BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada program magang ini, mahasiswa magang memiliki posisi atau kedudukan sebagai *back-end developer* untuk membuat dan mengembangkan *API* pada *website* bimbingan akademik. Selama magang berlangsung, mahasiswa juga dibantu oleh *supervisor* dan sekaligus sebagai salah satu *product owner*, yaitu Ibu Wella, S.Kom., M.MSI, serta tim IT yang membantu mahasiswa magang terkait teknis seperti *tools* atau bahasa pemrograman yang diperlukan dalam pengerjaan *back-end website*.

Program magang atau kegiatan magang ini dilakukan pada tanggal 6 September 2023 dan di hari pertama magang, mahasiswa menghubungi *product owner*, tim IT, dan juga rekan magang yang sudah melakukan kerja magang lebih awal untuk meminta *file* atau *folder project website* bimbingan akademik yang sebelumnya sudah ada di tahap penyelesaian bagian *front-end*. Setelah mahasiswa mendapati *folder project* tersebut, langkah yang dilakukan selanjutnya adalah mempelajari setiap *file* atau dokumen terkait dengan tujuan memahami alur bisnis *website*, serta sebagai acuan untuk mengembangkan *back-end*. *Tools* dan juga bahasa yang digunakan hampir sama seperti bagian *front-end*, tetapi untuk dapat mengembangkan *back-end*, mahasiswa mempelajari *framework Express* dan konsep *back-end* lainnya. Mahasiswa juga dianjurkan untuk memahami bahasa pemrograman *JavaScript* sebagai pondasi untuk memahami *framework React* yang ada pada bagian *front-end*. Selain *React*, terdapat *framework React-Redux* yang berguna untuk mengelola *state* pada *React*.

Meeting pertama pada program magang ini dilakukan pada tanggal 18 September 2023 untuk membahas pembelajaran *tools* atau bahasa pemrograman yang sudah dipelajari oleh mahasiswa, serta tugas mahasiswa selanjutnya. Secara garis besar, *meeting* perdana ini menganjurkan mahasiswa untuk menerapkan pembelajaran mandiri secara lebih lanjut agar benar-benar memahami *tools* atau

bahasa untuk penerapan *back-end* sehingga dapat mengembangkan *API* pada *website*. Kemudian, dikarenakan mahasiswa magang hanya berjumlah dua orang, maka pengerjaan fitur-fitur pada *website* dibagi secara rata, dimana fitur pertama yang akan dikerjakan adalah fitur *Calendar*. Untuk pengerjaan fitur lainnya setelah satu fitur tersebut dilakukan secara inisiatif dari mahasiswa magang.

Mahasiswa magang juga berdiskusi dengan tim IT seputar penerapan sistem dari *back-end website* selama melakukan kegiatan magang. Terdapat struktur atau standar yang telah ditetapkan oleh tim IT, seperti penggunaan *database Oracle* untuk menampung data-data pada *website*, kemudian penerapan MVC (*Model, View, Controller*) untuk menerapkan *API*, dan juga yang lainnya. Kemudian, untuk memudahkan kolaborasi antar mahasiswa magang, tim IT juga menyediakan *repository* pada *GitHub*.

Dalam melakukan koordinasi dan juga kegiatan magang mahasiswa, sehari-harinya dilakukan di Laboratorium Big Data yang terletak pada Gedung C UMN, di ruangan nomor 503 (C503). Ruangan tersebut juga dijadikan sebagai tempat atau lokasi untuk *meeting* yang dilakukan secara berkala. Koordinasi juga dapat dilakukan melalui *e-mail* seperti pada saat melakukan *update daily task* atau juga untuk menghubungi tim IT, serta *platform chat* seperti *WhatsApp* dan *Line*.

Kemudian secara garis besar, koordinasi yang dilakukan mahasiswa dengan *supervisor* atau *product owner* adalah untuk membahas hal seperti tampilan atau hasil jadi fitur yang dikerjakan, serta mendengarkan *feedback* terkait fitur yang dikerjakan untuk dilakukan revisi. Untuk koordinasi yang dilakukan antara mahasiswa dengan tim IT adalah membahas hal teknis seperti bahasa pemrograman yang digunakan, standar tertentu dalam menerapkan *API*, dan yang lainnya.

3.2 Tugas dan Uraian Kerja Magang

Sebagian besar tugas dan aktivitas magang yang dilakukan sebagai *back-end developer* adalah membuat *API* dan juga menghubungkan *API* tersebut pada fitur yang ada di *website* bimbingan akademik, kemudian juga membuat *ERD*, serta *Database* untuk keperluan fitur. Dalam mengembangkan dan membuat *API*

tersebut, mahasiswa magang menggunakan bahasa pemrograman *Node JS* dan *Express*, serta dibantu penggunaan *tools* seperti *SQL Developer* yang terintegrasi dengan *database Oracle* untuk memudahkan pembuatan *table* pada *database* yang ada di masing-masing fitur. Selain tugas pengembangan *back-end* dan *API*, mahasiswa juga sempat membantu penerapan fitur atau tampilan baru, salah satunya *Student Notes* yang ditempatkan pada halaman *Student Academics*. Tampilan tersebut dibuat menggunakan *framework React JS* dan *React-Redux* yang juga sudah digunakan pada semua fitur *website* bimbingan akademik.

Berikut ini adalah Tabel 3.1 yang menjelaskan aktivitas-aktivitas magang mahasiswa selama mengembangkan *back-end website* bimbingan akademik UMN.

Tabel 3. 1 *Timeline* Aktivitas Kerja Magang Mahasiswa

No	Kegiatan Magang	Mulai	Selesai
1	**Pengenalan *project website* bimbingan akademik**		
1.1	Meminta *folder project* kepada *product owner* dan melakukan *install tools* yang akan digunakan dalam mengembangkan *back-end website* bimbingan akademik	06/09/2023	08/09/2023
1.2	Mengurus administrasi berupa kontrak kerja magang dengan *HRD*	11/09/2023	12/09/2023
1.3	Memahami *business flow* dan *requirements* pada *website* bimbingan akademik	13/09/2023	15/09/2023
2	**Pembelajaran *tools* pada *project website* bimbingan akademik**		
2.1	Melakukan *meeting* untuk membahas pembagian tugas yang akan dikerjakan	18/09/2023	18/09/2023
2.2	Melakukan pembelajaran mandiri *Node JS, Express, Oracle,* dan *React-Redux*	19/09/2023	25/09/2023
3	**Membuat *API* untuk fitur *Calendar***		
3.1	Mempelajari *user flow* dan membuat *ERD* untuk fitur *Calendar*	26/09/2023	26/09/2023
3.2	Membuat *database* untuk fitur *Calendar* berdasarkan *ERD* yang telah dibuat	27/09/2023	27/09/2023
3.3	Membuat *API* untuk fitur *Calendar*	29/09/2023	29/09/2023

No	Kegiatan Magang	Mulai	Selesai
4	**Menerapkan *API* pada fitur *Calendar***		
4.1	Implementasi *API* dan menghubungkan pada fitur *Calendar*	02/10/2023	12/10/2023
4.2	*Testing* fitur dan menerapkan fungsi *React Redux* sebagai *notification action* pada fitur *Calendar*	13/10/2023	13/10/2023
5	**Menerapkan *API* pada fitur *Student Analytics***		
5.1	Membuat *ERD* dan juga *Database* untuk fitur *Student Analytics*	16/10/2023	20/10/2023
5.2	Membuat *API* dan menghubungkan pada fitur *Student Analytics*	23/10/2023	27/10/2023
5.3	*Meeting* membahas *update* pengerjaan fitur yang telah dilakukan	30/10/2023	30/10/2023
5.4	Implementasi *API*, *Testing* fitur dan konfigurasi fitur *Student Analytics*	31/10/2023	31/10/2023
6	**Menerapkan *API* pada fitur *Overview***		
6.1	Membuat *ERD* dan juga *Database* untuk fitur *Overview*	01/11/2023	03/11/2023
6.2	Membuat *API* dan menghubungkannya ke fitur *Overview*	06/11/2023	10/11/2023
7	**Mengembangkan fitur *Student Notes* dan *API*-nya**		
7.1	Membuat *user flow*, *ERD*, dan juga *Database* untuk fitur *Student Notes*	13/11/2023	17/11/2023
7.2	Melakukan *meeting* untuk membahas *update* pengerjaan tugas dan *feedback* terhadap fitur *Student Notes*	20/11/2023	20/11/2023
7.3	Membuat fitur *Student Notes* dan juga *API*-nya berdasarkan *feedback* hasil *meeting*	21/11/2023	30/11/2023
8	**Membuat dokumentasi pengerjaan *back-end website* bimbingan akademik**		
8.1	Mengerjakan dokumentasi untuk *project website* bimbingan akademik	01/12/2023	08/12/2023

3.2.1 Penjelasan *tools* yang akan digunakan pada pengembangan *back-end website* bimbingan akademik

Sebelum masuk ke pembahasan aktivitas kerja magang, terdapat beberapa *tools* ataupun bahasa pemrograman yang dipilih dalam membantu mengembangkan *back-end website* bimbingan akademik. Beberapa *tools* dipilih karena mudah digunakan, mudah diakses, sudah ditentukan oleh tim IT UMN, serta mahasiswa magang sudah pernah menggunakan *tools* atau bahasa pemrograman tersebut sehingga dapat dipahami dengan mudah secara lebih lanjut. Berikut ini adalah daftar *tools* atau bahasa pemrograman beserta penjelasannya dalam mendukung mahasiswa magang untuk mengembangkan *back-end* pada *website* bimbingan akademik:

1. *Visual Studio Code*

   Penggunaan *tools Visual Studio Code* bertujuan sebagai *code editor* untuk menulis *code* untuk *back-end server*, fitur, ataupun fungsi *API*. *Tools* yang dikembangkan *Microsoft* tersebut digunakan oleh mahasiswa magang karena selain mudah diakses dan digunakan, terdapat fitur berupa *extension* yang memudahkan untuk menulis *code* sesuai dengan bahasa pemrograman yang dipilih. Selain itu, mahasiswa magang juga sudah pernah menggunakan *Visual Studio Code* sehingga tidak kebingungan nantinya ketika melakukan pekerjaan magang.

2. *Javascript*

   Bahasa pemrograman yang digunakan dalam mengembangkan *back-end project website* bimbingan akademik secara keseluruhan adalah *Javascript*. Bahasa pemrograman tersebut juga sudah terkenal secara luas di kalangan *developer* karena mudah dipelajari dan banyak digunakan dalam mengembangkan aplikasi, salah satunya adalah aplikasi *website*. *Javascript* digunakan dalam *project website* bimbingan akademik karena pada periode pengembangan *front-end*, *framework* yang digunakan adalah *React* dengan basis yaitu *Javascript*. Selain itu,

Javascript bertujuan untuk mengembangkan *back-end* karena nantinya akan menggunakan *Node JS* dalam menjalankan *server back-end* dan *framework Express JS* dalam membuat *API* fitur-fitur pada *website* bimbingan akademik.

3. *Node JS*

   Tujuan utama digunakannya *Node JS* adalah untuk menjalankan *server front-end* dan juga *back-end* agar keduanya dapat diakses dan dihubungkan sehingga fungsional dalam melakukan pertukaran data dari *database*. *Node JS* juga digunakan dalam *project website* bimbingan akademik karena sudah digunakan dalam pengerjaan *front-end* di periode magang sebelumnya.

4. *Express JS*

   Framework Express JS merupakan *framework* yang digunakan dalam mengembangkan *REST API* pada *website* bimbingan akademik. Alasan digunakannya *framework* tersebut karena berdasarkan ketentuan oleh tim IT dan juga beberapa alasan lainnya seperti mudah diterapkan dan juga terhubung dengan *Node JS*. Dengan bantuan beberapa *module* lainnya yang juga digunakan bersamaan dengan *Express JS*, maka fungsi *API* dapat dihubungkan dari *back-end* ke bagian *front-end* dengan mudah.

5. *Oracle Database*

   Oracle Database sendiri digunakan sebagai *database* untuk tempat menyimpan data-data terkait fitur ataupun hasil dari *submit form user* seperti pada fitur *Calendar* nantinya. Penggunaan *Oracle Database* sudah ditentukan oleh tim IT sehingga mahasiswa magang hanya perlu mempelajari atau melakukan *review* penggunaan *SQL* dalam menulis *query* untuk membuat tabel data atau menampilkan data melalui *query* seperti *select*. Secara garis besar, *Oracle Database* yang akan digunakan untuk menampung seluruh data-data terkait dengan *user*.

6. *SQL Developer*

   Salah satu *tools* pendukung dalam membuat tabel data pada *Oracle Database* adalah *SQL Developer*. *Tools* ini digunakan juga dalam mengembangkan *query select*, *update*, *delete* atau yang lainnya untuk digunakan pada fungsi *API*. Alasan digunakannya *tools* tersebut adalah mudah digunakan dan mudah dihubungkan dengan *Oracle Database*.

7. *React*

   React menjadi *framework* atau *library* untuk pengembangan *front-end* di periode magang sebelumnya. Pada periode magang *back-end* ini, mahasiswa magang juga dianjurkan untuk memahami *React* agar dapat menerapkan *API* ke bagian *front-end* dengan mudah. *React* sendiri juga sudah sering digunakan di kalangan *developer website* karena terdapat banyak *library* atau *package* dapat digunakan dalam mengembangkan *front-end*. Selain itu, pada *React* terdapat penerapan komponen yang dapat digunakan secara ulang (*reusable component*) sehingga *developer* tidak perlu membuat *code* komponen yang sudah ada tersebut.

8. *React-Redux*

   Penggunaan *React-Redux* pada *website* bimbingan akademik adalah karena sudah digunakan pada *template* yang menjadi bagian *front-end*. Dengan adanya penerapan *React-Redux*, *developer* dapat menyimpan *state* pada *React* dan mengelola *state* tersebut, misalnya adalah *notification* terhadap *action* yang dilakukan *user*. Penerapan *React-Redux* ini sangat berguna untuk integrasi antara *front-end* dan *back-end* sehingga jika *user* melakukan suatu *action* seperti *form submit*, maka nantinya akan muncul sebuah *textbox* berupa *notification* pada bagian halaman *website* sebagai deskripsi *action* tersebut.

9. *Lucidchart*

Lucidchart merupakan sebuah *tools* yang berguna untuk membantu *developer* dalam membuat diagram. Diagram yang dibuat pada periode magang *back-end* ini yaitu *ERD* (*Entity Relationship Diagram*) dan berguna sebagai gambaran struktur *database* yang akan dibuat. Selain *ERD*, *tools* ini dapat membuat bermacam-macam diagram lainnya sesuai dengan kebutuhan *developer*. Untuk versi dari *Lucidchart* yang digunakan dalam pembuatan *ERD* menggunakan versi tidak berbayar dimana versi gratis tersebut sudah cukup dalam membantu pembuatan *ERD* untuk *database* fitur *website* bimbingan akademik. Salah satu alternatif dari *Lucidchart* sendiri adalah *draw.io*, tetapi berdasarkan pengalaman yang dirasakan ketika menggunakan kedua *tools* tersebut, *Lucidchart* menjadi pilihan terbaik karena terdapat banyak *template* atau komponen pendukung dalam membuat diagram.

10. *GitHub*

    Penggunaan *GitHub* dalam pengerjaan *API* atau *back-end website* bimbingan akademik adalah untuk membantu kolaborasi antar mahasiswa magang. Kolaborasi tersebut bertujuan dalam membantu satu sama lain jika terdapat *error* pada fitur yang dikerjakan, serta menggabungkan *code* yang sudah dikerjakan oleh masing-masing mahasiswa magang. Selain itu, *GitHub* yang sudah disediakan oleh tim IT tersebut dapat dijadikan sebagai *backup folder* karena jika terdapat banyak *error* yang membuat *website* tidak berjalan dengan semestinya, maka mahasiswa magang dapat menggunakan versi *project* sebelumnya dan mencari letak *error* atau perubahan *code* yang menyebabkan *error* tersebut.

3.2.2 Pengenalan *project website* bimbingan akademik (Minggu 1 – 2)

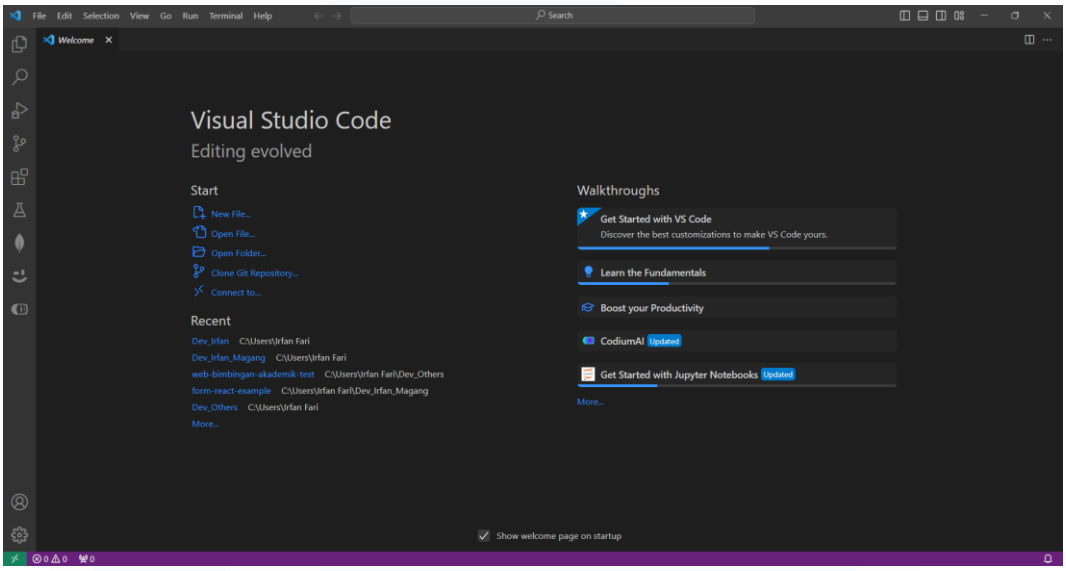
Aktivitas pertama ini merupakan aktivitas yang dilakukan pada minggu pertama melakukan kerja magang. Hari pertama melakukan kerja magang ini

dimulai pada tanggal 6 September 2023. Pada hari pertama, mahasiswa melakukan koordinasi dengan *supervisor* atau *product owner* untuk mendapatkan *folder project* bimbingan akademik yang berisikan *file application website*, *user-flow* dan tampilan *website* yang sudah dikerjakan pada tahapan pengembangan *front-end* di periode magang sebelumnya. Kemudian setelah mendapatkan *folder project* tersebut, mahasiswa melakukan koordinasi dengan rekan magang yang sudah masuk magang lebih awal pada bulan Agustus untuk menanyakan seputar *job description* atau pekerjaan yang akan dilakukan. Sebagian besar pekerjaan yang akan dilakukan adalah mengembangkan *back-end* untuk *website* bimbingan akademik dengan cara membuat *API* untuk masing-masing fitur yang ada pada *website* tersebut.

Kemudian, pada tanggal 11 September 2023 sampai dengan 12 September 2023 nantinya, mahasiswa diwajibkan mengurus administrasi kontrak kerja magang yang diberikan oleh *HRD* UMN melalui *email*. Kontrak kerja magang tersebut berisikan syarat dan ketentuan, durasi magang, serta deskripsi pekerjaan mahasiswa. Mahasiswa magang juga diminta untuk konfirmasi melalui *WhatsApp* dan mengirim KTP serta KTM sebagai dokumen pelengkap kontrak kerja magang.

Setelah mendapatkan *folder project* bimbingan akademik, mahasiswa mulai melakukan *install tools* terkait untuk dapat mengembangkan *back-end* dan *API* pada *website* bimbingan akademik. Beberapa *tools* dan *framework* pengembangan *website* yang digunakan masih sama seperti pada pengembangan *front-end*. *Tools* tersebut berupa *Visual Studio Code*, *Node JS*, dan juga *framework React*. Kemudian ada beberapa *tools* tambahan yang akan digunakan dalam membantu pengembangan *back-end* dan *API*, *tools* tersebut berupa *database Oracle*, serta *SQL Developer* untuk menjalankan *query database*. Kemudian, setelah melakukan instalasi *tools*, mahasiswa juga dianjurkan untuk memahami proses bisnis dari *website* bimbingan akademik.

3.2.2.1 *Visual Studio Code*

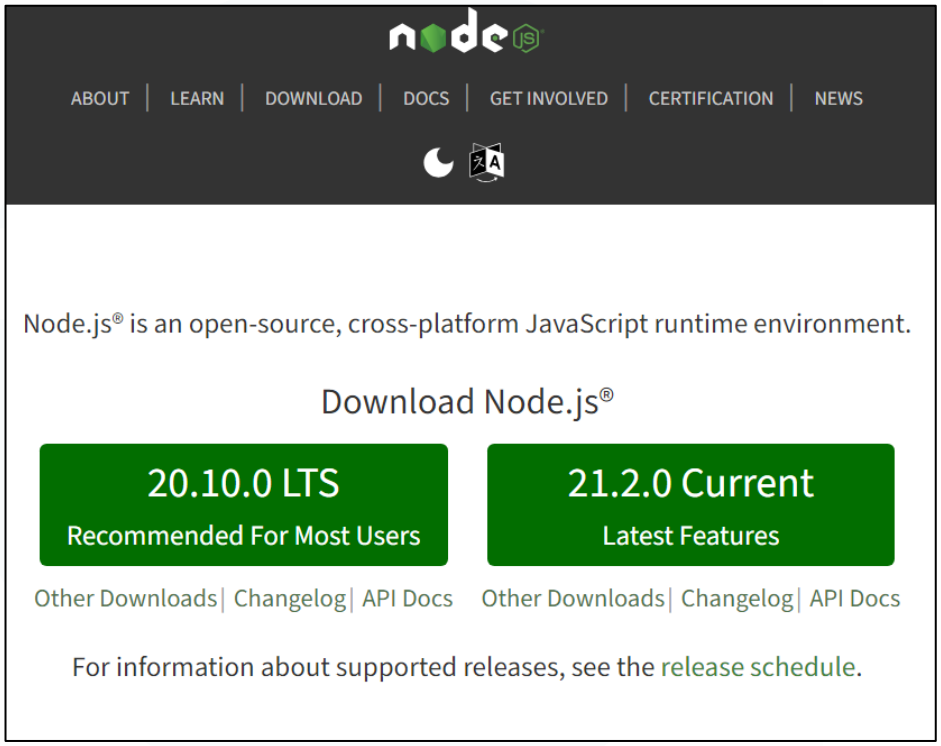


Gambar 3. 1 Tampilan *Visual Studio Code*

Gambar 3.1 merupakan tampilan dari aplikasi *Visual Studio Code*. Aplikasi tersebut merupakan *code editor* yang dikembangkan oleh *Microsoft* dan digunakan untuk menulis kode pemrograman, serta membantu *developer* aplikasi karena terdapat banyak *plugin* atau *extension* [12]. Aplikasi *Visual Studio Code* dapat dilihat dan diunduh pada halaman *website*-nya, yaitu https://code.visualstudio.com/.

Dalam pengembangan *back-end* nantinya, *Visual Studio Code* digunakan untuk menulis kode fungsi *API* dan *Database*, serta digunakan untuk menjalankan *script* agar *front-end* dan *back-end* dapat dijalankan secara bersamaan karena adanya *integrated terminal* yang memungkinkan untuk mengakses *console* atau *terminal* secara langsung di aplikasi *Visual Studio Code*.

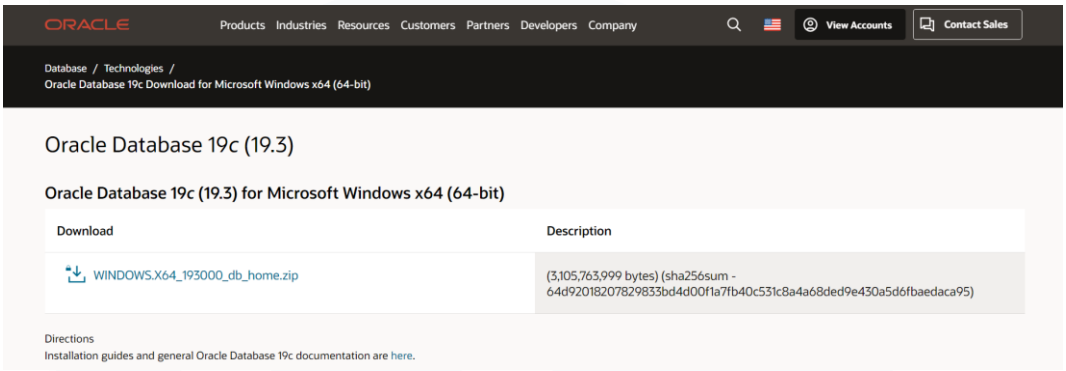
3.2.2.2 *Node JS*



Gambar 3. 2 Halaman *Website Node JS*

Gambar 3.2 adalah tampilan dari halaman *website Node JS* yang digunakan untuk menguduh *JavaScript runtime environment*. Tujuan dilakukan *install Node JS* adalah untuk dapat menjalankan *script front-end* dan juga *back-end*. Halaman tersebut dapat diakses di https://nodejs.org/en/, dimana paket instalasi *Node JS* sudah termasuk *node package manager* [13]. Dengan adanya *node package manager* atau *npm*, *console* pada aplikasi *Visual Studio Code* dapat menjalankan *server*, sehingga tampilan *front-end* dan *API* pada *back-end* dapat diakses di *localhost*. Selain itu, pada *npm* yang disediakan oleh *Node JS*, terdapat *package* atau *library* bernama *Express* yang akan digunakan dalam pengembangan *API* pada masing-masing fitur *website* bimbingan akademik .Untuk dapat melihat tampilan *front-end*, mahasiswa magang dapat menulis perintah *npm start* di *console* yang telah terhubung ke *folder project* pada *Visual Studio Code*.

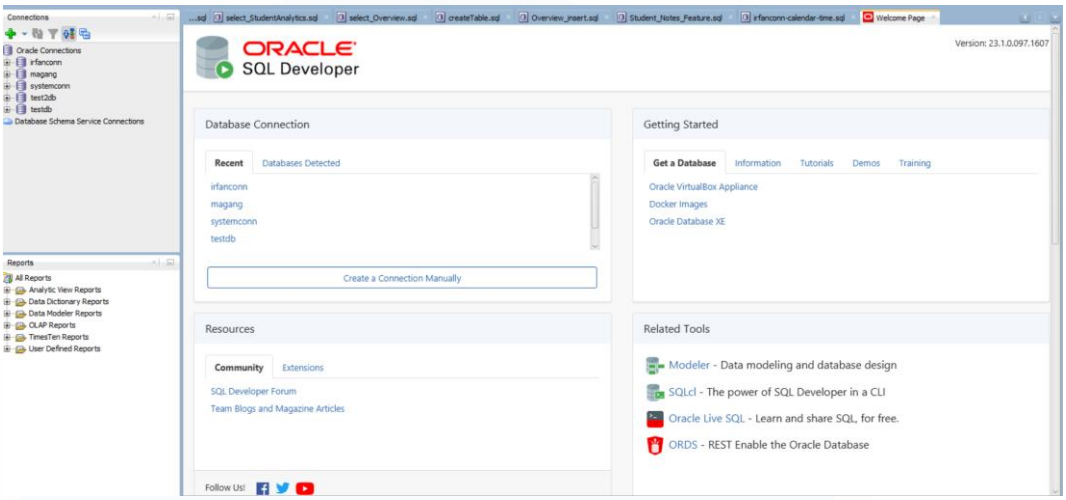
3.2.2.3 *Oracle Database*



Gambar 3. 3 Tampilan *Website Oracle Database*

Gambar 3.3 merupakan tampilan dari halaman *website* resmi *Oracle* yang digunakan untuk mengunduh *database Oracle* dan dapat diakses pada https://www.oracle.com/database/technologies/. *Database* tersebut berbentuk *SQL* (*Structured Query Language*) dan dapat diakses melalui *console* pada *operating system*. Penggunaan *Oracle* sebagai *database* adalah berdasarkan ketentuan dari tim IT, sehingga mahasiswa diwajibkan mengunduh *Oracle database* agar dapat menampung data-data yang dikirim pada fitur-fitur *website* dan diterima di dalam *database* dengan bantuan *API*.

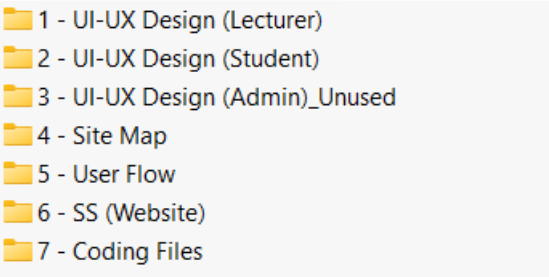
3.2.2.4 *SQL Developer*



Gambar 3. 4 Tampilan Aplikasi *SQL Developer*

Gambar 3.4 adalah tampilan dari aplikasi *SQL Developer* yang digunakan untuk menjalankan beberapa perintah *SQL* seperti *Create Table, Insert*, *Delete*, *Update*, dan yang lainnya. Proses mengunduh hampir sama seperti *database Oracle* karena masih dalam satu *website Oracle*. Kemudian, penggunaan *tools* ini adalah untuk mempermudah mahasiswa magang dalam membuat *table* di dalam *database* dan digunakan untuk menghubungkan fungsi-fungsi di masing-masing fitur *website* bimbingan akademik agar data dapat diterima dan dikirim dengan mudah melalui bantuan *API*. Selain itu, *SQL Developer* digunakan dalam membuat *Select, Insert*, hingga *Update* agar dapat digunakan sebagai *model query* pada *back-end* dan dapat dipakai pada pengembangan *API*.

3.2.2.5 Memahami Proses Bisnis Dan *Business Requirements*



Gambar 3. 5 *Folder Project Website* Bimbingan Akademik

Gambar 3.5 adalah beberapa *folder* dan juga *file* yang akan digunakan untuk pengembangan *website* bimbingan akademik. *Folder* dengan nama "*7 – Coding Files*" merupakan seluruh kode utama atau *source code website* bimbingan akademik yang telah dibuat dalam tahap *front-end* dan akan digunakan juga dalam mengembangkan *back-end*. Beberapa *folder* seperti *site map* dan juga *user flow* akan menjadi dokumen referensi dalam mengembangkan *back-end*, serta sebagai proses bisnis dan *business requirements*.

Dari isi *folder project* tersebut, mahasiswa dianjurkan untuk memahami keseluruhan alur bisnis ataupun keseluruhan fitur dan kegunaannya pada *website* bimbingan akademik berdasarkan dokumen-dokumen referensi yang telah disediakan. Maka, pada aktivitas magang selanjutnya dimana mahasiswa akan mengembangkan *API* di masing-masing fitur, dokumen *project* tersebut digunakan sebagai panduan implementasi *back-end* agar mahasiswa magang dapat menghemat waktu dan tidak membuat ulang alur bisnis ataupun yang lainnya.

3.2.3 Pembelajaran *tools* pada *project website* bimbingan akademik (Minggu 3 – 4)

Aktivitas magang selanjutnya adalah melakukan pembelajaran mandiri dalam menggunakan *tools* atau bahasa pemrograman yang akan digunakan dalam mengembangkan *back-end website* bimbingan akademik. Namun,

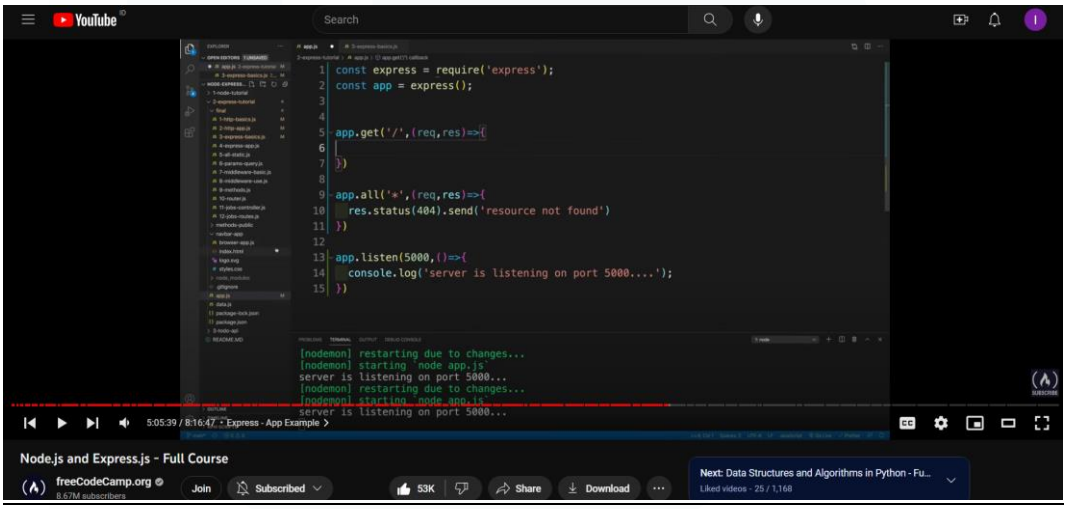
sebelum melakukan pembelajaran mandiri, mahasiswa mengikuti *meeting* yang dilaksanakan pada tanggal 18 September 2023 di ruang *laboratorium Big Data* (C503), Gedung C, UMN. *Meeting* tersebut dimulai pada sekitar jam 3 sore dan dihadiri oleh *product owner*, seperti Ibu Wella, Ibu Ririn, dan juga Ibu Fonita. Lalu, dihadiri juga oleh Bapak Danry dari perwakilan tim IT.

Tujuan dari *meeting* ini adalah untuk menentukan pembagian tugas yang akan dilakukan oleh mahasiswa magang, *update* pengerjaan tugas setiap harinya melalui *email*, dan terkait teknis. Untuk pembagian tugasnya sendiri, mahasiswa magang secara inisiatif memilih fitur yang akan dikerjakan. Fitur yang pertama kali dikerjakan adalah fitur *Calendar* yang berfungsi untuk menambah jadwal bimbingan oleh dosen. Kemudian, mahasiswa magang juga diperintahkan untuk selalu *update* mengenai pengerjaan fitur, kendala, dan yang lainnya melalui *email* sehingga *product owner* dan juga tim IT dapat memantau *progress* yang dilakukan oleh mahasiswa. Selain itu, dari tim IT sendiri menganjurkan mahasiswa magang untuk menggunakan konsep yang mirip seperti MVC (*Model, View, Controller*), dengan tambahan konfigurasi *database*, serta *routes* dalam mengembangkan *back-end website*. Nantinya, *model* akan digunakan sebagai penampung *query*, *view* menjadi *routes* sebagai *url API* agar dapat dihubungkan ke *front-end*, kemudian *controller* untuk menampung kode dari fungsi *API*, serta *database configuration* untuk menampung *user* dan *password* agar *database* dapat terhubung dengan *API*.

Setelah melakukan *meeting* tersebut, mahasiswa diberikan waktu kurang lebih 1 minggu untuk dapat mempelajari *tools* atau bahasa pemrograman terkait secara mandiri. Beberapa pembelajaran yang dilakukan tersebut adalah untuk memahami *Express JS* sebagai *framework* yang digunakan pada *API*, *Oracle* dan *SQL Developer* sebagai *database*, dan juga *React Redux* yang berguna untuk menyimpan *state* pada beberapa fitur, salah satunya adalah *Calendar*. Kemudian, beberapa bahasa pemrograman seperti *JavaScript, framework React*, dan juga *Node JS* tidak perlu mempelajari dari awal karena pada pengalaman magang sebelumnya, mahasiswa sudah mempelajarinya, dan

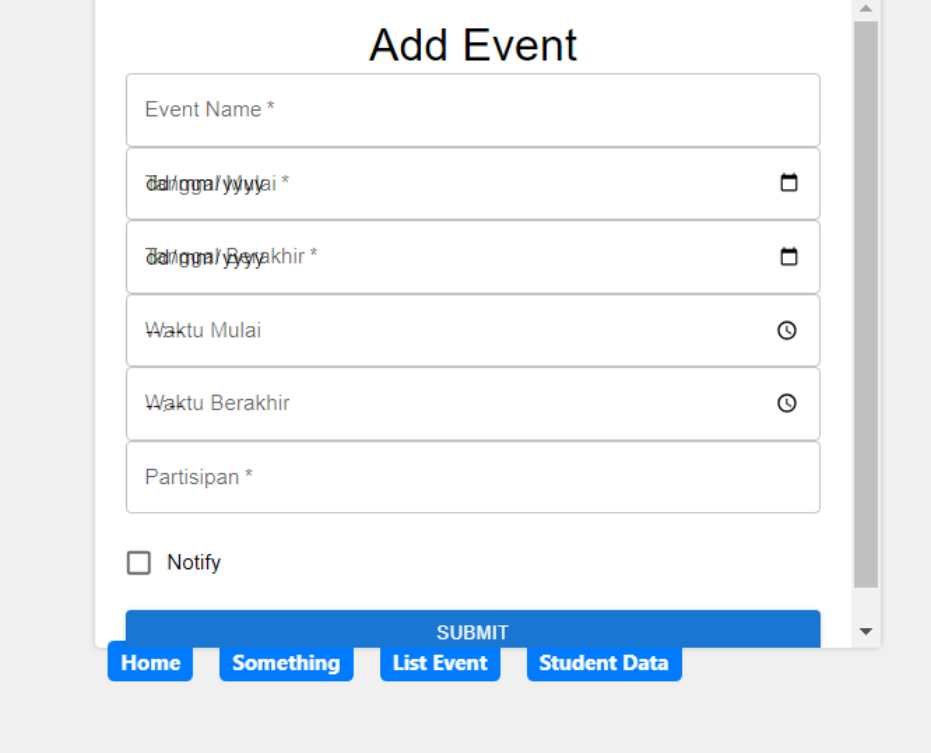
sekaligus hal ini dapat menghemat waktu dalam mempelajari *tools-tools* tersebut. Pembelajaran ini dilakukan melalui video pembelajaran yang sudah tersedia sangat banyak di *platform YouTube*, serta mahasiswa diharapkan mempelajarinya dengan cermat agar tidak ada konsep penting yang terlewati.

3.2.3.1 Mempelajari *Express* Pada *Node JS*



Gambar 3. 6 Video Pembelajaran *Express* & *Node JS*

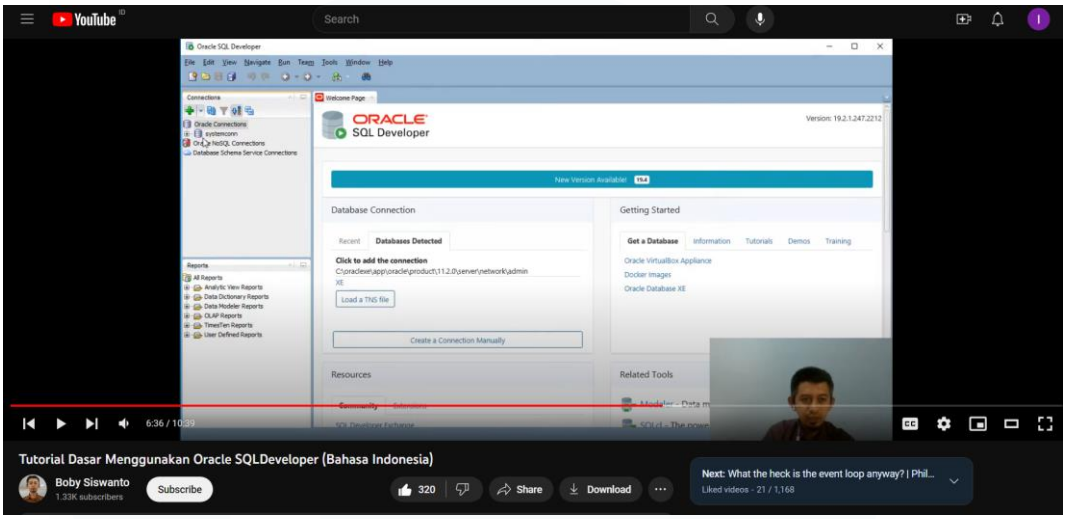
Gambar 3.6 adalah video pembelajaran dari *framework Express* yang juga merupakan bagian dari *module* atau *library Node JS* dalam mengembangkan *API*. Video yang berdurasi sekitar 8 jam tersebut dapat dilihat di https://www.youtube.com/watch?v=Oe421EPjeBE, kemudian sudah mencakup pembahasan mengenai konsep *REST API*, *HTTP*, hingga konsep penting pada *framework Express*. Agar dapat lebih memahami dan tidak cepat lupa mengenai beberapa konsep penting *Express*, mahasiswa magang membuat implementasi dari *API* dengan cara membuat *dummy project* menggunakan *React*, *Node JS*, dan tentunya *framework Express* yang hanya berisi tampilan *form*. Tujuan dari membuat *dummy project* tersebut juga adalah untuk memahami cara kerja *form* terutama untuk pengerjaan *API* pada fitur *Calendar* nantinya.



Gambar 3. 7 Tampilan Sederhana *Dummy Project Form*

Gambar 3.7 merupakan tampilan sederhana untuk implementasi *submit form* seperti pada fitur *Calendar* nantinya. Tampilan tersebut terlihat tidak begitu bagus dikarenakan fokus pembelajaran lebih ke arah untuk memahami fungsionalitas dari *form* tersebut. Selain itu, *dummy project* tersebut digunakan untuk pembelajaran terakhir mengenai penggunaan *React Redux* sebagai *state management* agar *data flow* dapat dikelola dengan baik di bagian *front-end*. Setelah memahami bagaimana cara *form* dapat mengirim data melalui *API* ke *database*, maka pembelajaran selanjutnya adalah cara menggunakan *SQL Developer* dan menghubungkannya dengan *database Oracle*.

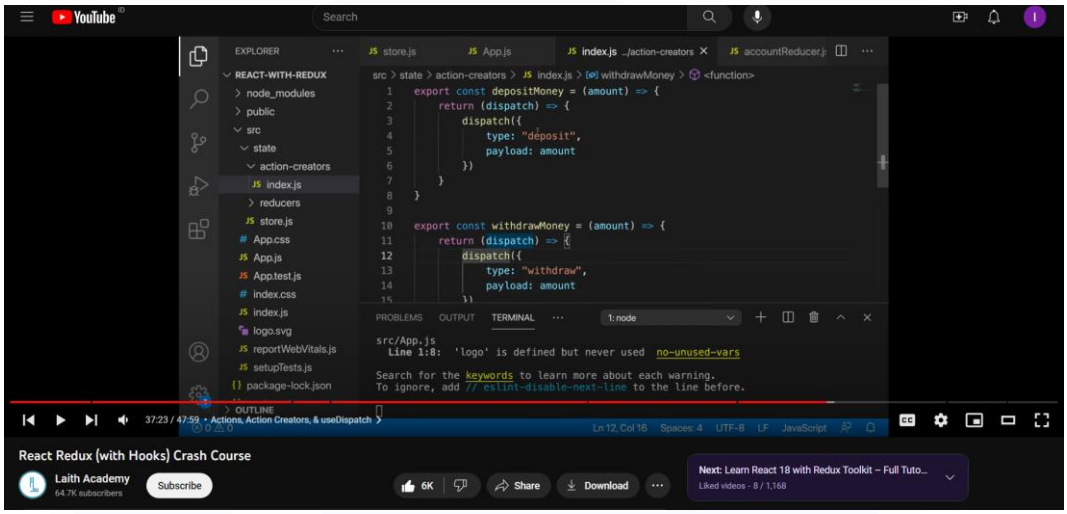
3.2.3.2 Mempelajari *SQL Developer* & *Oracle Database*



Gambar 3. 8 Video Pembelajaran *Tools SQL Developer*

Gambar 3.8 merupakan tampilan dari video pembelajaran yang bertujuan untuk memahami penggunaan *SQL Developer* dan menghubungkannya dengan *Oracle Database*. Video pembelajaran yang dapat diakses pada https://www.youtube.com/watch?v=hRwCEBRP1kI juga menjadi panduan mahasiswa magang agar dapat memberikan akses kepada *user* tertentu pada *database* karena pada beberapa kondisi, jika *user* tersebut tidak memiliki *privileges admin*, maka tidak dapat menjalankan perintah seperti *Select, Insert, Update, Delete* dan perintah lainnya. Oleh karena itu, video pembelajaran yang singkat tersebut sudah sangat detil dan jelas untuk memberitahu mahasiswa magang cara penggunaan *SQL Developer*. Selain itu, terdapat sumber pembelajaran dari internet lainnya seperti untuk membantu pemahaman *SQL* yang terdapat pada *Oracle database*.

3.2.3.3 Mempelajari *React Redux*



Gambar 3. 9 Video Pembelajaran *React Redux*

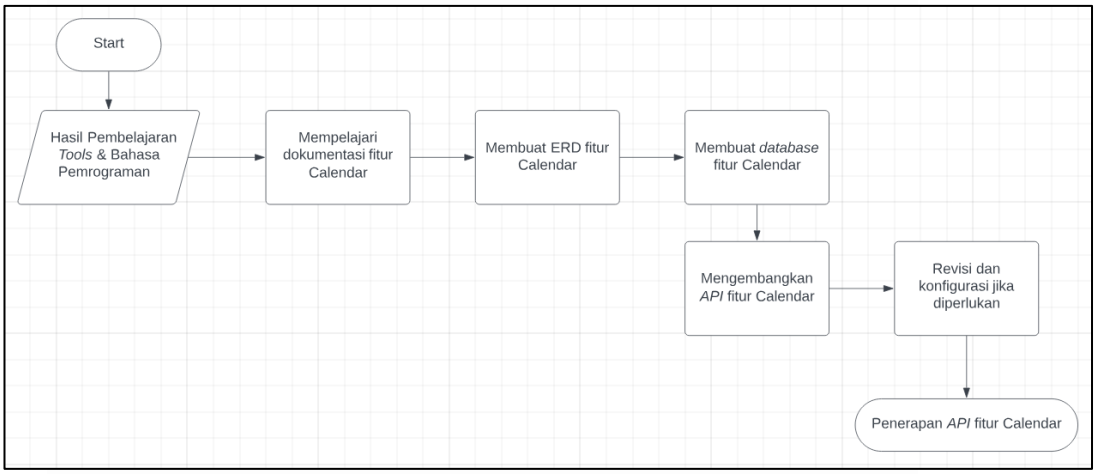
Gambar 3.9 adalah video pembelajaran penting dan singkat (*crash course*) untuk memahami konsep-konsep penting dari *React Redux* dan dapat dilihat pada https://www.youtube.com/watch?v=bbkBuqC1rU4. Konsep-konsep penting seperti *reducer*, *store*, *action*, *payload* dan yang lainnya sudah tercakup pada video pembelajaran tersebut. Dikarenakan penggunaan *React Redux* pada *project website* bimbingan akademik hanya digunakan pada beberapa fitur, jadi setidaknya mahasiswa diharapkan mengerti cara pengelolaan *state* pada *React Redux*, serta cara menerapkan *action* dan *dispatch action* untuk menerapkan fungsi *notification action* pada beberapa fitur nantinya. *Notification action* tersebut merupakan bagian penting pada *user experience* karena menunjukkan *action* yang dilakukan oleh *user*, seperti menekan tombol *submit form*, hingga *redirect user* ke halaman lainnya.

Setelah mempelajari *tools* dan bahasa pemrograman yang diperlukan, maka aktivitas magang yang dilakukan mahasiswa selanjutnya adalah untuk mengembangkan *back-end website*. Untuk dapat mengembangkan *back-end* secara menyeluruh pada *website* bimbingan akademik, mahasiswa

memulainya secara mendasar, yaitu dimulai dengan pengembangan *API* untuk fitur *Calendar* karena sebagai fitur yang pertama kali dikerjakan.

3.2.4 Membuat *API* untuk fitur *Calendar* (Minggu 4)

Tugas pengerjaan fitur dilakukan setelah mahasiswa magang mempelajari semua *tools* dan bahasa pemrograman yang diperlukan dalam mengembangkan *back-end* dan juga *API* pada masing-masing fitur. Fitur pertama yang akan diterapkan fungsionalitas *back-end* adalah fitur *Calendar*. Namun, proses pengerjaan fitur *Calendar* tidak langsung dikerjakan sekaligus dalam tahapan menambahkan fungsionalitas *API* di bagian *front-end* karena fitur *Calendar* sangat kompleks dari adanya *React Redux* yang digunakan dalam mengelola *state* di fitur tersebut. Oleh karena itu, tahapan dimulai dari mempelajari atau *review* alur bisnis fitur *Calendar* dan juga *user-flow* pada fitur tersebut. Kemudian, setelah mempelajari *user flow*, mahasiswa magang membuat *ERD* fitur *Calendar* sebagai pondasi atau panduan struktur *database* yang akan digunakan sebagai tempat menampung data setelah *user* melakukan *submit form* di fitur *Calendar*. Dengan adanya struktur *database* yang jelas, mahasiswa dapat mengembangkan *API* untuk fitur tersebut.



Gambar 3. 10 Proses Pengembangan *API* Fitur *Calendar*

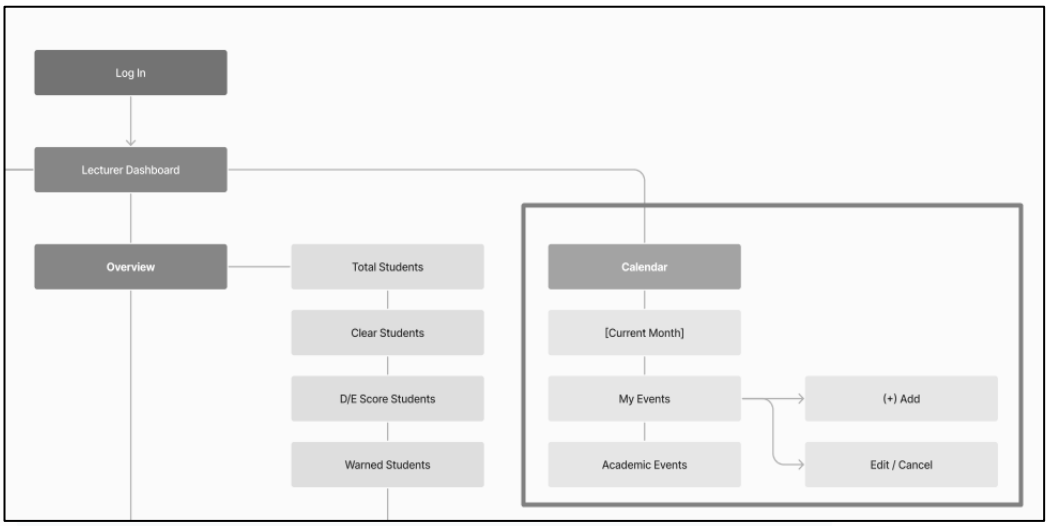
Gambar 3.10 adalah gambaran dari pengembangan *API* untuk fitur *Calendar* secara garis besar. Tahapan dimulai dari hasil pembelajaran yang

dijadikan sebagai acuan untuk dapat mengerjakan pekerjaan magang secara teknis, kemudian mahasiswa magang mempelajari dokumentasi seperti dokumen *user flow* fitur *Calendar* dan membuat *ERD* untuk struktur tabel *database*. Selanjutnya, mahasiswa magang membuat tabel *database* sebagai tempat menampung data dan digunakan di fungsi *API* yang akan dikembangkan pada proses berikutnya. Selain itu, mahasiswa magang juga melakukan konfigurasi kecil agar *API* berhasil diakses pada fitur *Calendar*. Proses tersebut diakhiri dengan berlanjut pada penerapan *API* fitur *Calendar* tersebut di aktivitas selanjutnya.

3.2.4.1 Mempelajari *user flow* dan membuat *ERD* untuk fitur *Calendar*

Pada tahapan ini, mahasiswa magang mempelajari kembali proses bisnis atau *business requirements website* bimbingan akademik. Proses tersebut dimulai dengan melihat dokumen internal *project* bimbingan akademik yang sudah disediakan sebelumnya dalam *folder project*. Dokumen internal tersebut dibuat pada saat tahapan pengembangan *front-end* atau periode magang sebelumnya. Dokumen ini juga akan menjadi panduan mahasiswa untuk dapat melihat cara kerja fitur ataupun *user flow* di fitur *Calendar*. Lalu, mahasiswa magang juga membuat *ERD* berdasarkan kebutuhan data-data yang diperlukan ketika *submit form* pada fitur *Calendar*. Berikut ini adalah proses-proses yang dilakukan:

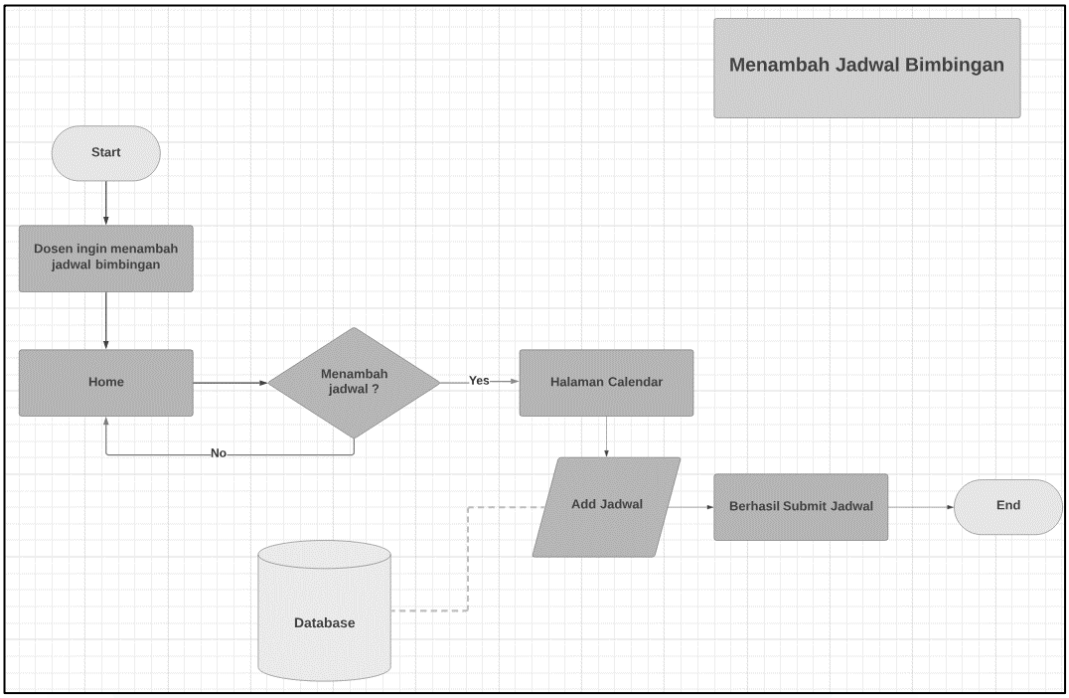
1. **Mempelajari *sitemap* dan *user flow***



Gambar 3. 11 *Sitemap* Fitur *Calendar*

Gambar 3.11 adalah dokumen *sitemap* yang menampilkan beberapa fitur termasuk fitur *Calendar*. Dokumen ini berasal dari dokumen internal *project* bimbingan akademik [14]. Definisi dari *sitemap* sendiri adalah suatu dokumen yang digunakan untuk menyampaikan informasi mengenai halaman *website* dan hubungannya dengan keseluruhan *website*, serta menampilkan isi pada halaman atau fitur utama dari sebuah *website* [15]. *Sitemap* ini juga dijadikan panduan untuk dapat memahami apa saja fitur dan fungsionalitas yang terdapat pada fitur *Calendar*.

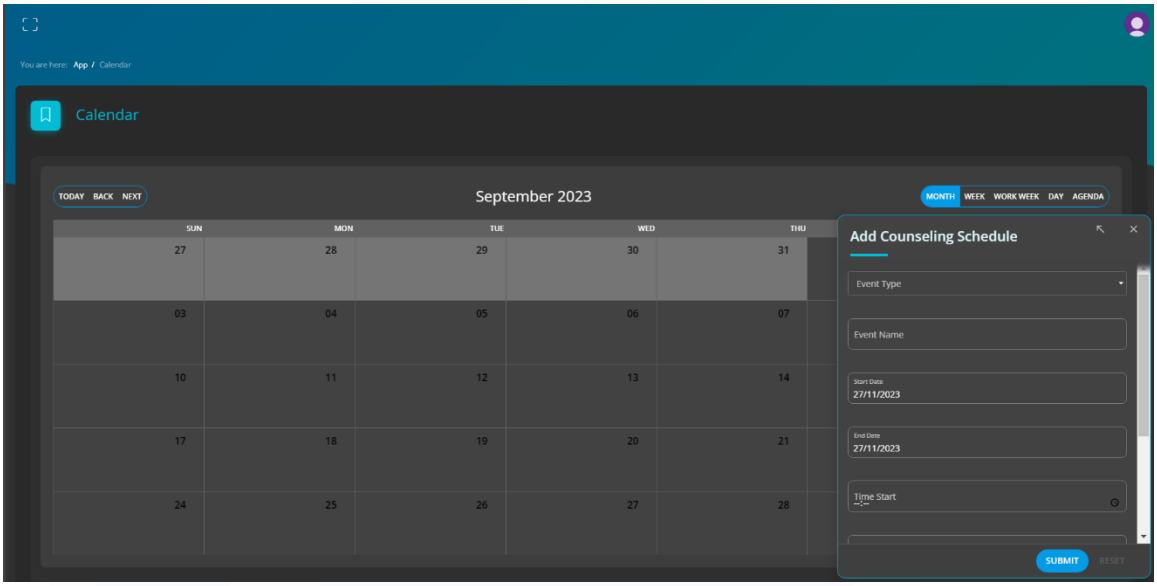
Pada *sitemap* fitur *Calendar*, terdapat beberapa fungsi atau fitur yang penting, seperti terdapat tampilan *Calendar* atau jadwal berdasarkan bulan (*current month*), kemudian terdapat bagian *event form* agar *user* dapat menambah jadwal bimbingan atau membatalkan *form* yang terdiri dari *Add Event* dan juga *Cancel*. Dari *sitemap* ini, mahasiswa magang mengambil kesimpulan bahwa terdapat fungsionalitas penting, seperti menampilkan data-data *event* dari *database* sehingga muncul di bagian jadwal, fungsionalitas *form* untuk menambahkan data *event* ke dalam *database*.



Gambar 3. 12 Dokumen *User Flow* Fitur *Calendar*

Gambar 3.12 merupakan *user flow* yang sudah pernah dibuat tim magang sebelumnya dan digunakan untuk menjelaskan tahapan-tahapan dalam menambahkan *event* atau jadwal bimbingan [14]. Kemudian, dokumen tersebut dibuat ulang untuk menyesuaikan dengan pengembangan terkini dan tahapannya dimulai dari *user* ingin menambahkan jadwal bimbingan. Kemudian *user* atau dosen mengunjungi menu utama, jika ingin menambahkan jadwal baru, maka melanjutkan ke fitur *Calendar*, jika tidak ingin menambah jadwal maka proses tidak berlanjut. Jika sudah ada di fitur *Calendar*, maka dosen membuka *add event form* dan mengisi *input* pada *form* tersebut dan menekan tombol *submit*. Proses tersebut diakhiri ketika data sudah di-*submit* dan masuk ke dalam *database*.

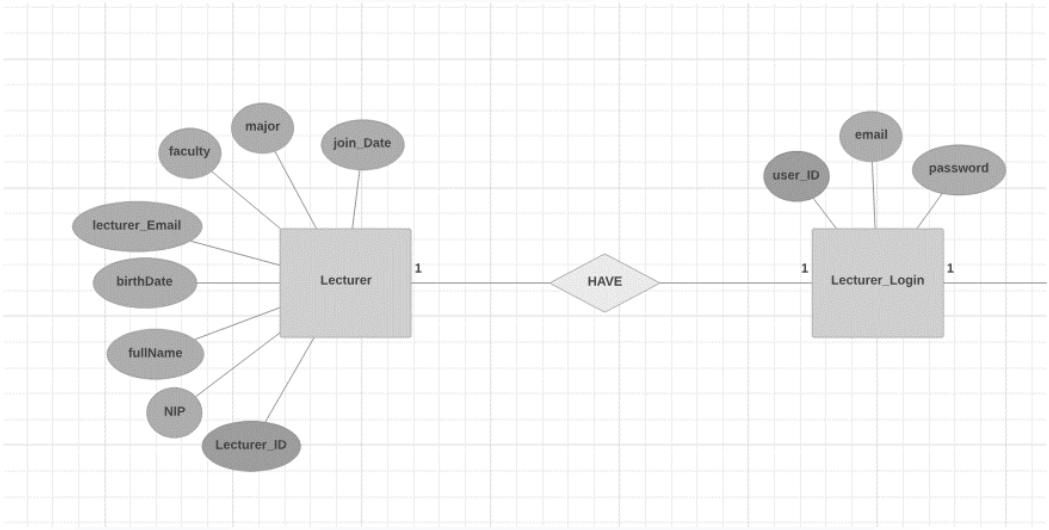
2. Mempelajari tampilan fitur *Calendar*



Gambar 3. 13 Tampilan Fitur *Calendar*

Gambar 3.13 adalah tampilan dari halaman atau fitur *Calendar* di *website* bimbingan akademik. Pada tampilan fitur *Calendar*, terdapat daftar jadwal yang dapat dilihat berdasarkan bulan, minggu, dan juga hari. Daftar jadwal yang ditampilkan sesuai tanggal tersebut akan dapat dihapus dan terhubung ke dalam *database*. Kemudian, pada fitur tersebut juga terdapat *form* dengan *label* "*Add Counseling Schedule*" dan akan berfungsi sebagai tempat mengirim data jadwal bimbingan ke dalam *database*. Setelah melihat tampilan fitur *Calendar*, mahasiswa magang melanjutkan ke tahapan pembuatan *ERD* fitur tersebut.

3. Membuat *ERD* fitur *Calendar*

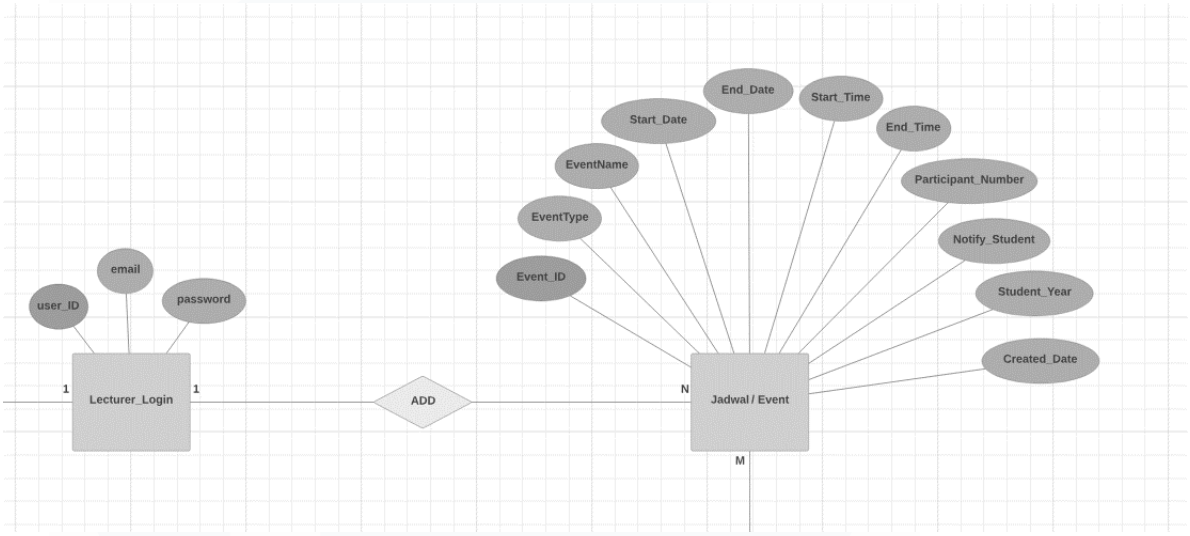


Gambar 3. 14 *ERD* Fitur *Calendar* Bagian *User*

Gambar 3.14 adalah tampilan dari *ERD* atau diagram yang menunjukkan hubungan objek untuk digunakan dalam membuat struktur tabel *database* fitur *Calendar*. *ERD* ini dibuat setelah mempelajari dokumen *user flow* dan tampilan *website* karena kedua dokumen tersebut sudah dibuat dan dijadikan sebagai acuan sistem kerja fitur *Calendar*. Kemudian, pada *ERD* fitur *Calendar* bagian *User* ini hanya sebagai pengandaian jika fungsi *authorization user* telah diimplementasikan, serta pada kegiatan magang yang dilakukan periode sekarang hanya fokus kepada fitur *website* bimbingan akademik.

Pada bagian *ERD user*, terdapat objek *Lecturer* dengan *primary key* yaitu *Lecturer_ID* merupakan struktur tabel dosen di *database*. Untuk objek *Lecturer_Login* sendiri yang diidentifikasikan *user_id* sebagai *primary key* adalah struktur tabel *database* dimana terdapat info *login* dosen yang akan digunakan sebagai identifikasi bahwa *user* tersebut adalah dosen. Identifikasi tersebut merupakan bagian dari proses *authorization* agar dosen dapat mengakses fitur-fitur *website* ketika sudah melakukan *login*. Hubungan yang ada pada objek dosen dan *lecture_login* adalah *one-to-one* dimana setiap satu dosen hanya

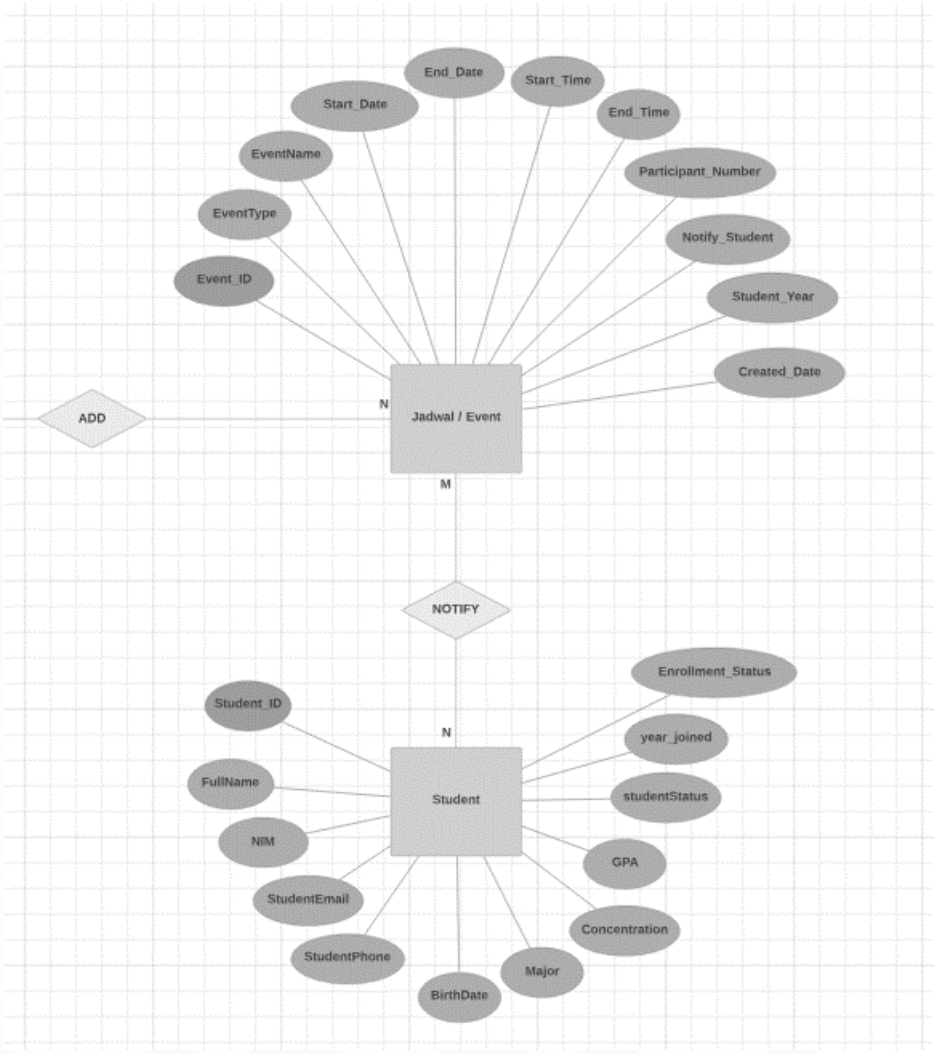
memiliki satu *user login*, yaitu *email* dan *password* untuk dapat mengakses *website* bimbingan akademik secara fungsional.



Gambar 3. 15 *ERD* Fitur *Calendar* Bagian *Event*

Gambar 3.15 adalah *ERD* utama pada fitur *Calendar* karena terdapat objek Jadwal atau *Event* yang digunakan sebagai struktur tabel *Calendar* di *database*. Tabel pada *database* tersebut dijadikan sebagai tempat untuk menampung data-data jadwal yang di-*input* dan di-*submit* oleh dosen ketika ingin menambah jadwal bimbingan. Hubungan antara *Lecturer_Login* dan *Event* adalah *one-to-many* dimana satu dosen dapat membuat banyak jadwal bimbingan.

Untuk objek *Event* sendiri mempunyai *primary key* bernama *Event_ID* dan beberapa atribut seperti *EventType*, *EventName*, *Start_Date*, *End_Date*, hingga yang lainnya ditambahkan berdasarkan *input form* di komponen *Add Counseling Schedule* fitur *Calendar*. Masing-masing nama atribut akan digunakan dalam pengembangan *API* karena berfungsi sebagai *identifier* pada *database* dan juga digunakan pada *request body* agar dapat menambah ataupun menampilkan data di bagian *front-end*.



Gambar 3. 16 *ERD* Fitur *Calendar* Bagian *Student*

Gambar 3.16 adalah *ERD* untuk fitur *Calendar* yang menampilkan hubungan *many-to-many* antara objek *Event* dan *Student*. Hubungan tersebut menunjukkan bahwa ketika dosen membuat jadwal maka jadwal-jadwal yang ditambah dosen akan dapat diberitahukan kepada banyak mahasiswa atau mahasiswa bimbingannya. Pada struktur tabel *Student* terdapat *Student_ID* sebagai *primary key* dan diikuti oleh atribut-atribut terkait mahasiswa. Namun, untuk dapat menghubungkan ke beberapa fitur ataupun fungsi-fungsi terkait *API* nantinya, yang digunakan sebagai *identifier* adalah *NIM* mahasiswa.

3.2.4.2 Membuat *database* untuk fitur *Calendar* berdasarkan *ERD* yang telah dibuat

Tahapan ini merupakan tahapan selanjutnya setelah mahasiswa magang mempelajari dokumen-dokumen terkait *website* bimbingan akademik dan membuat *ERD*. Pada tahapan ini, mahasiswa magang membuat tabel *database* di *SQL Developer* yang terintegrasi dengan *database Oracle*. Tabel *database* untuk fitur *Calendar* ini digunakan untuk menampung data-data hasil dari *submit* dosen ketika menambah jadwal di fitur tersebut. Cara data-data tersebut dapat disimpan ke dalam *database* adalah menggunakan bantuan *API*. Selain itu, data jadwal bimbingan yang sudah disimpan di *database* akan ditampilkan kembali di bagian *front-end* atau tampilan *website*.

```
CREATE TABLE CalendarTime(
    id NUMBER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),
    eventType VARCHAR(155),
    eventName VARCHAR(255),
    start_date DATE,
    end_date DATE,
    start_time VARCHAR(5),
    end_time VARCHAR(5),
    participant INT NOT NULL,
    notify VARCHAR(5) DEFAULT 'true',
    studentYear VARCHAR(5) DEFAULT 'All',
    createdDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

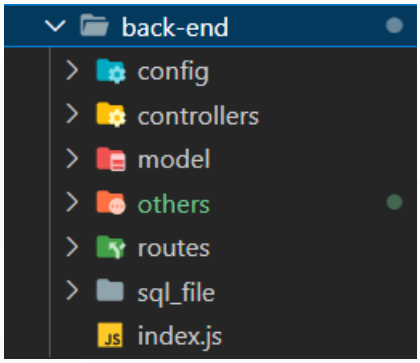
Gambar 3. 17 Tabel Jadwal Bimbingan Pada *Database*

Gambar 3.17 merupakan struktur tabel untuk menampung data jadwal bimbingan pada *database*. Struktur tabel tersebut dikembangkan berdasarkan *ERD* sebagai panduan dan beberapa nama atribut ada yang berbeda seperti di *ERD* karena berdasarkan kesepakatan dengan tim IT. Tabel jadwal bimbingan dengan nama *CalendarTime* tersebut memiliki *primary key* yang bernama *id* dan jika dosen menambah jadwal bimbingan, maka *id* tersebut akan terus bertambah satu nilai dari yang sebelumnya karena ada perintah *increment*. Kemudian, terdapat atribut-atribut lainnya

seperti *eventType* sebagai kategori jadwal yang akan ditambah, *eventName* sebagai nama jadwal, *start_date* dan *end_date* untuk menentukan hari bimbingan, hingga *createdDate* untuk menentukan waktu dan tanggal kapan jadwal tersebut ditambahkan ke dalam *database*. Selain itu, ada atribut dengan nama *notify* yang digunakan sebagai kondisi apakah jadwal bimbingan akan diteruskan atau diberitahukan ke mahasiswa atau tidak, serta *studentYear* untuk menentukan angkatan mahasiswa yang dipilih untuk diberitahukan jadwal bimbingan tersebut. Setelah tabel fitur *Calendar* di *database* dibuat, maka mahasiswa magang melanjutkan pengembangan atau pembuatan *API* untuk fitur tersebut.

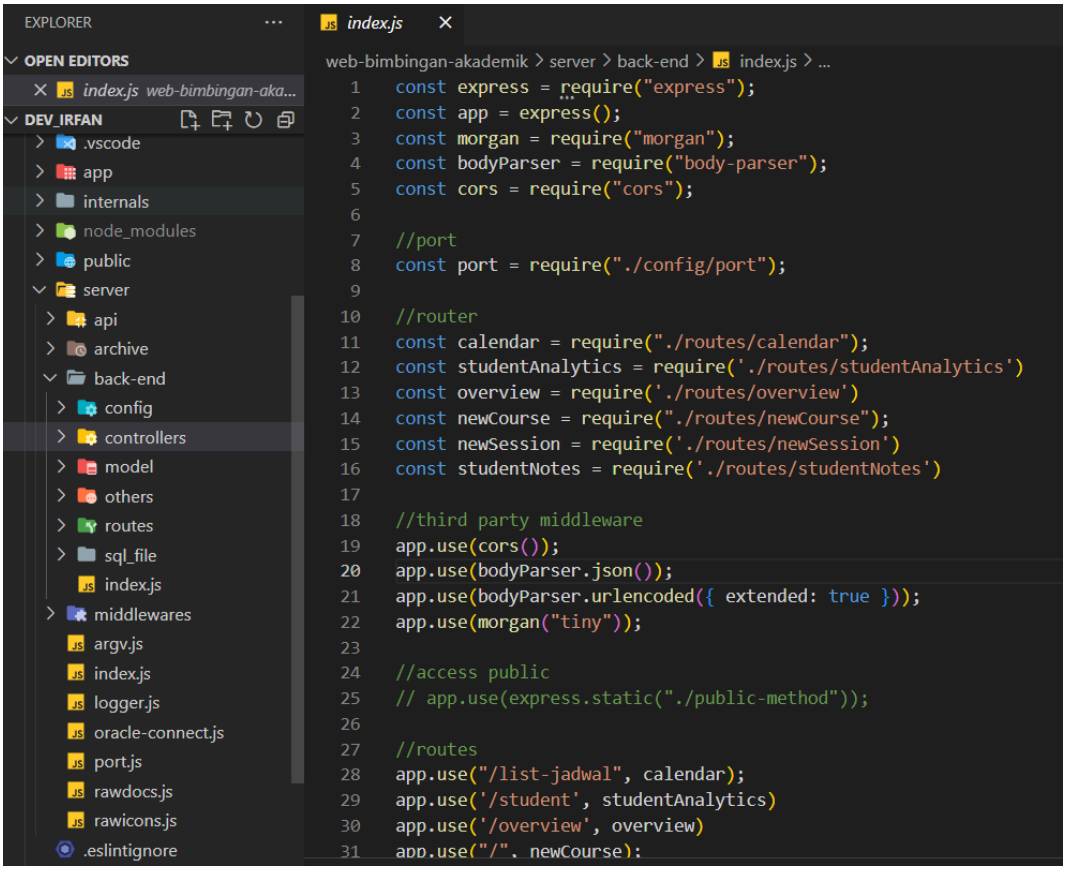
3.2.4.3 Membuat *API* untuk fitur *Calendar*

Tahapan selanjutnya setelah membuat *database* untuk fitur *Calendar* di *SQL Developer* adalah mengembangkan *API* (*Application Programming Interface*) atau disebut juga dengan antarmuka pemrograman aplikasi. *API* dalam *project* website bimbingan akademik ini merupakan fungsi yang digunakan agar *database* dan fitur-fitur *website* dapat terhubung. Pengembangan *API* untuk fitur *Calendar* merupakan awal mula dari pengembangan *back-end* yang menampung *API* fitur-fitur *website* bimbingan akademik lainnya. Proses yang dilakukan untuk dapat mengembangkan *API* fitur *Calendar* adalah menggunakan standar dari tim IT. Standar yang diterapkan tersebut adalah pada *folder back-end* akan terdapat *configuration, model, controllers, routes* dan juga *file index.js* untuk menjalankan *back-end server*.



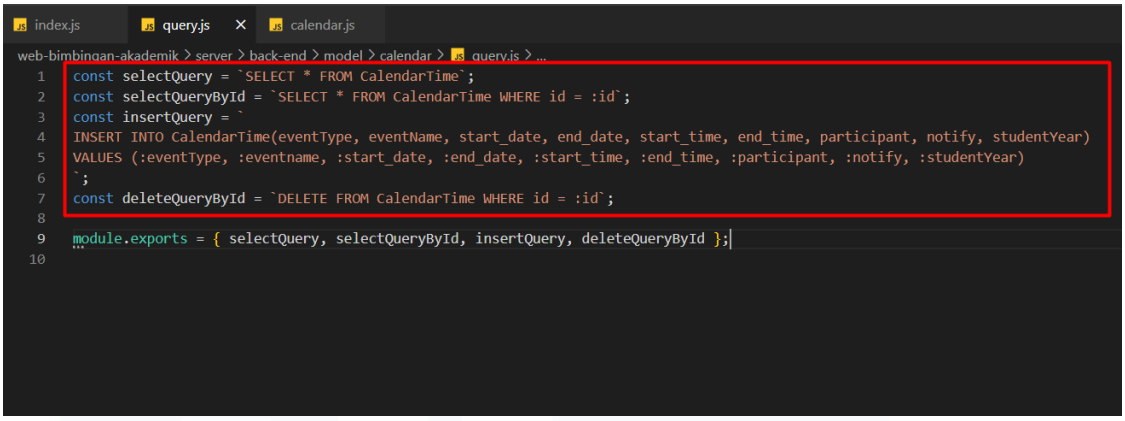
Gambar 3. 18 *Folder Back-End Website* Bimbingan Akademik

Gambar 3.18 adalah struktur *folder back-end* yang digunakan untuk mengembangkan *API* setiap fitur-fitur *website*. Dalam *folder back-end* juga terdapat *folder config* yang digunakan untuk menyimpan *port server back-end* dan juga *database user* untuk keperluan akses *database* di *API*. Terdapat *folder controllers* yang digunakan untuk menyimpan *file code* masing-masing *API* setiap fitur, terutama untuk saat ini fitur *Calendar*. Lalu, terdapat *folder model* untuk menampung *SQL query* seperti *insert*, *select*, *update*, dan juga *delete*. *Folder routes* digunakan untuk menampung *file code* fungsi-fungsi dalam menentukan *API endpoints* (*URL*) sehingga *request client* seperti mengakses data di *database* akan direspon melalui *URL* tersebut. Lalu, terdapat *file* dengan nama *index.js* untuk menjalankan seluruh *code API* dan menjalankan *server back-end* pada *port* 5001 agar keseluruhan fungsi *back-end* dapat bekerja secara fungsional. Selain itu, terdapat *folder sql_file* sebagai *folder backup file SQL* yang digunakan untuk membuat *database*, serta *folder others* untuk menyimpan *file-file* lainnya.



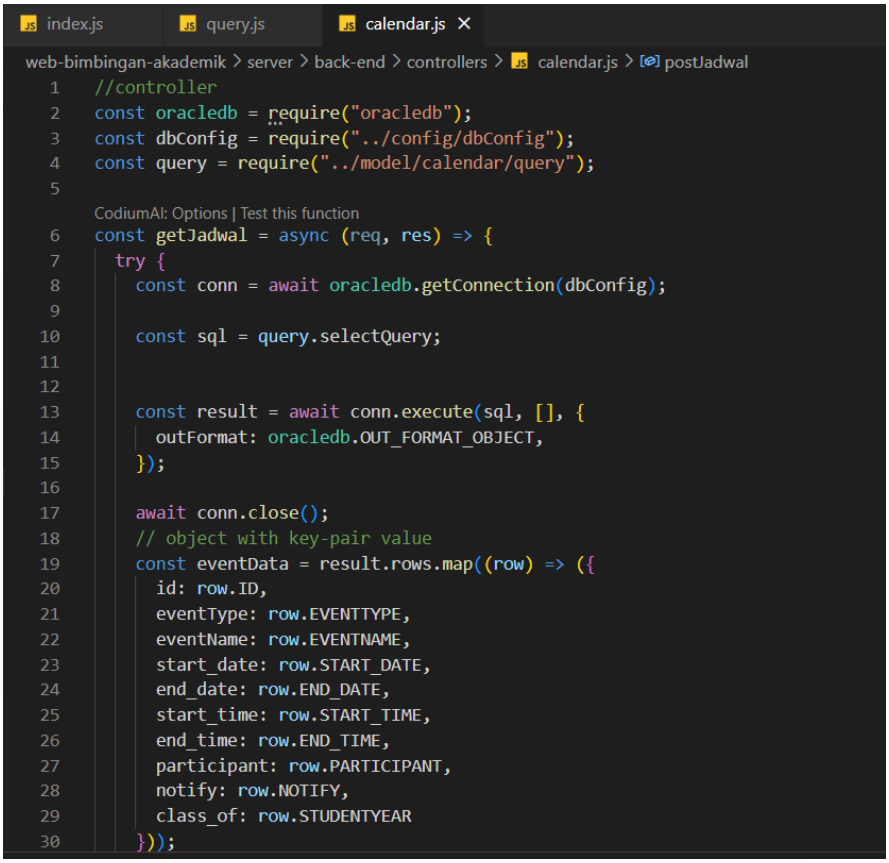
Gambar 3. 19 *File Code index.js* Sebagai *Back-End*

File index.js seperti Gambar 3.19 merupakan *file* utama untuk menjalankan *server back-end* karena menampung keseluruhan *routes* yang menjalnkan fungsi-fungsi *API* setiap fitur *website* bimbingan akademik. Pada *file* tersebut dilakukan *import module* menggunakan perintah *require* seperti *express* dan *module-module* lainnya agar dapat mengembangkan *back-end*. Kemudian, beberapa *routes* juga ditambahkan pada *file* ini agar *API* dapat diakses di *URL* yang telah ditentukan sebagai *endpoint* melalui perintah *app.use*. *File back-end* tersebut dapat dijalankan menggunakan beberapa perintah di *console* atau *terminal* seperti perintah *npm run back-end*, *npm run node-backend*, serta *node index.js* yang dapat dijalankan jika *console* menggunakan *folder* dimana *file* tersebut berada, yaitu *folder back-end*. Selain itu, ada beberapa *module* tambahan seperti *morgan*, *body-parser*, dan juga *cors* untuk keperluan *API*.



Gambar 3. 20 Perintah *SQL* Untuk *API* Fitur *Calendar*

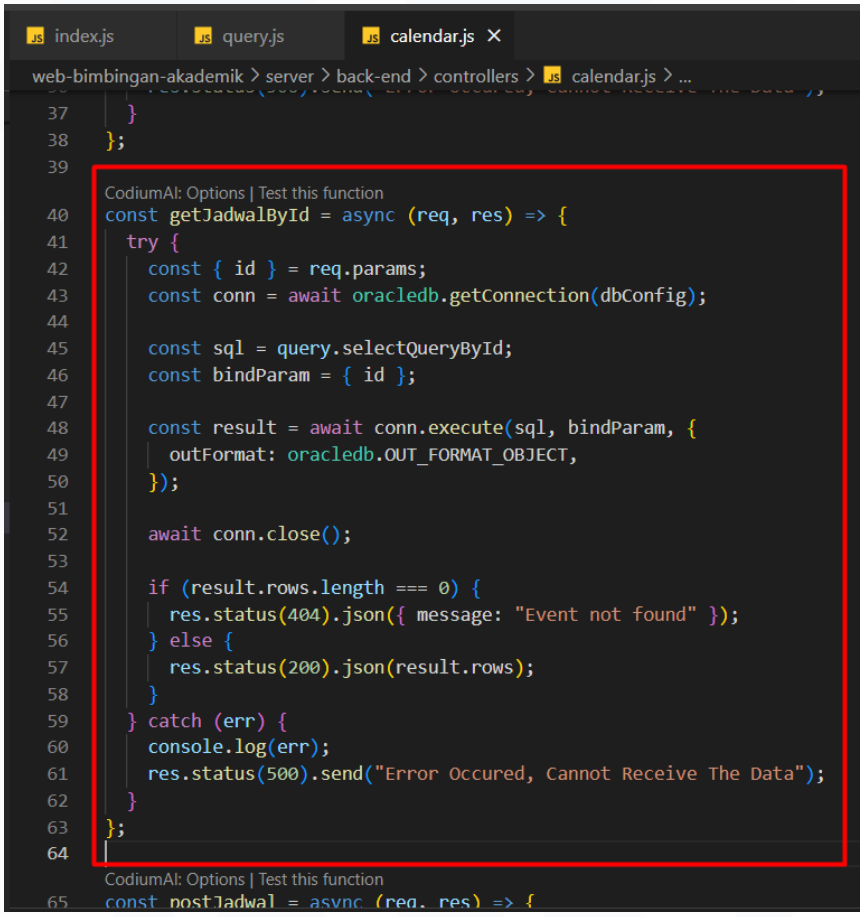
Pada Gambar 3.20, masing-masing perintah *SQL* untuk melakukan *select*, *insert*, dan juga *delete* yang akan diterapkan pada fitur *Calendar*. Perintah *SQL* tersebut disimpan ke dalam masing-masing variabel dan kemudian di-*export* melalui objek *module.exports* agar dapat diakses ke *file controller* fitur *Calendar*. Untuk perintah *select* sendiri terdapat dua variabel, dimana *selectQuery* untuk menampilkan keseluruhan data dalam tabel fitur *Calendar* di *database*, sedangkan *selectQueryById* untuk menampilkan data sesuai *id* jadwal tertentu. Kemudian, perintah *insert* memiliki parameter di bagian *VALUES* agar sesuai dengan *bind parameter* antara atribut di *database* dan juga atribut yang ada pada *request body* di bagian *controller* fitur *Calendar*. Perintah *delete* dideklarasikan dengan variabel *deleteQueryById* yang artinya jika jadwal bimbingan pada fitur *Calendar* akan dihapus, maka jadwal tersebut akan sesuai dengan *id* yang dipilih.



Gambar 3. 21 *Code GET Method* Fitur *Calendar*

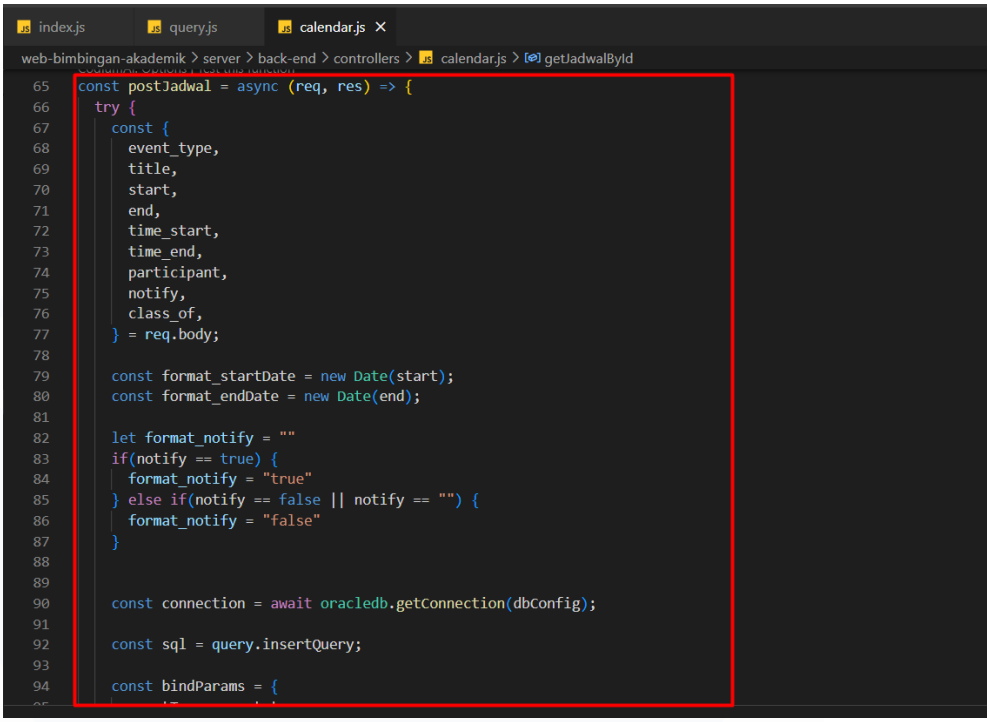
Gambar 3.21 adalah *file code controller* fitur *Calendar* dan menampilkan fungsi melakukan *GET Method* agar data-data *Calendar* pada *database* dapat dikirim ke bagian *front-end*. Selain itu, terdapat *module* yang di-*import* seperti *oracledb* untuk menghubungkan ke *database* dengan bantuan *module dbconfig* sebagai *user info database* tersebut. Kode konfigurasi *user info* tersebut tidak dapat ditampilkan karena untuk menjaga privasi *project*. Untuk perintah *select* yang telah dibuat sebelumnya di-*import* dari *file* bernama *query*. Selanjutnya, pengembangan *API* agar *GET Method* dapat dilakukan harus diterapkan *error handling* dengan adanya fungsi *try catch* yang disediakan oleh *Node Js*. Kemudian, ketika kode dieksekusi melalui perintah *conn.execute*, maka data-data yang akan digunakan sebagai *response* ke bagian *front-end* tersebut disesuaikan namanya agar mudah diakses. Bagian terakhir kode tersebut adalah dengan menjalankan perintah *res.status(200).json()* untuk melanjutkan *response* ke

bagian *front-end* dan angka 200 tersebut menunjukkan bahwa data berhasil dikirim, sedangkan pada bagian *catch* terdapat angka 500 yang artinya ada kesalahan pada *server* sehingga data gagal untuk dikirim.



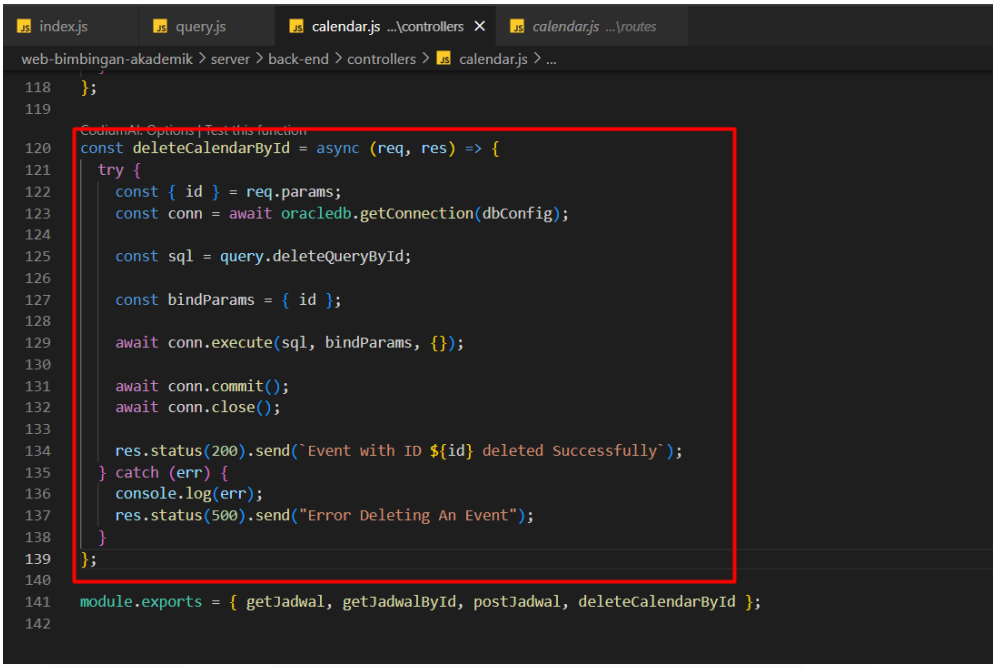
Gambar 3. 22 *Code GET Method* Fitur *Calendar* Berdasarkan *ID*

Gambar 3.22 merupakan *code API* untuk mengirim data ke bagian *front-end* sesuai dengan *id* tertentu dengan adanya objek *req.params*. Kemudian *id* yang berasal dari *req.params* akan dijadikan *bind param* dan dieksekusi bersamaan perintah *SQL* melalui *conn.execute*. Cara kerja dan proses pengembangan dari *API* ini kurang lebih sama seperti pada *API GET Method* untuk menampilkan keseluruhan data fitur *Calendar* yang dibuat sebelumnya.



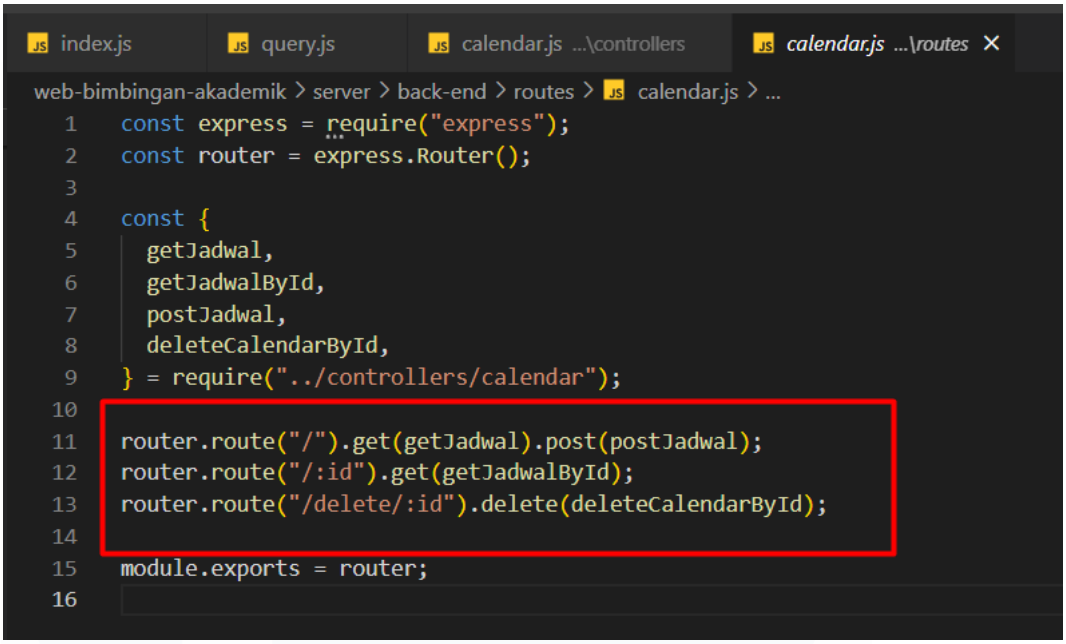
Gambar 3. 23 *Code POST Method* Fitur *Calendar*

Gambar 3.23 merupakan kode untuk menjalankan fungsi *API POST Method* agar data-data yang di-*input* oleh dosen ketika menambahkan jadwal bimbingan dapat masuk ke dalam *database*. Pada *POST Method* yang dibuat untuk fitur *Calendar* ini, terdapat objek *req.body* sebagai daftar nama atribut data-data yang di-*submit* pada *form Add Counseling Schedule*. Kemudian beberapa nama-nama atribut tersebut disesuaikan agar dapat diproses lebih lanjut ke dalam *database*. Kemudian, *req.body* tersebut akan dijadikan *bind param* dan dieksekusi bersamaan dengan perintah *SQL*. Setelah berhasil dieksekusi, maka transaksi ke *database* tersebut disimpan melalui perintah *connection.commit()*.



Gambar 3. 24 *Code DELETE Method* Fitur *Calendar*

Gambar 3.24 adalah kode untuk menjalankan fungsi *delete* jadwal bimbingan yang sudah ditambahkan oleh dosen. Pada fungsi tersebut, terdapat *req.params* yang digunakan untuk menentukan *id* jadwal tertentu sehingga jadwal yang dihapus sesuai dengan *id* tersebut. Setelah perintah *SQL* dan *bind param* dieksekusi, maka dilakukan *commit* agar perubahan tersebut tersimpan. Setelah semua fungsi-fungsi *API* dibuat, maka dilakukan *export* agar dapat diakses pada bagian *routes file*.

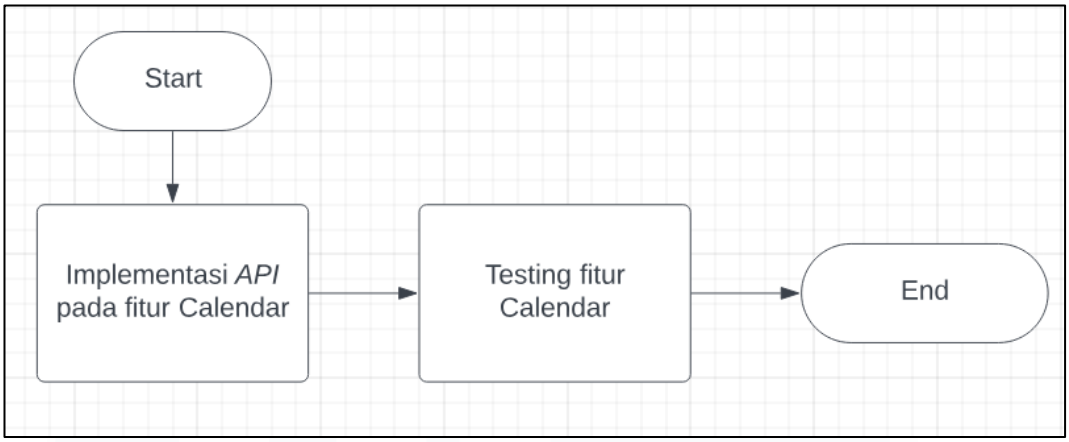


Gambar 3. 25 *Code Routes* Fitur *Calendar*

Gambar 3.25 adalah kode untuk menerapkan fungsi-fungsi *API* atau *method* yang telah di-*import* dari *file controller* fitur *Calendar*. *Router* pada *code* tersebut berfungsi untuk membuat *API Endpoint* atau *URL* agar pada bagian *front-end* dapat terhubung dengan *database* untuk keperluan pertukaran data. Pada bagian *getJadwalById* terdapat parameter yang ditandai dengan "*:id*", serta pada *deleteCalendarById* juga terdapat parameter yang sama. Hal tersebut berguna untuk memenuhi kondisi *id* jadwal bimbingan agar sesuai dengan *req.params* di bagian kode *API* fitur *Calendar*.

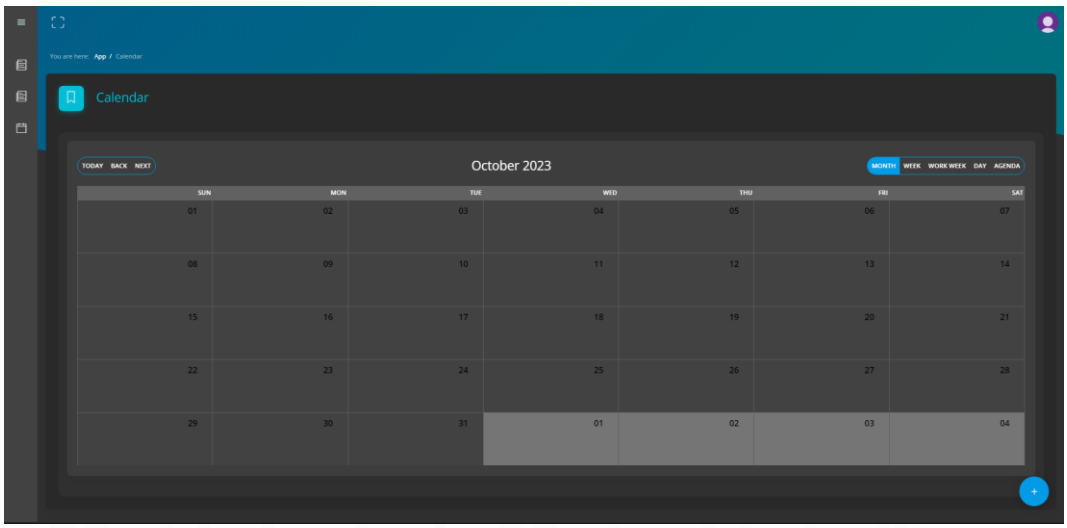
3.2.5 Menerapkan *API* pada fitur *Calendar* (Minggu 5-6)

Aktivitas magang berikutnya adalah menerapkan *API* yang sudah dibuat untuk fitur *Calendar* di bagian *front-end* agar data-data hasil *submit* jadwal bimbingan dapat terhubung atau dihubungkan ke *database*.



Gambar 3. 26 Proses Penerapan *API* Pada Fitur *Calendar*

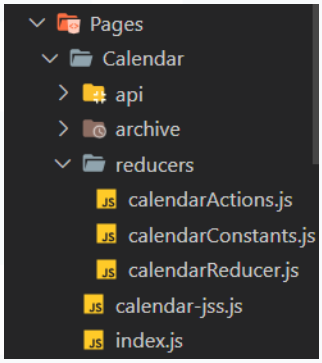
Gambar 3.26 merupakan gambaran umum pada proses implementasi *API* untuk fitur *Calendar*. Proses tersebut dimulai dari implementasi fungsi *API* dan menghubungkannya dengan bagian *front-end* fitur *Calendar*. Tujuan dihubungkannya dengan *API* tersebut adalah agar *user* dapat menambah jadwal bimbingan ataupun menghapus jadwal tersebut. Kemudian proses dilanjutkandan diakhiri dengan *testing* fitur *Calendar*, serta menerapkan fungsi *React-Redux* agar deskripsi dari *action user* dapat muncul sebagai *notification action*.



Gambar 3. 27 Halaman Utama Fitur *Calendar*

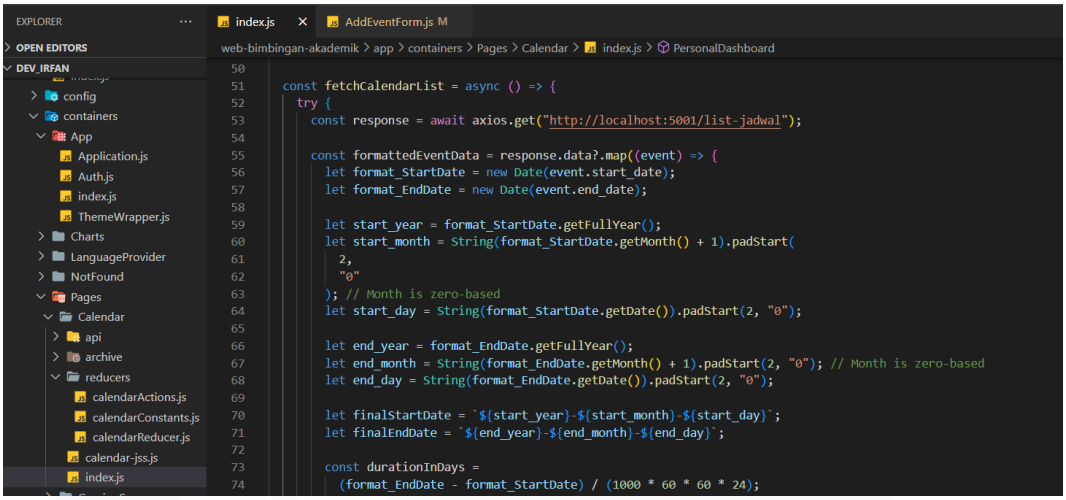
Pada Gambar 3.27 merupakan tampilan dari halaman fitur *Calendar* yang digunakan untuk menampilkan dan menambahkan jadwal bimbingan. Tampilan

front-end tersebut memiliki struktur *folder component* dan juga *pages*. *Folder component* berguna untuk menyimpan kode-kode komponen seperti menampilkan jadwal bimbingan, *form* untuk menambah jadwal bimbingan dan juga keseluruhan tampilan fitur *Calendar*. *Folder pages* sendiri digunakan untuk menyimpan *file index.js* fitur *Calendar* agar halaman tersebut dapat diakses secara menyeluruh termasuk komponennya. Selain itu, pada *folder pages* terdapat *folder* dan juga *file* yang akan digunakan dalam menerapkan fungsi *React Redux*.



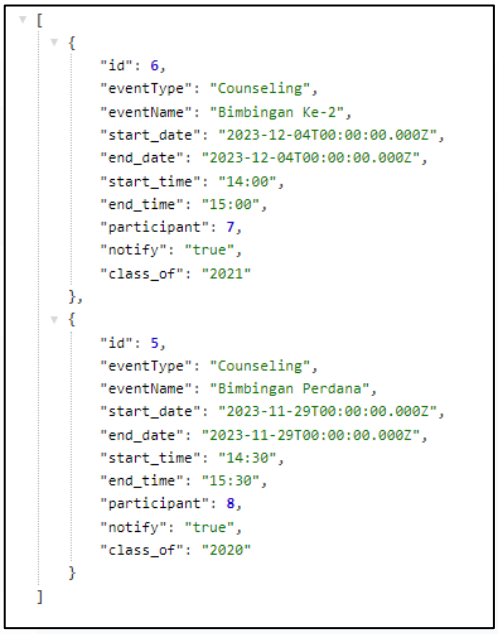
Gambar 3. 28 Fitur *Calendar* Pada *Folder Pages*

Gambar 3.28 merupakan struktur *folder pages* dan terdapat *folder* fitur *Calendar* dengan *file* utama pada *index.js*. *File* utama tersebut dijadikan sebagai tempat untuk menempatkan *component* terkait fitur *Calendar* dan juga menjalankan perintah *axios* untuk mengakses *database* berdasarkan *request method* yang digunakan melalui perantara *API Endpoint*.



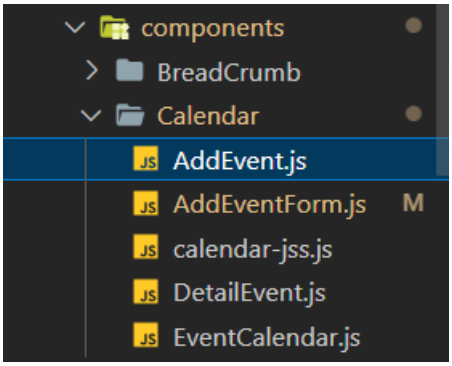
Gambar 3. 29 *Code* Utama Fitur *Calendar*

Gambar 3.29 menampilkan *code-code* pada *file index.js* yang dijadikan sebagai *file* utama fitur *Calendar*. Terdapat fungsi untuk mengambil data dari *database* yang dideklarasikan melalui variabel *fetchCalendarList*. Fungsi tersebut menjalankan perintah *axios.get* agar data jadwal bimbingan dapat diakses dan ditampilkan pada fitur *Calendar*. Kemudian, fungsi tersebut menjalankan perintah *axios* dengan cara *async await* yang berarti ketika fitur atau tampilan halaman *Calendar* di-*render*, fungsi *fetchCalendarList* akan dijalankan secara bersamaan atau paralel dan data-data jadwal bimbingan dapat ditampilkan secara bersamaan, tentunya juga dengan bantuan fungsi *useEffect* dari *React*. Kemudian, pada *file index.js* tersebut, data-data untuk fitur *Calendar* akan dikonversi pada bagian tanggal agar dapat ditampilkan dengan mudah. Lalu, setelah data jadwal bimbingan sudah masuk ke bagian *front-end*, maka diteruskan menggunakan *props* agar data dapat diakses ke komponen-komponen fitur *Calendar*.



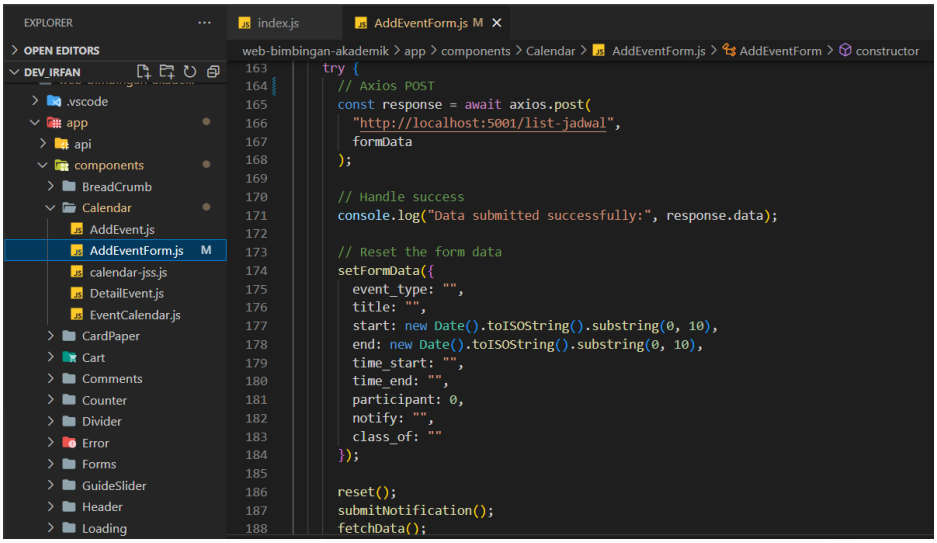
Gambar 3. 30 Data Jadwal Bimbingan Fitur *Calendar*

Gambar 3.30 adalah data jadwal bimbingan berbentuk *JSON* dan diakses menggunakan *GET method* melalui *API endpoint* untuk dapat mengakses *API* fitur *Calendar* yang sudah dibuat sebelumnya. Sebelum data tersebut dapat muncul, maka dosen harus melakukan *submit form* atau menambah jadwal bimbingan melalui komponen *AddEventForm*.



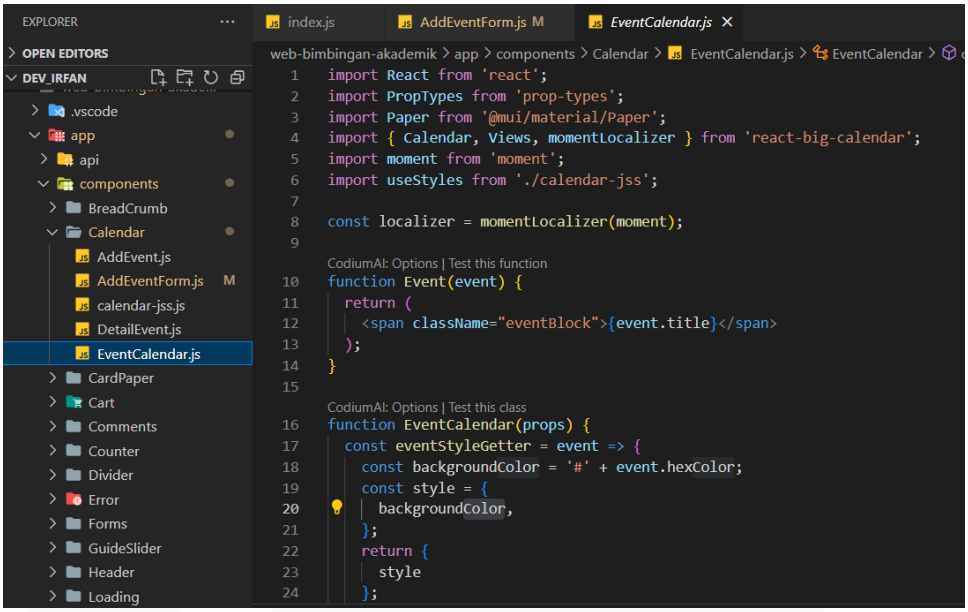
Gambar 3. 31 *Folder Component* Fitur *Calendar*

Gambar 3.31 adalah struktur komponen fitur *Calendar* yang ditempatkan di dalam *folder components*. Komponen-komponen tersebut dijadikan sebagai bagian dari fitur *Calendar* dan memiliki fungsinya masing-masing seperti untuk menampilkan jadwal bimbingan atau menambah jadwal bimbingan.



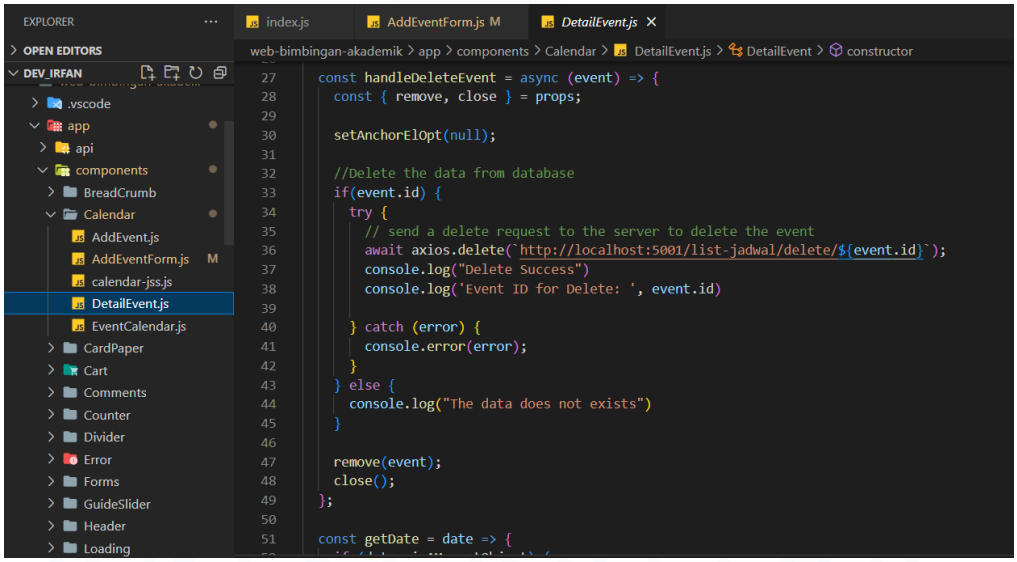
Gambar 3. 32 *Code AddEventForm* Fitur *Calendar*

Gambar 3.32 adalah *code* pada komponen fitur *Calendar* untuk dapat melakukan *submit form* jadwal bimbingan yang di-*input* oleh dosen. Data yang di-*input* melalui *form* jadwal bimbingan dapat masuk ke dalam *database* dengan bantuan perintah *axios.post* yang akan melakukan *POST method* sebagai *request* ke *API* agar tersimpan di *database*. Kemudian, dengan bantuan *useState* yang merupakan fungsi bawaan *React*, data-data tersebut dapat di-*submit* karena memiliki atribut-atribut sebagai *req.body* sama seperti di *file controller* fitur *Calendar* yang telah dibuat sebelumnya.



Gambar 3. 33 *Code EventCalendar* Fitur *Calendar*

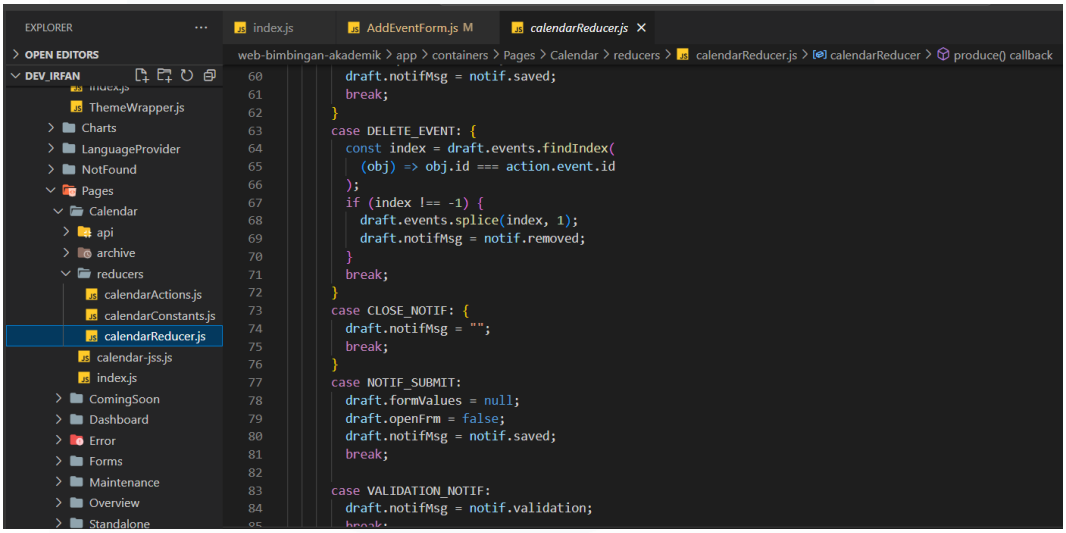
Gambar 3.33 adalah *file code* untuk dapat menampilkan jadwal bimbingan pada fitur *Calendar*. Komponen tersebut mendapatkan akses data melalui adanya *props* yang ada di *file index.js* sebelumnya.



Gambar 3. 34 *Code DetailEvent* Fitur *Calendar*

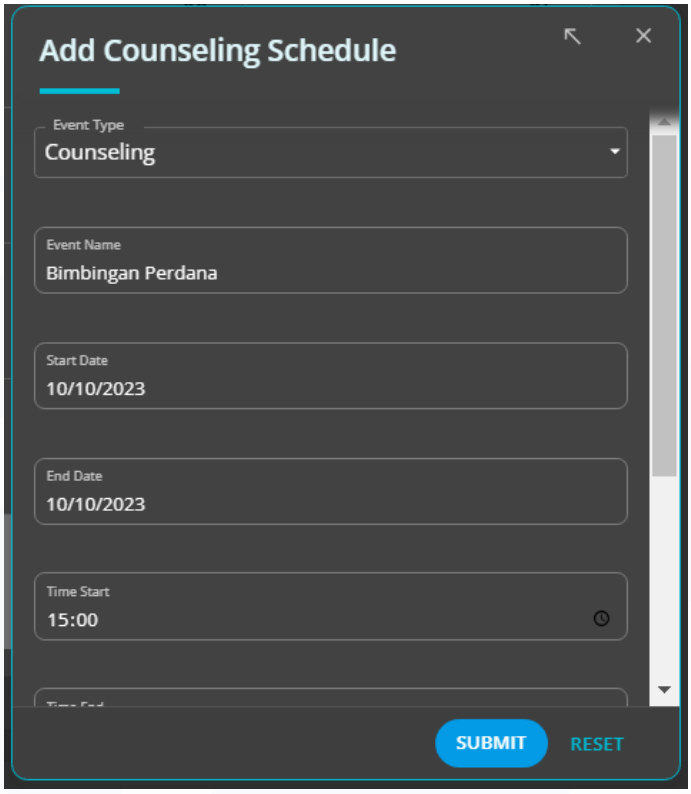
Gambar 3.34 adalah tampilan dari *code* yang terdapat di *file DetailEvent.js* dan berguna untuk menampilkan jadwal bimbingan secara lebih detil, sehingga terlihat tanggal bimbingan, waktu mulai dan berakhir bimbingan, jumlah

partisipan, hingga tahun angkatan mahasiswa yang akan bimbingan. Selain itu, terdapat fungsi untuk melakukan *delete* atau menghapus jadwal bimbingan menggunakan *DELETE method* dari perintah *axios.delete*. Jadwal tersebut dihapus di *database* dengan bantuan *API endpoint* yang terhubung dengan fungsi *API* di *back-end* dan juga terdapat *parameter event.id* agar jadwal yang terhapus sesuai dengan jadwal yang dipilih.



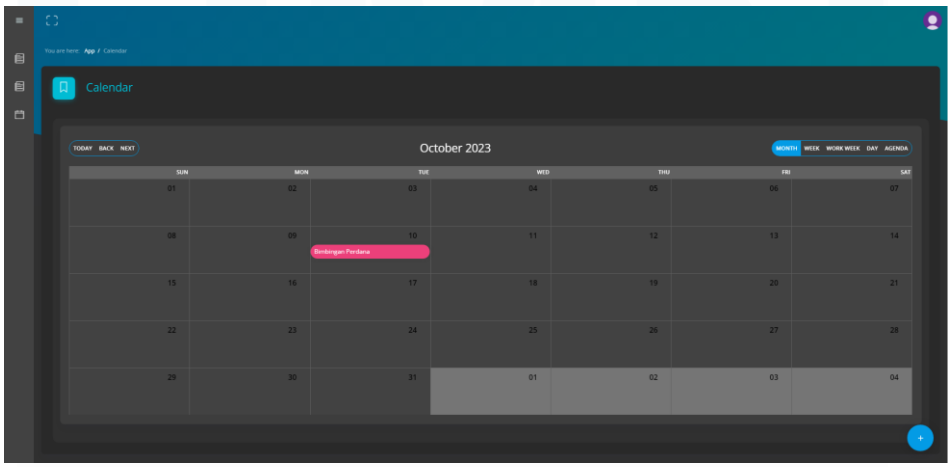
Gambar 3. 35 *Code React Redux* Fitur *Calendar*

Gambar 3.35 adalah tampilan dari *code* untuk menampung fungsi *React Redux* agar dapat memunculkan *action notification* ketika dosen menambah, membatalkan *input* jadwal, dan menghapus jadwal bimbingan. Agar fungsi *redux* tersebut dapat digunakan, pada bagian *file index.js* atau *file* utama fitur *Calendar* ditambahkan penggunaan *useDispatch* sehingga *notification* sesuai dengan *action* yang dilakukan oleh *user* atau dosen. Kemudian, setelah ditambahkan fungsi untuk dapat mengakses *API* pada fitur *Calendar* dan juga *React Redux* agar muncul *notification action*, maka selanjutnya adalah melakukan *testing* fitur agar dapat diketahui bahwa fitur *Calendar* fungsional dalam menerima dan mengirim data.



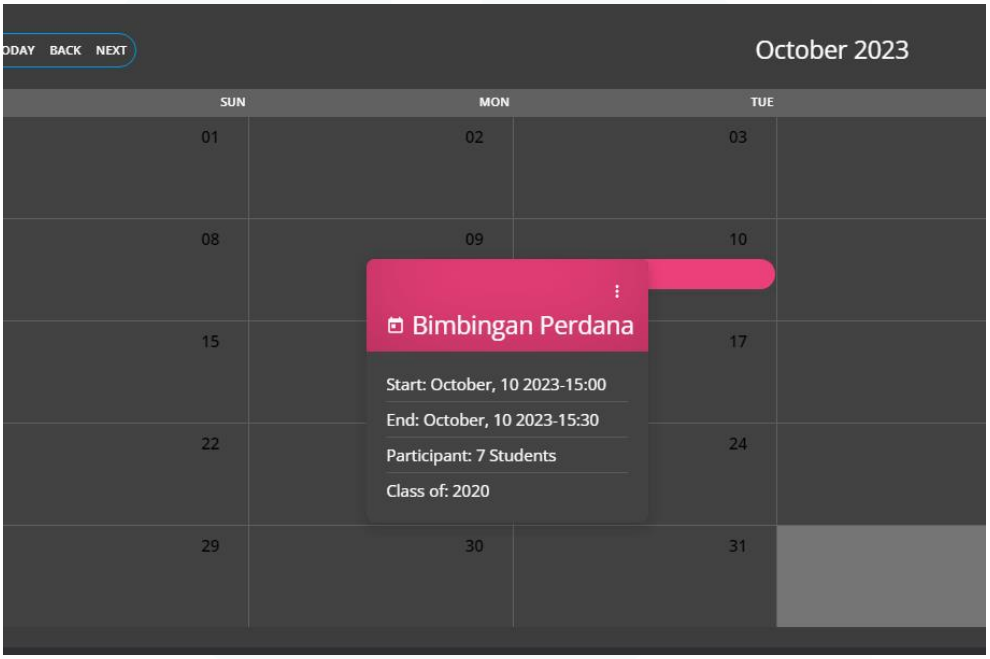
Gambar 3. 36 *Testing* Menambah Jadwal Bimbingan

Gambar 3.36 adalah tampilan komponen *AddEventForm* yang digunakan untuk menambah jadwal bimbingan. Setelah masing-masing *input form* tersebut diisi, kemudian dosen dapat menekan tombol *submit* untuk menambah jadwal dan tersimpan di *database*, atau menekan tombol *reset* agar isian *form* di-*reset*.



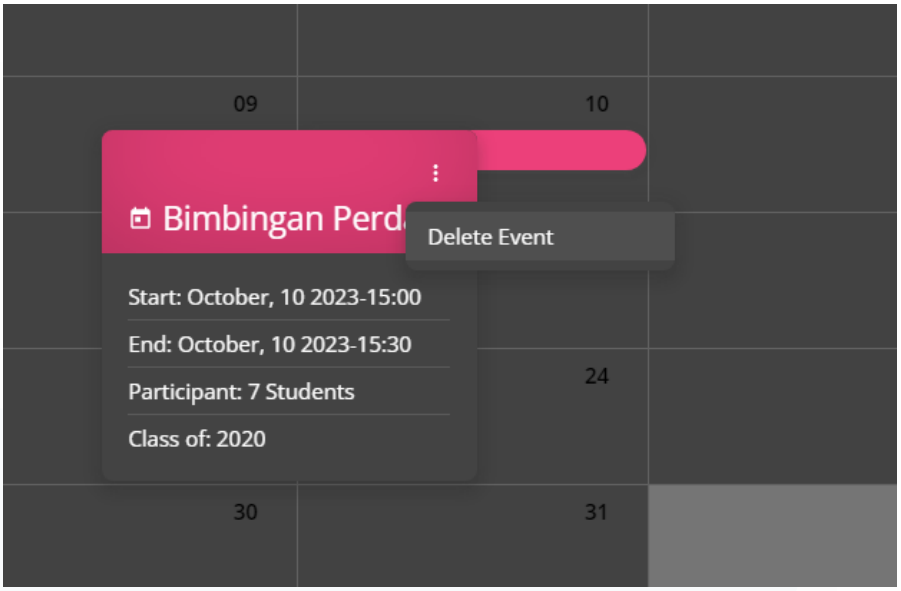
Gambar 3. 37 *Testing* Melihat Jadwal Bimbingan

Gambar 3.37 adalah tampilan fitur *Calendar* jika jadwal bimbingan ditambahkan oleh dosen. Untuk dapat melihat secara lebih detil, maka dosen akan menekan jadwal bimbingan tersebut.



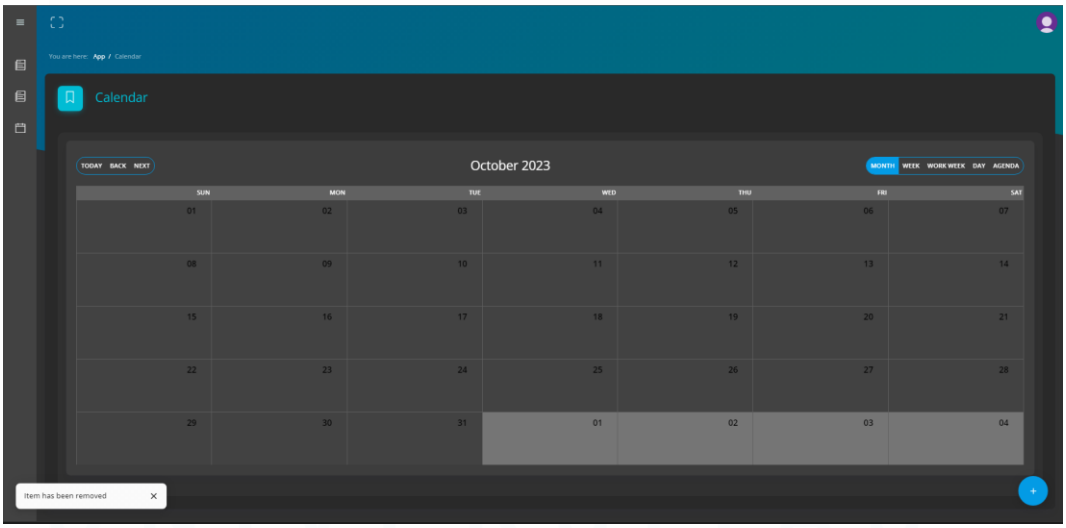
Gambar 3. 38 *Testing* Menampilkan Detil Jadwal Bimbingan

Gambar 3.38 adalah tampilan ketika jadwal bimbingan ditekan dan akan menampilkan jadwal secara lebih detil, serta terdapat tombol untuk menghapus pada bagian yang ditandai oleh *icon* "*titik tiga*".



Gambar 3. 39 *Testing* Menghapus Jadwal Bimbingan

Gambar 3.39 adalah tampilan dari tombol untuk menghapus jadwal bimbingan sesuai dengan *id* atau jadwal yang dipilih. Ketika tulisan *delete event* ditekan, maka akan muncul *notification action* pada bagian kiri bawah tampilan fitur *Calendar* dan data jadwal bimbingan tersebut terhapus pada *database*.



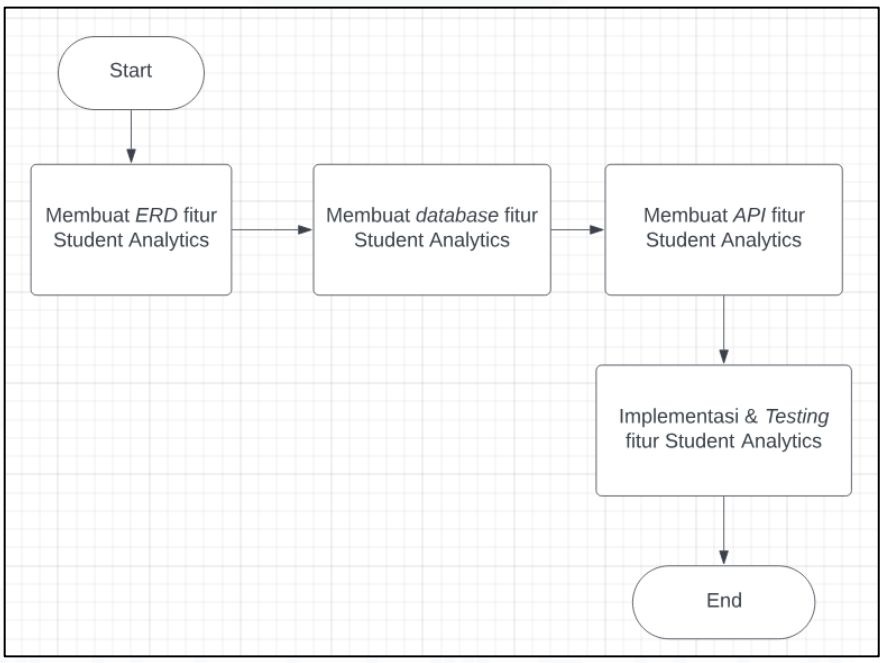
Gambar 3. 40 Tampilan *Notification Action* Fitur *Calendar*

Gambar 3.40 adalah tampilan keseluruhan fitur *Calendar* sesudah jadwal bimbingan yang ditambahkan sebelumnya telah dihapus. Pada bagian kiri bawah terdapat *notification action* yang menunjukkan bahwa jadwal bimbingan

telah dihapus. Setelah melakukan *testing* fitur *Calendar* ini, mahasiswa magang akan melanjutkan ke pengerjaan fitur berikutnya, yaitu fitur *Student Analytics* karena fitur *Calendar* sudah berhasil dijalankan dan bekerja secara fungsional.

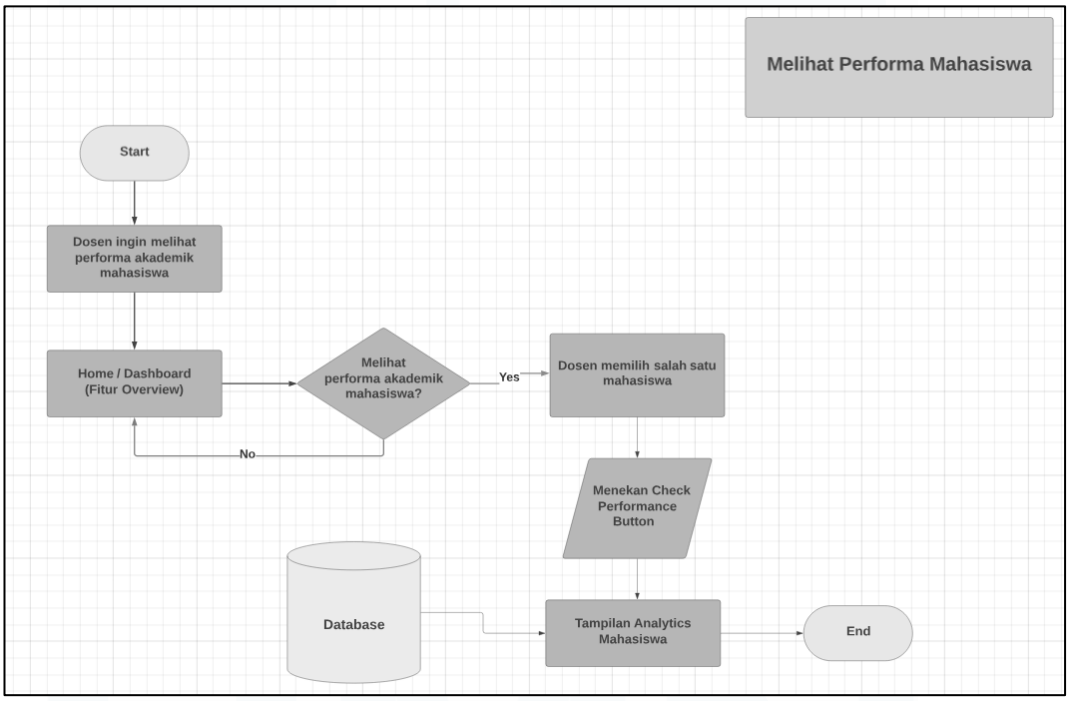
3.2.6 Menerapkan *API* pada fitur *Student Analytics* (Minggu 7-8)

Aktivitas atau kegiatan magang berikutnya adalah melanjutkan pengerjaan fitur yang lainnya, yaitu fitur *Student Analytics*. Fitur *Student Analytics* merupakan fitur yang digunakan untuk menampilkan data-data terkait performa akademik mahasiswa. Berdasarkan dokumen internal *project* dari periode magang *front-end* sebelumnya, tidak disebutkan secara spesifik mengenai *user-flow* yang ada di fitur *Student Analytics* karena fitur tersebut juga hanya berfungsi untuk menampilkan performa akademik mahasiswa tetapi fitur tersebut nantinya dapat diakses melalui fitur *Overview* melalui sebuah tombol di dalam komponen *studen table*.



Gambar 3. 41 Proses Penerapan *API* Fitur *Student Analytics*

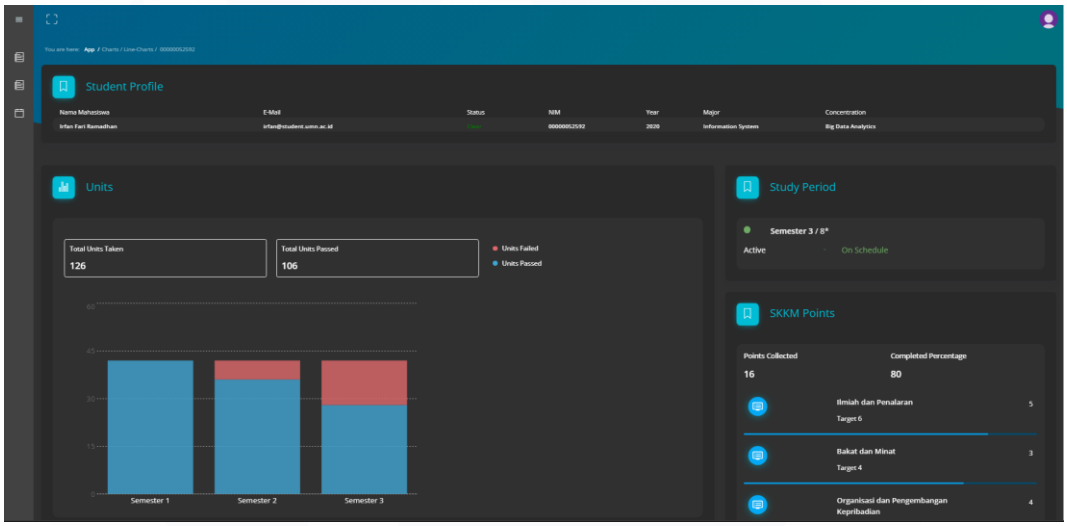
Gambar 3.41 adalah proses yang dilakukan dalam menerapkan *API* untuk fitur *Student Analytics*. Proses yang dilakukan pertama adalah membuat *ERD* untuk fitur tersebut dan melanjutkan pembuatan tabel *database* agar data-data *analytics* mahasiswa dapat ditampilkan. Setelah membuat tabel *database*, proses dilanjutkan dengan membuat *API* fitur *Student Analytics*. Proses tersebut diakhiri dengan implementasi dan *testing API* yang sudah dikembangkan tersebut sehingga dapat fungsional dan data berhasil ditampilkan.



Gambar 3. 42 *User Flow* Fitur *Student Analytics*

Gambar 3.42 adalah *user flow* untuk *user* agar dapat mengakses fitur *Student Analytics*. Dosen sebagai *user* dapat menuju ke halaman utama *website* bimbingan akademik yaitu halaman *dashboard* yang merupakan fitur *Overview*. Kemudian, jika dosen ingin melihat performa akademik salah satu mahasiswa, maka dapat menekan sebuah tombol dengan berisikan *text* bernama "*Check Performance*" agar dapat melihat peforma akademik tersebut. Setelah dosen menekan tombol tersebut, maka akan diarahkan ke halaman *Student Analytics* dan tampilan pada fitur tersebut berupa data-data mahasiswa yang berkaitan dengan akademik seperti jumlah SKS, IPS, IPK, jumlah SKKM, jumlah absensi

mahasiswa pada mata kuliah yang diambil, dan yang lainnya. Dengan adanya *user flow* fitur *Student Analytics*, mahasiswa magang dapat mengerjakan dan implementasi *API* secara lebih mudah karena dapat menjadi acuan untuk pembuatan *ERD*, *database*, ataupun *API* fitur tersebut.

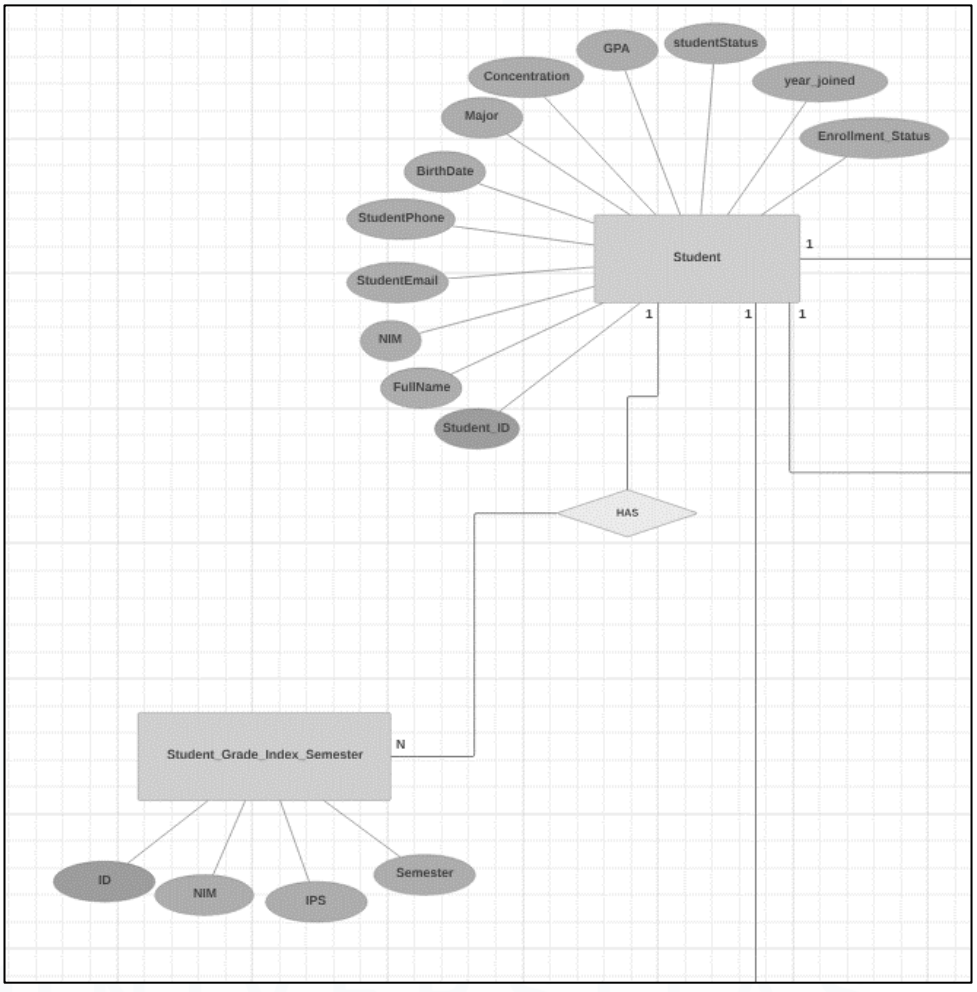


Gambar 3. 43 Halaman Utama Fitur *Student Analytics*

Gambar 3.43 adalah tampilan dari halaman utama untuk fitur *Student Analytics*. Terdapat beberapa komponen yang digunakan seperti *student profile* sebagai informasi singkat mahasiswa, *units* untuk melihat jumlah SKS mahasiswa, *study period* untuk menampilkan status mahasiswa dan juga semester aktifnya, *skkm points* untuk melihat jumlah SKKM yang didapatkan oleh mahasiswa. Kemudian, jika di-*scroll* ke bawah maka terdapat dua komponen lainnya seperti *gpa* untuk menampilkan IPK dan IPS mahasiswa di setiap semester, serta komponen *absence* untuk melihat absensi mahasiswa pada mata kuliah yang sedang diambil. Pekerjaan yang dilakukan sebelum implementasi *API* pada fitur tersebut adalah membuat *ERD*, *database*, dan tentunya *API*. Setelah *API* dibuat, maka dilanjutkan untuk menghubungkannya ke bagian *front-end*. Selain itu, nantinya akan ada *meeting* yang membahas mengenai pekerjaan seperti *update progress* pada fitur yang dikerjakan.

3.2.6.1 Membuat *ERD, Database,* dan *API* fitur *Student Analytics*

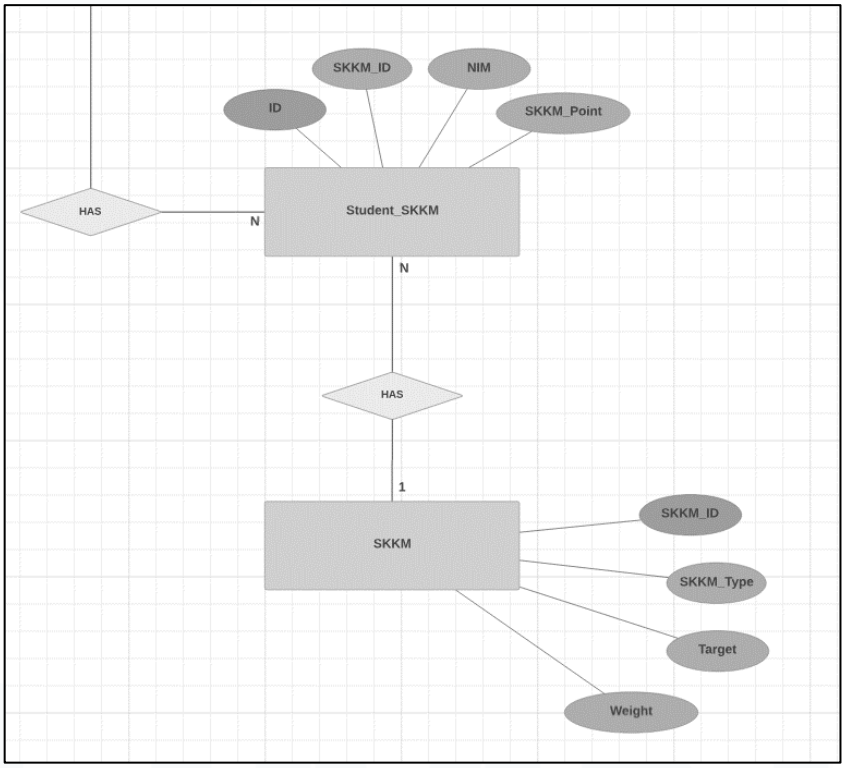
Tahapan pertama yang dilakukan adalah dengan membuat *ERD* untuk fitur *Student Analytics*. *ERD* untuk fitur nantinya menyatu dengan *ERD* fitur *Overview* karena beberapa objek atau *entity* sebagai struktur dasar dalam membuat *database* digunakan kembali di fitur *Overview*. Selain itu, semua *ERD* fitur, baik yang sudah dibuat atau akan dibuat merupakan *ERD* yang sudah dikonfirmasi dengan *product owner* dan telah di-*approve*.



Gambar 3. 44 *ERD* Fitur *Student Analytics* Bagian *Student*

Gambar 3.44 adalah *ERD* untuk fitur *Student Analytics* yang menampilkan objek *student* dan juga *student gpa*. Objek *student* dijadikan sebagai gambaran atau konsep untuk membuat tabel mahasiswa di

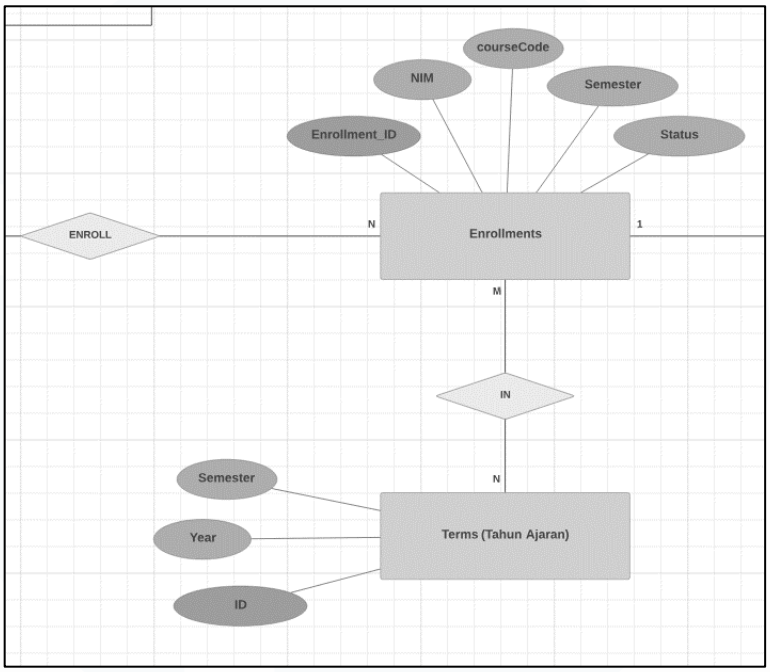
database. Terdapat objek *student gpa* dengan nama *Student_Grade_Index_Semester* yang dijadikan gambaran dalam membuat tabel yang menampung IPK dan IPS mahasiswa. Hubungan antara kedua *entity* tersebut adalah *one-to-many* dimana setiap satu mahasiswa dapat memiliki banyak *row* atau baris data karena *value* yang disimpan dihitung per semester.



Gambar 3. 45 *ERD* Fitur *Student Analytics* Bagian SKKM

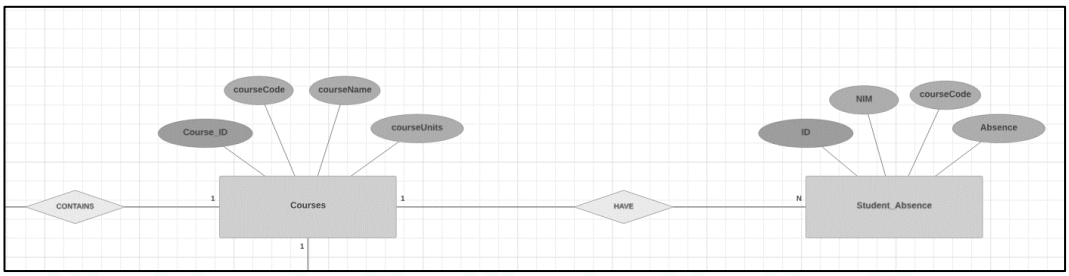
Gambar 3.45 menampilkan bagian *ERD* antara *student* dengan SKKM. *Entity* atau objek *Student_SKKM* terhubung dengan objek *SKKM* yang berfungsi untuk menyimpan nilai bawaan (*default value*) untuk masing-masing jenis SKKM, seperti Ilmiah Penalaran, Bakat dan Minat, dan yang lainnya. Hubungan dengan objek *student* yang sudah dibuat sebelumnya adalah *one-to-many* karena setiap mahasiswa memiliki banyak poin SKKM yang didapatkan dari aktivitas yang berbeda juga. Untuk hubungan antara *Student_SKKM* dan *SKKM* adalah *many-to-one* karena untuk masing-

masing poin yang dimiliki mahasiswa pastinya memiliki jenis SKKM, bobot SKKM dan juga *id* SKKM.



Gambar 3. 46 *ERD* Fitur *Student Analytics* Bagian *Enrollments*

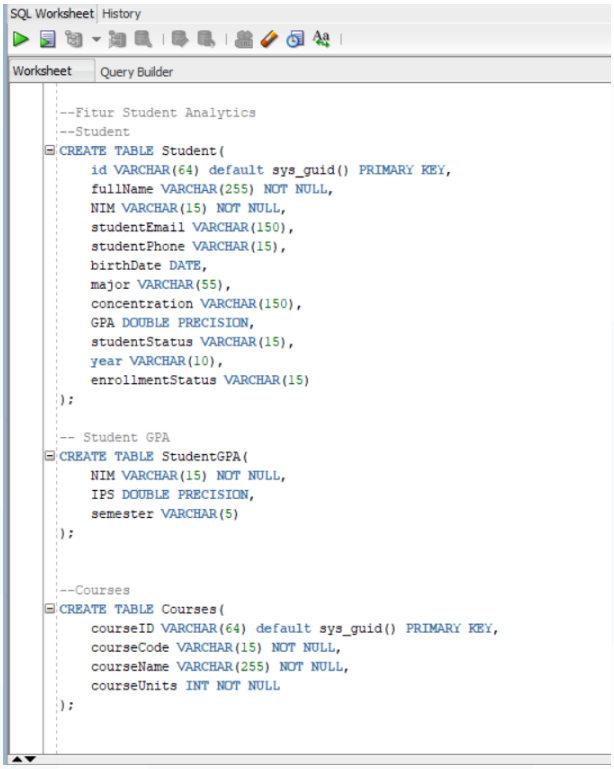
Gambar 3.46 merupakan bagian *ERD* fitur *Student Analytics* dimana terdapat *entity Enrollments* dan juga terhubung dengan *Terms*. *Entity Enrollments* menjadi struktur tabel untuk mata kuliah yang didaftarkan mahasiswa ketika melakukan KRS. Kemudian terdapat *entity Terms* yang merupakan tahun ajaran mata kuliah yang didaftarkan tersebut. Untuk hubungan antara *Enrollments* dengan mahasiswa adalah *many-to-many* karena satu mahasiswa dapat memilih banyak mata kuliah untuk didaftarkan di semester yang dipilih sesuai dengan tahun ajaran (*terms*).



Gambar 3. 47 *ERD* Fitur *Student Analytics* Bagian *Courses*

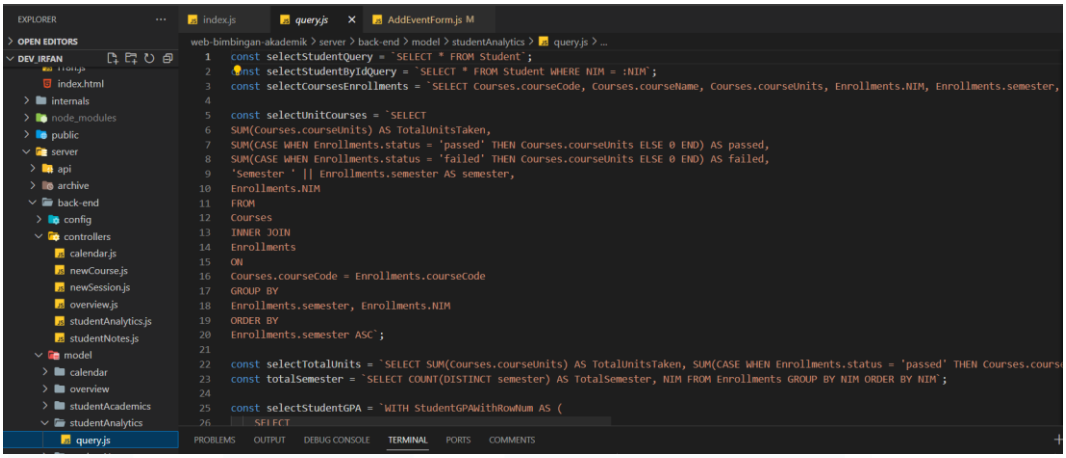
Gambar 3.47 adalah tampilan objek *Courses* dan terhubung dengan *Student Absence*. Objek tersebut akan dijadikan pedoman dalam membuat tabel-tabel terkait fitur *Student Analytics* di *database*. Objek *Courses* sebelumnya terhubung dengan objek *Enrollments*. Berbeda dengan *Enrollments*, objek *Courses* ini digunakan untuk menampung detil mata kuliah seperti nama dan jumlah SKS. Hubungan yang ada pada *Enrollments* dengan *Courses* adalah *one-to-one* karena setiap satu mata kuliah yang ada di *Enrollments* memiliki nama dan jumlah SKS yang hanya sesuai dengan satu kode mata kuliah tersebut. Selain itu, objek atau entitas *Courses* terhubung secara *one-to-many* dengan objek *Student Absence* karena setiap mata kuliah yang diambil mahasiswa memiliki beberapa jumlah absensi mahasiswa tersebut atau tidak sama sekali.

Setelah *ERD* fitur *Student Analytics* dibuat, maka mahasiswa magang melanjutkan ke tahapan pembuatan tabel di *database*. Tabel tersebut akan dihubungkan di bagian pengembangan *API* agar data-data tersebut dapat ditampilkan di bagian *front-end* fitur *Student Analytics*.



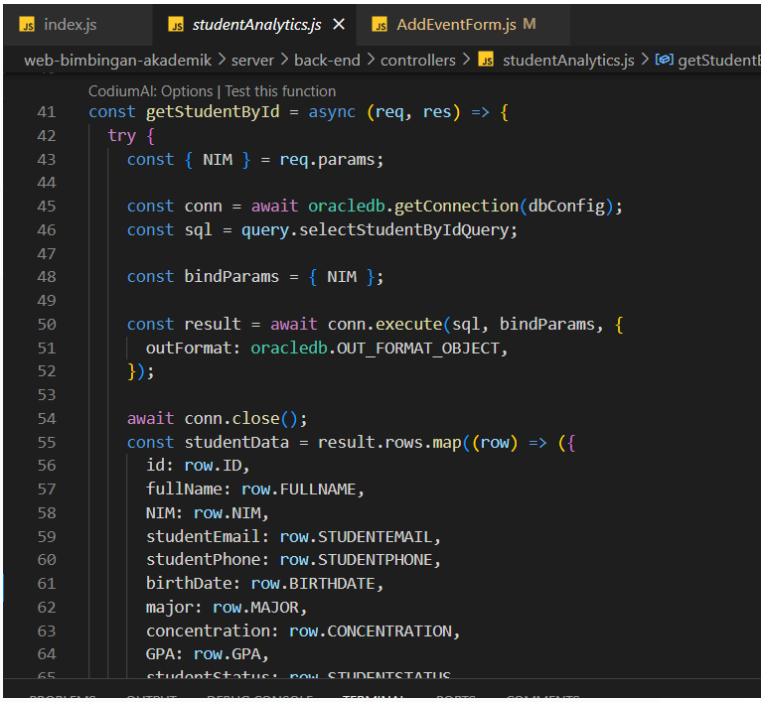
Gambar 3. 48 Tabel *Database* Fitur *Student Analytics*

Gambar 3.48 adalah tampilan struktur tabel *Student Analytics* yang dibuat berdasarkan *ERD* fitur tersebut. Tabel yang dibuat mulai dari *Student*, *StudentGPA*, *Courses*, dan yang lainnya untuk keperluan fitur *Student Analytics*. Beberapa nama kolom atau atribut per masing-masing tabel ada sedikit berbeda dari *ERD* karena disesuaikan untuk atribut yang ada di bagian *front-end* dengan tujuan untuk mempermudah implementasi *API*. Selain itu, tipe data untuk masing-masing atribut juga disesuaikan dengan bagian *front-end*. Kemudian, mahasiswa melanjutkan untuk menambahkan data-data ke dalam tabel tersebut agar fitur *Student Analytics* menjadi lebih fungsional.



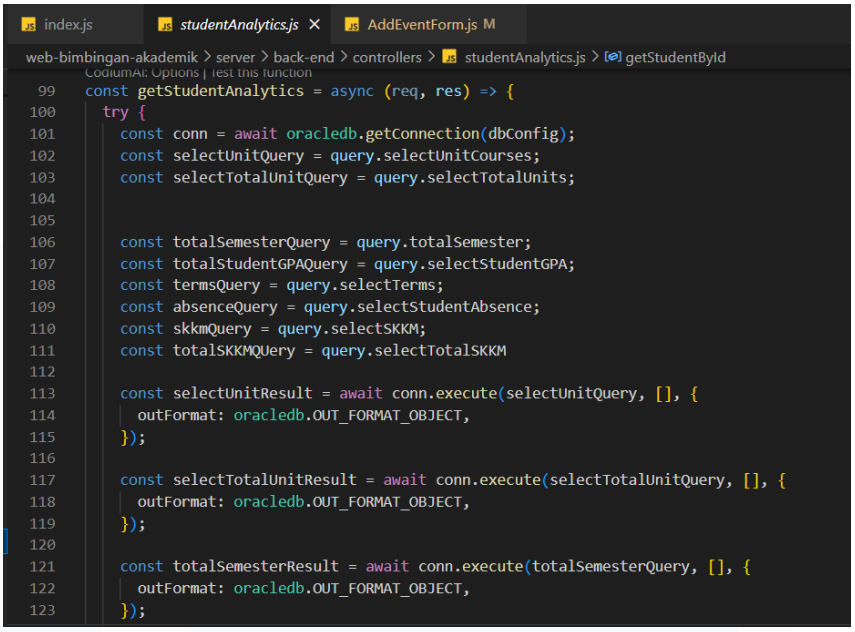
Gambar 3. 49 Perintah *SQL* Untuk *API* Fitur *Student Analytics*

Gambar 3.49 merupakan *code* perintah *SQL* berupa *select statement* untuk dapat menampilkan data-data pada fitur *Student Analytics*. Setiap variabel memiliki nama dan tujuan masing-masing, sebagian besar digunakan untuk dapat memilih (*select*) data mahasiswa, mata kuliah, SKKM, absensi, dan yang lainnya. Perintah tersebut akan di-*export* melalui perintah *module.exports* yang disediakan oleh *Node JS* ke dalam *file controller*.



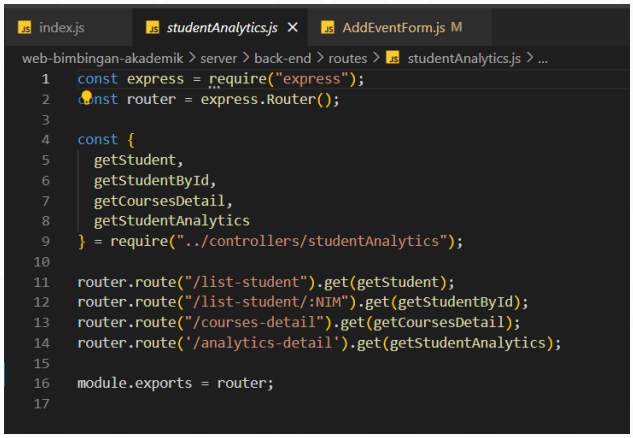
Gambar 3. 50 *Code GET Method* Fitur *Student Analytics*

Gambar 3.50 menunjukkan *code GET Method* agar data mahasiswa dapat muncul sesuai dengan NIM mahasiswa. Sebagian besar pengembangan *API* di fitur ini hampir sama seperti di fitur *Calendar* yang sebelumnya sudah dibuat. Pertama, dilakukan *getConnection* agar terhubung dengan *database*, kemudian menjadikan NIM sebagai *bind parameter*, dilanjutkan dengan eksekusi *query SQL* terkait dan *bind parameter* melalui perintah *conn.execute*. Setelah itu, koneksi database ditutup agar tidak memakan banyak *resource* seperti *memory* atau *RAM*. Setelah berhasil dieksekusi, data-data tersebut akan dikirim ke bagian *front-end*.



Gambar 3. 51 *Code GET Method* Bagian *Analytics* Mahasiswa

Gambar 3.51 adalah tampilan dari *code* fungsi *getStudentAnalytics* untuk mendapatkan data-data seperti SKS, *courses*, *terms*, SKKM, hingga absensi mahasiswa. *Method* yang digunakan adalah *GET request* sama seperti sebelumnya, tetapi pada fungsi *API* ini, eksekusi *query* untuk beberapa tabel *database* dijalankan bersamaan dan kemudian di-*merge*. Penyatuan data-data tersebut dilakukan untuk mempersingkat pengerjaan *API* karena terdapat banyak sekali perintah *SQL* yang berbeda-beda.

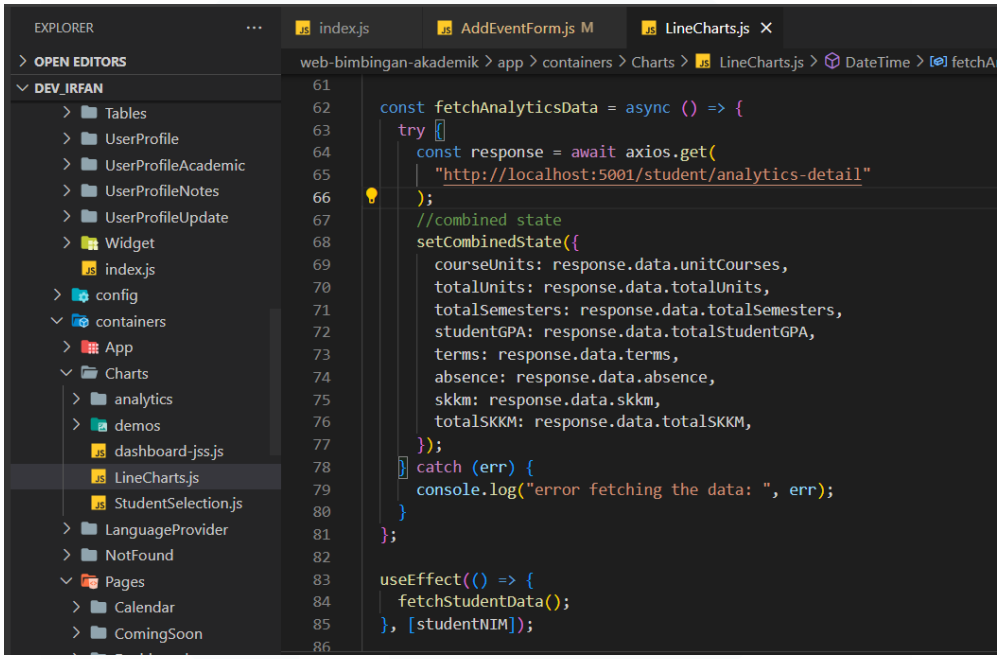


Gambar 3. 52 *Code Routes* Fitur *Student Analytics*

Gambar 3.52 adalah *code* untuk menentukan *URL* sebagai *API Endpoint* fitur *Student Analytics*. Beberapa *URL* tersebut digunakan untuk menampilkan data-data mahasiswa berdasarkan NIM, kemudian ada juga untuk menampilkan data-data mengenai *courses*, hingga data-data keseluruhan *analytics* yang terdiri dari SKS, SKKM, IPK, IPS, absensi dan yang lainnya.

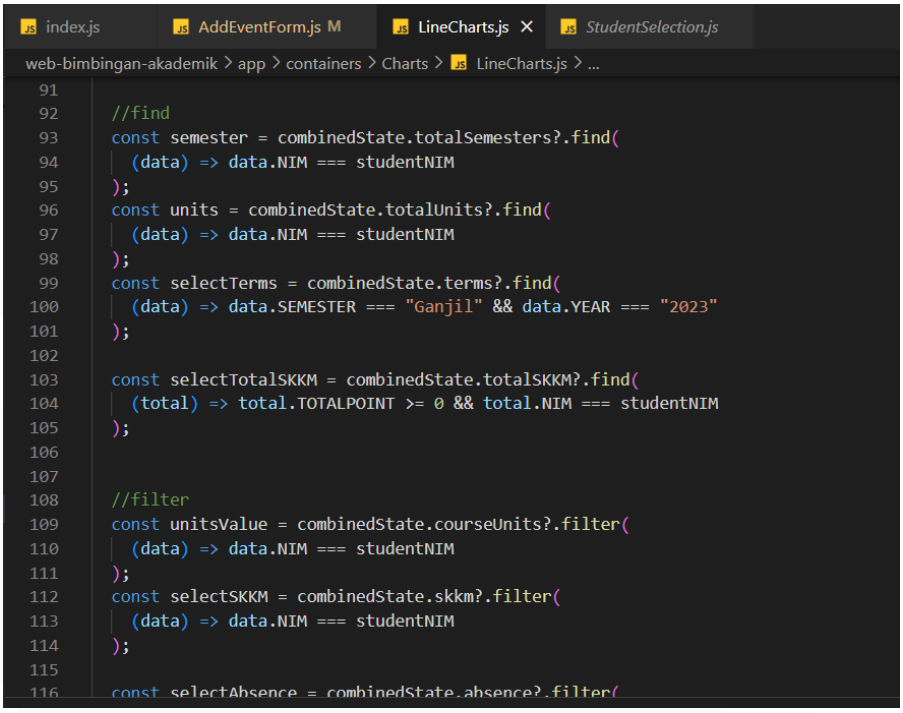
Setelah mahasiswa magang mengembangkan *API* fitur *Student Analytics*, nantinya akan melanjutkan ke implementasi *API* di bagian *front-end* fitur tersebut. Namun, sebelumnya, mahasiswa magang akan melakukan *meeting* melalui *platform Zoom* dengan *product owner* dan juga tim IT. *Meeting* yang dilaksanakan pada tanggal 30 Oktober 2023 membahas mengenai *update progress* pengerjaan, *feedback* untuk masing-masing fitur yang dikerjakan, dan *demo* beberapa fitur tersebut. *Meeting* diakhiri dimana mahasiswa magang ditugaskan untuk melanjutkan ke tahapan implementasi *API* fitur *Student Analytics* dan mengerjakan beberapa konfigurasi *minor* berdasarkan *feedback* hasil *meeting*.

3.2.6.2 Implementasi *API, Testing* fitur dan konfigurasi fitur *Student Analytics*



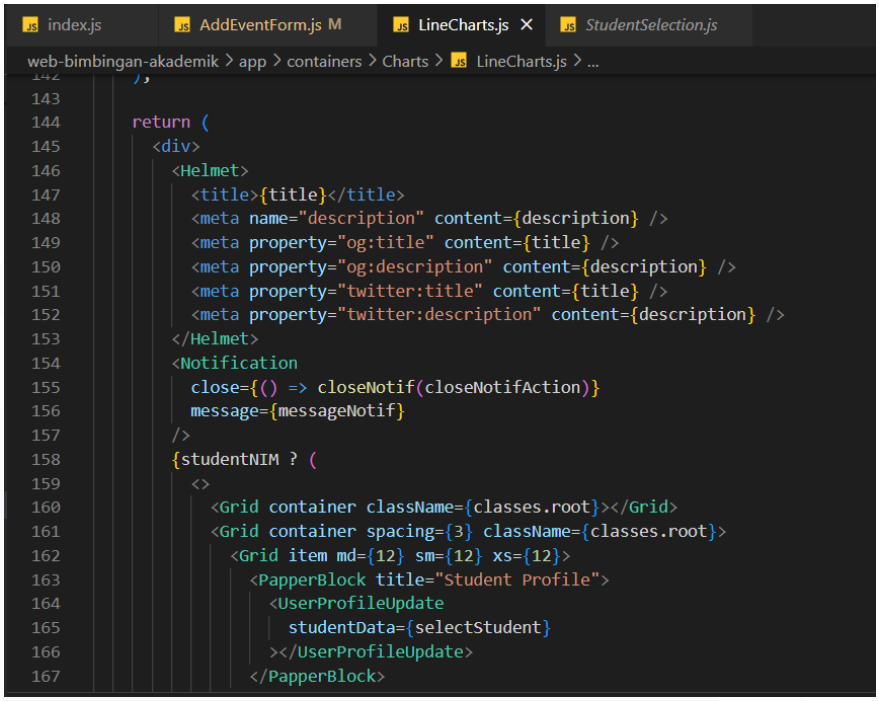
Gambar 3. 53 *Code* Utama Fitur *Student Analytics*

Gambar 3.53 merupakan tampilan *code* yang menampung seluruh komponen pada fitur *Student Analytics*. Terdapat fungsi *fetchAnalyticsData* yang berfungsi untuk menampilkan keseluruhan data yang di-*merge* pada *API* yang dibuat sebelumnya. Fungsi tersebut dijalankan melalui bantuan *useEffect* agar data-data di-*render* pada saat terjadi perubahan di NIM mahasiswa. *File code* ini terdapat pada *folder* bernama *Charts* yang berfungsi untuk menampung komponen atau *code* lainnya berkaitan dengan fitur *Student Analytics*.



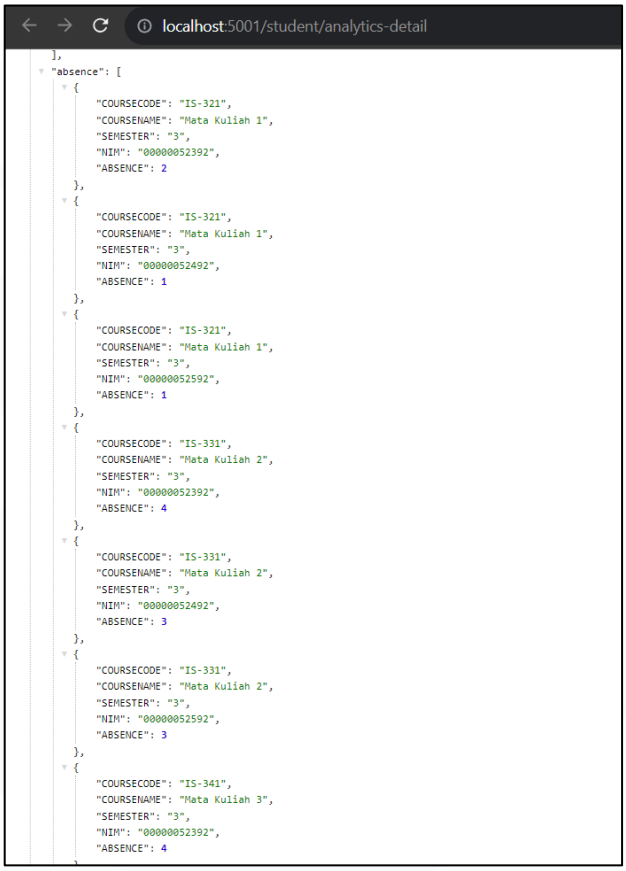
Gambar 3. 54 *Code* Untuk *Filter* Data Sesuai NIM

Gambar 3.54 adalah *code* untuk menyaring atau *filter* data-data yang diambil dari *API Endpoint* fitur *Student Analytics* agar sesuai dengan NIM mahasiswa. Tujuan dilakukan *filter* tersebut adalah karena fitur *Student Analytics* bertujuan untuk menampilkan performa akademik mahasiswa yang dipilih berdasarkan NIM. Oleh karena itu, data mahasiswa yang ditampilkan hanya satu dan diidentifikasikan berdasarkan NIM mahasiswa tersebut. Terdapat penggunaan *useParams* yang berasal dari *React Router* agar *URL* fitur *Student Analytics* memiliki parameter NIM dan data-data tersebut dapat di-*filter* sesuai dengan NIM mahasiswa.



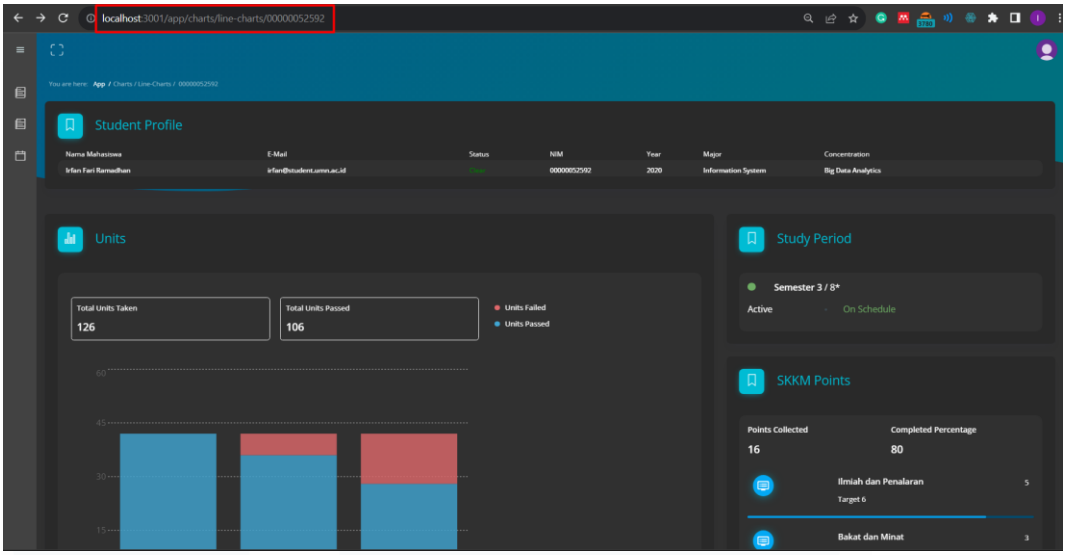
Gambar 3. 55 *Code Render* Fitur *Student Analytics*

Gambar 3.55 adalah tampilan dari *code* untuk melakukan *render* komponen-komponen terkait fitur *Student Analytics*. Komponen tersebut akan menerima data yang berasal dari *file* utama ini dengan menggunakan *props*. Setelah itu, data-data tersebut dapat muncul di masing-masing komponen, seperti *Student Profile*, *Units*, *SKKM Points*, *GPA*, dan yang lainnya.



Gambar 3. 56 *Snippet* Data *Analytics* Mahasiswa

Gambar 3.56 adalah potongan data untuk fitur *Student Analytics* dalam melihat performa akademik mahasiswa melalui *API Endpoint*. Data tersebut dapat ditampilkan di fitur *Student Analytics* dengan menggunakan perintah *axios.get* yang akan melakukan *GET Request* ke *API* terkait fitur tersebut.

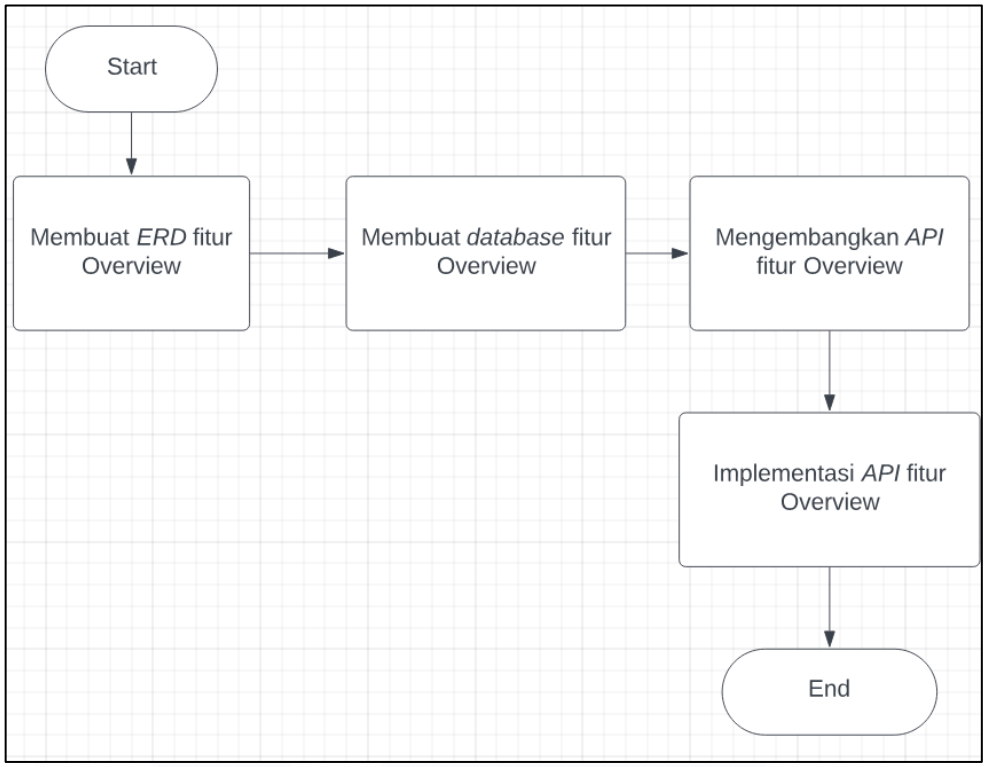


Gambar 3. 57 Tampilan Akhir Fitur *Student Analytics*

Gambar 3.57 adalah tampilan akhir untuk fitur *Student Analytics* dan dilakukan *testing* dengan cara melihat apakah keseluruhan data tersebut muncul di masing-masing komponen atau tidak. Kotak merah pada gambar menunjukkan bahwa *URL* fitur tersebut sudah ditambahkan dengan parameter NIM agar data-data yang muncul sesuai dengan NIM mahasiswa tersebut. Nantinya, fitur ini dapat diakses pada fitur *Overview* karena terdapat tombol untuk mengakses *Student Analytics*.

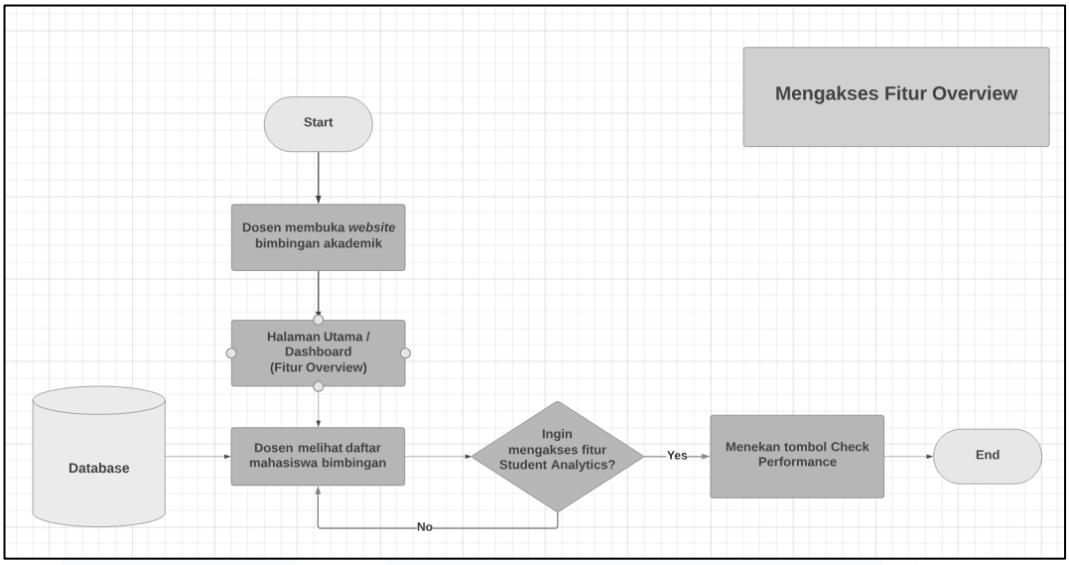
3.2.7 Menerapkan *API* pada fitur *Overview* (Minggu 9 – 10)

Setelah mahasiswa magang mengembangkan *API* dan implementasi *API* tersebut di fitur *Student Analytics*, maka fitur yang akan dikerjakan selanjutnya adalah fitur *Overview*. Sama seperti fitur sebelumnya, fitur *Overview* hanya berisikan *GET Request* untuk mendapatkan data-data mahasiswa bimbingan dan menampilkannya di komponen-komponen fitur tersebut.



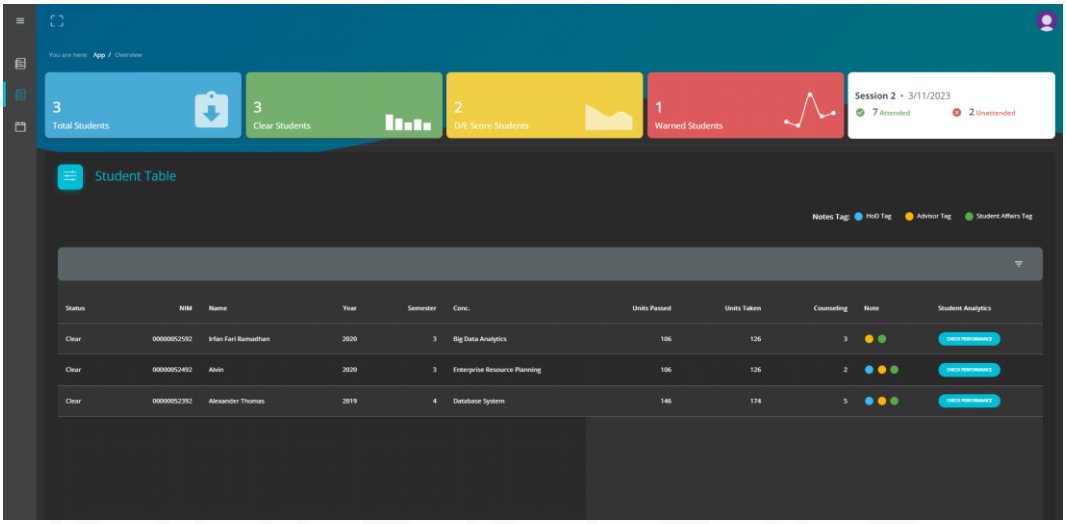
Gambar 3. 58 Proses Pengembangan *API* Fitur *Overview*

Gambar 3.58 adalah proses dari pengembangan dan penerapan *API* untuk fitur *Overview*. Seperti proses pada fitur sebelumnya, proses pertama dimulai dari membuat *ERD* dan dilanjutkan dengan pembuatan tabel *database* untuk fitur *Overview* sesuai dengan struktur *ERD* dan penyesuaian berdasarkan bagian *front-end*. Setelah itu, proses berlanjut ke pengembangan fungsi *API* agar data mahasiswa di fitur *Overview* dapat muncul. Setelah *API* berhasil dibuat, maka proses terakhir adalah implementasi *API* tersebut dengan menghubungkannya pada bagian *front-end*.



Gambar 3. 59 *User Flow* Fitur *Overview*

Gambar 3.59 adalah gambaran secara umum mengenai *user flow* pada fitur *Overview*. Fitur *Overview* sendiri merupakan tampilan utama ketika *user* atau dosen membuka *website* bimbingan akademik. Kemudian, ketika dosen sudah membuka halaman *Overview*, maka akan muncul tampilan berupa daftar mahasiswa bimbingan beserta informasi singkat mengenai mahasiswa tersebut. Fitur ini juga terhubung dengan fitur *Student Analytics* melalui tombol "*Check Performance*" dan jika dosen ingin melihat performa akademik salah satu mahasiswa, maka dapat menekan tombol tersebut.

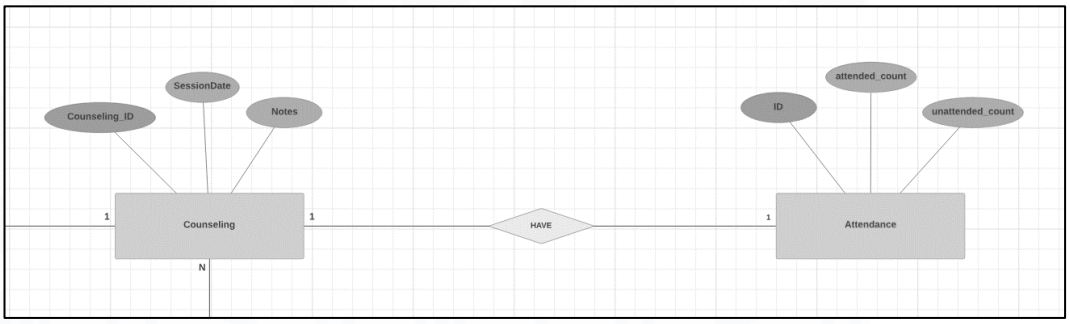


Gambar 3. 60 Halaman Utama Fitur *Overview*

Gambar 3.60 adalah tampilan dari halaman utama untuk fitur *Overview*. Pada fitur tersebut terdapat komponen *widget* di bagian atas yang menunjukkan jumlah mahasiswa bimbingan, status mahasiswa, jumlah mahasiswa yang mendapat nilai D atau E, mahasiswa yang diberikan peringatan, dan juga menampilkan sesi bimbingan akademik. Kemudian, terdapat komponen *Student Table* yang digunakan untuk menampung data-data mahasiswa dan menampilkan informasi berupa NIM, nama mahasiswa, status mahasiswa, angkatan, dan yang lainnya. Lalu, terdapat tombol pada kolom bernama *Student Analytics* yang akan melakukan *redirect user* atau dosen untuk melihat performa akademik mahasiswa yang dipilih tersebut. Selain itu, nantinya terdapat integrasi dengan fitur yang akan dibuat setelah fitur ini, yaitu fitur *Student Notes*. Sebelum dapat menerapkan *API* untuk fitur *Overview* dan menghubungkannya, mahasiswa magang melakukan pembuatan *ERD* dan juga tabel di *database* terlebih dahulu sama seperti pekerjaan-pekerjaan sebelumnya.

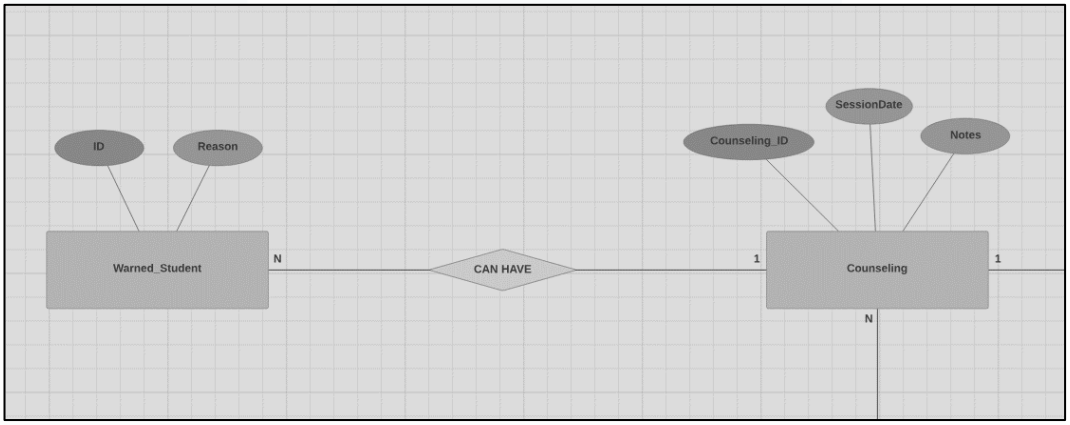
3.2.7.1 Membuat *ERD* dan juga *Database* untuk fitur *Overview*

Tahapan pertama sama seperti pengerjaan fitur-fitur sebelumnya, yaitu membuat *ERD* dan kemudian membuat tabel untuk keperluan fitur di *database*. Dalam hal ini, seperti yang sebelumnya sempat dijelaskan, untuk *ERD* fitur *Overview* sendiri terhubung dengan *ERD* pada fitur *Student Analytics*. Namun, penjelasan *ERD* akan dimulai dari objek atau *entity Student* dan juga beberapa objek lainnya.



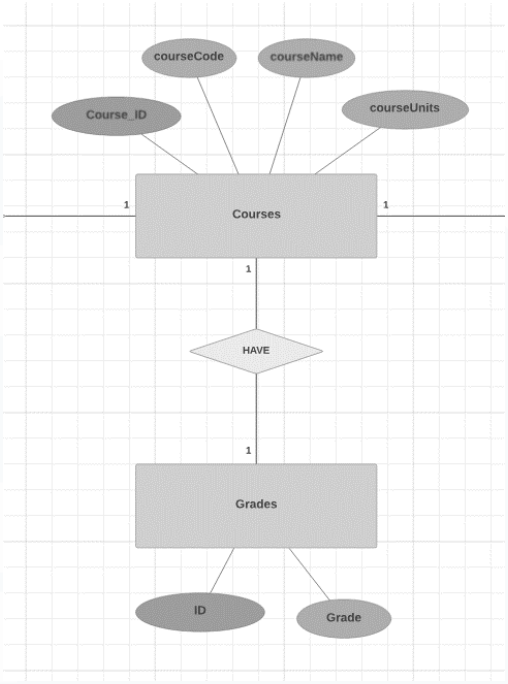
Gambar 3. 61 *ERD* Fitur *Overview* Bagian *Counseling*

Gambar 3.61 menampilkan *ERD* fitur *Overview* dimana terdapat objek *Counseling* dan juga *Attendance*. Objek *Counseling* terhubung dengan *Student* dan hubungan pada kedua objek tersebut adalah *many-to-many* karena semua mahasiswa dapat memiliki banyak jadwal atau sesi bimbingan akademik. Kemudian, dari jadwal bimbingan pastinya mempunyai jumlah mahasiswa yang hadir pada bimbingan tersebut, sehingga objek *Counseling* akan terhubung dengan *Attendance* dan memiliki hubungan *one-to-one* karena setiap sesi bimbingan mempunyai satu *record* jumlah mahasiswa yang hadir tersebut.



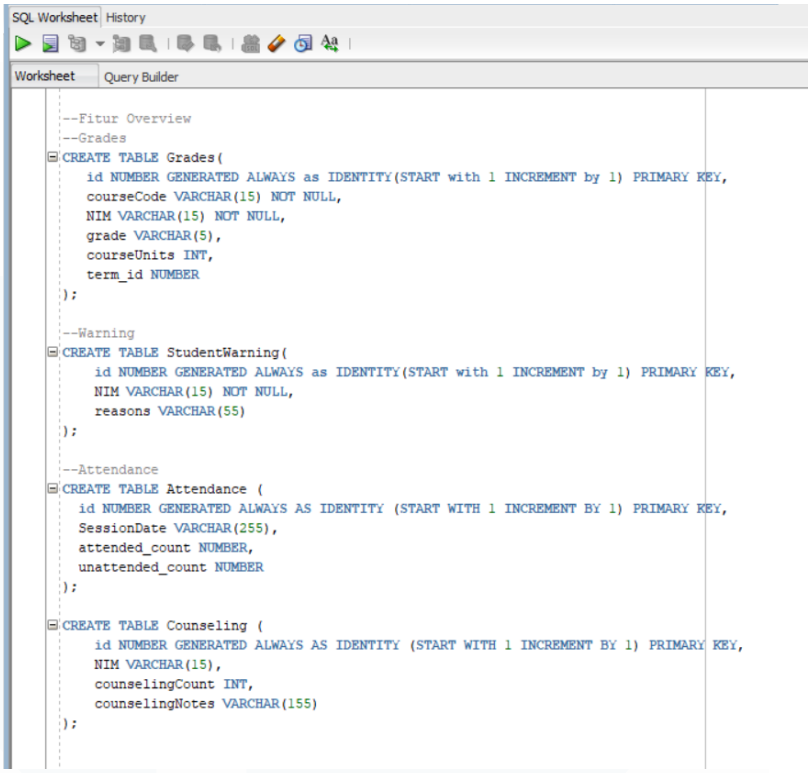
Gambar 3. 62 *ERD* Fitur *Overview* Bagian *Warned Student*

Gambar 3.62 adalah *ERD* yang menampilkan objek lanjutan dari hubungan antara *Student* dengan *Counseling*. Objek *Counseling* akan terhubung juga dengan objek *Warned Student* untuk menunjukkan mahasiswa yang diberikan peringatan dan juga alasan dari peringatan tersebut. Hubungan antara kedua objek tersebut adalah *one-to-many* karena pada setiap sesi bimbingan bisa terdapat banyak peringatan yang diberikan dari dosen kepada mahasiswa bimbingannya.



Gambar 3. 63 *ERD* Fitur *Overview* Bagian *Grades*

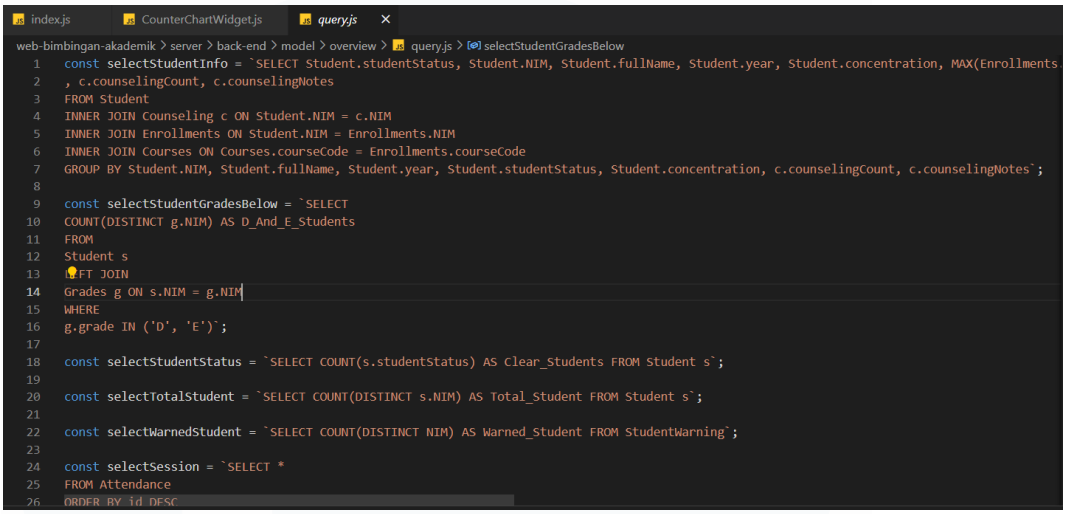
Gambar 3.63 adalah *ERD* yang menunjukkan objek *Courses* dan juga objek *Grades*. Sebelumnya, objek *Courses* memiliki hubungan dengan *Students* dengan perantara *Enrollments* untuk menunjukkan mata kuliah yang dipilih pada semester tertentu. Pada fitur *Overview*, tabel *Courses* terhubung dengan objek *Grades* dengan hubungan *one-to-one* karena setiap mata kuliah yang mahasiswa ambil, hanya memiliki satu *record* nilai untuk mata kuliah tersebut. Setelah *ERD* untuk fitur *Overview* dibuat, maka mahasiswa magang selanjutnya akan membuat tabel *database* agar *API* dapat menggunakan tabel tersebut untuk keperluan fitur.



Gambar 3. 64 Tabel *Database* Fitur *Overview*

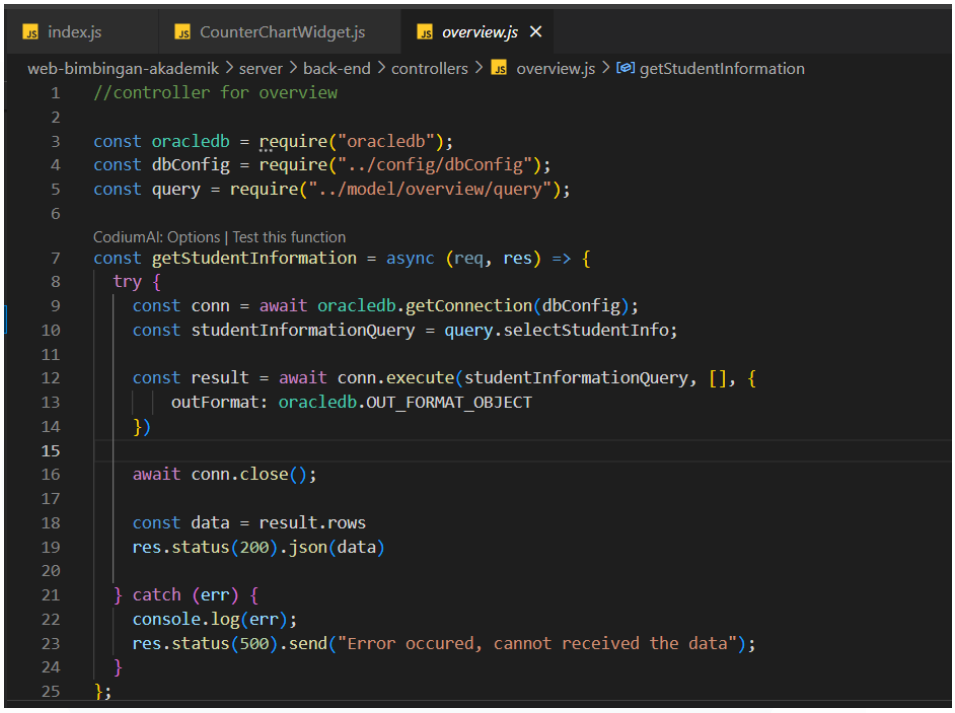
Gambar 3.64 menunjukkan perintah *SQL* berupa *create table* seperti *Grades*, *StudentWarning*, *Attendance* dan *Counseling*. Masing-masing tabel memiliki *id* sebagai *primary key* dan nantinya tabel-tabel tersebut dihubungkan pada *select statement* untuk penerapan *API* dengan NIM mahasiswa. Atribut pada tabel tersebut juga disesuaikan untuk mempermudah implementasi *API* di bagian *front-end*.

3.2.7.2 Membuat *API* dan menghubungkannya ke fitur *Overview*



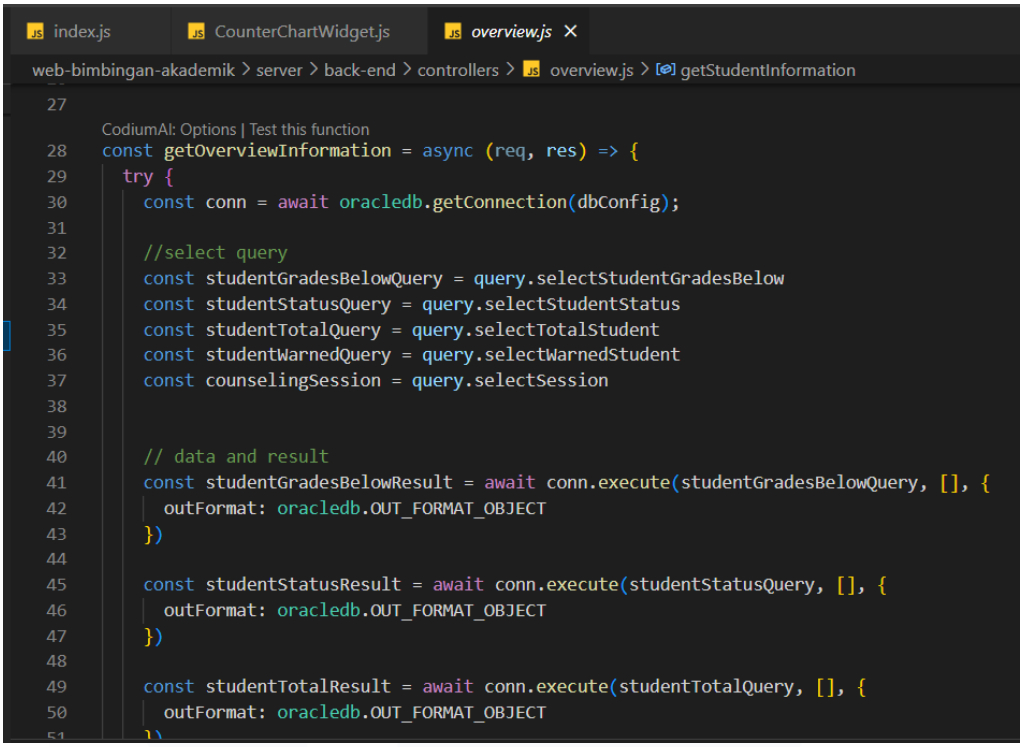
Gambar 3. 65 Perintah *SQL* Untuk *API* Fitur *Overview*

Gambar 3.65 adalah *code query* untuk keperluan *API* fitur *Overview* yang semua perintah merupakan *select statement* karena fitur tersebut memiliki kegunaan dalam menampilkan data-data mahasiswa bimbingan. Beberapa perintah *SQL* seperti *selectStudentInfo* untuk menampilkan data di komponen *Student Table* fitur *Overview*, serta perintah yang lainnya di-*export* melalui *module.exports* dan digunakan pada *file API* fitur *Overview*.



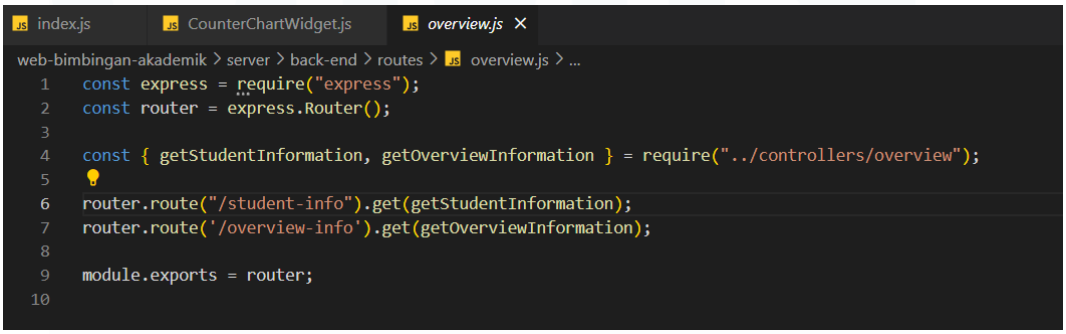
Gambar 3. 66 *Code GET Method Student Information* Fitur *Overview*

Gambar 3.66 merupakan *file code controller* untuk menampung fungsi *API* fitur *Overview*. Fungsi *API* yang ada pada Gambar 3.60 merupakan *API* untuk melakukan *GET Request* atau *GET Method* dalam menampilkan data-data berupa informasi mahasiswa. Untuk perintah *SQL*-nya sendiri didapatkan dengan menggunakan variabel pada *file query* yang dijelaskan sebelumnya. Kemudian, *query* tersebut dieksekusi ke dalam *database* melalui *conn.execute* dan data tersebut dikirim sebagai respon dengan adanya *API Endpoint*. Pada bagian *front-end* fitur *Overview*, *API Endpoint* tersebut diakses menggunakan *axios.get* agar data dapat dihubungkan ke komponen-komponen fitur tersebut.



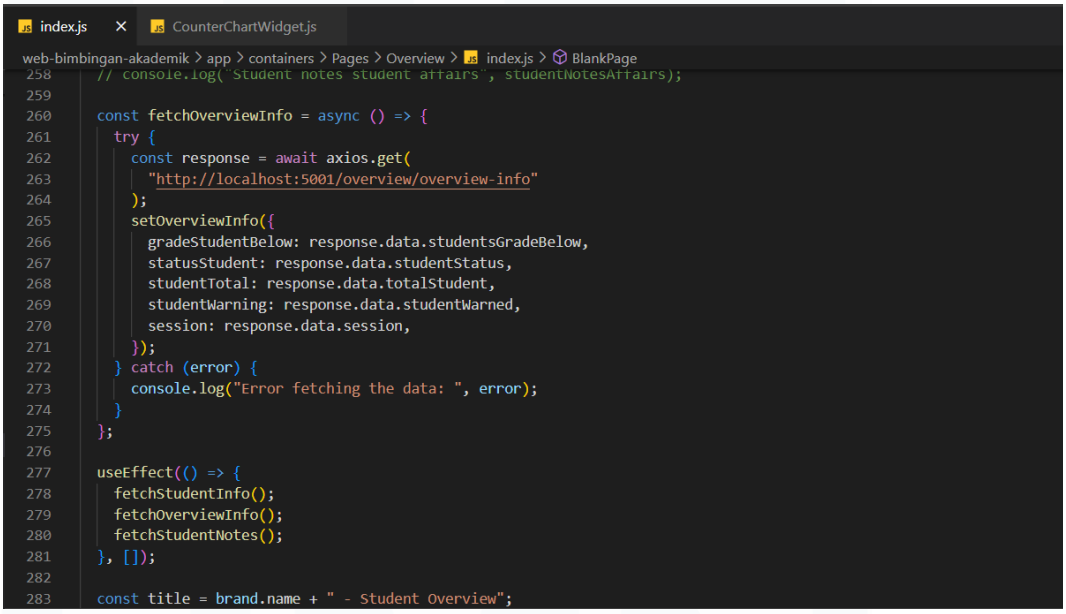
Gambar 3. 67 *Code GET Method Overview Information*

Gambar 3.67 merupakan tampilan *code* fungsi *API* untuk menampilkan informasi lainnya pada fitur *Overview*, terutama untuk dikirim ke komponen *widget* di fitur tersebut. Perintah *SQL* yang ada di fungsi tersebut dieksekusi bersamaan dan akan di-*merge* sehingga data-data tersebut muncul bersamaan jika dipanggil menggunakan *GET Request* oleh *axios.get* pada bagian *front-end*. Tujuan *merge* data-data tersebut adalah agar *API Endpoint* tidak terlalu banyak dan mempersingkat waktu pengerjaan.



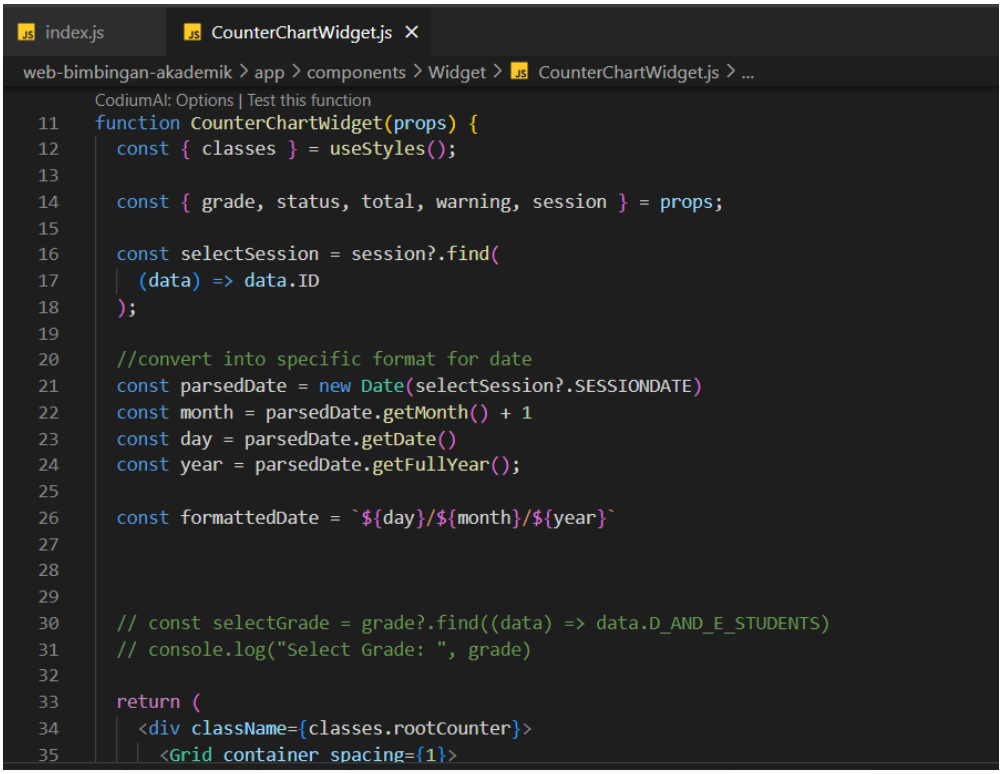
Gambar 3. 68 *Code Routes* Fitur *Overview*

Gambar 3.68 menunjukkan *code* yang digunakan dalam membuat *API Endpoint* atau *URL API* untuk fitur *Overview*. Untuk *route* atau *URL* dengan nama *student-info* digunakan sebagai *API Endpoint* yang menampung data-data mahasiswa bimbingan seperti nama, NIM, status, angkatan, dan yang lainnya. Sedangkan, *route* dengan nama *overview-info* digunakan sebagai data-data informasi *Overview* untuk komponen *widget* fitur *Overview*. Setelahnya, *code route* akan di-*export* dan digunakan di bagian *file index.js* atau *file* utama *back-end* agar *API Endpoint* dapat diakses.



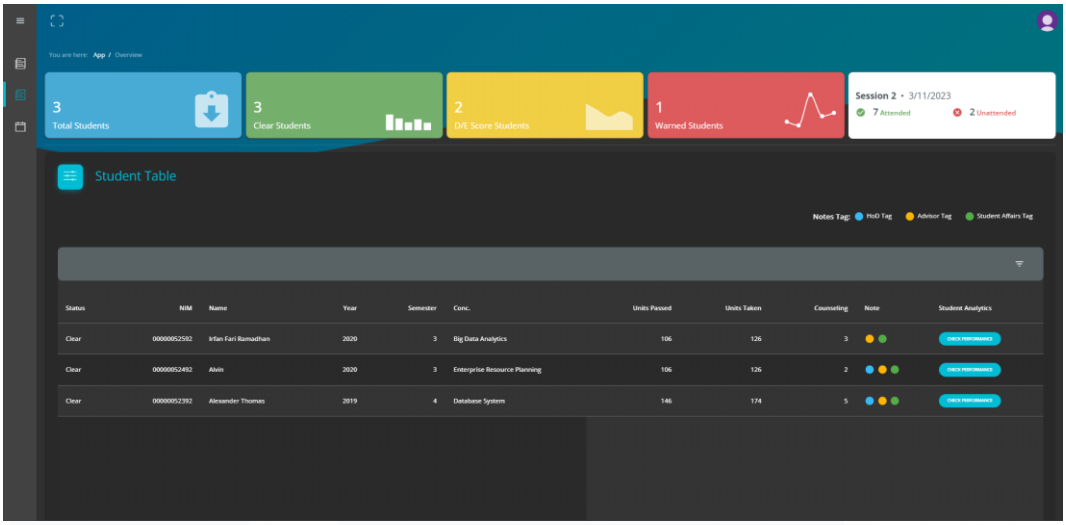
Gambar 3. 69 *Code* Utama Fitur *Overview*

Gambar 3.69 merupakan *code* utama untuk fitur *Overview*. Dikarenakan *code* fitur tersebut cukup panjang, maka tampilan pada Gambar 3.63 adalah berupa fungsi *fetchOverviewInfo* dalam menampilkan data-data *Overview information* melalui bantuan *axios.get* untuk dapat mengambil data dari *API Endpoint* yang diberikan tersebut. Data yang diambil dari *API Overview information* akan dikirim melalui *props* ke komponen *widget*. Sedangkan, untuk data-data mahasiswa bimbingan akan digunakan pada tabel di fitur *Overview* tersebut. Selain itu, pada tabel nantinya terdapat tombol agar dosen dapat mengakses fitur *Student Analytics*.



Gambar 3. 70 *Code Widget* Fitur *Overview*

Gambar 3.70 adalah *file code* untuk komponen *widget* yang menampilkan data-data *Overview information*. Data yang telah dikirim dari *file* utama *Overview* tersebut disesuaikan sehingga dapat ditampilkan. Jika data-data tersebut berhasil didapatkan, maka *API Endpoint* sudah selesai diimplementasikan di fitur *Overview* dan bekerja secara fungsional.



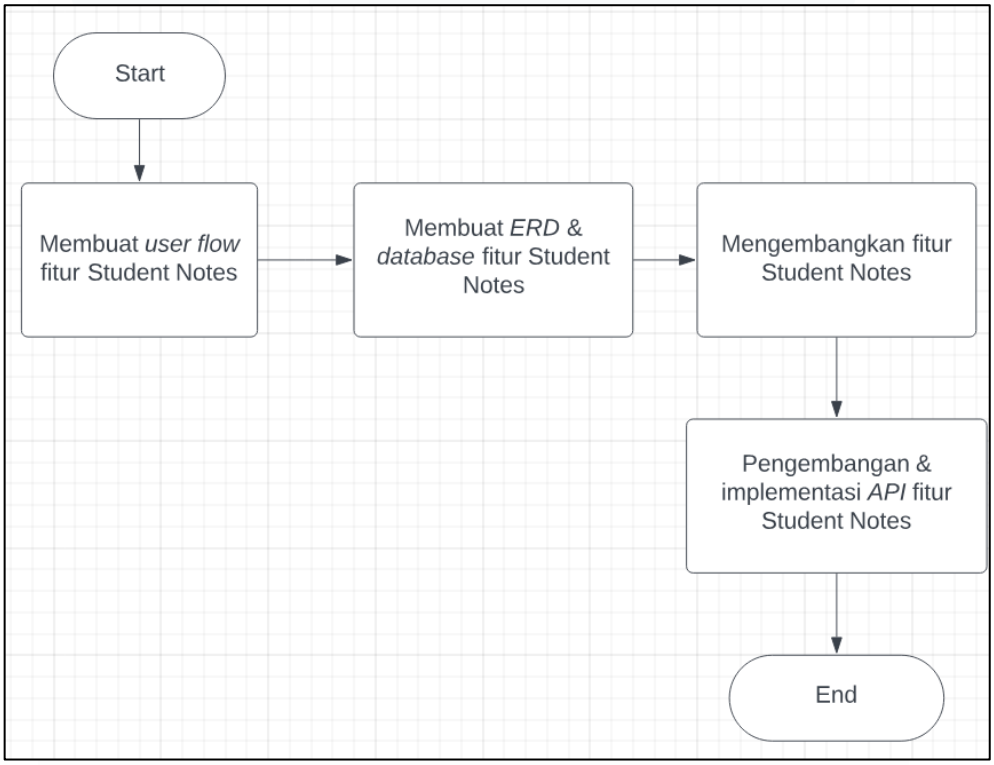
Gambar 3. 71 Tampilan Akhir Fitur *Overview*

Gambar 3.71 menampilkan tampilan akhir ketika *API* berhasil diimplementasikan pada fitur *Overview* tersebut. Pada bagian atas *Student Table* merupakan komponen *widget* dan sudah muncul data-data yang berhasil dikirim melalui *props* dari *file* utama ke *file widget*. Kemudian, pada komponen *Student Table* data-data mahasiswa bimbingan juga sudah berhasil ditampilkan, serta terdapat tombol yang ditandai warna biru untuk *redirect* dosen ke halaman atau fitur *Student Analytics*. Selain itu, pada aktivitas kerja magang selanjutnya adalah membuat fitur *Student Notes* dan pada *Student Table* di fitur *Overview* tersebut terdapat kolom *notes* untuk menampilkan data-data dari fitur *Student Notes*.

3.2.8 Mengembangkan fitur *Student Notes* dan *API*-nya (Minggu 11 – 12)

Kegiatan kerja magang selanjutnya adalah mengerjakan fitur yang terakhir, yaitu fitur *Student Notes*. Fitur tersebut dikerjakan berdasarkan *request* dari salah satu *product owner*, yaitu Ibu Fonita Theresia Yoliando, S.Ds., M.A. melalui *email*. Kemudian mahasiswa magang berdiskusi dan memberikan ide terhadap penerapan fitur tersebut dan hasilnya adalah fitur tersebut akan ditempatkan di dalam fitur *Student Academics*. Fitur *Student Notes* ditempatkan

di bagian bawah komponen *student profile*, dimana komponen tersebut termasuk komponen teratas sehingga ketika dosen masuk ke fitur *Student Academics*, maka *Student Notes* dapat langsung terlihat. Selanjutnya, kegunaan fitur tersebut adalah untuk menampilkan, menambah, dan memperbarui catatan terhadap mahasiswa bimbingan yang dipilih. Catatan terhadap mahasiswa tersebut dilakukan oleh dosen (*advisor*), kaprodi (*HoD*), dan juga kemahasiswaan (*student affairs*). Fitur *Student Notes* juga merupakan fitur baru dan karena itu, mahasiswa magang akan membuat *user-flow*, *ERD*, dan juga *database* terlebih dahulu. Selanjutnya, *API* untuk fitur tersebut akan dibuat dan juga diterapkan pada beberapa fitur lainnya, seperti *Overview* untuk menampilkan *notes* di tabel mahasiswa.

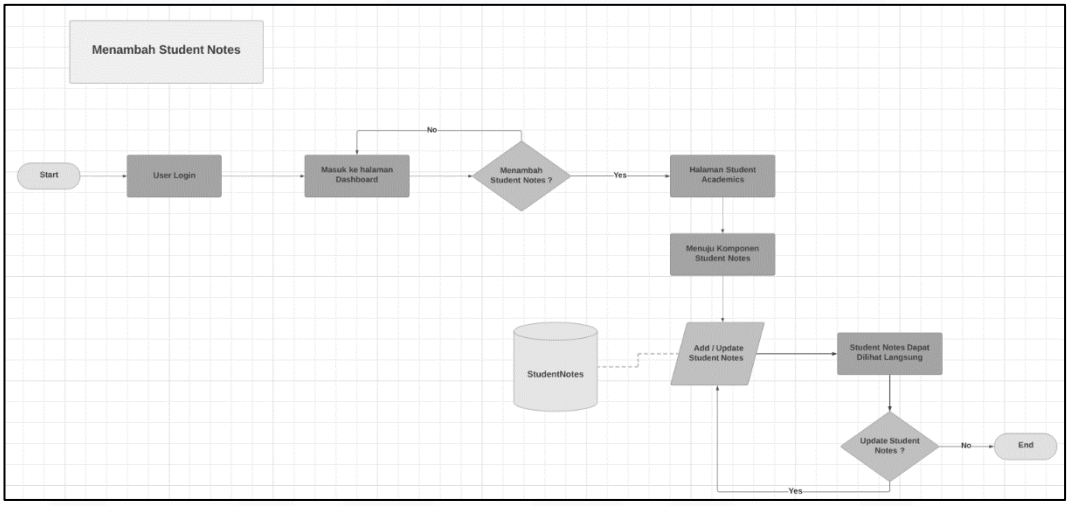


Gambar 3. 72 Proses Pengembangan Fitur *Student Notes*

Gambar 3.72 adalah gambaran secara umum mengenai proses pembuatan fitur *Student Notes*. Proses tersebut dimulai dari membuat *user flow* untuk menuju ke fitur *Student Notes* dan *user* tersebut dapat menambah atau *update notes* pada mahasiswa. Kemudian, proses dilanjutkan dengan membuat *ERD*

dan juga tabel *database* untuk menampung data *notes* yang telah ditambahkan oleh *user*. Setelah tabel *database* dibuat, proses dilanjutkan dengan mengembangkan bagian *front-end* dari fitur *Student Notes* berdasarkan *feedback* yang diberikan oleh *product owner* ketika melakukan *meeting*. Setelah *product owner* sepakat dengan bentuk atau tampilan fitur *Student Notes*, proses dilanjutkan dengan pengembangan *API* dan implementasi pada fitur *Student Notes* tersebut.

3.2.8.1 Membuat *user flow*, *ERD*, dan juga *Database* untuk fitur *Student Notes*



Gambar 3. 73 *User Flow* Fitur *Student Notes*

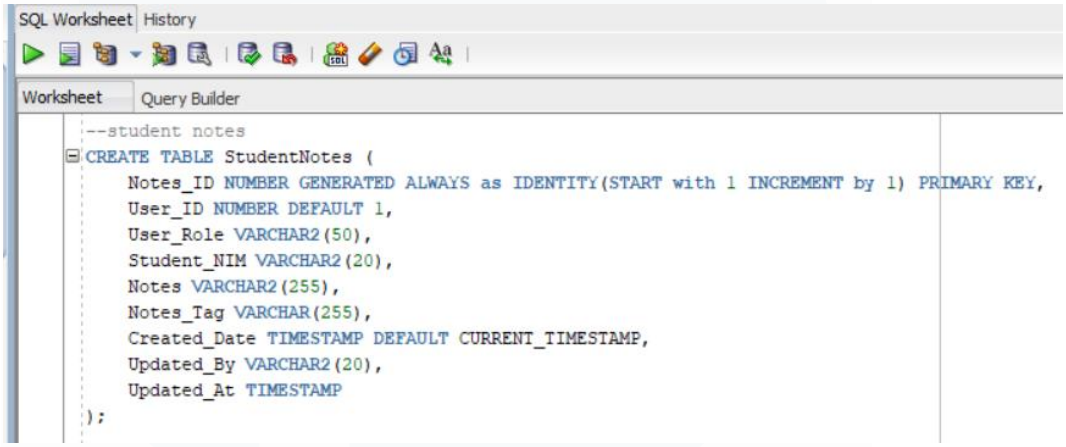
Gambar 3.73 adalah *user flow* untuk dapat menggunakan fitur *Student Notes*. Beberapa alur seperti *login* masih berupa pengandaian sama seperti *user flow* di fitur *Calendar* karena fitur *authorization* masih belum diimplementasikan. Alur tersebut dimulai ketika *user* melakukan *login* agar mendapatkan *id* dan juga *role* untuk menentukan *user* tersebut merupakan dosen, kaprodi atau kemahasiswaan. Setelah itu, pada halaman utama atau *dashboard*, *user* akan menentukan untuk menambahkan *student notes* atau tidak, jika tidak maka akan tetap berada di halaman *dashboard*, dan jika iya maka akan dilanjutkan ke halaman atau fitur *Student Academics*. Setelah berada di fitur *Student Academics*, *user* akan ke bagian *Student Notes*.

Kemudian, *user* akan melakukan *add* atau *update notes* dimana proses tersebut akan terhubung ke *database Student Notes*. Setelah menambahkan *notes* atau memperbaruinya, *user* akan langsung dapat melihat *notes* tersebut. Jika *user* akan melakukan *update notes*, maka proses tersebut diulang ke bagian menambah atau memperbarui *student notes*, jika tidak maka proses berakhir. Setelah *user flow* dibuat, mahasiswa magang melanjutkan ke pembuatan *ERD*.

Gambar 3. 74 *ERD* Fitur *Student Notes*

Gambar 3.74 adalah tampilan *ERD* untuk fitur *Student Notes*. *ERD* tersebut dibuat berdasarkan kesepakatan dan hasil diskusi dengan *product owner*. Setelah melakukan konfirmasi dengan *product owner*, mahasiswa magang membuat *ERD* tersebut. Secara garis besar, objek atau *entity* pada *ERD* fitur *Student Notes* adalah objek sebagai *role user* seperti *Lecturer*,

HoD, dan juga *Student Affairs*. Objek tersebut akan terhubung dengan objek *Student Notes* secara *many-to-many* karena masing-masing *user role* dapat menambahkan banyak *notes* kepada banyak mahasiswa juga termasuk mahasiswa bimbingan. Setelah *ERD* dibuat, maka selanjutnya mahasiswa magang akan membuat struktur tabel di *database* berdasarkan *ERD* tersebut untuk fitur *Student Notes*.



Gambar 3. 75 Tabel *Database* Fitur *Student Notes*

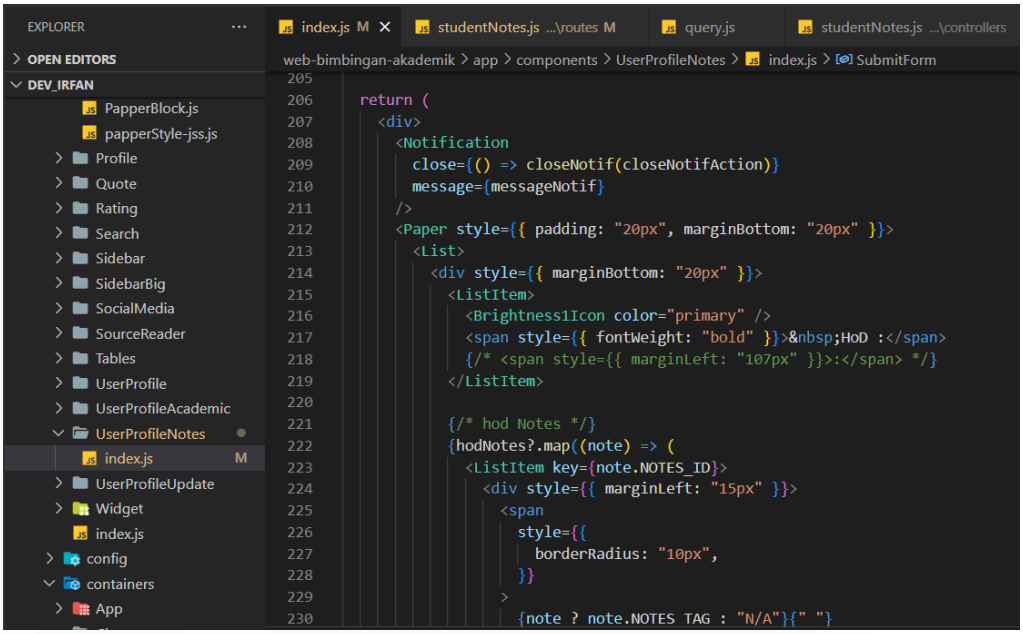
Gambar 3.75 adalah *SQL query* untuk *create table database* fitur *Student Notes*. Tabel tersebut dibuat berdasarkan dari objek *Student Notes* di *ERD* sebelumnya. Atribut-atribut tersebut terdiri dari *Notes_ID* sebagai *primary key*, kemudian *User_ID* dan *User_Role* untuk menentukan *user*, *Student_NIM* sebagai NIM mahasiswa, hingga terdapat atribut *Updated_At* untuk menyimpan tanggal *notes* ketika diperbarui. Selain itu, beberapa atribut akan disesuaikan dengan *request body* pada bagian *API* fitur *Student Notes* dalam mempermudah implementasi *API* di *front-end*. Setelah *user flow*, *ERD* dan *database* dibuat, mahasiswa magang akan melakukan *meeting* terlebih dahulu yang membahas *update* pengerjaan fitur, serta membahas fitur *Student Notes* tersebut.

Meeting dilaksanakan pada tanggal 20 November 2023 dan dihadiri oleh *product owner*. Secara keseluruhan, *meeting* tersebut membahas mengenai fitur-fitur apa saja yang sudah dikerjakan dan fitur tersebut ditampilkan

melalui *demo* fitur. Kemudian, mahasiswa magang dan *product owner* berdiskusi untuk membahas penerapan fitur *Student Notes*, seperti tampilan di bagian *front-end*, hingga penyesuaian *form* dengan atribut di *database*. Hasil dari *meeting* tersebut secara garis besar adalah dalam pengerjaan dan penerapan fitur *Student Notes*, mahasiswa magang dianjurkan untuk memperhatikan beberapa hal penting, seperti terdapat *tag notes* untuk menandai jenis dari *notes* yang di-*input*, dan *notes* dapat terlihat di halaman atau fitur *Overview* bagian komponen tabel mahasiswa.

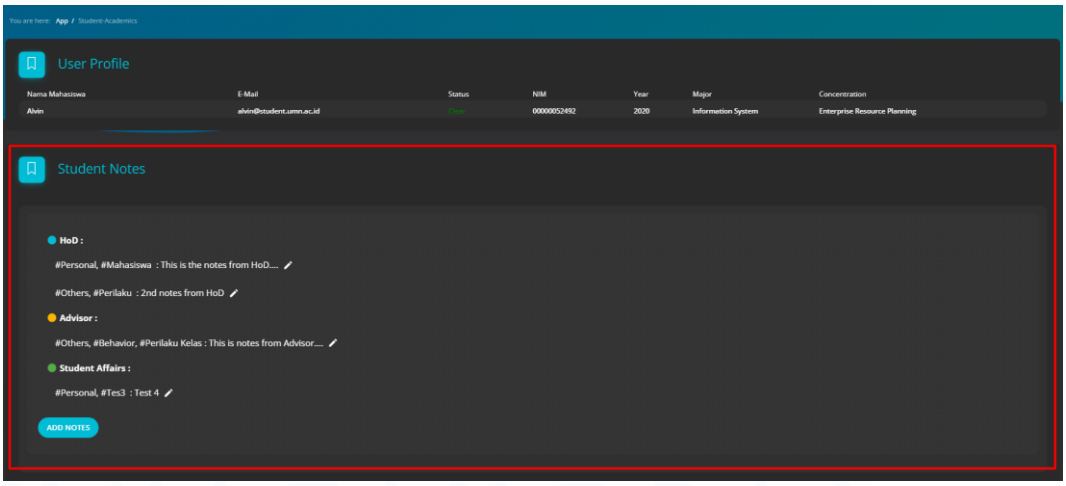
3.2.8.2 Membuat fitur *Student Notes* dan juga *API*-nya berdasarkan *feedback* hasil *meeting*

Tahapan atau proses selanjutnya dalam mengerjakan fitur *Student Notes*, mahasiswa magang membuat tampilan *front-end* dan juga *API* untuk dapat diterapkan pada fitur tersebut sebagai bagian dari implementasi *back-end*. Pada fitur *Student Notes*, terdapat *form* yang akan menggunakan *API* agar fungsional dengan *POST Method* dan *PUT Method* dalam menambah atau memperbarui *notes*. Selain itu, terdapat *GET Method* untuk dapat menampilkan *notes* secara langsung ketika *form* di-*submit*.



Gambar 3. 76 *Code* Fitur *Student Notes*

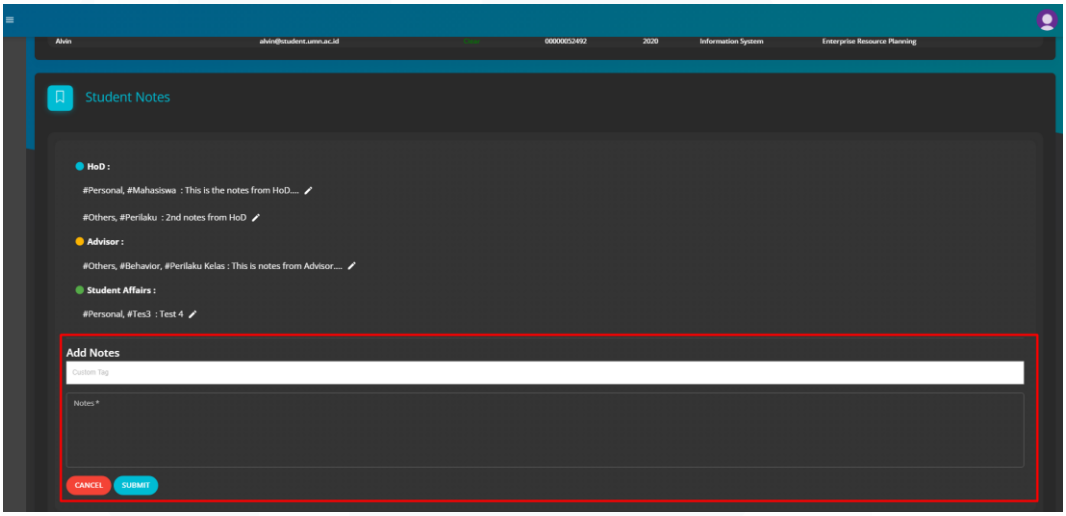
Gambar 3.76 adalah tampilan dari *code front-end* fitur *Student Notes*. *Code* tersebut disimpan pada *folder* dengan nama *UserProfileNotes* di *file index.js*. *File* ini akan dijadikan sebagai tampilan utama *Student Notes* di fitur *Student Academics*, dimana terdapat bagian yang menampilkan *notes* dan juga *form* untuk dapat menambah dan memperbarui *notes*.



Gambar 3. 77 Tampilan Fitur *Student Notes*

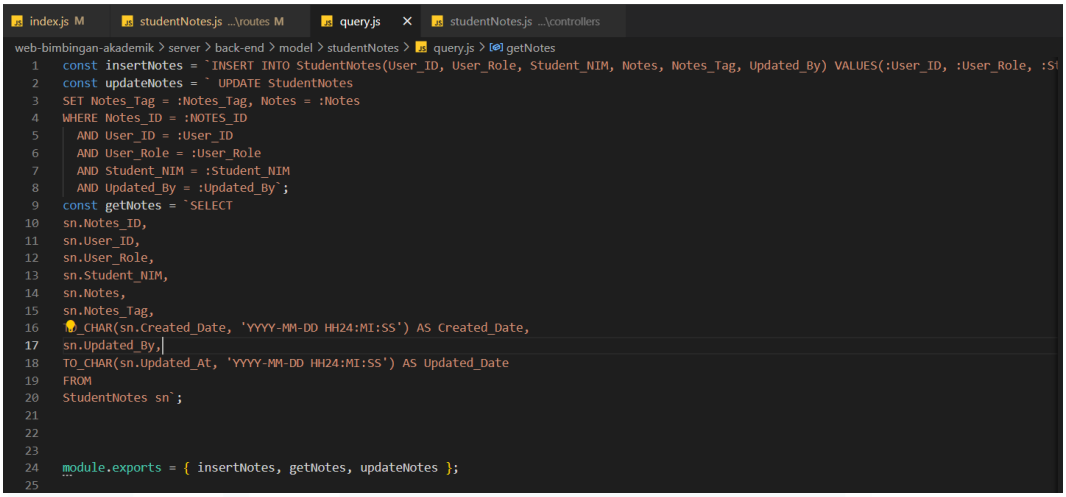
Gambar 3.77 adalah tampilan dari *Student Notes* yang terdapat di halaman *Student Academics*. Terdapat bagian yang menampilkan *notes*

sesuai dengan *user role*, seperti *HoD*, *Advisor*, dan juga *Student Affairs*. Kemudian, di bagian sisi kanan *notes* terdapat *icon* berbentuk pensil untuk melakukan *update notes*. Untuk dapat menambahkan *notes*, *user* dapat menekan tombol *Add Notes* dan nantinya akan muncul tampilan *form*.



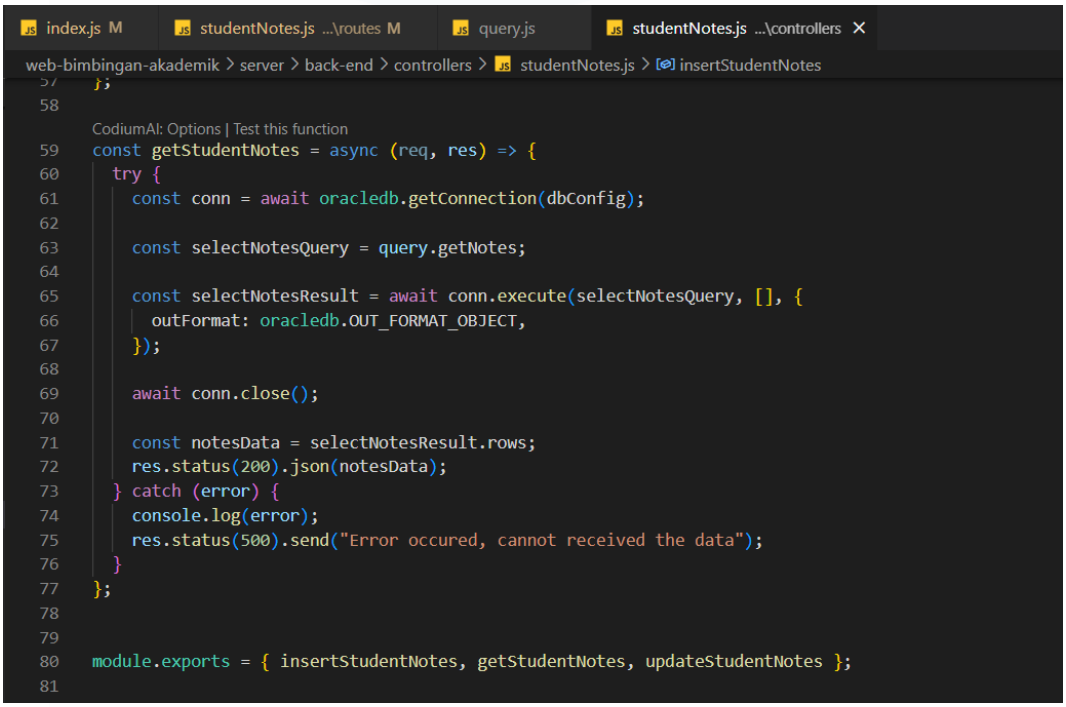
Gambar 3. 78 Tampilan *Form* Fitur *Student Notes*

Gambar 3.78 adalah tampilan ketika tombol *Add Notes* ditekan dan akan memunculkan *form* agar *user* dapat mengisi *notes tag* dan juga *notes*. Untuk tampilan *update* sendiri hampir sama seperti tampilan *Add Notes*, perbedaannya hanya di nama label saja, pada tampilan *update* bernama *Update Note*. Setelah tampilan *front-end* dibuat, mahasiswa magang akan membuat *API* dan menerapkannya pada fitur tersebut.



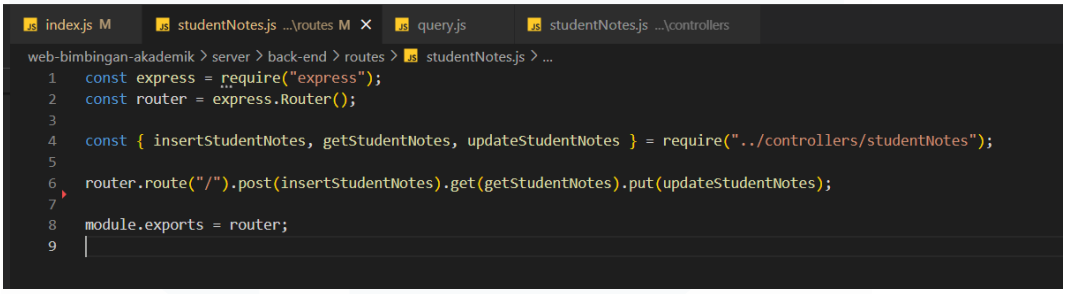
Gambar 3. 79 Perintah *SQL* Fitur *Student Notes*

Gambar 3.79 adalah *code query* yang menampung perintah *SQL* seperti *insert*, *select*, dan *update* untuk fitur *Student Notes*. Perintah tersebut digunakan di bagian fungsi *API* agar dapat melakukan *GET*, *POST*, dan *PUT Method* sehingga data-data *Student Notes* dapat diterima atau dikirim oleh *database*.



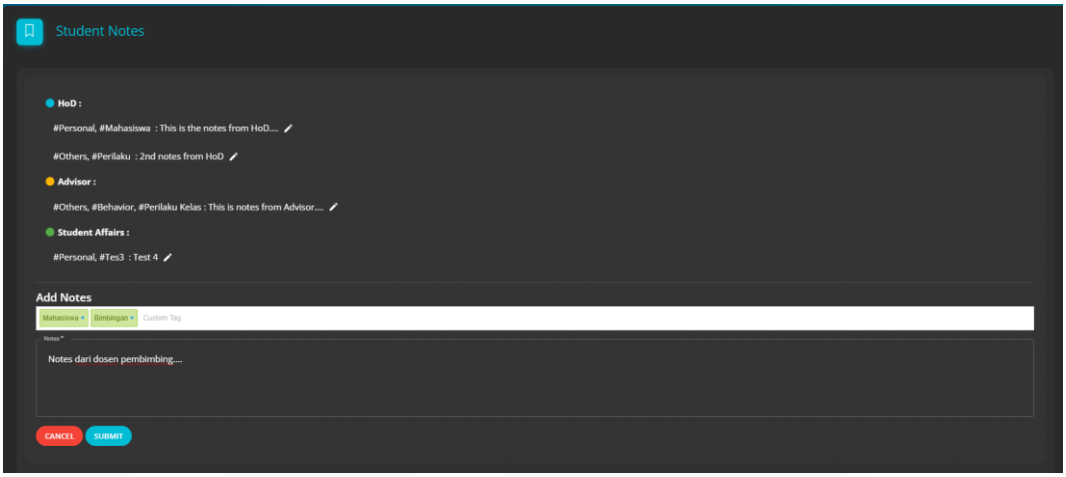
Gambar 3. 80 *Code Controller* Fitur *Student Notes*

Gambar 3.80 adalah tampilan *code* pada *file controller* untuk fitur *Student Notes* yang digunakan dalam menampung fungsi *API*. Fungsi-fungsi *API* yang ditampung adalah *insertStudentNotes* menggunakan *POST Method* agar data-data *form* dapat ditambah ke *database*. Kemudian fungsi *getStudentNotes* yang menggunakan *GET Method* agar data-data hasil *submit form* di *Student Notes* dapat ditampilkan. Terdapat juga fungsi *updateStudentNotes* untuk melakukan *update* data di *database* dengan menggunakan *PUT Method*. Setelah fungsi-fungsi tersebut dibuat maka akan di-*export* ke bagian *routes* fitur *Student Notes*.



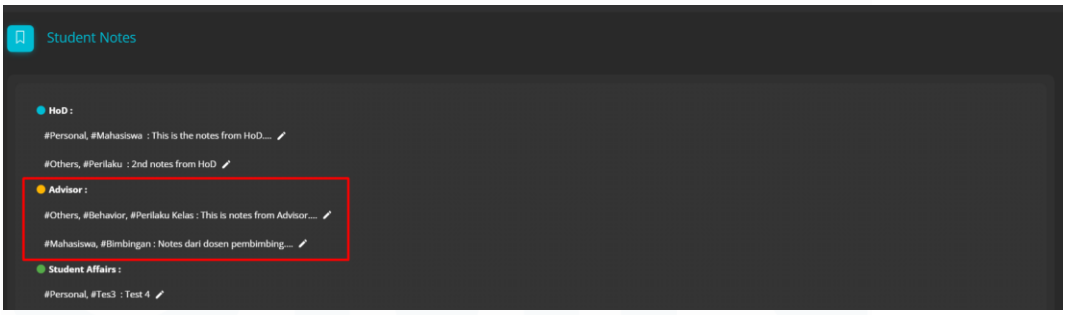
Gambar 3. 81 *Code Routes* Fitur *Student Notes*

Gambar 3.81 adalah *code* dimana fungsi-fungsi *API* fitur *Student Notes* di-*assign* ke dalam suatu *URL* sebagai bentuk dari *API Endpoint* agar bagian *front-end* dapat mengirim atau menerima data pada *database*. Untuk *URL* fitur tersebut hanya ada satu, tetapi pada satu *URL* dapat melakukan *POST*, *GET*, dan *PUT Method* sekaligus. Kemudian, *routes* tersebut akan digunakan pada *file* utama *back-end* agar dapat digunakan. Tahapan selanjutnya adalah melakukan implementasi dari *API* yang sudah dibuat pada bagian *front-end* fitur *Student Notes*.



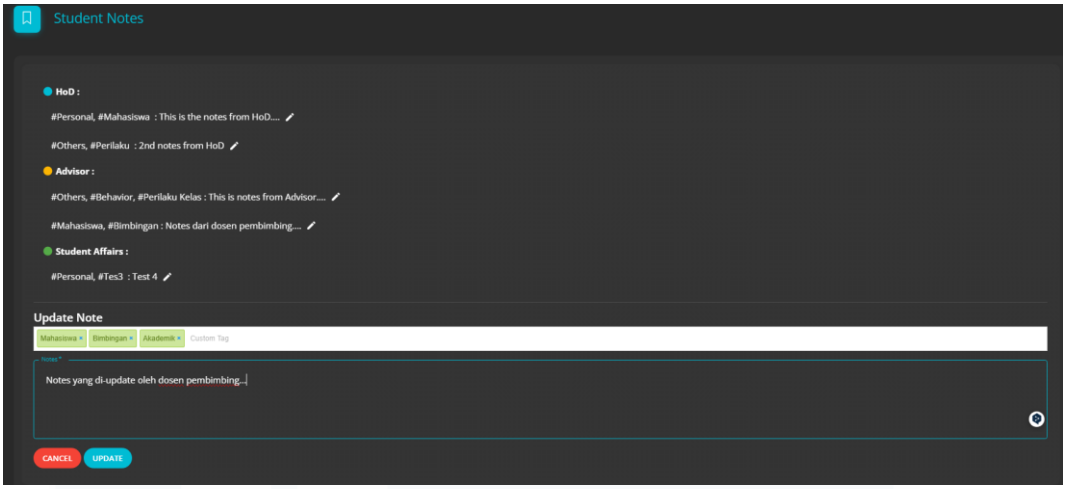
Gambar 3. 82 Proses Menambah *Student Notes*

Gambar 3.82 adalah tampilan ketika melakukan *input form* di fitur *Student Notes* untuk menambah *notes* pada mahasiswa. *User* dapat mengisi bagian *notes tag* untuk menambah *tag* pada *notes* yang akan ditambahkan. Setelah itu, *user* dapat mengisi bagian *notes* sebagai catatan untuk mahasiswa tersebut. Ketika keduanya sudah di-*input* maka *user* dapat menekan *submit* untuk menambahkan ke *database* atau *cancel* untuk membatalkan penambahan *student notes*.



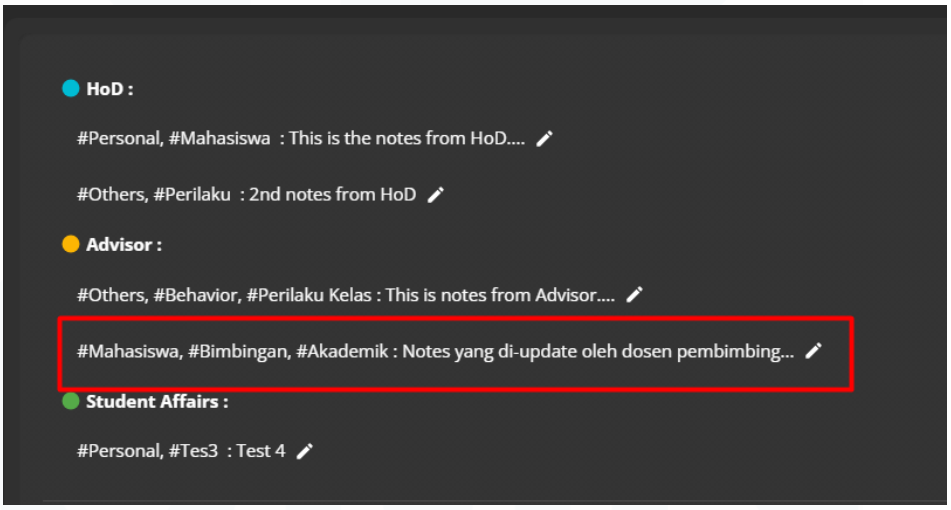
Gambar 3. 83 Tampilan *Student Notes* Berhasil Ditambahkan

Gambar 3.83 adalah tampilan ketika *student notes* berhasil ditambahkan dan sesuai dengan *user role* saat melakukan *input*, yaitu *role* sebagai *advisor*.



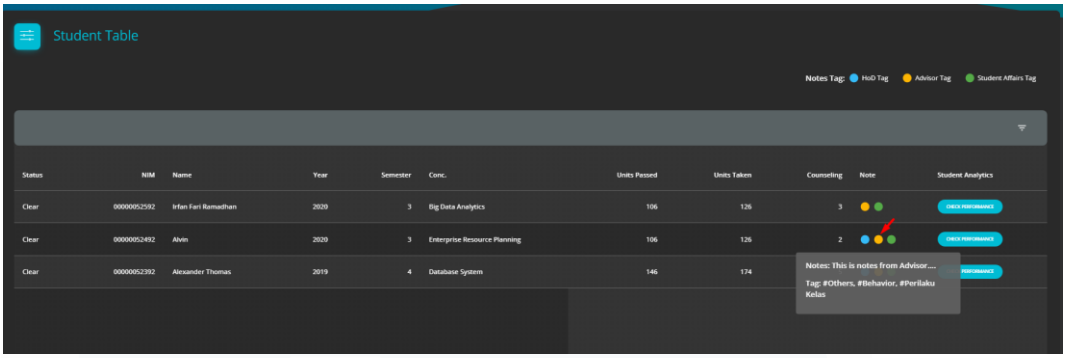
Gambar 3. 84 Proses *Update Student Notes*

Gambar 3.84 adalah tampilan proses *update student notes* dan ketika tombol *Update* ditekan, maka *notes* yang ditambahkan sebelumnya akan ter-*update* sesuai dengan *input* baru *user* tersebut.



Gambar 3. 85 Tampilan *Student Notes* Berhasil Diperbarui

Gambar 3.85 adalah tampilan ketika *student notes* berhasil diperbarui dengan *input* pada *form update* sebelumnya. Kemudian, *user* dapat melihat tampilan dari *notes* pertama yang ditambahkan di halaman *Overview* dan terintegrasi dengan fitur *Student Notes*.



Gambar 3. 86 Tampilan *Student Notes* Pada Fitur *Overview*

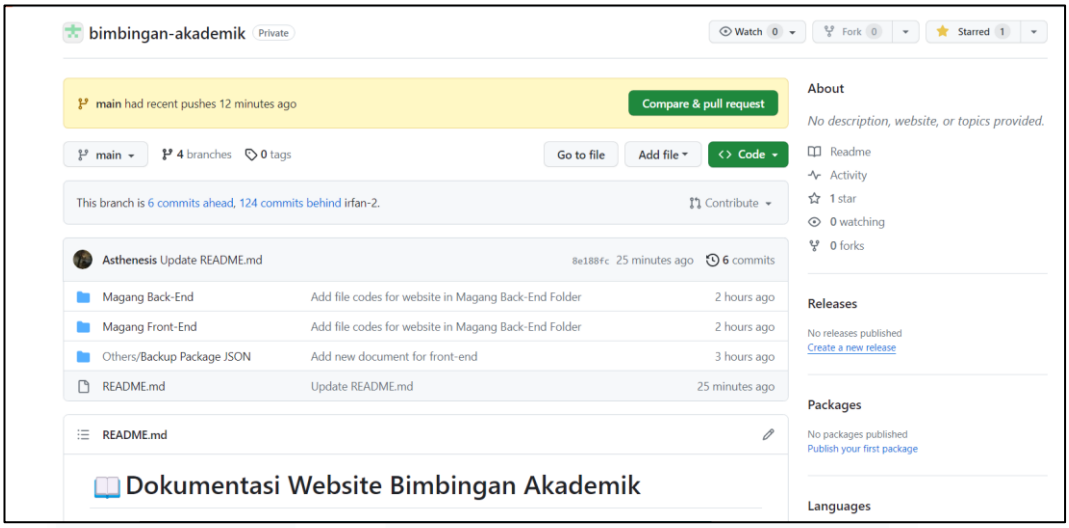
Gambar 3.86 adalah tampilan *student notes* di bagian *Student Table* fitur *Overview*. *Notes* yang ditampilkan adalah *notes* yang pertama kali ditambahkan oleh *user*. Kemudian, untuk dapat melihat *notes* tersebut, *user* dapat melakukan *hover cursor mouse* ke bagian *icon* sesuai dengan *role* yang dipilih, seperti *advisor* dengan warna *oranye*.

Gambar 3. 87 *Snippet* Data *Student Notes*

Gambar 3.87 adalah tampilan data-data yang berhasil di-*input* ke *database* fitur *Student Notes*. Data-data tersebut juga berfungsi untuk ditampilkan pada bagian *student notes* fitur *Overview* yang sudah diimplementasikan sebelumnya. Setelah *API* berhasil diterapkan di bagian *front-end* maka pengerjaan fitur yang dilakukan mahasiswa magang telah selesai, sehingga pekerjaan selanjutnya adalah membuat dokumentasi. Dokumentasi yang akan dibuat nantinya berfungsi untuk memberitahu *developer* pada periode magang berikutnya mengenai *project website* bimbingan akademik, mengenai fitur-fitur yang sudah memiliki *API* atau dilakukan implementasi *back-end*, serta mengenai struktur *folder front-end* dan juga *back-end*. Selain itu, dokumentasi yang dibuat berisikan tata cara melakukan instalasi beberapa *tools* dan juga *database*.

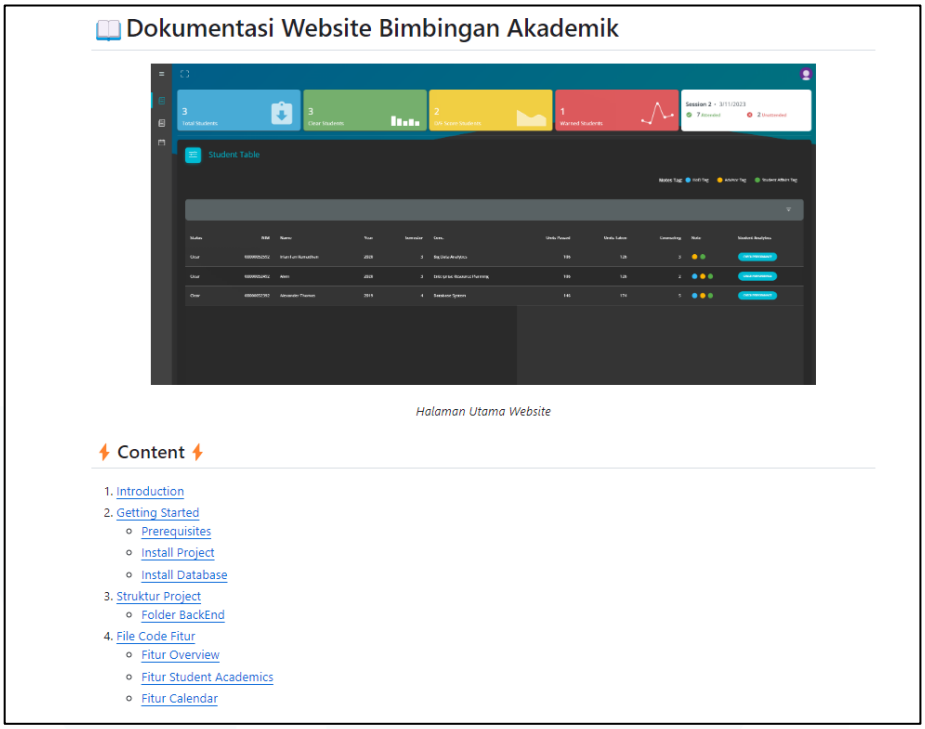
3.2.9 Membuat dokumentasi pengerjaan *back-end website* bimbingan akademik (Minggu 13 – 14)

Aktivitas selanjutnya adalah pembuatan dokumentasi terhadap pekerjaan yang sudah dilakukan oleh mahasiswa magang. Dokumentasi tersebut berupa cara instalasi beberapa *tools* atau *framework*, kemudian dilanjutkan dengan melampirkan *folder project website* bimbingan akademik dari periode magang *front-end* sampai periode magang pengerjaan *back-end*. Kegiatan kerja magang ini menjadi aktivitas terakhir mahasiswa magang sebagai *back-end developer*. Untuk dapat membuat dokumentasi tersebut, mahasiswa magang diwajibkan untuk mempunyai akun *GitHub*. Setelah memiliki akun tersebut, nantinya mahasiswa magang melakukan koordinasi dengan tim IT agar diberikan akses ke dalam *folder repository* di *GitHub* tersebut.



Gambar 3. 88 *Repository Website* Bimbingan Akademik di *GitHub*

Gambar 3.88 adalah tampilan *folder repository project website* bimbingan akademik yang telah disediakan tim IT dan di-*upload* oleh mahasiswa magang. Tampilan dari *repository* tersebut menunjukkan *main branch* atau cabang utama di *GitHub*. Pada *main branch* tersebut terdapat beberapa *folder* seperti *folder* "Magang *Back-End*" yang digunakan untuk menampung keseluruhan pengerjaan mahasiswa magang sebagai *back-end developer*. Selain itu, terdapat *folder* "Magang *Front-End*" sebagai tempat penyimpanan *project* tahap *front-end* pada periode magang sebelumnya. Untuk *folder* bernama "*Others*" digunakan dalam menyimpan *file* atau *folder* yang berkaitan dengan *project*, salah satunya adalah untuk menyimpan *file backup package.json*. *File package.json* tersebut berguna dalam menyimpan *script front-end* dan juga *back-end*, serta menyimpan daftar *package* yang diperlukan ketika mahasiswa magang melakukan *npm install*. Kemudian, mahasiswa magang menambahkan *file* bernama "*README.md*" dengan *format markdown* sebagai *file* untuk pembuatan dokumentasi.



Gambar 3. 89 Tampilan Dokumentasi *Website* Bimbingan Akademik

Gambar 3.89 merupakan potongan dari *file README.md* yang digunakan dalam membuat dokumentasi *project website* bimbingan akademik. Dokumentasi ini berguna untuk membantu *developer* atau mahasiswa magang dalam mempelajari struktur *project*, cara instalasi *tools*, hingga memberitahu fitur-fitur yang penting pada *website* bimbingan akademik. Akses yang ada pada dokumentasi menyesuaikan dengan akses *repository* dimana bersifat *private* karena untuk menghindari kebocoran data atau *file* penting pada *project*. Secara garis besar, terdapat tampilan dari halaman utama *website* untuk memberitahu bahwa gambar tersebut merupakan halaman utama ketika dosen atau *user* membuka *website* bimbingan akademik. Kemudian, terdapat daftar isi atau *content* untuk mempermudah navigasi dalam membaca dokumentasi *project* termasuk isi dokumentasi. Isi dari dokumentasi tersebut terdiri dari cara instalasi *tools*, struktur *project*, hingga menampilkan letak *file* atau *folder* masing-masing fitur *website* bimbingan akademik.

3.3 Kendala yang Ditemukan

Selama mahasiswa magang melakukan pekerjaan magang sebagai *back-end developer* di bawah naungan program studi Sistem Informasi UMN, terdapat beberapa kendala yang menghambat pengerjaan dan kelancaran kerja magang. Beberapa kendala yang dihadapi mahasiswa magang adalah sebagai berikut:

1. Koordinasi dan komunikasi antara mahasiswa magang dengan *product owner* sempat terkendala karena *meeting* secara berkala yang dilakukan dua minggu sekali tersebut beberapa kali ditunda. Hal ini membuat komunikasi yang dilakukan menjadi tidak efisien dan mampu menciptakan miskomunikasi, salah satunya dalam hal *feedback* terhadap fitur yang dikerjakan mahasiswa magang. *Feedback* yang seharusnya jelas dan menjadi acuan untuk revisi ataupun mengerjakan fitur tersebut, sempat menjadi kendala karena terdapat perbedaan *feedback* dan permintaan *product owner* di hari-hari berikutnya.
2. Mahasiswa magang sempat terkendala dalam mempelajari dan memahami hal-hal teknis seperti instalasi *tools*, kemudian cara mengembangkan *API* dan menerapkan *back-end* pada *website* bimbingan akademik. Kendala tersebut dihadapi mahasiswa magang karena beberapa *tools* dan *framework* seperti *Express* masih kurang familiar. Selain itu, mahasiswa magang sempat kebingungan dalam melakukan instalasi *tools* seperti *SQL Developer*, dan juga terutama dalam instalasi *database Oracle*.
3. Mahasiswa magang juga dihadapi kendala dalam mempelajari struktur *project* atau dokumen-dokumen terkait *project website* bimbingan akademik. Kendala tersebut adalah berupa struktur *project* yang sangat kompleks dan dapat dikatakan sulit untuk dipahami. Dokumen dan struktur *project* mengikuti *template website* yang sudah dikembangkan pada tahapan magang *front-end* di periode sebelumnya. Oleh karena itu, mahasiswa magang diwajibkan untuk dapat beradaptasi dengan *template website*, *framework React*, dan beberapa komponen-komponen

front-end termasuk *Material UI* dan *styling* pada setiap komponen tersebut.

3.4 Solusi atas Kendala yang Ditemukan

Dari uraian kendala yang sudah dijelaskan sebelumnya, mahasiswa magang berupaya untuk mencari solusi sehingga permasalahan kendala tersebut dapat diatasi dan diselesaikan. Solusi terhadap kendala tersebut di antaranya:

1. Mahasiswa magang melakukan komunikasi dengan *product owner* melalui saluran lain seperti *WhatsApp* dan *email* sehingga tidak terikat dengan *meeting* yang sering ditunda tersebut. Kemudian, mahasiswa magang juga mencatat *feedback* yang diberikan oleh *product owner* terhadap pengerjaan fitur sehingga tidak lupa dan dapat dikonfirmasi kembali dengan mudah. Dengan begitu, komunikasi antara *product owner* dan juga mahasiswa magang menjadi lebih jelas terkait pengerjaan fitur dan permintaan *product owner* pada fitur.
2. Mahasiswa magang melakukan pembelajaran secara lebih detil dan mencoba untuk menulis ulang *code* yang ada pada saat pembelajaran seperti pada saat melihat tutorial terkait *tools* dan *framework*. Kemudian, mahasiswa magang mencari sumber belajar lebih di internet sehingga dapat lebih paham mengenai konsep-konsep *back-end* dan pengembangan *API*. Selanjutnya, ketika melakukan instalasi *database Oracle*, mahasiswa magang melakukan koordinasi dengan tim IT sehingga versi *database* sesuai dengan ketentuan. Selain itu, untuk instalasi *database* dan *tools* berupa *SQL Developer*, mahasiswa magang mencari tutorial instalasi berbentuk video di *YouTube* agar lebih paham.
3. Dalam mempelajari struktur *project* yang kompleks tersebut, mahasiswa magang melihat kembali dokumentasi seperti *user flow*, tampilan *UI* / *UX website* bimbingan akademik, hingga laporan magang mahasiswa di periode magang sebelumnya. Selain itu,

mahasiswa magang membaca dokumentasi *template website*, yaitu dokumentasi *Dandelion* sebagai *template website* agar lebih memahami struktur *website* dan juga letak-letak *file* fitur *website*. Adaptasi yang dilakukan mahasiswa magang juga dibantu dengan pembelajaran dan juga *review* terkait bahasa pemrograman ataupun *framework* pada *website* bimbingan akademik.