KERJA PRAKTIK - IF184801

Perancangan Sistem Informasi Pengelolaan Jamaah Masjid Baitul Haq Keputih

PT. Akfisa Nusantara
Jl. Keputih Tegal Timur A4/37 Surabaya, Jawa Timur 60111

Periode: 13 Maret 2023 - 12 Mei 2023.

Oleh:

Bayu Surya Bawono 05111840000114
Ryan Rasyid Azizi 05111840000079

Pembimbing Jurusan
Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2023

KERJA PRAKTIK – IF184801

Perancangan Sistem Informasi Pengelolaan Jamaah Masjid Baitul Haq Keputih

PT. Akfisa Nusantara
Jl. Keputih Tegal Timur A4/37 Surabaya, Jawa Timur 60111
Periode: 13 Maret 2023 - 12 Mei 2023.

Oleh:

Bayu Surya Bawono	05111840000114
Ryan Rasyid Azizi	05111840000079

Pembimbing Jurusan
Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2023

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

LEMBAR PENGESAHAN
KERJA PRAKTIK

Perancangan Sistem Informasi Pengelolaan Jamaah Masjid Baitul Haq Keputih

Oleh:

Bayu Surya Bawono	05111840000114
Ryan Rasyid Azizi	05111840000079

Disetujui oleh Pembimbing Kerja Praktik:

1. Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.
   NIP. 198508262015042002

   (Pembimbing Departemen)

Muhammad Isa Senoaji, S. Kom.

(Pembina Kerja Praktik)

[Halaman ini sengaja dikosongkan]

Perancangan Sistem Informasi Pengelolaan Jamaah Masjid Baitul Haq keputih

Nama Mahasiswa	: Bayu Surya Bawono
NRP	: 05111740000114
Nama Mahasiswa	: Ryan Rasyid Azizi
NRP	: 05111840000079
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. M.Comp.Sc.

ABSTRAK

PT. Akfisa Nusantara merupakan perusahaan yang didirikan oleh alumni Informatika ITS yang menerima pembuatan atau pengembangan yang berkaitan dengan Pemrograman Web dan Android. Produk yang kami kerjakan saat melakukan Kerja Praktik adalah Sistem Database untuk masjid Baitul Haq Keputih berupa website yang dapat diakses menggunakan akun yang telah didaftarkan, yaitu sistem pendataan jamaah masjid, pengabsenan, laporan dan lainnya. Pengguna utama aplikasi yang kami kembangkan adalah pengurus dari jamaah masjid Baitul Haq Keputih.

Aplikasi dibuat dengan menggunakan framework Laravel dan memiliki beberapa fitur, beberapa diantaranya adalah penambahan data, informasi data, pengeditan data, penghapusan data, baik data jamaah maupun data users. Kami diminta untuk menyiapkan arsitektur yang tepat agar sistem dapat diakses oleh pengurus dengan kendala seminimal mungkin.

Kata Kunci : Pemrograman Web, Android, Website, Laravel

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Perancangan Sistem Informasi Pengelolaan Jamaah Masjid Baitul Haq Keputih.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Ibu Adhatus Solichah Ahmadiyah, S.Kom., M.Sc. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 23 Juni 2023
Bayu Surya Bawono dan Ryan Rasyid Azizi

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini akan membahas tentang poin utama dalam pembuatan laporan kerja praktik yang meliputi latar belakang, manfaat, rumusan masalah, tujuan, lokasi dan waktu kerja praktik, metodologi kerja praktik, serta sistematika laporan.

1.1. Latar Belakang

Perkembangan teknologi dalam idang informasi berkembang sangat pesat dalam beberapa dekade berikut. Mulai dari mesin-mesin sederhana yang hanya bisa mengirim kode- kode sederhana hingga saat ini yang bisa membagikan informasi secara real-time bahkan dengan kapasitas yang sangat besar. Beberapa contoh media informasi saat ini seperti penyiaran televisi, penyiaran radio, penyiaran yang berbasis internet, maupun penyebaran berita melalui teks yang lebih sederhana seperti website.

Fleksibilitas teknologi saat ini dapat dimanfaatkan dengan membangun sistem informasi seperti pendataan, penjadwalan, pembuatan agenda, pencatatan keuangan, serta pembuatan laporan. Hal-hal tersebut di terapkan dalam pembangunan sistem informasi pada masjid Baitul Haq di Keputih. Dalam satu pekan masjid Baitul Haq mengadakan kegiatan pengajian, kebersamaan, serta musyawarah. Oleh karena itu dibutuhkan suatu sistem untuk mempermudah dalam pengelolaan serta pendataan jamaah masjid baitul Haq. Terutama dengan bertambahnya jumlah jamaah masjid serta jumlah kegiatan yang direncanakan. Pada saat kerja praktik, kamu diberi kesempatan untuk merancang dan mengimplementasikan sistem informasi jamaah masjid Baitul Haq keputih.

1.2. Tujuan

Tujuan kerja praktik ini adalah membantu masjid dalam mempermudah sistem informasi dalam masjid Baitul Haq di keputih, Kota Surabaya

1.3. Manfaat

Manfaat yang diperoleh dengan adanya website sistem informasi masjid Baitul Haq antara lain mempermudah dalam pendataan jamaah masjid, mempermudah dalam pelaksanaan agenda kegiatan masjid. Jadi tidak perlu mencatat dalam bentuk laporan buku, namun sudah terintegrasi dengan mudah dalam website yang cukup mudah diakses.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana arsitektur database yang dapat memberikan layanan aplikasi web Sistetem Informasi Jamaah Masjid Baitul Haq?
2. Bagaimana rekayasa yang dapat sistem yang terbentuk bisa menangani ketika terdapat pengguna yang memiliki peran yang berbeda?

1.5. Lokasi dan Waktu Kerja Praktik

Pekerjaan praktik ini dilakukan secara *remote*. Adapun kerja praktik dimulai pada tanggal 13 Maret 2023 sampai 12 Mei 2023..

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Untuk mengetahui kebutuhan dari website, kami mengikuti rapat bersama tim dari PT. Akfisa Nusantara serta perwakilan dari pengurus masjid Baitul Haq. Pada saat rapat kamu dijelaskan bagaimana konsep tugas dan proses dalam peran masing-masing pengurus yang ada kepengurusan masjid Baitul Haq. Setelah dijelaskan, tim PT. Akfisa merumuskan fitur-fitur apa saja yang akan diterapkan pada website yang akan dibuat.

1.6.2. Studi Literatur

Setelah mendapat gambaran bagaimana sistem tersebut berjalan, kami diberitahu tinjauan apa saja yang akan diimplementasikan untuk membuat website beroperasi. Tinjauan yang dipakai meliputi Laravel, PHP >= 7.4, InnoDB, dan lain-lain. Selain itu kami dijelaskan aturan dalam menulis konfigurasi agar konfigurasi yang terbentuk mudah dipahami oleh tim developer lainnya.

1.6.3. Analisis dan Perancangan Sistem

Setelah dilakukan tinjauan, sistem yang dirancang perlu adanya sebuah desain arsitektur sistem yang baik. Pada website ini tim developer setuju menggunakan arsitektur desain MVC (Model - View - Controller).

1.6.4. Implementasi Sistem

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini sistem yang sudah terbentuk akan dilakukan *testing* oleh tim developer.

1.6.5. Pengujian dan Evaluasi

Setelah website yang telah direncanakan telah jadi, perlu adanya evaluasi untuk menguji apakah website sesuai dengan harapan client. Jika masih belum sesuai atau perlu menambah fitur, rapat akan dilakukan lagi untuk memperbaiki atau menambahkan fitur yang diperlukan.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7. Sistematika Laporan

Sistematika laporan dalam pembuatan buku kerja praktik meliputi :

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum PT. Akfisa Nusantara mulai dari profil, lokasi perusahaan.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

Bab ini akan membahas tentang perusahaan kerja praktik yang meliputi profil perusahaan serta lokasi perusahaan.

2.1. Profil PT. Akfisa Teknologi Nusantara

Akfisa Nusantara merupakan perusahaan PT (Perseroan Terbatas) yang didirikan oleh Muhammad Isa Senoaji, dan Akbar Noto Ponco Bimantoro yang mana beliau merupakan alumni Informatia ITS. Akfisa didirikan pada tahun 2018 yang berada di Provinsi Jawa Timur, Surabaya, Keputih. Akfisa Nusantara adalah sebuah Start-up yang bergerak dibidang pembuatan dan pengembangan perangkat lunak/software. Akfisa Nusantara melayani pembuatan maupun pengembangan aplikasi berbagai platform diantaranya Android, Web dan Desktop. Informasi mengenai perusahaan dapat diakses melalui Instagram @akfisanusantara.

Akfisa Nusantara terdiri dari departemen Administrasi & Finance serta departemen IT. Departemen administrasi & Finance bertanggung jawab atas pengelolaan keuangan serta perencanaan dan pengawasan. Sedangakan departemen IT bertanggung jawab atas pengembangan dan pemeliharaan aplikasi, dukungan teknis serta inovasi teknologi.

Akfisa Nusantara telah melakukan beberapa *project*, beberapa diantaranya presensi *with face recognition, security administration, patrol guard, scheduled cleaning task & report*, *logistic tracker* dan *sales data tracker*.

2.2. Lokasi dan Kontak

Jl. Keputih Tegal Timur A4/37 Surabaya, Jawa Timur 60111.
Nomor Hp: +62 858 1463 7516.
Email :nusantaraakfisa@gmail.com.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

Pada bagian ini akan memaparkan tentang referensi ilmiah yang menjadi landasan kerja praktik serta studi literatur terkait kerja praktik yang telah dilakukan sebelumnya. Studi literatur tersebut mencakup tinjauan penerapan *website* terdahulu, tinjauan teori tentang *website* serta perancangan *website*.

3.1. Sistem Informasi

Sistem informasi adalah suatu rangkaian yang saling terkait dan bekerja sama untuk mengumpukan, menyimpan, mengelola, memproses dan menyampaikan informasi yang relevan kepada pengguna [1]. Dalam manajemen sistem informasi bukanlah termasuk tantangan baru saat ini, karena hal ini sudah berjalan dalam beberapa dekade [2]. Saat ini memanajemen sistem informasi dibantu dengann banyak teknologi yang mempermudah informasi diolah dan ditata. Sehingga dalam praktisi sistem informasi saat ini membutuhkan strategi baru dalam pengelolaan sistem informasi seperti menggabungkan pengetahuan teori dan pengetahuan manajemen untuk mencapai tujuan manajemen informasi yang diinginkan.

3.2. Website

Website (Halaman Web) adalah kumpulan halaman digital yang berisi informasi berupa teks, gambar, animasi, video, suara, maupun gabungan dari kesemua hal tersebut yang terkoneksi dengan internet serta elemen interaktif lainnya [3], sehingga dapat dilihat oleh siapapun yang terterkoneksi dengan jaringan internet [4]. Jenis-jenis kategori website terbagi menjadi:

3.1.1. Web Statis

Web Statis merupakan jenis website yang mempunyai halaman yang tidak berubah (statis). Perubahan suatu halaman dilakukan secara manual dengan mengedit kode (*source code*) yang menjadi struktur dari website tersebut. Selain dari itu *website* statis tidak memiliki elemen interaksi langsung antara pengguna dengan *website* [5]. *Website* statis sering digunakan untuk tujuan yang sederhana seperti perkenalkan perusahaan, penyajian informasi layanan, maupun portofolio. Walaupun dibilang tidak meiliki intraksi langsung serta memiliki keterbatasan dalam perubahan konten, web statis memiliki beberapa keuntungan seperti pembuatan dan pemuatan yang cenderung lebih cepat, konten serta pengaksesan yang bisa dibilang ringan.

3.1.2. Web Dinamis

Web dinamis merupakan jenis website yang secara terstruktur diperuntukan untuk diperbaharui secara berkala. Selain dari itu web dinamis juga merupakan web yang menghasilkan konten secara dinamis berdasarkan input pengguna ataupun data lainnya [6]. Secara umum website tersebut disediakan halaman tersendiri untuk melakukan perubahan konten dari halaman web tersebut. Seperti: web portal, web berita, dan lain sebagainya.

3.1.3. Web Interaktif

Web Interaktif merupakan jenis website yang berinteraksi antara penggunanya menggunakan konten ataupun elemen yang dikandungnya. Web ini dirancang untuk memberikan pengalaman yang lebih aktif dan segar kepada pengguna. Umumnya berupa forum diskusi maupun blog. Dimana adanya moderator sebagai pengatur alur jalannya diskusi. [1]

3.3. Pemrograman Web

Menurut Ani dkk., pemrograman web adalah pembuatan aplikasi program web dengan bahasa skrip yang akan menghasilkan sebuah aplikasi yang dapat diakses menggunakan web browser. Bahasa skrip merupakan kode-kode yang menjalankan fungsi-fungsi tertentu, mengatur tampilan interaksi pengguna, serta pengelolaan dan memanipulasi data disisi klien maupun server. Bahasa skrip diantaranya adalah HTML, CSS, PHP, dan JavaScript. Selain itu pengembang aplikasi program web dapat menggunakan framework guna mengembangkan berbagai jenis web yang lebih kompleks dan interaktif serta mempermudah dalam pembuatan website, salah satunya adalah Laravel. [1]

3.4. HTML

Hypertext Markup Language (HTML) adalah bahasa pemrograman yang digunakan untuk menampilkan sebuah halaman web. HTML termasuk salah satu bahasa pemrograman gratis yang dapat diartikan sebagai bahasa yang dikembangkan bersama-sama secara global, yang mana HTML merupakan bahasa standar internasional [1].

Dalam bekerja dengan HTML, menggunakan struktur kode sederhana berupa tag dan atribut untuk memformat halaman website, seperti misalnya membuat paragraf dengan tag pembuka dan tag penutup . Meskipun telah mengalami banyak perubahan sejak dirilis pertama kali pada tahun 1991 oleh Tim Berners-Lee, HTML terus berkembang dengan versi-versi baru yang menambahkan tag dan atribut baru untuk mendukung tuntutan perkembangan teknologi web.

3.4.1. Elemen HTML

HTML memiliki elemen yang merupakan dasar dalam HTML. Setiap elemen terdiri dari sebuah tag (pembuka dan penutup),

sedangkan konten berada diantara tag pembuka dan penutup. Contoh elemen dan konten HTML diunjukkan pada sumber kode 3.1.

```
1    <p>Ini adalah paragraf contoh.</p>
```

Sumber kode 3. 1 Elemen dan konten laravel

Pada kode sumber 3.1 menampilkan contoh elemen HTML berupa tag untuk paragraf. Selain dari itu, elemen HTML dapat ditempatkan dalam elemen lainnya dengan contoh pada sumber kode 3.2.

```
1    <div>
2      <p>Ini adalah paragraf dalam div.</p>
3    </div>
```

Sumber kode 3. 2 Elemen HTML di dalam elemen HTML lainnya

3.4.2. Struktur Dasar HTML

HTML memiliki struktur dasar berupa beberapa tag yang menandakan bagian-bagian dari html. Struktur HTML terbagi menjadi beberapa tag, pertama tag “<html>” yang menandakan awal dari dokumen HTML, kedua “<head>” yang berisi tentang dokumen HTML seperti judul halaman, tautan file eksternal (CSS dan JavaScript), meta-data. Struktur ketiga “<body>” yang berisi semua konten yang akan ditampilkan pada halaman web seperti teks, gambar, tautan, video dan elemen lainnya. Contoh sumber kode 3.3 merupakan struktur dasar HTML.

```
<html>
 <head>
   <title>Halaman Web dengan CSS</title>
 </head>

 <body>
   <h1>Selamat Datang</h1>
   <p>Ini adalah paragraf contoh struktur HTML.</p>
 </body>
</html>
```

Sumber kode 3. 3 Contoh struktur HTML

Dengan menggunakan HTML, pengembang dapat menggambarkan struktur dan isi semantik dari sebuah dokumen web. Selain itu, dalam pengembangan web, HTML sering digabungkan dengan CSS (*Cascading Style Sheets*) dan JavaScript. CSS digunakan untuk memperindah tampilan visual halaman web, sementara JavaScript digunakan sebagai bahasa pemrograman di sisi klien untuk memberikan interaktivitas dan fungsionalitas pada halaman web tersebut. Dengan menggabungkan HTML, CSS, dan JavaScript, pengembang dapat menciptakan halaman web yang menarik secara visual dan memiliki interaksi yang dinamis dengan pengguna.

3.5. PHP

Hypertext Preprocessor (PHP) adalah bahasa skrip berbasis server (server-side) yang mampu mengolah kode PHP dari kode web dengan ekstensi .php, sehingga menghasilkan tampilan halaman web yang dinamis di sisi client (browser) [1].

Bahasa pemrograman PHP memiliki perbedaan dengan JavaScript dan Python dalam penggunaannya. PHP lebih umum digunakan sebagai bahasa untuk komunikasi sisi server, sementara JavaScript dapat digunakan baik untuk frontend maupun backend. Sedangkan Python cenderung digunakan hanya untuk sisi server (backend).

Bahasa penulisan skrip, seperti PHP, berfungsi untuk mengotomatiskan eksekusi tugas dalam environment runtime khusus. Tugas ini meliputi instruksi kepada halaman statis yang telah dibuat dengan HTML dan CSS untuk melakukan tindakan tertentu sesuai dengan aturan yang telah ditetapkan.

Sebagai contoh, dengan menggunakan skrip, kita dapat memvalidasi apakah semua kolom dalam sebuah form telah diisi sebelum form tersebut dikirimkan kembali ke server. Jika terdapat kolom yang masih kosong, maka skrip tersebut akan memberikan peringatan kepada pengguna. Bahasa penulisan skrip, seperti PHP, juga sering digunakan untuk menciptakan efek drop-down ketika kursor menyoroti menu utama, melakukan rollover tombol dan animasi, membuka kotak dialog, dan berbagai tindakan lainnya.

Bahasa penulisan skrip bisa dijalankan di sisi klien (frontend) atau di sisi server (backend). Skrip sisi klien diproses oleh web browser, sementara skrip sisi server diolah oleh server sebelum dikirimkan ke browser. Dengan demikian, kode sumber dalam skrip sisi server dapat disembunyikan dari pengguna, sedangkan skrip sisi klien bisa dilihat dengan mudah oleh pengguna.

Secara dasar, struktur bahasa pemrograman PHP memiliki beberapa komponen penting yang membentuk kerangka dasar pemrograman PHP.

3.5.1. Tag PHP

Setiap kode PHP harus dimulai dengan tag pembuka “<?php” yang menandakan awal dari blok kode PHP. Akhir dari blok kode PHP akan ditutup dengan “?>” yang merupakan pemisah dari blok kode PHP dengan blok kode lainnya yang biasanya dipadukan dengan HTML. Sumber kode 3.4 merupakan contoh dari PHP yang dipadukan dengan HTML.

```
1    <!DOCTYPE html>
2    <html>
3    <head>
4      <title>Contoh Print String</title>
5    </head>
6    <body>
7      <?php
8        $pesan = "Ini contoh HTML dan PHP";
9        echo $pesan;
10     ?>
11   </body>
12   </html>
```

Sumber kode 3. 4 Contoh PHP dipadukan dengan HTML

Pada sumber kode 3.4 baris kode 7 hingga 10 merupakan baris kode PHP. Pada sumber kode 3.4 kolaborasi dari HMTL dan PHP akan menampilkan keluaran berupa kalimat "Ini contoh HTML dan PHP".

3.5.2. Variabel dan Output PHP

Variabel merupakan sebuah media atau penyimpanan data sementara dalam bahasa pemrograman. Dalam bahasa pemrograman PHP variabel harus dimulai dengan tanda dolar "$" yang diikuti dengan nama variabel yang diinginkan. Selain dari itu, variabel bersifat dinamis yang artinya tidak perlu mendeklasrasikan tipe datanya secara eksplisit. Contoh variabel dalam bahasa pemrograman PHP ditunjukkan pada sumber kode 3.5.

```
1    <?php
2     $nama = "John";
3     $umur = 30;
4    ?>
```

Sumber kode 3. 5 Contoh variabel PHP

untuk menampilkan isi data dalam pemrograman PHP digunakan fungsi “echo” atau “print”. Sumber kode 3.6 menampilkan contoh keluaran dari PHP.

```
1    <?php
2     $nama = "John";
3     $umur = 30;
4     echo $umur;
5     print $umur;
6    ?>
```

Sumber kode 3. 6 Contoh keluaran dalam PHP

Pada sumber kode 3.6 baris 4 dan 5 merupakan baris kode PHP untuk mengeluarkan isi data dari variabel “$umur”.

3.6. Bootstrap

Bootstrap merupakan kerangka kerja atau *framework front-end* yang digunakan untuk mengembangkan desain dan tampilan situs web secara responsif dan mudah diakses [7]. Dalam bootstrap terdapat beberapa poin penting, diantaranya:

a. Responsif
   Bootstrap dirancang untuk menciptakan situs web yang responsif, yang artinya tampilannya akan menyesuaikan dengan perangkat dan ukuran layar seperti komputer, tablet ataupun *smartphone*. Dengan tampilan yang responsif, pengguna dapat mengakses situs dengan nyaman.
b. Komponen Siap Pakai
   Bootstrap menyediakan berbagai komponen antarmuka yang siap pakai seperti tombol, formulir, navigasi, carousel, *card*, dll. Dengan menggunakan komponen ini, pengembang dapat

menghemat waktu dan usaha karena tidak perlu membangun semuanya dari awal.

c. Kustomisasi Mudah
   Meskipun memiliki desain dan gaya bawaan, Bootstrap memungkinkan pengembang untuk dengan mudah menyesuaikan tampilan situs sesuai kebutuhan. Pengembang dapat menambahkan atau mengganti kelas CSS, mengubah variabel, atau menulis gaya khusus sesuai preferensi.
d. Dokumentasi yang Lengkap
   Bootstrap menyediakan dokumentasi yang komprehensif dan mudah diakses. Dokumentasi ini berisi panduan lengkap tentang cara menggunakan dan mengintegrasikan framework ini, sehingga pengembang dapat dengan mudah memahami dan memanfaatkannya.
e. Grid System
   Bootstrap menggunakan grid system yang fleksibel untuk mengatur tata letak halaman web. Grid system membagi layar menjadi kolom-kolom yang dapat diatur, memudahkan pengembang untuk menyesuaikan dan mengorganisir konten situs dengan mudah.

3.7. Javascript

JavaScript merupakan sebuah bahasa pemrograman tingkat tinggi yang bersifat dinamis. Keunggulan JavaScript terletak pada berbagai fungsionalitas yang dapat diaplikasikan, seperti pengembangan aplikasi web, backend, aplikasi desktop, Internet of Things (IoT), dan banyak lagi. Dalam konteks buku kerja praktik ini, JavaScript digunakan sebagai bahasa scripting di sisi klien yang terintegrasi dengan HTML pada suatu website. Selain itu, JavaScript juga menyediakan beragam library yang dapat

dimanfaatkan, seperti Node.js, Axios.js, Bluebird.js, Vue.js, Angular.js, React.js, Animate.js, dan sejumlah lainnya. Library-library tersebut memperluas kemampuan JavaScript dan memfasilitasi pengembang dalam membangun aplikasi yang lebih kompleks dan interaktif.

Dahulunya, JavaScript dikenal dengan nama LiveScript sebelum kemudian berganti nama menjadi JavaScript agar bisa terlihat sejajar dengan bahasa pemrograman Java milik mitra mereka, Sun Microsystems. Sejak awalnya, JavaScript terus mengalami perkembangan seiring dengan kemunculan browser-web baru seperti Mozilla Firefox dan Google Chrome. Bahkan, saat ini ada mesin JavaScript modern yang disebut V8 yang bertugas untuk mengompilasi bytecode menjadi kode mesin asli, mempercepat eksekusi program JavaScript.

3.8. Laravel

Laravel adalah sebuah framework aplikasi web yang juga bersifat open source, dikembangkan dengan menggunakan bahasa pemrograman PHP. Framework ini menggunakan konsep Model-View-Controller (MVC) yang memungkinkan pengembang untuk membangun aplikasi web dengan lebih efisien. Salah satu keunggulan Laravel adalah kemampuannya dalam menyediakan berbagai fitur dan alat bantu yang kuat untuk mempercepat proses pengembangan. Framework ini menyediakan komponen-komponen yang siap pakai, seperti autentikasi pengguna, manajemen sesi, routing yang fleksibel, dan integrasi dengan sistem basis data. Dengan adanya fitur-fitur ini, pengembang dapat fokus pada logika bisnis yang spesifik, tanpa perlu membangun fitur dasar dari nol.

Selain itu, Laravel memiliki sintaks yang mudah dipahami dan konsisten, sehingga memudahkan pengembang dalam menulis kode yang bersih dan terstruktur. Framework ini juga menyediakan sistem template yang kuat, yang memisahkan logika presentasi (View) dari logika aplikasi (Controller), sehingga memungkinkan

pengembangan aplikasi web yang mudah di-maintain dan di-extend. Dokumentasi Laravel yang komprehensif dan berlimpah contoh implementasi kodenya juga menjadi nilai tambah. Hal ini memudahkan pengembang dalam mempelajari dan mengimplementasikan fitur-fitur Laravel dengan mudah [2].

3.8.1. Composer dan Artisan

Composer adalah alat manajemen ketergantungan yang digunakan dalam Laravel untuk mengelola pustaka-pustaka PHP yang dibutuhkan dalam proyek. Dengan Composer, pengembang dapat dengan mudah menambahkan dan menghapus pustaka-pustaka serta memastikan bahwa proyek menggunakan versi pustaka yang sesuai.

Artisan adalah alat baris perintah (CLI) bawaan Laravel yang memudahkan pengembang dalam melakukan berbagai tugas seperti pembuatan struktur proyek, pengelolaan basis data, pembuatan migrasi, dan sebagainya. Artisan menyediakan perintah yang siap pakai yang sangat membantu dalam kegiatan pengembangan sehari-hari.

3.8.2. Blade Template Engine

Blade adalah engine template bawaan Laravel yang digunakan untuk menghasilkan tampilan (HTML) dari sisi server. Blade menyediakan sintaks ringkas dan mudah dipahami, seperti loop, kondisional, dan pemanggilan komponen, yang memudahkan pengembang dalam mengelola tampilan dengan lebih terstruktur dan mudah dipelihara.

3.8.3. Routing

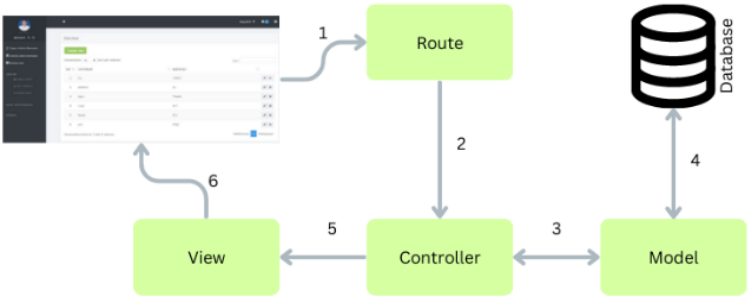
Laravel menyediakan sistem routing yang kuat yang memungkinkan untuk menentukan bagaimana URL ditangani oleh aplikasi. Melalui sistem routing ini, Pengembang dapat mendefinisikan URL ke tindakan (controller) tertentu dalam aplikasi, sehingga ketika pengguna mengakses URL tersebut, aplikasi akan mengeksekusi tindakan yang sesuai.

3.8.4. ORM (Object-Relational Mapping)

Laravel menggunakan ORM yang disebut Eloquent. ORM adalah teknik yang memungkinkan pengembang untuk berinteraksi dengan basis data menggunakan objek-objek PHP, bukan dengan menggunakan kueri SQL secara langsung. Dengan Eloquent, pengembang dapat dengan mudah mengambil, menyimpan, mengubah, dan menghapus data dari basis data menggunakan sintaks yang lebih mudah dipahami dan menghindari paparan langsung ke SQL.

3.8.5. Dasar Framework Model View Controller

Laravel mengikuti pola arsitektur MVC (Model-View-Controller) yang merupakan pendekatan terstruktur untuk mengembangkan aplikasi web. Model mengelola logika bisnis dan interaksi dengan basis data, View menangani tampilan atau antarmuka pengguna, dan Controller mengatur aliran aplikasi serta menghubungkan Model dan View. Dengan pendekatan MVC, pengembangan aplikasi web menjadi lebih terorganisir dan memudahkan dalam pemeliharaan serta pengembangan fitur baru. Ilustrasi hubungan antara model, view dan controller ditunjukkan pada gambar 3.1.



Gambar 3. 1 Relasi antara model, view dan controller

Pada gambar 3.1 menunjukkan antara hubungan dan relasi antara model, view dan controller yang ditunjukkan dengan angka-angka. Angka 1 menunjukkan memasukkan permintaan dari pengguna kepada route, yang kemudian route akan melakukan proses pada angka 2, angka 2 merupakan sebuah route mencari controller yang sesuai dengan permintaan dari pengguna, yang kemudian akan dilanjutkan dengan angka 3 jika permintaan dari pengguna membutuhkan interaksi dengan database, angka 3 merupakan controller berinteraksi dengan model terkait pemrosesan permintaan pengguna yang akan mengambil data dari database, secara tidak langsung model merupakan jembatan antara controller dengan database. Setelah pemrosesan pada controller telah selesai maka controller memberikan hasil dari permintaan pengguna kepaada view yang ditunjukkan pada angka 5. Dan terkahir pada angka 6 permintaan pengguna akan direnderpada view.

A. Model

Komponen Model bertanggung jawab untuk mengelola logika bisnis dan data aplikasi. Model merepresentasikan entitas atau objek dalam aplikasi, seperti pengguna, produk, atau pesanan. Model menyimpan dan memanipulasi data, serta berisi aturan-aturan bisnis yang berlaku untuk entitas tersebut. Selain itu, Model juga berkomunikasi dengan basis data untuk menyimpan, mengambil, memperbarui, dan menghapus data yang diperlukan.

B. View

Komponen View menangani tampilan atau antarmuka pengguna. View bertugas untuk menampilkan data dari Model kepada pengguna dalam bentuk yang sesuai. Tampilan bisa berupa halaman web, form, grafik, atau elemen tampilan lainnya. Dalam pola MVC, View tidak berisi logika bisnis, melainkan hanya berfungsi sebagai representasi visual dari data yang diberikan oleh Model. View juga menerima input dari pengguna, seperti form input, yang kemudian akan diteruskan ke Controller untuk diproses.

C. Controller

Komponen Controller adalah pengendali utama dalam pola MVC. Controller bertanggung jawab untuk mengatur alur aplikasi dan berfungsi sebagai perantara antara Model dan View. Ketika pengguna berinteraksi dengan aplikasi melalui View, Controller menangkap permintaan tersebut dan menentukan tindakan yang harus diambil berdasarkan permintaan tersebut. Controller kemudian berinteraksi dengan Model untuk mengambil data yang diperlukan, memproses

data, dan menyiapkan data yang relevan untuk ditampilkan oleh View. Setelah itu, Controller akan mengirimkan data tersebut ke View yang sesuai untuk ditampilkan kepada pengguna.

3.8.6. Environment dan Middleware

Environment (lingkungan) dalam Laravel merujuk pada konfigurasi dan pengaturan yang berbeda yang dapat diatur sesuai kebutuhan aplikasi. Setiap aplikasi Laravel berjalan dalam suatu lingkungan tertentu yang dapat diatur dalam file .env. File ini berisi variabel-variabel konfigurasi yang mempengaruhi cara aplikasi beroperasi di lingkungannya. *Environment* ini memungkinkan pengembang untuk menyesuaikan konfigurasi aplikasi sesuai dengan tahap pengembangan (*development*), pengujian (*testing*), dan produksi (*production*).

Dalam file .env, variabel-variabel seperti koneksi database, kunci enkripsi, level error yang ditampilkan, dan penyedia layanan tambahan dapat dikonfigurasi. Variabel-variabel ini dapat diakses dengan mudah dalam kode menggunakan fungsi env() yang disediakan oleh Laravel. Penggunaan lingkungan memungkinkan untuk mengubah pengaturan aplikasi tanpa mengubah kode inti, memungkinkan lebih mudahnya pengelolaan dan keamanan pada setiap lingkungan yang berbeda.

Middleware adalah fitur dalam Laravel yang digunakan untuk memproses permintaan HTTP sebelum atau setelah tindakan (*action*) yang terkait dengan permintaan tersebut dijalankan. Middleware memberikan cara untuk memodifikasi permintaan atau respons sebelum atau setelah mencapai tindakan yang dituju, dan juga digunakan untuk menerapkan logika tertentu yang diperlukan dalam pengolahan permintaan.

Contoh penggunaan middleware termasuk otentikasi pengguna, verifikasi CSRF (*Cross-Site Request Forgery*), pengecekan izin, dan banyak lagi. *Middleware* dijalankan dalam urutan tertentu dan dapat diterapkan pada rute-rute tertentu atau grup rute dalam file routes/web.php atau routes/api.php.

Middleware diimplementasikan sebagai kelas yang berisi logika yang harus dijalankan pada permintaan tertentu. Laravel menyediakan beberapa *middleware* bawaan yang berguna, dan juga dapat membuat middleware kustom sesuai kebutuhan aplikasi.

Dengan menggunakan middleware, logika tertentu dapat dipisahkan dari tindakan utama (controller) dan memastikan bahwa setiap permintaan melewati tahap-tahap tertentu sebelum atau sesudah diproses. Hal ini membantu dalam menambahkan lapisan keamanan dan logika bisnis ke dalam aplikasi dengan cara yang terstruktur dan terorganisir.

3.9. InnoDB

InnoDB adalah mesin penyimpanan dalam sistem manajemen basis data SQL. Mesin ini digunakan untuk menyimpan data dengan relasi antar data yang lain. Dalam InnoDB, operasi seperti membuat, membaca, mengubah, dan menghapus data dilakukan melalui query yang dikirim ke basis data MariaDB. Query tersebut kemudian diproses oleh InnoDB sesuai dengan perintah yang terdapat dalam query tersebut. InnoDB memiliki fitur-fitur penting seperti dukungan integritas referensial, dan foreign key. Fitur-fitur ini membantu menjaga konsistensi dan keakuratan data dalam basis data, serta memungkinkan operasi kompleks pada data dengan kekokohan yang tinggi.

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

Pada bab ini akan memaparkan tentang analisis dan perancangan infrastruktur sistem yang di bangun

4.1. Analisis Sistem

Pada sub bab ini akan dijelaskan mengenai tahapan dalam membangun infrastruktur aplikasi sistem infotmasi jamaah masjid baitul haq yaitu analisis dari infrastruktur sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan.

4.1.1. Definisi Umum Aplikasi

Secara umum, Perancangan Sistem Informasi Jamaah Masjid Baitul Haq merupakan sistem pengelolaan jamaah masjid Baitul Haq berbasis website yang diperuntukkan pengurus masjid Baitul Haq Keputih. Pembuatan aplikasi ini berlandasan karena banyaknya jamaah masjid baik dari pribumi ataupun pendatang yang kurang terorganisir dengan baik sehingga pendataan menjadi tidak teratur.

Selain dari itu, untuk memudahkan pengurus masjid dalam mengelola informasi mengenai jamaah dan memantau tingkat keaktifan jamaah dalam berpartisipasi dalam kegiatan masjid, diperlukan suatu sistem yang terorganisir dengan baik. Sistem ini akan membantu mengumpulkan dan menyimpan data tentang jamaah serta memudahkan pemantauan terkait kehadiran dan partisipasi mereka dalam kegiatan keagamaan.

Dengan adanya sistem yang efisien, pengurus masjid akan dapat dengan lebih mudah mengakses dan mengelola informasi yang relevan tentang jamaah, seperti jumlah anggota, status keanggotaan, dan catatan kehadiran dalam berbagai acara masjid. Selain itu, sistem ini juga akan memudahkan pengurus masjid

dalam mengidentifikasi jamaah yang masih aktif dan yang perlu mendapatkan perhatian lebih untuk meningkatkan partisipasi mereka. Pengelolaan informasi yang dibangun meliputi pengelolaan data jamaah, penambahan data jamaah, penghapusan data jamaah, pengelolaan pengurus masjid, penambahan pengurus masjid dan penghapusan pengurus masjid. Sistem ini memiliki penggunaan dari berbagai peran yang mengaksesnya seperti pengurus masjid, admin, dan super admin.

4.2.2. Analisis Kebutuhan

Dengan penjelasan deifinisi aplikasi pada sub bab 4.2.1, maka analisis kebutuhan yang dilakiukan ssebagai berikut:

a. Otentikasi dan pembagian *role* atau peran sesuai akses pada aplikasi yang dibangun untuk membedakan super admin, admin, dan juga pengurus masjid. Pengurus masjid terdiri dari kyai (KI), wakil KI, Mubaligh (MT), keuangan (KU), penerobos dan pengabsen. Peran ditunjukkan pada tabel 4.1.

Peran
Super admin
Admin
KI
Wakil
MT
KU
Penerobos
Pengabsen

Tabel 4. 1 Peran pada sistem

b. *Create, read, update* dan *delete* pada master data jamaah masjid untuk mempermudah dalam pengelolaan data-data jamaah sehingga dapat memanntau tingkat keaktifan jamaah serta tingkat partisipasi dalam kegiatan masjid.

4.2. Job Desk

Pada sub bab ini akan dijelaskan mengenai pembagian *job desk* yang diberikan dalam pembangunan aplikasi sistem. *Job desk* tersebut sebagai berikut:

A. Break Down Laravel :
   a. Pembuatan *template* (Bayu).
   b. *Design database* (Rasyid).
   c. Pembuatan *migration* (Rasyid).
   d. Pembuatan *controller* (Bayu).
   e. Pembuatan *middleware* (Bayu).

B. Fitur :
   a. Master data jama'ah (Rasyid).
   b. Manajemen user dan role (Bayu).

C. Requirement :
   a. Otentikasi dan pembagian role sesuai akses (Bayu).
   b. *Create, read, update* dan *delete* master data (Rasyid).

D. Optional Requirement :

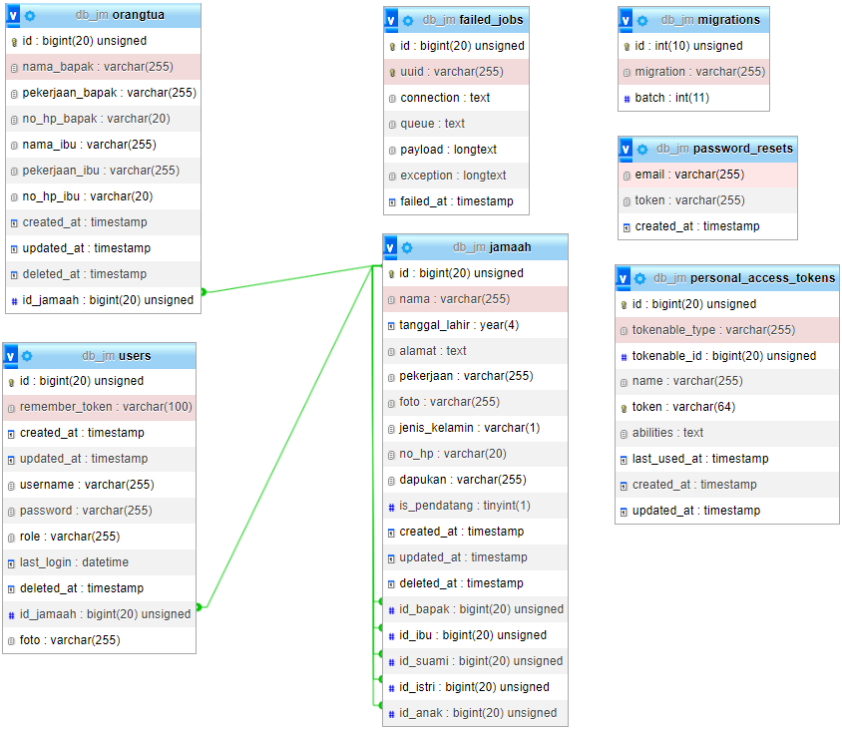
   Pembuatan foto *private* yang tidak bisa diakses publik :
   i. Menampilkan foto user yang tidak bisa diakses publik pada *template* web (Bayu).
   ii. Menampilkan foto privat jama'ah masjid yang tidak bisa diakses publik (Rasyid).

4.3. Perancangan Infrastruktur Sistem

Pada sub bab ini akan menjelaskan terkait desain database sistem dalam membangun infrastruktur perancangan sistem informasi jamaah masjid baitul haq.

4.3.1. Desain Sistem Database

Database diperlukan untuk menyimpan data-data untuk website. Berikut ini adalah *design* database yang diperlukan dan bersifat sementara dikarenakan akan ada pengembangan-pengembangan yang lebih lanjut untuk masa yang akan datang. *Design* database dapat dilhat pada Gambar 4.1.



Gambar 4. 1 design database

Pada gambar 4.1 merupakan desain database yang bersifat sementara untuk perancangan sistem informasi Masjid Baitul Haq Keputih, demikian hal tersebut bersifat sementara dikarenakan memungkinkan untuk pengembang menambahkan fitur fitur baru.

Dalam Gambar 4.1, desain database tersebut digunakan untuk menyimpan data-data jamaah masjid Baitul Haq Keputih.

Pada desain database tersebut terdapat tabel yang memiliki *self-join* yang mana hal tersebut memungkinkan tabel jamaah memiliki relasi menuju id-id bersifat *nullable* (dapat dikosongkan) didalam tabel itu sendiri. Dari semua tabel pada desain database tersebut menyimpan data pribadi, mulai dari individu tersebut maupun orangtua maupun pasangan individu. Kemudian terdapat tabel user yang mana tabel tersebut digunakan untuk mengakses halaman web yang telah kami kerjakan, dan memungkinkan user itu sendiri bukan termasuk bagian dari tabel jamaah.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang dibangun. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi load balancing PHP, implementasi load balancing database, dan implementasi database utama.

5.1. Konfigurasi Laravel

Konfigurasi laravel berfokus pada pengaturan environment atau lingkungan pengembangan aplikasi. serta komponen-komponen mendukung yang dibentuk pada laravel sebagai berikut:

5.1.1. Konfigurasi Environment

Pada implementasi laravel, environment adalah konfigurasi yang digunakan untuk menyesuaikan aplikasi berdasarkan lingkungan yang digunakan, serta mempermudah pengelolaan konfigurasi aplikasi dengan mudah, ringkas dan padat. Adapun konfigurasi ditampilkan pada sumber kode 5.1:

```
1    APP_NAME=Laravel
2    APP_ENV=local
3    APP_KEY=base64:INvevRi2IqKy6EM2SwFbuTOmU39iq42gmgr828Oiznw=
4    APP_DEBUG=true
5    APP_URL=http://localhost
6    LOG_CHANNEL=stack
7    LOG_DEPRECATIONS_CHANNEL=null
8    LOG_LEVEL=debug
9    DB_CONNECTION=mysql
10   DB_HOST=127.0.0.1
11   DB_PORT=3306
12   DB_DATABASE=db_jm
13   DB_USERNAME=root
14   DB_PASSWORD=
```

Kode sumber 5. 1 Konfigurasi environment laravel

Pada kode sumber 5.1 menampilkan bahwa *environment* atau lingkungan pengembangan aplikasi berdasar pada baris kode 2, aplikasi berjalan secara *local* yang berarti aplikasi berjalan pada lingkungan pengembangan (*development*). Pada baris kode 3, menunjukkan kunci rahasia yang digunakan untuk enkripsi data dalam aplikasi. Pada baris kode 11, menunjukkan driver database yang digunakan. Pada contoh ini, digunakan driver MySQL. Baris kode 12, menyimpan alamat host database, yaitu "127.0.0.1" (localhost). Baris kode 13 adalah port yang digunakan untuk koneksi ke database MySQL, yaitu port 3306. Baris kode 14 hingga 16 agalaj nama database, *username* untuk mengakses data base dan password untuk mengakses database.

5.1.2. Konfigurasi Middleware

Konfigurasi *middleware* dilakukan bertujuan untuk menyaring pengguna yang akan masuk, serta diarahkan pada laman sesuai dengan tugas masing-masing peran. Selain dari pada itu middleware juga dapat berfungsi sebagai mendeteksi pengguna masih memiliki sesi akses pada laman web. Konfigurasi tersebut ditampilkan pada pseudo kode 5.1, pseudo kode 5.2, pseudo kode 5.3, dan sumber kode 5.4:

```
1    procedure handle(request, next, role)
2        if Auth.check() and Auth.user.role = role then
3            return next(request)
4        else
5            switch Auth.user.role
6                case Config.get('constants.role.superadmin'):
7                    return redirect().route('superadmin.beranda')
8                case Config.get('constants.role.admin'):
9                    return redirect().route('admin.beranda')
10               case Config.get('constants.role.ki'):
11                   return redirect().route('ki.beranda')
```

```
            case Config.get('constants.role.wakil'):
                return redirect().route('wakil.beranda')
            case Config.get('constants.role.ku'):
                return redirect().route('ku.beranda')
            case Config.get('constants.role.mt'):
                return redirect().route('mt.beranda')
            case Config.get('constants.role.pnb'):
                return redirect().route('pnb.beranda')
            case Config.get('constants.role.pengabsen'):
                return redirect().route('pengabsen.beranda')
        end switch
    end if
    return redirect().route('login').with('failed', 'Access
denied.')
end procedure
```

Kode pseudo 5. 1 Konfigurasi middleware pembagian peran

Dalam kode pseudo 5.1 menampilkan konfigurasi dari *middleware* untuk membagi peran ketika pengguna masuk sesuai dengan peran yang telah terdaftar dalam database.

Dalam laravel, setiap pengguna memiliki waktu sesi yang terhitung dari terakhir pengguna aktif dalam aplikasi web. Ketika durasi mengakses dari pengguna belum habis, maka pengguna diperbolehkan untuk mengakses halaman permintaan pengguna. Namun, jika sesi mengakses dari pengguna telah habis, pengguna akan diarahkan pada halaman login yang ditunjukkan pada kode pseudo 5.2.

```
procedure handle(request, next)
    if Auth.check() then
        return next(request)
    end if

    return redirect().route('login').with('failed', 'Sesi anda habis')
end procedure
```

Kode pseudo 5. 2 Konfigurasi middleware jika pengguna masih memiliki sesi

Kode pseudo 5.2 menunjukkan bahwa sistem akan melakukan validasi apakah pengguna masih meiliki sesi atau tidak, jika pengguna masih memiliki sesi, pengguna akan diarahkan pada halam yang di-*request*. Namun, jika sesi pengguna telah habis, pengguna akan diarahkan pada halaman login untuk mengakses aplikasi kembali. Contohnya seperti ketika pengguna mengkases halaman beranda tanpa melewati halaman login.

Kemudian pada kasus jika pengguna sudah memiliki sesi yang habis ditunjukkan pada kode pseudo 5.3.

```
1    procedure handle(request, next)
2        if not Auth.check() then
3            return next(request)
4        else
5            switch Auth.user.role
6                case 'superadmin':
7                    return redirect().route('superadmin.beranda')
8                case 'admin':
9                    return redirect().route('admin.beranda')
10               case 'ki':
11                   return redirect().route('ki.beranda')
12               case 'wakil':
13                   return redirect().route('wakil.beranda')
14               case 'ku':
15                   return redirect().route('ku.beranda')
16               case 'mt':
17                   return redirect().route('mt.beranda')
18               case 'pnb':
19                   return redirect().route('pnb.beranda')
20               case 'pengabsen':
21                   return redirect().route('pengabsen.beranda')
22           end switch
23       end if
24   end procedure
```

Kode pseudo 5. 3 Konfigurasi middleware jika pengguna sesi pengguna telah habis

Kode pseudo 5.3 menampilkan bahwa jika pengguna sudah habis sesinya, pengguna akan diarahkan pada *request* pengguna selanjutnya, namun jika pengguna masih memiliki sesi, pengguna akan diarahkan pada halaman beranda. Contoh kasusnya ketika pengguna mengakses halaman login.

Konfigurasi-konfigurasi *middleware* tersebut akan didaftarkan kedalam file konfigurasi yang bernama Kernel.php yang ditunjukkan pada kode sumber 5.4. file konfigurasi Kernel.php merupakan pusat kontrol bagi seluruh *middleware* dalam laravel.

```
1  'id_role' => \App\Http\Middleware\CheckAccess::class,
2
3  'is_logged' => \App\Http\Middleware\IsLogged::class,
4  'isnot_logged' => \App\Http\Middleware\IsNotLogged::class,
```

Kode pseudo 5. 4 Konfigurasi kernel middleware

Untuk mengimplementasikan penggunaan *middleware*, *middleware* harus di terapkan pada bagian *routing*, yangmana bagian *routing* merupakan gerbang untuk mengarahkan permintaan pengguna sesuai dengan permintaan. Kode sumber 5.2 menampilkan pengaplikasian middleware pada *routing*.

```
1  // middleware superadmin
2  Route::middleware(['is_logged','id_role:superadmin'])-
3  >group(function () {});
```

Kode sumber 5. 2 Contoh penerapan middleware pada routing

Kode sumber 5.2 menampilkan bahwa untuk mengakses permintaan pengguna, pengguna haruslah lolos validasi dari middleware “is_logged” dan memiliki “id_role” berupa “superadmin”.

5.2. Implementasi Templating Laravel

Implementasi templating ini memisahkan pembagian logika dan user interface pada laravel. Struktur templating bertujuan supaya tidak adanya mencegah developer melakukan pengulangan pengetikan sumber kode guna meningkatkan efisien aplikasi serta project yang ada. Selain daripada itu, templating juga sering disebut kerangka dasar dalam pembuatan aplikasi web laravel, karena hal templating menjadi hal dasar yang dibentuk terlebih dahulu dalam pembangunan aplikasi laravel.

Laravel memberikan kemudahan bagi pengembang aplikasi dalam pembentukan template, yaitu extends, include, serta yield. Extends memberikan struktur data dan konten file lain yang ditentukan. Include membagi kode kedalam file terpisah untuk menghindari duplikasi. Yield memberikan atau menyediakan ruang yang dapat diisi dengan konten khusus dari file yang menggunakan tempalte ini.

Adapun implementasi templating laravel sebagai berikut:

5.2.1. Templating layouts

Pada implementasi templating partials berguna menyimpan komponen-komponen dasar tampilan untuk halaman website. Template layouts sendiri merupakan template yang statis, atau bisa dibilang template untuk konten yang kerangka dasar yang terdiri dari sumber kode css, meta, head pada sumber kode 5.3, script pada pseudo kode 5.5, alert pada pseudo kode 5.6, dan main pada pseudo kode 5.7.

```
1   <title>
2
3       @if (isset($title))
4           {{ $title }}
5
6       @else
7           Sistem Informasi Jamaah Masjid Keputih
```

```
8       @endif
9
10  </title>
11
12  @extends('layouts.meta')
13
14  @extends('layouts.css')
```

Kode sumber 5. 3 Sumber kode layouts head.blade.php

Template sumber kode head menunjukkan bahwa sumber kode meta dan css dibuat secara terpisah, hal ini bertujuan mempermudah dalam pengelolaan sumber kode css dan meta jika ada perubahan serta pengelolaan. Selain dari itu, pada sumber kode tersebut menunjukkan bahwa jika variabel $title tidak diberikan pada laman tersebut, laman tersebut akan memberikan nilai default "Sistem Informasi Jamaah Baitul Haq Keputih".

```
1   LoadScript(asset('/js/codebase.core.min.js'));
2
3   LoadScript('/js/codebase.app.min.js');
4   LoadScript('https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/
    js/bootstrap.bundle.min.js');
5
6   LoadScript(asset('/js/plugins/datatables/jquery.dataTables.min
    .js'));
7   LoadScript(asset('/js/plugins/datatables/dataTables.bootstrap4
    .min.js'));
8
9   LoadScript(asset('/js/codebase.app.min.js'));
10
11  Function updateProfile(event)
12      Variable pass1 = GetElementById("side-overlay-profile-
    password");
13      Variable pass2 = GetElementById("side-overlay-profile-
    password-confirm");
14
15      If pass1 != pass2 Then
```

```
16          Alert("Passwords did not match");
17
18      SubmitForm("updateProfile");
19  End Function
```

Kode pseudo 5. 5 Pseudo kode layouts script.blade.php

Template sumber kode script bertujuan untuk memasukkan fungsi-sungsi yang akan digunakan pada seluruh laman aplikasi laravel. Baik fungsi yang di-*import* dari *template*, maupun fungsi dasar non-*template* seperti fungsi updateProfile, fungsi tersebut diletakkan pada layouts karena fungsi tersebut akan digunakan di semua laman aplikasi.

```
1   If Session::has('success') Then
2       Alert success message
3           Strong "Success!"
4           Display Session::get('success')
5           Close button with data-dismiss="alert" aria-label="Close"
6   End If
7   If Session::has('failed') Then
8       Alert warning message
9           h4 "Failed!"
10          Display Session::get('failed')
11          Close button with data-dismiss="alert" aria-label="Close"
12  End If
13  If $errors->any() Then
14      Alert danger message
15          For Each error In $errors->all()
16              Strong Display error
17          End For
18          Close button with data-dismiss="alert" aria-label="Close"
19  End If
```

Kode pseudo 5. 6 Pseudo kode layouts alert.blade.php

Template pseudo kode alert betujuan untuk menampilkan peringatan dari aplikasi yang bertujuan memberitahu pada

pengguna bahwa permintaan yang dilakukan berhasil, gagal, ataupiun terjadi galat.

```
1    Declare doctype html
2    Declare html language as "en"
3
4    Declare head section
5        Include 'layouts.head'
6        Yield 'additional_head'
7    End head section
8
9    Declare body section
10       Declare div with id "page-container" and class "sidebar-o
     enable-page-overlay page-header-inverse sidebar-inverse main-
     content-boxed"
11           Include 'partials.sideOverlay'
12           Include 'partials.sidebar'
13           Include 'partials.navbar'
14           Yield 'content'
15       End div
16
17       Include 'layouts.script'
18       Yield 'additional_script'
19   End body section
20
21   End html
```

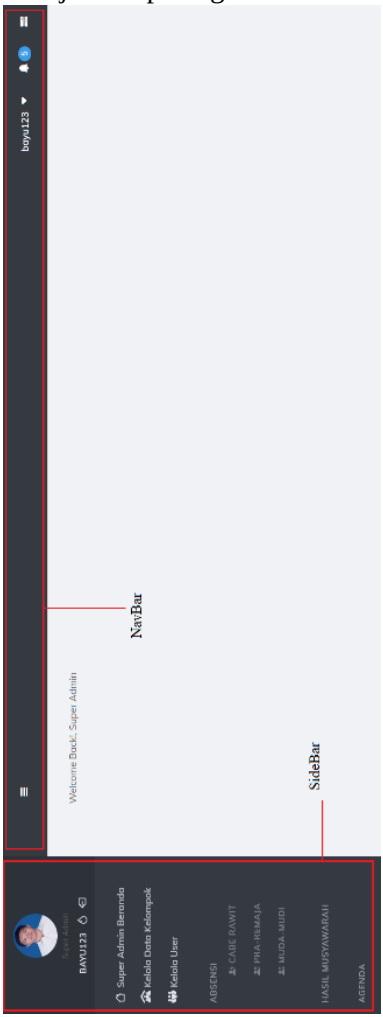
Kode pseudo 5. 7 Kode pseudo layouts main.blade.php

Template pseudo kode main adalah template akhir dari template layout, yang mana template ini merangkum seluruh template yang akan digunakan pada laman aplikasi.

5.2.2. Templating Partials

Pada implementasi templating partials berguna menyimpan komponen-komponen tampilan yang digunakan secara berulang dalam beberapa halaman. Template terdiri dari beberapa bagian, seperti template sidebar, navbar, serta sideOverlay yang

ditunjukkan pada halaman web gambar 5.1. Sidebar merupakan template dinamis yang menyediakan menu pilihan sesuai peran yang login. Navbar memberikan menu tentang informasi tambahan bagi user. SideOverlay berguna sebagai bagian edit profile bagi pengguna yang ditunjukkan pada gambar 5.2.



Gambar 5. 1 Halaman web

Pada gambar 5.1 menunjukkan sidebar yang terletak bagian kiri laman web. Sidebar menampilkan *menu* bagi pengguna, memu merupakan kegiatan atau hal-hal yang bisa dilakukan oleh pengguna. Navbar terletak pada bagian atas, navbar menyediakan *menu* bagi pengguna, menu tersebut berbeda dengan *menu* **y**ang disediakan oleh sidebar. *Menu* yang disediakan oleh navbar merupakan menu yang berupa notifikasi, edit profile, serta *minimize* sidebar.

Gambar 5. 2 SideOverlay User

Pada gambar 5.2 menunjukkan sideOverlay yang berguna mengupdate profile dari pengguna saat ini, mengupdate terdiri dari username, foto profil, serta password.

5.2.3. Templating Route

Pada implementasi templating route adalah pembuatan url request untuk aplikasi yang dibangun. *Route* menjadi penghubung antara *controller* dan view. Template ditampilkan pada kode sumber 5.4 berikut:

```
1  Route::get('/',[AuthController::class, 'index'])->name('login')-
   >middleware('isnot_logged');
2  Route::post('/login',[AuthController::class, 'auth_login'])-
   >name('login_process');
3  Route::post('/logout', [AuthController::class, 'logout'])-
   >name('logout');
```

Kode sumber 5. 4 Contoh sumber kode routing

Pada kode sumber 5.3 pada baris kode 1 menampilkan bahwa route akan menghubungkan antara AuthController dan view "/" atau sering disebut halaman utama yang berupa halaman login. Dalam baris kode 1 menampilkan bahwa untuk mengakses halaman login pengguna harus memenuhi middleware "isnot_logged".

Kode sumber lengkap web.php dapat dilihat pada Lampiran 9.

5.2.4. Templating Controller

Pada implementasi templating controller adalah penerapan logika aplikasi yang dibangun. *Controller* pada perancangan sistem informas jamaah masjid Baitul Haq terdiri dari beberapa *controller* yaitu UserController, AuthController dan PengurusController.

Masing-masing *controller* memiliki tugas masing-masing, dalam AuthController bertugas sebagai pemrosesan data *login* dan *logout*. Dalam contoh login, ketika ada user masuk kedalam aplikasi, controller akan mengarahkan pengguna pada halaman

beranda sesuai dengan peran dari pengguna. Namun jika peran pengguna tidak memiliki halaman beranda, maka sistem akan mengeluarkan peringatan berupa “Akun tidak memiliki akses diweb!”. Namun, jika pengguna yang berusaha masuk tidak tervalidasi oleh sistem, maka sistem akan memberikan peringatan berupa “Akun tidak valid!”.

Contoh sumber kode *controller* ditampilkan pada sumber kode 5.5 berikut:

```
1           try {
2               if(Auth::attempt($credentials)){
3                   $request->session()->regenerate();
4
5                   switch (Auth::user()->role) {
6
7                       case \Config::get('constants.role.superadmin'):
                            return redirect()-
    >route('superadmin.beranda');
8                           break;
9                       case \Config::get('constants.role.admin'):
10                          return redirect()->route('admin.beranda');
11                          break;
12                      case \Config::get('constants.role.ki'):
13                          return redirect()->route('ki.beranda');
14                          break;
15
16                      case \Config::get('constants.role.wk'):
17                          return redirect()->route('wakil.beranda');
18                          break;
19
20                      case \Config::get('constants.role.ku'):
21                          return redirect()->route('ku.beranda');
22                          break;
23
25                      case \Config::get('constants.role.mt'):
26                          return redirect()->route('mt.beranda');
27                          break;
```

```
28
29                      case \Config::get('constants.role.pnb'):
30                          return redirect()->route('pnb.beranda');
31                          break;
32
33                      default:
34                          Auth::logout();
35                          return redirect()->back()-
36  >with(["failed"=>"Akun tidak memiliki akses di web!"])-
    >withInput();
37                          break;
38                  }
39
40              }
41              else {
42                  throw new \Exception("Akun tidak Valid", 503);
43              }
```

Kode sumber 5. 5 Contoh sumber kode controller

Kode sumber lengkap AuthController.php dapat dilihat pada Lampiran 10.

5.3. Implementasi Migration Laravel

Implementasi migration ini merupakan implementasi dari design database pada gambar 4.1. Adapun implementasi migration laravel sebagai berikut:

5.3.1. Migration Default Laravel

Laravel memiliki default migration untuk para pengguna Laravel, Ketika inisiasikan Laravel maka Laravel secara default mengenerate migration-migration berikut.

Laravel memiliki default migration untuk para pengguna Laravel, Ketika inisiasikan Laravel maka Laravel secara default mengenerate migration-migration berikut. Kode sumber ditunjukkan pada Pseudo code 5.8.

```
1    Create Users Table Migration
2
3    Procedure up():
4        Call Schema::create('users', function (Blueprint $table):
5            Call $table->id() to create 'id' column
6            Call $table->string('name') to create 'name' column
7            Call $table->string('email')->unique() to create 'email'
     column with unique constraint
8            Call $table->timestamp('email_verified_at')->nullable() to
     create 'email_verified_at' column that allows null values
9            Call $table->string('password') to create 'password' column
10           Call $table->rememberToken() to create 'remember_token'
     column
11           Call $table->timestamps() to create 'created_at' and
     'updated_at' columns
12       End Schema::create('users')
13
14   Procedure down():
15       Call Schema::dropIfExists('users')
16   End Procedure
```

Kode pseudo 5. 8 Migrasi user

Kode sumber lengkap kode Default Laravel Migration dapat dilihat pada Lampiran 11-14.

5.3.2. Migration Database

Berdasarkan design database yang telah ditentukan Bersama. Berikut adalah migration-migration yang dibutuhkan untuk database yang kami butuhkan dan telah ditentukan Bersama. Pseudo code ditunjukkan pada Pseudo code 5.9.

```
1    Create Jamaah Table Migration
2
3    Procedure up():
```

```
4       Call Schema::create('jamaah', function (Blueprint $table):
5           Call $table->id() to create 'id' column
6           Call $table->string('nama', 255) to create 'nama'
    column of string type with maximum length 255 characters
7           Call $table->year('tanggal_lahir') to create
    'tanggal_lahir' column of year type
8           Call $table->text('alamat') to create 'alamat' column
    of text type
9           Call $table->string('pekerjaan', 255) to create
    'pekerjaan' column of string type with maximum length 255
    characters
10          Call $table->string('foto', 255) to create 'foto'
    column of string type with maximum length 255 characters
11          Call $table->string('jenis_kelamin', 1) to create
    'jenis_kelamin' column of string type with length 1
12          Call $table->string('no_hp', 20) to create 'no_hp'
    column of string type with maximum length 20 characters
13          Call $table->string('dapukan', 255)->nullable() to
    create 'dapukan' column of string type with maximum length 255
    characters, allowing null values
14          Call $table->boolean('is_pendatang') to create
    'is_pendatang' column of boolean type
15          Call $table->timestamps() to create 'created_at' and
    'updated_at' columns
16          Call $table->softDeletes($column = 'deleted_at',
    $precision = 0) to create 'deleted_at' column for soft deletes
    with precision 0
17      End Schema::create('jamaah')
18  Procedure down():
19      Call Schema::dropIfExists('jamaah')
20  End Procedure
```

Kode pseudo 5. 9 Laravel Database Migration

Kode sumber lengkap kode Default Laravel Migration dapat dilihat pada Lampiran 15-21.

5.4. Implementasi Model Laravel

Implementasi model ini merupakan implementasi dari design database pada Gambar 4.1. Adapun implementasi model laravel menyesuaikan dengan migration dikarenakan model sebagai penghubung antara database dengan web (penghubung antara pengembang *front-end* dan *back-end*. Implementasi model Laravel sebagai berikut:

5.4.1. Model Default Laravel

Laravel memiliki default model untuk para pengguna Laravel, Ketika meng-inisiasikan Laravel maka Laravel secara default mengenerate berikut, namun dikarenakan kami menggunakan table users maka terdapat sejumlah perubahan dalam kode default Laravel untuk model users. Pseudo code ditunjukkan pada Pseudo Code 5.10.

```
1    User Model
2
3    Class User extends Authenticatable:
4        Use HasApiTokens, HasFactory, Notifiable, SoftDeletes
5        Define protected $table as 'users'
6        Define protected $dates as ['deleted_at']
7        Define protected $primaryKey as 'id'
8        Define protected $fillable as an array with the following
9    fields:
10           - 'username'
11           - 'password'
12           - 'role'
13           - 'last_login'
14           - 'id_jamaah'
15           - 'nama'
16       Define protected $guarded as an array with 'id' field
17       Define protected $hidden as an array with the following fields:
18           - 'password'
19           - 'remember_token'
20   End Class
```

```
21
```

Kode pseudo 5. 10 Model Default Migration

Kode sumber lengkap kode Default Laravel Migration dapat dilihat pada Lampiran 22.

5.4.2. Model Database Laravel

Berdasarkan design database yang telah ditentukan, kami membutuhkan beberapa model untuk database kami. Berikut model-model database yang kami butuhkan. Pseudo code ditunjukkan pada Pseudo code 5.11.

```
1    Class Jamaah extends Model
2            Use HasFactory, SoftDeletes
3
4            Define protected $table as 'jamaah'
5            Define protected $dates as ['deleted_at']
6            Define protected $primaryKey as 'id'
7            Define protected $fillable as an array with the following
     fields:
8                - 'nama'
9                - 'tanggal_lahir'
10               - 'alamat'
11               - 'pekerjaan'
12               - 'foto'
13               - 'jenis_kelamin'
14               - 'no_hp'
15               - 'dapukan'
16               - 'is_pendatang'
17               - 'id_bapak'
18               - 'id_ibu'
19               - 'id_suami'
20               - 'id_istri'
21       End Class
22
23   End Namespace
```

Kode pseudo 5. 11 Laravel Database Model

Kode sumber lengkap kode Default Laravel Migration dapat dilihat pada Lampiran 23 dan 24.

5.5. Implementasi CRUD Laravel

CRUD (Create Read Update Delete) merupakan fungsi dalam mysql untuk menyimpan, menampilkan, menambahkan, menghapus data dari database yang telah disediakan.

5.5.1. Create

Fungsi create dalam Laravel bertujuan untuk menambahkan dan menyimpan data baru kedalam database.

a. Create Jamaah

Create jamaah berguna meenambahkan data jamaah pada database, data jamaah akan digunakan sebagai data awal yang dibutuhkan sistem setelah data super admin karena dalam pembuatan user lainnya akan membutuhkan id_jamaah seperti yang sudah ditunjukkan pada gambar 4.1. Pada pembuatan jamaah dibutuhkan beberapa data yang dimasukkan kedalam database, diantaranya nama, tanggal lahir, alamat, pekerjaan, jenis kelamin, nomor hp, is pendatang (jamaah masjid tersebut merupakan personal yang bukan berasal dari wilayah keputih) serta foto. Alur kode ditunjukkan pada Pseudo code 5.12

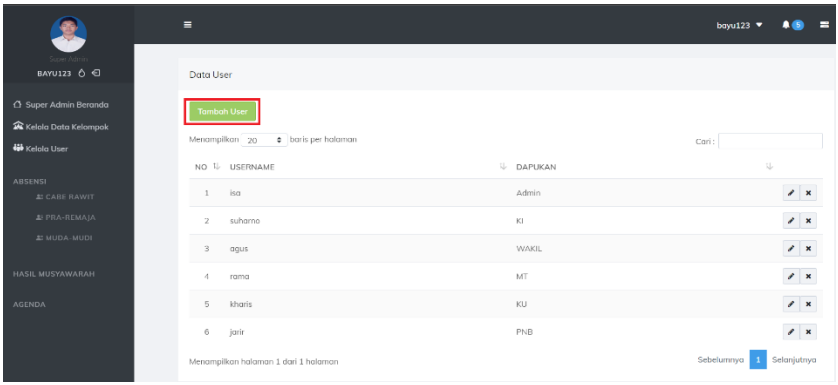
```
1    PengurusController – Create Jamaah Function
2    Class PengurusController extends Controller
3
4    Define public function store(Request $request)
5    Try
6    Validate the $request using the following rules:
7    - 'nama'          => 'required'
8    - 'tanggal_lahir' => 'required'
9    - 'alamat'        => 'required'
10   - 'pekerjaan'     => 'required'
11   - 'jenis_kelamin' => 'required'
```

```
12   - 'no_hp'         => 'required'
13   - 'is_pendatang'  => 'required'
14   - 'foto'          => 'required|jpg, jpeg, gif, png|max:2048'
15
16   Create a new Jamaah object with the following data:
17   - 'nama'          => $request->nama
18   - 'tanggal_lahir' => $request->tanggal_lahir
19   - 'alamat'        => $request->alamat
20   - 'pekerjaan'     => $request->pekerjaan
21   - 'foto'          => $request->foto
22   - 'jenis_kelamin' => $request->jenis_kelamin
23   - 'no_hp'         => $request->no_hp
24   - 'dapukan'       => $request->dapukan
25   - 'is_pendatang'  => $request->is_pendatang
26   - 'id_bapak'      => $request->id_bapak
27   - 'id_ibu'        => $request->id_ibu
28   - 'id_suami'      => $request->id_suami
29   - 'id_istri'      => $request->id_istri
30
31   Redirect to the specified route with a success message
32   Catch \Exception $e
33   Redirect back with a failed message and the error message from the
34   exception
35
36   End Function
37
38   End Class
```

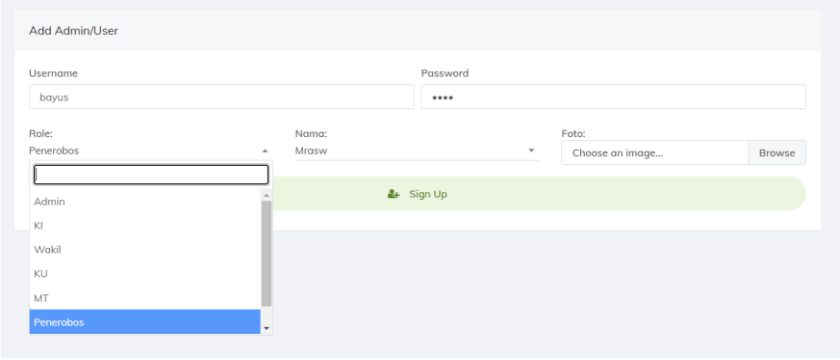
Kode pseudo 5. 12 Kode Laravel Create Jamaah

b. Create User

Create user memungkinkan super admin untuk melakukan penambahan pada pengguna dalam aplikasi sistem yang dibangun. Pembuatan pengguna jika termasuk pembuatan admin, maka hanya super admin yang bisa melakukannya karena super admin memiliki otoritas tertinggi pada aplikasi sistem. Halaman pembuatan user ditampilan pada gambar 5.3 dan gambar 5.4



Gambar 5. 3 Melakukan klik pada tombol tambah user



Gambar 5. 4 Melakukan pengisian username, password, role, nama jamaah, dan foto

Pada pembuatan user dibutuhkan beberapa data yang dimasukkan kedalam database, diantaranya username, foto, role, nama, serta id_jamaah. Role merupakan peran yang akan diterima oleh user berupa tugas dari masing-masing peran yang diterima. Id_jamaah digunakan karena pengguna atau pengurus masjid haruslah jamaah dari masjid Baitul Haq keputih. Dengan catatam ketika pembuatan user, username tidak boleh ada yang sama.

Pada pembuatan user dengan syarat usename tidak boleh sama dengan melakukan pengecekan pada database user jika username sudah terambil yang ditunjukkan pada kode pseudo 5.13.

```
1       if existsInDatabase(validatedData['username']):
2           throwException("nama terduplikasi", 503)
```

Kode pseudo 5. 13 Validasi username ketika menambah user

Pada kode pseudo 5.13 pada baris kode 1 menunampilkan bahwa username akan dilakukan cek apakah username sudah digunakan oleh pengguna lainnya atau belum, jika sudah, maka akan dilakukan *throwException* "nama terduplikasi". Setelah lolos dari validasi username, akan dilanjutkan dengan melakukan enkripsi password yang ditunjukkan pada kode pseudo 5.14.

```
1   validatedData['password']=encryptPassword(validatedData['password'])
```

Kode pseudo 5. 14 Kode laravel create user

Setelah password dienkripsi, username, password dan foto akan disimpan dalam database user yang ditunjukkan pada kode pseudo 5.15.

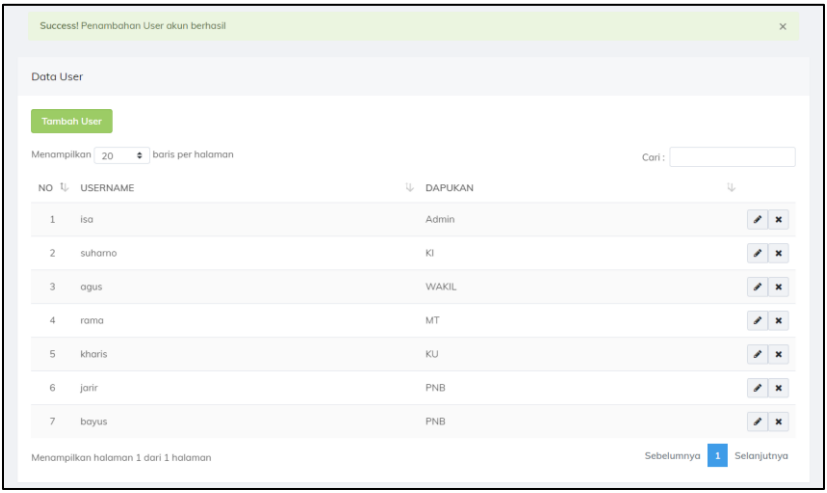
```
1   if hasFile('foto'):
2           validatedData['foto'] = saveFile(request['foto'], 'user-
    foto')
3
4       user = createUser(validatedData)  // Membuat objek user baru
    dengan data yang divalidasi
5
6       if saveUser(user):
7           flashMessage('success', 'Penambahan User akun berhasil')
8           redirectToRoute('authorshared.data_user')
9       else:
10          throwException('Gagal menyimpan user')
```

```
11
12   exceptionHandler(\Exception $e):
13       redirectBackWithErrors(['error' => $e->getMessage()])
```

Kode pseudo 5. 15 Kode laravel create user

Pada kode pseudo 5.15 pada baris kode 2 melakukan pemyimpanan file foto pada *directory* yang bernama "user-foto" dan pada baris kode 4 menampilkan bahwa username, password yang sudah terenskripsi serta nama foto disimpan dalam database.

Jika data berhasil disimpan, maka data diarahkan pada route "authorshared.data_user" yang merupakan menampilkan seluruh list pengguna pada baris kode 8. Namun jika gagal, sistem akan menampilkan peringatan berupa "Gagal menyimpan user" yang ditunjukkan pada baris kode 10. Gambar 5.5 menampilkan user berhasil dibuat.



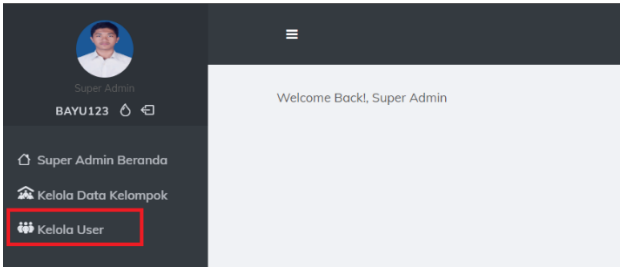
Gambar 5. 5 User berhasil dibuat

5.5.2 Read

Read merupakan fungsi laravel yang digunakan untuk membaca data dari database.

a. Read User

Read user merupakan fungsi yang digunakan untuk membaca seluruh data user dalam aplikasi sistem yang dibangun. Fungsi membaca user hanya bisa dilakukan oleh super admin, karena superadmin merupakan pemegang otoritas tertinggi. Halaman *read* user ditampilkan pada gambar 5.6 .



Gambar 5. 6 Melakukan klik pada tombol menu kelola user

Pada aksi gambar 5.6 akan terjadi beberapa proses oleh sistem. Pertama, sistem akan mendeteksi aksi yang memicu route "authorshared.data_user" berjalan kose sumber ditampilkan pada kode sumber 5.6.

```
1    <a href="{{ route('authorshared.data_user') }}"><i class="fa fa-
     people-group"></i><span class="sidebar-mini-hide">Kelola
     User</span></a>
```

Kode sumber 5. 6 pemicu route menu kelola user

Proses permintaan pengguna akan menuju pada *routing* laravel yang terdapat pada web.php yang ditunjukkan pada kode sumber 5.7.

```
1    Route::controller(UserController::class)->prefix('shared-author')-
     >name('authorshared.')->group(function(){
2        Route::middleware(['author'])->group(function () {
3            Route::get('/show_user','show_user')->name('data_user');
4        });
5    });
```

Kode sumber 5. 7 route untuk authorshared.data_user

Pada kode sumber 5.6 menampilkan bahwa untuk "data_user" terdapat dalam UserController yang ditunjukkan pada baris kode 1 serta dalam fungsi "show_user" dan untuk mengakses data_user harus lolos validasi dari middleware author.

Dalam fungsi "show_user" pada UserController, sistem akan menampilkan data selain dari super admin jika yang mengakses adalah super admin. Tujuannya adalah supaya meminimalisir terjadinya kesalahan manusia ketika super admin tidak sengaja melakukan edit atau penghapusan pada datanya sendiri. Kode pseudo dari pencegahan tersebut ditampilkan pada kode pseudo 5.16.

```
1    if Auth::user()->role == 'superadmin':
2            user = User::where('role', '!=', 'superadmin')->get()
```

Kode pseudo 5. 16 Mengambil data selain superadmin

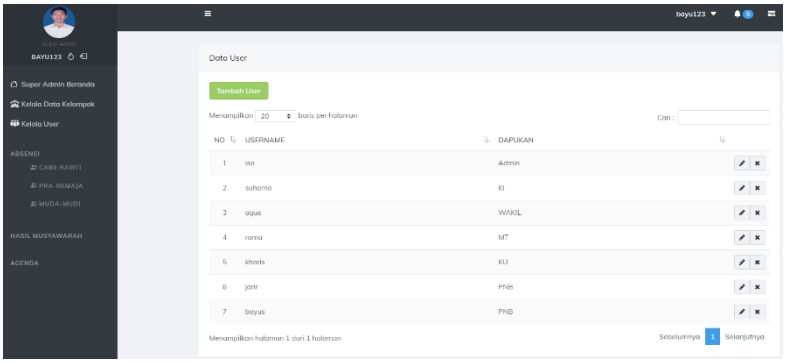
Pada kode pseudo 5.14 menampilkan jika yang mengakses adalah super admin, maka user yang akan ditampilkan selain super admin.

Hal demikian juga diterapkan pada admin, ketika admin yang mengakses, admin tidak bisa melihat data user lain selain dari pengurus masjid. Kode pseudo dari pencegahan tersebut ditampilkan pada kode pseudo 5.17.

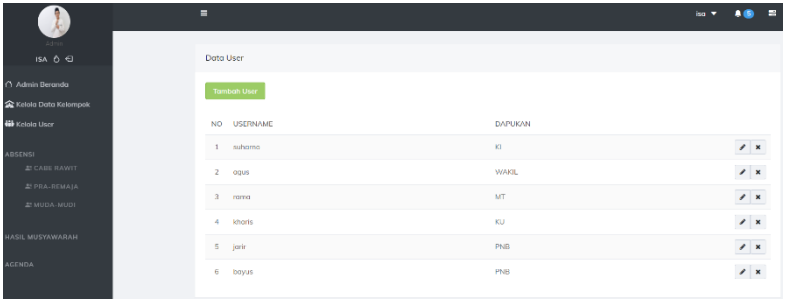
```
1    else:
2            user = User::where('role', '!=', 'superadmin')-
     >where('role', '!=', 'admin')->get()
```

Kode pseudo 5. 17 Mengambil data selain superadmin dan admin

Pada kode pseudo 5.15 menampilkan jika yang mengakses adalah admin, maka user yang akan ditampilkan hanyalah pengurus masjid. Hasil membaca user oleh super admin ditunjukkan pada gambar 5.7 serta hasil membaca user oleh admin ditunjukkan pada gambar 5.8.



Gambar 5. 7 Hasil super admin membaca user



Gambar 5. 8 Hasil admin membaca user

b. Read Jamaah

Read jamaah merupakan fungsi yang digunakan untuk membaca seluruh data jamaah dalam aplikasi sistem yang dibangun. Pseudo code dari fungsi create user ditampilkan pada pseudo kode 5.18.

.

```
1    jamaah_show
2
3    Function jamaah_show():
4        get_role_text = \Config::get('constants')
5        apaitu = Auth::user()->role
6        return view('shared.kelola_jamaah', [
7            'title' => "Data Jama'ah" . "-" . \Config::get('constants.role_te
8            'foto' => Auth::user()->foto,
9            'id_role' => \Config::get('constants.role_text.'.$apaitu),
10           'nama' => Auth::user()->username,
11           'jamaah' => Jamaah::all(),
12       ])
13   End Function
```

Kode pseudo 5. 18 Kode Laravel Read Jamaah

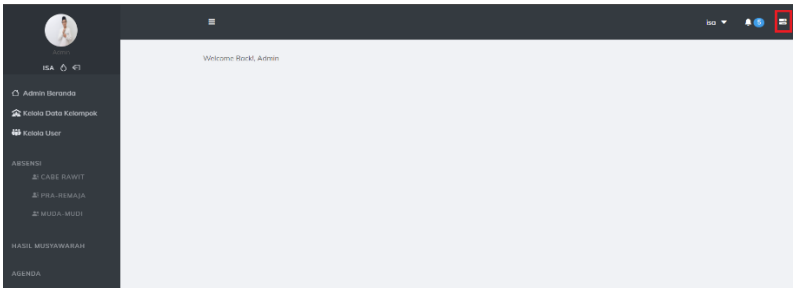
5.5.3. Update

Update dalam laravel memungkinkan pengguna melakukan perubahan data dalam database.

a. Update Profil

Update profil merupakan perubahan data yang bisa dilakukan oleh seluruh user pada prosil masing-masing user. Data yang dirubah dalam update profil berupa username, foto profil, dan password.

Merubah profil dilakukan dengan melakukan aksi klik pada tombol untuk menampilkan SideOverlay yang terdapat pada pojok kanan atas navbar yang ditampilkan pada gambar 5.9.

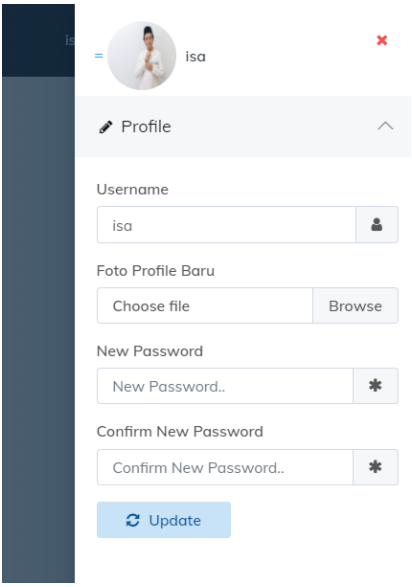


Gambar 5. 9 Melakukan klik pada tombol menu SideOverlay

Ketika tombol sudah dipicu, maka menu dari SideOverlay akan muncul yang ditampilkan pada gambar 5.10.

Gambar 5. 10 SideOverlay muncul

Dalam SideOverlay, akan terdapat menu melakukan perubahan profil yang bisa untuk merubah username, foto profil, dan password. Ketika tombol “update” diklik yang ditunjukkan pada gambar 5.11 maka akan memicu sistem menjalankan permintaan berupa route pada web.php yang ditampilkan pada kode sumber 5.8.



Gambar 5. 11 Upate profil pada SideOverlay

```
1    <form id="updateProfile"
     action="{{ route('authorshared.update_profile') }}" method="post"
     enctype="multipart/form-data">
```

Kode sumber 5. 8 Pemicu perubahan profil

Proses permintaan pengguna akan menuju pada *routing* laravel yang terdapat pada web.php yang ditunjukkan pada kode sumber 5.9.

```
1    Route::controller(UserController::class)->prefix('shared-author')-
     >name('authorshared.')->group(function(){
2        Route::post('/update_profile', 'updateProfile')-
     >name('update_profile');
3    });
```

Kode sumber 5. 9 Route untuk authorshared.update_profile

Kode sumber 5.8 menunjukkan bahwa “update_profile” terdapat dalam UserController yang terdapat dalam fungsi updateProfile. Dalam merubah profil, username tidak boleh terduplikasi. Adapun untuk memvalidasi hal tersebut, dilakukan dengan melakukan pengecekan username baru sudah digunakan oleh pengguna lainnya atau belum. Hal tersebut ditampilkan pada kode pseudo 5.19.

```
1    exist = User::where('username', validatedData['side-overlay-
     username'])->exists()
2
3            if exist:
4                throwException("nama terduplikasi", 503)
5
```

Kode pseudo 5. 19 Kode Laravel Read Jamaah

Pada kode pseudo 5.16, menunjukkan jika username yang terbaru sama dengan username yang ada dalam database user, maka akan melakukan throwException “nama terduplikasi”. Jika username yang baru tidak sama dengan username yang ada dalam database, maka proses akan berlanjut pada memvalidasi data-data baru seperti username, password dan foto.

Dalam melakukan validasi username ditunjukkan pada kode pseudo 5.20

```
1    data->username = validatedData['side-overlay-username']
```

Kode pseudo 5. 20 Melakukan validasi username

Kode pseudo 5.20 menampilkan bahwa username terbaru dimasukkan ke dalam data final yang sebagai data akhir profil. Selanjutnya dalam validasi password ditunjukkan pada kode pseudo 5.21.

```
1    if validatedData['side-overlay-profile-password'] != null:
         data->password = encryptPassword(validatedData['side-
     overlay-profile-password'])
```

Kode pseudo 5. 21 Melakukan validasi password

Kode Pseudo 5.17 menampilkan validasi password, jika ada perubahan password yang dilakukan, maka password terbaru dimasukkan dalam data final. Terakhir adalah validasi foto yang ditunjukkan pada kode pseudo 5.22.

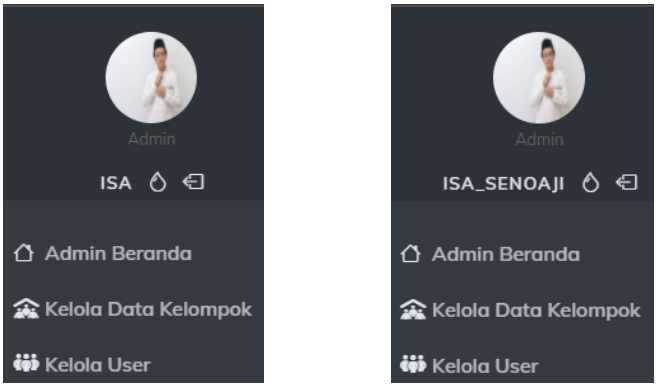
```
1    oldFoto = data->foto
2
3        if hasFile('side-overlay-profile-pict'):
4            validatedData['side-overlay-profile-pict'] =
     saveFile(request['side-overlay-profile-pict'], 'user-foto')
5            data->foto = validatedData['side-overlay-profile-pict']
6
7            if oldFoto != null and existsFile(oldFoto):
8                deleteFile(oldFoto)
```

Kode pseudo 5. 22 Melakukan validasi foto

Kode pseudo 5.22 baris kode 1 menampilkan pengambilan data foto lama yang disimpan dalam variabel oldFoto. Kemudian dilakukan pengecekan jika terdapat foto baru yang akan dirubah, jika iya maka proses akan dilanjutkan dengan menyimpan data foto baru kedalam directory yang bernama “user-foto” dan kemudian disimpan dalam data final.

Kemudian foto yang sebelumnya akan dihapus supaya tidak memenuhi dari memori penyimpanan sistem yang dibuat dengan melakukan pengecekan pada variabel oldFoto pada baris kode 7, apakah tidak kosong, jika tidak kosong, maka file foto yang sebelumnya akan dihapus.

Hasil kahir dari perubahan profil dapat dilihat pada gambar 5.12.

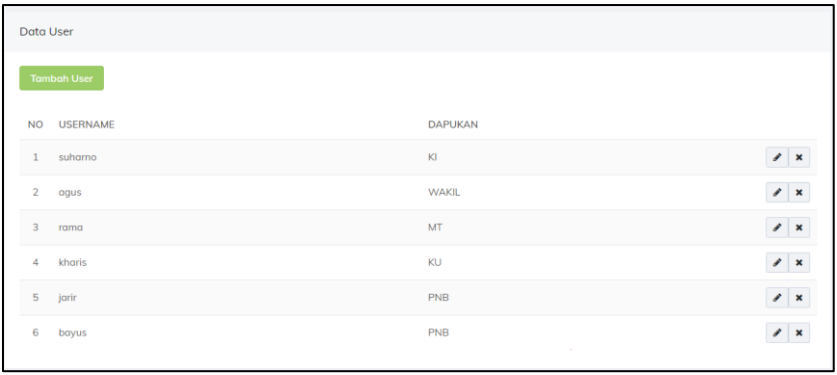


(a) (b)

Gambar 5. 12 Perbandingan (a) sebelum meperbarui profil dan (b) sesudah memperbarui profil

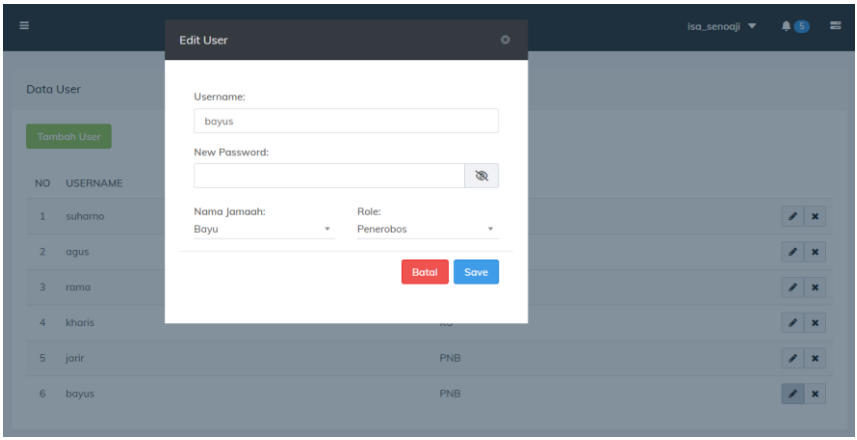
b. Update User

Update user merupakan perubahan data yang hanya dilakukan oleh super admin dan admin karena dalam perubahan data user admin dan super admin lah yang bertanggung jawab dalam data user yang ada. Pembaruan data user dilakukan dengan mengklik tombol yang berlambangkan pensil pada menu kelola user ditunjukkan pada gambar 5.13.



Gambar 5. 13 Data user

Ketika tombol *update* atau perbaruan data user yang dilambangkan dengan pensil ditekan, maka akan muncul modal dari bootstrap serta menjalanakan fungsi javascript "updateUser" fungsi tersebut berguna untuk mengambil data dari database yang akan dimasukkan dalam modal. Pada modal tersebut, akan menampilkan langsung username, nama jamaah, serta *role* dari user yang ditunjukkan pada gambar 5.14.



Gambar 5. 14 Modal edit user

Username, nama jamaah dan *role* didapatkan dengan menggunakan bantuan *routing* laravel 'authorshared.get_user' yang akan menjalankan fungsi get_user pada UserController. Kode sumber 5.10 menampilkan fungsi get_user.

```
try {
            $data = User::find($id);

            if ($data) {
                return response()->json($data);
            } else {
                return response()->json(['message' => 'Data not found'], 404);
            }

        } catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            return response()->json(['message' => 'Failed to find data'], 500);
        }
```

Kode sumber 5. 10 fungsi get_user dalam UserController

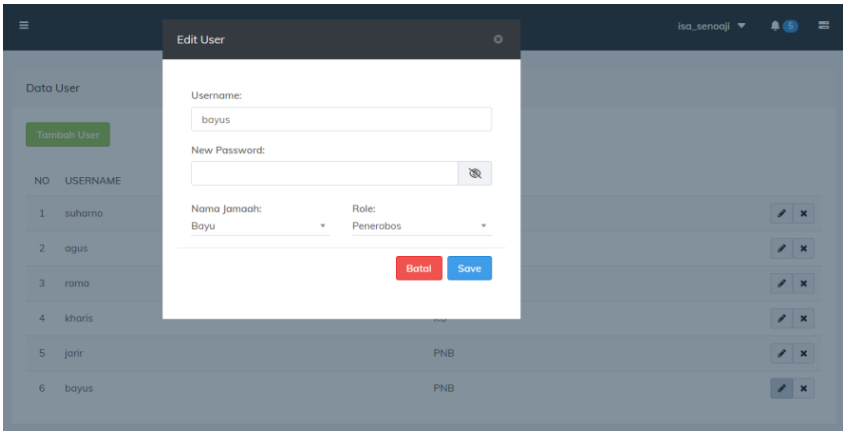
Dalam kode sumber 5.10 menunjukkan data akan dicari berdasarkan id dari user yang akan dilakukan pembaruan data, yang kemudian jika data ditemukan, data akan dikirim pada fungsi updateUser. Namun, sistem akan memberikan peringatan "Data not found" serta "Failed to find data".

Ketika data diterima dari UserController maka data akan dimasukkan kedalam modal sesuai dengan bagian masing-masing yang ditunjukkan pada sumber kode 5.11.

```
document.getElementById('updateUserName').value = data.username;
document.querySelector('select[name="id_jamaah"]').value = data.id_jamaah;
document.querySelector('select[name="role"]').value = data.role;
```

Kode sumber 5. 11 Pembagian data dalam fungsi updateUser

Kode sumber 5.11 menampilkan bahwa isi data dalam modal akan disesuaikan dengan isi dari data yang dikirimkan oleh database. Hasil penerimaah ditunjukkan pada gambar 5.15.



Gambar 5. 15 Modal perbaruan user

Gambar 5.15 menampilkan bahwa data username, nama jamaah dan *role* sudah terusu seusai dengan data yang diterima dari database, hal ini memudahkan pengguna dalam memperbarui data user. Kemudian data akan dilakukan pembaruan dan tombol “save” diklik dan menyimpan data kedalam database.

Proses penyimpanan dalam sistem berjalan sebagai berikut, ketika tombol”save” pada modal diklik, maka *routing* laravel “authorshared.update_user” akan terpicu dan melakukan update pada UserController dalam fungsi “update_user”.

Dalam update user memungkinkan untuk merubah data seperti username, role serta id_jamaah. Dalam username hanya boleh menggunakan huruf lowercase dan angka serta username harus unique yang tidak boleh terdapat username yang sama dalam satu sistem. Pengaturan username ditunjukkan pada kode sumber 5.12.

```
1    data = User::find(id)
2    if request['username'] == data->username:
3            validatedData = validateInput(request, [
4                'username' => 'required|regex:/(^([a-z0-9]+)(\d+)?$)/u',
5                'password' => 'max:255',
6                'role' => 'required',
7                'id_jamaah' => 'required',
8            ])
9        else:
10           validatedData = validateInput(request, [
                 'username' => 'required|unique:users|regex:/(^([a-z0-
11   9]+)(\d+)?$)/u',
12               'password' => 'max:255',
13               'role' => 'required',
14               'id_jamaah' => 'required',
15           ])
```

Kode sumber 5. 12Validasi username

Username dilakukan valdiasi sebanyak dua kali dikarenakan untuk memvalidasi username tersebut sudah digunakan oleh userlainnya atau tidak. Kemudian username juga dilakukan validasi *uppercase* dan *special character* yang ditunjukkan dengan menggunakan regex pada baris kode 4 dan 11. Regex "/(^([a-z0-9]+)(\d+)?$)/u" mencocokkan string yang dimulai dengan satu atau lebih karakter huruf kecil atau angka dan dapat diikuti oleh satu atau lebih angka.

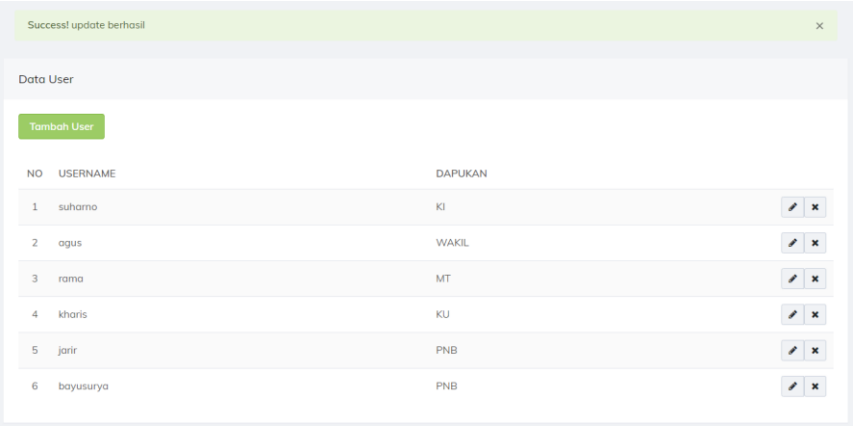
Selain dari itu, dilakukan juga validasi password yang maksimal sepanjang 255 karakter. Kemudian setelah data tervalidasi akan dilanjutkan pada proses yang ditunjukkan pada kode sumber 5.13.

```
1    if not data:
2                throwException("akun tidak ada", 400)
3
4            data->username = validatedData['username']
```

```
5
6           if validatedData['password'] is not null:
7               data->password =
8   encryptPassword(validatedData['password'])
9
10          data->role = validatedData['role']
11          data->id_jamaah = validatedData['id_jamaah']
12
13  if saveUser(data):
14              redirectBackWithSuccessMessage('update berhasil')
15          else:
16              throwException('Gagal menyimpan data')
17
18      exceptionHandler(\Exception $e):
19          redirectBackWithErrorMessage('update gagal' + ' ' + e-
    >getMessage())
```

Kode sumber 5. 13 penyimpanan data baru dalam variabel data final

Sumber kode 5.13, data akan dicek, jika data user pada baris kode 1 tidak ada, maka akan menampilkan peringatan akun tidak ada. Namun, jika data pengguna ada, maka proses akan dilanjutkan dengan mengenkripsi password, jika pengguna melakukan perubahan pada passwordnya, serta melakukan pembaruan pada *role* dan *id_jamaah*. Terakhir pada baris kode 13, data akan disimpan, namun jika gagal, sistem akan menampilkan peringatan "Gagal menyimpan data". Hasil dari update data dapat dilihat pada gambar 5.16.



Gambar 5. 16 Hasil data user setelah diperbarui

Gambar 5.16 menampilkan terdapat perubahan data dibandingkan dengan gambar 5.13. pada username pengguna keenam, berubah dari bayus menjadi bayusurya.

c. Update Jamaah

Update jamaah merupakan perubahan data yang dilakukan oleh user pada data jamaah, user tersebut termasuk dari super admin, admin, dan seluruh user yang terkait. Dalam update jamaah memungkinkan untuk merubah data seperti nama, tanggal lahir, alamat, pekerjaan, jenis kelamin, nomor hp, *is_pendatang* dan foto. Pseudo code ditunjukkan pada pseudo code 5.23.

.

```
update

Function update(id, request):
    try:
        validateInput(request, {
            'nama' => 'required',
            'tanggal_lahir' => 'required',
            'alamat' => 'required',
            'pekerjaan' => 'required',
```

```
            'jenis_kelamin' => 'required',
            'no_hp' => 'required',
            'is_pendatang' => 'required',
            'foto' => 'required'
        })

        data = Jamaah::find(id)

        data->nama = request->nama
        data->tanggal_lahir = request->tanggal_lahir
        data->alamat = request->alamat
        data->pekerjaan = request->pekerjaan
        data->foto = request->foto
        data->jenis_kelamin = request->jenis_kelamin
        data->no_hp = request->no_hp
        data->dapukan = request->dapukan
        data->is_pendatang = request->is_pendatang
        data->id_bapak = request->id_bapak
        data->id_ibu = request->id_ibu
        data->id_suami = request->id_suami
        data->id_istri = request->id_istri

        data->save()

        redirectToRoute('ki.data_kelompok',
withSuccessMessage('Sukses Update Data!'))

    exceptionHandler(\Exception $e):
        redirectBackWithErrorMessage('Gagal Update Data!, terjadi
error: ' + e->getMessage())

End Function
```

Kode pseudo 5. 23 Kode Laravel Update Jamaah

5.5.4. Delete

Fungsi Deletes terbagi menjadi Soft-deletes dan Hard-deletes.

a. Soft-Delete

Soft-Delete adalah fungsi untuk menghapus data secara tidak permanen. Dalam hal ini data yang terhapus tetap berada didalam database akan tetapi tidak dapat diread. Sehingga memungkinkan pengguna untuk memunculkannya Kembali (restore). Pseudo code ditunjukkan pada pseudo code 5.24.

.

```
1    delete
2
3    Function delete(id):
4        try:
5            data = Jamaah::find(id)
6            data->delete()
7
8            return jsonResponse({
9                'status' => 'success',
10               'message' => 'Sukses Delete Data!'
11           })
12
13       exceptionHandler(\Exception $e):
14           return jsonResponse({
15               'status' => 'failed',
16               'message' => 'Gagal Delete Data!, terjadi error: ' + e-
     >getMessage()
17           })
18
19   End Function
```

Kode pseudo 5. 24 Kode Laravel Soft-Deletes

Dalam implementasi pseudo code 5.24. tidak terdapat fungsi soft-deletes dikarenakan implementasi untuk soft-delete digunakan didalam Model dari tabel tersebut. Oleh karna itu hanya menggunakan fungsi delete biasa untuk controller pada bagian tersebut.

b. Hard-Delete

Hard-Delete merupakan kebalikan dari fungsi soft-deletes yaitu untuk menghapus data secara permanen. Dalam hal ini data yang terhapus akan terhapus dan tidak dapat di pulihkan Kembali didalam database.

Kode sumber lengkap kode Laravel CRUD dapat dilihat pada Lampiran 25.

5.6. Fitur Tambahan

Fitur tambahan pada perancangan sistem informasi jamaah masjid Baitul Haq berguna untuk membantu dalam mempermudah penggunaan aplikasi. Fitur-fitur tambahan tersebut sebagai berikut:

5.6.1. Search

Fitur *search* atau fitur pencarian berada pada super admin dan admin, fitur tersebut berguna untuk membantu pengguna mencari data yang diinginkan pada tabel, hal tersebut mempermudah pengguna dalam memangkas waktu untuk menemukan data yang diinginkan. Fitur *search* atau pencarian dibangun dengan memanfaatkan *plugin* dari JavaScript yang bernama "dataTable". *Plugin* "dataTable" mempermudah dalam pencarian dan pengurturan data yang berada pada tabel yang diinginkan. Implementasi penggunaan *plugin* "dataTable" ditunjukkan pada kode sumber 5.13.

```
1    <table class="table table-striped table-vcenter dataTable">
```

Kode sumber 5. 14 Plugin dataTable pada HTML

Kode sumber 5.13 menunjukkan *plugin* "dataTable" dipanggil dalam tabel menggunakan HTML. Namun, untuk menggunakannya dataTabel harus diinisialisasikan terlebih dahulu yang ditunjukkan pada kode sumber 5.15.

```
1    <link rel="stylesheet"
     href="/js/plugins/datatables/dataTables.bootstrap4.css">
```

Kode sumber 5. 15 Inisialisasi plugin dataTable

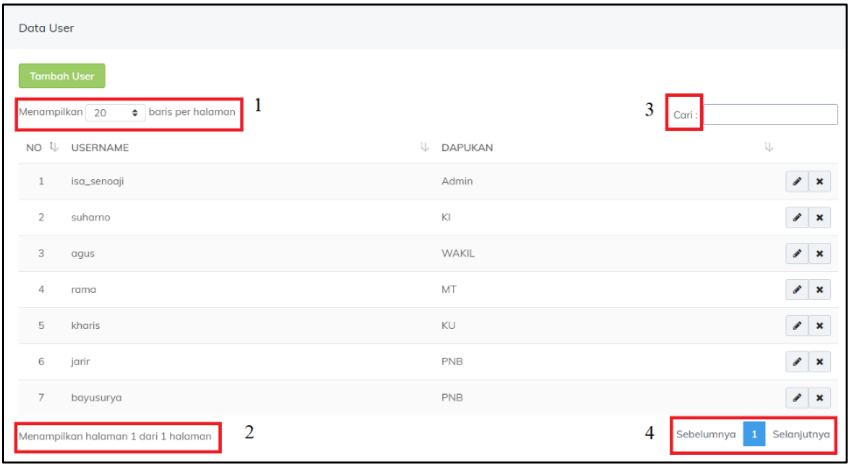
Pada kode sumber 5.14 menunjukkan bahwa dataTable dinisialisasikan sebelum digunakan. Setelah hal tersebut dilakukan, maka dataTabel diterapkan dalam baris kode JavaScript pada kode sumber 5.16.

```
1    $(".dataTable").dataTable({
2        columnDefs: [{
3                orderable: true,
4                searchable: true,
5                targets: 1
6            },
7            {
8                orderable: false,
9                searchable: false,
10               targets: -1
11           }
12       ],
13       scrollx:true,
14       scrolly:true,
15       pageLength: 20,
16       lengthMenu: [
17           [5, 8, 15, 20,-1],
18           [5, 8, 15, 20,"Semua"]
19       ],
20       language:{
21           "lengthMenu": "Menampilkan _MENU_ baris per halaman",
22           "zeroRecords": "Maaf, data yang dicari tidak
```

```
23  ditemukan!",
24          "info": "Menampilkan halaman _PAGE_ dari _PAGES_
25  halaman",
26          "infoEmpty": "Belum ada data yang tersedia!",
27          "infoFiltered": "(Disaring dari _MAX_ total baris
28  data!)",
29          "sSearch": "Cari :",
30          "paginate": {
31              "previous": "Sebelumnya",
32              "next": "Selanjutnya",
            }
        }
    })
```

Kode sumber 5. 16 dataTable dalam JavaScript

Pada kode sumber 5.15 pengaplikasian dataTable, baris kode 3 dan 4 menampilkan bahwa dataTable bisa melakukan *order* atau pengurutan serta dapat melakukan pencarian pada tabel data. Gambar merupakan hasil dari implementasi dataTabel.



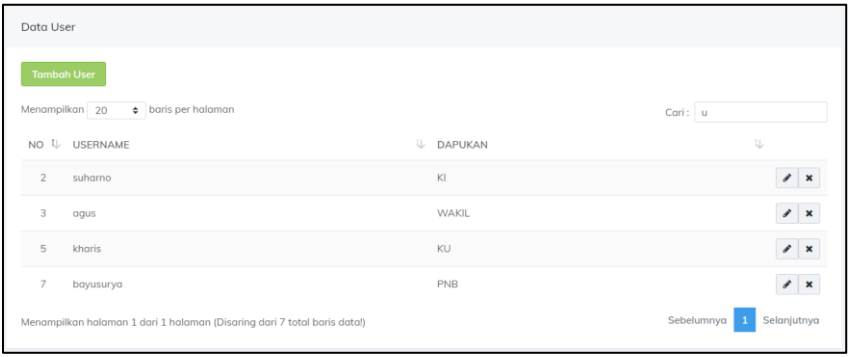
Gambar 5. 17 Hasil implementasi dataTable

Gambar 5.17 menunjukkan tabel yang ada terdapat beberapa atribut dari pengimplementasian kode sumber dataTable yang

ditunjukkan dengan angka. Angka 1 menampikan bahwa pengaturan jumlah data yang ditampilkan, ditunjukkan pada kode sumber 5.15 baris kode 21, serta ukuran dari jumlah yang ingin ditampilkan pada baris kode 18.

Angka 2 menampilkan halaman saat ini dari total jumlah data. Ditampilkan pada kode sumber 5.15 baris kode 23. Angka 3 menampilkan pencarian yang ditunjukkan pada sumber kode 5.15 baris kode 26. Dan terakhir adalah angka 4 yang digunakan untuk pembagian halaman yang ditunjukkan pada kode sumber 5.15 dari baris kode 27 hingga 29.

Hasil pencarian menggunakan dataTabel ditunjukkan pada gambar 5.18.



Gambar 5. 18 hasil pencarian menggunakan dataTabel

Gambar 5.18 menampilkan pencarian data yang mengandung huruf ‘u’ pada seluruk kolom. Ketika data ditemukan, data akan ditampilkan sesuai dengan pencarian yang diinginkan. Namun jika data tidak ditemukan, tidak akan ada yang ditampilkan. Hasil pencarian data tidak ditemukan pada gambar 5.19.

Gambar 5. 19 Hasil data tidak ditemukan.

Gambar 5.19 menampilkan pencarian data yang mengandung kata "coba" pada seluruh kolom, namun data tidak ditemukan sehingga menampilkan keluaran berupa "Maaf, data yang dicari tidak ditemukan". Hasil keluaran terebut dikarenakan pada sumber kode 5.15 baris kode 22 yang akan menampilkan pesan "Maaf, data yang dicari tidak ditemukan".

5.6.2. Dropdown Search

Fitur *dropsown search* atau fitur pencarian pada *dropdown* berada pada super admin dan admin, fitur tersebut berguna untuk membantu pengguna mencari data yang diinginkan pada pilihan *dropdown*, hal tersebut mempermudah pengguna dalam memangkas waktu untuk menemukan data yang diinginkan. Fitur *dropdown search* atau pencarian pada *dropdown* dibangun dengan memanfaatkan *plugin* dari JavaScript yang bernama "Select2". *Plugin* "Select2" mempermudah dalam pencarian data yang berada pada pilihan *dropdown* yang diinginkan. Implementasi penggunaan *plugin* "Select2" ditunjukkan pada kode sumber 5.17.

```
1    <link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-
     rc.0/dist/css/select2.min.css" rel="stylesheet" />
```

Kode sumber 5. 17 Inisialisasi plugin Select2

Setelah *plugin* dinisialisasikan, Select2 dipanggil dalam tombol pilihan *dropdown* yang diinginkan. Contoh sumber kode 5.18 pemanggilan Select 2 pada pilihan *dropdown*.

```
1    <select id="select_jamaah" class="select_pilihan form-select"
```

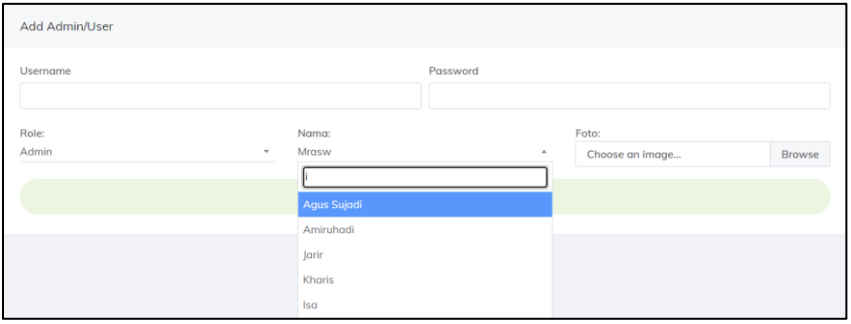
Kode sumber 5. 18 Pemanggilan Select2 pada HTML

Kode sumber 5.17 ketika pilihan *dropdown* terpicu, maka akan diarahkan pada fungsi Select2 yang berapa pada script JavaScript yang ditunjukkan pada sumber kode 5.19.

```
1    $('#select_jamaah).select2({
2        'width':'100%'
3    });
```

Kode sumber 5. 19 Script Select2

Kode sumber 5.19 akan menerapkan Select2 pada elemen HTML yang memiliki id "select_jamaah" sehingga, dalam penerapannya, pilihan *dropdown* pada yang ber-id "select_jamaah" dapat mencari data dengan memanfaatkan Select2. Gambar 5.20 merupakan hasil penerapan menggunakan *plugin* Select2.



Gambar 5. 20 Hasil penerapan Select2

Gambar 5.20 menampilkan hasil dari penerapan select 2 pada pencarian “Nama” yang mengandung huruf ‘i’.

5.6.3. Private Picture

Fitur *private picture* atau foto privat pada laman web aplikasi sistem digunakan pada keseluruhan pengguna yang mengakses aplikasi sistem. Fitur foto privat berguna sebagai menjaga privasi pengguna dari oknum yang ingin menyalah gunakan data berupa gambar. Dalam penerapan pembangunan fitur foto privat dengan memanfaatkan model “User”. Alasan penggunaan model adalah memastikan hak akses yang hanya bisa dilakukan oleh pengguna model. Kode sumber 5.20 dan 5.21 menampilkan penerapan sumber kode dalam pembangunan fitur foto privat.

```
1    <img class="img-avatar " src="{{ Auth::user()->getFotoProfile() }}"
     alt="">
```

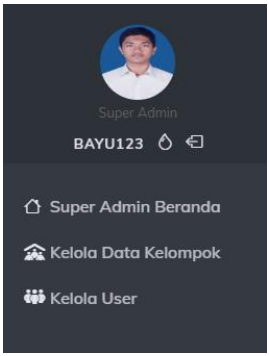
Kode sumber 5. 20 Pemanggilan foto menggunakan model

```
1   public function getFotoProfile(){
3       $path_foto = $this->foto;
4
5       if($path_foto!=null){
6        return
7   "" .base64_encode(Storage::get($path_foto));
        }else{
8           return  asset('/media/avatars/avatar15.jpg');
9       }
10  }
```

Kode sumber 5. 21 funngsi getFotoProfile dalam model

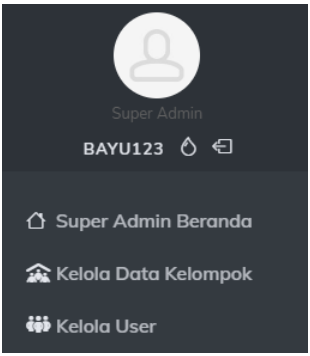
Pada kode sumber 5.21 menampilkan bahwa gambar akan digenerate langsung dengan memanggil fungsi “getFotoProfile”

dalam model “User”. Kemudian pada kode sumber 5.21 menampilkan fungsi “getFotoProfile” yang akan mengambil file gambar dari file “path_foto” sesuai dengan path foto yang disimpan yang berada pada database. Gambar 5.21 menampilkan hasil dari foto privat.



Gambar 5. 21 Hasil foto privat

Namun, jika pengguna tidak memiliki foto profil, maka akan menampilkan gambar default yang ditunjukkan oleh gamber 5.22.



Gambar 5. 22 Jika pengguna tidak memiliki foto privat

Gambar 5.22 menampilkan foto default jika pengguna tidak memiliki foto profil. Dalam kode sumber 5.20 baris kode 8

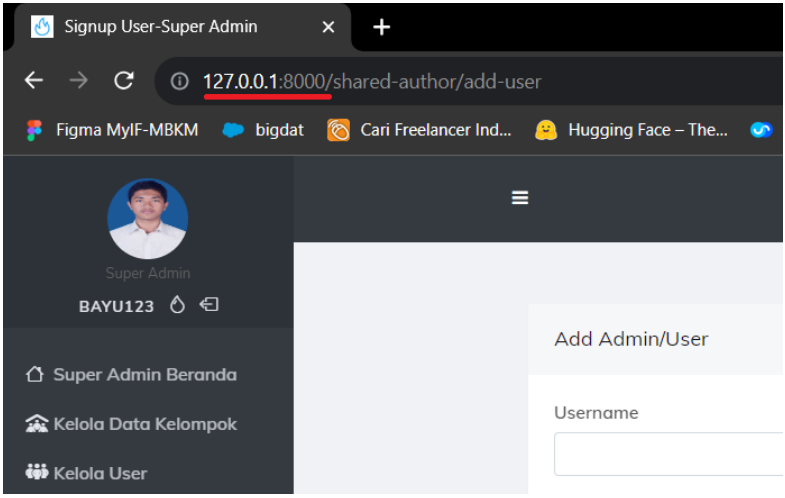
menunjukkan bahwa jika pengguna tidak memiliki foto profil, maka akan menggunakan foto default.

Untuk memastikan bahwa foto yang diunggah merupakan foto privat, adalah mengakses *path* file gambar pada web browser. Gambar 5.23 merupakan mengambil path gambar dari database, yang kemudian akan coba diakses dengan cara menggabungkan pada server.

username	password	role	last_login	deleted_at	id_jamaah	foto
bayu123	$2y$10$7.pm0xE0HEAHZUzaMV5y//.HqagHLPVJe3RBJA2bCDef...	superadmin	NULL	NULL	2	user-foto/jRWjNg66LJ3sgddH2BQvU76LxQxXs01GCiBEuGnw...

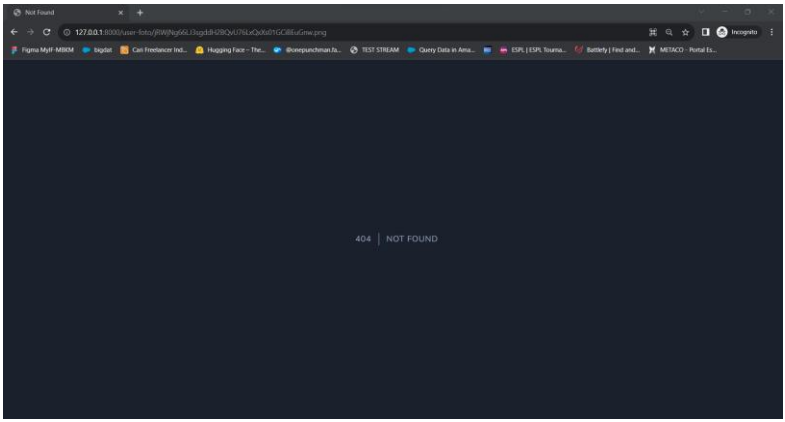
Gambar 5. 23 Sisi database untuk mengambil path foto

Setelah *path* didapatkan dari database, path tersebut akan digabungkan dengan *path* IP address atau url server yang ditampilkan pada gambar 5.24.



Gambar 5. 24 IP address arau URL server

Setelah didapatkan *path* IP address dan *path* gambar. *Path-path* tersebut digabungkan dan ditelusuri dan menghasilkan gambar 5.25.



Gambar 5. 25 Hasil menlakukan pencarian pada foto

Gambar 5.25 menunjukkan bahwa gambar tidak bisa diakses oleh publik, secara tidak langsung menunjukkan bahwa fitur foto privat berhasil dibuat.

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Perancangan Sistem Informasi Jamaah Masjid Baitul Haq Keputih. Pengujian dilakukan untuk memastikan fungsionalitas dan kesesuaian hasil implementasi arsitektur dengan analisis dan perancangan arsitektur.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap Perancangan Sistem Informasi Jamaah Masjid Baitul Haq Keputih guna menguji kemampuan arsitektur dalam penggunan sistem aplikasi.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut :

a. Kemampuan sistem untuk melayani tampilan aplikasi.
b. Kemampuan sistem untuk melayani query data dari aplikasi ke database.
c. Kemampuan sistem untuk menyimpan data yang diinput melalui aplikasi.
d. Kemampuan sistem untuk melayani upload file gambar dari aplikasi.
e. Kemampuan sistem untuk melakukan edit data melalui aplikasi.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai user yang akan menjalankan fitur-fitur. Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut :

1. User dapat mengakses halaman aplikasi user sesuai peran.
2. User dapat melihat data yang diinput.
3. User dapat melakukan menginput data.
4. User dapat melakukan edit data.
5. User dapat mengupload gambar.
6. User dapat lihat gambar.
7. User dapat melakukan edit gambar.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem aplikasi "Sistem Informasi jamaah Masjid Baitul Haq Keputih" terhadap kasus skenario uji coba. Tabel 6.1 di bawah ini menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

Table 1 Hasil Evaluasi Pengujian

Kriteria Pengujian	**Hasil Pengujian**
Sistem dapat melayani tampilan aplikasi	Terpenuhi
Sistem dapat melayani tampilan aplikasi sesuai peran	Terpenuhi
Sistem dapat melakukan penyimpanan data gambar ke database	Terpenuhi
Sistem dapat menampilkan isi database kepada pengguna	Terpenuhi
Sistem dapat melayani	Terpenuhi

perubahan data dalam database	
Sistem dapat melayani upload foto	Terpenuhi
Sistem dapat melayani menampilkan foto	Terpenuhi
Sistem dapat melayani edit foto	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan perancangan aplikasi Sistem Informasi Jamaah Masjid Baitul Haq Keputih adalah sebagai berikut :

a. Arsitektur database dalam perancangan sistem informasi pengelolaan jamaah masjid Baitul Haq Keputih dengan menggunakan manajemen basis data MySQL, pembautan tabel yang memisahakan antara user (termasuk pengurus masjid) dengan jamaah masjid, serta pengelolaan data dengan memanfaatkan *create, read, update* dan *delete.*
b. Rekayasa sistem dalam menanganni peran yang berbeda adalah dengan memanfaatkan *middleware* yang terdapat dalam laravel penjelasan *middleware* lebih detail ada pada sub bab 5.1.2.

7.2. Saran

Saran untuk perancangan aplikasi Sistem Informasi Masjid Baitul Haq Keputih adalah sebagai berikut :

a. Lakukan indexing pada kolom-kolom yang sering digunakan sebagai parameter pencarian gna mempercepat performa pencarian.

DAFTAR PUSTAKA

[1] N. Y. Arifin, R. I. Borman, I. Ahmad, S. S. Tyas, H. Sulistiani, A. Hardiansyah and G. P. Suri, Analisa Perancangan Sistem Informasi, Batam: Yayasan Cendikia Mulia Mandiri, 2021.

[2] J. Burch, F. R. Strater and G. Grudnitski, Information Systems: Theory and Practice, Hamilton: New York, 1974.

[3] M. Anniss, What is Website and How Do I Use It?, Britannica: Britannica Educational Publishing, 2014.

[4] N. Brügger, "Website history and the website as an object of study," *New Media & Society*, 2009.

[5] S. Neupane, "Developing A Static Website And Deploying It To," Centria, 2020.

[6] C. Amza, A. Chandra, A. L. Cox, S. Elkinety, R. Gil and K. Rajamani, "Specification and Implementation of Dynamic Web Site Benchmarks," *IEEE*, 2002.

LAMPIRAN

Lampiran 1.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:INvevRi2IqKy6EM2SwFbuTOmU39iq42gmgr828Oiznw=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
# DB_DATABASE=dummy_jm
DB_DATABASE=db_jm
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
```

```
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Lampiran 2.

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CheckAccess
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)
$next
     * @return
\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
```

```
     */
    public function handle(Request $request, Closure $next,
$role)
    {

        $get_role_text = \Config::get('constants');

        if(Auth::check() && Auth::user()->role == $role){
            return $next($request);
        }else{

            // dd(Auth::user()->id_role);
            switch (Auth::user()->role) {
                case $get_role_text['role']['superadmin']:
                    // dd(1);
                    return redirect()-
>route('superadmin.beranda');
                    break;
                case $get_role_text['role']['admin']:
                    // dd(1);
                    return redirect()->route('admin.beranda');
                    break;
                case $get_role_text['role']['ki']:
                    // dd(1);
                    return redirect()->route('ki.beranda');
                    break;

                case $get_role_text['role']['wakil']:
                    return redirect()->route('wakil.beranda');
                    # code...
                    break;

                case $get_role_text['role']['ku']:
                    return redirect()->route('ku.beranda');
                    # code...
                    break;

                case $get_role_text['role']['mt']:
                    return redirect()->route('mt.beranda');
                    # code...
                    break;
```

```
                case $get_role_text['role']['pnb']:
                    return redirect()->route('pnb.beranda');
                    # code...
                    break;
                case $get_role_text['role']['pengabsen']:
                    return redirect()-
>route('pengabsen.beranda');
                    # code...
                    break;
            }
        }

        return redirect()->route('login')->with('failed',
'Access denied.');
    }
}
```

Lampiran 3

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class IsLogged
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)
$next
     * @return
\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
```

```
     */
    public function handle(Request $request, Closure $next)
    {
        // dd($id_role);
        if(Auth::check()){
            return $next($request);
        }

        return redirect()->route('login')->with('failed', 'Sesi
anda habis');
    }
}
```

Lampiran 4

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class IsNotLogged
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)
$next
     * @return
\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next)
    {
        // dd($id_role);
        if(!Auth::check() ){
```

```
            // dd(Auth::check());
            return $next($request);
        }
        else{

            // dd(Auth::user()->id_role);
            switch (Auth::user()->role) {
                case 'superadmin':
                    // dd(1);
                    return redirect()-
>route('superadmin.beranda');
                    break;
                case 'admin':
                    // dd(1);
                    return redirect()->route('admin.beranda');
                    break;
                case 'ki':
                    // dd(1);
                    return redirect()->route('ki.beranda');
                    break;

                case 'wakil':
                    return redirect()->route('wakil.beranda');
                    # code...
                    break;

                case 'ki':
                    return redirect()->route('ku.beranda');
                    # code...
                    break;

                case 'mt':
                    return redirect()->route('mt.beranda');
                    # code...
                    break;

                case 'pnb':
                    return redirect()->route('pnb.beranda');
                    # code...
                    break;
```

```
                case 'pengabsen':
                    return redirect()-
>route('pengabsen.beranda');
                    # code...
                    break;
            }
        }
    }
}
```

Lampiran 5

```
<?php

namespace App\Http;

use GuzzleHttp\Middleware;
use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your
application.
     *
     * @var array<int, class-string|string>
     */
    protected $middleware = [
        // \App\Http\Middleware\TrustHosts::class,
        \App\Http\Middleware\TrustProxies::class,
        \Fruitcake\Cors\HandleCors::class,

\App\Http\Middleware\PreventRequestsDuringMaintenance::class,

\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
```

```
        \App\Http\Middleware\TrimStrings::class,

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull
::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            //
\Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],

        'api' => [
            //
\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStatef
ul::class,
            'throttle:api',

\Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];

    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used
individually.
```

```
     *
     * @var array<string, class-string|string>
     */
    protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' =>
\Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'cache.headers' =>
\Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' =>
\App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' =>
\Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' =>
\Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' =>
\Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' =>
\Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,


        // Middleware role
        'id_role' => \App\Http\Middleware\CheckAccess::class,

        'is_logged' => \App\Http\Middleware\IsLogged::class,
        'isnot_logged' =>
\App\Http\Middleware\IsNotLogged::class,
    ];
}
```

Lampiran 6

```
<!-- Header -->
    <header id="page-header">
        <!-- Header Content -->
        <div class="content-header">

            <!-- Left Section -->
```

```
            <div class="content-header-section">

                <button type="button" class="btn btn-circle btn-
dual-secondary" data-toggle="layout" data-
action="sidebar_toggle">
                    <i class="fa fa-navicon"></i>
                </button>
            </div>
            <!-- END Left Section -->

            <!-- Right Section -->
            <div class="content-header-section">
                <!-- User Dropdown -->
                <div class="btn-group" role="group">
                    <button type="button" class="btn btn-rounded
btn-dual-secondary" id="page-header-user-dropdown" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <i class="fa fa-user d-sm-none"></i>
                        <span class="d-none d-sm-inline-
block">{{ $nama }}</span>
                        <i class="bi bi-caret-down-fill ml-
5"></i>
                    </button>
                    <div class="dropdown-menu dropdown-menu-
right min-width-200" aria-labelledby="page-header-user-
dropdown">
                        <h5 class="h6 text-center py-10 mb-5
border-b text-uppercase">User</h5>
                        <a class="dropdown-item"
href="be_pages_generic_profile.html">
                            <i class="si si-user mr-5"></i>
Profile
                        </a>
                        <a class="dropdown-item d-flex align-
items-center justify-content-between"
href="be_pages_generic_inbox.html">
                            <span><i class="si si-envelope-open
mr-5"></i> Inbox</span>
                            <span class="badge badge-
primary">3</span>
                        </a>
```

```
                        <a class="dropdown-item"
href="be_pages_generic_invoice.html">
                            <i class="si si-note mr-5"></i>
Invoices
                        </a>
                        <div class="dropdown-divider"></div>

                        <!-- Toggle Side Overlay -->
                        <a class="dropdown-item"
href="javascript:void(0)" data-toggle="layout" data-
action="side_overlay_toggle">
                            <i class="si si-wrench mr-5"></i>
Settings
                        </a>
                        <!-- END Side Overlay -->

                        <div class="dropdown-divider"></div>
                            <form id="logout-form"
action="{{ route('logout') }}" method="POST">
                                @csrf
                            </form>

                            <a href="#"
onclick="event.preventDefault();
document.getElementById('logout-form').submit();"
class="dropdown-item">
                                <i class="bi bi-box-arrow-
left"></i> Sign Out
                            </a>
                    </div>
                </div>
                <!-- END User Dropdown -->

                <!-- Notifications -->
                <div class="btn-group" role="group">
                    <button type="button" class="btn btn-rounded
btn-dual-secondary" id="page-header-notifications" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                        <i class="bi bi-bell-fill"></i>
                        <span class="badge badge-primary badge-
pill">5</span>
```

```
                    </button>
                    <div class="dropdown-menu dropdown-menu-
right min-width-300" aria-labelledby="page-header-
notifications">
                        <h5 class="h6 text-center py-10 mb-0
border-b text-uppercase">Notifications</h5>
                        <ul class="list-unstyled my-20">
                            <li>
                                <a class="text-body-color-dark
media mb-15" href="javascript:void(0)">
                                    <div class="ml-5 mr-15">
                                        <i class="fa fa-fw fa-
check text-success"></i>
                                    </div>
                                    <div class="media-body pr-
10">
                                        <p class="mb-0">You've
upgraded to a VIP account successfully!</p>
                                        <div class="text-muted
font-size-sm font-italic">15 min ago</div>
                                    </div>
                                </a>
                            </li>
                            <li>
                                <a class="text-body-color-dark
media mb-15" href="javascript:void(0)">
                                    <div class="ml-5 mr-15">
                                        <i class="fa fa-
exclamation-triangle text-warning"></i>
                                    </div>
                                    <div class="media-body pr-
10">
                                        <p class="mb-0">Please
check your payment info since we can't validate them!</p>
                                        <div class="text-muted
font-size-sm font-italic">50 min ago</div>
                                    </div>
                                </a>
                            </li>
                            <li>
                                <a class="text-body-color-dark
```

```
media mb-15" href="javascript:void(0)">
                                    <div class="ml-5 mr-15">
                                        <i class="fa fa-fw fa-
times text-danger"></i>
                                    </div>
                                    <div class="media-body pr-
10">
                                        <p class="mb-0">Web
server stopped responding and it was automatically
restarted!</p>
                                        <div class="text-muted
font-size-sm font-italic">4 hours ago</div>
                                    </div>
                                </a>
                            </li>
                            <li>
                                <a class="text-body-color-dark
media mb-15" href="javascript:void(0)">
                                    <div class="ml-5 mr-15">
                                        <i class="fa fa-fw fa-
exclamation-triangle text-warning"></i>
                                    </div>
                                    <div class="media-body pr-
10">
                                        <p class="mb-0">Please
consider upgrading your plan. You are running out of space.</p>
                                        <div class="text-muted
font-size-sm font-italic">16 hours ago</div>
                                    </div>
                                </a>
                            </li>
                            <li>
                                <a class="text-body-color-dark
media mb-15" href="javascript:void(0)">
                                    <div class="ml-5 mr-15">
                                        <i class="fa fa-fw fa-
plus text-primary"></i>
                                    </div>
                                    <div class="media-body pr-
10">
                                        <p class="mb-0">New
```

```
purchases! +$250</p>
                                        <div class="text-muted
font-size-sm font-italic">1 day ago</div>
                                    </div>
                                </a>
                            </li>
                        </ul>
                        <div class="dropdown-divider"></div>
                        <a class="dropdown-item text-center mb-
0" href="javascript:void(0)">
                            <i class="fa fa-flag mr-5"></i> View
All
                        </a>
                    </div>
                </div>
                <!-- END Notifications -->
                <button type="button" class="btn btn-circle btn-
dual-secondary" data-toggle="layout" data-
action="side_overlay_toggle">
                    <i class="fa fa-tasks"></i>
                </button>

            </div>
            <!-- END Right Section -->
        </div>
        <!-- END Header Content -->

        <!-- Header Loader -->
        <div id="page-header-loader" class="overlay-header bg-
primary">
            <div class="content-header content-header-fullrow
text-center">
                <div class="content-header-item">
                    <i class="fa fa-sun-o fa-spin text-
white"></i>
                </div>
            </div>
        </div>
        <!-- END Header Loader -->
    </header>
<!-- END Header -->
```

Lampiran 7

```
        <nav id="sidebar">

            <div class="sidebar-content">
                <!-- Side User -->
                <div class="content-side content-side-full
content-side-user px-10 align-parent">
                    <!-- Visible only in mini mode -->
                    <div class="sidebar-mini-visible-b align-v
animated fadeIn">
                        <img class="img-avatar img-avatar32"
src="{{ asset('/media/avatars/avatar15.jpg') }}" alt="">

                    </div>
                    <!-- END Visible only in mini mode -->

                    <!-- Visible only in normal mode -->
                    <div class="sidebar-mini-hidden-b text-
center">
                        <a class="" >

                                <img class="img-avatar "
src="{{ Auth::user()->getFotoProfile() }}" alt="">

                            <br>
                            <small>{{ $id_role }}</small>
                        </a>
                        <ul class="list-inline mt-5">
                            <li class="list-inline-item">
                                <a class="link-effect text-dual-
primary-dark font-size-xs font-w600 text-uppercase"
href="#">{{ $nama }}</a>
                            </li>

                            <li class="list-inline-item">
```

```
                                <!-- Layout API, functionality
initialized in Template._uiApiLayout() -->
                                <a class="link-effect text-dual-
primary-dark" data-toggle="layout" data-
action="sidebar_style_inverse_toggle" href="javascript:void(0)">
                                    <i class="bi bi-
droplet"></i>
                                </a>
                            </li>

                            {{-- logout --}}
                            <li class="list-inline-item">

                                <form id="logout-form"
action="{{ route('logout') }}" method="POST">
                                    @csrf
                                </form>

                                <a href="#"
onclick="event.preventDefault();
document.getElementById('logout-form').submit();" class="link-
effect text-dual-primary-dark">
                                    <i class="bi bi-box-arrow-
left"></i>
                                </a>
                            </li>
                        </ul>
                    </div>
                    <!-- END Visible only in normal mode -->
                </div>
                <!-- END Side User -->

                <!-- Side Navigation -->
                <div class="content-side content-side-full">
                    <ul class="nav-main">

                        <li>
                            <a
href="{{ route('superadmin.beranda') }}"><i class="bi bi-
house"></i><span class="sidebar-mini-hide">{{ $id_role."
"."Beranda" }}</span></a>
```

```
                        </li>

                        <li>
                            <a
href="{{ route('authorshared.data_kelompok') }}"><i class="fa
fa-people-roof"></i><span class="sidebar-mini-hide">Kelola Data
Kelompok</span></a>
                        </li>
                        <li>
                            <a
href="{{ route('superadmin.data_user') }}"><i class="fa fa-
people-group"></i><span class="sidebar-mini-hide">Kelola
User</span></a>
                        </li>

                        <li class="nav-main-heading"><span
class="sidebar-mini-visible"></span><span class="sidebar-mini-
hidden">Absensi</span>
                            <a href="#" class="bi bi-person-
lines-fill"> Cabe Rawit</a>
                            <a href="#" class="bi bi-person-
lines-fill"> Pra-Remaja</a>
                            <a href="#" class="bi bi-person-
lines-fill"> Muda-Mudi</a>
                        </li>
                        <li class="nav-main-heading"><span
class="sidebar-mini-visible"></span><span class="sidebar-mini-
hidden">Hasil Musyawarah</span></li>
                        <li class="nav-main-heading"><span
class="sidebar-mini-visible"></span><span class="sidebar-mini-
hidden">Agenda</span></li>

                    </ul>
                </div>
                <!-- END Side Navigation -->
            </div>
            <!-- Sidebar Content -->
        </nav>
        <!-- END Sidebar -->
```

Lampiran 8

```
<aside id="side-overlay">
     <!-- Side Header -->
     <div class="content-header content-header-fullrow">
        <div class="content-header-section align-parent">
            <!-- Close Side Overlay -->
            <!-- Layout API, functionality initialized in
Template._uiApiLayout() -->
            <button type="button" class="btn btn-circle btn-
dual-secondary align-v-r" data-toggle="layout" data-
action="side_overlay_close">
                <i class="fa fa-times text-danger"></i>
            </button>
            <!-- END Close Side Overlay -->

            <!-- User Info -->
            <div class="content-header-item">
                <a class="img-link mr-5"
href="be_pages_generic_profile.html">

                    {{-- <img class="img-avatar img-avatar32"
src="assets/media/avatars/avatar15.jpg" alt=""> --}}
                    <img class="img-avatar "
src="{{ Auth::user()->getFotoProfile() }}" alt="">

                </a>
                <a class="align-middle link-effect text-primary-
dark font-w600"
href="be_pages_generic_profile.html">{{ $nama }}</a>
            </div>
            <!-- END User Info -->
        </div>
    </div>
    <!-- END Side Header -->

    <div class="content-side">

        <div class="block pull-r-l">
            <div class="block-header bg-body-light">
```

```
                <h3 class="block-title">
                    <i class="fa fa-fw fa-pencil font-size-
default mr-5"></i>Profile
                </h3>
                <div class="block-options">
                    <button type="button" class="btn-block-
option" data-toggle="block-option" data-
action="content_toggle"></button>
                </div>
            </div>

            <div class="block-content">
                <form id="updateProfile"
action="{{ route('authorshared.update_profile') }}"
method="post" enctype="multipart/form-data">
                    @csrf

                    <div class="form-group mb-15">
                        <label for="side-overlay-
username">Username</label>
                        <div class="input-group">
                            <input type="text" class="form-
control" id="side-overlay-username" name="side-overlay-username"
placeholder="Your name.." value="{{ $nama }}">
                            <div class="input-group-append">
                                <span class="input-group-text">
                                    <i class="fa fa-user"></i>
                                </span>
                            </div>
                        </div>
                    </div>

                    <div class="form-group mb-15">
                        <label for="side-overlay-profile-
pict">Foto Profile Baru</label>
                        <div class="input-group">
                            <input type="file" class="form-
control" accept="image/*" id="side-overlay-profile-pict"
name="side-overlay-profile-pict" data-toggle="custom-file-
input">
                            <div class="input-group-append">
```

```
                                <span class="input-group-text">
                                    <i class="fa fa-image"></i>
                                </span>
                            </div>
                        </div>
                    </div>

                    <div class="form-group mb-15">
                        <label for="side-overlay-profile-
password">New Password</label>
                        <div class="input-group">
                            <input type="password" class="form-
control" id="side-overlay-profile-password" name="side-overlay-
profile-password" placeholder="New Password..">
                            <div class="input-group-append">
                                <span class="input-group-text">
                                    <i class="fa fa-
asterisk"></i>
                                </span>
                            </div>
                        </div>
                    </div>
                    <div class="form-group mb-15">
                        <label for="side-overlay-profile-
password-confirm">Confirm New Password</label>
                        <div class="input-group">
                            <input type="password" class="form-
control" id="side-overlay-profile-password_confirmation"
name="side-overlay-profile-password_confirmation"
placeholder="Confirm New Password..">
                            <div class="input-group-append">
                                <span class="input-group-text">
                                    <i class="fa fa-
asterisk"></i>
                                </span>
                            </div>
                        </div>
                    </div>
                    <div class="form-group row">
                        <div class="col-6">
                            <button type="submit"
```

```
onclick="updateProfile(event)" class="btn btn-block btn-alt-
primary">
                                    <i class="fa fa-refresh mr-
5"></i> Update
                                </button>
                            </div>
                        </div>
                    </form>
                </div>

            </div>

        </div>
```

Lampiran 9

```
<?php
use GuzzleHttp\Middleware;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Auth\AuthController;
use App\Http\Controllers\User\UserController;
use App\Http\Controllers\Pengurus\PengurusController;

Route::get('/',[AuthController::class, 'index'])->name('login')-
>middleware('isnot_logged');
Route::post('/login',[AuthController::class, 'auth_login'])-
>name('login_process');

Route::post('/auth_signup',[AuthController::class,
'auth_signup'])->name('signup_process');

Route::post('/logout', [AuthController::class, 'logout'])-
>name('logout');

// middlware admin
Route::middleware(['id_role:admin','is_logged'])->group(function
() {
```

```
    Route::controller(UserController::class)->prefix('user')-
>name('admin.')->group(function(){

        Route::get('/admin','admin_index')->name('beranda');
        Route::get('/add-user','admin_add_user')-
>name('add_user');

        // Route::post('/auth-admin_add-user','admin_add_user')-
>name('add_user_process');

    });
});

// middleware superadmin
Route::middleware(['id_role:superadmin','is_logged'])-
>group(function () {

    Route::controller(UserController::class)->prefix('user')-
>name('superadmin.')->group(function(){

        Route::get('/superadmin','superadmin_index')-
>name('beranda');

Route::get('/superadmin_show_user','superadmin_show_user')-
>name('data_user');

        Route::get('/superadmin_add-
user','superadmin_add_user')->name('add_user');
        Route::post('/auth-superadmin_add-
user','auth_superadmin_add_user')->name('add_user_process');

        Route::post('/superadmin_update_user/{id}',
'update_user')->name('update_user');
        Route::get('/superadmin_get_user/{id}','get_user')-
>name('get_user');
        Route::delete('/superadmin_delete_user/{id}',
'destroy_user')->name('delete_user');
    });
});
Route::controller(UserController::class)->prefix('shared-
```

```
author')->name('authorshared.')->group(function(){

    Route::get('/add-jamaah','add_jamaah')->name('add_jamaah');
    Route::post('/auth-_add-user','admin_add_user')-
>name('add_user_process');

    Route::get('/jamaah_show','jamaah_show')-
>name('data_kelompok');

    Route::post('/update_jamaah/{id}', 'update')-
>name('update_jamaah');
    Route::get('/get_jamaah/{id}','get_jamaah')-
>name('get_jamaah');
    Route::delete('/delete_jamaah/{id}', 'destroy')-
>name('delete_jamaah');

    Route::post('/update_profile', 'updateProfile')-
>name('update_profile');
});

Route::middleware(['id_role:KI'])->prefix('ki')->name('ki.')-
>group(function () {
    Route::get('/pengurus',[PengurusController::class,
'index'])->name('beranda');

});
Route::middleware(['id_role:wakil'])->prefix('wakil')-
>name('wakil.')->group(function () {
    Route::get('/pengurus',[PengurusController::class,
'index'])->name('beranda');

});
Route::middleware(['id_role:KU'])->prefix('ku')->name('ku.')-
>group(function () {
    Route::get('/pengurus',[PengurusController::class,
'index'])->name('beranda');

});
Route::middleware(['id_role:MT'])->prefix('mt')->name('mt.')-
>group(function () {
    Route::get('/pengurus',[PengurusController::class,
```

```
'index'])->name('beranda');

});
Route::middleware(['id_role:PNB'])->prefix('pnb')->name('pnb.')-
>group(function () {
    Route::get('/pengurus',[PengurusController::class,
'index'])->name('beranda');
});

Route::prefix('kelompok')->name('shared.')->group( function (){
    Route::get('/show',[PengurusController::class, 'show'])-
>name('data_kelompok');
});

Route::controller(PengurusController::class)->group(function(){

    Route::get('/pengurus','index')->name('pengurus-beranda');
    Route::get('/show','show')->name('pengurus-data_kelompok');
    Route::get('/add', 'add')->name('pengurus-add_data');
    Route::post('/store', 'store')->name('store');
    Route::get('/edit/{id}', 'edit')->name('pengurus-
edit_data');
    Route::post('/update/{id}', 'update')->name('update');
    Route::post('/delete/{id}', 'delete')->name('delete');

    Route::prefix('kelompok')->name('shared.')->group( function
(){
        Route::get('/show',[PengurusController::class, 'show'])-
>name('data_kelompok');
    });
});
```

Lampiran 10

```
<?php
namespace App\Http\Controllers\User;
use App\Models\User;
use App\Models\Jamaah;
use Illuminate\Http\Request;
```

```
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;


class UserController extends Controller
{
    //Super Admin Function
    public function superadmin_index(){
        return view('SuperAdmin.home',[
            'title' => 'Dashboard'."-
".\Config::get('constants.role_text.superadmin'),
            'foto' => Auth::user()->foto,
            'idProfile' => Auth::user()->id,
            'id_role' =>
\Config::get('constants.role_text.superadmin'),
            'nama' => Auth::user()->username,
        ]);
    }

    public function superadmin_add_user(){
        return view('SuperAdmin.add_admin', [
            'title'=>'Signup User'."-
".\Config::get('constants.role_text.superadmin'),
            'foto' => Auth::user()->foto,
            'id_role' =>
\Config::get('constants.role_text.superadmin'),
            'nama' => Auth::user()->username,
            'jamaah' => Jamaah::all(),
        ]);
    }


    public function superadmin_show_user(){
        $get_role_text = \Config::get('constants');
        $user = User::where('role','!=', 'superadmin')->get();
        return view('SuperAdmin.kelola_user',[
            'title' => "Data Jama'ah"."-
".\Config::get('constants.role_text.superadmin'),
            'foto' => Auth::user()->foto,
```

```
            'id_role' =>
\Config::get('constants.role_text.superadmin'),
            'nama' => Auth::user()->username,
            'user' => $user->all(),
            'jamaah' => Jamaah::all(),
        ]);
    }

    public function auth_superadmin_add_user(Request $request){
        // ddd($request);

        $validatedData = $request->validate([
            'username' => 'required',
            'foto' => 'image',
            'role' => 'required',
            'password' => 'required',
            'id_jamaah' => 'required'
        ]);

        try{
            $exist = User::where('username',
$validatedData['username'])->exists();

            if($exist){
                throw new \Exception("nama terduplikasi", 503);
            }

            $validatedData['password'] =
bcrypt($validatedData['password']);

            if($request->file('foto')){
                $validatedData['foto'] = $request->file('foto')-
>store('user-foto');
            }



            $user = new User;
            $user->username = $validatedData['username'];
            $user->role = $validatedData['role'];
            $user->password = $validatedData['password'];
```

```
            $user->id_jamaah = $validatedData['id_jamaah'];
            $user->foto = $validatedData['foto'];
            // $user->save();

            if ($user->save()) {
                # code...
                $request->session()->flash('success',
'Penambahan User akun berhasil');

                return redirect()-
>route('superadmin.data_user');
            }

        } catch (\Exception $e) {
            //throw $th;
            return redirect()->back()->withErrors(['error' =>
$e->getMessage()]);
        }
    }

    public function get_user($id)
    {
        // Ambil data yang akan diedit dari database

        try {
            $data = User::find($id);

            if ($data) {
                return response()->json($data);
            } else {
                return response()->json(['message' => 'Data not
found'], 404);
            }

        } catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            return response()->json(['message' => 'Failed to
find data'], 500);  //throw $th;
        }
    }
```

```
    public function update_user(Request $request,  $id)
    {
        $data = User::find($id);

        if($request['username'] == $data->username){

            $validatedData = $request->validate([
                'username' => 'required|regex:/(^([a-z0-
9]+)(\d+)?$)/u',
                'password' => 'max:255',
                'role' => 'required',
                'id_jamaah' =>'required',
            ]);

        }else{
            $validatedData = $request->validate([
                'username' =>
'required|unique:users|regex:/(^([a-z0-9]+)(\d+)?$)/u',
                'password' => 'max:255',
                'role' => 'required',
                'id_jamaah' =>'required',
            ]);

        }

        try {
            if (!$data) {
                throw new \Exception("akun tidak ada", 400);
            }

            $data->username = $validatedData['username'];

            if($validatedData['password']!=null){
                $data->password =
bcrypt($validatedData['password']);
            }

            $data->role = $validatedData['role'];
            $data->id_jamaah = $validatedData['id_jamaah'];

            $data->save();
```

```
            return redirect()->back()->with('success', 'update
berhasil');

        }catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            // return response()->json(['message' => 'Failed to
delete data'], 500);   //throw $th;
            return redirect()->back()->with('failed', 'update
gagal'." ".$e->getMessage());
        }
    }

    public function destroy_user($id)
    {
        try {

            if ($data = User::findOrFail($id)) {
                $data->delete();
                return response()->json(['message' => 'Data
deleted successfully'], 200);
            }

        } catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            // return response()->json(['message' => 'Failed to
delete data'], 500);
            return response()->json(['message' => $e-
>getMessage()], 500);
            // return redirect()->back()->withErrors(['error' =>
$e->getMessage()]);
        }
    }
    //Admin Function
    public function admin_index(){
        $foto_test = Auth::user()->foto;
        $path_foto = Auth::user()->foto;
        return view('Admin.home',[
            'title' => 'Dashboard'."-
".\Config::get('constants.role_text.admin'),
            // 'foto' => storage_path('app/'.$path_foto),
```

```
            'foto' => $path_foto,
            'id_role' =>
\Config::get('constants.role_text.admin'),
            'nama' => Auth::user()->username,
        ]);
    }


    public function admin_add_user(){
        $get_role_text = \Config::get('constants');
        $apaitu = Auth::user()->role;
        // dd($apaitu);
        return view('Admin.add_user', [
            'title'=>'Signup User'."-
".\Config::get('constants.role_text.'.$apaitu),
            'foto' => Auth::user()->foto,
            'id_role' =>
\Config::get('constants.role_text.'.$apaitu),
            'nama' => Auth::user()->username,
            'jamaah' => Jamaah::paginate(5),
        ]);
    }
    // shared
    public function jamaah_show(){
        $get_role_text = \Config::get('constants');
        $apaitu = Auth::user()->role;
        // dd($apaitu);
        return view('Shared.kelola_jamaah',[
            'title' => "Data Jama'ah"."-
".\Config::get('constants.role_text.'.$apaitu),
            'foto' => Auth::user()->foto,
            'id_role' =>
\Config::get('constants.role_text.'.$apaitu),
            'nama' => Auth::user()->username,
            'jamaah' => Jamaah::all(),
        ]);
    }

    public function add_jamaah(){
        $get_role_text = \Config::get('constants');
        $apaitu = Auth::user()->role;
```

```
        return view('Shared.add_jamaah', [
            'title'=>'Signup User'."-
".\Config::get('constants.role_text.'.$apaitu),
            'foto' => Auth::user()->foto,
            'id_role' =>
\Config::get('constants.role_text.'.$apaitu),
            'nama' => Auth::user()->username,
            'jamaah' => Jamaah::all(),
        ]);
    }

    public function get_jamaah($id)
    {
        // Ambil data yang akan diedit dari database

        try {
            //code...
            $data = Jamaah::find($id);

            // dd($data);

            return response()->json($data);

        } catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            return response()->json(['message' => 'Failed to
delete data'], 500);  //throw $th;
        }
    }

    public function update(Request $request,  $id)
    {
        $validatedData = $request->validate([
            'nama' => 'required',
            'alamat' => 'required',
            'tanggal_lahir' => 'required',
            'jenis_kelamin' => 'required',
            'foto',
            'nomor_hp' => 'required',
        ]);
```

```
        try {
            $data = Jamaah::find($id);
            $data->nama = $validatedData['nama'];
            $data->alamat = $validatedData['alamat'];
            $data->tanggal_lahir =
$validatedData['tanggal_lahir'];
            $data->jenis_kelamin =
$validatedData['jenis_kelamin'];
            $data->foto = $validatedData['foto'];
            $data->nomor_hp = $validatedData['nomor_hp'];

            $data->save();

            return response()->json(['message' => 'Data updated
successfully'], 200);

        }catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            return response()->json(['message' => 'Failed to
delete data'], 500);   //throw $th;
        }
    }

    public function destroy($id)
    {
        try{
            $data = Jamaah::find($id);
            $data->delete();

            return response()->json([
                'status'    => 'success',
                'message'   => 'Sukses Delete Data!'
            ]);

        } catch(\Exception $e){
            return response()->json([
                'status'    => 'failed',
                'message'   => 'Gagal Delete Data!, terjadi
error: '.$e->getMessage()
            ]);
        }
```

```
    }

    public function updateProfile(Request $request)
    {

        $validatedData = $request->validate([
            'side-overlay-username' => 'required',
            'side-overlay-profile-pict' => 'image',
            'side-overlay-profile-password' =>
'max:255|confirmed'
        ]);

        try {

            $id = Auth::user()->id;

            if(Auth::user()->username!=$request['side-overlay-
username']){
                $exist = User::where('username',
$validatedData['side-overlay-username'])->exists();

                if($exist){
                    throw new \Exception("nama terduplikasi",
503);
                }
            }

            $data = User::find($id);

            if($validatedData['side-overlay-profile-
password']!=null){
                $data->password = bcrypt($validatedData['side-
overlay-profile-password']);
            }

            $oldFoto = $data->foto;

            if($request->file('side-overlay-profile-pict')){
                $validatedData['side-overlay-profile-pict'] =
$request->file('side-overlay-profile-pict')->store('user-foto');
                $data->foto = $validatedData['side-overlay-
```

```
profile-pict'];

                if($oldFoto!=null&&Storage::exists($oldFoto)){
                    Storage::delete($oldFoto);
                }
            }

            $data->username = $validatedData['side-overlay-
username'];

            $data->save();

            return redirect()->back()->with('success', 'update
berhasil');

        }catch (\Exception $e) {
            // Tangani kesalahan jika terjadi
            // return redirect()->back()->with('failed', 'update
gagal'." ".$e->getMessage());   //throw $th;
            return redirect()->back()->withErrors(['error' =>
$e->getMessage()]);   //throw $th;
        }
    }
}
```

Lampiran 11

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
```

```
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Lampiran 12

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePasswordResetsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
```

```
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint
$table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
```

Lampiran 13

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateFailedJobsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
```

```
        Schema::create('failed_jobs', function (Blueprint
$table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
}
```

Lampiran 14

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePersonalAccessTokensTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
```

```
    {
        Schema::create('personal_access_tokens', function
(Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('personal_access_tokens');
    }
}
```

Lampiran 15

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateJamaahTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
```

```
    public function up()
    {
        Schema::create('jamaah', function (Blueprint $table) {
            $table->id();
            $table->string('nama',255);
            $table->year('tanggal_lahir');
            $table->text('alamat');
            $table->string('pekerjaan',255);
            $table->string('foto',255);
            $table->string('jenis_kelamin',1);
            $table->string('no_hp',20);
            $table->string('dapukan',255)->nullable();
            $table->boolean('is_pendatang');

            $table->timestamps();
            $table->softDeletes($column = 'deleted_at',
$precision=0);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('jamaah');
    }
}
```

Lampiran 16

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class CreateOrangtuaTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('orangtua', function (Blueprint $table) {
            $table->id();
            $table->string('nama_bapak',255);
            $table->string('pekerjaan_bapak',255);
            $table->string('no_hp_bapak',20);
            $table->string('nama_ibu',255);
            $table->string('pekerjaan_ibu',255);
            $table->string('no_hp_ibu',20);

            $table->timestamps();
            $table->softDeletes($column = 'deleted_at',
$precision=0);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('orangtua');
    }
}
```

Lampiran 17

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class UpdateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('name');
            $table->dropColumn('email');
            $table->dropColumn('email_verified_at');

            $table->string('password',255)->change();

            $table->string('username',255);
            $table->string('role',255);
            $table->dateTime('last_login')->nullable();

            $table->softDeletes($column = 'deleted_at',
$precision=0);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
```

```
            $table->string('password')->change();

            $table->dropColumn('username');
            $table->dropColumn('role');
            $table->dropColumn('last_login');

            $table->dropSoftDeletes();
        });
    }
}
```

Lampiran 18

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddFkUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->unsignedBigInteger('id_jamaah')->nullable();
            $table->foreign('id_jamaah')->references('id')-
>on('jamaah')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
```

```
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropForeign('users_id_jamaah_foreign');
            $table->dropColumn('id_jamaah');
        });
    }
}
```

Lampiran 19

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddFkJamaahTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('jamaah', function (Blueprint $table) {
            $table->unsignedBigInteger('id_bapak')->nullable();
            $table->foreign('id_bapak')->references('id')-
>on('jamaah')->onDelete('cascade');

            $table->unsignedBigInteger('id_ibu')->nullable();
            $table->foreign('id_ibu')->references('id')-
>on('jamaah')->onDelete('cascade');

            $table->unsignedBigInteger('id_suami')->nullable();
            $table->foreign('id_suami')->references('id')-
>on('jamaah')->onDelete('cascade');
```

```
            $table->unsignedBigInteger('id_istri')->nullable();
            $table->foreign('id_istri')->references('id')-
>on('jamaah')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('jamaah', function (Blueprint $table) {
            $table->dropForeign('jamaah_id_bapak_foreign');
            $table->dropColumn('id_bapak');

            $table->dropForeign('jamaah_id_ibu_foreign');
            $table->dropColumn('id_ibu');

            $table->dropForeign('jamaah_id_suami_foreign');
            $table->dropColumn('id_suami');

            $table->dropForeign('jamaah_id_istri_foreign');
            $table->dropColumn('id_istri');
        });
    }
}
```

Lampiran 20

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateJoinOrangtuaJamaahTable extends Migration
```

```
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('join_orangtua_jamaah', function
(Blueprint $table) {
            $table->unsignedBigInteger('id_orangtua')-
>nullable();
            $table->foreign('id_orangtua')->references('id')-
>on('orangtua')->onDelete('cascade');

            $table->unsignedBigInteger('id_jamaah')->nullable();
            $table->foreign('id_jamaah')->references('id')-
>on('jamaah')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('join_orangtua_jamaah');
    }
}
```

Lampiran 21

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
class UpdateUsersTable3 extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            //
            $table->string('foto')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('users', function (Blueprint $table) {
            //
            $table->dropColumn('foto');
        });
    }
}
```

Lampiran 22

```
<?php

namespace App\Models;

use Laravel\Sanctum\HasApiTokens;
use Illuminate\Support\Facades\Storage;
use Illuminate\Notifications\Notifiable;
```

```
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, SoftDeletes;

    protected $table = 'users';
    protected $dates = ['deleted_at'];
    protected $primaryKey = 'id';
    protected $fillable = [
        'username',
        'password',
        'role',
        'last_login',
        'id_jamaah',
        'nama'
    ];

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $guarded = ['id'];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    public function getFotoProfile(){
        $path_foto = $this->foto;
        // 
{{ base64_encode(Storage::get($foto)) }}

        if($path_foto!=null){
            return
"" .base64_encode(Storage::get($path_foto));
```

```
        }else{
            return  asset('/media/avatars/avatar15.jpg');
        }
    }

}
```

Lampiran 23

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Jamaah extends Model
{
    use HasFactory, SoftDeletes;

    protected $table = 'jamaah';
    protected $dates = ['deleted_at'];
    protected $primaryKey = 'id';
    protected $fillable = [
        'nama',
        'tanggal_lahir',
        'alamat',
        'pekerjaan',
        'foto',
        'jenis_kelamin',
        'no_hp',
        'dapukan',
        'is_pendatang',
        'id_bapak',
        'id_ibu',
        'id_suami',
        'id_istri'
    ];
```

```
}
```

Lampiran 24

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Orangtua extends Model
{
    use HasFactory, SoftDeletes;

    protected $table = 'orangtua';
    protected $dates = ['deleted_at'];
    protected $primaryKey = 'id';
    protected $fillable = [
        'nama_bapak',
        'pekerjaan_bapak',
        'no_hp_bapak',
        'nama_ibu',
        'pekerjaan_ibu',
        'no_hp_ibu'
    ];
}
```

Lampiran 25

```
<?php

namespace App\Http\Controllers\Pengurus;

use App\Models\Jamaah;
```

```
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class PengurusController extends Controller
{

    public function index(){

        $get_role_text = \Config::get('constants');
        $get_role = Auth::user()->role;
        $title = 'Dashboard'."-
".$get_role_text['role_text'][$get_role];
        $role = $get_role_text['role_text'][$get_role];
        $nama = Auth::user()->username;

        switch (Auth::user()->role){
            case \Config::get('constants.role.ki'):
                return view('Pengurus.ki.index',
compact("title","role","nama"));
                break;
            case \Config::get('constants.role.wk'):
                return view('Pengurus.wk.index',
compact("title","role","nama"));
                break;
            case \Config::get('constants.role.ku'):
                return view('Pengurus.ku.index',
compact("title","role","nama"));
                break;
            case \Config::get('constants.role.pnb'):
                return view('Pengurus.pnb.index',
compact("title","role","nama"));
                break;
            case \Config::get('constants.role.mt'):
                return view('Pengurus.mt.index',
compact("title","role","nama"));
                break;
            case \Config::get('constants.role.pengabsen'):
                return view('Pengurus.pengabsen.index',
compact("title","role","nama"));
                break;
```

```
            default:
                Auth::logout();
                return redirect()->back()->with('failed','Akun
tidak memiliki akses. Galat: '.$e->getMessage());
                break;
        }
    }

    public function show(){
        $get_role_text = \Config::get('constants');
        $get_role = Auth::user()->role;
        $title = 'Dashboard'."-
".$get_role_text['role_text'][$get_role];
        $role = $get_role_text['role_text'][$get_role];
        $nama = Auth::user()->username;

        $jamaah = Jamaah::all();

        switch (Auth::user()->role){
            case \Config::get('constants.role.ki'):
                return view('Pengurus.ki.data_kelompok',
compact("title","role","nama","data"));
                break;
            default:
                // Auth::logout();
                return redirect()->back()->with('failed','Akun
tidak memiliki akses. Galat: '.$e->getMessage());
                break;
        }
    }

    public function add(){
        $get_role_text = \Config::get('constants');
        $get_role = Auth::user()->role;
        $title = 'Dashboard'."-
".$get_role_text['role_text'][$get_role];
        $role = $get_role_text['role_text'][$get_role];

        return view('Pengurus.ki.add_data',
compact("title","role","nama"));
    }
```

```
    public function store(Request $request){
        try{
            $this->validate($request, [
                'nama'          => 'required',
                'tanggal_lahir' => 'required',
                'alamat'        => 'required',
                'pekerjaan'     => 'required',
                'jenis_kelamin' => 'required',
                'no_hp'         => 'required',
                'is_pendatang'  => 'required',
                'foto'          => 'required|jpg, jpeg, gif,
png|max:2048'
            ]);
            $data = Jamaah::create([
                'nama'          => $request->nama,
                'tanggal_lahir' => $request->tanggal_lahir,
                'alamat'        => $request->alamat,
                'pekerjaan'     => $request->pekerjaan,
                'foto'          => $request->foto,
                'jenis_kelamin' => $request->jenis_kelamin,
                'no_hp'         => $request->no_hp,
                'dapukan'       => $request->dapukan,
                'is_pendatang'  => $request->is_pendatang,
                'id_bapak'      => $request->id_bapak,
                'id_ibu'        => $request->id_ibu,
                'id_suami'      => $request->id_suami,
                'id_istri'      => $request->id_istri
            ]);
            return redirect()->route('ki.data_kelompok')-
>with('success','Sukses Nambah Data!');
        } catch (\Exception $e){
            return redirect()->back()->with('failed','Gagal
Nambahin Data!, terjadi error: '.$e->getMessage());
        }
    }

    public function edit($id){
        $get_role_text = \Config::get('constants');
        $get_role = Auth::user()->role;
        $title   = 'Dashboard'."-
```

```
".$get_role_text['role_text'][$get_role];
        $role = $get_role_text['role_text'][$get_role];
        $nama       = Auth::user()->username;

        $data = Jamaah::find($id);

        return view('Pengurus.ki.edit_data',
compact("title","role","nama","data"));
    }

    public function update($id, Request $request){
        try{
            $this->validate($request,[
                'nama'          => 'required',
                'tanggal_lahir' => 'required',
                'alamat'        => 'required',
                'pekerjaan'     => 'required',
                'jenis_kelamin' => 'required',
                'no_hp'         => 'required',
                'is_pendatang'  => 'required',
                'foto'          => 'required'
            ]);
            $data = Jamaah::find($id);

            $data->update([
                'nama'          => $request->nama,
                'tanggal_lahir' => $request->tanggal_lahir,
                'alamat'        => $request->alamat,
                'pekerjaan'     => $request->pekerjaan,
                'foto'          => $request->foto,
                'jenis_kelamin' => $request->jenis_kelamin,
                'no_hp'         => $request->no_hp,
                'dapukan'       => $request->dapukan,
                'is_pendatang'  => $request->is_pendatang,
                'id_bapak'      => $request->id_bapak,
                'id_ibu'        => $request->id_ibu,
                'id_suami'      => $request->id_suami,
                'id_istri'      => $request->id_istri
            ]);

            return redirect()->route('ki.data_kelompok')-
```

```
>with('success','Sukses Update Data!');
        } catch (\Exception $e){
            return redirect()->back()->with('failed','Gagal
Update Data!, terjadi error: '.$e->getMessage());
        }
    }

    public function delete($id){
        try{
            $data = Jamaah::find($id);
            $data->delete();

            return response()->json([
                'status'    => 'success',
                'message'   => 'Sukses Delete Data!'
            ]);
        } catch(\Exception $e){
            return response()->json([
                'status'    => 'failed',
                'message'   => 'Gagal Delete Data!, terjadi
error: '.$e->getMessage()
            ]);
        }
    }
}
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS I

Nama	: Bayu Surya Bawono
Tempat, Tanggal Lahir	: Bima, 23 Januari 2001
Jenis Kelamin	: Laki-laki
Telepon	: +6285333096854
Email	: bayusb313@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2018
Semester	: 10 (Sepuluh)

BIODATA PENULIS II

Nama	: Ryan Rasyid Azizi
Tempat, Tanggal Lahir	: Jakarta, 28 Agustus 2000
Jenis Kelamin	: Laki-Laki
Telepon	: +6281284636650
Email	: ryanrasyidaz@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2018
Semester	: 10 (Sepuluh)