BAB III
PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Saya melakukan pekerjaan selalama pelaksanaan kerja profesi di Badan Pengembangan Teknologi dan Sistem Informasi, dimana berfokus dalam bidang pengembangan dan pembangunan aplikasi. Pada pelaksanaan kerja profesi, saya menjalankan beberapa tugas yang diberikan oleh mentor saya, seperti mempelajari dan mengimplementasikan *framework* Apache Cordova, mempelajari dan mengimplementasikan penggunaan CodeIgniter4 untuk membangun *back-end* dan API *Server*, mempelajari dan mengimplementasikan *native* Android Studio, mempelajari dan mengimplementasikan *framework* Flutter pada Android Studio, serta mengembangkan aplikasi berbasis *website* yang ada di Universitas Nasional (UNAS). Akan tetapi, fokus utama saya adalah membangun sebuah aplikasi presensi yang dapat digunakan oleh mahasiswanya menggunakan teknologi *QR Code* berbasis Android. Pekerjaan ini melibatkan penggunaan Android Studio dengan *framework* Flutter untuk pengembangan antarmuka pengguna dan CodeIgniter4 (CI4) sebagai *server side* API yang menghubungkan *front end* Flutter dengan *database* MySQL.

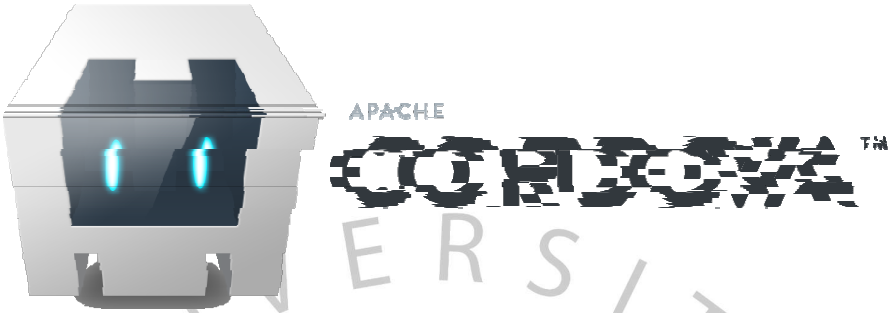
3.2 Pelaksanaan Kerja

Seperti yang sudah saya jelaskan sebelumnya, Pada pelaksanaan kerja profesi, saya menjalankan beberapa tugas, seperti mempelajari dan mengimplementasikan aplikasi Apache Cordova, mempelajari dan mengimplementasikan penggunaan CodeIgniter4 untuk membangun *back-end* dan API Server, mempelajari dan mengimplementasikan *native* Android Studio, mempelajari dan mengimplementasikan *framework* Flutter pada Android Studio, serta mengembangkan aplikasi berbasis *website* yang telah saya gambarkan dengan *gantt chart* dalam tabel 3.1 berikut:

Tabel 3. 1 *Gantt Chart* Kerja Profesi

No.	Kegiatan	Bulan			
		Juli	Agustus	September	Oktober
1.	Mempelajari dan Mengimplementasikan penggunakan *framework* Apache Cordova				
2.	Mempelajari dan mengimplementasikan penggunaan CodeIgniter4 untuk membangun *back-end* dan API Server				
3.	Mempelajari dan Mengimplementasikan penggunaan Android Studio *native*				
4.	Mempelajari dan mengimplementasikan *framework* Flutter pada Android Studio				
5.	Membangun aplikasi presensi dengan *QR Code* berbasis Android				
6.	Mengembangkan aplikasi berbasis *website*				

3.2.1 Apache Cordova



Gambar 3. 1 Apache Cordova

Apache Cordova merupakan kerangka pengembangan aplikasi mobile yang awalnya dikembangkan oleh Nitobi. Tepatnya di tahun 2011, dimana Adobe akan mengakuisisi Nitobi, kemudian akan mengubah nama alat tersebut menjadi *Phone*-Gap, serta merilis versi open-source yang dikenal sebagai Apache Cordova [3]. Apache Cordova dalam penggunaan kontennya yaitu menggunakan konten seperti halnya HTML, lalu CSS dan juga JavaScript untuk pengembang *web* yang mereka pakai untuk membuat aplikasi *native* untuk berbagai platform seluler. Pada awal pelaksanaan kerja profesi, mentor meminta praktikkan untuk membuat sebuah aplikasi presensi yang akan digunakan oleh mahasiswa menggunakan *QR Code* yang berbasis Android sehingga praktikkan diminta mentor untuk mempelajari dan menggunakan *framework* Apache Cordova, dikarenakan mentor merasa *framework* ini mudah untuk dipelajari dan digunakan mengingat *framework* ini bersifat *multi-cross platform* dan menggunakan bahasa *native* HTML, CSS, dan JavaScript. Akan tetapi, setelah kurang lebih 2 minggu mempelajari dan menggunakan *framework* ini, praktikkan merasa *framework* ini sulit untuk digunakan dikarenakan terdapat beberapa kendala yang dihadapi:

1. *Framework* Apache Cordova jarang digunakan oleh pengembang aplikasi *mobile* pada zaman sekarang ini, serta tidak dilanjutkannya pengembangan *framework* Apache Cordova

sehingga praktikkan kesulitan untuk mencari referensi, *dependencies,* serta *plugin* dalam menggunakan *framework* ini.

2. Instalasi Apache Cordova dapat dikatakan sulit dilakukan dikarenakan banyakanya *dependencies* yang harus di-*install* serta sulitnya memilih versi *requirements* yang sesuai.
3. *Plugins* pada Apache Cordova banyak yang memiliki keterbatasan dalam kesesuaian dengan *platform* dikarenakan banyak *plugins* yang tidak diperbarui.

Berdasarkan kendala-kendala tersebut, praktikkan mengajukan kesulitan untuk melanjutkan penggunaan *framework* Apache Cordova kepada mentor sehingga mentor menyilahkan untuk menggunakan *framework* lain dalam pembuatan aplikasi presensi. Praktikkan mengajukan untuk menggunakan Android Studio, dan mentor menerima ajuan praktikkan sehingga aplikasi presensi akan dibuat menggunakan Android Studio.

3.2.2 *Back-End* dan API *Server* dengan CodeIgniter4

Setelah mentor menyetujui pratikkan menggunakan Android Studio untuk membuat aplikasi presensi, mentor meminta praktikkan untuk mempelajari API *Server* dengan menggunakan CodeIgniter4 terlebih dahulu dikarenakan mentor ingin aplikasi presensi Android ini dapat terintegrasi dengan aplikasi lain atau terintegrasi dalam suatu sistem nantinya. Universitas Nasional (UNAS) menggunakan ini sebagai *framework* pengembangan aplikasi. Oleh karena itu, dalam tahap ini, praktikkan mempelajari cara membangun API *server* yang baik dengan menggunakan CodeIgniter4.

CodeIgniter adalah suatu framework pengembangan web yang bersifat *open-source*, diciptakan dengan tujuan untuk mempercepat proses pengembangan situs web dinamis dengan memanfaatkan bahasa pemrograman PHP [4]. *Back-end* merupakan bagian dari situs web yang tidak terlihat oleh pengguna. *Back-end* berperan dalam mengelola data di latar belakang, termasuk penyimpanan dan pengorganisasian data, serta memastikan fungsi dan keberlanjutan hal yang terlihat oleh pengguna berjalan dengan baik. *Application Programming Interface* (API) adalah

saluran yang memungkinkan dua atau lebih program komputer untuk saling berkomunikasi [5].

Gambar 3. 2 CodeIgniter

Landasan yang kokoh untuk mengembangkan aplikasi web yang dinamis dan efisien terdiri dari tiga elemen utama*: back-end*, API *server*, dan CodeIgniter. *Back-end* berperan sebagai inti operasional dengan tanggung jawab mengelola data dan logika bisnis, serta berfungsi sebagai dasar untuk keamanan dan stabilitas aplikasi. Sementara itu, API *server* berperan sebagai perantara yang memfasilitasi komunikasi yang lancar antara berbagai komponen aplikasi, memungkinkan pertukaran data yang diperlukan.

Dalam konteks ini, CodeIgniter menjadi kunci penting dalam mengarahkan pengembangan dengan pendekatan yang terstruktur. CodeIgniter menyediakan kerangka kerja PHP yang tangguh dan efisien, membantu pengembang membangun aplikasi *web* yang responsif dan dapat diandalkan. Kelebihan CodeIgniter terletak pada kemampuannya untuk menyediakan struktur yang terorganisir, mempermudah proses pengembangan, dan memastikan konsistensi dalam pengelolaan kode.

Dengan menggabungkan ketiga elemen ini, pengembang dapat menciptakan solusi *web* yang terintegrasi secara menyeluruh. Dengan fondasi *back-end* yang kuat, komunikasi yang efisien melalui API *server*, dan kerangka kerja CodeIgniter yang handal, aplikasi web yang dihasilkan menjadi handal, responsif, dan dapat berkembang sesuai kebutuhan. Kolaborasi ketiganya membentuk ekosistem yang kuat untuk aplikasi modern, memberikan fleksibilitas dan skalabilitas yang diperlukan untuk memenuhi tuntutan perkembangan teknologi dan kebutuhan pengguna.

3.2.3 Android Studio (*Native)*

Android Studio merupakan aplikasi yang dapat dibilang sebagai *Integrated Development Environment* (IDE) yang secara alami dan diresmikan untuk dikembangkan oleh Google dalam pengoprasian sistem android. IDE ini diciptakan menggunakan beberapa perangkat lunak seperti IntelliJ IDEA dari JetBrains untuk dapat dirancang secara khusus dalam mempermudah proses perkembangan dari aplikasi Android. Android Studio menyajikan lingkungan terpadu dan komprehensif bagi para pengembang Android, memungkinkan mereka untuk membuat, menguji, dan melakukan debugging aplikasi dengan efisien. IDE ini menjadi perangkat penting bagi pengembang yang ingin menciptakan aplikasi inovatif untuk *platform* Android [6].

Gambar 3. 3 Android Studio

Setelah mempelajari tentang API *server* dan CodeIgniter4, saya mempelajari Android Studio. Pada tahap ini, saya menggunakan bahasa Java *native* untuk membuat tampilan-tampilan aplikasi Android. Saya membuat tampilan (*front-end) dashboard* atau halaman utama untuk aplikasi presensi saya. Akan tetapi, penggunaan Java *native* ini dianggap

mentor saya kurang efisien sehingga diminta untuk menggunakan *framework*. Oleh karena itu, saya tidak melanjutkan pembuatan *front-end* dengan Java *native*.

3.2.4 Android Studio (Flutter)

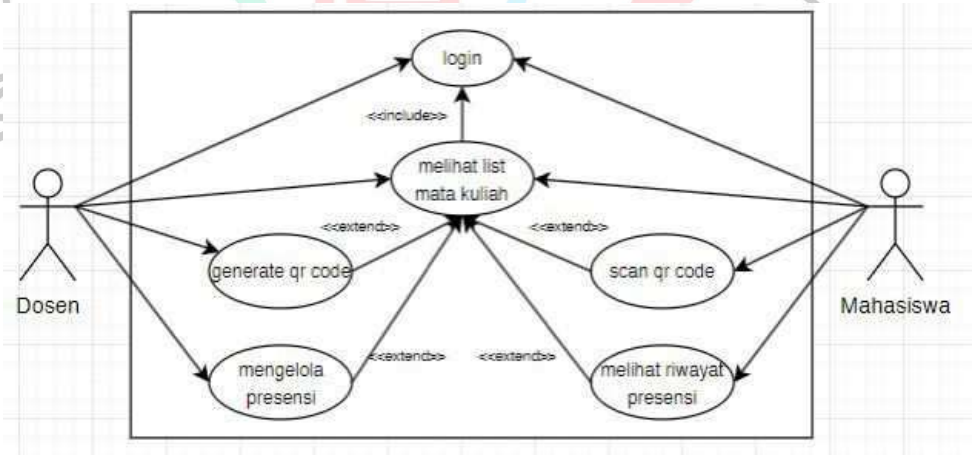
Gambar 3. 4 Flutter

Flutter adalah suatu kerangka pengembangan perangkat lunak open source yang dirancang oleh Google untuk membangun antarmuka pengguna (UI) aplikasi yang dapat digunakan secara lintas platform. Dalam pengembangan aplikasi menggunakan Flutter, sebagian besar pengembang dipastikan Android Studio sebagai *Integrated Development Environment* (IDE) untuk pilihnya karena memberikan dukungan penuh untuk pengembangan dengan Flutter. Android Studio menyertakan *plugin* Flutter yang dapat mempermudah pengembangan aplikasi Flutter dengan menyediakan berbagai fitur, termasuk pembuatan proyek baru, manajemen dependensi, dan alat pengembangan lainnya yang mendukung pengembangan menggunakan Flutter. Selain itu, Android Studio menyediakan *emulator* atau perangkat fisik untuk menjalankan dan menguji aplikasi Flutter. Dengan memanfaatkan Android Studio, pengembang dapat menggunakan berbagai alat dan fitur bantu untuk

meningkatkan efisiensi dalam pengembangan aplikasi Flutter, mengambil keuntungan dari kekuatan lingkungan pengembangan terpadu ini. Flutter sendiri ditulis menggunakan bahasa pemrograman Dart dan berjalan di atas mesin virtual Dart yang dilengkapi dengan mesin eksekusi kompilasi waktu nyata. Saat melakukan pemrograman atau *debugging* aplikasi, Flutter memanfaatkan kompilasi waktu nyata untuk melakukan *"hot reload"* yang memungkinkan hasil perubahan kode ditambahkan langsung ke aplikasi yang sedang berjalan [7].

3.2.5 *Use Case Diagram*

Diagram *use case* atau biasa yang juga disebut dengan (*use case diagram*) adalah representasi grafis dalam *Unified Modeling Language,* secara nyata digunakan untuk mengilustrasikan interaksi antara berbagai aktor, baik itu pengguna atau sistem lain, dengan suatu sistem atau aplikasi tertentu. Dalam diagram ini, hubungan antara *use case* dan aktor-aktor eksternal tersebut dapat dijelaskan secara visual. Dibawah ini merupakan *Use Case Diagram* dari sebuah aplikasi yang sedang dikembangkan:



Gambar 3. 5 *Use Case Diagram*

Use Case di atas tersebut terdiri dari 2 aktor, yaitu Mahasiswa dan Dosen yang memiliki beberapa perbedaan dalam menggunakan aplikasi. Mahasiswa dan Dosen harus melakukan *login* untuk dapat

menggunakan aplikasi. Setelah *login*, mahasiswa dan dosen dapat melihat list mata kuliah. Selain itu, dosen dapat melakukan *generate qr code* dan mengelola presensi, sementara mahasiswa dapat melakukan *scan qr code* dan melihat riwayat presensi. Deskripsi lebih lengkap mengenai *Use Case* dapat dilihat pada *Use Case Description* berikut:

1. **Login**

   Use Case ini menggambarkan aktivitas *login* oleh *user* (Dosen dan Mahasiswa). Aktivitas *login* melibatkan pengisian *username* dan *password* dan akan terjadi jika Aktor (Dosen dan Mahasiswa) membuka aplikasi dengan syarat Dosen dan Mahasiswa harus sudah memiliki *username* dan *password.* Hasil dari aktivitas ini adalah *user* dapat mengakses *dashboard* utama. Dimana *Use Case Login* dapat tertampil dalam tabel 3.2 yang ada di bawah ini.

Tabel 3. 2 *Use Case Login*

Use Case Name	Login	
Actor	Dosen dan Mahasiswa	
Description	Pada aktivitas ini menggambarkan proses Login User	
Trigger	User membuka aplikasi	
Pre-Condition	Memiliki username dan password	
Normal Course	**User Action**	**System**
		Sistem menampilkan form login
	User mengisi form login	
	User mengklik tombol login	
		Sistem menampilkan dashboard utama
Post-Condition	User dapat mengakses dashboard utama	
Alternative Flows	Menampilkan "Error, Isilah username dan password dengan benar."	

2. **Melihat List Mata Kuliah**

Use Case ini menggambarkan aktivitas Melihat List Mata Kuliah oleh *user* (Dosen dan Mahasiswa). Aktivitas Melihat List Mata Kuliah adalah aktivitas *user* untuk mengakses *dashboard* utama yang berisi list mata kuliah. Pada Dosen, akan tampil list dari beberapa mata kuliah untuk mengajar mahasiswanya. Sementara, pada mahasiswa, akan tampil beberapa list dari mata kuliah yang diambil oleh mahasiswa tersebut. Aktivitas ini akan terjadi jika Aktor (Dosen dan Mahasiswa) mengakses page *dashboard* dengan syarat Dosen dan Mahasiswa sudah *login.* Hasil yang tercipta yaitu sistem dapat menampilkan list mata kuliah. *Use Case* Melihat List Mata Kuliah dapat tertampil bada tabel dibawah ini.

Tabel 3. 3 *Use Case* Melihat List Mata Kuliah

Use Case Name	Melihat List Mata Kuliah	
Actor	Dosen dan Mahasiswa	
Description	Pada aktivitas ini menggambarkan proses melihat list mata kuliah	
Trigger	User mengakses page dashboard	
Pre-Condition	Sudah Login	
Normal Course	**User Action**	**System**
	User berhasil login	
		Sistem menampilkan halaman dashboard
Post-Condition	Sistem menampilkan list mata kuliah.	
Alternative Flows	Menampilkan page dashboard yang berisi list mata kuliah	

3. Scan QR Code

Use Case ini menggambarkan aktivitas Scan dari QR Code oleh Mahasiswa. Aktivitas Scan QR Code adalah akitivitas *scan* QR *code* untuk melakukan presensi. Aktivitas ini akan terjadi jika Mahasiswa menekan ikon QR pada *dashboard* Mahasiswa. Hasil yang tercipta yaitu sistem dapat menampilkan pesan presensi Mahasiswa berhasil. *Use Case* Scan QR Code tertampil pada tabel dibawah ini.

Tabel 3. 4 *Use Case* Scan QR Code

Use Case Name	Scan QR Code	
Actor	Mahasiswa	
Description	Pada aktivitas ini melakukan Scan QR untuk presensi	
Trigger	Mahasiswa menekan ikon QR	
Pre-Condition	Membuka halaman dashboard Mahasiswa	
Normal Course	**User Action**	**System**
	User menekan ikon Scan QR Pada Page Mata Kuliah	
		Sistem membuka camera scanner
	User melakukan Scan QR	
		User melakukan Scan QR
Post-Condition	Sistem menampilkan pemberitahuan bahwa presensi berhasil disimpan.	
Alternative Flows	Menampilkan "Maaf presensi gagal disimpan, silahkan coba lagi."	

4. Melihat Riwayat Presensi

Use Case ini menggambarkan aktivitas Melihat Riwayat Presensi oleh Mahasiswa. Aktivitas Melihat Riwayat Presensi adalah akitivitas mahasiswa untuk melihat riwayat presensi mahasiswa dalam suatu mata kuliah. Aktivitas ini akan terjadi jika Mahasiswa menekan tulisan "Course Details" pada *dashboard* Mahasiswa. Hasil dari aktivitas ini adalah sistem menampilkan detail dari mata kuliah serta riwayat presensi mahasiswa dalam mata kuliah yang dipilih. *Use Case* Melihat Riwayat Presensi tertampil pada tabel 3. 5 berikut.

Tabel 3. 5 *Use Case* Melihat Riwayat Presensi

Use Case Name	Melihat Riwayat Presensi	
Actor	Mahasiswa	
Description	Pada aktivitas ini menggambarkan user melihat Riwayat Presensi	
Trigger	Mahasiswa menekan "Course Details"	
Pre-Condition	Membuka halaman dashboard Mahasiswa	
Normal Course	User Action	System
	User menekan button Course Details	
		Menampilkan detail mata kuliah dan riwayat presensi
Post-Condition	Sistem menampilkan riwayat presensi mahasiswa	
Alternative Flows	Menampilkan "Maaf, terdapat kesalahan dalam server"	

5. Generate QR Code

Use Case ini menggambarkan aktivitas Generate dari QR Code oleh Dosen. Aktivitas tersebut adalah akitivitas *generate* dari QR *code,* dimana digunakan untuk melakukan presensi mahasiswa. Aktivitas ini akan terjadi jika Dosen menekan ikon QR pada *dashboard* Dosen. Hasil yang tercipta dari aktivitas ini adalah sistem dapat menampilkan QR *Code*. *Use Case* Generate QR Code yang ditampilkan pada tabel berikut ini.

Tabel 3. 6 *Use Case* Generate QR Code

Use Case Name	Generate QR Code	
Actor	Dosen	
Description	Pada aktivitas ini dosen membuat QR Code untuk presensi mahasiswa	
Trigger	Dosen klik icon QR	
Pre-Condition	Membuka halaman dashboard Dosen	
Normal Course	**User Action**	**System**
	User menekan ikon QR pada mata kuliah	
		Menampilkan QR Generate Screen
	Dosen Mengklik button Generate QR	
		Menampilkan QR Code yang sudah digenerate
Post-Condition	Sistem menampilkan QR Code	

Alternative Flows	Menampilkan "QR gagal dibuat, silahkan coba lagi"

6. Mengelola Presensi Mahasiswa

Use Case ini menggambarkan aktivitas Mengelola Presensi Mahasiswa oleh Dosen. Aktivitas dilakukan oleh dosen untuk melihat serta mengelola data presensi mahasiswa pada suatu mata kuliah yang diajar dosen. Aktivitas ini akan terjadi jika Dosen menekan "Student List" pada *dashboard* Dosen. Hasil dari aktivitas ini adalah sistem menampilkan list mahasiswa yang terdapat dalam mata kuliah tersebut serta sistem akan menyimpan presensi mahasiswa. *Use Case* Mengelola Presensi Mahasiswa yang tertampil dalam tabel 3.7 di bawah ini.

Tabel 3. 7 *Use Case* Mengelola Presensi Mahasiswa

Use Case Name	Mengelola Presensi Mahasiswa	
Actor	Dosen	
Description	Pada aktivitas ini dosen mengelola presensi mahasiswa	
Trigger	Dosen klik icon Presensi	
Pre-Condition	Membuka halaman dashboard admin	
Normal Course	**Action**	**System**
	Dosen menekan "Student List" pada mata kuliah yang diinginkan	
		Menampilan Attendance List Screen
	Dosen mengelola Attendance Mahasiswa	
		Merekam dan menyimpan perubahan
Post-Condition	Presensi Mahasiswa berhasil disimpan	

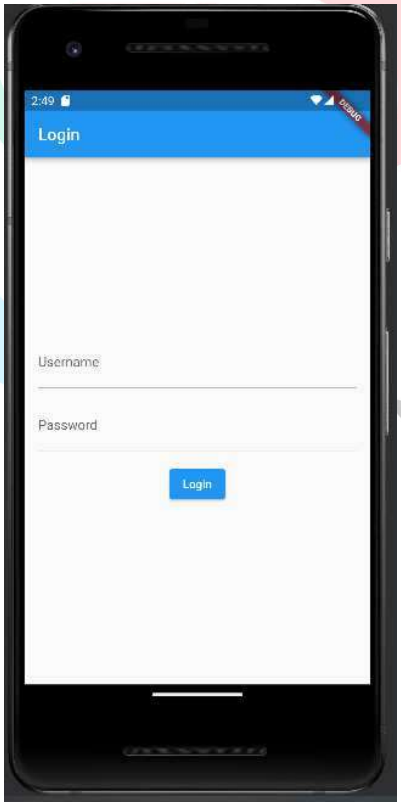
Alternative Flows	Menampilkan "Presensi gagal disimpan, silahkan coba lagi"

3.2.6 Pengembangan Aplikasi

Berdasarkan *Use* Case yang sudah dijelaskan, saya membuat beberapa modul dalam aplikasi. Modul-modul tersebut di antaranya:

1. **Home (*Login*)**

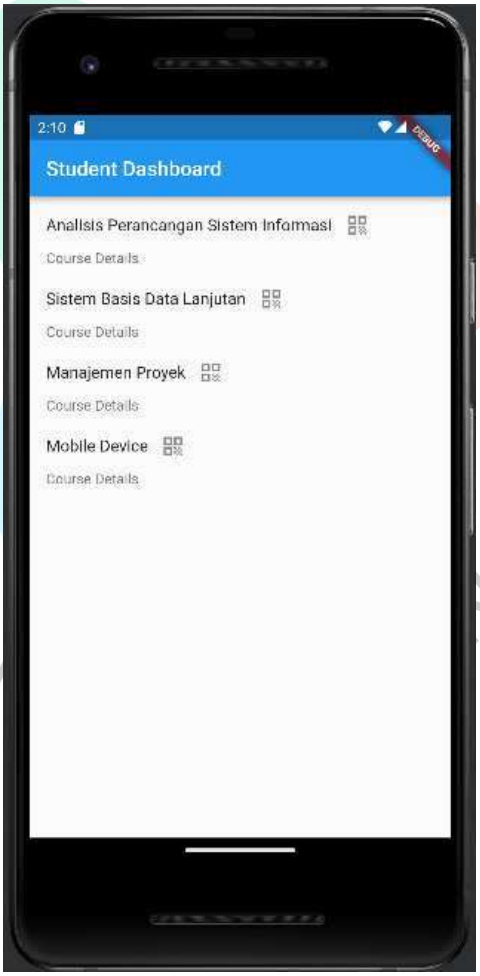
   Pertama, saya membuat halaman "Home" yang berisi formulir *login*, memungkinkan pengguna untuk memasukkan *username* dan *password* mereka. Setelah mengisi informasi ini, mereka dikatakan dapat mengklik tombol "Login" untuk dapat masuk ke *dashboard*. *Username* dan *Password* tersebut didapat dari akun akademik UNAS yang nantinya sudah terintegrasi dengan aplikasi. Tampilan halaman "Home" dapat tertampil pada gambar dibawah ini.



Gambar 3. 6 Tampilan Home

2. **Dashboard Mahasiswa**

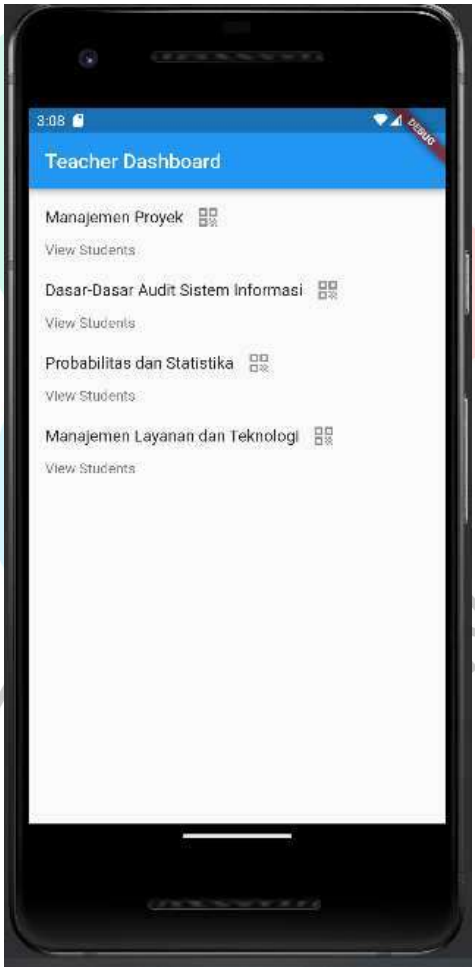
Saya merancang halaman ini untuk memenuhi kebutuhan mahasiswa. Mahasiswa dapat melihat daftar mata kuliah yang mereka ambil. Mahasiswa dapat melihat ikon *QR Code* pada setiap mata kuliah. Jika ikon tersebut ditekan, mahasiswa akan membuka *scanner camera* untuk memindai *QR Code* dan melakukan presensi. Di bawah nama mata kuliah, terdapat tulisan "Course Details" yang jika tulisan tersebut atau nama mata kuliah ditekan, kemudian hal tersebut menunjukan ke halaman "Course Details". Tampilan halaman Dashboard Mahasiswa dapat ditampilkan pada gambar berikut ini.



Gambar 3. 7 Tampilan Dashboard Mahasiswa

3. **Dashboard Dosen**

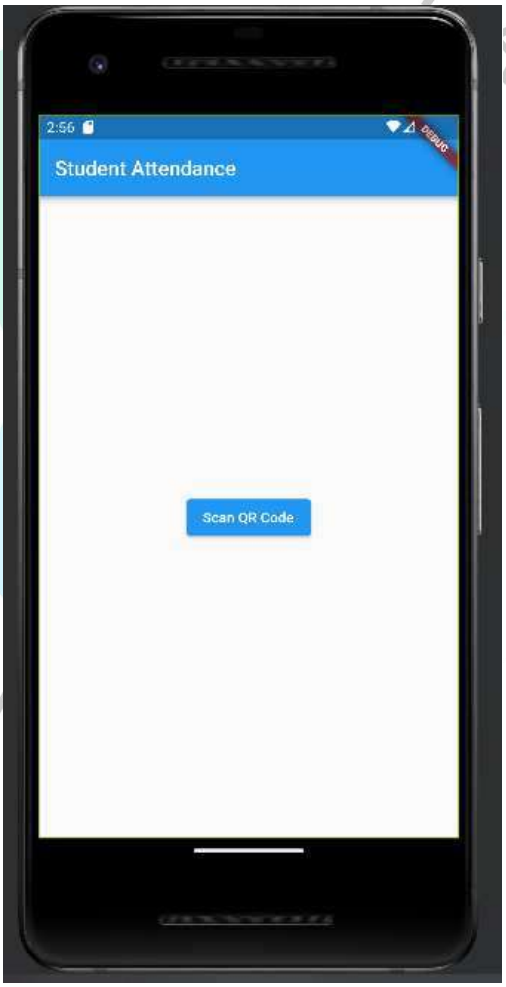
Saya merancang halaman ini untuk memenuhi kebutuhan dosen. Dosen dapat melihat mata kuliah yang diajarinya. Dosen dapat melihat ikon *QR Code* pada setiap mata kuliah. Jika ikon tersebut ditekan, dosen dapat masuk ke halaman *QR Generate* untuk melakukan *generate QR Code*. Di bawah nama mata kuliah, terdapat tulisan "View Students" yang jika tulisan tersebut atau nama mata kuliah ditekan, akan diarahkan ke halaman "Student List". Tampilan halaman dari Dashboard Dosen tertampil pada gambar berikut.



Gambar 3. 8 Tampilan Dashboard Dosen

4. **Scan QR Code (Mahasiswa)**

Saya membuat halaman ini untuk mahasiswa yang berisi tombol "Scan QR Code" dimana jika tombol tersebut ditekan, akan membuka *scanner* untuk melakukan *scan* terhadap *QR Code*. Akan tetapi, presensi dapat dilakukan tidak hanya dengan *scan QR Code* secara fisik, *scanner* juga bisa memilih gambar dalam galeri untuk di*scan*. Jadi, mahasiswa dapat menyimpan *QR Code* yang bisa dikirim oleh dosen. Tampilan halaman "Scan QR Code" dapat tertampil pada gambar dibawah ini.



Gambar 3. 9 Tampilan Scan QR Code

5. **Generate QR Code (Dosen)**

   Saya membuat halaman ini untuk dosen yang berisi tombol “Generate QR Code” dan jika ditekan akan menghasilkan *QR Code* yang digunakan untuk melakukan presensi mahasiswa. Tampilan halaman dari “Generate QR Code” sebelum mengklik *button* “Generate QR Code” dapat tertampil pada gambar dibawah ini.

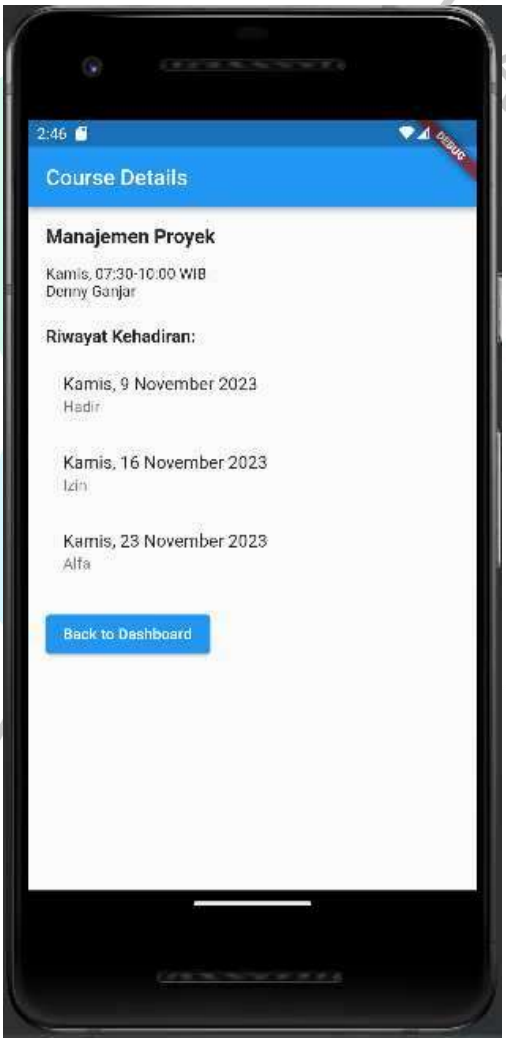
Gambar 3. 10 Tampilan Generate QR Code 1

Setelah tombol ditekan akan muncul *QR Code* dan juga tombol "Delete QR Code" yang berfungsi untuk menghapus *QR Code* dan datanya sehingga jika tombol hapus ini ditekan, mahasiswa tidak lagi dapat melakukan *scan* terhadap *QR Code* tersebut. Tampilan halaman "Generate QR Code" setelah menekan tombol "Generate QR Code" yang ditampilkan pada gambar berikut ini.

Gambar 3. 11 Tampilan Generate QR Code 2

6. **Course Details (Mahasiswa)**

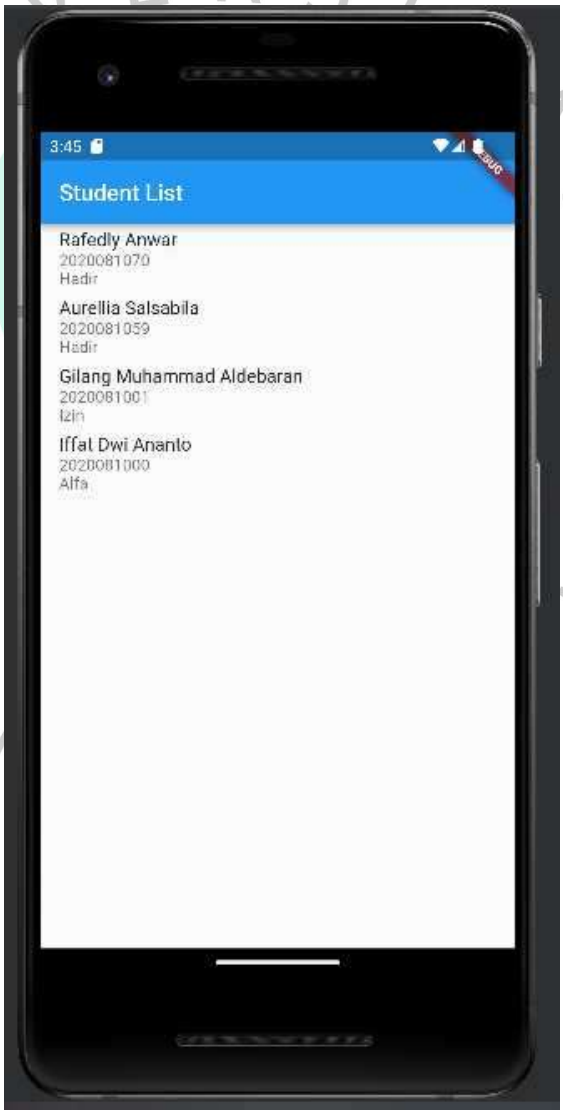
Saya membuat halaman ini yang memungkinkan mahasiswa untuk melihat detail mata kuliah tertentu serta melihat riwayat kehadiran mahasiswa dalam mata kuliah yang dipilih. Mahasiswa dapat mengakses halaman ini dengan mengklik nama mata kuliah atau tulisan "Course Details" pada halaman "Dashboard Mahasiswa". Terdapat tombol "Back to Dashboard" untuk dapat kembali ke *dashboard* paling awal, praktikan akan tampilkan dibawah ini.



Gambar 3. 12 Tampilan Course Details

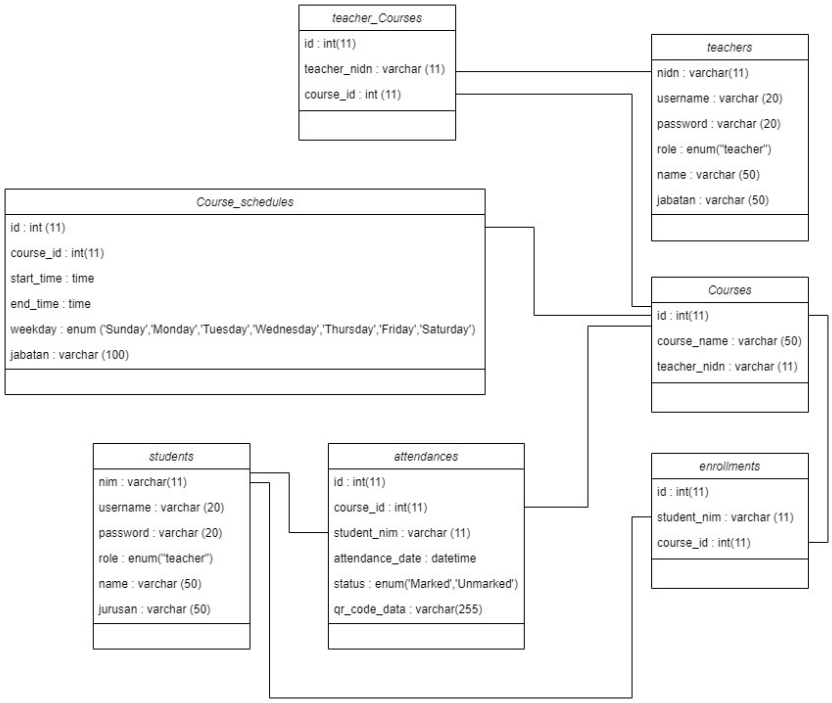
7. **Student List (Dosen)**

Saya membuat halaman ini untuk dosen. Ini memungkinkan dosen untuk melihat list seluruh mahasiswa yang terdaftar dalam kelas dan melihat kehadiran mahasiswa. Pada halaman ini, dosen seharusnya dapat melakukan pengelolaan presensi mahasiswa. Akan tetapi, praktikkan belum menambahkan fitur tersebut, dikarenakan aplikasi ini merupakan aplikasi yang masih dikembangkan dan belum tahap final. Berikut tampilannya dibawah ini.



Gambar 3. 13 Tampilan Student List

Aplikasi Flutter tersebut saya buat masih dengan menggunakan *database* tidak asli, dikarenakan aplikasi ini masih dalam tahap pengembangan. Saya menggunakan *database* MySQL yang dipanggil melalui API *Server* CodeIgniter4. Berikut adalah *Class Diagram* berisi struktur *database* MySQL yang telah saya buat untuk pembuatan sementara aplikasi:



Gambar 3. 14 *Class Diagram*

Database tersebut dipanggil atau dihubungkan dengan *front-end* pada Flutter melalui API *Server* (*back-end)* CodeIgniter4. Berikut adalah modul-modul yang terdapat pada CodeIgniter4*:*

1. **AttendanceController**

   Controller ini berisi *code* untuk menerima *post* dan mengirim *respond* berupa data "$deleted", mengirim pesan, serta status 200 atau 400 ke *front-end* melalui *function* GenerateQRCode*,* ScanQRCode*, dan* DeleteQRCode.

2. **AttendanceListController**

   Controller ini berisi *code* untuk mengirim *respond* berupa data "$attendanceList" dan status 200 atau 400 ke *front-end* melalui *function* getAttendanceList.

3. **AuthController**

   Controller ini berisi *code* untuk menerima *post* dan mengirim *respond* berupa data "$responseData", status 200 atau 400, dan pesan gagal ke *front-end* melalui *function* Login.

4. **CourseDetailsController**

   Controller ini berisi *code* untuk mengirim *respond* berupa data "$response" dan status 200 atau 400 ke *front-end* melalui *function* getCourseDetails.

5. **StudentsController**

   Controller ini berisi *code* untuk mengirim *respond* berupa data "$enrolledCourses" dan status 200 atau 400 ke *front-end* melalui *function* getEnrolledCourses.

6. **TeacherDashboardController**

   Controller ini berisi *code* untuk mengirim *respond* berupa data "$enrolledCourses" dan status 200 atau 400 ke *front-end* melalui *function* getCoursesByTeacher.

3.3 Kendala Yang Dihadapi

Dari pelaksanaan kerja profesi yang dilakukan, berikut merupakan kendalanya:

1. **Belum Berpengalaman dengan Flutter**

   Saya belum memiliki pengalaman dalam menggunakan Flutter untuk pengembangan aplikasi Android. Namun, saya berhasil mengatasi masalah ini dengan berdiskusi dengan rekan kerja dan belajar dari tutorial di internet. Proses ini membantu saya memahami bagaimana menggunakan Flutter untuk pengembangan Android.

2. **Kesulitan dalam Pembangunan API dengan CodeIgniter4**

   Saya mengalami kesulitan dalam membangun API dengan CodeIgniter4 dan menghubungkannya dengan aplikasi Flutter. Namun, saya mengatasi kendala ini dengan belajar melalui

sumber daya *online*. Saya menggunakan metode *http.get*, *http.delete* atau *http.post* untuk menghubungkan aplikasi Flutter dengan API dan berhasil dalam pertukaran data antara keduanya.

3. **Kesulitan dalam Pembuatan Data QR Code**

   Saya juga mengalami kesulitan dalam menganalisis jenis data yang dibutuhkan oleh aplikasi, terutama data *QR Code*. Namun, saya menyelesaikan masalah ini dengan mempelajari informasi dari internet dan berkonsultasi dengan rekan kerja. Akhirnya, saya berhasil membuat data *QR Code* yang diperlukan dan memahami cara kerjanya.

4. **Kesulitan dalam Instalasi dan Penggunaan Apache Cordova**

   Saya mengalamai kesulitan saat instalasi dan menggunakan Apache Cordova, dikarenakan *dependencies* yang sulit untuk didapatkan (contoh: versi yang tidak *up-to-date,* keterbatasan *plugins* yang tersedia, dokumentasi yang kurang lengkap).

3.4 Cara Mengatasi Kendala

Dalam mengatasi dari kendala diatas yang saya hadapi selama melkukan kerja profesi, saya mengambil langkah-langkah berikut:

1. Untuk kendala pertama, saya melakukan pembelajaran intensif tentang Flutter dengan belajar dari tutorial, berdiskusi dengan rekan kerja yang berpengalaman, dan mengikuti panduan *online*. Ini membantu saya memahami dasar-dasar penggunaan Flutter dalam waktu singkat.
2. Kendala kedua diatasi dengan memperdalam pengetahuan saya tentang CodeIgniter4 melalui dokumentasi resmi dan sumber daya *online*.
3. Kendala ketiga diatasi dengan studi mendalam tentang struktur data yang diperlukan aplikasi dan mencari pemahaman yang lebih baik tentang teknologi QR Code melalui sumber daya *online*.

4. Saya mengatasi kendala keempat dengan observasi secara daring serta berkomunikasi pada orang-orang yang berkaitan dengan Apache Cordova (contoh: *Youtuber* yang membuat *video* tentang Apache Cordova).

3.5 Pembelajaran Yang Diperoleh dari Kerja Profesi

Selama pelaksanaan kerja profesi ini, saya begitu mendapat banyak sekali wawasan berharga, seperti:

1. Memahami cara menggunakan *framework* Flutter untuk pengembangan aplikasi Android. Praktikkan tidak hanya dapat menggunakan Flutter untuk pengembangan aplikasi Android, tetapi disarankan untuk terus mendalami fitur-fitur canggih dan teknik pengembangan terbaru yang ditawarkan oleh Flutter. Mengeksplorasi *widget-widget* kustom, *state management*, dan integrasi dengan platform lain dapat memperkaya pemahaman dan keterampilan dalam membangun aplikasi Android yang responsif dan menarik.
2. Memahami cara membangun aplikasi Android native dengan Android Studio. Selain menguasai dasar-dasar pembangunan aplikasi Android dengan Android Studio, praktikkan dapat mengeksplorasi fitur-fitur lanjutan seperti pengoptimalan performa, penggunaan arsitektur komponen Android (Android *Architecture Components*), dan integrasi dengan layanan *cloud*. Ini akan memperkaya pengalaman dalam mengembangkan aplikasi yang dapat bersaing di pasar yang kompetitif.
3. Memahami cara membangun aplikasi Android dengan framework Apache Cordova. Selain mengenal dasar-dasar pengembangan menggunakan Apache Cordova, praktikkan dapat mengeksplorasi penggunaan *plugin* tambahan dan memahami cara mengatasi potensial perbedaan kinerja dengan pengembangan aplikasi natif. Menjadi mahir dalam menggabungkan keunggulan *cross-platform* dengan performa tinggi akan menjadi nilai tambah yang signifikan.

4. Memahami cara menggunakan CodeIgniter4 untuk membangun *back-end* dan API. Selain menguasai dasar-dasar pengembangan *back-end* dengan CodeIgniter4, praktikkan dapat memperdalam pengetahuan tentang *middleware*, *routing* yang kompleks, dan integrasi dengan *database*. Menjadi ahli dalam mengoptimalkan performa *back-end* akan menjadi keunggulan penting dalam membangun sistem yang responsif dan efisien.
5. Memahami cara mengembangkan aplikasi web dengan *framework* CodeIgniter4. Praktikkan dapat memperluas pemahaman mereka tentang CodeIgniter4 untuk mengembangkan aplikasi web yang kompleks.
6. Memiliki pemahaman yang lebih baik mengenai penggunaan *QR Code* dalam aplikasi. Pemahaman yang lebih mendalam tentang potensi *QR Code* dapat menginspirasi ide-ide kreatif dalam pengembangan aplikasi.
7. Memperoleh keterampilan komunikasi yang lebih baik dalam berkolaborasi dengan rekan kerja dan mencari solusi ketika menghadapi kendala. Selain mengasah keterampilan teknis, praktikkan dapat terus mengembangkan keterampilan interpersonal mereka. Mengikuti pelatihan komunikasi dan berkolaborasi secara efektif dapat membantu dalam membangun tim yang produktif dan mengatasi tantangan proyek dengan lebih baik.
8. Memperoleh pengalaman yang baik serta wawasan baru yang lebih luas. Melibatkan diri dalam berbagai aspek pengembangan perangkat lunak, seperti manajemen proyek dan analisis kebutuhan pengguna, dapat memberikan perspektif yang lebih luas dan meningkatkan keterampilan profesional secara menyeluruh.