KERJA PRAKTIK - EF234603

Perancangan dan Implementasi Front-end dan Back-end Boarding School System (BSS) Berbasis Aplikasi Mobile

Nurul Fikri Boarding School Bogor
Jl. Jami Atas, Sukaluyu, Kec. Tamansari, Kabupaten Bogor, Jawa Barat 16610.

Periode: 28 Agustus 2023 - 28 November 2023

Oleh:

Maisan Auliya	5025201137
Samuel	5025201187

Pembimbing Departemen
Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.
Pembimbing Lapangan
Khoirul Mustofa, S.Kom.

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

KERJA PRAKTIK – EF234603

Perancangan dan Implementasi Front-end dan Back-end Boarding School System (BSS) Berbasis Aplikasi Mobile

Nurul Fikri Boarding School Bogor
Jl. Jami Atas, Sukaluyu, Kec. Tamansari, Kabupaten Bogor, Jawa Barat 16610.

Periode: 28 Agustus 2023 – 28 November 2023

Oleh:

Maisan Auliya	**5025201137**
Samuel	**5025201187**

Pembimbing Departemen
Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.
Pembimbing Lapangan
Khoirul Mustofa, S.Kom.

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

LEMBAR PENGESAHAN
KERJA PRAKTIK

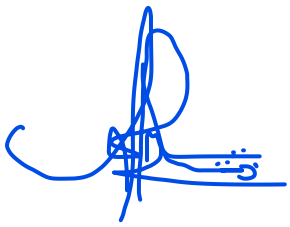
Perancangan dan Implementasi Front-end dan Back-end Boarding School System (BSS) Berbasis Aplikasi Mobile

Oleh:

Maisan Auliya	5025501137
Samuel	5025501187

Disetujui oleh Pembimbing Kerja Praktik:

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



(Pembimbing Departemen)

2. Khoirul Mustofa, S.Kom.

(Pembimbing Lapangan)

Surabaya, 20 Desember 2023

Perancangan dan Implementasi Front-end dan Back-end Boarding School System (BSS) Berbasis Aplikasi Mobile

Nama Mahasiswa	: Maisan Auliya (5025201137)
	Samuel (5025201187)
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Adhatus Solichah Ahmadiyah, S.Kom., M.Sc.
Pembimbing Lapangan	: Khoirul Mustofa, S.Kom.

ABSTRAK

Nurul Fikri Boarding School (NFBS) Bogor ingin membawa sistem Boarding School System (BSS) yang sudah dibuat dalam bentuk web ke dalam aplikasi mobile agar bisa diakses oleh smartphone dengan lebih praktis.

Aplikasi mobile merupakan portal atau tempat yang menyediakan layanan permintaan perizinan para santri untuk meninggalkan sekolah diluar jam yang telah ditentukan. Aplikasi mobile ini juga tentunya memudahkan alur perizinan untuk santri, dimulai dari melakukan pengajuan perizinan hingga persetujuan perizinan dari wali kelas dan juga kepala sekolah terkait. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman mobile Dart dengan framework Flutter. Untuk API dibangun dengan TypeScript dan MariaDB sebagai DBMS. Aplikasi ini diharapkan dapat mempermudah perizinan para santri, juga menjadi solusi praktis untuk permintaan perizinan tanpa perlu membuka browser terlebih dahulu.

Kata kunci : NFBS, Aplikasi Mobile, Boarding School System

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa karena atas berkat limpahan rahmat dan lindungan-Nya kami dapat melaksanakan salah satu kewajiban sebagai mahasiswa Departemen Informatika, yakni Kerja Praktik (KP).

Kami menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan KP maupun penyusunan buku laporan ini. Namun, kami berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Kami mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan KP ini.

Melalui laporan ini, kami juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan KP hingga penyusunan laporan. Orang-orang tersebut antara lain adalah:

1. Orang tua penulis.
2. Ibu Adhatus Solichah Ahmadiyah, S.Kom., M.Sc., selaku dosen pembimbing KP.
3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., selaku koordinator KP.
4. Bapak Khoirul Mustofa, S.Kom., dan Tim LRC NFBS Bogor selaku pembimbing lapangan kami di NFBS Bogor.
5. Teman-teman penulis yang senantiasa memberikan semangat dan dukungan kepada penulis selama melaksanakan KP.

Surabaya, 20 Desember 2023
Maisan Auliya
Samuel

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1. Latar Belakang

Digitalisasi telah menjadi kekuatan revolusioner dalam transformasi global, mempercepat perubahan dalam cara kita bekerja, berkomunikasi, dan mengakses informasi. Dengan cepatnya adopsi teknologi digital, organisasi dan individu dapat mengoptimalkan proses, meningkatkan efisiensi, dan memanfaatkan peluang inovatif. Digitalisasi tidak hanya mempermudah konektivitas global, tetapi juga menciptakan ekosistem yang dinamis untuk kolaborasi, pengembangan, dan pertumbuhan. Sebagai pendorong utama perubahan, digitalisasi telah membuka pintu bagi kemajuan signifikan di berbagai sektor, merangkul era dimana teknologi dapat merangkul kebutuhan sehari-hari.

Nurul Fikri Boarding School (NFBS) melihat potensi tersebut lalu mengimplementasikan sebuah Boarding School System (BSS) agar orang tua santri maupun para santri dapat melihat dan mengetahui aktivitas yang dilakukan di sekolah. Ini juga menjadi sebuah bentuk transparansi dan sarana transformasi digital bagi NFBS.

Untuk saat ini, BSS diimplementasikan pada aplikasi web. Beberapa layanan yang disediakan antara lain perizinan dan melihat berita serta acara yang akan dibuat. Layanan ini sudah cukup membantu para orang tua sampai pada tahap dimana mereka ingin BSS diimplementasikan pada aplikasi mobile. Selain itu NFBS juga ingin membuat fitur baru yaitu prediksi hafalan para santri yang akan diimplementasikan ke aplikasi mobile mereka.

Maka untuk menghadapi masalah tersebut, solusi yang ditawarkan adalah dengan membuat sebuah Aplikasi BSS yang berbasis mobile untuk melakukan perizinan, melihat berita dan acara terbaru yang akan dihelat serta prediksi hafalan santri. Diharapkan dengan aplikasi mobile ini, para orang tua dapat mengakses BSS dengan lebih mudah dan praktis.

1.2. Tujuan

Tujuan KP ini adalah untuk menyelesaikan kewajiban kuliah kerja praktik di Institut Teknologi Sepuluh Nopember dengan beban 2 SKS. Selain itu juga untuk membantu *management* perizinan dan hafalan santri di Nurul Fikri Boarding School Bogor

Tujuan dari pengimplementasian aplikasi tersebut antara lain:

1. Proses bisnis menjadi lebih mudah diakses pada aplikasi berbasis mobile dibandingkan dengan aplikasi berbasis web yang sudah ada.
2. Meningkatkan pemanfaatan digitalisasi dalam sistem pendidikan Nurul Fikri Boarding School Bogor.
3. Menghemat waktu untuk proses perizinan santri dibandingkan harus berkunjung ke web.

1.3. Manfaat

Manfaat yang dapat diperoleh dengan adanya Aplikasi BSS Mobile antara lain adalah:

1. Mempermudah pihak Nurul Fikri Boarding School Bogor dalam *management* semua hal yang berkaitan dengan santri dari instansi tersebut, dalam kasus kerja praktik ini yaitu dalam bidang perizinan dan hafalan santri.
2. Membantu dalam digitalisasi sistem pendidikan pada instansi terkait.

1.4. Rumusan Masalah

Berikut ini rumusan masalah pada KP pembuatan Aplikasi BSS:

1. Bagaimana solusi efektif dalam pembuatan aplikasi yang dapat mudah dikembangkan bagi pihak Nurul Fikri Boarding School Bogor?
2. Bagaimana supaya dapat menghasilkan aplikasi yang mudah digunakan oleh orang tua santri dan sivitas pendidik Nurul Fikri Boarding School Bogor?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut :

Lokasi : Nurul Fikri Boarding School Bogor
Alamat: Jl. Jami Atas, Sukaluyu, Kec. Tamansari, Kabupaten Bogor, Jawa Barat 16610.
Waktu : 28 Agustus 2023 - 28 November 2023

1.6. Metodologi Kerja Praktik

Tahapan pengerjaan kerja praktik dapat dijabarkan sebagai berikut:

1. Perumusan Masalah

Untuk mengetahui permasalahan apa yang harus diselesaikan, diberikan penjelasan mengenai alasan mengapa aplikasi ini dibutuhkan. Dijelaskan pula secara rinci mengenai bagaimana alur sistem itu akan berjalan. Penjelasan mengenai hal ini dijelaskan oleh pembimbing lapangan KP. Dari penjelasannya dihasilkan catatan-catatan penting mengenai gambaran sistem berbasis mobile ini akan dibuat. Dengan begitu dapat diputuskan untuk membuat sistem berbasis web dengan menggunakan bahasa pemrograman Flutter untuk mobile, Typescript untuk API dan Python untuk machine learning serta menggunakan database Mariadb.

2. Studi Literatur

Setelah ditentukan database, bahasa pemrograman, serta tools tambahan yang akan digunakan, dilakukan studi literatur mengenai cara implementasinya dalam membangun sistem sesuai yang dibutuhkan. Pada tahap ini dilakukan proses pencarian, pembelajaran, pengumpulan dan pemahaman informasi serta literatur yang berkaitan untuk membantu dalam implementasi aplikasi ini. Informasi bisa didapat dari internet untuk istilah-istilah umum yang digunakan dalam mengimplementasikan suatu sistem informasi.

3. Analisis dan Perancangan Sistem

Langkah ini meliputi penjelasan awal tentang sistem. Bagaimana cara kerja sistem dengan skenario tertentu. Dari penjelasan awal telah didapatkan beberapa kebutuhan fungsional dan non-fungsional secara garis besar. Kemudian dilanjutkan dengan memperjelas dan menspesifikkan kebutuhan- kebutuhan tersebut. Maka dibuat sebuah diagram kasus penggunaan yang mewakili skenario- skenario untuk penggunaan sistem Aplikasi BSS Mobile.

4. Implementasi Sistem

Implementasi sistem didasarkan pada perancangan dan analisis sebelumnya. Kasus penggunaan dan penentuan tools juga turut mendasari pengimplementasian sistem ini. Pada tahap ini setidaknya ada dua pekerjaan utama yang dilakukan, yakni desain mobile pada tampilan atau front-end, desain fungsi-fungsi yang bekerja dalam sistem atau dikenal sebagai back-end, dan juga pembuatan model yang akan digunakan untuk machine learning. Pengerjaan dilakukan dengan progres setiap hari, dengan setiap harinya menargetkan perkembangan dari hari sebelumnya. Progres penyelesaian aplikasi terus dipantau oleh tim LRC Nurul Fikri Boarding School Bogor sebagai pelopor Aplikasi BSS Mobile.

5. Pengujian dan Evaluasi

Pengujian dilakukan dengan menguji fitur-fitur yang telah dibuat. Kesesuaian sistem dengan kebutuhan akan menentukan keberhasilan dalam pengujian. Hal ini akan menghasilkan hasil evaluasi apakah sistem sudah sesuai dengan tujuan dan kebutuhan atau belum.

1.7. Sistematika Laporan

Laporan KP ini terdiri dari tujuh bab dengan rincian sebagai berikut:

1. Bab I Pendahuluan

Pada bab ini dijelaskan tentang latar belakang permasalahan, tujuan, waktu pelaksanaan serta sistematika pengerjaan KP dan juga penulisan laporan KP.

2. Bab II Profil Perusahaan

Pada bab ini, dijelaskan secara rinci tentang profil perusahaan tempat melaksanakan KP, yakni DPTSI (Direktorat Pengembangan Teknologi dan Sistem Informasi).

3. Bab III Tinjauan Pustaka

Pada bab ini, dijelaskan mengenai tinjauan pustaka dan literatur yang digunakan dalam penyelesaian KP.

4. Bab IV Analisis dan Perancangan Sistem

Pada bab ini, dijelaskan hasil pembelajaran atau analisis terhadap apa saya yang diperlukan dan harus diperhatikan dalam pengembangan aplikasi yang dikerjakan selama KP.

5. Bab V Implementasi Sistem

Pada bab ini, berisi penjelasan tahap-tahap yang dilakukan untuk proses implementasi aplikasi.

6. Bab VI Pengujian dan Evaluasi

Pada bab ini, dijelaskan tentang hasil pengujian dan evaluasi dari sistem yang telah dikembangkan selama pelaksanaan KP.

7. Bab VII Kesimpulan dan Saran

Pada bab ini, dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan KP

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1 Sejarah Perusahaan

Cikal bakal NFBS berawal dari bimbingan belajar BIMBINGAN BELAJAR NURUL FIKRI BINA PRESTASI, yang cabangnya tersebar di Jabodetabek. Pada 19 Desember 1999, Yayasan Pesantren Ibnu Salam dibentuk. Seorang pewakaf yakni Malik Abdus Salam memberikan tanahnya untuk pembangunan sekolah. Pada tahun ajaran pertamanya, NFBS Serang membuka jenjang SMP yang dihadiri 48 siswa dengan tenaga pengajar sebanyak 10 orang guru. Alumni angkatan pertama ini diwisuda pada 2002.

Nurul Fikri Boarding School Bogor merupakan cabang ketiga dari boarding school yang dibuat oleh Nurul Fikri. Cabang pertama merupakan Nurul Fikri Boarding School Serang yang berdiri pada 19 Desember 1999. Beralamat di Desa Bantarwaru, Kec. Cinangka, Kab. Serang – Banten. Terletak 8,5 Km dari Pantai Anyer. Pada tahun 2010, berdirilah cabang kedua yaitu Nurul Fikri Boarding School Lembang. Cabang ini merupakan lembaga pendidikan dibawah Yayasan Pendidikan Islam Madani Lembang. Sesuai dengan namanya, cabang ini berada di Lembang, Bandung Barat. Terakhir dan terbaru, yaitu Nurul Fikri Boarding School Bogor. Cabang ini dibangun diatas tanah sebesar 7.6 hektar yang berada pada Tamansari, Bogor.

2.2 Profil Nurul Fikri Boarding School Bogor

Nurul Fikri Boarding School Bogor dibangun di atas lahan seluas 7,6 hektar yang mencakup unit sekolah menengah pertama dan menengah atas. Dikelilingi oleh pemandangan indah Bogor, Sekolah Asrama Nurul Fikri Bogor memberikan prioritas pada kenyamanan belajar bagi siswanya. Dengan siswa yang berasal dari berbagai wilayah di Indonesia bahkan dari luar negeri dengan keunikan mereka sendiri, Sekolah Asrama Nurul Fikri Bogor menjadi titik pertemuan multikultural yang memungkinkannya menghasilkan siswa dengan tingkat modal kepemimpinan yang tinggi [1].

2.3 Visi dan Misi Perusahaan

Visi : Menjadi Sekolah Berasrama Rujukan Dalam Membina Generasi Pemimpin Masa Depan.

Misi :

1. Menciptakan lingkungan belajar yang islami.
2. Mengembangkan model pembelajaran yang efektif dan kondusif.
3. Membina dan Mengembangkan Karakter Kepemimpinan.
4. Menyelenggarakan Pendidikan Menengah Untuk Pembentukan Karakter Siswa Yang Sholeh, Muslih, Cerdas, Mandiri, dan Terampil.
5. Mengoptimalkan Peran Serta Orang Tua, Alumni, Masyarakat, dan Pemerintah [2]

BAB III
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai dasar teori yang digunakan selama proses pengerjaan dan pengembangan aplikasi.

3.1. *Cross-platform Application*

Aplikasi berbasis c*ross-platform* adalah sebuah teknologi yang memungkinkan sebuah aplikasi dapat dijalankan di *platform* yang berbeda. Dengan pengembangan *cross-platform* ini dapat memudahkan *developer* karena pengembangannya cepat dan hemat biaya [3].

3.1.1. *Dart*

Dart adalah bahasa pemrograman yang bersifat open-source. Ini adalah bahasa pemrograman yang sepenuhnya berorientasi objek dengan sintaksis gaya C. Dart awalnya dikembangkan oleh Google pada tahun 2011 dan kemudian disetujui sebagai standar oleh ECMA. Dart adalah bahasa yang dioptimalkan untuk mengembangkan aplikasi *cross-platform*. DART digunakan untuk pengembangan aplikasi *mobile* and website sekaligus [4].

3.1.2. *Flutter*

Flutter adalah SDK open-source untuk mengembangkan aplikasi mobile yang memiliki kinerja tinggi dan lebih dapat diandalkan untuk sistem operasi seperti iOS dan Android. Fitur signifikan dari Flutter adalah kompilasi *just-in-time* yang mengeksekusi kode komputer yang mencakup kompilasi selama eksekusi program pada saat *runtime* daripada sebelum eksekusi [5].

3.1.3. *Android Studio*

Android studio adalah Integrated Development Environment (IDE) resmi untuk pengembangan aplikasi Android, yang didasarkan dari IntelliJ IDEA sebuah IDE popular yang dikembangkan oleh Jetbrains untuk pemrograman Java dan Kotlin. Android studio menawarkan fitur yang mampu meningkatkan produktivitas dalam pengembangan aplikasi [6].

3.2. Pemrograman Backend

Backend merujuk pada bagian dari sebuah aplikasi atau sistem yang berfokus pada pemrosesan data, logika bisnis, dan penyimpanan informasi, biasanya terletak di sisi server dan bertanggung jawab untuk mendukung fungsi-fungsi yang tidak terlihat oleh pengguna [7].

3.2.1. *TypeScript*

TypeScript adalah bahasa pemrograman yang berbasis pada JavaScript dan memiliki tipe data yang kuat, memberikan alat bantu yang lebih baik dalam skala apapun. TypeScript menambahkan sintaksis tambahan ke JavaScript untuk mendukung integrasi yang lebih erat dengan editor Anda, sehingga memungkinkan deteksi kesalahan secara dini saat Anda sedang mengembangkan. Ini memastikan pengalaman penulisan kode yang lebih

kokoh dan efisien, memberikan kekuatan kepada para pengembang untuk membangun aplikasi yang dapat diandalkan dan minim kesalahan [8].

3.2.2. *ExpressJS*

ExpressJS adalah kerangka kerja aplikasi web untuk NodeJS yang menyederhanakan proses pembuatan aplikasi web dan API. Framework ini dirancang agar minimal dan fleksibel, sehingga pengembang dapat menyusun kode sesuai keinginan masing - masing. ExpressJS menyediakan serangkaian fitur yang kuat untuk aplikasi web dan seluler, termasuk middleware untuk menangani permintaan dan respons, mesin templating untuk membuat konten dinamis pada halaman web, dan utilitas untuk merender respons HTTP dinamis. Ini adalah kerangka kerja backend yang paling populer untuk NodeJS dan banyak digunakan untuk mengembangkan API [9].

3.3. Machine Learning

3.3.1. Facebook Prophet

Facebook Prophet adalah sebuah perangkat lunak open-source yang dikembangkan oleh tim Facebook untuk memprediksi data deret waktu (time series data). Algoritma ini dirancang untuk membuat prediksi dengan mudah dan efisien, terutama untuk data deret waktu yang memiliki karakteristik khusus seperti musiman, tren, dan hari libur. Facebook Prophet bekerja paling baik dengan deret waktu yang memiliki efek musiman yang kuat dan beberapa musim data historis. Prophet tahan terhadap data yang hilang dan pergeseran dalam tren, dan umumnya dapat menangani outlier dengan baik [10].

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem

Pada bab ini akan dijelaskan mengenai tahapan dalam membangun Aplikasi BSS Mobile berupa analisis dari sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan fungsional.

4.1.1. Definisi Umum Aplikasi

Secara umum, Aplikasi BSS Mobile merupakan sistem berbasis mobile yang digunakan untuk me-*manage* hal-hal yang berkaitan dengan digitalisasi pada NFBS Bogor. Aplikasi ini merupakan pengembangan berikutnya dari sistem berbasis web yang telah dikembangkan oleh tim dari NFBS Bogor itu sendiri. Pada kegiatan kerja praktek ini, penulis hanya membuat dua fitur utama, yaitu perizinan dan hafalan Al-quran santri dari NFBS Bogor. Adapun pengguna aplikasi ini didefinisikan sebagai berikut:

- Wali Kelas dan Kepala Sekolah yang dapat memberikan izin untuk sistem perizinan santri.
- Wali Santri, Orang tua dari santri NFBS Bogor yang dapat meminta izin untuk anak-anaknya dan melakukan *controlling* terhadap hafalan Al-Qur'an nya.

4.1.2. Analisis Kebutuhan

Dalam aplikasi ini, terdapat fungsi-fungsi yang harus dipenuhi oleh sistem. Kebutuhan ini terbagi ke dalam dua jenis, yakni kebutuhan fungsional dan kebutuhan non-fungsional.

4.1.2.1. Kebutuhan Fungsional

Kebutuhan fungsional pada aplikasi ini menjelaskan bagaimana sistem itu bekerja. Kebutuhan fungsional dari Aplikasi BSS Mobile dijelaskan pada Tabel 4.1.

Tabel 4. 1 Kebutuhan Fungsional BSS Mobile

Kode Kebutuhan	Deskripsi Kebutuhan
F-001	Melakukan Login
F-002	Memilih atau Mengganti Santri
F-003	Melakukan Perizinan Santri
F-004	Mengisi Formulir Pengajuan Perizinan
F-005	Melihat Daftar Perizinan yang Sedang Berlangsung
F-006	Melihat Detail Perizinan
F-007	Melihat Daftar History Perizinan

F-008	Melihat Detail History Perizinan
F-009	Memberikan Approval Perizinan
F-010	Menampilkan Notifikasi Perizinan yang Masuk
F-011	Menampilkan Daftar Notifikasi
F-012	Melihat Prediksi Hafalan Santri
F-013	Menampilkan Daftar News
F-014	Melihat Detail News
F-015	Menampilkan Daftar Event
F-016	Melihat Detail Event

Pada tabel 4.1 terdapat kebutuhan fungsional BSS Mobile yang dapat diakses oleh aktor dan memiliki fitur yang berbeda bergantung pada *roles* dari aktor.

4.1.2.2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah kebutuhan pengguna untuk mendefinisikan bagaimana batasan dan karakteristik dari sebuah sistem yang dibangun. Kebutuhan non-fungsional dari Aplikasi BSS Mobile terdapat pada Tabel 4.2.

Tabel 4. 2 Kebutuhan Non-Fungsional BSS Mobile

Kode Kebutuhan	Deskripsi Kebutuhan
NF-001	Sistem dapat diakses oleh pengguna
NF-002	Sistem dapat diakses pada iOS maupun Android
NF-003	Sistem memiliki tampilan antarmuka yang mudah dipahami
NF-004	Sistem dapat memastikan bahwa data yang digunakan dalam sistem harus terlindung dari akses yang tidak berwenang

Pada tabel 4.2 terdapat kebutuhan non-fungsional BSS Mobile yang meliputi akses pengguna, OS yang kompetibel, antarmuka yang mudah dipahami, dan keamanan data pengguna.

4.2. Scope Pengerjaan Aplikasi

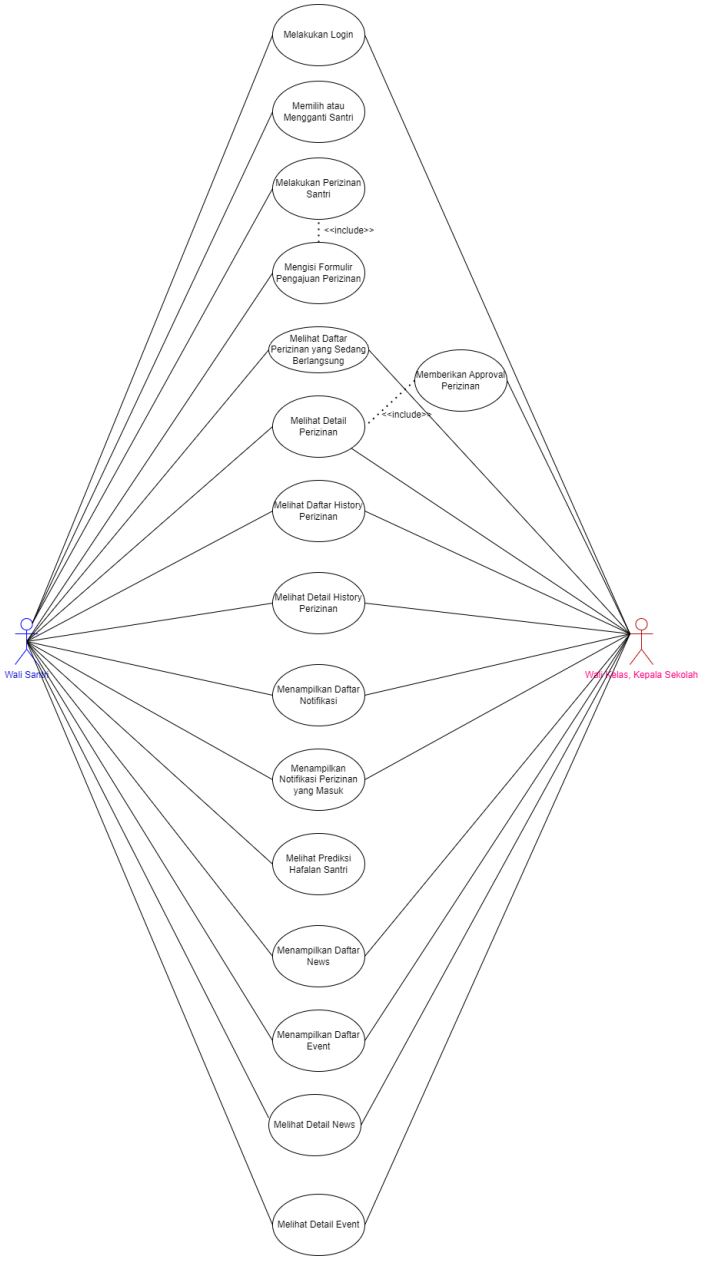
Pada pengerjaan aplikasi ini terdapat dua kelompok yang bekerja sama. Kelompok 1 merupakan Maisan Auliya dan Samuel, sedangkan kelompok 2 merupakan Gabriel Solomon Sitanggang dan William Zefanya. Untuk pembagian pengerjaan aplikasi ini dapat dilihat dalam tabel 4.3 di bawah ini.

Table 4. 3. Scope Pengerjaan Aplikasi

Kelompok	Kode Kebutuhan	Kebutuhan Aplikasi
1	F-001	Melakukan Login
	F-002	Memilih atau Mengganti Santri
	F-003	Melakukan Perizinan Santri
	F-004	Mengisi Formulir Pengajuan Perizinan
	F-005	Melihat Daftar Perizinan yang Sedang Berlangsung
	F-006	Melihat Detail Perizinan
	F-007	Melihat Daftar History Perizinan
	F-008	Melihat Detail History Perizinan
	F-009	Memberikan Approval Perizinan
	F-010	Menampilkan Notifikasi Perizinan yang Masuk
	F-011	Menampilkan Daftar Notifikasi
2	F-012	Melihat Prediksi Hafalan Santri
	F-013	Menampilkan Daftar News
	F-014	Melihat Detail News
	F-015	Menampilkan Daftar Event
	F-016	Melihat Detail Event

4.3. Diagram Kasus Penggunaan

Pembahasan dengan pembimbing lapangan tentang fitur-fitur yang perlu ada dalam Aplikasi BSS Mobile menghasilkan beberapa fitur yang dijadikan diagram kasus penggunaan (Use Case Diagram) sehingga memudahkan untuk dipahami. Use Case Diagram yang telah dibuat dapat dilihat pada Gambar 4.1.



Gambar 4. 1 Diagram Use Case Aplikasi BSS Mobile

Pada Gambar 4.1 terdapat 2 *roles* yang masing-masing memiliki fitur yang dapat diakses secara berbeda. Wali Santri, Wali Kelas, dan Kepala Sekolah dapat mengakses fitur *login*, melihat perizinan yang sedang berlangsung, melihat detail perizinan, melihat *history* perizinan beserta detail *history* perizinan, menampilkan notifikasi perizinan yang telah masuk, menampilkan berita terbaru beserta dengan detail beritanya, menampilkan daftar *event* beserta dengan detail *event*. Namun Wali Santri memiliki fitur lebih banyak daripada Wali Kelas dan Kepala Sekolah seperti dapat melakukan login, memilih santri yang akan diakses, melakukan perizinan santri yang sudah di *submit* oleh santri dan mengisi formulir yang tersedia, dan melihat prediksi hafalan santri.

4.4. Spesifikasi Kasus Penggunaan

4.3.1. Melakukan Login

Tabel 4.3 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melakukan Login.

Tabel 4. 4 Table Use Case BSS Mobile Melakukan Login

Nama	Melakukan login
Kode	UC001
Deskripsi	Aktor dapat masuk ke akun sesuai role
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘log in’ setelah mengisi username dan password pada halaman login BSS Mobile
Aktor	Wali Santri, Wali Kelas, Kepala Sekolah
Kondisi Awal	Form login ditampilkan
Kondisi Akhir	Aktor dapat melakukan kegiatan pada sistem sesuai kewenangannya
Alur Kejadian Secara Normal	1. Aktor mengisi form login 2. Aktor menekan tombol ‘Log in’ 3. Sistem mencocokkan data login dengan database 4. Sistem menampilkan halaman utama
Alur Kejadian Alternatif	1. a Aktor tidak mengisi formulir dengan lengkap 1. Sistem memberi peringatan bahwa kolom harus diisi. 2. Kembali ke Alur Normal nomor 2. 2. a Data yang diinputkan tidak cocok dengan basis data 1. Sistem memberi peringatan bahwa email atau password salah. 2. Kembali ke Alur Normal nomor 4.
Pengecualian	-

4.3.2. Memilih atau Mengganti Santri

Tabel 4.4 berikut merupakan tabel use case dari Aplikasi BSS Mobile Memilih atau Mengganti Santri.

Tabel 4. 5 Tabel Use Case BSS Mobile Memilih atau Mengganti Santri

Nama	Memilih atau Mengganti Santri
Kode	UC002
Deskripsi	Aktor dapat memilih current user (santri) berdasarkan banyaknya anak Wali Santri
Tipe	Fungsional
Pemicu	Aktor menekan data diri santri di halaman utama
Aktor	Wali Santri
Kondisi Awal	Aktor ingin mengganti current user (santri)
Kondisi Akhir	Aktor dapat mengganti current user (santri)
Alur Kejadian Secara Normal	1. Aktor menekan data diri santri di halaman utama 2. Sistem akan menampilkan card berisi anak-anak dari wali santri
Alur Kejadian Alternatif	1. Aktor menekan data diri santri pada halaman utama 2. Sistem akan menampilkan card berisi anak-anak dari wali santri 3. Aktor menekan salah satu santri (selain current user) 4. Sistem menampilkan dialog berisi konfirmasi persetujuan penggantian current user. 5. Apabila aktor menekan tombol ‘Yes’ maka user akan berganti dan card akan tertutup, jika aktor menekan tombol ‘No’ maka kembali ke alur nomor 2.
Pengecualian	-

4.3.3. Melakukan Perizinan Santri

Tabel 4.5 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melakukan Perizinan Santri.

Tabel 4. 6 Tabel Use Case BSS Mobile Melakukan Perizinan Santri

Nama	Melakukan Perizinan Santri
Kode	UC003
Deskripsi	Wali Santri dapat melakukan perizinan untuk anaknya (santri)
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘Permission’ di bagian apps, card dibawah profile pada halaman utama
Aktor	Wali Santri
Kondisi Awal	Aktor ingin melakukan perizinan
Kondisi Akhir	Aktor dapat melakukan perizinan
Alur Kejadian Secara Normal	1. Aktor menekan tombol ‘permission’ di bagian apps 2. Sistem menampilkan menu ‘Permission’ 3. Aktor harus mengisi form
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.4. Mengisi Formulir Pengajuan Perizinan

Tabel 4.6 berikut merupakan tabel use case dari dari Aplikasi BSS Mobile Mengisi Formulir Pengajuan Perizinan.

Tabel 4. 7 Tabel Use Case BSS Mobile Mengisi Formulir Pengajuan Perizinan

Nama	Mengisi formulir pengajuan peminjaman lisensi
Kode	UC004
Deskripsi	Aktor dapat mengisi formulir pengajuan perizinan
Tipe	Fungsional

Pemicu	Aktor memasuki halaman ‘permission’
Aktor	Wali santri
Kondisi Awal	Aktor ingin mengajukan data-data perizinan
Kondisi Akhir	Data perizinan santri tersimpan pada basis data
Alur Kejadian Secara Normal	1. Aktor mengisi formulir pengajuan perizinan secara lengkap 2. Aktor dapat menekan tombol ‘Submit’ 3. Sistem akan menyimpan data yang telah diisi pada basis data 4. Sistem menampilkan Pop-up pengajuan perizinan berhasil.
Alur Kejadian Alternatif	1. Aktor mengisi formulir pengajuan perizinan dengan tidak lengkap atau ada data yang kurang 2. Aktor dapat menekan tombol ‘Submit’ 3. Sistem menampilkan Pop-up untuk melengkapi data. 4. Aktor mengisi formulir kembali.
Pengecualian	-

4.3.5. Melihat Daftar Perizinan yang Sedang Berlangsung

Tabel 4.7 berikut merupakan tabel use case dari dari Aplikasi BSS Mobile Melihat daftar perizinan yang sedang berlangsung.

Tabel 4. 8 Tabel Use Case BSS Mobile Melihat Daftar Perizinan

Nama	Melihat Daftar Perizinan yang Sedang Berlangsung
Kode	UC005
Deskripsi	Aktor dapat melihat daftar perizinan yang sedang berlangsung
Tipe	Fungsional
Pemicu	Aktor memasuki halaman ‘Permission’
Aktor	Wali kelas dan Kepala sekolah

Kondisi Awal	Aktor ingin melihat daftar perizinan yang sedang berlangsung
Kondisi Akhir	Daftar perizinan berhasil ditampilkan dari basis data.
Alur Kejadian Secara Normal	1. Aktor membuka halaman ‘Permission’ 2. Sistem menampilkan daftar perizinan dari basis data
Alur Kejadian Alternatif	1. Sistem menampilkan daftar perizinan dari basis data dan ternyata data nya kosong atau tidak ada. 2. Sistem menampilkan peringatan bahwa data kosong.
Pengecualian	1. Terdapat gangguan pada jaringan aktor. 2. Sistem tidak bisa mendapatkan data dari basis data. 3. Sistem menampilkan peringatan bahwa terdapat error dengan alasan khusus. 4. Aktor melakukan refresh atau pengecekan kembali untuk kembali normal.

4.3.6. Melihat Detail Perizinan

Tabel 4.8 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Detail Perizinan.

Tabel 4. 9 Tabel Use Case BSS Mobile Melihat Detail Perizinan

Nama	Melihat history peminjaman lisensi
Kode	UC006
Deskripsi	Aktor dapat melihat detail perizinan santri
Tipe	Fungsional
Pemicu	Aktor menekan salah satu item dari daftar perizinan
Aktor	wali kelas dan kepala sekolah

Kondisi Awal	Aktor ingin melihat detail perizinan santri
Kondisi Akhir	Aktor dapat melihat detail perizinan santri
Alur Kejadian Secara Normal	1. Aktor menekan salah satu item dari daftar perizinan yang ditampilkan sistem 2. Sistem akan menampilkan halaman detail khusus dari item yang dipilih.
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.7. Melihat Daftar History Perizinan

Tabel 4.9 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Daftar History Perizinan

Tabel 4. 10 Tabel Use Case BSS Mobile Melihat Daftar History Perizinan

Nama	Melihat Daftar Peminjaman Lisensi yang Aktif
Kode	UC007
Deskripsi	Aktor dapat melihat daftar history perizinan
Tipe	Fungsional
Pemicu	Aktor menekan icon ‘history’ pada kanan atas *appbar* di halaman ‘Permission’.
Aktor	wali santri, wali kelas, kepala sekolah
Kondisi Awal	Aktor ingin melihat daftar history perizinan
Kondisi Akhir	Aktor dapat melihat daftar history perizinan

Alur Kejadian Secara Normal	1. Aktor membuka halaman ‘Permission History’ 2. Sistem menampilkan daftar riwayat perizinan dari basis data
Alur Kejadian Alternatif	1. Sistem menampilkan daftar riwayat perizinan dari basis data dan ternyata data nya kosong atau tidak ada. 2. Sistem menampilkan peringatan bahwa data kosong.
Pengecualian	-

4.3.8. Melihat Detail History Perizinan

Tabel 4.10 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Detail history perizinan.

Tabel 4. 11 Tabel Use Case BSS Mobile Melihat Detail History Perizinan

Nama	Melihat Detail History Perizinan
Kode	UC008
Deskripsi	Aktor dapat melihat detail dari history perizinan
Tipe	Fungsional
Pemicu	Aktor menekan salah satu item dari daftar yang ditampilkan sistem.
Aktor	wali santri, wali kelas, kepala sekolah
Kondisi Awal	Aktor ingin melihat detail dari riwayat perizinan
Kondisi Akhir	Aktor dapat melihat detail dari riwayat perizinan
Alur Kejadian Secara Normal	1. Aktor menekan salah satu item dari daftar perizinan yang ditampilkan sistem 2. Sistem akan menampilkan halaman detail khusus dari item yang dipilih.
Alur Kejadian Alternatif	-

Pengecualian	-

4.3.9. Memberikan Approval Perizinan

Tabel 4.11 berikut merupakan tabel use case dari Aplikasi BSS Mobile Memberikan *approval* perizinan.

Tabel 4. 12 Tabel Use Case BSS Mobile Memberikan *Approval* Perizinan

Nama	Memberikan approval perizinan
Kode	UC009
Deskripsi	Aktor dapat melihat memberikan approval terhadap perizinan
Tipe	Fungsional
Pemicu	Aktor memasuki halaman detail permission
Aktor	wali santri, kepala sekolah
Kondisi Awal	Aktor ingin memberikan approval untuk perizinan.
Kondisi Akhir	Aktor memberikan approval untuk perizinan.
Alur Kejadian Secara Normal	1. Aktor memasuki halaman detail dari perizinan yang dipilih. 2. Sistem akan menampilkan halaman detail dari perizinan. 3. Aktor menekan tombol ‘Accept’ yang terdapat di bawah layar. 4. Sistem memasukkan data ke dalam basis data. 5. Sistem menampilkan dialog pemberian approval berhasil.
Alur Kejadian Alternatif	1. Aktor memasuki halaman detail dari perizinan yang dipilih. 2. Sistem akan menampilkan halaman detail dari perizinan. 3. Aktor menekan tombol ‘Reject’ yang terdapat di bawah layar. 4. Sistem memasukkan data ke dalam basis data. 5. Sistem menampilkan dialog pemberian approval berhasil.

Pengecualian	-

4.3.10. Menampilkan Notifikasi Perizinan Yang Masuk

Tabel 4.12 berikut merupakan tabel use case dari Aplikasi BSS Mobile Menampilkan Notifikasi Perizinan yang masuk.

Tabel 4. 13 Tabel Use Case BSS Mobile Menampilkan Notifikasi Perizinan Yang Masuk

Nama	Menampilkan notifikasi perizinan yang masuk
Kode	UC010
Deskripsi	Aktor dapat mengetahui update perizinan dari notifikasi
Tipe	Fungsional
Pemicu	Adanya update terkait perizinan oleh aktor
Aktor	Wali santri, Wali kelas, Kepala sekolah
Kondisi Awal	Aktor melakukan update pada perizinan
Kondisi Akhir	Notifikasi muncul pada perangkat *smartphone* aktor
Alur Kejadian Secara Normal	1. Aktor (wali santri) membuat perizinan untuk santri. 2. Sistem akan menampilkan notifikasi kepada perangkat aktor lain (wali kelas dan kepala sekolah) yang berhubungan dengan perizinan tersebut.
Alur Kejadian Alternatif	1. Aktor (wali kelas atau kepala sekolah) membuat perizinan untuk santri. 2. Sistem akan menampilkan notifikasi kepada perangkat aktor lain (wali santri) yang berhubungan dengan perizinan tersebut.
Pengecualian	-

4.3.11. Menampilkan Daftar Notifikasi

Tabel 4.13 berikut merupakan tabel use case dari Aplikasi BSS Mobile Menampilkan Daftar Notifikasi.

Table 4. 14 Tabel Use Case BSS Mobile Melakukan Pergantian Password Lisensi Yang Aktif

Nama	Menampilan Daftar Notifikasi
Kode	UC011
Deskripsi	Aktor dapat melihat daftar notifikasi
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘Notifikasi’ pada bottom navigation.
Aktor	Wali santri, Wali kelas, Kepala Sekolah
Kondisi Awal	Aktor ingin melihat daftar notifikasi yang masuk
Kondisi Akhir	Aktor dapat melihat daftar notifikasi yang masuk
Alur Kejadian Secara Normal	1. Aktor menekan tombol ‘Notifikasi’ pada bottom navigation. 2. Sistem akan menampilkan halaman berisi daftar notifikasi yang masuk.
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.12. Melihat Prediksi Hafalan Santri

Tabel 4.14 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Prediksi Hafalan Santri.

Tabel 4. 15 Tabel Use Case BSS Mobile Melihat Prediksi Hafalan Santri

Nama	Melihat Prediksi Hafalan Santri

Kode	UC012
Deskripsi	Aktor dapat melihat prediksi hafalan Al-Qur’an santri
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘Tahfidz’ pada bagian ‘Apps’ di menu utama.
Aktor	Wali santri
Kondisi Awal	Aktor ingin melihat laporan prediksi hafalan Al-Qur’an
Kondisi Akhir	Aktor dapat melihat laporan prediksi hafalan Al-Qir’an
Alur Kejadian Secara Normal	1. Aktor menekan tombol ‘Tahfidz’ pada menu Apps di halaman utama 2. Sistem akan menampilkan halaman data-data hafalan santri dan prediksi nya.
Alur Kejadian Alternatif	-
Pengecualian	-

4.3.13. Menampilkan Daftar News

Tabel 4.15 berikut merupakan tabel use case dari Aplikasi BSS Mobile Menampilkan Daftar *News*.

Tabel 4. 16 Tabel Use Case BSS Mobile Menampilkan Daftar *News*

Nama	Menampilkan Daftar News
Kode	UC013
Deskripsi	Aktor dapat melihat daftar news NFBS Bogor
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘News’ pada bottom navigation.
Aktor	Wali Santri, Wali Kelas, Kepala Sekolah
Kondisi Awal	User ingin melihat daftar news
Kondisi Akhir	User dapat melihat daftar news
Alur Kejadian Secara Normal	1. Aktor menekan tombol ‘News’ pada bottom navigation 2. Sistem menampilkan daftar news dari basis data
Alur Kejadian Alternatif	1. Aktor menekan tombol ‘News’ pada bottom navigation 2. Sistem menampilkan daftar news dari basis data 3. Apabila data kosong, maka tampilan akan menunjukkan bahwa ‘tidak ada news’.

Pengecualian	-

4.3.14. Melihat Detail News

Tabel 4.16 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Detail *News*.

Tabel 4. 17 Tabel Use Case BSS Mobile Melihat Detail *News*

Nama	Melihat Detail News
Kode	UC014
Deskripsi	Aktor dapat melihat detail news yang dipilih
Tipe	Fungsional
Pemicu	Aktor menekan salah satu item pada daftar news
Aktor	Wali Santri, Wali Kelas, Kepala Sekolah
Kondisi Awal	Aktor ingin melihat detail news tertentu
Kondisi Akhir	Aktor melihat detail news tertentu
Alur Kejadian Secara Normal	1. Aktor menekan salah satu item dari daftar news yang ditampilkan sistem 2. Sistem akan menampilkan halaman detail khusus dari item yang dipilih.

Alur Kejadian Alternatif	-
Pengecualian	-

4.3.15. Menampilkan Daftar Event

Tabel 4.17 berikut merupakan tabel use case dari Aplikasi BSS Mobile Menampilkan Daftar *Event*.

Tabel 4. 18 Tabel Use Case BSS Mobile Menampilkan Daftar *Event*

Nama	Menampilkan Daftar Event
Kode	UC015
Deskripsi	Aktor dapat melihat daftar event NFBS Bogor
Tipe	Fungsional
Pemicu	Aktor menekan tombol ‘event’ pada tab bar ‘events’ pada halaman news
Aktor	Wali Santri, Wali Kelas, Kepala Sekolah
Kondisi Awal	Aktor ingin melihat daftar event
Kondisi Akhir	Aktor dapat melihat daftar event

Alur Kejadian Secara Normal	1. Aktor menekan tombol ‘News’ pada bottom navigation, dan memilih tab ‘events’ pada tab bar. 2. Sistem menampilkan daftar event dari basis data
Alur Kejadian Alternatif	1. Aktor menekan tombol ‘News’ pada bottom navigation, dan memilih tab ‘events’ pada tab bar. 2. Sistem menampilkan daftar event dari basis data. 3. Apabila data kosong, maka tampilan akan menunjukkan bahwa ‘tidak ada event’.
Pengecualian	-

4.3.16. Melihat Detail Event

Tabel 4.18 berikut merupakan tabel use case dari Aplikasi BSS Mobile Melihat Detail *Event*.

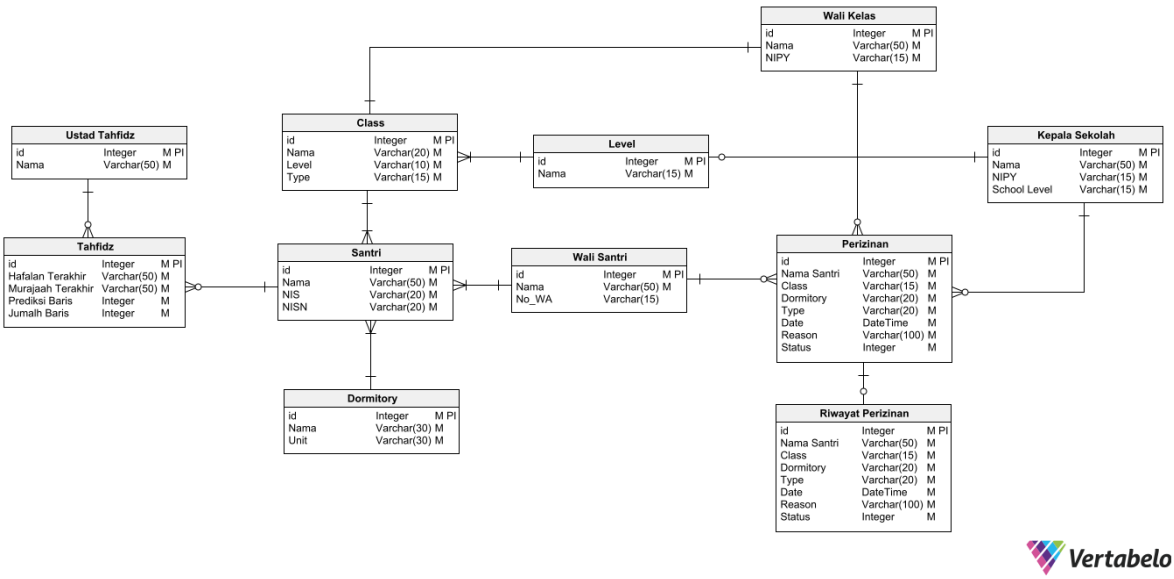
Tabel 4. 19 Tabel Use Case BSS Mobile Melihat Detail *Event*

Nama	Melihat Detail Event
Kode	UC016
Deskripsi	Aktor dapat melihat detail dari event yang dipilih.
Tipe	Fungsional
Pemicu	Aktor menekan salah satu item pada daftar event.

Aktor	Wali Santri, Wali Kelas, Kepala Sekolah
Kondisi Awal	Aktor ingin melihat detail dari event tertentu
Kondisi Akhir	Aktor dapat melihat detail dari event tertentu
Alur Kejadian Secara Normal	1. Aktor menekan salah satu item dari daftar news yang ditampilkan sistem. 2. Sistem akan menampilkan halaman detail khusus dari item yang dipilih.
Alur Kejadian Alternatif	-
Pengecualian	-

4.5. Conceptual Data Model

Gambar 4.2 berikut adalah Conceptual Data Model dari Aplikasi BSS Mobile.



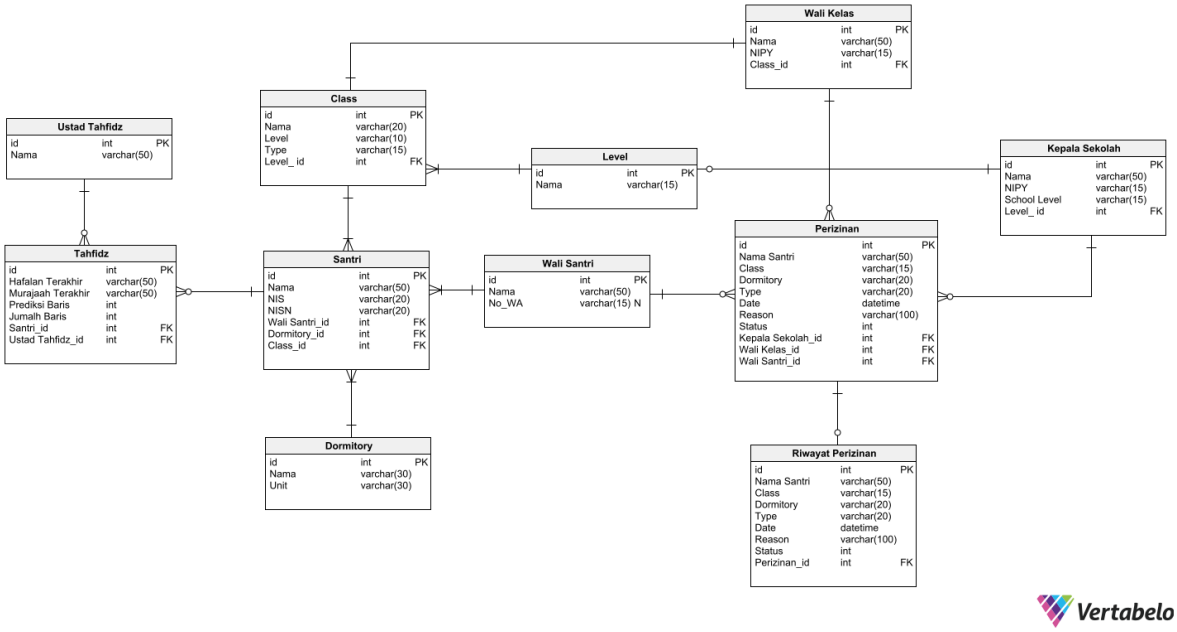
Gambar 4. 2 Gambar *Conceptual Data Model* Aplikasi BSS Mobile

Pada Gambar 4.2 terdapat 11 *table* yang merepresentasikan data yang akan dipakai dalam BSS Mobile. *Table* tersebut meliputi Ustad Tahfidz untuk menyimpan nama pengajar tahfidz nya. Kemudian ada Tahfidz untuk menyimpan data hafalan terakhir, murajaah terakhir, jumlah baris, dan prediksi baris. Selanjutnya ada santri menyimpan nama, NIS, NISN. Kemudian ada kelas yang menyimpan nama, level, type. Kemudian ada dormitory menyimpan nama dan unit. Kemudian ada

level yang terdiri dari nama. Kemudian ada wali kelas menyimpan nama dan NIPY. Selanjutnya ada kepala sekolah dengan menyimpan nama, NIPY, School level. Selanjutnya wali santri terdiri dari nama dan no_wa. Selanjutnya ada perizinan meliputi nama santri, class, dormitory, type, date, reason, status. Yang terakhir ada riwayat perizinan terdiri dari nama santri, class, dormitoty, type, date, reason, status.

4.6. Physical Data Model

Gambar 4.3 berikut adalah Physical Data Model dari Aplikasi BSS Mobile.



Gambar 4. 3 Physical Data Model Aplikasi BSS Mobile

Pada Gambar 4.3 terdapat *Physical Data Model* (PDM) dari aplikasi BSS Mobile yang merepresentasikan terkait dengan implementasi teknis dari *Conceptual Data Model* pada Gambar 4.3. Dalam PDM ini, akan menentukan rincian teknis seperti tipe data kolom, indeks, dan *key* tiap *table*.

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari perancangan sistem dan pengaplikasian sistem dalam bentuk situs web.

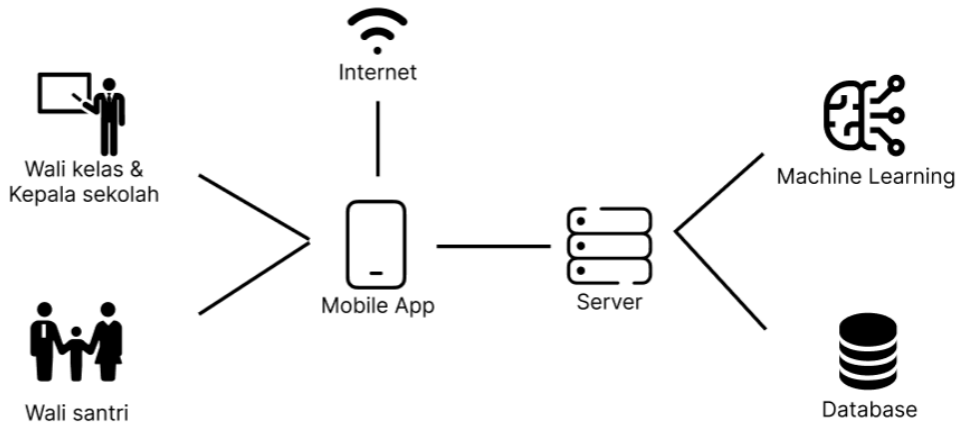
5.1. Implementasi Sistem

Sistem yang dibuat merupakan sistem digital untuk management sistem pendidikan di Nurul Fikri Boarding School Bogor dengan fitur-fitur pengembangan saat ini berupa sistem perizinan dan hafalan Al-Qur'an santri. Adapun fitur-fitur dari BSS Mobile adalah membuat pengajuan perizinan santri oleh wali santri, melihat status perizinan, melihat riwayat perizinan, pemberian izin dari wali kelas dan kepala sekolah,melihat event dan news di NFBS Bogor, melihat detail hafalan santri beserta prediksi nya dan sistem notifikasi yang membuat user dapat mengontrol sudah sejauh mana status perizinan yang diajukan.

Aplikasi ini merupakan penerapan dalam bentuk mobile apps dari pengembangan web yang sudah ada dengan penambahan beberapa fitur. Aplikasi ini dibuat menggunakan Flutter untuk bagian front-end, TypeScript untuk bagian back-end, dan Python untuk machine learning nya.

5.2. Implementasi Arsitektur Sistem

Pada bagian ini akan digambarkan arsitektur sistem. Adapun diagram arsitektur sistem yang diterapkan untuk Aplikasi BSS Mobile terdapat pada Gambar 5.1.



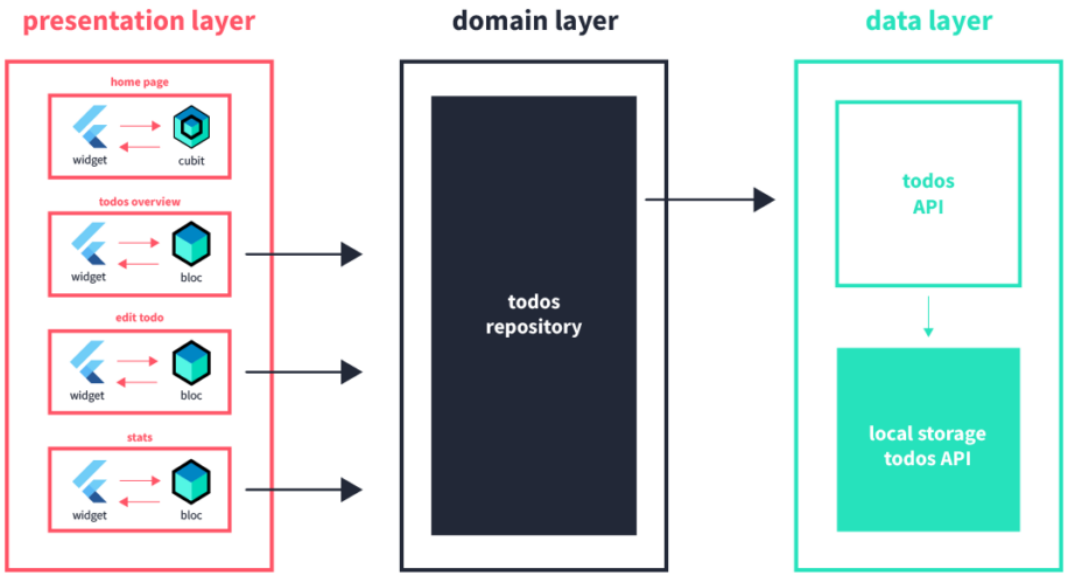
Gambar 5. 1 Diagram Arsitektur Sistem BSS Mobile

Pada Gambar 5.1 terdapat diagram arsitektur sistem BSS Mobile dimana wali kelas & kepala sekolah serta wali santri dapat mengakses *mobile app* BSS Mobile dengan syarat harus memiliki internet. Kemudian data yang tersimpan dalam aplikasi ini akan diintegrasikan oleh server dan dimasukkan ke dalam *database*. Kemudian ada juga implementasi dari *machine learning* untuk pembuatan model untuk sistem prediksi hafalan santri.

Selanjutnya, terdapat implementasi bagian front-end aplikasi, yang berupa tampilan antarmuka dan pengolahan API yang dikirim oleh bagian back-end, disusul dengan implementasi bagian back-end aplikasi, yang berupa integrasi dengan server, database dan machine learning, dan terakhir yaitu implementasi bagian machine learning, yaitu pembuatan model untuk sistem prediksi hafalan santri.

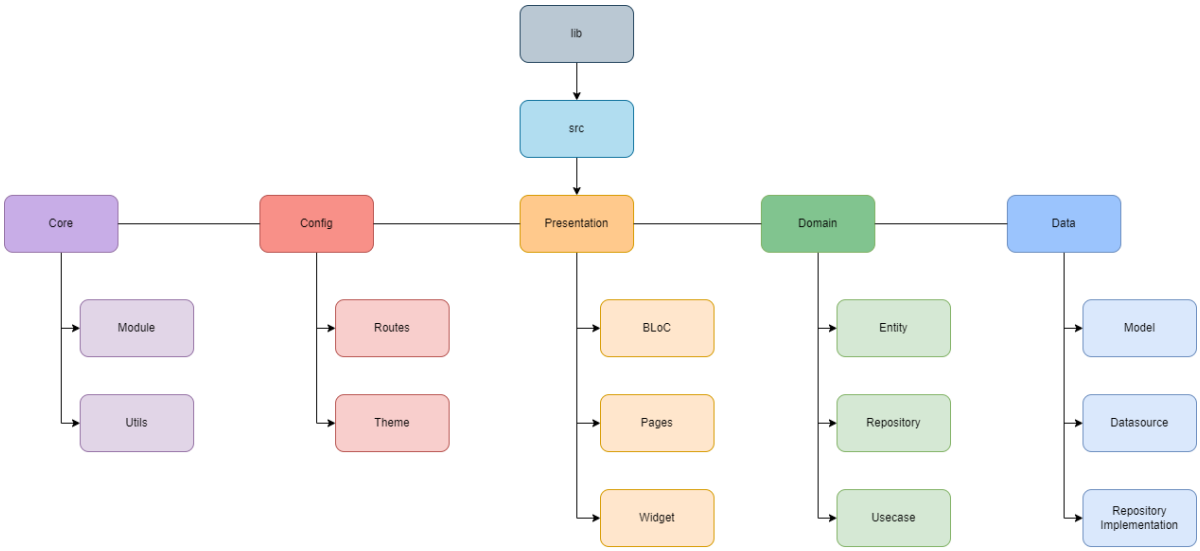
5.2.1. Implementasi Front-End

Pada bagian ini akan dibahas mengenai implementasi pada front-end aplikasi. Front-end sistem aplikasi ini dibangun menggunakan sebuah arsitektur yaitu *clean architecture*. *Clean Architecture* adalah pendekatan arsitektur perangkat lunak yang berfokus pada pemisahan kode berdasarkan tanggung jawab nya.



Gambar 5. 2 Clean Architecture Pada Flutter

Pada Gambar 5.2 terdapat tiga lapisan utama yang terdapat dalam arsitektur ini, yaitu presentation layer, berfungsi sebagai lapisan paling luar dan berinteraksi langsung dengan user melalui tampilan antarmuka aplikasi. Kemudian ada domain layer, bertanggung jawab untuk memproses input tersebut dan menghasilkan output. Output tersebut kemudian dikirim kembali ke presentation layer untuk ditampilkan kepada pengguna. Terakhir yaitu data layer, bertanggung jawab untuk berinteraksi dengan dunia nyata, misalnya dengan database, jaringan, atau sistem operasi. Data layer menyediakan data yang dibutuhkan oleh domain layer untuk memproses input dari pengguna.



Gambar 5. 3 Overview Dan Struktur Proyek BSS Mobile

Pada Gambar 5.3 berisi tentang overview dan struktur Proyek BSS Mobile untuk bagian *front-end* yang mengimplementasikan layer *Clean Architecture* yang ada pada Gambar 5.2. Berdasarkan ilustrasi dari Gambar 5.3, akan dijelaskan implementasi dari masing-masing komponen pada struktur *project*:

1. Entity

   Entity memegang peranan penting sebagai model inti yang merepresentasikan data dan perilaku dalam domain bisnis.

```
class EventEntity extends Equatable {
  final String? title;
  final String? location;
  final String? slug;
  final String? imageUrl;
  final String? date;

  const EventEntity(
    this.title,
    this.location,
    this.slug,
    this.imageUrl,
    this.date
  );

  @override
  List<Object?> get props {
    return [
      title,
      location,
      slug,
      imageUrl,
      date
    ];
  }
}
```

Kode Sumber 5. 1 Contoh Implementasi Entity

Kode diatas merupakan salah satu contoh implementasi dalam pembuatan sebuah class entity, dimana pertama-tama akan dideklarasikan atribut atau perilaku dari entity event, kemudian dibuatkannya constructor berdasarkan perilaku-perilaku tersebut. Penggunaan *library Equatable* pada kode diatas, digunakan untuk membandingkan apakah properti dari kedua objek yang dibandingkan sama atau tidak.

2. Repository

   Repository bertanggung jawab sebagai kontrak atau metode dari *business logic* yang terdapat pada aplikasi.

```
abstract class EventRepository {

  Future<DataState<List<EventEntity>>> getAllEvent(int page);

}
```

Kode Sumber 5. 2 Contoh Implementasi Repository

```
abstract class DataState<T> {
  final T ? data;
  final DioException ? exception;
  final String ? message;
```

```
  const DataState({
    this.data,
    this.exception,
    this.message
  });
}

class DataSuccess<T> extends DataState<T>{
  const DataSuccess(T data) : super(data: data);
}

class DataFailed<T> extends DataState<T>{
  const DataFailed(DioException exception) : super(exception: exception);
}
```

Kode Sumber 5. 3 Interface DataState

Dalam kode 5.2, dibuat sebuah *abstract class repository* yang bertujuan untuk membuat kelas ini agar independen atau memiliki tanggung jawab hanya untuk membuat sebuah kontrak dari fitur yang dikembangkan. Penggunaan *abstract class* ini bertujuan untuk diwariskan oleh kelas konkret yang akan mengimplementasikan metode-metode nya, *abstract class* ini akan diimplementasikan oleh *class* pada *Repository Implementation*. Penggunaan *syntax Future* digunakan agar membuat metode tersebut menjadi *asynchronous* atau dapat berjalan secara paralel dibelakang. Dalam kasus pada kode diatas, dibuat sebuah *method* yang bernama getAllEvent yang bertipe data *list* dari sebuah *entity* yang bernama EventEntity, *method* ini memiliki parameter berupa *integer*, *method* tersebut memiliki tanggung jawab untuk mendapatkan seluruh event dari basis data. Penggunaan DataState pada kode 5.2 bertujuan untuk *management state* pada saat proses pengambilan data. Hal ini dijelaskan pada kode 5.3, dimana proses pengambilan data bisa saja berhasil ataupun gagal, oleh karena itu dibuatlah *interface* ini yang bertujuan untuk bisa menangani dua kemungkinan tersebut.

3. UseCase

UseCase adalah komponen yang bertanggung jawab untuk melakukan operasi bisnis tertentu.

```
class EventUseCase implements UseCase<DataState<List<EventEntity>>, int> {
  final EventRepository _eventRepository;
  EventUseCase(this._eventRepository);
  @override
  Future<DataState<List<EventEntity>>> call({int ? params}) {
    return _eventRepository.getAllEvent(params!);
  }
}
```

Kode Sumber 5. 4 Contoh Implementasi UseCase

```
abstract class UseCase<Type, Params>{
  Future<Type> call({Params params});
}
```

Kode Sumber 5. 5 Interface UseCase

Sesuai dengan prinsip *clean architecture*, yaitu pemisahan kode berdasarkan tanggung jawabnya, terlihat bahwa dibuatnya sebuah kelas khusus untuk menangani tanggung jawab tersebut. class EventUseCase akan mengimplementasikan sebuah *interface* UseCase yang telah dibuat, *interface* ini terbuat dari sebuah *abstract class* yang didalamnya terdapat method *call* dan nantinya hanya akan di *override* pada kelas konkret nya, dimana *return value* dan parameter nya harus sama dengan *dependency* yang diambil, dalam kasus ini, EventUseCase menerima EventRepository sebagai dependency nya. UseCase ini nanti akan terhubung

dengan *presentation layer* pada komponen BLoC.

4. Model

Model digunakan pada data layer untuk mewakili data dari sumber eksternal, seperti data dari *database*, API, atau file. Model pada data layer dapat mengimplementasikan entity dari domain layer.

```
class EventModel extends EventEntity {
  const EventModel({
    String? title,
    String? location,
    String? slug,
    String? imageUrl,
    String? date
  }):super(
    title,
    location,
    slug,
    imageUrl,
    date
  );

  factory EventModel.fromJson(Map<String, dynamic> map) {
    return EventModel(
      title: map['title'] ?? emptyString,
      location: map['location'] ?? emptyString,
      slug: map['slug'] ?? emptyString,
      imageUrl: map['imgURL'] ?? emptyString,
      date: map['date'] ?? emptyString
    );
  }
}
```

Kode Sumber 5. 6 Contoh Implementasi Model

Berdasarkan kode 5.6, terlihat bahwa atribut-atribut dari Entity akan diwariskan di Model. Dalam kasus ini, EventModel akan mengimplementasikan EventEntity dan harus memenuhi persyaratan *constructor* yang ada pada EventEntity menggunakan *method super*. Terlihat juga pembuatan *factory method* yang bernama fromJson, metode ini digunakan untuk membuat object EventModel dari data yang akan diambil dari API nanti. Metode ini menggunakan *map* sebagai parameter untuk menyamakan *key* pada JSON dan memberikan nilai *default* apabila data yang ditemukan berupa *null*.

5. DataSource

DataSource merupakan komponen dalam *clean architecture* yang digunakan untuk mengambil data dari eksternal, entah itu secara *remote* maupun lokal. Dalam project ini, kami menggunakan DataSource *remote* yang berasal dari API dan menggunakan *library Retrofit* dan *Dio*.

```
@RestApi(baseUrl: baseUrl)
abstract class EventApiService {

  factory EventApiService(Dio dio) = _EventApiService;

  @GET(event)
  Future<HttpResponse<List<EventModel>>> getAllEvent({
    @Query('page') int ? page
  });
}
```

Kode Sumber 5. 7 Contoh Implementasi DataSource

```dart
part of 'event_api_service.dart';

class _EventApiService implements EventApiService {
  _EventApiService(
    this._dio, {
    this.baseUrl,
  });

  final Dio _dio;

  String? baseUrl;

  @override
  Future<HttpResponse<List<EventModel>>> getAllEvent({int? page}) async {
    const _extra = <String, dynamic>{};
    final queryParameters = <String, dynamic>{r'page': page};
    queryParameters.removeWhere((k, v) => v == null);
    final _headers = <String, dynamic>{};
    final Map<String, dynamic>? _data = null;
    final _result = await _dio.fetch<List<dynamic>>(
        _setStreamType<HttpResponse<List<EventModel>>>(Options(
      method: 'GET',
      headers: _headers,
      extra: _extra,
    )
            .compose(
              _dio.options,
              '/event',
              queryParameters: queryParameters,
              data: _data,
            )
            .copyWith(
                baseUrl: _combineBaseUrls(
              _dio.options.baseUrl,
              baseUrl,
            ))));
    var value = _result.data!
        .map((dynamic i) => EventModel.fromJson(i as Map<String, dynamic>))
        .toList();
    final httpResponse = HttpResponse(value, _result);
    return httpResponse;
  }

  RequestOptions _setStreamType<T>(RequestOptions requestOptions) {
    if (T != dynamic &&
        !(requestOptions.responseType == ResponseType.bytes ||
            requestOptions.responseType == ResponseType.stream)) {
      if (T == String) {
        requestOptions.responseType = ResponseType.plain;
      } else {
        requestOptions.responseType = ResponseType.json;
      }
    }
    return requestOptions;
  }

  String _combineBaseUrls(
    String dioBaseUrl,
    String? baseUrl,
  ) {
    if (baseUrl == null || baseUrl.trim().isEmpty) {
      return dioBaseUrl;
    }

    final url = Uri.parse(baseUrl);
```

```
    if (url.isAbsolute) {
      return url.toString();
    }

    return Uri.parse(dioBaseUrl).resolveUri(url).toString();
  }
}
```

Kode Sumber 5. 8 File Yang Dibuat Oleh Library Build_Runner

Terlihat dari Kode 5.7, *abstract class* ini menggunakan anotasi RestAPI yang menandakan kelas ini sebagai API *client* yang dibuat dengan *library* Retrofit dan membutuhkan *base* URL dari API yang akan diakses. *Abstract class* ini menggunakan salah satu metode RestAPI yaitu GET yang berfungsi untuk mengambil data dari luar menggunakan *end point* konstan yang sudah didefinisikan, memiliki anotasi tambahan sebagai parameternya seperti Query yang merupakan bagian dari *library* ini juga. *Abstract class* ini tidak diinstansiasi secara langsung, tetapi melalui *factory constructor* EventApiService(Dio dio) yang dibuat oleh Retrofit. Penggunaan sintaks *part 'event_api_service.g.dart'* berfungsi untuk menunjukkan bahwa *file* ini merupakan bagian dari *file* yang dihasilkan oleh Retrofit. Setelah itu, menjalankan command pada terminal menggunakan sintaks *dart run build_runner build* yang bertujuan untuk menjalankan *library* Build_Runner dan kemudian akan menghasilkan *file* baru seperti yang ditunjukkan pada kode 5.8.

6. Repository Implementation

Repository Implementation atau *RepositoryImpl* merupakan salah satu komponen dalam *clean architecture* yang akan mengimplementasikan Repository yang terdapat pada *domain layer* dan akan memanggil *method* dari DataSource.

```
class EventRepositoryImpl implements EventRepository {
  final EventApiService _eventApiService;

  const EventRepositoryImpl({required EventApiService eventApiService}) :
_eventApiService = eventApiService;

  @override
  Future<DataState<List<EventModel>>> getAllEvent(int page) async {
    try {
      final httpResponse = await _eventApiService.getAllEvent(
        page: page
      );

      if(httpResponse.response.statusCode == HttpStatus.ok){
        return DataSuccess(httpResponse.data);
      } else {
        return DataFailed(
          DioException(
            error: httpResponse.response,
            response: httpResponse.response,
          requestOptions: httpResponse.response.requestOptions
          )
        );
      }
    } on DioException catch(e){
      switch(e.type){
        case DioExceptionType.connectionTimeout:
        case DioExceptionType.receiveTimeout:
        case DioExceptionType.sendTimeout:
          'Request Timeout';
          break;
        case DioExceptionType.badCertificate:
          'Bad Certificate';
        case DioExceptionType.badResponse:
```

```
          'Bad Response';
        case DioExceptionType.cancel:
          'Request Cancel';
        case DioExceptionType.connectionError:
          'Connection Error';
        case DioExceptionType.unknown:
          'Request Unknown';

        default: 'Request Unknown';
      }
      return DataFailed(e);
    }
  }

}
```

Kode Sumber 5. 9 Contoh Implementasi Repository Implementation

Pada Kode Sumber 5.9, terlihat bahwa kelas konkret yang dalam kasus ini adalah EventRepositoryImpl mengimplementasi abstract class EventRepository. Kelas ini juga menerima DataSource (EventApiService) dependency, yang digunakan untuk mengakses API. Pada kelas ini lah dilakukannya request API dan kemudian menangani response nya, apabila status response nya 200, maka akan mengembalikan data melalui interface DataState lewat kelas DataSuccess, dan apabila response status code nya selain itu, maka akan mengembalikan data lewat kelas DataFailed seperti pada Kode Sumber 5.9.

7. BLoC

BLoC adalah singkatan dari Business Logic Component, yang merupakan design pattern yang digunakan untuk memisahkan logika bisnis dari tampilan dalam aplikasi Flutter.BLoC menggunakan konsep event-driven, di mana perubahan data di aplikasi dipicu oleh event. Event ini kemudian ditangani oleh bloc, yang merupakan komponen yang bertanggung jawab untuk mengelola logika bisnis. BLoC terdiri dari tiga komponen yaitu State, Event dan Bloc.

```
abstract class NewsState extends Equatable{
  final List<NewsEntity> ? news;
  final List<EventEntity> ? events;
  final DioException ? exception;

  const NewsState({this.news, this.events, this.exception});

  @override
  List<Object?> get props => [news, events, exception];
}

class NewsInit extends NewsState{
  const NewsInit();
}

class NewsLoading extends NewsState{
  const NewsLoading();
}

class NewsSuccess extends NewsState{
  const NewsSuccess(List<NewsEntity> ? news) : super(news: news);
}

class EventSuccess extends NewsState{
  const EventSuccess(List<EventEntity> ? events) : super(events: events);
}

class NewsError extends NewsState{
  const NewsError(DioException ? exception) : super(exception: exception);
}
```

```
class EventError extends NewsState{
  const EventError(DioException ? exception) : super(exception: exception);
}
```

Kode Sumber 5. 10 Contoh Implementasi BLoC State

State adalah representasi dari kondisi data saat ini dalam aplikasi. State biasanya didefinisikan sebagai kelas atau *struct* yang berisi data yang diperlukan untuk menampilkan aplikasi. Terlihat pada Kode Sumber 5.10, terlihat bahwa kondisi data bisa saja dalam kondisi inisialisasi (NewsInit), *loading* (NewsLoading), *Success* (NewsSuccess), maupun dalam kondisi *Error* (NewsError). kondisi-kondisi inilah yang nantinya akan ditangani didalam Pages, dengan mengetahui berbagai macam kondisi dari sebuah data, maka cara menangani implementasi untuk tampilan antarmuka menjadi lebih mudah dan teratur.

```
abstract class NewsEvent extends Equatable{
  const NewsEvent();

  @override
  List<Object?> get props => [];
}

class GetAllNews extends NewsEvent{
  final int page;
  const GetAllNews(this.page);
}

class GetAllEvent extends NewsEvent{
  final int page;
  const GetAllEvent(this.page);
}
```

Kode Sumber 5. 11 Contoh Implementasi BLoC Event

Event adalah objek yang mewakili perubahan data yang dapat terjadi di aplikasi. Event biasanya didefinisikan sebagai kelas atau struct yang berisi data yang diperlukan untuk mengubah state. Hal ini dapat dilihat pada Kode Sumber 5.11.

```
class NewsBloc extends Bloc<NewsEvent, NewsState>{

  final NewsUseCase _newsUseCase;
  final EventUseCase _eventUseCase;

  NewsBloc(this._newsUseCase, this._eventUseCase) : super(const NewsInit()){
    on<GetAllNews>(onGetAllNews);
    on<GetAllEvent>(onGetAllEvent);
  }

  void onGetAllNews(GetAllNews event, Emitter<NewsState> emit) async {
    final dataState = await _newsUseCase.call(params: event.page);

    if(dataState is DataSuccess){
      emit(NewsSuccess(dataState.data!));
    }

    if(dataState is DataFailed){
      emit(
        NewsError(dataState.exception!)
      );
    }
  }

  void onGetAllEvent(GetAllEvent event, Emitter<NewsState> emit) async {
    final dataState = await _eventUseCase.call(params: event.page);

    if(dataState is DataSuccess){
      emit(
```

```
        EventSuccess(dataState.data!)
      );
    }

    if(dataState is DataFailed){
      emit(
        EventError(dataState.exception!)
      );
    }
  }
}
```

Kode Sumber 5. 12 Contoh Implementasi BLoC Bloc

Bloc adalah komponen yang bertanggung jawab untuk menangani event dan mengubah state. Bloc biasanya diimplementasikan sebagai kelas yang memiliki metode handleEvent() untuk menangani event dan metode state untuk mengembalikan state saat ini. Hal ini dapat dilihat pada Kode Sumber 5.12.

8. Pages

Pages merupakan implementasi tampilan antarmuka dari suatu halaman menjadi sebuah kode yang terstruktur. Untuk membuat tampilan antar muka dalam Flutter, diperlukan import suatu package terlebih dahulu, yaitu *import 'package:flutter/material.dart'*. package ini dapat membuat kita mengakses berbagai macam syntax widget dan mengkombinasikannya menjadi sebuah tampilan antarmuka halaman yang sesuai. Pada dasarnya, terdapat 2 jenis kelas yang pasti digunakan, pertama yaitu StatefullWidget, yaitu menjadikan tampilan antarmuka bisa menjadi dinamis tanpa perlu me-restart page, kedua yaitu StatelessWidget, yaitu hanya membuat page hanya dalam kondisi statis. Didalam pages ini juga kita menginisalisasi BLoC dan menggunakannya, Hal ini dapat dilihat lebih jelas pada Kode Sumber 5.13.

```
class EventPage extends StatefulWidget {
  final NewsBloc eventBloc;
  const EventPage({super.key, required this.eventBloc});

  @override
  State<EventPage> createState() => _EventPageState();
}

class _EventPageState extends State<EventPage>
    with AutomaticKeepAliveClientMixin<EventPage> {

  @override
  bool get wantKeepAlive => true;

  final     PagingController<int,     EventEntity>     _pagingController     =
PagingController(firstPageKey: 1);

  Future<List<EventEntity>> generate(int pageKey, int pageSize) async {
    widget.eventBloc.add(GetAllEvent(pageKey));

    final state = await widget.eventBloc.stream.firstWhere((event) => event
is EventSuccess || event is EventError);

    if(state is EventSuccess){
      return state.events!;
    }

    if(state is EventError){
      _pagingController.error = state.exception;

      if (!context.mounted) return state.events!;
      showDialogError(context,   state.exception!.message.toString(),   'Load
Events Error');
    }
```

```
    return state.events!;
  }

  _moveToDetail(EventEntity data){
    Navigator.pushNamed(context, '/DetailEvent', arguments: data.slug);
  }

  Widget _listWidget(BuildContext context){
    return PaginationScrollPage<EventEntity>(
        pageSize: 6,
        firstPageKey: 1,
        fetch: generate,
        pagingController: _pagingController,
        noItemFoundMsg: 'There is no events now',
        itemBuilder: (context, item, index) {
          return Padding(
            padding: const EdgeInsets.only(left: 20, right: 20, top: 20),
            child: CardItem(
              title: item.title!,
              description: item.location!,
              date: item.date!,
              imageUrl: item.imageUrl!,
              onTapHandler: (){
                _moveToDetail(item);
              },
            ),
          );
        });
  }

  @override
  Widget build(BuildContext context) {
    super.build(context);

    return BlocBuilder(
      bloc: widget.eventBloc,
      builder: (context, _) => _listWidget(context)
    );
  }
}
```

Kode Sumber 5. 13 Contoh Implementasi Pages

9. Widget

Widget adalah unit dasar dari tampilan dalam Flutter. Widget digunakan untuk membangun antarmuka pengguna (UI) aplikasi Flutter. Widget dapat berupa elemen dasar seperti teks, gambar, atau tombol, atau widget kompleks yang terdiri dari beberapa widget lain. Widget dapat dikombinasikan untuk membuat tampilan yang kompleks. Untuk implementasinya dapat dilihat pada Kode Sumber 5.14.

```
class TopBar extends StatelessWidget {
  const TopBar({super.key});

  @override
  Widget build(BuildContext context) {
    double height = MediaQuery.of(context).size.height;

    return Container(
      width: double.infinity,
      height: height * 0.10,
      decoration: gradientBackground(),
      child: Align(
          alignment: Alignment.center,
```

```
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Image.asset('assets/logo.png', width: 50, height: 50),
              const SizedBox(width: 10),
              SvgPicture.asset('assets/logo_text.svg')
            ],
          )),
    );
  }
}
```

Kode Sumber 5. 14 Contoh Implementasi Widget

10. Routes

Routes merupakan salah satu komponen tambahan yang diperlukan untuk menerapkan *clean architecture*. Routes merupakan satu kelas khusus yang mengatur semua navigasi beserta *argument* wajib nya yang terdapat pada aplikasi, hal ini tentunya sesuai dengan prinsip awal *clean architecture* yaitu pemisahan kode berdasarkan tanggung jawab nya. Untuk implementasinya dapat dilihat pada Kode Sumber 5.15

```
class AppRoutes {
  static Route onGenerateRoutes(RouteSettings settings) {
    switch (settings.name) {
      case '/':
        return _materialRoute(
            SplashScreen(
                preferenceModule: ioc.get(),
            )
        );

      case '/Authentication':
        return _materialRoute(
            LoginPage(
                authBloc: ioc.get(),
                preferenceModule: ioc.get(),
            )
        );

      case '/AuthenticationTeacher':
        return _materialRoute(
            LoginForTeacherPage(
                authBloc: ioc.get(),
                preferenceModule: ioc.get()
            )
        );

      case '/Navigation':
        return _materialRoute(
            MainNavigation(
                preferenceModule: ioc.get(),
                newsBloc: ioc.get(),
                authBloc: ioc.get(),
                notificationBloc: ioc.get(),
                index: settings.arguments as int,
            )
        );

      case '/News':
        return _materialRoute(
            NewsTabPage(
                newsBloc: ioc.get()
            )
        );

      case '/DetailNews':
```

```
        return _materialRoute(
            NewsDetailPage(
                slug: settings.arguments as String,
                detailBloc: ioc.get()
            )
        );

      case '/DetailEvent':
        return _materialRoute(
            DetailEventPage(
                slug: settings.arguments as String,
                detailBloc: ioc.get()
            )
        );

      case '/About':
        return _materialRoute(const AboutPage());

      case '/Permission':
        return _materialRoute(
          PermissionPage(
              preferenceModule: ioc.get(),
              bloc: ioc.get(),
          )
        );

      case '/Tahfidz':
        return _materialRoute(
          TahfidzPage(
            sharedPreferenceModule: ioc.get(),
            bloc: ioc.get(),
          )
        );

      case '/PermissionHistory':
        return _materialRoute(
          PermissionHistoryPage(
              preferenceModule: ioc.get(),
              bloc: ioc.get(),
          )
        );

      case '/PermissionDetail':
        return _materialRoute(
          DetailPermissionPage(
              permission: settings.arguments as PermissionEntity,
          )
        );

      case '/TeacherPermission':
        return _materialRoute(
          TeacherPermissionPage(
              preferenceModule: ioc.get(),
              bloc: ioc.get(),
          )
        );

      case '/HeadmasterPermission':
        return _materialRoute(
            HeadmasterPermissionPage(
              preferenceModule: ioc.get(),
              bloc: ioc.get(),
            )
        );

      case '/TeacherPermissionHistory':
        return _materialRoute(
            TeacherPermissionHistoryPage(
```

```
                bloc: ioc.get(),
                preferenceModule: ioc.get(),
            )
        );

      case '/HeadmasterPermissionHistory':
        return _materialRoute(
            HeadmasterPermissionHistoryPage(
              bloc: ioc.get(),
              preferenceModule: ioc.get(),
            )
        );

      case '/TeacherPermissionDetail':
        List<dynamic> args = settings.arguments as List<dynamic>;
        return _materialRoute(
            TeacherPermissionDetailPage(
                permission: args[0],
                bloc: ioc.get(),
                token: args[1],
            )
        );

      case '/HeadmasterPermissionDetail':
        List<dynamic> args = settings.arguments as List<dynamic>;
        return _materialRoute(
            HeadmasterPermissionDetailPage(
              permission: args[0],
              bloc: ioc.get(),
              token: args[1],
            )
        );

      case '/TeacherPermissionHistoryDetail':
        return _materialRoute(
            TeacherPermissionHistoryDetailPage(
                permission : settings.arguments as PermissionEntity
            )
        );

      case '/HeadmasterPermissionHistoryDetail':
        return _materialRoute(
            HeadmasterPermissionHistoryDetailPage(
                permission : settings.arguments as PermissionEntity
            )
        );

      default:
        throw Exception('Invalid Route: ${settings.name}');
    }
  }

  static Route<dynamic> _materialRoute(Widget view) {
    return MaterialPageRoute(builder: (_) => view);
  }
}
```

Kode Sumber 5. 15 Contoh Implementasi Routes

11. Theme

Theme merupakan satu file khusus yang berfungsi untuk mengatur tema dan *style default* dari widget pada aplikasi. Dengan menggunakan theme, maka tidak diperlukannya penulisan yang berulang-ulang, seperti mengatur ukuran dan warna *font default*, warna latar belakang maupun jenis *font* yang digunakan. Hal ini dapat dilihat pada Kode Sumber 5.16.

```
ThemeData theme() {
```

```
  return ThemeData(
    scaffoldBackgroundColor: whitePrimary,
    fontFamily: 'Poppins',
    primaryColor: bluePrimary,
    cardColor: whitePrimary,
  );
}
```

Kode Sumber 5. 16 Contoh Implementasi Theme

12. Module

Module merupakan kelas-kelas yang memiliki tanggung jawab untuk meningkatkan fungsionalitas pada aplikasi, seperti Network Module, Shared Preference Module, Firebase Module, maupun Dependency Injection Module.

```
class NetworkModule {
  final Dio _dio;
  final String _baseURL;

  const NetworkModule({required Dio dio, required String baseURL})
      : _dio = dio,
        _baseURL = baseURL;

  BaseOptions _dioOptions() {
    BaseOptions opts = BaseOptions();
    opts.baseUrl = _baseURL;
    return opts;
  }

  Dio provideDio() {
    _dio.options = _dioOptions();
    return _dio;
  }
}
```

Kode Sumber 5. 17 Contoh Implementasi Network Module

```
class SharedPreferenceModule {
  final SharedPreferences pref;
  static const String _prefUser= 'user_data';
  static const String _currentUser = 'current_user_data';
  static const String _userRole = 'user_role';
  static const String _employeeStatus = 'employee_status';
  static const String _fcmToken = 'fcm_token';
  static const String _deviceId = 'device_id';
  static const String _onTokenRefresh = 'token_refresh';

  SharedPreferenceModule({required this.pref});

  void clear() {
    pref.remove(_prefUser);
    pref.remove(_currentUser);
    pref.remove(_userRole);
    pref.remove(_employeeStatus);
    pref.remove(_onTokenRefresh);
  }

  void removeFcmToken() => pref.remove(_fcmToken);

  void saveUserData(String userDataInJson) => pref.setString(_prefUser,
userDataInJson);

  void saveCurrentUserData(String userDataInJson) =>
pref.setString(_currentUser, userDataInJson);
```

```
  void saveUserRole(String userRole) => pref.setString(_userRole, userRole);

  void saveEmployeeStatus(bool isEmployee) => pref.setBool(_employeeStatus,
isEmployee);

  void saveFcmToken(String token) => pref.setString(_fcmToken, token);

  void saveDeviceId(String deviceId) => pref.setString(_deviceId, deviceId);

  void saveOnTokenRefresh(bool onRefresh) => pref.setBool(_onTokenRefresh,
onRefresh);

  String getUserData() {
    String userDataInJson = pref.getString(_prefUser) ?? '';
    return userDataInJson;
  }

  String getCurrentUserData() {
    String userDataInJson = pref.getString(_currentUser) ?? '';
    return userDataInJson;
  }

  String getUserCurrentRole() {
    String userStatus = pref.getString(_userRole) ?? '';
    return userStatus;
  }

  String getFcmToken() {
    String token = pref.getString(_fcmToken) ?? '';
    return token;
  }

  String getDeviceId() {
    String deviceId = pref.getString(_deviceId) ?? '';
    return deviceId;
  }

  bool getEmployeeStatus() {
    bool isEmployee = pref.getBool(_employeeStatus) ?? false;
    return isEmployee;
  }

  bool getOnTokenRefresh() {
    bool onRefresh = pref.getBool(_onTokenRefresh) ?? false;
    return onRefresh;
  }
}
```

Kode Sumber 5. 18 Contoh Implementasi Shared Preference Module

```
class FirebaseAPI {
  final _fcm = FirebaseMessaging.instance;
  static final FlutterLocalNotificationsPlugin
  _flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
  final _sharedPreferenceModule = SharedPreferenceModule(pref: ioc.get());

  Future<void> initNotifications() async {
    await _fcm.requestPermission(
      alert: true,
      announcement: true,
      badge: true,
      carPlay: false,
      criticalAlert: false,
      provisional: false,
      sound: true,
    );
```

```
    final fcmToken = await _fcm.getToken();
    if(_sharedPreferenceModule.getFcmToken().isEmpty){
      _sharedPreferenceModule.saveFcmToken(fcmToken!);
    }

    _fcm.onTokenRefresh.listen((fcmToken) {
      _sharedPreferenceModule.removeFcmToken();
      _sharedPreferenceModule.saveFcmToken(fcmToken);
      _sharedPreferenceModule.saveOnTokenRefresh(true);
    })
    .onError((err){
      throw 'Error on handling new token refresh';
    });

    // get Device Id
    final String? deviceId = await getId();
    if(_sharedPreferenceModule.getDeviceId().isEmpty && deviceId != null){
      _sharedPreferenceModule.saveDeviceId(deviceId);
    }
  }

  Future<String?> getId() async {
    var deviceInfo = DeviceInfoPlugin();
    if (Platform.isIOS) { // import 'dart:io'
      var iosDeviceInfo = await deviceInfo.iosInfo;
      return iosDeviceInfo.identifierForVendor; // unique ID on iOS
    } else if(Platform.isAndroid) {
      var androidDeviceInfo = await deviceInfo.androidInfo;
      return androidDeviceInfo.id; // unique ID on Android
    }
    return null;
  }

  // initialize local notifications
  static Future localNotificationInit() async {
    // initialise the plugin. app_icon needs to be a added as a drawable
resource to the Android head project
    const AndroidInitializationSettings initializationSettingsAndroid =
    AndroidInitializationSettings('@mipmap/ic_launcher');
    final DarwinInitializationSettings initializationSettingsDarwin =
    DarwinInitializationSettings(
      onDidReceiveLocalNotification: (id, title, body, payload) {},
    );
    final InitializationSettings initializationSettings =
    InitializationSettings(
        android: initializationSettingsAndroid,
        iOS: initializationSettingsDarwin,
    );
    _flutterLocalNotificationsPlugin.initialize(initializationSettings,
        onDidReceiveNotificationResponse: onNotificationTap,
        onDidReceiveBackgroundNotificationResponse: onNotificationTap);
  }

  // on tap local notification in foreground
  static void onNotificationTap(NotificationResponse notificationResponse) {
    navigatorKey.currentState!
        .pushNamed('/Navigation', arguments: 2);
  }

  // show a simple notification
  Future showSimpleNotification({
    required String title,
    required String body,
    required String payload,
  }) async {
    AndroidNotificationDetails androidNotificationDetails =
    const AndroidNotificationDetails(
        'pushNotification',
```

```
        'pushNotificationChannel',
        channelDescription: 'your channel description',
        importance: Importance.max,
        priority: Priority.max,
        ticker: 'ticker'
    );
    NotificationDetails notificationDetails =
    NotificationDetails(android: androidNotificationDetails);
    await _flutterLocalNotificationsPlugin
        .show(0, title, body, notificationDetails, payload: payload);
  }

  static Future<void> firebaseBackgroundMessage(RemoteMessage message) async
{
    String payloadData = jsonEncode(message.data);
    if(message.notification != null){
      FirebaseAPI().showSimpleNotification(
          title: message.notification!.title!,
          body: message.notification!.body!,
          payload: payloadData
      );
    } else {
      throw 'Notification not coming yet';
    }
  }
}
```

Kode Sumber 5. 19 Contoh Implementasi Firebase Module

```
final ioc = GetIt.instance;
final dio = Dio();


Future<void> initDependencies() async {
  ioc.registerSingletonAsync<SharedPreferences>(() =>
SharedPreferences.getInstance());

  ioc.registerSingletonWithDependencies<SharedPreferenceModule>(
    () => SharedPreferenceModule(pref: ioc<SharedPreferences>()),
    dependsOn: [SharedPreferences]
  );

  ioc.registerLazySingleton<Dio>(
    () => NetworkModule(dio: dio, baseURL: baseUrl).provideDio());

  // data sources
  ioc.registerLazySingleton<AuthApiService>(
    () => AuthApiService(
        ioc(),
    )
  );

  ioc.registerLazySingleton<AuthTeacherApiService>(
    () => AuthTeacherApiService(
      ioc()
    )
  );

  ioc.registerLazySingleton<NewsApiService>(
    () => NewsApiService(
      ioc(),
    ),
  );

  ioc.registerLazySingleton<EventApiService>(
    () => EventApiService(
      ioc(),
    ),
```

```
  );

  ioc.registerLazySingleton<DetailNewsApiService>(
    () => DetailNewsApiService(
      ioc()
    )
  );

  ioc.registerLazySingleton<DetailEventApiService>(
    () => DetailEventApiService(
      ioc()
    )
  );

  ioc.registerLazySingleton<PermissionApiService>(
    () => PermissionApiService(
      ioc()
    )
  );

  ioc.registerLazySingleton<NotificationApiService>(
    () => NotificationApiService(
        ioc()
    )
  );

  ioc.registerLazySingleton<TahfidzApiService>(
      () => TahfidzApiService(
        ioc()
      )
  );

  // repositories
  ioc.registerLazySingleton<AuthRepository>(
    () => AuthRepositoryImpl(
      authApiService: ioc(),
      authTeacherApiService: ioc(),
      preferenceModule: ioc()
    ),
  );

  ioc.registerLazySingleton<NewsRepository>(
    () => NewsRepositoryImpl(
      newsApiService: ioc()
    ),
  );

  ioc.registerLazySingleton<EventRepository>(
    () => EventRepositoryImpl(
      eventApiService: ioc()
    ),
  );

  ioc.registerLazySingleton<DetailNewsRepository>(
    () => DetailNewsRepositoryImpl(
      detailNewsApiService: ioc()
    )
  );

  ioc.registerLazySingleton<DetailEventRepository>(
    () => DetailEventRepositoryImpl(
      detailEventApiService: ioc()
    )
  );

  ioc.registerLazySingleton<PermissionRepository>(
    () => PermissionRepositoryImpl(
      permissionApiService: ioc()
```

```
    )
  );

  ioc.registerLazySingleton<NotificationRepository>(
    () => NotificationRepositoryImpl(
        notificationApiService: ioc()
    )
  );

  ioc.registerLazySingleton<TahfidzRepository>(
      () => TahfidzRepositoryImpl(
        tahfidzApiService: ioc()
      )
  );

  // use-cases
  ioc.registerLazySingleton<AuthUseCase>(
    () => AuthUseCase(ioc())
  );

  ioc.registerLazySingleton<AuthTeacherUseCase>(
    () => AuthTeacherUseCase(ioc())
  );

  ioc.registerLazySingleton<NewsUseCase>(
    () => NewsUseCase(ioc()),
  );

  ioc.registerLazySingleton<EventUseCase>(
    () => EventUseCase(ioc()),
  );

  ioc.registerLazySingleton<DetailNewsUseCase>(
    () => DetailNewsUseCase(ioc())
  );

  ioc.registerLazySingleton<DetailEventUseCase>(
    () => DetailEventUseCase(ioc())
  );

  ioc.registerLazySingleton<PermissionUseCase>(
    () => PermissionUseCase(ioc())
  );

  ioc.registerLazySingleton<HistoryPermissionUseCase>(
    () => HistoryPermissionUseCase(ioc())
  );

  ioc.registerLazySingleton<HrtHistoryPermissionUseCase>(
    () => HrtHistoryPermissionUseCase(ioc())
  );

  ioc.registerLazySingleton<HeadmasterHistoryPermissionUseCase>(
    () => HeadmasterHistoryPermissionUseCase(ioc())
  );

  ioc.registerLazySingleton<HrtGivePermissionApprovalUseCase>(
    () => HrtGivePermissionApprovalUseCase(ioc())
  );

  ioc.registerLazySingleton<HeadmasterGivePermissionApprovalUseCase>(
    () => HeadmasterGivePermissionApprovalUseCase(ioc())
  );

  ioc.registerLazySingleton<HrtOnGoingPermissionUseCase>(
    () => HrtOnGoingPermissionUseCase(ioc())
  );
```

```
  ioc.registerLazySingleton<HeadmasterOnGoingPermissionUseCase>(
    () => HeadmasterOnGoingPermissionUseCase(ioc())
  );

  ioc.registerLazySingleton<NotificationUseCase>(
    () => NotificationUseCase(ioc())
  );

  ioc.registerLazySingleton<UpdateDeviceNotificationUseCase>(
    () => UpdateDeviceNotificationUseCase(ioc())
  );

  ioc.registerLazySingleton<LogoutDeviceNotificationUseCase>(
    () => LogoutDeviceNotificationUseCase(ioc())
  );

  ioc.registerLazySingleton<GetListNotificationUseCase>(
    () => GetListNotificationUseCase(ioc())
  );

  ioc.registerLazySingleton<TahfidzUseCase>(
    () => TahfidzUseCase(ioc())
  );


  // blocs
  ioc.registerLazySingleton<AuthBloc>(
    () => AuthBloc(
      ioc(),
      ioc(),
      ioc(),
      ioc(),
      ioc()
    )
  );

  ioc.registerLazySingleton<NewsBloc>(
    () => NewsBloc(
      ioc(),
      ioc()
    )
  );

  ioc.registerLazySingleton<DetailNewsBloc>(
    () => DetailNewsBloc(
      ioc(),
      ioc()
    )
  );

  ioc.registerLazySingleton<PermissionBloc>(
    () => PermissionBloc(
      ioc(),
      ioc(),
    )
  );

  ioc.registerLazySingleton<TeacherPermissionBloc>(
    () => TeacherPermissionBloc(
      ioc(),
      ioc(),
      ioc()
    )
  );

  ioc.registerLazySingleton<HeadmasterPermissionBloc>(
    () => HeadmasterPermissionBloc(
      ioc(),
```

```
      ioc(),
      ioc()
    )
  );

  ioc.registerLazySingleton<NotificationBloc>(
    () => NotificationBloc(
      ioc(),
    )
  );

  ioc.registerLazySingleton<TahfidzBloc>(
    () => TahfidzBloc(
      ioc()
    )
  );

}
```

Kode Sumber 5. 20 Contoh Implementasi Dependency Injection Module

13. Utils

Utils merupakan tempat yang bertanggung jawab untuk menyimpan kelas-kelas yang akan membantu utilitas dari aplikasi. Contoh dari Utils adalah kelas Constant yang berfungsi untuk menyimpan variabel, fungsi maupun widget constant atau yang tidak akan pernah berubah dengan tujuan untuk memudahkan dan merapikan penulisan kode. Contoh lainnya yaitu adalah enumerations atau enums, enums berfungsi untuk memastikan konsistensi dan keakuratan dalam penulisan kode. Untuk lebih jelasnya dapat dilihat pada Kode Sumber 5.20 dan 5.21.

```
// ADDITIONAL
const emptyString = '';
const noneString = '-';

// INFORMATION
const appVersion = '1.0.0';

// BASE URL
const baseUrl = String.fromEnvironment('baseUrl');

// API-KEY
const apiKey = String.fromEnvironment('xApiKey');

// END POINT
// Auth Feature
const login = '/student_guardian/login';
const teacherLogin = '/employee/login';
// News Feature
const news = '/news';
const event = '/event';
const detailNews = '/news/{slug}';
const detailEvent = '/event/{slug}';
// Permission Feature
const permission = '/permission';
const permissionHistoryStudent = '/permission/student/{id}';
const permissionHistoryHrt = '/permission/hrt';
const permissionHistoryHeadmaster = '/permission/headmaster';
const onGoingPermissionHrt = '/permission/ongoing/hrt';
const onGoingPermissionHeadmaster = '/permission/ongoing/headmaster';
const permissionApprovalByHrt = '/permission/hrt/{permissionID}';
const permissionApprovalByHeadmaster=
'/permission/headmaster/{permissionID}';
// Profile Feature
const me = '/student_guardian/me';
// Notification Feature
```

```
const notification = '/notification';
const notificationLogout = '/notification/logout/{userID}';
const notificationHistory = '/notification/history';
// Tahfidz Feature
const tahfidzPrediction = '/tahfidz/predict/{studentID}';


// LINK
const instagramLink = 'https://www.instagram.com/nfbsbogorofficial';
const websiteLink = 'https://nfbs-bogor.sch.id';
const facebookLink =
'https://www.facebook.com/nurulfikriboardingschoolbogor';


// WIDGET
void showDialogError(BuildContext context, String message, String title){
  Constant.showGeneralDialog(
      context,
      CustomDialogReadOnlySingle(
          title: title,
          content: message
      ),
      true
  );
}

Widget loadingWidget(){
  return const Center(
    child: CircularProgressIndicator(),
  );
}

Widget blueLoadingWidget(){
  return const Center(
    child: CircularProgressIndicator(
      color: bluePrimary,
    ),
  );
}

class Constant {
  static Future showDevelopmentDialog(BuildContext context){
    return showDialog(
        context: context,
        builder: (BuildContext context){
          return const DevelopmentPage();
        }
    );
  }

  static String putToken(String token) => 'Bearer $token';

  static Future showGeneralDialog(BuildContext context, Widget dialog, bool
closeWithTapAnywhere){
    return showDialog(
        context: context,
        barrierDismissible: closeWithTapAnywhere,
        builder: (BuildContext context){
          return dialog;
        }
    );
  }

  static String convertDate(String timestampTz, String formatTime){
    if (timestampTz.isEmpty) {
      return '-';
    }
```

```
    try{
      DateTime dateTime = DateTime.parse(timestampTz);
      dateTime = dateTime.toLocal();
      final formattedDate = DateFormat(formatTime).format(dateTime);
      return formattedDate;
    } on FormatException {
      return 'Salah format';
    }
  }

  static Future dialogPrivacyPolicy(BuildContext context){
    return showDialog(
        context: context,
        builder: (BuildContext context){
          return Dialog(
            shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
            child: const Padding(
              padding: EdgeInsets.symmetric(vertical: 12, horizontal: 14),
              child: SingleChildScrollView(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      'Privacy Policy',
                      style: TextStyle(fontSize: 20, fontWeight:
FontWeight.w600),
                    ),
                    Text(
                      privacyPolicy,
                      style: TextStyle(fontSize: 12),
                      textAlign: TextAlign.justify,
                    )
                  ],
                ),
              ),
            ),
          );
        }
    );
  }
}
```

Kode Sumber 5. 21 Contoh Implementasi Constant

```
enum PermissionType{ permission, sick }

class PermissionEnums{
  static String getValueRequest(PermissionType type){
    switch (type){
      case PermissionType.permission:
        return 'PERMISSION';
      case PermissionType.sick:
        return 'SICK';
    }
  }

  static PermissionType getPermissionType(String text){
    switch (text){
      case 'Permission':
        return PermissionType.permission;
      case 'Sick':
        return PermissionType.sick;
      default:
        return PermissionType.permission;
```

```
    }
  }
}
```

Kode Sumber 5. 22 Contoh Implementasi Enums

14. Main

Main merupakan file atau kode utama yang pertama kali akan dijalankan, Disini akan diinisialisasikan Module, Routes, dan Theme. Pada Main biasanya juga akan diatur bagaimana tingkah laku aplikasi kita secara general, seperti hanya mengizinkan aplikasi dalam kondisi vertical dan tidak bisa dirotasi secara horizontal, pengizinan aplikasi dalam pemberian notifikasi. Untuk implementasi lengkapnya dapat dilihat pada Kode Sumber 5.23.

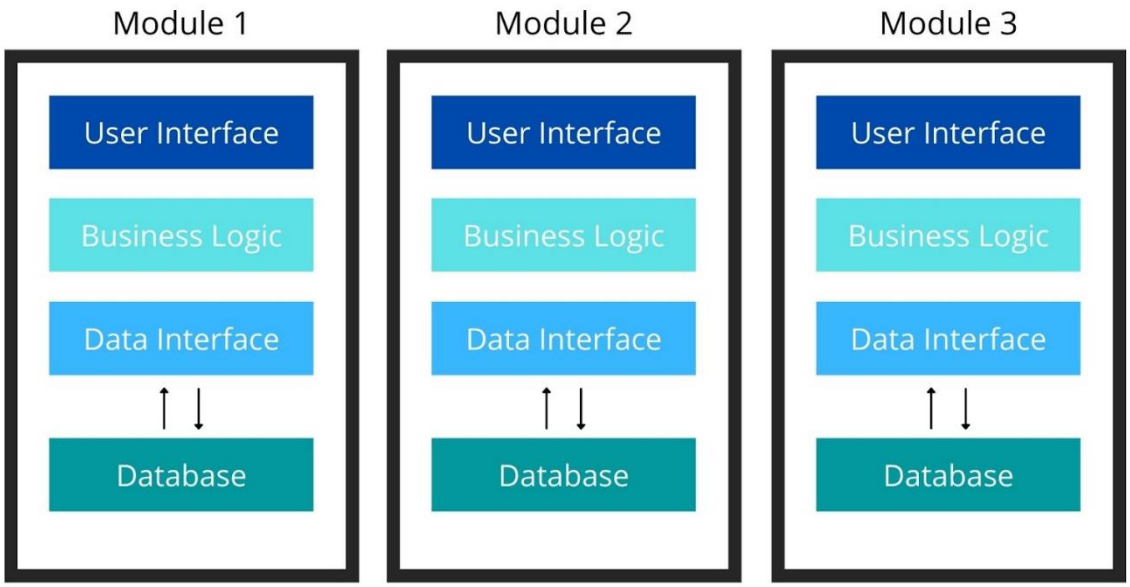
```
final navigatorKey = GlobalKey<NavigatorState>();

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  SystemChrome.setSystemUIOverlayStyle(
    const SystemUiOverlayStyle(
      statusBarColor: Colors.transparent,
      statusBarIconBrightness: Brightness.dark),
  );

  await SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
  ]);
  await initDependencies();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  await FirebaseAPI().initNotifications();
  FirebaseAPI.localNotificationInit();

  FirebaseMessaging.onBackgroundMessage((message)                                    =>
FirebaseAPI.firebaseBackgroundMessage(message));

  FirebaseMessaging.onMessage.listen((message) {
    String payloadData = jsonEncode(message.data);
    if(message.notification != null){
      FirebaseAPI().showSimpleNotification(
          title: message.notification!.title!,
          body: message.notification!.body!,
          payload: payloadData
      );
    }
  });

  // on background notification tapped
  FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {
    if (message.notification != null) {
      navigatorKey.currentState!.pushNamed('/Navigation', arguments: 2);
    }
  });

  // for handling in terminated state
  final RemoteMessage? message =
  await FirebaseMessaging.instance.getInitialMessage();

  if (message != null) {
    String payloadData = jsonEncode(message.data);
    FirebaseAPI().showSimpleNotification(
        title: message.notification!.title!,
```

```
        body: message.notification!.body!,
        payload: payloadData
    );
    Future.delayed(const Duration(seconds: 1), () {
      navigatorKey.currentState!.pushNamed('Navigation', arguments: 2);
    });
  }

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      navigatorKey: navigatorKey,
      title: 'BSS App',
      debugShowCheckedModeBanner: false,
      initialRoute: '/',
      theme: theme(),
      onGenerateRoute: AppRoutes.onGenerateRoutes,
    );
  }
}
```

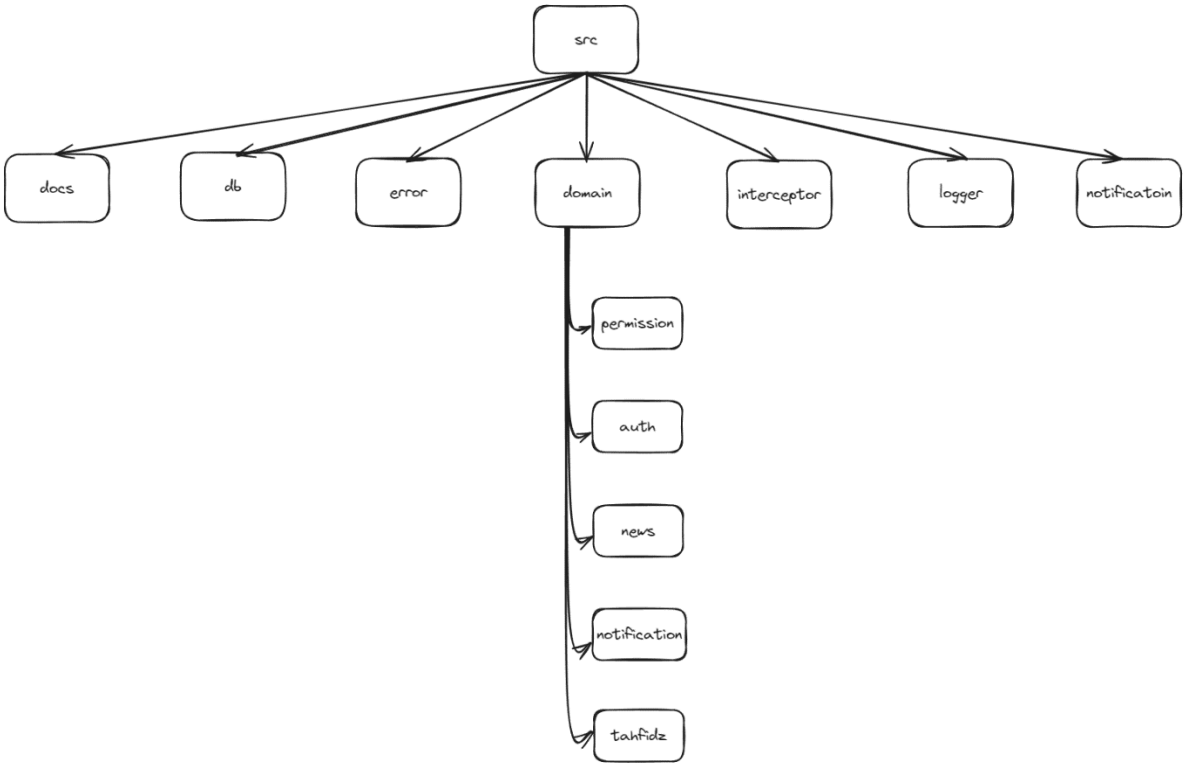
Kode Sumber 5. 23 Contoh Implementasi Main

5.2.2. Implementasi Back-End

Pada bagian ini akan dibahas mengenai implementasi pada back-end aplikasi. Back-end sistem aplikasi ini dibangun menggunakan sebuah arsitektur yaitu *Modular Monolith*. *Modular Monolith* adalah sebuah arsitektur membagi logika kita terlebih dahulu ke dalam modul-modul, dan setiap modul akan berdiri sendiri dan terisolasi. Kemudian, setiap modul harus memiliki logika bisnisnya sendiri serta kebutuhan yang mendukung logika bisnis tersebut. Menggunakan arsitektur ini akan membuat manajemen dependensi yang lebih baik dan juga struktur folder yang lebih mudah untuk dibaca, karena nama folder adalah sebuah feature, serta semua hal yang dibutuhkan untuk membuat API, seperti *routing*, *external API call*, dan sebagainya hanya berlaku untuk folder tersebut. Untuk lebih jelasnya dapat dilihat pada Gambar 5.3.



Gambar 5. 4 Gambaran Arsitektur Modular Monolith



Gambar 5. 5 Overview Dan Struktur Proyek BSS API

Berdasarkan ilustrasi dari Gambar 5.5, akan dijelaskan implementasi dari masing-masing komponen pada struktur *project*:

5.3. Docs

Di folder docs, hanya ada 1 file, yang berfungsi untuk *generate* swagger yang akan digunakan oleh *mobile developer* dalam masa pengembangan aplikasi. Ini membuat dokumentasi API menjadi lebih terstruktur dan rapi.

```
const options: swaggerJsDoc.Options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'BSS Mobile API - OpenAPI 3.0.3',
      description:
        '`BSS Mobile API` documentation based on the OpenAPI 3.0
specification.',
      contact: {
        name: 'NFBS Dev',
        url: 'https://github.com/nfbs-dev',
      },
      license: {
        name: 'Apache 2.0',
        url: 'http://www.apache.org/licenses/LICENSE-2.0.html',
      },
      version: '1.1.0',
    },
    components: {
      securitySchemes: {
```

```
        bearerAuth: {
          type: 'http',
          description: 'Bearer token to access these api endpoints',
          scheme: 'bearer',
          bearerFormat: 'JWT',
        },
        apiKeyAuth: {
          type: 'apiKey',
          description: 'API Key for sending data',
          in: 'header',
          name: 'x-api-key',
        },
      },
    },
  },
  apis: [
    './src/domain/auth/*.ts',
    './src/domain/news/*.ts',
    './src/domain/permission/*.ts',
    './src/domain/notification/*.ts',
  ],
};

const spec = swaggerJsDoc(options);

export const serveDocs = (app: Express, port: number) => {
  app.use('/docs', swaggerUI.serve, swaggerUI.setup(spec));

  app.get('docs.json', (_: Request, w: Response) => {
    w.setHeader('Content-Type', 'application/json');
    w.send(spec);
  });

  logger.info(`Docs available at http://localhost:${port}/docs`);
};
```

Kode Sumber 5. 24 Contoh Implementasi Docs

5.4. DB

DB/database adalah tempat konfigurasi dan pendefinisian segala hal yang akan berhubungan dengan database. Pada folder ini terdapat 2 file, instance.ts dan schema.ts.

```
interface DBConfig {
  host: string;
  user: string;
  password: string;
  dbName: string;
  port?: number;
}

export type DB = Kysely<Tables>;

export const getDBInstance = ({
  host,
  user,
  password,
  dbName,
  port,
}: DBConfig) => {
  const dialect = new MysqlDialect({
    pool: createPool({
      host: host,
      user: user,
      password: password,
      database: dbName,
```

```
      port: port || 5432,
    }),
  });

  const db = new Kysely<Tables>({
    dialect,
    plugins: [new ParseJSONResultsPlugin()],
    log(event): void {
      logger.debug({
        query: event.query.sql,
        param: event.query.parameters,
        time: `${event.queryDurationMillis} ms`,
      });
    },
  });

  return db;
};
```

Kode Sumber 5. 25 Contoh Implementasi DB folder instance.ts

Dapat dilihat bahwa instance.ts bertugas dalam hal konfigurasi ke database, dimana akan dihubungkan ke database yang tersedia dengan memberikan nama host, user, password, database name, dan port nya.

```
export type Generated<T> = T extends ColumnType<infer S, infer I, infer U>
  ? ColumnType<S, I | undefined, U>
  : ColumnType<T, T | undefined, T>;

export interface AcademicYears {
  academic_year: Generated<number | null>;
  created_at: Generated<Date | null>;
  default: Generated<'ACTIVE' | 'NOT ACTIVE' | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  order: Generated<number | null>;
  semester: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface Classes {
  class_name: Generated<string | null>;
  class_type: Generated<'ADDITION' | 'PRIMARY' | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  level: Generated<string | null>;
  updated_at: Generated<Date | null>;
}

export interface DataClasses {
  academic_year: Generated<number | null>;
  class_id: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  updated_at: Generated<Date | null>;
  user_id: Generated<number | null>;
}

export interface DataStudentClasses {
  academic_year: Generated<number | null>;
  class_id: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
```

```
  id: number;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface DataStudentDormitories {
  academic_year: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  dormitory_id: Generated<number | null>;
  id: number;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface DataStudentGuardianStudents {
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  student_guardian_id: Generated<number | null>;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface Dormitories {
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  dormitory_name: Generated<string | null>;
  id: number;
  unit: Generated<string | null>;
  updated_at: Generated<Date | null>;
}

export interface EmployeeMaster {
  created_at: Generated<Date>;
  created_by: Generated<string | null>;
  deleted_at: Generated<Date | null>;
  deleted_by: Generated<string | null>;
  employee_status_id: number;
  fungsional: Generated<string>;
  gender: Generated<'FEMALE' | 'MALE' | null>;
  golongan: Generated<string>;
  id: number;
  name: string;
  nipy: string;
  struktural: Generated<string>;
  telegram_id: Generated<number>;
  updated_at: Generated<Date>;
  updated_by: Generated<string | null>;
  user_id: Generated<number | null>;
}

export interface PermissionGohome {
  academic_year: Generated<number | null>;
  approval: Generated<number | null>;
  code: Generated<string | null>;
  created_at: Generated<Date | null>;
  date: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  description: Generated<string | null>;
  headmaster_chat_id: Generated<string | null>;
  hrt_chat_id: Generated<string | null>;
  id: Generated<number>;
  permission: Generated<string | null>;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}
```

```ts
export interface Schools {
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  level: Generated<string | null>;
  name: Generated<string | null>;
  updated_at: Generated<Date | null>;
}

export interface StudentGuardians {
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  email: Generated<string | null>;
  gender: Generated<string | null>;
  id: number;
  name: Generated<string | null>;
  no_whatsapp: Generated<string | null>;
  updated_at: Generated<Date | null>;
  user_id: Generated<number | null>;
}

export interface Students {
  alamat_ktp: Generated<string | null>;
  alamat_tinggal: Generated<string | null>;
  alasan_keluar: Generated<string | null>;
  anak_ke: Generated<number | null>;
  asal_sekolah: Generated<string | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  email: Generated<string | null>;
  id: number;
  jenis_kelamin: string;
  jenis_pendaftaran: Generated<number | null>;
  jumlah_saudara: Generated<number | null>;
  jurusan: Generated<'SMA' | 'SMP' | null>;
  kecamatan_id: Generated<string | null>;
  keluar_karena: Generated<string | null>;
  kelurahan_id: Generated<string | null>;
  kewarganegaraan: Generated<'WNA' | 'WNI' | null>;
  kode_pos: Generated<string | null>;
  kota_kabupaten_id: Generated<string | null>;
  nama_ayah: Generated<string | null>;
  nama_ibu: Generated<string | null>;
  nama_lengkap: Generated<string | null>;
  nama_wali: Generated<string | null>;
  nik: Generated<string | null>;
  nik_ayah: Generated<string | null>;
  nik_ibu: Generated<string | null>;
  nik_wali: Generated<string | null>;
  nis: Generated<string | null>;
  nisn: Generated<string | null>;
  no_hp: Generated<string | null>;
  no_hp_ayah: Generated<string | null>;
  no_hp_ibu: Generated<string | null>;
  no_hp_wali: Generated<string | null>;
  no_ijazah_lama: Generated<string | null>;
  no_kartu_keluarga: Generated<string | null>;
  no_peserta_ujian_lama: Generated<string | null>;
  no_registrasi_akta_lahir: Generated<string | null>;
  no_skhun: Generated<string | null>;
  pekerjaan_ayah: Generated<string | null>;
  pekerjaan_ibu: Generated<string | null>;
  pekerjaan_wali: Generated<string | null>;
  pendidikan_ayah: Generated<string | null>;
  pendidikan_ibu: Generated<string | null>;
  pendidikan_wali: Generated<string | null>;
  penghasilan_ayah: Generated<number | null>;
  penghasilan_ibu: Generated<number | null>;
```

```
  penghasilan_wali: Generated<number | null>;
  photo: Generated<string | null>;
  province_id: Generated<string | null>;
  rt: Generated<string | null>;
  rw: Generated<string | null>;
  status: Generated<'ACTIVE' | 'DROP OUT' | 'PASSED' | null>;
  tahun_angkatan: Generated<string | null>;
  tanggal_keluar: Generated<Date | null>;
  tanggal_lahir: Generated<Date | null>;
  tanggal_lahir_ayah: Generated<Date | null>;
  tanggal_lahir_ibu: Generated<Date | null>;
  tanggal_lahir_wali: Generated<Date | null>;
  tanggal_masuk: Generated<Date | null>;
  tempat_lahir: Generated<string | null>;
  updated_at: Generated<Date | null>;
  user_id: Generated<number | null>;
}

export interface Users {
  category: Generated<
    'EMPLOYEE' | 'NEW' | 'STUDENT' | 'STUDENT GUARDIAN' | 'THIRD PARTY' |
null
  >;
  created_at: Generated<Date | null>;
  email: Generated<string | null>;
  email_verified_at: Generated<Date | null>;
  gender: Generated<'FEMALE' | 'MALE' | null>;
  google_id: Generated<string | null>;
  id: number;
  name: string;
  password: string;
  password_last_update: Generated<Date | null>;
  photo: Generated<string | null>;
  remember_token: Generated<string | null>;
  status: Generated<'ACTIVE' | 'NOT ACTIVE' | null>;
  updated_at: Generated<Date | null>;
  username: string;
}

export interface DataSchools {
  academic_year: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  headmaster: Generated<number | null>;
  id: number;
  principal_curriculum: Generated<number | null>;
  principal_student_affairs: Generated<number | null>;
  school_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface NotificationUsers {
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  device_id: Generated<string | null>;
  fcm_token: string;
  id: Generated<number>;
  is_logout: number;
  user_id: number;
}

export interface PermissionNotificationHistories {
  body: string;
  created_at: Generated<Date>;
  id: Generated<number>;
  permission_id: number;
  title: string;
  user_id: number;
```

```ts
}

export interface AlquranMemorize {
  academic_year: Generated<number | null>;
  alquran_schedule_id: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  end_verse: Generated<number | null>;
  id: number;
  row_total: Generated<number | null>;
  start_verse: Generated<number | null>;
  student_id: Generated<number | null>;
  surah_alquran_id: Generated<number | null>;
  type_memorize: Generated<string | null>;
  updated_at: Generated<Date | null>;
  user_id: Generated<number | null>;
}

export interface SurahAlquran {
  arti: Generated<string | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  deskripsi: Generated<string | null>;
  id: number;
  jumlah_ayat: Generated<number | null>;
  nama: Generated<string | null>;
  nama_latin: Generated<string | null>;
  tempat_turun: Generated<string | null>;
  updated_at: Generated<Date | null>;
}

export interface DataStudentAlquranTeachers {
  academic_year: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
  user_id: Generated<number | null>;
}

export interface DataStudentAlquranLevels {
  academic_year: Generated<number | null>;
  created_at: Generated<Date | null>;
  deleted_at: Generated<Date | null>;
  id: number;
  level: Generated<string | null>;
  meeting_target: Generated<number | null>;
  row_target: Generated<number | null>;
  student_id: Generated<number | null>;
  updated_at: Generated<Date | null>;
}

export interface Tables {
  academic_years: AcademicYears;
  alquran_memorize: AlquranMemorize;
  classes: Classes;
  data_schools: DataSchools;
  data_classes: DataClasses;
  data_student_alquran_levels: DataStudentAlquranLevels;
  data_student_alquran_teachers: DataStudentAlquranTeachers;
  data_student_classes: DataStudentClasses;
  data_student_dormitories: DataStudentDormitories;
  data_student_guardian_students: DataStudentGuardianStudents;
  dormitories: Dormitories;
  employee_master: EmployeeMaster;
  notification_users: NotificationUsers;
  permission_gohome: PermissionGohome;
```

```
  permission_notification_histories: PermissionNotificationHistories;
  schools: Schools;
  student_guardians: StudentGuardians;
  students: Students;
  surah_alquran: SurahAlquran;
  users: Users;
}
```

Kode Sumber 5. 26 Contoh Implementasi DB folder schema.ts

Ini adalah daftar isi table yang dituliskan ulang ke dalam schema.ts. Penulisan ulang ini bertujuan untuk membuat query yang dilakukan nanti di repository menjadi safety (tidak ada kesalahan nama kolom pada tabel).

5.5. Error

Folder error akan bertanggung jawab dalam urusan error yang terjadi pada aplikasi. Setiap request berpotensi menghasilkan error dengan tipe berbeda. Folder ini adalah jenis error yang mungkin terjadi pada aplikasi. Contoh seperti berikut

```
export class QueryError extends Error {
  constructor(
    private readonly errorReason: string,
    private readonly serverError: boolean
  ) {
    super(errorReason);
  }

  public get reason() {
    return this.serverError
      ? 'Something wrong. Please try again later.'
      : this.errorReason;
  }

  public get code() {
    return this.serverError ? 500 : 400;
  }
}
```

Kode Sumber 5. 27 Contoh Implementasi salah satu file pada folder error

Dapat dilihat bahwa salah satu sampel kode diatas adalah potensi error yang terjadi saat proses request, yaitu error pada query di SQL. Error ini bisa saja sering terjadi, maka dari itu perlu sebuah tipe error seperti ini.

5.6. Interceptor

Interceptor ataupun *middleware* adalah folder yang berisi sekumpulan file yang mampu menginterupsi *lifecycle* pada request. Dalam satu kali proses request, ada beberapa proses yang repetitif, contohnya manajemen error. Jika pada proses request terdapat error, maka interceptor dapat menghandle kasus ini. Contohnya seperti berikut.

```
import { QueryError } from '@/error/QueryError';
import { logger } from '@/logger/logger';
import { Request, Response, NextFunction } from 'express';
import { ZodError } from 'zod';

export const errorHandler = async (
  err: Error,
  req: Request,
  res: Response,
```

```
  next: NextFunction
) => {
  if (err instanceof ZodError) {
    return res.status(422).send({
      msg: 'failed validation',
      errors: customizeZodError(err),
    });
  }

  if (err instanceof SyntaxError) {
    return res.status(400).send({
      msg: err.message,
    });
  }

  if (err instanceof QueryError) {
    logger.error(`[QUERY ERROR]: ${err.message}`);

    return res.status(err.code).send({
      msg: err.reason,
    });
  }

  logger.error({
    error: `[UNHANDLED ERROR]: ${err.message}`,
    type_error: typeof err,
    request_uri: req.path,
    stack_trace: err.stack,
  });

  res.status(500).send({
    msg: 'Internal server error. Please try again later.',
  });

  next(err);
};

const customizeZodError = (errors: ZodError) => {
  const issues = errors.issues;
  const errs: { [key: string]: string } = {};

  issues.forEach(({ path, message }) => {
    const key = path[0];
    errs[key] = message;
  });

  return errs;
};
```

Kode Sumber 5. 28 Contoh Implementasi salah satu file pada folder interceptor

Dapat dilihat bahwa salah satu sampel file kode diatas adalah bagaimana cara untuk manajemen error yang terjadi, mulai dari error tipe status kode 422, 400, ataupun 500. Setiap error yang mungkin akan terjadi, bisa berbeda juga bagaimana cara mengaturnya.

5.7. Logger

Logger sangat penting dalam sebuah aplikasi, karena bisa berguna dalam berbagai hal, seperti debugging dan juga tracing jika ada error yang tidak diduga. Berikut contoh implementasi nya.

```
const { format } = winston;

const consoleTransport = new winston.transports.Console();
```

```
const transports: Transport[] = [consoleTransport];

if (ENV.telegram.channel || ENV.telegram.token) {
  const telegramTransport = new TelegramTransport({
    token: ENV.telegram.token,
    chatId: ENV.telegram.channel,
    level: ENV.stage === 'DEV' ? 'silly' : 'warn',
    disableNotification: false,
  });

  transports.push(telegramTransport);
} else {
  console.warn(
    'Telegram is not set because either TELEGRAM_CHANNEL or
TELEGRAM_TOKEN is empty'
  );
}

export const logger = winston.createLogger({
  level: ENV.stage === 'DEV' ? 'silly' : 'warn',
  format:       format.combine(format.ms(),       format.timestamp(),
format.json()),
  defaultMeta: {
    service: 'bss-api-mobile',
    node_version: process.version,
    env: ENV.stage,
  },
  transports: transports,
});
```

Kode Sumber 5.29 Contoh Implementasi pada Logger

5.8. Notification

Di BSS API, terdapat fitur untuk mengirimkan notifikasi. Maka dari itu disini adalah konfigurasi dan implementasi bagaimana cara mengirimkan notifikasi kepada user beserta isi notifikasi nya. Berikut adalah implementasinya.

```
const INVALID_TOKEN = '/messaging/invalid-argument';

export interface MessageParam {
  title: string;
  body: string;
  data?: {
    [key: string]: string;
  };
  deviceToken: string[];
  imgUrl?: string;
}

export const sendNotification = async (param: MessageParam) => {
  const msg: MulticastMessage = {
    notification: {
      title: param.title,
      body: param.body,
      imageUrl: param.imgUrl,
    },
    data: param.data,
    tokens: param.deviceToken,
    android: {
      notification: {
        channelId: 'pushNotification',
        priority: 'high',
        visibility: 'public',
```

```
            },
          },
        };

        try {
          const result = await getMessaging().sendEachForMulticast(msg);

          if (result.failureCount > 0) {
            result.responses.forEach((response, index) => {
              if (!response.success) {
                const errorCode = response.error?.code || '';

                if (INVALID_TOKEN.includes(errorCode)) {
                  logger.warn(`Wrong token detected:,
${param.deviceToken[index]}`);
                } else if (toleratedError.includes(errorCode)) {
                  retry(param, param.deviceToken[index]);
                }
              }
            });
          }
        } catch (error) {
          if (error instanceof Error) {
            logger.error(`[FCM sendNotification]: ${error.message}`);
          } else {
            logger.error(`[FCM sendNotification]: ${error}`);
          }
        }
      };

      const retry = async (param: MessageParam, failedToken: string) => {
        const msg = {
          notification: {
            title: param.title,
            body: param.body,
            imgUrl: param.imgUrl,
          },
          token: failedToken,
        };

        try {
          await backOff(() => getMessaging().send(msg), {
            numOfAttempts: 5,
            startingDelay: 1000,
            timeMultiple: 3,
          });
        } catch (error) {
          if (error instanceof Error) {
            logger.warn(`[FCM retry]: ${error.message}`);
          } else {
            logger.warn(`[FCM retry]: ${error}`);
          }
        }
      };

      const toleratedError = [
        'messaging/unknown-error',
        'messaging/internal-error',
        'messaging/server-unavailable',
      ];
```

Kode Sumber 5.30 Contoh Implementasi pada notification

Dapat dilihat bahwa disini notifikasi menggunakan teknologi Firebase Notification (dan tidak akan pernah berubah). Semua error yang mungkin didapat juga diatur disini beserta

tipe error yang mungkin akan terjadi.

5.9. Domain

Pada domain, ada banyak sekali kemungkinan file yang ada, tergantung pada kompleksitas kode yang ada. Namun yang pasti akan ada 4 file berikut.

1. Routes

Routes adalah daftar rute yang tersedia untuk fitur yang ada pada folder tersebut. Berikut implementasi untuk fitur auth.

```
const authRouter = (router: Router, db: DB, jwtSecretKey: string) => {
  const repo = authRepo(db);
  const handler = authHandler(repo, jwtSecretKey);
  router.post('/student_guardian/login', handler.studentGuardianLogin);
  router.post('/employee/login', handler.employeeLogin);
  router.get('/me', handler.whoami);
};

export default authRouter;
```

Kode Sumber 5.31 Contoh Implementasi Domain pada file routes.ts

2. Types

Types adalah tipe data dan juga interface yang akan digunakan pada seluruh file yang tersedia pada 1 fitur tersebut. Berikut implementasi untuk fitur auth.

```
export interface IAuthRepo {
  getStudentGuardian(email: string): Promise<StudentGuardianUser | null>;
  getEmployee(email: string): Promise<TeacherUser | HeadmasterUser | null>;
}

export type DataSchoolResult = {
  name: string | null;
  level: string | null;
  headmasterID: string | null;
};

export type StudentGuardianUser = {
  userID: number;
  name: string;
  category:
    | 'EMPLOYEE'
    | 'NEW'
    | 'STUDENT'
    | 'STUDENT GUARDIAN'
    | 'THIRD PARTY'
    | null;
  phoneNumber: string | null;
  email: string | null;
  students: [
    {
      id: number | null;
      name: string | null;
      nis: string | null;
      nisn: string | null;
      email: string | null;
      phoneNumber: string | null;
      gender: string | null;
      birthPlace: string | null;
      birthDate: string | null;
```

```
      address: string | null;
      dormitory: {
        id: number | null;
        name: string | null;
        unit: string | null;
      } | null;
      class: {
        id: number | null;
        name: string | null;
        type: 'ADDITION' | 'PRIMARY' | null;
        level: string | null;
        teacherName: string | null;
        teacherEmail: string | null;
      } | null;
    }
  ];
};

export type TeacherUser = {
  userID: number | null;
  employeeID: number | null;
  name: string;
  nipy: string;
  role: string;
  class: {
    id: number | null;
    name: string | null;
    type: 'ADDITION' | 'PRIMARY' | null;
    level: string | null;
  } | null;
  totalStudents: number;
};

export type HeadmasterUser = {
  userID: number | null;
  employeeID: number | null;
  schoolID: number | null;
  schoolName: string | null;
  schoolLevel: string | null;
  role: string;
  name: string | null;
  nipy: string | null;
  struktural: string | null;
  fungsional: string | null;
};
```

Kode Sumber 5.32 Contoh Implementasi Domain pada file types.ts

3. Handler

Handler adalah business logic dari fitur yang tersedia. Disini adalah sebuah instruksi bagaimana request harus diolah. Berikut implementasi untuk fitur auth.

```
export const authHandler = (repo: IAuthRepo, jwtSecretKey: string) => {
  return {
    studentGuardianLogin: async (r: Request, w: Response) => {
      const data = await loginRequest.parseAsync(r.body);
      const result = await repo.getStudentGuardian(data.email);

      if (!result) {
        return w.status(400).send({ msg: 'Invalid credentials' });
      }

      const token = signJWT(jwtSecretKey, result);

      return w.status(200).send({
```

```
        profile: result,
        token: token,
      });
    },

    employeeLogin: async (r: Request, w: Response) => {
      const data = await loginRequest.parseAsync(r.body);
      const result = await repo.getEmployee(data.email);

      if (!result) {
        return w.status(400).send({ msg: 'Invalid credentials' });
      }

      const token = signJWT(jwtSecretKey, result);

      return w.status(200).send({
        profile: result,
        token: token,
      });
    },

    whoami: async (r: Request, w: Response) => {
      if(r.studentGuardianUser) {
        return w.status(200).send(r.studentGuardianUser);
      }

      if(r.teacherUser) {
        return w.status(200).send(r.teacherUser);
      }

      if(r.headmasterUser) {
        return w.status(200).send(r.headmasterUser);
      }

      return w.status(401).send({ msg: 'Unknown user' });
    },
  };
};
```

Kode Sumber 5.33 Contoh Implementasi Domain pada file handler.ts

4. Repo

Repo ataupun repository, adalah bagaimana cara mengambil data di *database*. Disini data dari *database* akan diambil sesuai kebutuhan *business logic* nya. Berikut implementasi untuk fitur auth.

```
export const authRepo = (db: DB): IAuthRepo => {
  return {
    getStudentGuardian: async (email: string) => {
      const result = await db
        .selectFrom('users')
        .where('users.email', '=', email)
        .innerJoin('student_guardians',                    'users.id',
'student_guardians.user_id')
        .innerJoin(
          'data_student_guardian_students',
          'student_guardians.id',
          'data_student_guardian_students.student_guardian_id'
        )
        .innerJoin(
          'students',
          'students.id',
          'data_student_guardian_students.student_id'
        )
```

```
      .innerJoin(
        'data_student_dormitories',
        'students.id',
        'data_student_dormitories.student_id'
      )
      .innerJoin(
        'dormitories',
        'data_student_dormitories.dormitory_id',
        'dormitories.id'
      )
      .innerJoin(
        'data_student_classes',
        'students.id',
        'data_student_classes.student_id'
      )
      .innerJoin(
        'data_classes',
        'data_classes.class_id',
        'data_student_classes.class_id'
      )
      .innerJoin('classes', 'data_student_classes.class_id', 'classes.id')
      .innerJoin('users as teacher', 'data_classes.user_id', 'teacher.id')
      .select((eb) => [
        'users.id as userID',
        'users.name',
        'users.category',
        'student_guardians.no_whatsapp as phoneNumber',
        'users.email',
        sql`JSON_ARRAYAGG(${jsonBuildObject({
          id: eb.ref('students.id'),
          name: eb.ref('students.nama_lengkap'),
          nis: eb.ref('students.nis'),
          nisn: eb.ref('students.nisn'),
          email: eb.ref('students.email'),
          phoneNumber: eb.ref('students.no_hp'),
          gender: eb.ref('students.jenis_kelamin'),
          birthPlace: eb.ref('students.tempat_lahir'),
          birthDate: eb.ref('students.tanggal_lahir'),
          address: eb.ref('students.alamat_tinggal'),
          dormitory: jsonBuildObject({
            id: eb.ref('dormitories.id'),
            name: eb.ref('dormitories.dormitory_name'),
            unit: eb.ref('dormitories.unit'),
          }),
          class: jsonBuildObject({
            id: eb.ref('classes.id'),
            name: eb.ref('classes.class_name'),
            level: eb.ref('classes.level'),
            type: eb.ref('classes.class_type'),
            teacherName: eb.ref('teacher.name'),
            teacherEmail: eb.ref('teacher.email'),
          }),
        })})`.as('students'),
      ])
      .executeTakeFirst();

    const isEveryValueNull =
      result && Object.values(result).every((value) => value === null);

    if (isEveryValueNull || isEveryValueNull === undefined) {
      return null;
    }

    return result as StudentGuardianUser;
  },

  getEmployee: async (email: string) => {
    const result = await db
```

```
      .selectFrom('users')
      .innerJoin('employee_master', 'users.id', 'employee_master.user_id')
      .where('email', '=', email)
      .select([
        'users.id as userID',
        'employee_master.id as employeeID',
        'employee_master.name',
        'employee_master.nipy',
        'employee_master.struktural',
        'employee_master.fungsional',
      ])
      .executeTakeFirst();

    if (!result) {
      return null;
    }

    const dataSchools = await db
      .selectFrom('data_schools')
      .innerJoin('schools', 'data_schools.school_id', 'schools.id')
      .select([
        'data_schools.headmaster as headmasterID',
        'schools.name as name',
        'schools.level as level',
        'schools.id as schoolID',
      ])
      .execute();

    if (!dataSchools) {
      throw Error('Data schools not found');
    }

    for (const dataSchool of dataSchools) {
      if (dataSchool.headmasterID === result.employeeID) {
        return Object.assign(result, {
          role: 'HEADMASTER',
          schoolName: dataSchool.name,
          schoolLevel: dataSchool.level,
          schoolID: dataSchool.schoolID,
        }) as HeadmasterUser;
      }
    }

    const teacher = await db
      .with('class', (db) =>
        db
          .selectFrom('data_classes')
          .innerJoin('classes', 'data_classes.class_id', 'classes.id')
          .where('data_classes.user_id', '=', result.userID)
          .select([
            'classes.id as id',
            'classes.class_name as name',
            'classes.class_type as type',
            'classes.level as level',
          ])
      )
      .selectFrom('class')
      .select([
        'name',
        'level',
        'type',
        'id',
        sql<number>`(SELECT  COUNT(student_id)  FROM  data_student_classes
where class_id = class.id)`.as(
          'totalStudents'
        ),
      ])
      .executeTakeFirst();
```

```
      if (!teacher) {
        return null;
      }

      return {
        userID: result.userID,
        employeeID: result.employeeID,
        name: result.name,
        nipy: result.nipy,
        role: 'TEACHER',
        class: {
          id: teacher.id,
          name: teacher.name,
          type: teacher.type,
          level: teacher.level,
        },
        totalStudents: teacher.totalStudents,
      } as TeacherUser;
    },
  };
};
```

Kode Sumber 5.34 Contoh Implementasi Domain pada file repo.ts

5.2.3. Implementasi Machine Learning

Pengimplementasian machine learning pada sistem yang dibuat adalah untuk memprediksi jumlah baris ayat Alquran yang mungkin dapat dihafalkan oleh siswa. Setelah menerima data, kami menganalisis data yang diberikan dan memutuskan untuk menggunakan model yang dikembangkan oleh Facebook yaitu Prophet karena bentuk data yang diberikan berbentuk time series .

Dalam pembangunan model, kami membangun model dengan beberapa tahap :

1. Install Requirment

```
!pip install Cython
!pip install pystan prophet
```

Kode Sumber 5.35 Install Requirement

perintah yang dapat digunakan untuk menginstal beberapa paket Python yang diperlukan untuk menggunakan Prophet. Cython berfungsi untuk meningkatkan kecepatan eksekusi kode, pystan merupakan bahasa pemodelan probabilistik yang digunakan untuk menentukan model statistik yang rumit dan Prophet merupakan algoritma Facebook yang sudah dijelaskan tadi.

2. Preprocessing

```
import pandas as pd
from prophet import Prophet
file_path = "D:/KP/DataHafalanSiswa.csv"
```

```
df = pd.read_csv(file_path)
Df.dtypes
```

Kode Sumber 5.36 Memasukkan File Data

Kita akan mengambil data yang akan kita olah ke dalam df, setelah itu kita bisa meihat tipe data dari masing-masing kolom. Namun date masih bertipe objek dan untuk memastikan jika semua ayat dan jumlah baris beripe integer kita bisa mengubahnya menjadi tipe data yang kita butuhkan.

```
df['Date']=pd.to_datetime(df['Date'])
df['jumlah_baris']=pd.to_numeric(df['jumlah_baris'])
df['user_id']=pd.to_numeric(df['user_id'])
```

Kode Sumber 5.37 Mengubah Tipe Data

Disini kita mengubah semua *Date* yang awalnya bertipe *data object* menjadi *date time*, *jumlah_baris* menjadi *numeric* dan *id* dari *user* menjadi *numeric*.

```
nama_lengkap              object
Date            datetime64[ns]
sesi                      object
presensi                  object
ayat_terakhir              int64
jumlah_baris               int64
user_id                    int64
```

Gambar 5. 6. Tipe Data Kolom Setelah

Pada Gambar 5.6 diperlihatkan bahwa *date* sudah bertipe data datetime64 yang sesuai.Setelah memastikan semua tipe datanya sudah kita ubah, kita akan membuat sebuah variabel baru untuk menampung semua kolom (tabel baru). Kita menampungnya ke dalam variabel baru agar tidak merusak/mengubah file data yang lama.

```
kolom = ['sesi',
'presensi','ayat_terakhir','user_id','nama_lengkap']
frame = df[df['user_id'] == student_id].copy()
frame.drop(kolom, axis=1, inplace=True)
frame.columns = ['ds', 'y']
```

Kode Sumber 5. 38 Membuang Kolom Tak Dibutuhkan

Pada source code di atas kita akan mengubah kolom yang tidak dibutuhkan yang berada pada variabel kolom. Dimana berarti hanya bersisa Date dan jumlah baris yang dihafalkan.

3. Predicting

```
m = Prophet(interval_width=0.95)
```

```
training_run = m.fit(frame)
future = m.make_future_dataframe(periods=30, freq='D')
forecast = m.predict(future)
```

Kode Sumber 5. 39 Predicting

Kami membuat model di dalam variabel m dari kelas Prophet. Kami menetapkan parameter interval_width dengan nilai 0.95, yang menentukan Confidence Interval untuk prediksi. Setelah itu akan melakukan training model kepada data yang ada di dalam frame. Lalu kami membuat DataFrame future yang berisi tanggal-tanggal masa depan yang ingin diprediksi. Kami membuat prediksi pada tanggal-tanggal untuk 30 hari ke depan (periods=30) dengan frekuensi harian (freq='D'). Setelah itu hasil prediksi disimpan dalam DataFrame forecast.

4. Result

```
sum_yhat = forecast['yhat'].sum()
sum_yhat = int(sum_yhat)
length_yhat = len(forecast['yhat'])
predict_result = f"Jumlah baris yang mungkin akan bisa
dihafalkan dalam waktu {length_yhat} hari adalah {sum_yhat}
baris"
print (predict_result)
```

Kode Sumber 5. 40 Result

Karena hasil prediksi yang ada di dalam forecast merupakan hasil dari hafalan perhari maka kita harus menjumlahkan semua baris yang ada. Setelah itu kita itu akan mengeluarkan prediksi yang sudah kita buat di dalam variabel predict_result. Back end akan memanggil hasil dari prediksi menggunakan variabel predict_result.

5. Export Model

```
import pickle
with open('model.pkl', 'wb') as file:
    pickle.dump(forecast, file)
```

Kode Sumber 5. 41 Export

Kami telah menyimpan model prediksi yang dihasilkan oleh Prophet (DataFrame forecast) dalam format biner ke dalam file 'model.pkl'. Kita dapat menggunakan file ini di masa mendatang untuk memuat kembali model dan melakukan prediksi tanpa harus melatih ulang model. Untuk meload model, kita dapat menggunakan fungsi pickle.load() dan membaca kembali DataFrame dari file.

5.3. Tampilan Antarmuka

Berdasarkan implementasi yang telah diterapkan baik dalam front-end, back-end, maupun machine learning, maka didapatkannya hasil akhir berupa sebuah aplikasi mobile dengan tampilan akhir sebagai berikut. Hal ini dapat dilihat pada Gambar 5.6 hingga 5.13.

1. Tampilan Antarmuka Halaman Awal

Gambar 5. 7. Tampilan Antarmuka Halaman Awal

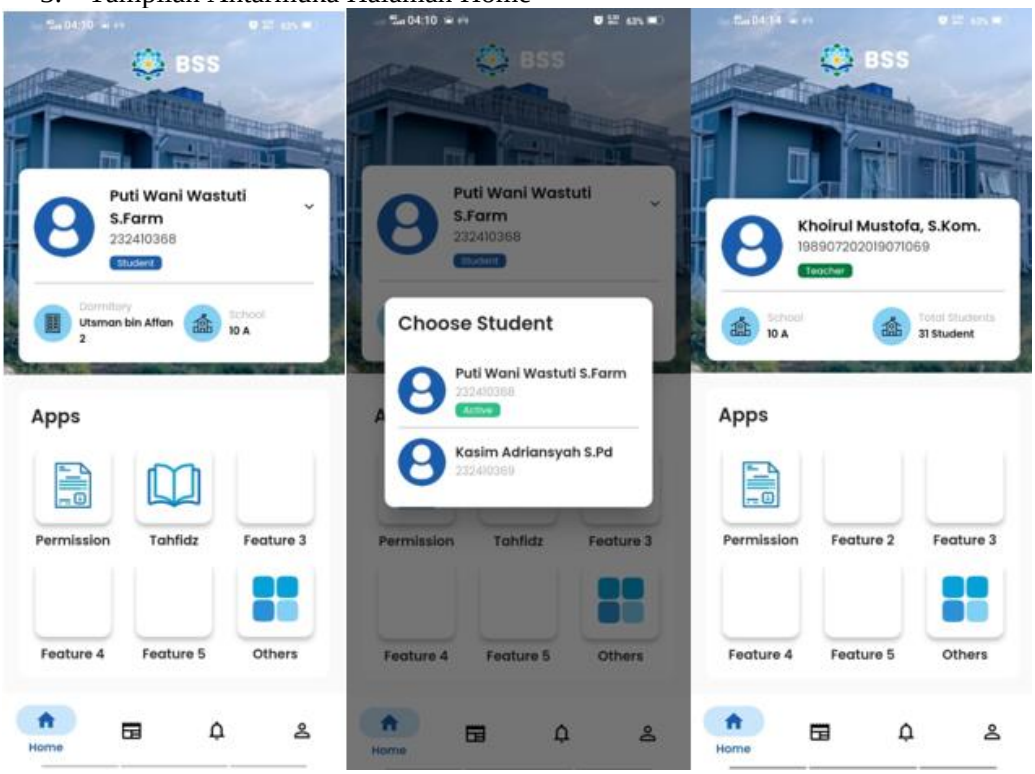
Pada gambar 5.7 ditampilkan halaman awal saat membuka aplikasi BSS Mobile yang telah dikembangkan.

2. Tampilan Antarmuka Halaman Login

Gambar 5. 8. Tampilan Antarmuka Halaman Login

Pada Gambar 5.8 ditampilkan halaman login untuk aplikasi BSS Mobile yang mempunyai 2 *roles.*

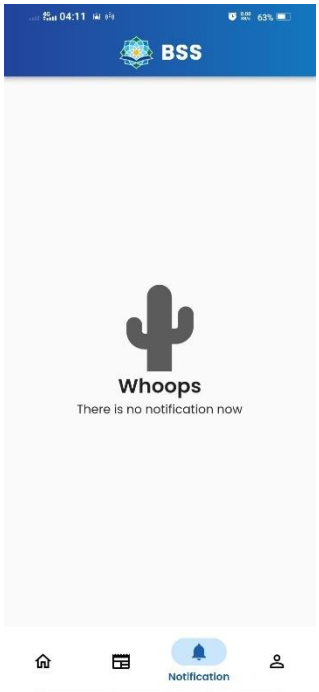
3. Tampilan Antarmuka Halaman Home



Gambar 5. 9. Tampilan Antarmuka Halaman Home

Pada Gambar 5.9 terdapat 3 halaman yang menampilkan antarmuka halaman home dan memilih santri.

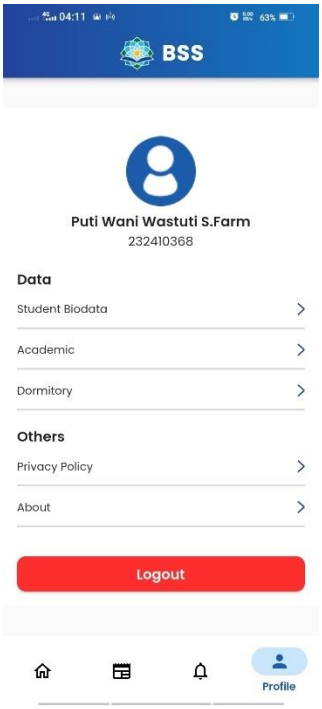
4. Tampilan Antarmuka Halaman Notification



Gambar 5. 10. Tampilan Antarmuka Halaman Notification

Pada Gambar 5.10 ditampilkan antarmuka halaman notifikasi yang masih kosong, namun akan terisi jika ada perizinan baru.

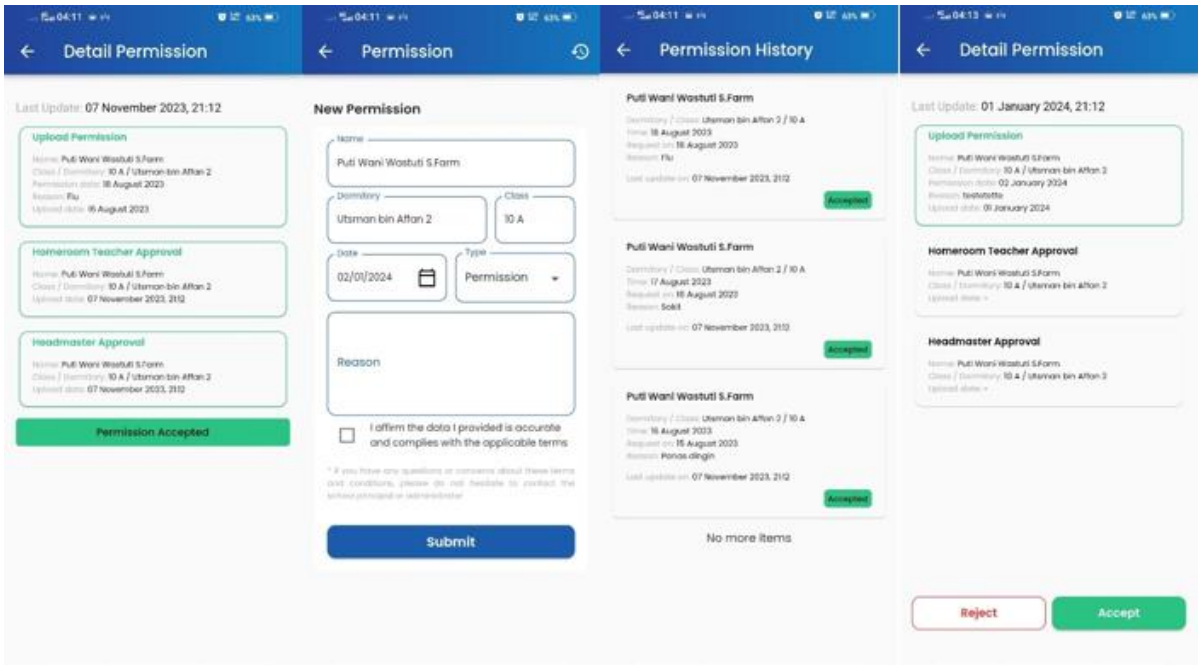
5. Tampilan Antarmuka Halaman Profile



Gambar 5. 11. Tampilan Antarmuka Halaman Profile

Pada Gambar 5.11 terdapat halaman antarmuka *profile* yang menampung informasi tentang pengguna.

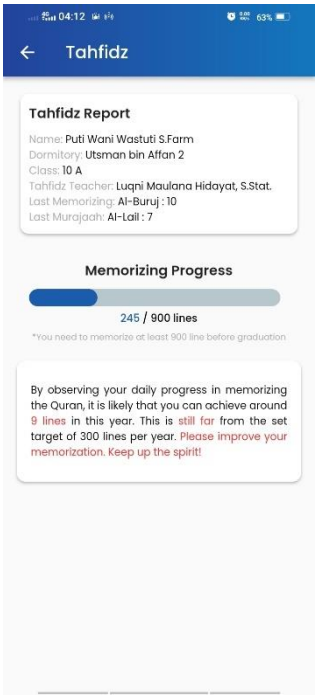
6. Tampilan Antarmuka Halaman Permission



Gambar 5. 12. Tampilan Antarmuka Halaman Permission

Pada Gambar 5.12 terdapat 4 halaman yang merepresentasikan alur perizinan untuk santri yang kemudian akan disetujui oleh wali santri

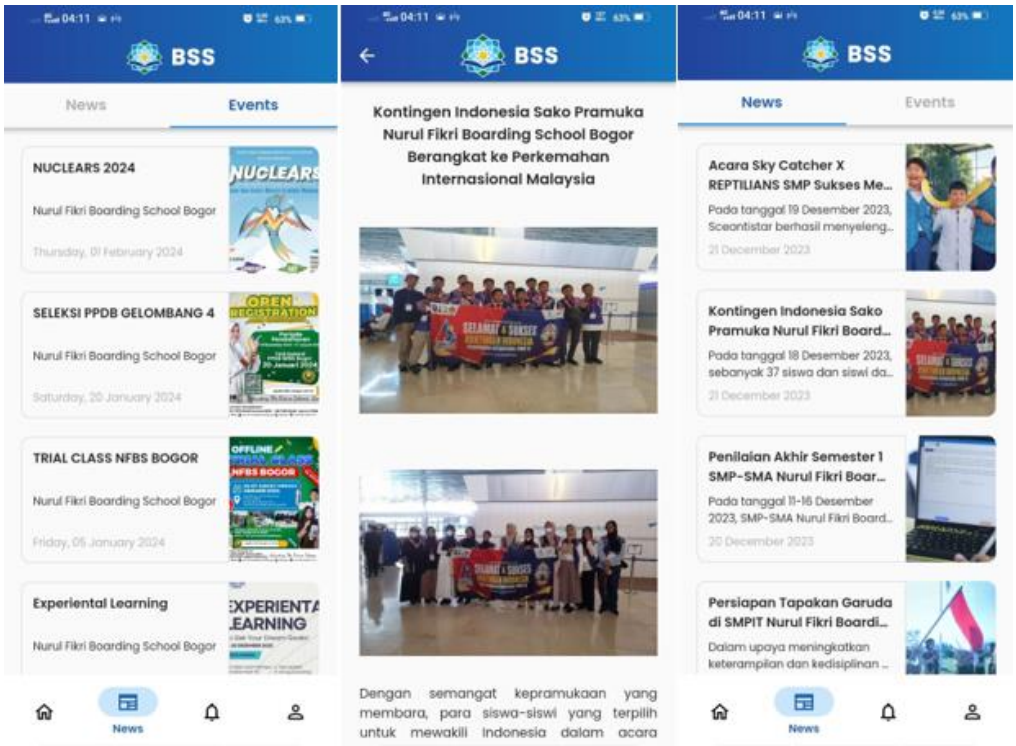
7. Tampilan Antarmuka Halaman Tahfidz



Gambar 5. 13. Tampilan Antarmuka Halaman Tahfidz

Pada Gambar 5.13 terdapat tampilan antarmuka halaman tahfidz yang memuat tentang prediksi hafalan santri.

8. Tampilan Antarmuka Halaman News



Gambar 5. 14. Tampilan Antarmuka Halaman News dan Event

Pada Gambar 5.14 terdapat 3 halaman yang memuat tentang tampilan halaman berita

dan kegiatan yang akan dilakukan.

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba dilakukan terhadap Aplikasi BSS Mobile. Pengujian dilakukan untuk memastikan kualitas perangkat lunak yang dibangun dan kesesuaian hasil eksekusi perangkat lunak dengan analisis dan perancangan perangkat lunak.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap Aplikasi BSS Mobile guna menguji kesesuaian dan ketepatan fungsionalitas dari seluruh sistem aplikasi

6.2. Kriteria Pengujian

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

a. Kemampuan aplikasi menampilkan berbagai form mulai dari form login dan form pengajuan perizinan santri.
b. Kemampuan aplikasi menampilkan berbagai tabel mulai dari riwayat perizinan santri, daftar perizinan yang butuh approval dari wali kelas dan kepala sekolah, daftar news dan event serta data-data user.
c. Kemampuan aplikasi menampilkan detail, seperti detail perizinan, detail news dan event, serta detail hafalan santri.
d. Kemampuan aplikasi untuk memberikan prediksi berapa lama santri dapat mengejar targetnya berdasarkan riwayat hafalan santri.
e. Kemampuan aplikasi untuk mendapatkan notifikasi realtime ketika ada perizinan yang masuk.
f. Kemampuan aplikasi memenuhi kebutuhan lainnya, yaitu dapat memilih santri untuk meminta izin tanpa harus mengganti akun, mencari nama santri tertentu dalam daftar perizinan dan halaman about yang bertujuan mengirim langsung user ke official akun NFBS Bogor.
g. Kesesuaian dalam memenuhi kebutuhan non-fungsional aplikasi, yaitu:
    - Pengaksesan sistem dari area yang terhubung internet
    - Sistem memiliki tampilan (antarmuka) yang mudah dipahami

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai Wali Santri, dosen atau tendik, serta mahasiswa yang akan menjalankan fitur-fitur dan seluruh kebutuhan fungsional dari sistem. Langkah-langkah untuk setiap kebutuhan fungsional yaitu sebagai berikut:

6.3.1. Wali Santri

a. Wali Santri melakukan login.
b. Wali Santri dapat melakukan penggantian santri apabila memiliki anka lebih dari 1.
c. Wali Santri mengisi formulir pengajuan perizinan.
d. Wali Santri melihat status perizinan.
e. Wali Santri melihat detail perizinan.
f. Wali Santri melihat riwayat perizinan yang telah dilakukan.
g. Wali Santri melihat data dan prediksi hafalan santri.
h. Wali Santri mendapatkan notifikasi apabila pengajuan perizinan diterima atau ditolak.
i. Wali Santri melihat daftar notifikasi yang masuk.
j. Wali Santri melihat daftar news.
k. Wali Santri melihat detail dari news tertentu.
l. Wali Santri melihat daftar events
m. Wali Santri melihat detail dari event tertentu.
n. Wali Santri melihat data dari santri saat ini.

6.3.2. Wali Kelas dan Kepala Sekolah

a. Wali Kelas dan Kepala Sekolah melakukan login.
b. Wali Kelas dan Kepala Sekolah melihat daftar perizinan yang butuh persetujuan.
c. Wali Kelas dan Kepala Sekolah melihat detail perizinannya.
d. Wali Kelas dan Kepala Sekolah memberikan atau menolak izin dari santri tertentu.
e. Wali Kelas dan Kepala Sekolah melihat riwayat pemberian persetujuan izin.
f. Wali Kelas dan Kepala Sekolah melihat detail dari riwayat izin tertentu.
g. Wali Kelas dan Kepala Sekolah mendapatkan notifikasi apabila ada pengajuan perizinan.
h. Wali Kelas dan Kepala Sekolah melihat daftar notifikasi yang masuk.
i. Wali Kelas dan Kepala Sekolah melihat daftar news.
j. Wali Kelas dan Kepala Sekolah melihat detail dari news tertentu.
k. Wali Kelas dan Kepala Sekolah melihat daftar event.
l. Wali Kelas dan Kepala Sekolah melihat detail dari event tertentu.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem Aplikasi BSS Mobile terhadap kasus skenario uji coba. Pengujian dilakukan oleh pihak pengembang, pengguna, dan pembimbing lapangan. Tabel 6.1 menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

Tabel 6.1 Hasil Evaluasi Pengujian Aplikasi BSS Mobile

Kriteria Pengujian	Hasil Pengujian
Melakukan login	Terpenuhi
Memilih atau Mengganti Santri (Wali Santri)	Terpenuhi
Melakukan Perizinan Santri (Wali Santri)	Terpenuhi
Mengisi Formulir Pengajuan Perizinan (Wali Santri)	Terpenuhi
Melihat Daftar Perizinan yang Sedang Berlangsung	Terpenuhi
Melihat Detail Perizinan	Terpenuhi
Melihat Daftar History Perizinan	Terpenuhi
Melihat Detail History Perizinan	Terpenuhi
Memberikan Approval Perizinan (Wali Kelas dan Kepala Sekolah)	Terpenuhi
Menampilkan Notifikasi Perizinan yang Masuk	Terpenuhi
Menampilkan Daftar Notifikasi	Terpenuhi
Melihat Prediksi Hafalan Santri (Wali Santri)	Terpenuhi
Menampilkan Daftar News	Terpenuhi
Melihat Detail News	Terpenuhi
Menampilkan Daftar Event	Terpenuhi
Melihat Detail Event	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan pengembangan aplikasi pada kegiatan KP di NFBS Bogor adalah sebagai berikut:

- Aplikasi yang dibagun berguna untuk membantu management sistem pendidikan NFBS Bogor yang digunakan oleh wali santri dan sivitas akademika NFBS Bogor.
- Dengan adanya Aplikasi BSS Mobile, mempermudah proses-proses dalam sistem pendidikan di NFBS Bogor.

7.2. Saran

Berikut ini adalah saran yang penulis berikan untuk arah perkembangan selanjutnya, yaitu sebaiknya dalam penerapan fitur-fitur berikutnya agar membuat sistem pendidikan NFBS Bogor lebih terdigitalisasi.

DAFTAR PUSTAKA

[1] *Home NFBS Bogor*. Diakses pada 18 Desember 2023 dari: https://nfbs-bogor.sch.id/ , Desember 2023.

[2] *Vision-Mission NFBS Bogor*. Diakses pada 08 Desember 2023 dari: https://nfbs-bogor.sch.id/vision-mission , Desember 2023.

[3] Awati, Rahul, 2023. *Cross Platform Mobile Development*. Diakses pada 08 Desember 2023 dari: https://www.techtarget.com/searchmobilecomputing/definition/cross-platform-mobile-development , Desember 2020.

[4] Alvian, 2019. *Panduan Awal Belajar Pemrograman Web Dalam 10 Menit*. Diakses pada 08 Desember 2023 dari: https://sis.binus.ac.id/2019/02/25/hubungan-dan-perbedaan-javascript-html-css-jquery-dan-php-di-dalam-web-development/, Desember 2023.

[5] Yasin, 2019. *Pengertian MySQL, Fungsi, dan Cara Kerjanya (Lengkap)*. Diakses pada 08 Desember 2023 dari: https://www.niagahoster.co.id/blog/mysql-adalah/, Desember 2023.

[6] Wijaya, Ketut Krisna, 2016. *Visual Studio Code: Aplikasi Editor Kode dari Microsoft untuk Windows, Linux, dan OS X*. Diakses pada 08 Desember 2023 dari: https://id.techinasia.com/visual-studio-code-editor-kode-microsoft, Desember 2023.

[7] Yasin, 2018. *Apa Itu Web Server dan Fungsinya?*. Diakses pada 08 Desember 2023 dari: https://www.niagahoster.co.id/blog/web-server-adalah/, Desember 2023.

[8] *Perbedaan Web Hosting, VPS, dan Cloud Server yang Perlu Diketahui Sebelum Membangun Website*. Diakses pada 08 Desember 2023 dari: https://blog.gamatechno.com/perbedaan-web-hosting-vps-cloud-server/, Desember 2023.

[9] Yasin, 2019. *Pengertian jQuery Serta Fungsi dan Contohnya*. Diakses pada 08 Desember 2023 dari: https://www.niagahoster.co.id/blog/jquery-adalah/, Desember 2023.

[10] Ham, Hanry. *Kelebihan Menggunakan Laravel Web Development*. Diakses pada 08 Desember 2023 dari: https://socs.binus.ac.id/2018/12/13/kelebihan-menggunakan-laravel-web-development/, Desember 2023.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Maisan Auliya, lahir pada tanggal 22 September 2002 di Jakarta. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS). Penulis berminat dalam bidang Mobile Development dan Machine Learning.

Samuel, lahir pada tanggal 31 Mei 2002 di Bandung. Penulis merupakan mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember (ITS). Penulis berminat dalam bidang Software Engineer dan juga DevOps.