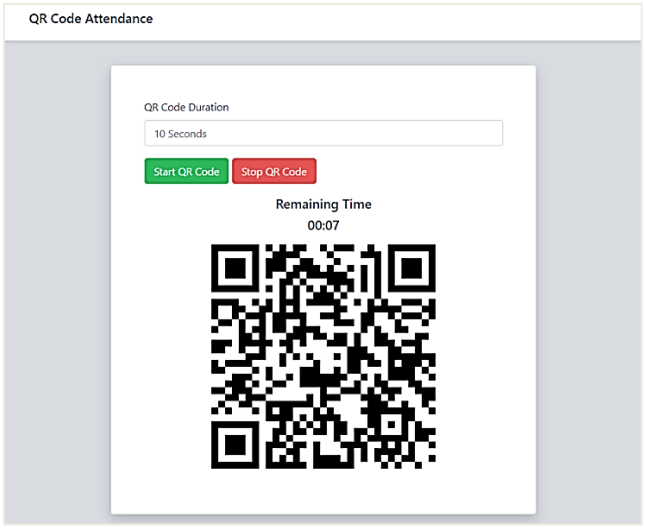
Saat *QR Code* berjalan sistem akan membuat sebuah *QR Code* yang berisi *token* yang terdiri dari lima huruf acak. *Token* tersebut kemudian disimpan ke dalam tabel *qr_tokens* dan akan digunakan untuk mengecek validitas *token* pada *QR Code* yang dipindai oleh karyawan.

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 11, dapat dilihat tampilan dari *QR Code* yang sedang berjalan.

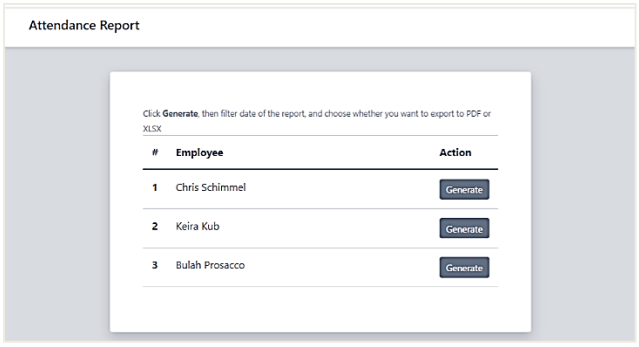
4.3.1.8 Pembuatan Laporan Kehadiran

Berdasarkan hasil analisis fitur yang telah dilakukan, fitur *administrator* yang lainnya adalah fitur untuk membuat laporan bulanan kehadiran karyawan. *Administrator* dapat membuat laporan dengan mengakses halaman *Attendance Report* yang dapat dilihat pada Gambar 12. 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.



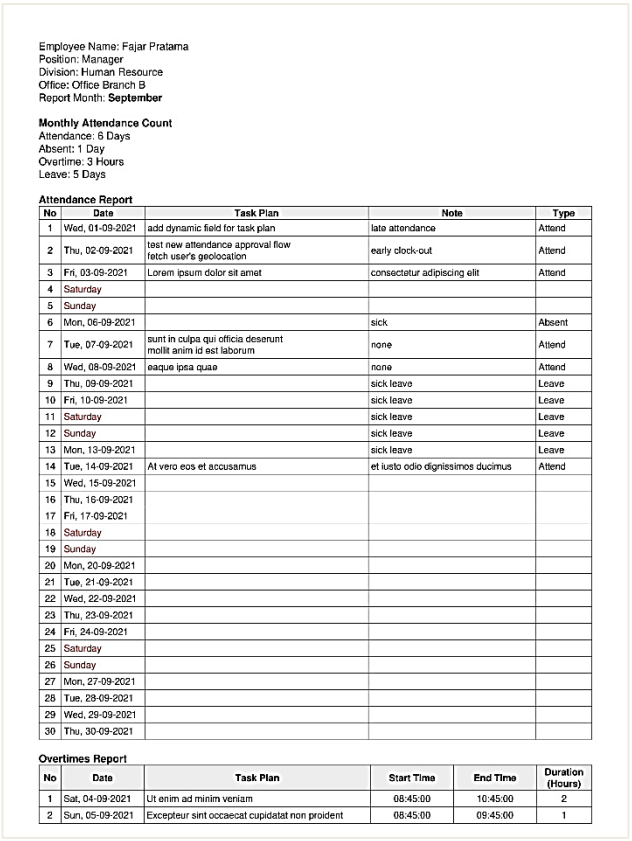
Gambar 11. Tampilan *QR Code yang sedang berjalan*



Gambar 12. 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 13.


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 13. Contoh laporan dalam format *PDF*

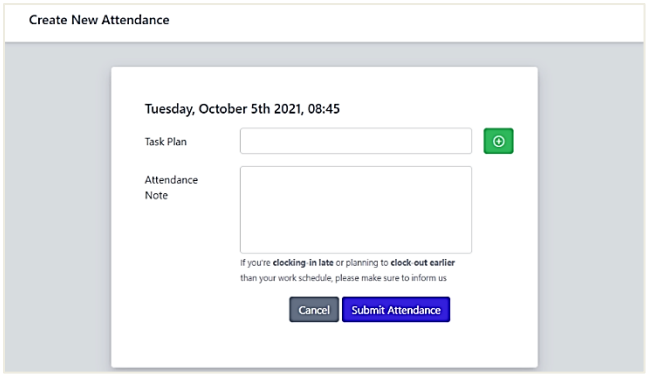
4.3.2 Fitur Karyawan

Setelah fitur-fitur *administrator* selesai dikembangkan, proses pengembangan sistem dilanjutkan dengan mengembangkan fitur-fitur untuk karyawan. Berdasarkan hasil analisis fitur pengguna, secara umum karyawan dapat menggunakan beberapa fitur seperti pengajuan presensi, lembur, izin, dan cuti. Fitur-fitur ini dapat digunakan baik pada aplikasi berbasis *web* ataupun perangkat *mobile.*

4.3.2.1 Pengajuan Presensi

Pada fitur pengajuan presensi menggunakan *web browser*, karyawan dapat mengakses *form* pada halaman *Create New Attendance* yang dapat dilihat pada Gambar 14. *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.

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. Setelah mengajukan presensi, karyawan dapat melihat detail presensi yang diajukan pada halaman *Attendance Details.*

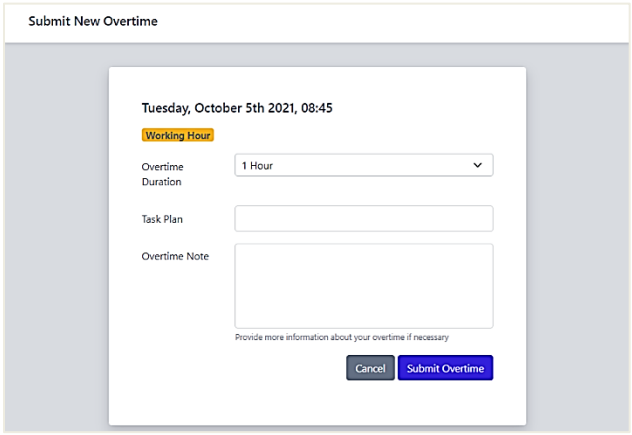


Gambar 14. Tampilan halaman *Create New Attendance*

Detail presensi yang terdapat pada halaman tersebut adalah tanggal, *clock-in time*, *clock-out time*, *task plan* atau tugas yang dikerjakan, *task report* atau laporan pengerjaan tugas, catatan kehadiran, *approval status*, dan *parent* atau atasan karyawan tersebut. Selain itu, jika presensi tidak mendapatkan persetujuan dari atasan, maka *rejection note* atau alasan mengapa presensi tersebut ditolak akan ditampilkan pada halaman tersebut.

4.3.2.2 Pengajuan Lembur

Fitur karyawan yang kedua adalah pengajuan lembur yang dapat dilakukan dengan mengakses halaman *Submit New Overtime* seperti yang ditunjukkan pada Gambar 15.

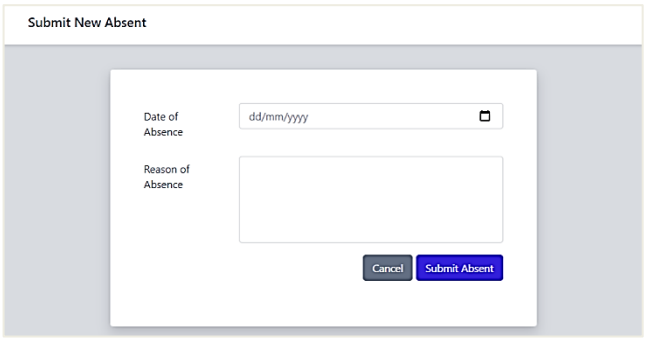


Gambar 15. Halaman *Add New Overtime*

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. 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.

4.3.2.3 Pengajuan Izin

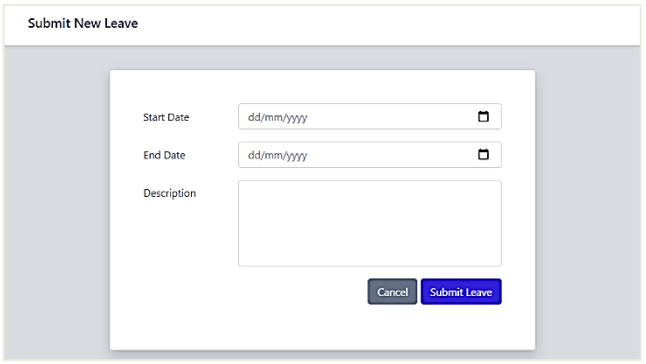
Fitur karyawan yang selanjutnya adalah fitur untuk mengajukan izin. Izin diajukan saat karyawan tidak dapat hadir pada hari kerja karena sakit atau jika ada keadaan yang mendesak. Untuk mengajukan izin, karyawan dapat mengakses *form* pengajuan izin yang terdapat pada halaman *Submit New Absent* dan mengisi beberapa *field* yang ada seperti tanggal pengajuan izin dan alasan izin yang dapat dilihat pada Gambar 16.



Gambar 16. *Form* pengajuan izin

4.3.2.4 Pengajuan Cuti

Pengembangan fitur CRUD untuk karyawan yang selanjutnya adalah fitur untuk mengajukan cuti. Karyawan dapat mengajukan cuti dengan mengakses *form* yang terdapat pada halaman *Submit New Leave* yang ditunjukkan Gambar 17. *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 17. Halaman *Submit New Leave*

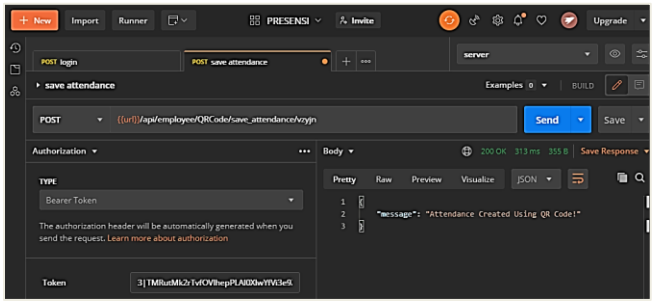
4.3.2.5 Presensi QR Code

Fitur karyawan selanjutnya adalah pengajuan presensi menggunakan *QR Code.* Fitur ini ditujukan kepada karyawan yang bekerja dari kantor atau *on site.* Karyawan yang bekerja dari kantor dapat menggunakan aplikasi *mobile* untuk memindai *QR Code* yang dibuat oleh *administrator*.

Proses yang terjadi ketika *QR Code* dipindai adalah sistem akan mengalihkan karyawan menuju alamat atau *url* yang mengecek apakah *token* yang ada pada *QR Code* tersebut sama dengan *token* yang ada pada *database.* Saat presensi menggunakan *QR Code* berhasil dibuat, sistem akan mengambil lokasi karyawan menggunakan *latitude* dan *longitude* dari *ip address* karyawan, serta mengirimkan *email* notifikasi kepada *parent* atau atasan karyawan untuk meminta persetujuan.

Selain itu, karyawan tidak bisa menentukan *task plan* dan catatan kehadiran pada saat mengajukan presensi karena presensi dibuat menggunakan *QR Code*, maka dari itu *task plan* dan catatan kehadiran pada presensi yang menggunakan *QR Code* harus disunting oleh karyawan yang bersangkutan. Pengajuan presensi menggunakan *QR Code* dapat dilihat pada Gambar 18. Pengujian *endpoint* tersebut dilakukan menggunakan *Postman* melalui *url*

127.0.0.1:8000/api/employee/QRCode/save_attendance/{token}, dimana *token* pada *url* tersebut adalah *token* pada *QR Code* yang dipindai oleh karyawan.

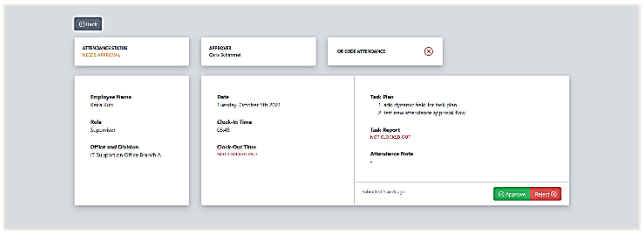


Gambar 18. Pengujian presensi *QR Code* dengan *Postman*

4.3.2.6 Fitur Parent atau Atasan

Salah satu fitur yang terdapat pada sistem ini adalah fitur bagi *parent* atau atasan untuk memberikan persetujuan atau *approval* kepada kehadiran karyawan. *Approval* dapat diberikan kepada semua jenis kehadiran seperti presensi, lembur, izin, dan cuti.

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 19.



Gambar 19. 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.*

4.3.3 Implementasi Service Class

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*. *Laravel* tidak memiliki *command* khusus untuk membuat *service class*, sehingga *service class* tidak dapat dibuat menggunakan *artisan command.* Pada sistem ini *service class* disimpan pada folder app\Http\Services.

4.3.4 Implementasi Cache untuk Optimasi Sistem

Pengembangan sistem kemudian dilanjutkan dengan optimasi *request duration* ke *database* saat halaman sistem pada *web* dimuat. 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 tujun 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.

Sebagai contoh, 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.

Untuk melakukan *debug* dan melihat berapa lama sebuah halaman dimuat serta jumlah *query* yang dijalankan, sistem ini menggunakan sebuah *package* bernama *Laravel Debugbar*. Sebelum cache diimplementasikan, waktu yang dibutuhkan untuk memuat halaman *index* presensi adalah 403ms dengan *query* yang dijalankan sebanyak 16 buah. Sedangkan, setelah *cache* diimplementasikan durasi halaman *index* presensi dimuat menjadi hanya 296ms dengan *query* dijalankan sebanyak 15 buah.

4.3.5 Implementasi Queue and Jobs untuk Mengirim Email Notifikasi

Selain menggunakan *cache* ke dalam *Redis*, sistem ini juga menggunakan *background task* untuk membuat sistem semakin optimal. *Background task* akan digunakan untuk mengirim *email* notifikasi kepada *parent* atau atasan saat ada sebuah presensi yang memerlukan persetujuan. Selain itu, *background task* juga digunakan untuk melakukan 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.

Sebelum *background task* diimplementasikan, proses pengiriman *email forgot password* kepada *administrator* memakan waktu sebanyak 4.99ms atau hampir 5 detik. Sedangkan setelah *background task* diimplementasikan, durasi yang dibutuhkan untuk mengirim *email* berkurang drastis menjadi hanya 327ms .

5. KESIMPULAN

Setelah kegiatan pengembangan Sistem Presensi dilaksanakan, 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*.

6. REFERENSI

[1] Fitria, N. J. L. 2020. Penerapan Work From Home Dan Work From Office Dengan Absensi Online Sebagai Implikasi E-Government Di Masa New Normal Implementation of Work From Home and Work From Office With Online Absence As an E-Government. *Civil Service*. 14, 1, 69–84.

[2] Ken, S. and Jeff, S. 2020. Panduan Definitif untuk Scrum: Aturan Permainan. *Scrum.Org* (November 2020), 1–17.

[3] Khoiriyah, N. L., Marisa, F. and 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. DOI:https://doi.org/10.37438/jimp.v3i1.89.

[4] Husain, A., Prastian, A. H. A. and 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. DOI:https://doi.org/10.33050/tmj.v2i1.319.

[5] Ferdika, R. and Nasution, R. D. 2020. Changes in Orientation of Employee Motivation in The Application of E-Absensi in Ponorogo District. *Jurnal Penelitian Komunikasi Dan Opini Publik*. 24, 1 (Juli 2020), 71–84. DOI:https://doi.org/10.33299/jpkop.24.1.2439.

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

[7] REST vs. SOAP: Choosing the best web service. 2016. Retrieved October 26, 2021 from https://searchapparchitecture.techtarget.com/tip/REST-vs-SOAP-Choosing-the-best-web-service.

[8] Akbar, M. 2018. *Pengembangan Restful Api Untuk Application Specific High Level Location Service*. Universitas Islam Indonesia.

[9] Pranajaya, M.H. 2018. *Perancangan Aplikasi Web Dan Rest Api Untuk Sistem Kehadiran Mahasiswa Berbasis Qr Code Dan Lokasi*. Universitas Sumatera Utara.

[10] Masalha, F. and Hirzallah, N. 2014. A Students Attendance System Using QR Code. *International Journal of Advanced Computer Science and Applications*. 5, 3, 75–79.

[11] Fowler, M., Rice, D., Foemmel, M., Hieatt, E., Mee, R. and Stafford, R. 2002. *Patterns of Enterprise Application Architecture*. Pearson Education.

[12] Kosasi, S. and Liauw, D. 2013. Penerapan Design Pattern Dalam Perancangan Web Order. *Jurnal Teknologi*. 6, 1, 1–9.