BAB III
PELAKSANAAN KERJA PROFESI

3.1. Bidang Kerja

Dalam melaksanakan kerja profesi di MileApp (PT. Paket Informasi Digital), Praktikan ditempatkan pada divisi teknologi oleh *Head Of Tech* sebagai *Quality Assurance Engineer* untuk membantu proses perancangan, pengembangan dan pengujian secara manual dan otomatis dari sisi *API* (*Application Programming Interface*) dan UI (*User Interface*) pada modul *Billing MileApp* Versi 3.0. Praktikan ditugaskan dalam sebuah tim untuk bekerjasama dan berkolaborasi dengan *Back-End Engineer, Front-End Engineer* dan *Product Owner*. Pada proses pembuatan dan pengembangan sistem Billing praktikan bertanggung jawab untuk melakukan pengujian sistem, baik dari dari sisi *Back-end* yaitu *API*(*Application Programming Interface)* dan tampilannya dan dari sisi *Front-end* menggunakan tools seperti Postman dan Katalon. Proses pengujian dilakukan berdasarkan *test case* dan *test scenario* yang telah dibuat oleh praktikan ketika proses perancangan sistem sedang berlangsung. Setelah proses pengujian dilakukan maka praktikan sebagai *QA Engineer* membuat dokumentasi pengujian. Jika pada sistem masih ditemukan *bug* maka praktikan akan berdiskusi terkait *bug* yang ada pada *Back-End* atau *Front End* dan melakukan pengujian ulang apabila *bug* yg ditemukan sudah diperbaiki.

3.2 Pelaksanaan Kerja

Praktikan melaksanakan Kerja Profesi dimulai dari 2 Juni 2022 sampai dengan 2 September 2022 secara WFH (*Work From Home*) di bagian divisi teknologi atau yang sering disebut *tech division*. Praktikan bertanggung jawab sebagai *Quality Assurance Engineer* untuk melakukan pengujian secara manual dan otomatis pada modul *Billing MileApp V3.* Pengujian dilakukan secara bertahap dimulai dari sisi *back-end* dan *front-end.* Pengujian yang

dilakukan menerapkan ilmu yang pernah didapatkan oleh praktikan selama masa pembelajaran di perkuliahan yaitu pengujian *Black-box* dan *White-box*. Dimana pengujian *Black-box* dilakukan untuk mengetahui fungsionalitas aplikasi tersebut dengan menguji tampilan UI/UX dari sisi *front-end* modul *billing V3* sedangkan untuk pengujian *white-box* merupakan pengujian yang dilakukan untuk mengetahui *code* dan struktur dari aplikasi yang dikembangkan. Praktikan melakukan pengujian *black-box* dengan menguji *endpoint* API (*Application Programming Interface*) dari sisi *back-end* menggunakan *tools* postman dan katalon. Berikut ini merupakan tujuan dan sasaran dari pengujian modul *Billing* MileApp :

1. Pengujian dilakukan untuk menemukan kesalahan atau *bug* yang ada pada sistem baik dari segi tampilan dan kode program.
2. Pengujian dilakukan untuk menjamin kualitas *software* sudah sesuai dengan standar dan kebutuhan *user*.
3. Pengujian dilakukan untuk mendeteksi adanya *error* dan melakukan validasi sistem berjalan semestinya.

Pelaksanaan kerja menggunakan metode yaitu *Software Development Life Cycle (SDLC)* dengan menggunakan metode *agile* yaitu *scrum* yang terdiri dari proses perencanaan, analisis, desain, implementasi, pengujian dan pemeliharaan sistem. Menurut Ken Schwaber & Jeff Sutherland (2020) dalam bukunya menjelaskan bahwa *Scrum* merupakan metode pengembangan sistem yang fleksibel dan mampu beradaptasi secara cepat jika terjadi perubahan. Metode jenis *scrum* membutuhkan *scrum team* yang terdiri dari *scrum master*, *product owner* dan *developers*. *Scrum team* bersifat lintas fungsi dimana setiap anggota tim memiliki keterampilan yang diperlukan untuk mencapai *product goal.* Berikut adalah keterampilan yang harus dimiliki oleh *scrum team* :

- ***Scrum Master***

Scrum master bertanggung jawab untuk membantu tim dalam memastikan proyek yang dikembangkan berjalan lancar tanpa hambatan dan membantu anggota tim memahami *product backlog item* dengan jelas dan

ringkas agar dapat mencapai tujuan yaitu *product goal.*

- ***Product Owner***

  Product owner memiliki tugas dan tanggung jawab untuk memaksimalkan produk yang dikembangkan yaitu *Billing MileApp V3. Product owner* bertugas sebagai perantara antara *user* dan *scrum team* biasanya *product owner* akan mengkomunikasikan apa saja yang menjadi *product backlog* dan *product goal* secara transparan dan mudah dipahami kepada *scrum team* yaitu *developers.*

- **Developers**

  Developers adalah anggota *scrum team* yang bertugas untuk membuat rencana untuk *sprint* dan *sprint backlog*. Pada *scrum developers* ini terdiri dari *back-end engineer, front-end engineer* dan *quality assurance engineer. Developers* akan mengimplementasikan kerangka kerja pada proses pengembangan *Billing MileApp V3.*

Pada implementasi kerangka kerja menggunakan *scrum* praktikan menerapkan beberapa *scrum artifact* yaitu :

- ***Product Backlog***

  Product Backlog adalah daftar yang muncul dan tersusun tentang apa yang dibutuhkan untuk meningkatkan produk. Ini adalah satu-satunya sumber pekerjaan yang dilakukan oleh *Scrum Team*. *Product Backlog Item* yang dapat dilakukan oleh *Scrum Team* dalam satu *Sprint* dianggap siap untuk dipilih dalam *Sprint Planning.*

- ***Sprint Backlog***

  Sprint Backlog terdiri dari *Sprint Goal* yang terdiri dari serangkaian *Product Backlog Item* yang dipilih untuk *Sprint*, serta rencana yang dapat ditindaklanjuti untuk menghasilkan *Increment*. *Sprint Backlog* adalah kerangka kerja untuk *Developers.*

- ***Product Increment***

  Product increment adalah perhitungan atau peningkatan jumlah dari *product backlog* yang telah diselesaikan selama *sprint* berjalan. *Product increment* merupakan jembatan menuju *product goal*. *Product increment* harus

memenuhi *definition of done.*

3.2.1 Perencanaan Pengujian

Tahapan perencanaan pengujian merupakan tahapan akhir dalam proses pembuatan dan pengembangan modul *Billing* MileApp V3 sebelum *release* ke *public*. Tahap pengujian berada di urutan ke-4 dalam metode *SDLC (Software Development Life Cycle)* menggunakan *agile* yaitu *scrum* di MileApp. Tentunya sebelum fitur *billing* release, praktikan sebagai *Quality Assurance Engineer* bertanggung jawab untuk memastikan tidak adanya *bug,error* atau *defect* pada sistem. Sebelum melakukan pengujian tersebut praktikan harus melakukan perencanaan pengujian yaitu dengan memahami proses bisnis yang berjalan pada sistem dan *user requirement* agar tidak terjadi kesalahpahaman. Dalam tahap ini praktikan sebagai *Quality Assurance Engineer* berkolaborasi dengan *Product Owner, Back-end Engineer* dan *Front-end Engineer.* Setelah memahami bagaimana sistem tersebut bekerja praktikan akan membuat rencana pengujian.Rencana pengujian merupakan tahapan menentukan pengujian apa yang akan dilakukan untuk menguji dan proses pembuatan dokumen uji. Pengujian sistem yang dilakukan terbagi menjadi dua tahapan yaitu pengujian secara fungsionalitas dan struktur *code* menggunakan metode pengujian *black box* dan *white-box*. Dalam proses pengujian *billing MileApp v3* terbagi menjadi 2 cara yaitu pengujian secara manual dan pengujian otomatis menggunakan sebuah *tools testing*. Tahap pengujian manual dan otomatis dilakukan berdasarkan *test case* dan *test scenario* yang telah dibuat oleh praktikan. Setelah pengujian selesai maka praktikan akan membuat dokumentasi hasil pengujian.

3.2.2 *User Story*

Menurut Novan Kurnian (2020) menjelaskan bahwa *user story* merupakan deskripsi persyaratan sistem dalam bahasa alami yang mudah dipahami oleh pengguna akhir tanpa latar belakang TI. *User story* merupakan salah satu dokumentasi dari kebutuhan sistem yang digunakan dalam

metodologi *agile*. Modul *Billing MileApp V3* merupakan pengembangan fitur dari MileApp versi 3.0 yang sebelumnya sudah ada. Aplikasi MileApp memilki beberapa menu diantaranya adalah menu *task*, *route*, *setting* dan *billing*. Adanya pengembangan fitur pada modul billing bertujuan agar pengguna dapat berlangganan aplikasi MileApp dengan melakukan pembayaran secara langsung. Pengembangan yang dilakukan ini berbeda dengan versi sebelumnya dimana pengguna yang ingin berlangganan aplikasi harus menghubungi pihak *marketing* dan melakukan pembayaran secara manual dengan transfer ke nomor rekening perusahaan dan mengirimkan bukti transfer sedangkan untuk *billing mileapp V3* pungguna dapat berlangganan secara langsung dengan membeli fitur *mile route* dan *mile field*.

Dalam modul *billing* terdapat 4 menu diantaranya sebagai berikut :

- **Menu *Plan***

  Pada menu ini *user* dapat melihat informasi *plan mile route* dan *mile field* yang sedang berjalan di bagian *"Current Plan"*. Pada bagian plan juga akan ada penanda *"Active"* dan pengguna dapat melihat informasi detail mengenai daftar lisensi. Terdapat tombol *Downgrade, Upgrade*, dan penanda lisensi yang sedang *Active* . Ada informasi detail perbedaan fitur per lisensinya. *User* juga bisa memilih *subscribe* tahunan atau bulanan.

- **Menu *Subscription***

  Pada menu ini user bisa mengubah plan di halaman *Subscription*, ketika tombol *Change Plan* ditekan, akan mengarah ke halaman *Plan*. Lalu user bisa *Downgrade* atau *Upgrade* plan di halaman Plan. *User* bisa melihat subtotal apabila ada perubahan lisensi pada *hub*. Pada menu ini terdapat tombol *Pay* untuk melakukan pembayaran. Ada 2 tab untuk melihat detail subscription *mile route* dan *mile field*.

- **Menu *Payment Methods***

  Pada menu ini *user* bisa membayar lisensi sesuai total yang ditampilkan dan memilih metode pembayaran selain itu pada menu ini *user* bisa melihat detail metode pembayaran dan bisa melakukan perubahan data serta menambahkan metode pembayaran.

- **Menu *History***

  Pada menu ini user bisa melihat semua daftar invoice yang sudah dibayar. Khusus tampilan *Billing History*, saat ini hanya muncul yang sudah Paid, jadi tidak ada status *unpaid*, tombol *cancel*, dan tombol *pay.*

3.2.3 Requirement Pengujian Manual

Pengujian manual merupakan pengujian perangkat lunak dimana *Quality Assurance* melakukan pengujian secara manual untuk menemukan *bug, error,* dan *defect* pada suatu *software*. Sebelum melakukan pengujian manual seorang *quality assurance* harus menyiapkan dokumen uji terlebih dahulu untuk membantu menjelaskan pendekatan terperinci dan sistematis terhadap sistem yang akan diuji. Dokumen pengujian ini disebut sebagai *test case* dan *test scenario* yang biasanya dibuat sebelum melakukan pengujian. *Test case* merupakan dokumen yang menggambarkan *input* dan tindakan yang dimasukan dalam sistem sesuai dengan respon atau *output* yang diharapkan. Tujuan dari *test case* adalah untuk memastikan bahwa sistem dapat berjalan dengan baik sesuai dengan proses bisnis yang telah dibuat. *Test scenario* adalah langkah-langkah yang dilakukan dalam pengujian untuk menentukan aspek positif dan negatif. Praktikan melakukan pengujian manual untuk *user interface* atau tampilan antarmuka modul *Billing MileApp* V3. Pengujian ini dilakukan untuk memastikan atau menguji tampilan sudah sesuai dengan desain dari tim UI/UX dan *user.* Berikut adalah dokumen uji atau *test case* dan *scenario* yang dibuat oleh praktikan dalam menguji tampilan antarmuka modul *Billing MileApp* V3. Berikut adalah *test case* yang dibuat sebelum melakukan implementasi pengujian manual *billing mileapp V3*. *Test Case* yang dibuat oleh praktikan terdiri dari beberapa *scenario* pengujian diantaranya pengujian positif dan negatif sebagai berikut :

- ***Test Case* Pengujian Untuk Proses Registrasi**

  Test case pengujian untuk proses registrasi user yang belum memiliki akun di MileApp Versi 3.0. *User* tidak dapat melakukan *login* jika belum registrasi terlebih dahulu.

Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
	Register	B01RG01	Register dengan mengisi seluruh data yang valid	Positive	1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button Sign Up	1. Berhasil melakukan registar 2. Berhasil masuk ke halaman web	26/06/2022	Passed by Anisa Lutviani
		B01RG02	Register dengan mengisi seluruh data yang tidak valid	Negative	1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button Sign Up	1. Tidak berhasil melakukan register 2. Tidak berhasil masuk ke halaman web	26/06/2022	Passed by Anisa Lutviani
		B01RG01	Register dengan mengosongkan email dan password	Negative	1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button Sign Up	1. Tidak berhasil melakukan register 2. Tidak berhasilmasuk ke halaman web	26/06/2022	Passed by Anisa Lutviani

Sumber : Hasil dokumentasi praktikan

Gambar 3. 1 Test Case pengujian manual registrasi

- ***Test Case* Pengujian Untuk Proses Login**

Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
	Login	B01LG01	Login dengan menginput semua data yang valid (email dan password)	Positive	1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button login	1. Berhasil login 2. Berhasil masuk ke halaman web	26/06/2022	Passed by Anisa Lutviani
		B01LG02	Login dengan menginput semua data yang tidak valid (email dan password tidak valid)	Negative	1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button Login	1. Tidak Berhasil login 2. Tidak Berhasil masuk ke halaman web	26/06/2022	Passed by Anisa Lutviani
		B01LG03	Login dengan mengosongkan salah satu field yang required (email valid dan password kosong)		1. Masuk https://route.mile.app 2. Masukan username dan password 3. Klik button Login	1. Tidak Berhasil login 2. Tidak Berhasil masuk ke halaman web	26/06/2022	Passed by Anisa Lutviani

Gambar 3. 2 Test Case Pengujian Manual *Login*

Sumber : Hasil dokumentasi praktikan

- **Test Case Pengujian Untuk Proses Plan**

  Test case pengujian manual pada menu plan bertujuan untuk memeriksa *functional* menu *plan* dari sisi *front-end* atau tampilan.

	Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
8		Plan	B01PL01	Melakukan upgrade plan mile route bulanan dengan klik button upgrade pro	Positive	1. login web mileapp 2. klik menu billing 3. pilih submenu plan 4. klik button monthly klik button upgrade	Direct ke menu subscription untuk proses konfirmasi dan pembayaran	26/06/2022	Passed by Anisa Lutviani
9			B01PL02	Melakukan downgrade plan mile route dengan klik button downgrade basic	Positive	1. login web mileapp 2. klik menu billing 3. pilih submenu plan 4. klik button monthly klik button downgrade	Berhasil melakukan downgrade jika masa berlaku plan sebelumnya telah selesai	26/06/2022	Passed by Anisa Lutviani

Gambar 3. 3 Test Case Pengujian Manual Pada halaman *Plan*

Sumber : Hasil dokumentasi praktikan

- ***Test Case* Pengujian Halaman *Subscription***

  Test case pengujian pada halaman *subscription* dibuat dengan tujuan untuk memastikan bahwa proses *subscription* telah sesuai dengan kebutuhan *user* dan terbebas dari *bug.*

	Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
11		Subscription	B01SU01	Upgrade subscription bulanan dari free ke pro dengan jumlah field user bertambah dan vehicle tetap	Positive		Berhasil upgrade dan subscription akan berubah ke pro setelah pembayaran berhasil	18/07/2022	Passed by Anisa Lutviani
12			B01SU02	Upgrade dari free ke pro bulanan dengan field user dan vehicle bertambah	Positive		Berhasil upgrade dan subscription akan berubah ke pro setelah pembayaran berhasil	18/07/2022	Passed by Anisa Lutviani
13			B01SU03	Upgrade dari freeke pro bulanan dengan jumlah field user dan vehicle = 0	Negative		Tidak berhasil melakukan upgrde dan subscription tidak berubah	18/07/2022	Passed by Anisa Lutviani

Sumber : Hasil dokumentasi praktikan

Gambar 3. 4 *Test Case* Pengujian Manual Pada Menu *Subscription*

- **Test case pengujian pada halaman *payment method***

  Test case pengujian *payment method* dibuat sebagai perencanaan pengujian untuk menu *payment method*. Pada halaman ini pengujian dilakukan untuk

mengetahui bahwa metode pembayaran telah berhasil ditambahkan.

Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
	Payment Method	BOPMU01	Menambahkan data payment method baru dengan input card holder name menggunakan tipe data string	Positive	1. Login web mile.app 2. Klik menu billing 3. Klik submenu payment method 4. Klik button new payment method 5. Input card number (valid) 5. Input expired date (valid) 6. Input card holder name (valid) 7. Input CCV (valid) 8. klik save	Berhasil melakukan input data payment method dan dapat digunakan untuk proses pembayaran		
		BOPMU02	Menambahkan data payment method baru dengan input card holder name menggunakan tipe data number	Negative	1. Login web mile.app 2. Klik menu billing 3. Klik submenu payment method 4. Klik button new payment method 5. Input card number (valid) 5. Input expired date (valid) 6. Input card holder name (invalid) 7. Input CCV (valid) 8. klik save	Tidak berhasil input metode pembayaran dan tidak dapat melanjutkan transaksi		

Gambar 3. 5 Test Case Pengujian Manual Pada Halaman *Payment Method*

Sumber : Hasil dokumentasi praktikan

- **Test Case Pengujian Pada Halaman Billing History**

  Pada halaman ini pengujian dilakukan untuk mengetahui pembayaran yang dilakukan telah berhasil masuk ke halaman *billing history* dengan beberapa status seperti sudah selesai, menunggu pembayaran dan belum dibayar.

Module	Feature	ID Scenario	Scenario	Type	Test Steps	Expected Result	Last Tested	Manual Tested
	Billing History	BL008TC1	Melihat billing history	Positive	1. Login Mile.app 2. Klik menu billing 3. Klik submenu billing history	1. Berhasil melakukan login 2. Masuk ke menu billing 3. Tampil halaman billing history dan muncul dialog pembayran serta menampilkan semua daftar invoice yang sudah dibayar atau paid	20/07/2022	Passed by Anisa Lutviani
		BL008TC2	View invoice details	Positive	1. Login Mile.app 2. Klik menu billing 3. Klik submenu billing history 4. Klik icon mata	1. Berhasil melakukan login 2. Masuk ke menu billing 3. Menampilkan detail invoice (client, invoice number , invoice date dan subscription, total, item)	20/07/2022	Passed by Anisa Lutviani

Gambar 3. 6 Test Case Pengujian Pada Halaman Billing History

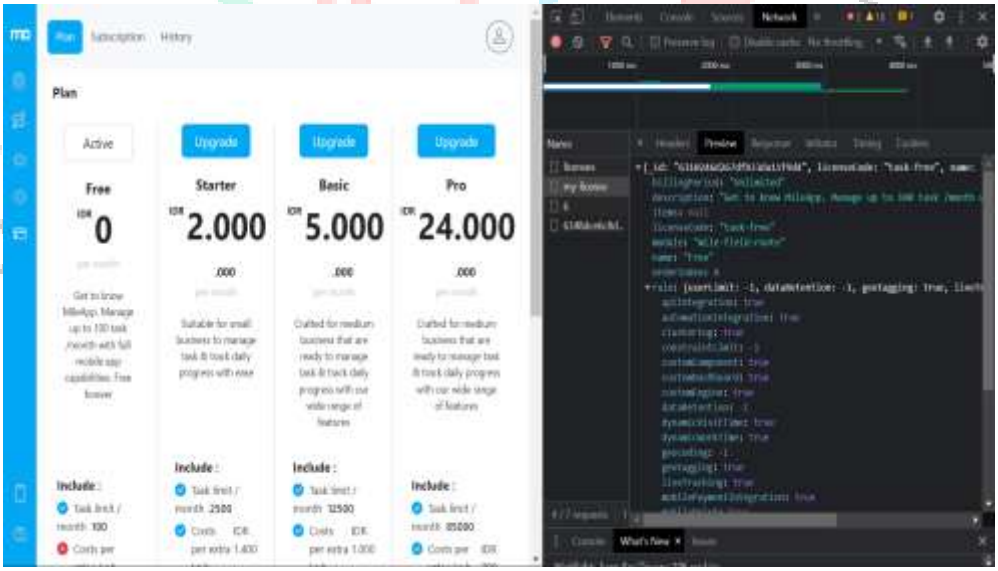
Sumber : Hasil dokumentasi praktikan

3.2.4 Pengujian Manual Pada *Front-End*

Pengujian manual merupakan jenis pengujian perangkat lunak dimana *Quality Assurance* bertanggung jawab untuk menemukan *bug, error,* dan *defect* pada suatu *software* satu persatu secara manual. Pengujian manual merupakan implementasi dari metode *black-box testing* . Menurut Nidhra dan Dondeti (2012:1), *black-box testing* juga disebut *functional testing*, sebuah teknik pengujian fungsional yang merancang test case berdasarkan informasi dari spesifikasi. Pengujian ini berperan untuk memeriksa fungsionalitas aplikasi dari sisi *user* dimana penguji tidak mengetahui atau memeriksa struktur internal dan kode dari aplikasi tersebut.

Berikut pengujian manual yang dilakukan oleh praktikan pada modul *billing MileApp V3* terdapat menu *plan, subscription dan billing history*

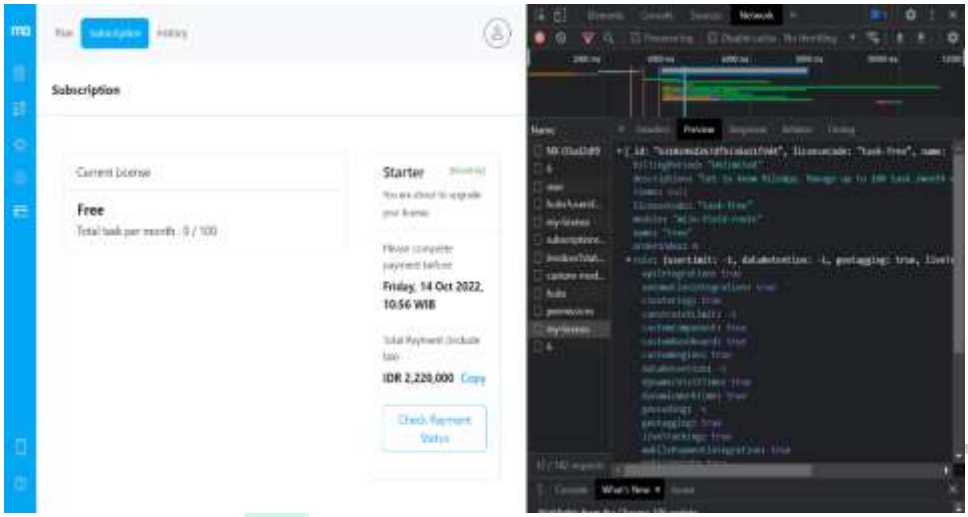
- **Menu *Plan***



Gambar 3. 7 Proses Pengujian Manual Pada Menu *Plan*

Sumber : Hasil dokumentasi praktikan

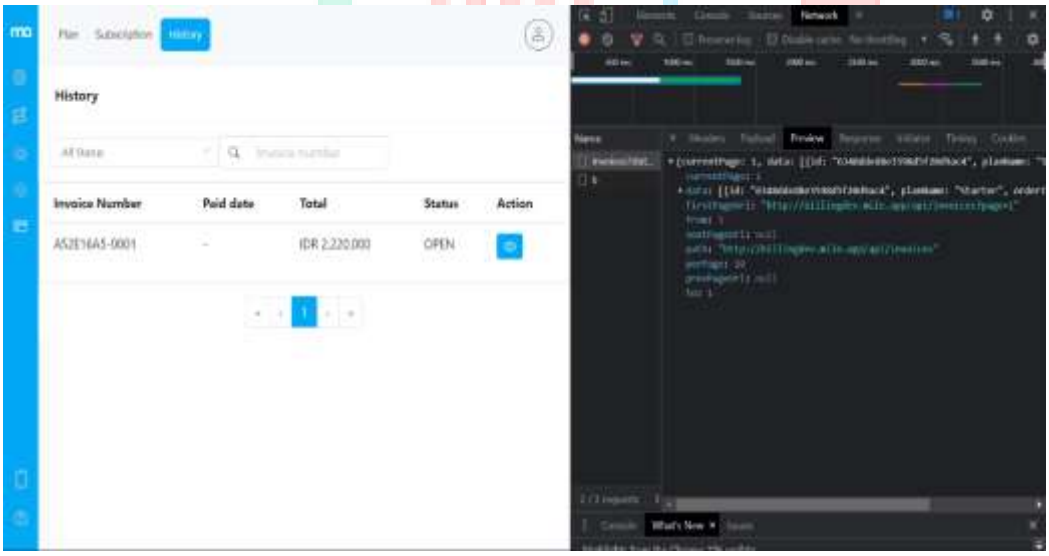
- **Menu *Subscription***



Gambar 3. 8 Proses Pengujian Manual Pada Menu *Subscription*

Sumber : Hasil dokumentasi praktikan

- **Menu *Billing History***



Gambar 3. 9 Proses Pengujian Manual Pada Menu *Billing History*

Sumber : Hasil dokumentasi praktikan

3.2.5 *Requirement* Pengujian Otomatis

Pengujian otomatis atau yang biasa disebut *automation testing* merupakan pengujian yang dilakukan untuk menemukan *error, bug* dan *defect* pada sistem dengan menggunakan *testing tools*. Pengujian otomatis sering diterapkan karena waktu yang dibutuhkan untuk melakukan pengujian lebih singkat dan meminimalisasi adanya *human error* dalam pengujian. Salah satu *testing tools* yang digunakan oleh praktikan dalam melakukan pengujian otomatis adalah Postman dan Katalon. Postman sendiri merupakan sebuah aplikasi yang berfungsi sebagai *rest client* untuk melakukan pengujian *REST API.* Pada sistem Billing MileApp V3 praktikan melakukan pengujian otomatis terhadap *API (Application Programming Interface)* menggunakan *automation testing* yaitu *postman.*

3.2.5.1 API (Application Programming Interface)

Menurut Faradila (2022) pada artikelnya menjelaskan bahwa API (Application Programming Interface) adalah antarmuka yang fungsinya sebagai penghubung antara *client* dan *server,* API sangat penting dalam sebuah aplikasi karena dengan adanya API memungkinkan dua komponen pada aplikasi dapat saling berhubungan dan berkomunikasi melalui serangkaian protokol, baik pada satu platform yang sama maupun lintas platform. API bekerja dengan cara *user* mengakses aplikasi API lalu API akan membuat permintaan ke server dan server akan memberikan merespons API. Dengan adanya API memudahkan para *developer* dalam menyelesaikan tugas dan pekerjaan karena API bisa digunakan untuk mengkomunikasikan bahasa pemrograman yang berbeda. Pada masa kerja praktik, praktikan melakukan pengujian API yang digunakan dalam modul Billing MileApp V3 menggunakan *method* sebagai berikut :

Tabel 3. 1 Method Pengujian

Method	**Fungsi**
GET	Menampilkan data berdasarkan *request* dari sisi *client*
POST	Melakukan *create* data
PUT	Melakukan *update* atau *edit* seluruh data
PATCH	Melakukan *update* atau *edit* sebagian data
DELETE	Menghapus data

Sumber : Hasil dokumentasi praktikan

Berikut *endpoint* API yang digunakan untuk melakukan pengujian

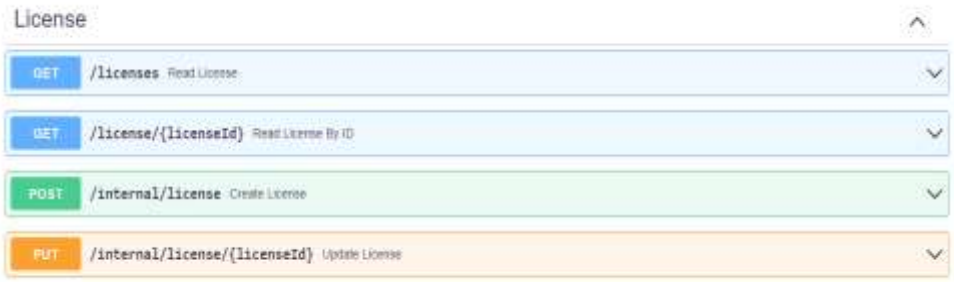
- *Endpoint API Invoices* digunakan pada modul *Billing MileApp* V3 di bagian menu *history* untuk menampilkan data *history* pembayaran *mile route* atau *mile field* yang telah dibayar oleh *user* menggunakan *endpoint API* method seperti dibawah ini :

Gambar 3. 10 Endpoint API Invoices

Sumber : Hasil dokumentasi praktikan

- Endpoint API License digunakan pada modul *Billing MileApp* V3 yaitu di menu *plan* untuk menambah, merubah dan menampilkan list data *billing*

plan yang berguna bagi *user* melihat detail harga *mile route* dan *mile field* sebelum subscribe menggunakan *endpoint API* method seperti dibawah ini :



Gambar 3. 11 Endpoint API Licenses

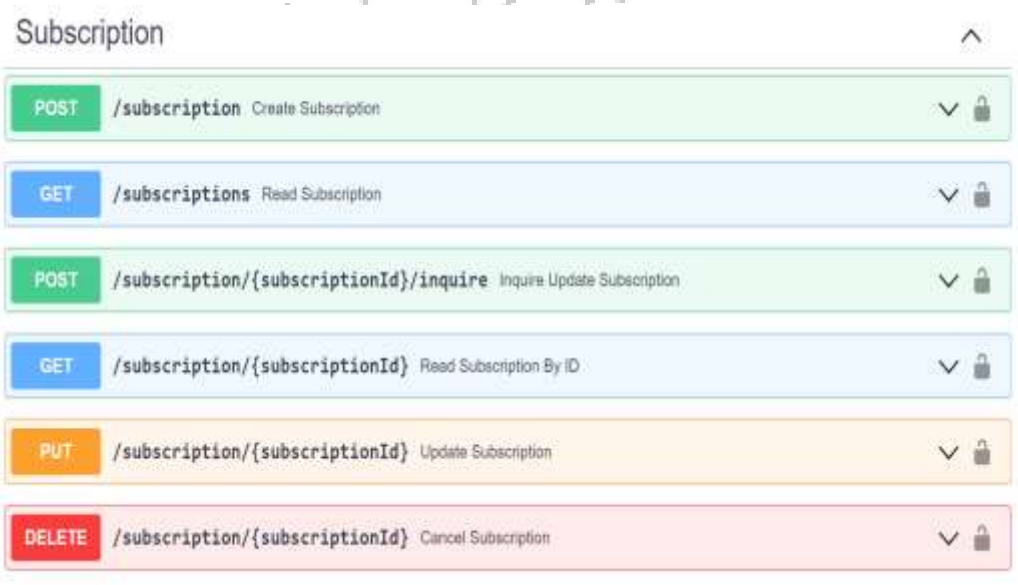
Sumber : Hasil dokumentasi praktikan

- Endpoint API *payment methods* digunakan pada menu *payment methods* untuk proses menambahkan metode pembayaran, melihat detail pembayaran dan menghapus metode pembayaran dengan minimal menyisakan satu jenis *credit card* menggunakan *endpoint API* method seperti dibawah ini :

Gambar 3. 12 Endpoint API Payment Method

Sumber : Hasil dokumentasi praktikan

- Endpoint API subscription digunakan pada menu *subscription* untuk melihat detail plan *mile route* dan *mile field* yang sedang berjalan setelah berhasil melakukan pembayaran dan detail *plan* yang telah berakhir dan yang akan segera berakhir menggunakan *endpoint API* method seperti dibawah ini :



Gambar 3. 13 Endpoint API Subscription

Sumber : Hasil dokumentasi praktikan

3.2.5.2 Pengujian API (Application Programming Interface)

Menurut Fadhila Rizki (2020) menjelaskan dalam artikelnya bahwa pengujian API (*Application Programming Interface*) atau yang biasa disebut *API testing* merupakan pengujian perangkat lunak yang berguna untuk memastikan dan memvalidasi antarmuka pemrograman aplikasi dari sisi *back-end* dengan metode *white-box testing*. *API testing* ini bertujuan untuk memastikan fungsionalitas, kinerja dan keamanan antarmuka pemrograman berjalan sesuai dengan standarisasi. Pengujian dilakukan dengan menggunakan *tools* yang dapat mengirim *request* ke API agar mendapatkan *respons.* Salah satu *tools* yang digunakan oleh praktikan dalam melakukan pengujian otomatis *billing mileapp V3* yaitu *Postman. Postman* merupakan

sebuah aplikasi yang fungsinya sebagai *REST CLIENT* untuk melakukan pengujian API *(Application Programming Interface).* Postman biasa digunakan oleh para *developers* termasuk praktikan sebagai *Quality Assurance Engineer* untuk melakukan pengujian API. Cara kerjanya yaitu dengan mengirimkan *request method*, *url* dan *body* dan token jika diperlukan lalu postman akan mengirimkan *respons* tertentu sesuai dengan kode status HTTP. Berikut adalah kode status HTTP yang biasanya muncul pada saat praktikan melakukan pengujian API pada modul *billing mileapp V3*.

Kode *Error* yang biasa muncul pada saat praktikan melakukan pengujian :

Tabel 3. 2 Kode error yang muncul pada saat pengujian otomatis

No	Error Code	Text	Description
1.	401	*Unauthorized*	*Unauthorized* adalah kode status HTTP yang menunjukan bahwa *request* yang dikirim tidak terautentikasi karena ada beberapa kesalahan dalam ID *user* dan password.
2.	400	*Bad Request*	*Bad request* adalah kode status HTTP yang menunjukan adanya *request* yang tidak valid sehingga *server* tidak memahami *request* karena *syntax invalid*

3.	404	*Not Found*	*Not found* adalah kode status HTTP yang menunjukan bahwa permintaan URL website tidak ditemukan atau tidak bisa dibuka
4.	405	*Method not allowed*	*Method not allowed* merupakan *error* yang menunjukan bahwa *method* yang digunakan salah dan biasanya URL yang diinput sudah benar tetapi *web server* tidak mengizinkan pengguna membuka halaman karena penggunaan *method* yang *invalid*
5.	500	*Unexpected Server Error*	*Unexpected Server Error* merupakan *error* yang mendeteksi adanya kesalahan pada *server* atau *server* tidak bisa mendeteksi adanya kesalahan namun

			tidak tahu bagaimana penyebabnya
6.	503	*Server Unavailable*	*Server Unavailable* merupakan kode status HTTP yang menunjukan penggunaan *resource* melebihi kapasitas dan *server* tidak bisa menangani permintaan.

Sumber : Hasil Dokumentasi Praktikan

Berikut adalah *test case* yang dibuat sebelum melakukan implementasi pengujian otomatis

- **Test case API untuk melakukan Register**

	Feature	Endpoint API	Method	ID Scenario	Scenario	Type	Expected Result	Automation Test
2	Registrasi	api/register	POST	BLGTC001	POST register dengan data yang valid	Positive	1. Sukses 200 OK 2. Berhasil melakukan POST register	Passed
3				BLGTC002	POST register dengan data email yang invalid	Positive	1. Not found 400 2. Tidak berhasil menampilkan data	Passed
4				BLGTC003	POST register dengan data email yang udah terdaftar	Positive	1. Not found 400 2. Tidak berhasil menampilkan data	Passed

Gambar 3. 14 Test Case pengujian otomatis register menggunakan *method* POST

Sumber : Hasil dokumentasi praktikan

- **Test case API untuk Login**

	Feature	Endpoint API	Method	ID Scenario	Scenario	Type	Expected Result	Automation Test
3	Login	api/login	POST	BLGTC004	POST login dengan mengisi semua field menggunakan data yang valid	Positive	1. Sukses 200 OK 2. Berhasil melakukan POST login 3. Menampilkan data user login	Passed
4				BLGTC005	POST login dengan mengisi semua field menggunakan data yang tidak valid	Negative	1. Not found 400 2. Tidak berhasil menampilkan data user login	Passed
5				BLGTC006	POST login dengan mengosongkan salah satu field required (username dikosongkan sedangkan password diisi)	Negative	1. Not found 400 2. Tidak berhasil menampilkan data user login	Passed

Sumber : Hasil dokumentasi praktikan

Gambar 3. 15 Test Case pengujian otomatis *login* menggunakan *method* POST

- **Test Case Pengujian API License**

	Feature	Endpoint API	Method	ID Scenario	Scenario	Type	Expected Result	Automation Test
8	Plan	api/licenses	GET	BLGTC007	GET data license dengan menggunakan data yang valid	Positive	1. Sukses 200 OK 2. Menampilkan data license	Passed
9				BLGTC008	GET data license tanpa menggunakan auth	Negative	Tidak berhasil menampilkan data license	Passed
10				BLGTC009	GET data license tanpa menggunakan	Negative	Muncul pesan error	Passed

Gambar 3. 17 Test Case pengujian otomatis license menggunakan *method* GET pada halaman *plan*

Sumber : Hasil dokumentasi praktikan

- **Test case pengujian API untuk halaman subscription**

	Feature	Endpoint API	Method	ID Scenario	Scenario	Type	Expected Result	Automation Test
11	Subscription	api/subscription/{subscriptionId}	POST	BLGTC010	POST subscription dengan input data route yang valid	Positive	1. Sukses 200 OK 2. Menampilkan data subscription	Passed
12				BLGTC011	POST subscription dengan input data route yang invalid	Negative	1. 400 : not found 2. Tidak berhasil menampilkan data route	Passed
13				BLGTC012	POST subscription dengan mengosongkan data route	Negative	1. 400 : not found 2. Tidak berhasil menampilkan data route	Passed

Sumber : Hasil dokumentasi praktikan

Gambar 3. 18 Test Case pengujian otomatis menggunakan *method POST* pada halaman *plan*

- **Test case API payment method**

	Feature	Endpoint API	Method	ID Scenario	Scenario	Type	Expected Result	Automation Test
11				BLGTC0012	POST subscription dengan menggoongkan data route	Negative	1. 400 : not found 2. Tidak berhasil menampilkan data route	Passed
14	Payment Method	api/payment-method	GET	BLGTC0013	GET payment method menggunakan data valid	Positive	1. Sukses 200 OK 2. Menampilkan data payment method	Passed
15				BLGTC0014	GET payment method menggunakan data tidak valid	Negative	1. 400 : not found 2. Tidak berhasil menampilkan data payment method	Passed
16				BLGTC0015	GET payment method menggunakan method yang tidak sesuai	Negative	1. 400 : not found 2. Tidak berhasil menampilkan data payment method	Passed

Gambar 3. 19 Test Case pengujian otomatis menggunakan method GET pada halaman payment method

Sumber : Hasil dokumentasi praktikan

3.2.5.3 Pengujian Otomatis Pada *Back-End*

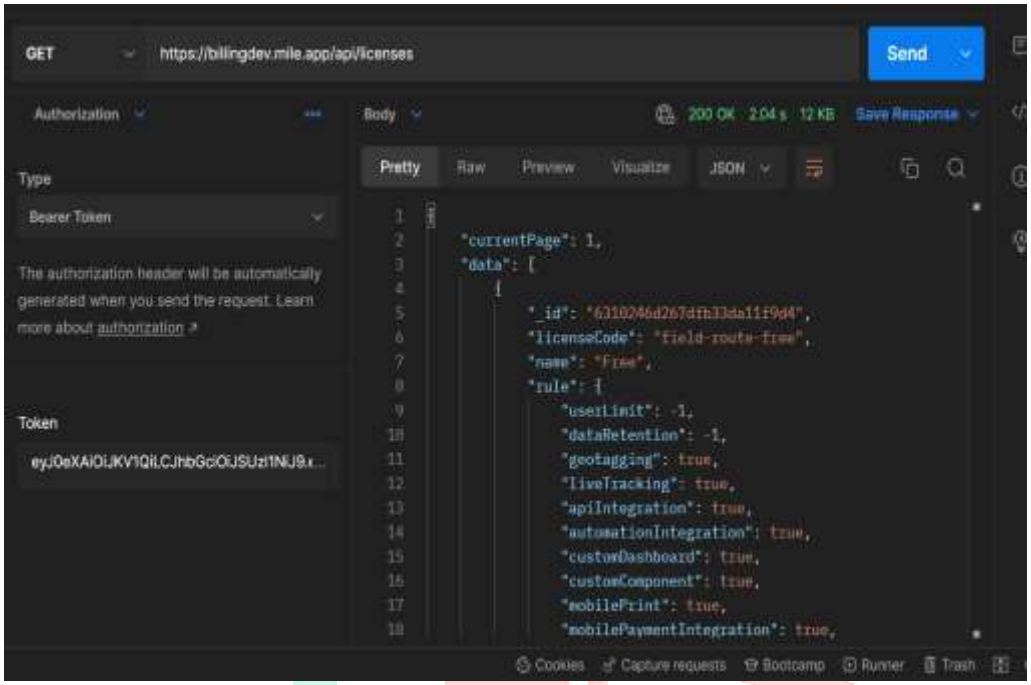
Pengujian Endpoint API Pada Modul Billing MileApp V3

Pengujian ini dilakukan untuk menguji API yang akan digunakan pada modul *billing* :

- **Method GET**

  Pengujian ini dilakukan untuk menguji API license yang fungsinya untuk GET data atau list license pada menu *billing plan.* Berikut cara kerja pengujian API menggunakan Postman :

1. Pilih method yang akan digunakan yaitu method GET
2. Masukan *request url*
3. Masukan authorization
4. Klik button “send” untuk mengirimkan permintaan
5. Postman akan memberikan response dan hasil



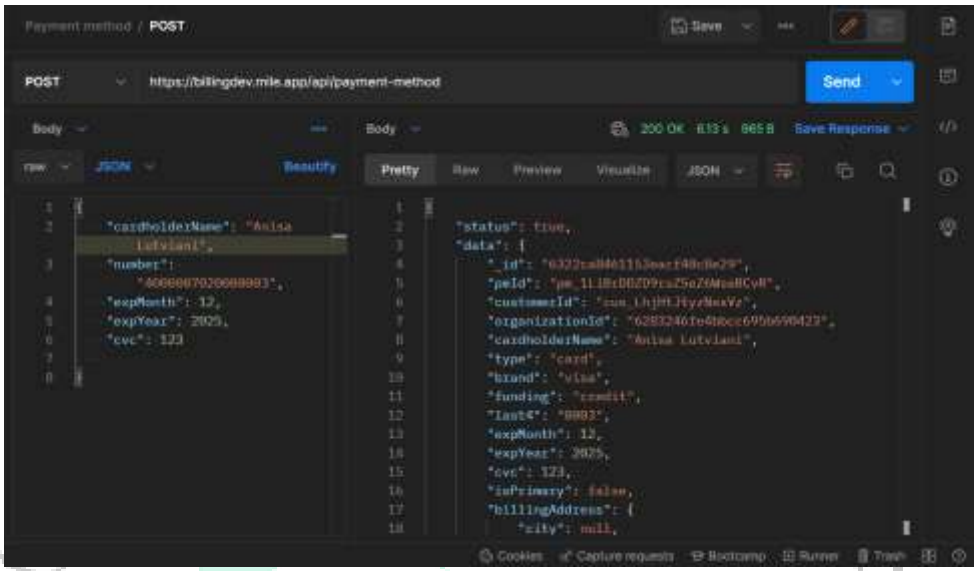
Gambar 3. 20 Pengujian Otomatis Method GET

Sumber : Hasil dokumentasi praktikan

- **Method POST**

  Pengujian ini dilakukan untuk menguji API payment method fungsinya untuk menambahkan data pembayaran yang akan digunakan oleh *user* transaksi pada menu payment method. Berikut cara kerja pengujian API menggunakan Postman :

  1. Pilih method yang akan digunakan yaitu method POST
  2. Masukan *request url*
  3. Masukan authorization
  4. Masukan *request body*
  5. Klik button “send” untuk mengirimkan permintaan
  6. Postman akan memberikan response dan hasil



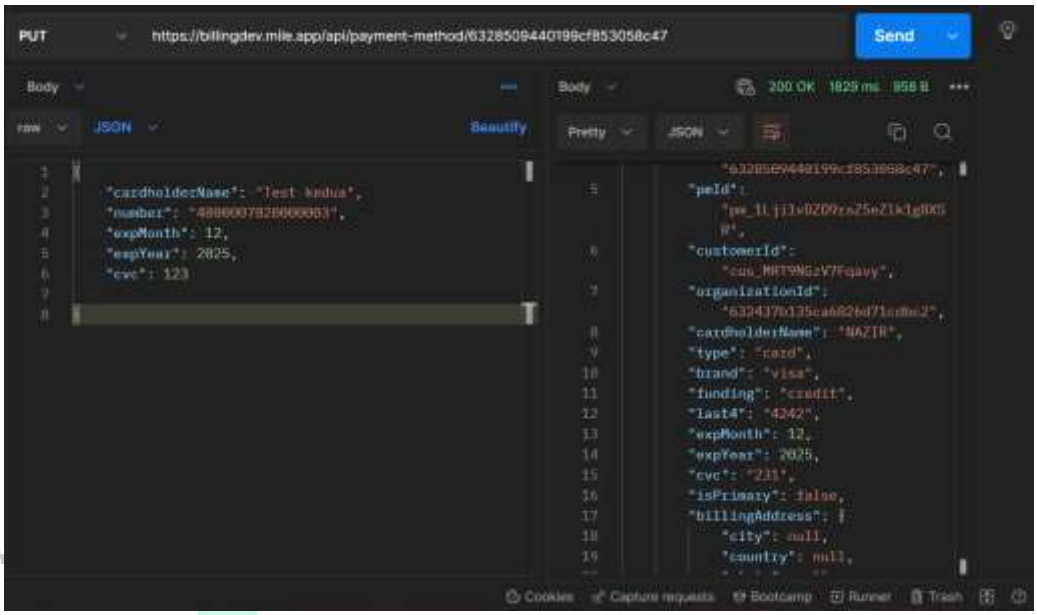
Gambar 3. 21 Penggujian Otomatis Method POST

Sumber : Hasil dokumentasi praktikan

- **Method PUT**

  Pengujian method PUT pada menu payment method bertujuan untuk menguji fungsional edit data payment method dari sisi *back-end* :

  1. Pilih method yang akan digunakan yaitu method PUT
  2. Masukan *request url*
  3. Masukan authorization
  4. Masukan *request body*
  5. Klik button “send” untuk mengirimkan permintaan
  6. Postman akan memberikan response dan hasil



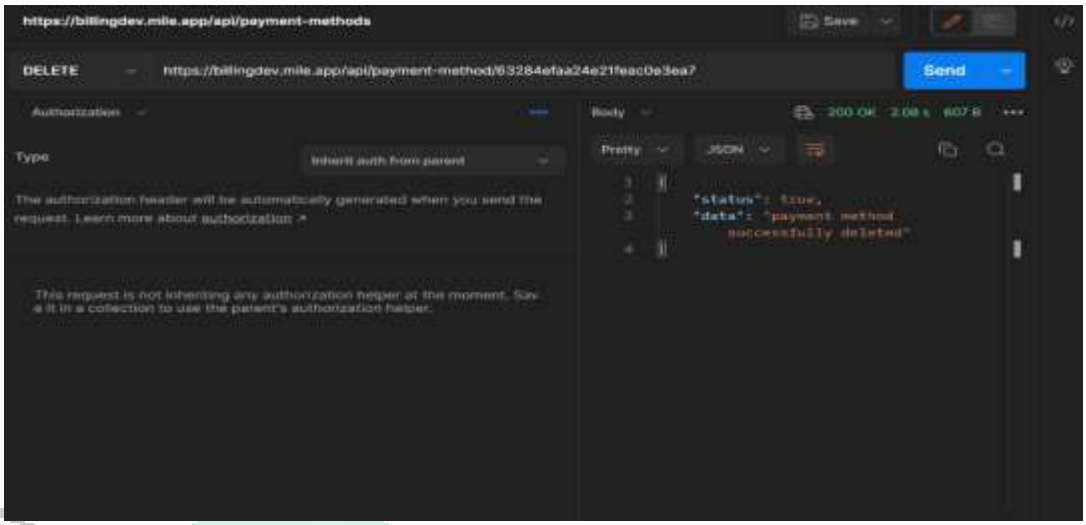
Gambar 3. 22 Penggujian Otomatis Method PUT

Sumber : Hasil dokumentasi praktikan

- **Method DELETE**

  Pengujian method DELETE pada menu payment method bertujuan untuk menguji fungsional edit data payment method dari sisi *back-end* :

  1. Pilih method yang akan digunakan yaitu method DELETE
  2. Masukan *request url* beserta Id payment method
  3. Klik button “send” untuk mengirimkan permintaan
  4. Postman akan memberikan response dan hasil



Gambar 3. 23 Pengujian Otomatis Method Delete

Sumber : Hasil dokumentasi praktikan

3.2.4 Desain Sistem dan Implementasi Sistem

- **Halaman Login**

Gambar 3. 24 Mock-Up Halaman Login

Sumber : Hasil dokumentasi praktikan

Merupakan tampilan halaman login pada halaman ini *user* yang telah melakukan pendaftaran bisa masuk ke halaman dashboard dengan memasukan email dan password yang valid. Jika email tidak valid maka akan muncul pesan bahwa data yang diinput tidak valid.

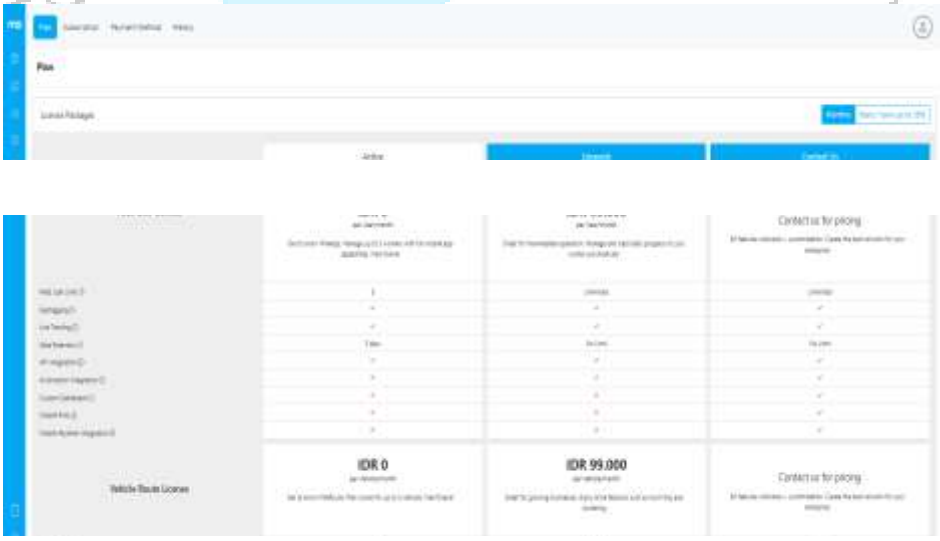
- **Halaman Sign Up**

Gambar 3. 25 Mock-Up Halaman Sign-Up

Sumber : Hasil dokumentasi praktikan

Merupakan tampilan pada halaman Sign Up. Dihalaman ini user dapat melakukan pendaftaran jika ingin masuk ke dalam dashboard MileApp

- **Halaman Menu Plan**



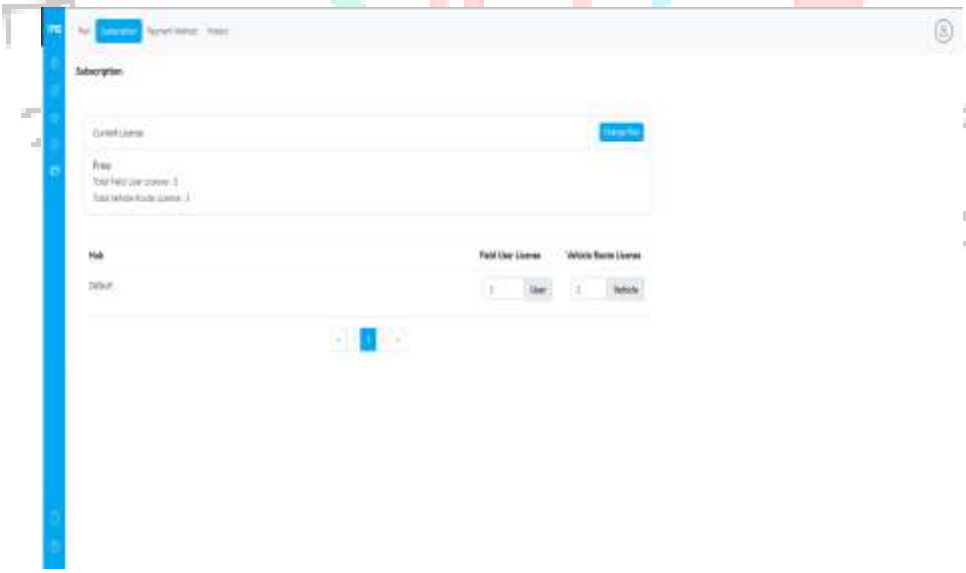
Gambar 3. 26 Mock-Up Halaman Plan

Sumber : Hasil dokumentasi praktikan

Pada halaman ini *user* bisa melihat informasi *plan mile route* dan *mile field* yang sedang berjalan di bagian *"Current Plan"*. Pada bagian plan juga akan ada penanda *"Active"* dan pengguna dapat melihat informasi detail mengenai daftar lisensi. Terdapat tombol *Downgrade, Upgrade*, dan penanda lisensi yang sedang *Active* . Ada informasi detail perbedaan fitur per lisensinya. User juga bisa memilih *subscribe* tahunan atau bulanan.

- **Halaman Menu Subscription**

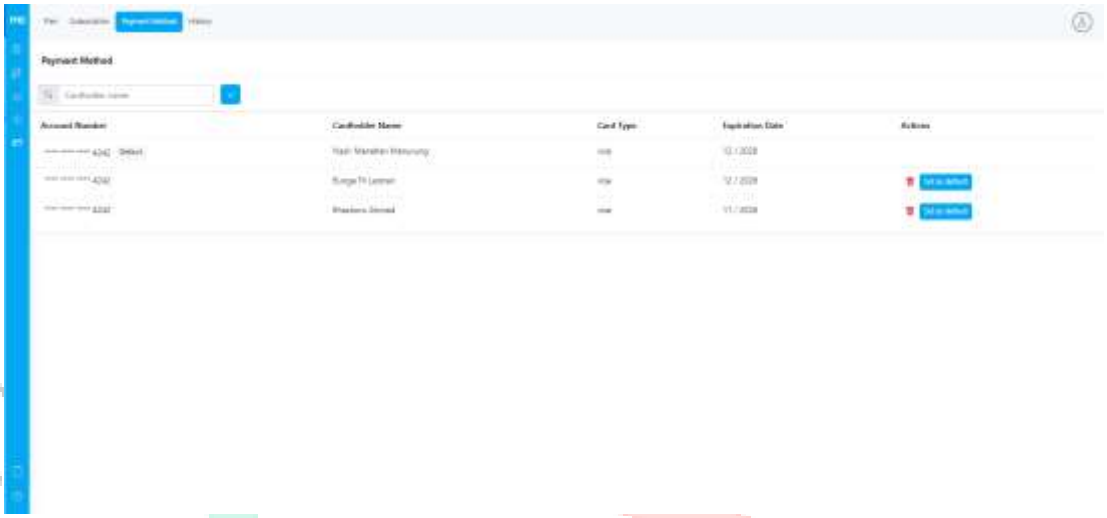
  Pada menu ini user bisa mengubah plan di halaman *Subscription*, ketika tombol *Change Plan* ditekan, akan mengarah ke halaman *Plan*. Lalu user bisa Downgrade atau Upgrade plan di halaman Plan. User bisa melihat subtotal apabila ada perubahan lisensi pada hub. Pada menu ini terdapat tombol Pay untuk melakukan pembayaran. Ada 2 tab untuk melihat detail subscription *mile route* dan *mile field*.



Gambar 3. 27 Mock-Up Halaman Subscription

Sumber : Hasil dokumentasi praktikan

- **Halaman Menu Payment Method**



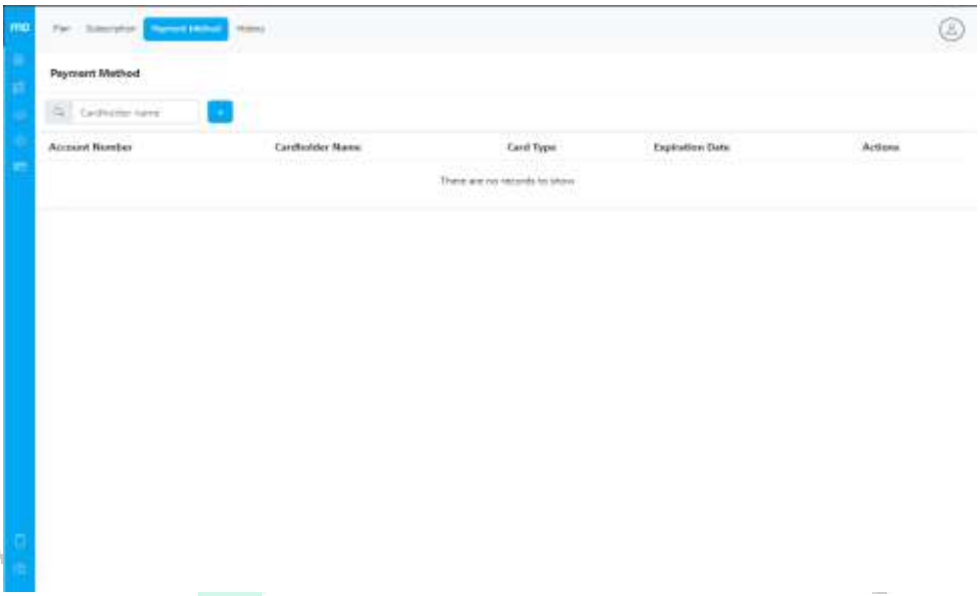
Gambar 3. 28 Halaman Payment Method

Sumber : Hasil dokumentasi praktikan

Pada menu ini user bisa membayar lisensi sesuai total yang ditampilkan dan memilih metode pembayaran selain itu pada menu ini user bisa melihat detail metode pembayaran dan bisa melakukan perubahan data serta menambahkan metode pembayaran.

- **Halaman Menu History**

Pada menu ini user bisa melihat semua daftar invoice yang sudah dibayar. Khusus tampilan Billing History, saat ini hanya muncul yang sudah Paid, jadi tidak ada status Unpaid, tombol Cancel, dan tombol Pay**.**



Gambar 3. 29 Mock-Up Halaman Billing History

Sumber : Hasil dokumentasi praktikan

1.2 Kendala yang Dihadapi

Kendala yang dihadapi oleh praktikan dalam kegiatan kerja profesi sebagai berikut :

1. Kerja profesi yang dilakukan secara WFH (*Work From Home)* mengakibatkan komunikasi kurang berjalan lancar dan maksimal terutama dengan product owner. Komunikasi yang kurang maksimal ini menyebabkan beberapa kegiatan dalam Scrum seperti *sprint planning*, *sprint review*, *sprint retrospective* tertunda dan berjalan lambat akibat gangguan koneksi dan jadwal yang padat dari pihak *product*.
2. Proses perancangan modul Billing yang baru, berjalan tanpa bantuan *system analyst* dan UI/UX designer sehingga terdapat beberapa desain dan alur sistem yang tidak sesuai dan harus diubah secara berulang kali setelah *sprint review.*
3. Pada saat awal pelaksanaan kerja profesi praktikan tidak mengetahui bagaimana mengoperasikan pengujian otomatis *API (Application Programming Interface)* menggunakan Postman dan Katalon Studio.

4. Praktikan tidak mengetahui bagaimana menuangkan dokumentasi dan report hasil pengujian jika ada *bug* atau *issue* dikarenakan kurangnya penyampaian informasi terkait detail job desk.

1.3 Cara Mengatasi Kendala

Praktikan mencari cara untuk mengatasi kendala yang dihadapi. Berikut adalah cara yang dilakukan oleh praktikan untuk mengatasi kendala yang dihadapi :

1. Berkomunikasi dengan *team product* dan memastikan sebelum agenda pelaksanaan s*print planning*, *sprint review*, s*print retrospective* dimulai tidak ada hambatan atau penghalang serta agenda harus berjalan dengan semestinya. Jika agenda tersebut harus tertunda maka s*print planning*, *sprint review*, *sprint retrospective* dilakukan secara mandiri oleh internal tim.
2. Memanfaatkan media discord untuk berdiskusi dengan pembimbing dan developers terkait desain yang sesuai dengan standar. Serta mencari referensi terkait standarisasi bootstrap.
3. Mencari berbagai referensi dari banyak sumber melalui internet terkait cara mengoperasikan pengujian otomatis menggunakan Postman dan Katalon.
4. Mencari berbagai referensi terkait pembuatan dokumentasi yang dibuat oleh *Quality Assurance* dan mempelajarinya secara mandiri.

3.4 Pembelajaran Yang Diperoleh Dari Kerja Profesi

Selama menjalankan kerja profesi di MileApp (PT.Paket Informasi Digital) praktikan mendapatkan pembelajaran dan pengalaman mengenai beberapa hal yaitu:

1. Mampu memahami tentang tugas dan tanggung jawab sebagai *Quality Assurance Engineer* sebagai penjamin mutu software pada perusahaan .
2. Memahami penerapan SDLC (*Software Development Life Cycle*)

menggunakan metode *agile development* yaitu *scrum* pada sebuah perusahaan.

3. Mampu mengimplementasikan pembelajaran yang didapatkan selama kuliah kedalam kerja praktek yaitu pengujian manual dan otomatis menggunakan metode *Blackbox* dan *Whitebox.*
4. Mahasiswa mampu beradaptasi dengan lingkungan kerja dan berkolaborasi bersama team product dan developers
5. Meningkatkan *softskill* dan *hardskill* selama melaksanankan kerja profesi