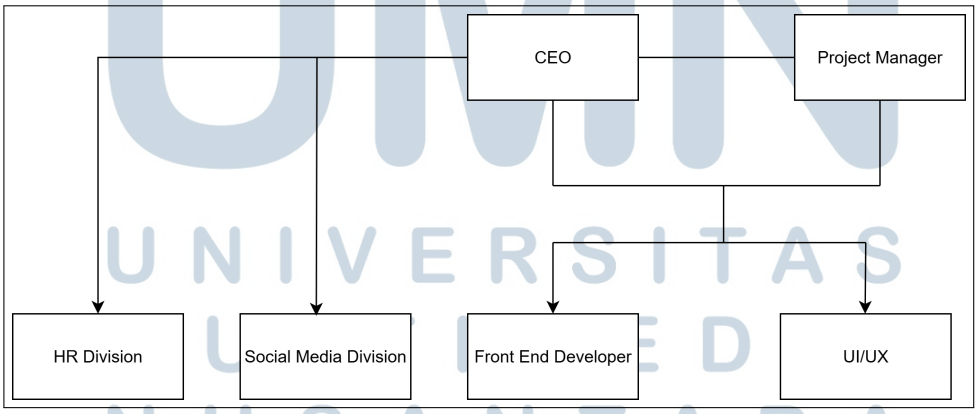
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kedudukan selama proses magang di Flow Byte Digital ini sebagai *Frontend Developer* yang berada pada divisi teknologi. Divisi Teknologi memiliki 3 anggota, dengan 1 anggota yang berfokus sebagai *Project Manager* yang bertanggung jawab mengatur jalannya sebuah proyek yang sedang dikerjakan, dan sisa 2 orang lainnya berfokus pada pengembangan antarmuka pengguna, yaitu sebagai *frontend developer*. Selama melakukan proses kerja magang, terdapat beberapa tugas seperti pembuatan komponen-komponen antarmuka, *slicing design*, dan implementasi web yang interaktif, serta responsif menggunakan Vue.js.

Dalam mengerjakan proyek magang menggunakan Visual Studio Code (VSCode) untuk pengembangan Vue.js. Selama proses pengerjaan kerja magang, koordinasi antara sesama anggota tim dilakukan dengan menggunakan aplikasi Discord dan web ClickUp yang merupakan *tools* manajemen proyek yang memungkinkan sejumlah pengguna dalam tim untuk merencanakan, mengelola, dan mendesain pekerjaan mereka secara terorganisir dan kolaboratif [5]. Lalu, *platform* Discord juga digunakan sebagai media utama untuk rapat harian yang diadakan secara daring setiap pukul 21.00 WIB. Penggunaan *platform* ini membantu kelancaran komunikasi dan kolaborasi tim selama magang berlangsung.



Gambar 3.1. Koordinasi Tugas Magang

Gambar 3.1 merupakan proses pemberian tugas selama masa magang.

3.2 Tugas yang Dilakukan

Slicing desain *UI/UX* ke dalam bentuk antarmuka pengguna merupakan tugas utama yang dilakukan selama program magang di Flow Byte Digital. Proses *slicing* ini sangat penting dalam pengembangan *frontend* karena menjadi jembatan antara desain visual statis yang dibuat oleh tim *UI/UX* dengan tampilan nyata pada aplikasi web yang dapat digunakan oleh pengguna. Melalui *slicing* yang presisi dan konsisten, desain dapat diimplementasikan secara akurat dalam bentuk komponen *frontend* menggunakan Vue.js dan TailwindCSS, sehingga menghasilkan antarmuka yang tidak hanya menarik secara visual, tetapi juga sesuai dengan standar pengembangan antarmuka modern.

Tim *frontend* bertanggung jawab untuk mengubah desain yang telah dibuat oleh tim *UI/UX* (dalam bentuk *file Figma*) menjadi tampilan web dengan menambahkan animasi atau efek interaktif, dan responsif, serta menggunakan komponen *reusable*. Implementasi dilakukan dengan menggunakan Vue.js dan TailwindCSS. Adapun desain yang diubah menjadi tampilan antarmuka meliputi halaman *landing page*, autentikasi (*login* dan *forgot password*), hingga seluruh detail tampilan *dashboard* siswa yang akan dijelaskan pada bagian selanjutnya. Pengembangan sistem pada proyek magang ini difokuskan hanya pada fitur dan tampilan antarmuka untuk aktor *student*, berdasarkan pembagian tanggung jawab dalam tim. Fitur untuk aktor lain seperti *admin* atau mentor tidak termasuk dalam ruang lingkup tugas selama pelaksanaan magang.

3.3 Uraian Pelaksanaan Magang

Untuk memberikan gambaran yang terperinci tentang langkah-langkah teknis dan alur kerja yang dijalankan. Pelaksanaan kerja magang diuraikan seperti pada tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Memahami, mempelajari, menganalisis kebutuhan dalam pembuatan proyek LMS *bootcamp* serta melakukan *setup* proyek Vue.Js, TailwindCSS, Vite dan menyesuaikan dengan semua kebutuhan *framework* yang diperlukan, yaitu Chart.js.
2 - 4	Menerapkan desain *UI/UX* pada *frontend* web dengan melakukan *slicing* desain *landing page*, yaitu pada halaman *bootcamp*, *free class*, dan *testimonials*.
5 - 6	Menerapkan desain *UI/UX* pada *frontend* web dengan melakukan *slicing* desain halaman *login*, dan *forgot password*. Kemudian, menghubungkan setiap navigasi yang berkaitan.
7 - 9	Menerapkan desain UI/UX pada *frontend* web dengan melakukan *slicing* desain halaman *dashboard student*. Kemudian, menghubungkan setiap navigasi yang berkaitan.
10 - 14	Menerapkan desain *UI/UX* pada *frontend* web dengan melakukan *slicing* desain pada *available class*, dan *class studied* termasuk detailnya, yaitu informasi kelas, *task history*, *submission history*, *leaderboard*, *learning materials*, *detail learning materials*, dan *dialog box* yang bersangkutan.
15 - 16	Menerapkan desain *UI/UX* pada *frontend* web dengan melakukan *slicing* desain notifikasi, dan *setting profile*.

3.4 Implementasi Antarmuka

3.4.1 Teknologi yang Digunakan

Website Flow Camp ID dikembangkan menggunakan Vue.js 3 sebagai *frontend framework* yang bersifat *component-based* untuk membangun *UI* yang dapat digunakan kembali. Selain itu, TailwindCSS sebagai *utility-first CSS framework* guna mempercepat proses *styling* secara efisien, di mana penggunaan TailwindCSS sebagai kerangka kerja *CSS* dapat mempersingkat waktu pemrosesan gaya *CSS* dan memberikan pengguna kebebasan untuk mendesain sesuai keinginan dengan semua fitur yang memudahkannya [6].

Proses *build* dan pengembangan untuk proyek ini dilakukan menggunakan Vite. Sebagai sebuah *build tool* untuk pengembangan *front-end*, Vite dirancang

untuk mengelola berbagai aktivitas teknis yang bertujuan meningkatkan alur kerja dan performa.

3.4.2 Metodologi *Slicing Design* UI/UX ke Antarmuka Frontend

Proses *slicing design* merupakan tahapan konversi desain statis dari Figma menjadi antarmuka web interaktif dan responsif menggunakan *framework* Vue.js dan TailwindCSS. Tahapan ini dilakukan secara sistematis agar hasil akhir sesuai dengan desain acuan dan mendukung fungsionalitas sistem LMS yang dikembangkan. Adapun tahapan *slicing design* yang dilakukan berdasarkan kebijakan perusahaan adalah sebagai berikut.

1. Analisis Desain *UI/UX*

   Desain yang dibuat oleh tim *UI/UX* dianalisis untuk memahami struktur *layout*, hierarki informasi, penggunaan warna, dan penempatan komponen visual. Tujuannya adalah agar implementasi antarmuka dapat sesuai secara fungsional dan visual.

2. Pemetaan Komponen Antarmuka

   Bagian-bagian dari desain kemudian dipetakan menjadi struktur komponen modular yang sesuai dengan prinsip pengembangan berbasis Vue.js. Komponen yang bersifat berulang atau umum dirancang agar dapat digunakan kembali (*reusable*).

3. Ekspor Aset Desain

   Elemen grafis seperti ikon, ilustrasi, dan gambar diekspor dari Figma dalam format *SVG* atau *PNG* untuk kemudian diintegrasikan ke dalam antarmuka menggunakan pendekatan *asset management* dalam proyek Vue.js.

4. *Setup* Proyek dan Struktur Direktori

   Proyek *frontend* diinisialisasi menggunakan Vite. Struktur direktori dikonfigurasi agar terdiri dari folder *components*, *assets*, *views*, dan *router* untuk menjaga kerapian dan keteraturan dalam pengembangan.

5. Implementasi *Layout* dan *Styling*

   Struktur halaman dibangun dengan pendekatan *layout* berbasis *grid* dan *flexbox* yang disediakan oleh TailwindCSS. *Styling* dilakukan berdasarkan

panduan desain, dengan fokus pada keterbacaan, keselarasan warna, serta konsistensi antar elemen. Visual tidak diharuskan 100% identik dengan desain Figma, namun tetap mengikuti acuan secara umum agar tampil rapi dan selaras dengan desain Figmanya, bahkan dalam beberapa kasus, *supervisor* langsung memberikan arahan perubahan *layout* tanpa melalui proses revisi dari tim *UI/UX*. Akibatnya, desain Figma yang tersedia tidak mengalami pembaruan, sementara implementasi *frontend* tetap mengacu pada perubahan langsung dari *supervisor*.

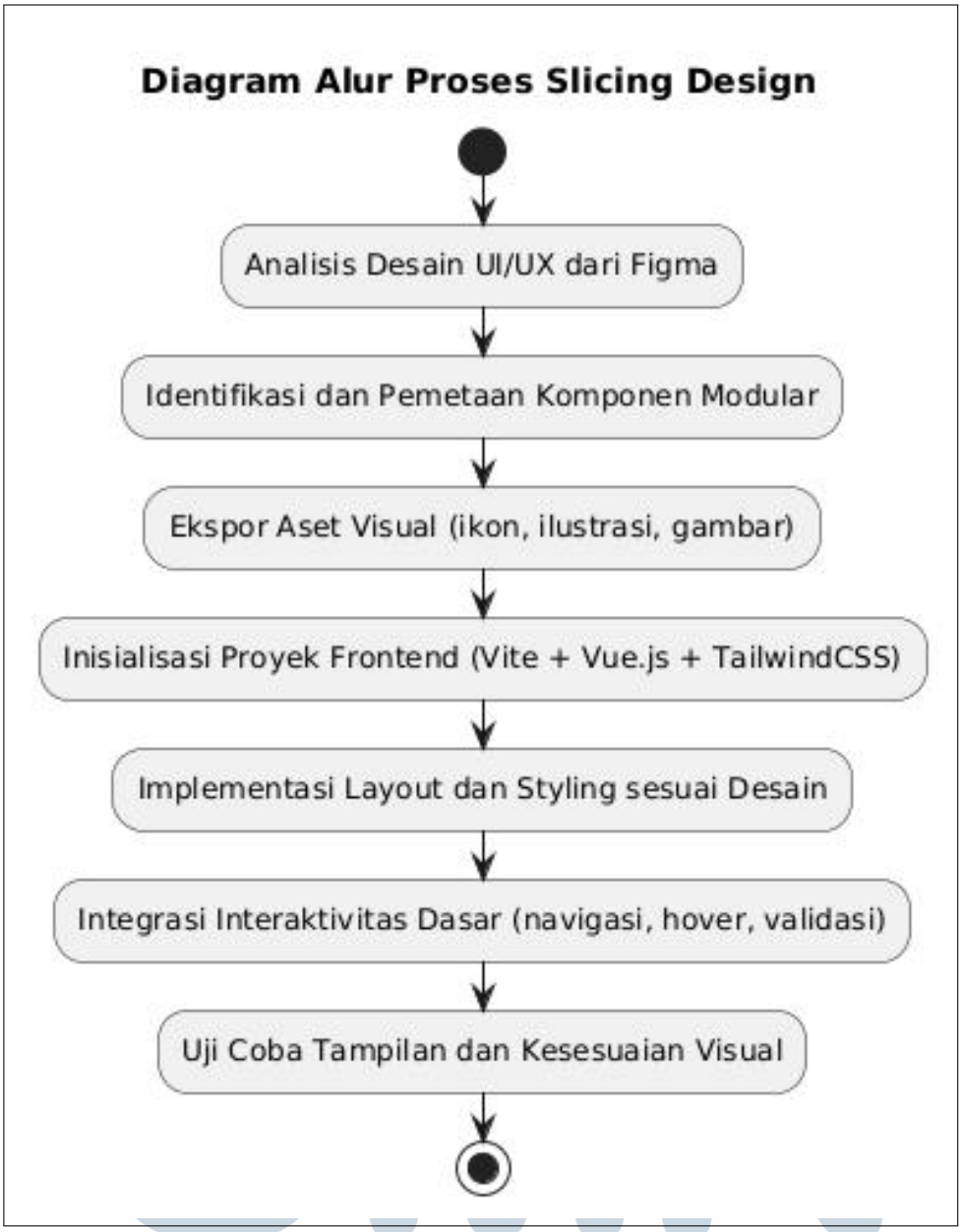
6. Integrasi Navigasi

   Setelah tampilan utama selesai, dilakukan konfigurasi navigasi antar halaman menggunakan Vue Router. Beberapa interaksi seperti tombol aksi, dan *tab* konten juga diimplementasikan untuk mendukung fungsionalitas aplikasi.

7. Uji Tampilan Sederhana

   Hasil *slicing* diuji secara manual menggunakan *browser* untuk memastikan tampilan antarmuka sudah berfungsi dan tampil sesuai dengan kebutuhan, meskipun hanya untuk tampilan *desktop* karena tidak diwajibkan dalam proyek.

Untuk memperjelas tahapan kerja yang dilakukan dalam proses *slicing design*, berikut ditampilkan diagram alur proses pada gambar 3.2 yang menunjukkan urutan langkah implementasi antarmuka pengguna berdasarkan desain Figma.



Gambar 3.2. Diagram Alur Proses *Slicing Design*

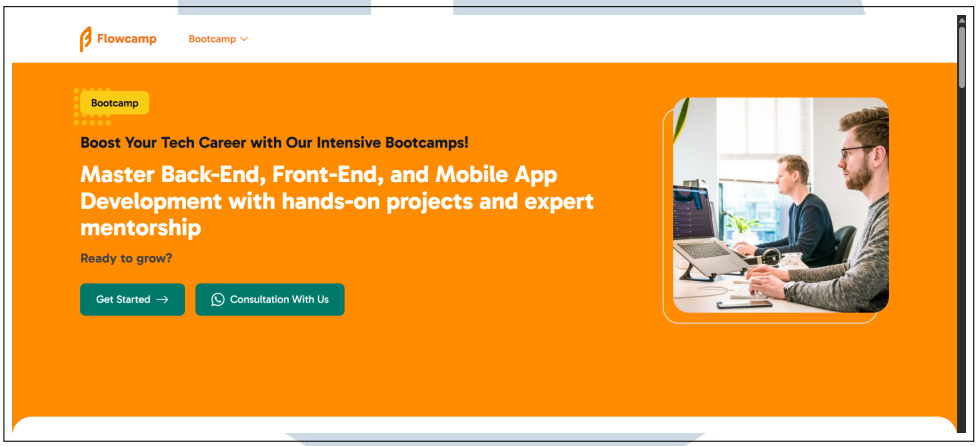
3.4.3 Hasil Implementasi Antarmuka

A Tampilan *Landing Page*

A.1 *Landing Page "Bootcamp"*

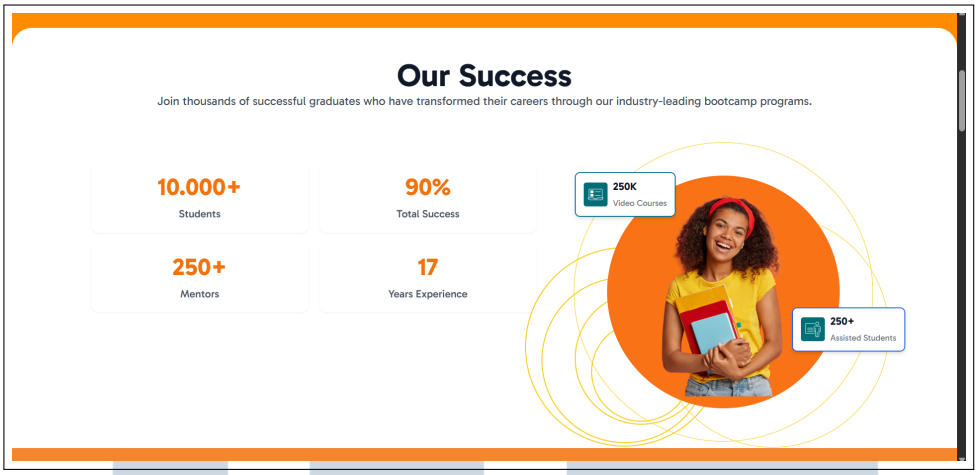
Pada gambar 3.3 menampilkan *hero section* dari halaman *bootcamp* pada LMS Flowcamp. Pada bagian atas terlihat *navbar* berisi logo Flowcamp sebagai nama *bootcamp*, dan menu navigasi, yaitu, "Program", "*Bootcamp*", dan "*Free*

Class". Di bawah *navbar* terdapat *headline* yang mencolok disertai subjudul singkat. Terdapat pula tombol *call-to-action* "*Get Started*" yang ketika diklik akan mengarah ke halaman register. Lalu, terdapat tombol "*Consultation With Us*" yang ketika diklik akan mengarahkan pengguna ke WhatsApp. Bagian ini merupakan komponen dan pada bagian "*Bootcamp*", dan tipografi yang ada dibuat menggunakan *props* untuk menerima parameter dengan tujuan agar dapat digunakan di halaman yang lain.



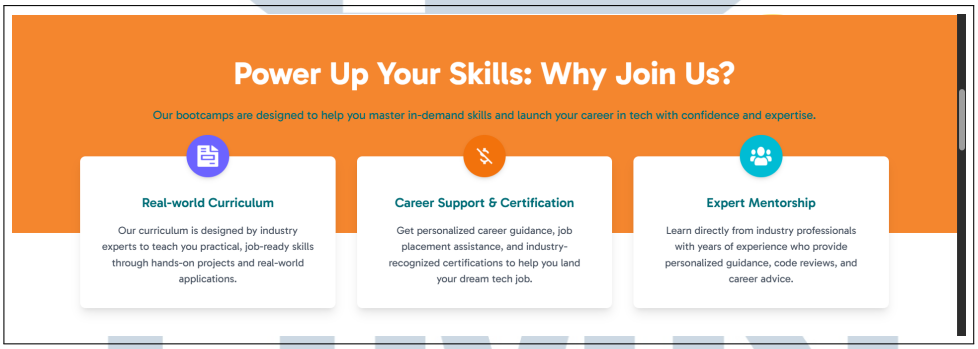
Gambar 3.3. Halaman *Bootcamp* Bagian 1

Gambar 3.4 letaknya berada di bawah gambar 3.3 dan merupakan bagian *bootcamp* yang menunjukkan informasi mengenai jumlah siswa yang lulus dari *bootcamp* ini beserta persentasenya. Kemudian, terdapat jumlah mentor beserta pengalamannya dalam menunjang *bootcamp* ini. Bagian ini menggunakan pendekatan desain yang inovatif dengan kombinasi *background* oranye dan *container* putih yang memiliki *border* radius melengkung di bagian atas. *Layout* menggunakan *Grid*. Kemudian pada bagian kanannya menggunakan teknik *layering* dengan *z-index management* untuk menciptakan *depth* visual yang menarik. *Badge* yang ditampilkan memberikan informasi tambahan tentang *video courses* dan *assisted students* yang memperkuat nilai *bootcamp*.



Gambar 3.4. Halaman *Bootcamp* Bagian 2

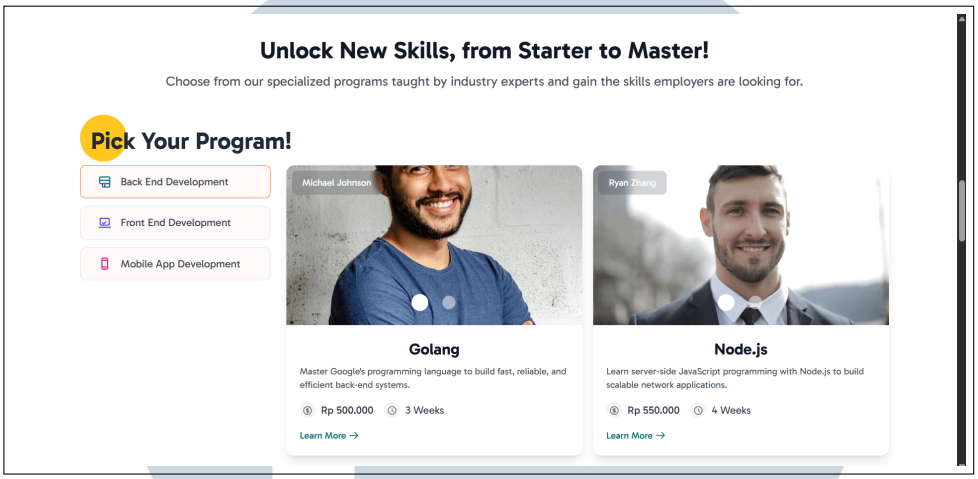
Pada gambar 3.5 menyajikan informasi yang menjelaskan alasan-alasan penting mengapa seseorang perlu mengikuti program *bootcamp* ini. Setiap *skill card* menggunakan *floating icon design* dengan *negative margin technique* untuk menciptakan efek ikon yang melayang di atas *card*.



Gambar 3.5. Halaman *Bootcamp* Bagian 3

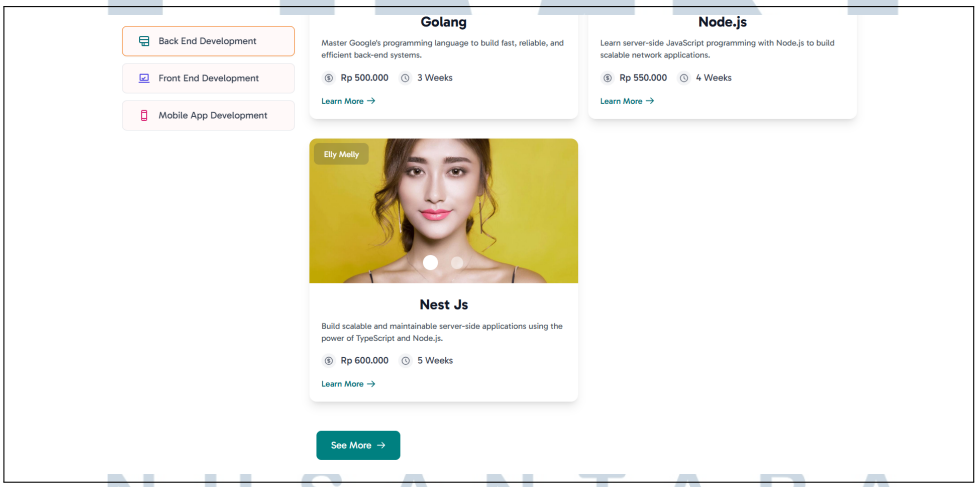
Lalu masih pada halaman yang sama, pada gambar 3.6 terdapat komponen *card* dengan data props yang berbeda-beda sesuai dengan *card*nya. pada bagian ini juga terdapat *filtering* untuk 3 kategori program yang bisa dilihat, yaitu ”*Back End Development*”, ”*Front End Development*”, dan ”*Mobile App Development*”, dimana dalam setiap kategori terdapat data *card* yang sesuai dengan kategori yang dipilih. Pada *card* ini juga menggunakan *pagination* yang menunjukkan bahwa dalam satu *card* dapat memiliki beberapa program/bahasa pemrograman. Ketika pengguna mengklik *pagination* pada *card* tersebut, maka data gambar, nama program, deskripsi, harga, dan lama programnya itu akan menyesuaikan datanya. Bagian ini mengimplementasikan sistem *filtering* dengan *sidebar navigation* yang menggunakan *sticky positioning*. *Sidebar filter* mempertahankan *visibility* saat

user melakukan *scroll* yang meningkatkan *usability*. Sistem *filtering* menggunakan *reactive state management* dengan *computed properties* yang secara otomatis memfilter dan membatasi tampilan program berdasarkan kategori yang dipilih.



Gambar 3.6. Halaman *Bootcamp* Bagian 4

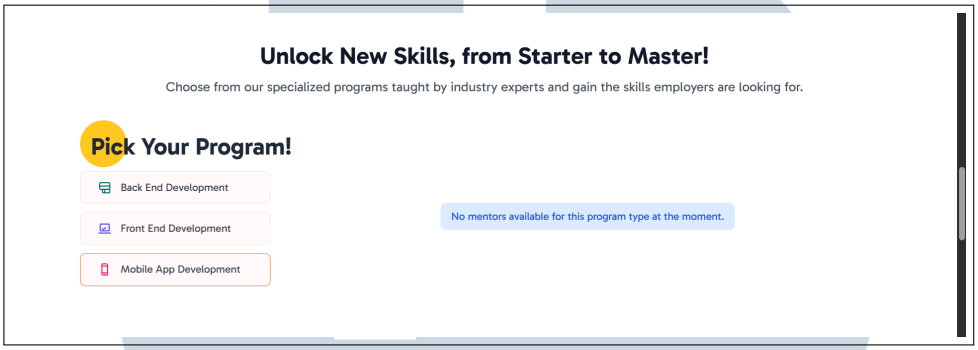
Gambar 3.7 merupakan tampilan lanjutan dari 3.6, dimana terdapat button ”*See More*” dengan *right arrow* yang dibuat menggunakan *format Scalable Vector Graphics* (SVG) untuk memastikan kualitas visual yang optimal dan skalabilitas yang baik pada berbagai ukuran tampilan. Tombol ini akan muncul jika terdapat *card* pada kategori program tersebut, dan jika tidak terdapat *card*- nya, maka tombol ”*See More*” nya tidak akan muncul pada tampilan *website*-nya.



Gambar 3.7. Halaman *Bootcamp* Bagian 4 See More Button

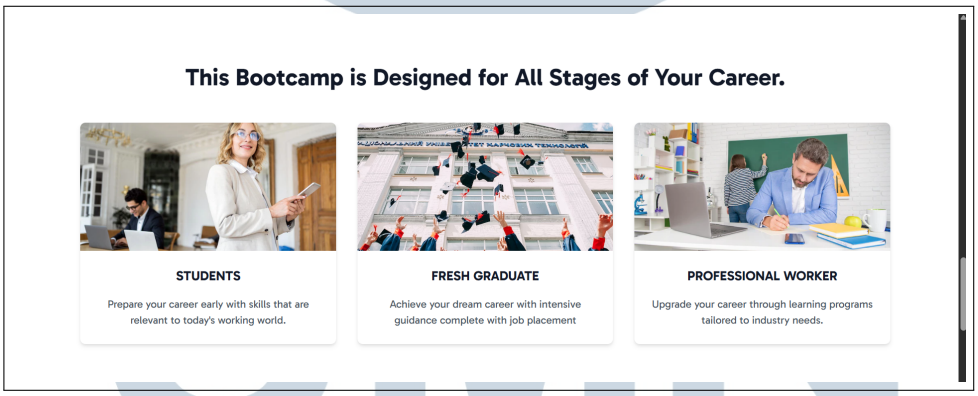
Lalu, ketika suatu kategori tidak memiliki data *card*, maka akan menampilkan kondisi *fallback state* seperti yang terlihat pada gambar 3.8. Alih-alih menampilkan *card* program, antarmuka memperlihatkan pesan teks “*No mentors*

available for this program type at the moment". Secara pengalaman pengguna (UX), desain ini memberi umpan balik visual yang jelas bahwa pengguna perlu menunggu atau kembali nanti, menghindari kesan *broken UI*. Implementasi Vue.js-nya memanfaatkan *v-if*/*v-else* untuk menampilkan *template* alternatif.



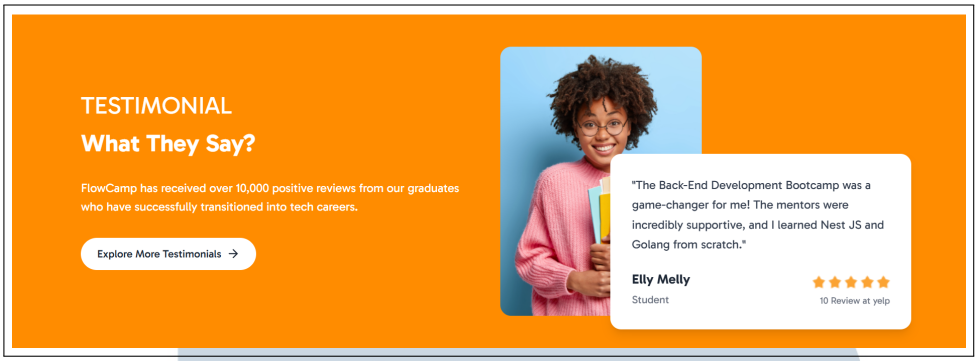
Gambar 3.8. Halaman *Bootcamp* Bagian 4 saat Tidak Ada Program

Kemudian, gambar 3.9 menunjukkan bahwa *bootcamp* ini bisa diikuti oleh berbagai macam kalangan. Bagian ini menampilkan tiga segmen target *audience* dengan implementasi *hover effects* yang memberikan *interactive feedback*.



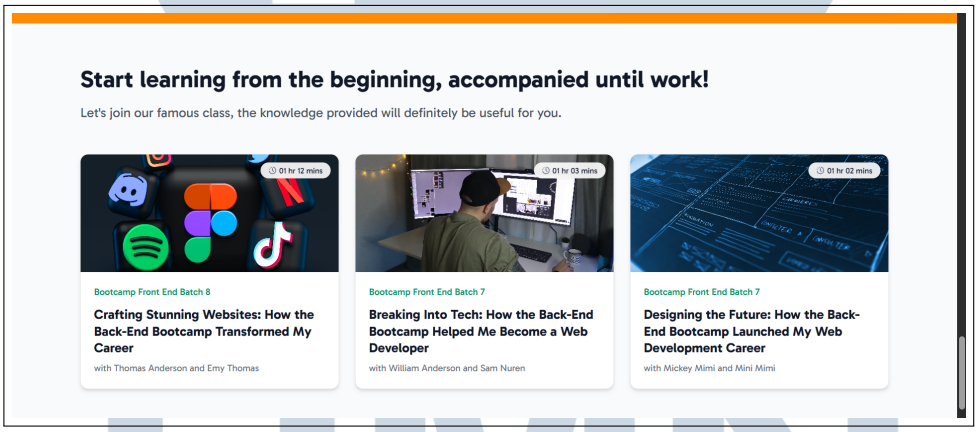
Gambar 3.9. Halaman *Bootcamp* Bagian 5

Pada gambar 3.10 terdapat informasi mengenai testimoni dari siswa yang sudah pernah menyelesaikan *bootcamp* ini. Pada bagian ini juga terdapat tombol "*Explore More Testimonials*" yang ketika diklik akan diarahkan ke tampilan testimoni yang lebih banyak lagi. Bagian ini menggunakan *layout* grid, dan *card feedback* nya menggunakan *absolute positioning* dengan *negative values* untuk menciptakan *overlay effect* yang natural di atas gambar *student*.



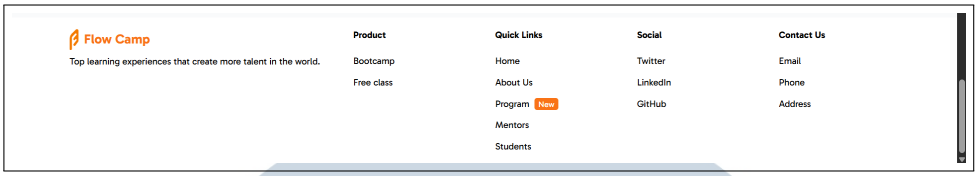
Gambar 3.10. Halaman *Bootcamp* Bagian 6

Kemudian, tampilan di bawah gambar 3.10, terdapat gambar 3.11 yang menunjukkan *batch-batch* yang sudah pernah dibuka oleh *bootcamp* ini dengan informasi yang ditampilkan melalui komponen *card*. Bagian ini menggunakan *computed property* atau *props* untuk menampilkan data *courses* yang tersedia.



Gambar 3.11. Halaman *Bootcamp* Bagian 7

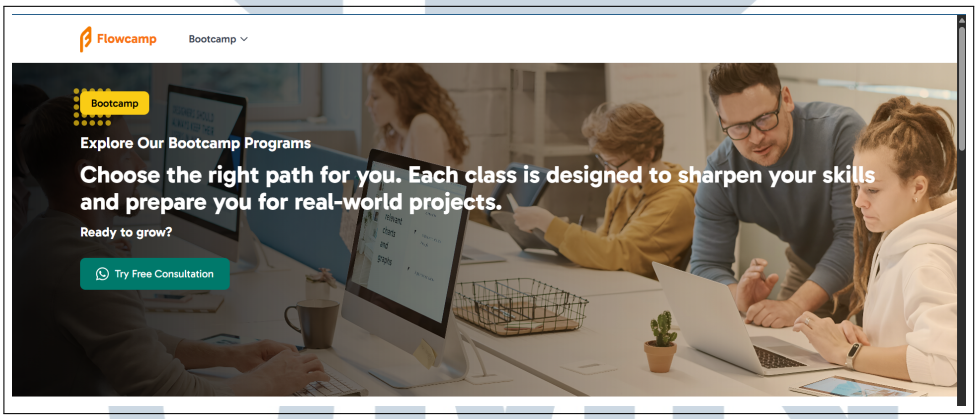
Lalu, gambar 3.12 menunjukkan *footer* pada *landing page* yang menggunakan *layout grid* 12 kolom. Komponen ini terdiri dari lima *section* utama, yaitu *branding section* yang menampilkan logo dan *tagline* Flowcamp dengan span 4 kolom, *section* ”Product” yang berisi *link* ke ”*Bootcamp*” dan ”Free class”, *section* ”Quick Links” yang menampilkan navigasi utama *website* termasuk ”Home”, ”About Us”, ”Program”, ”Mentors”, dan ”Students”, *section* ”Social” yang berisi *link* ke *platform* media sosial Twitter, LinkedIn, dan GitHub, serta *section* ”Contact Us” yang menyediakan informasi kontak *Email*, *Phone*, dan *Address*. Setiap *navigation link* menggunakan *hover effect* dengan transisi warna yang dari *black* ke *orange* untuk memberikan *feedback* interaktif.



Gambar 3.12. *Footer*

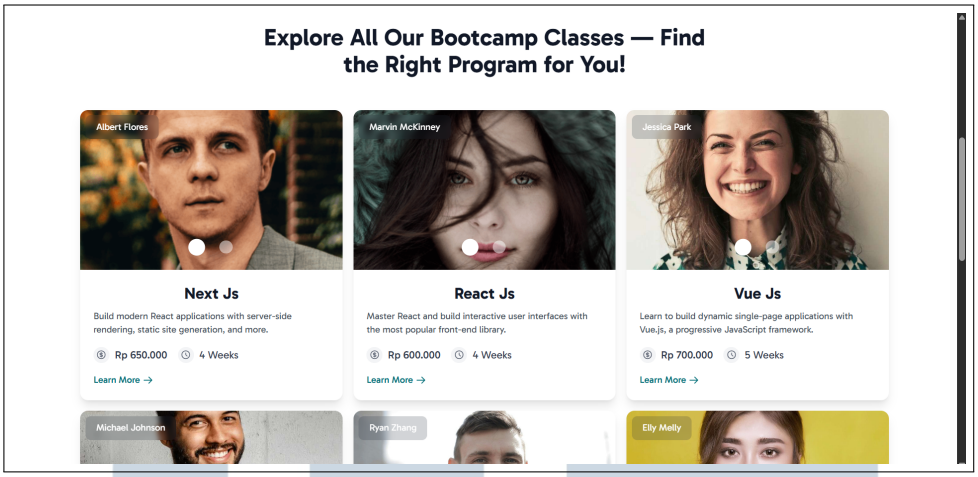
A.2 *Landing Page "Bootcamp See More"*

Landing page ini merupakan tampilan hasil navigasi dari interaksi pengguna dengan tombol "See More" yang terdapat pada gambar 3.7. Di bagian atas dari halaman ini terdapat *navbar* dengan navigasi "*bootcamp*", lalu terdapat *hero section* dengan tampilan seperti pada gambar 3.13. Pada halaman ini, tombol "Get Started" yang ada di gambar 3.3 itu dihilangkan, dan *typography* nya diganti untuk komponen *hero* ini dengan menggunakan *props*, sehingga pada bagian ini tombol yang terlihat hanyalah tombol "Try Free Consultation". Lalu, *background*-nya juga diganti menggunakan gambar dan *overlay* dengan *boolean*.



Gambar 3.13. Halaman *Bootcamp See More* Bagian 1

Kemudian, di bagian bawahnya terdapat tampilan semua program yang ada di *bootcamp* ini seperti yang terlihat pada gambar 3.14. Pada bagian ini, menggunakan komponen *card* yang sama seperti yang ada pada gambar 3.6, sehingga terdapat *pagination* dengan data yang menyesuaikan juga. Bagian ini menampilkan *grid* 3 kolom yang menggunakan komponen *card* untuk setiap program. Kemudian, terdapat implementasi *loading state* menggunakan *reactive ref* dengan *loading indicator* berupa *spinning animation*, dan *conditional rendering* menggunakan *v-if*/*v-else* untuk menampilkan *loading state* atau *grid* program *cards* tergantung pada status *loading*.



Gambar 3.14. Halaman *Bootcamp See More* Bagian 2

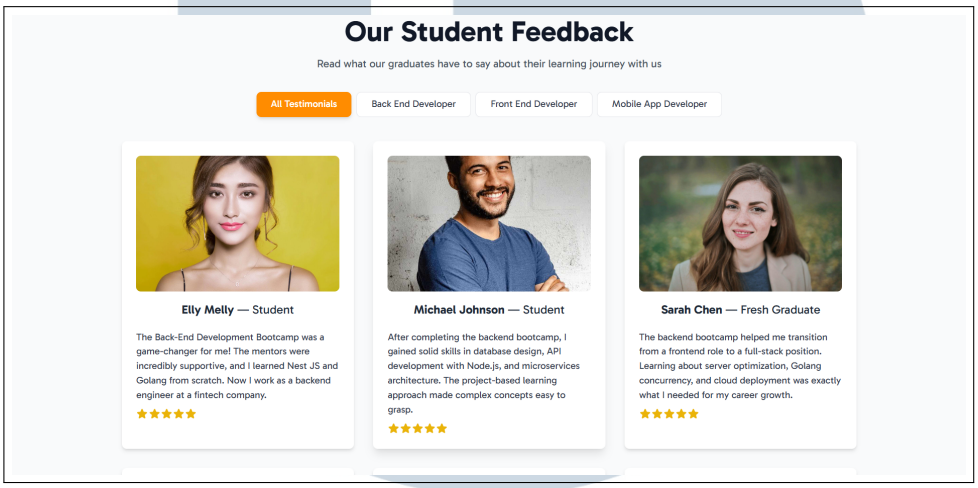
A.3 *Landing Page "Testimonial"*

Tampilan *landing page* yang diimplementasikan selanjutnya, yaitu bagian detail *testimonial*. Bagian ini merupakan halaman yang menampilkan ulasan dari alumni *bootcamp* FlowCamp. Halaman ini akan terlihat ketika pengguna mengklik tombol "*Explore More Testimonials*" yang dapat dilihat pada gambar 3.10. Lalu, tampilan di bagian atas dari halaman ini terdapat *navbar* dengan navigasi "*bootcamp*", kemudian terdapat *hero section* dengan tampilan seperti pada gambar 3.15. *Hero section* ini juga merupakan komponen, sehingga *typography* dan gambar yang ada disini, termasuk *background*-nya dapat diubah menggunakan *props*.

Gambar 3.15. Halaman *Testimonials* Bagian 1

Kemudian, di bagian bawahnya terdapat tampilan seperti pada gambar 3.16 terdapat tampilan detail testimoni dari orang-orang yang sudah pernah mengikuti *bootcamp* yang dilengkapi dengan sistem filter kategori menggunakan

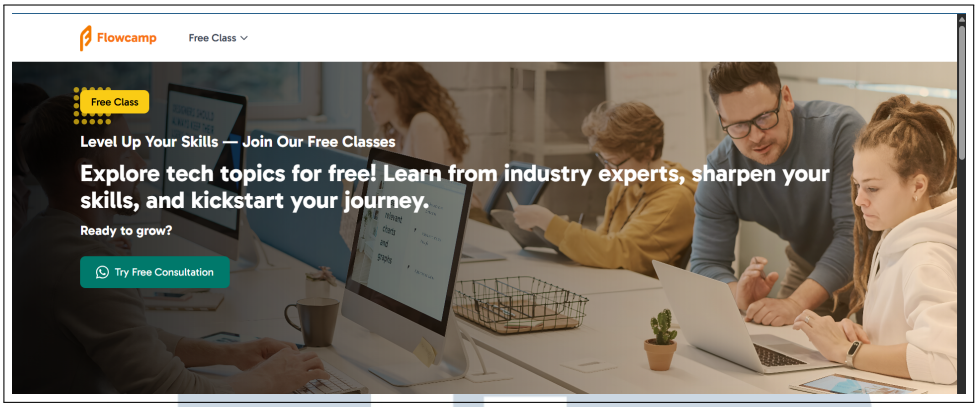
tab navigation untuk empat kategori utama yaitu ”All Testimonials”, ”Back End Developer”, ”Front End Developer”, dan ”Mobile App Developer”. Grid *layout testimonials* menggunakan *breakpoints* dengan komponen *card* untuk setiap *testimonial*, implementasi *infinite scrolling* menggunakan *Intersection Observer API* yang mendeteksi ketika *user scroll* mendekati bagian bawah dan secara otomatis memuat 6 *testimonial* tambahan dengan *loading indicator* yang menggunakan animasi TailwindCSS.



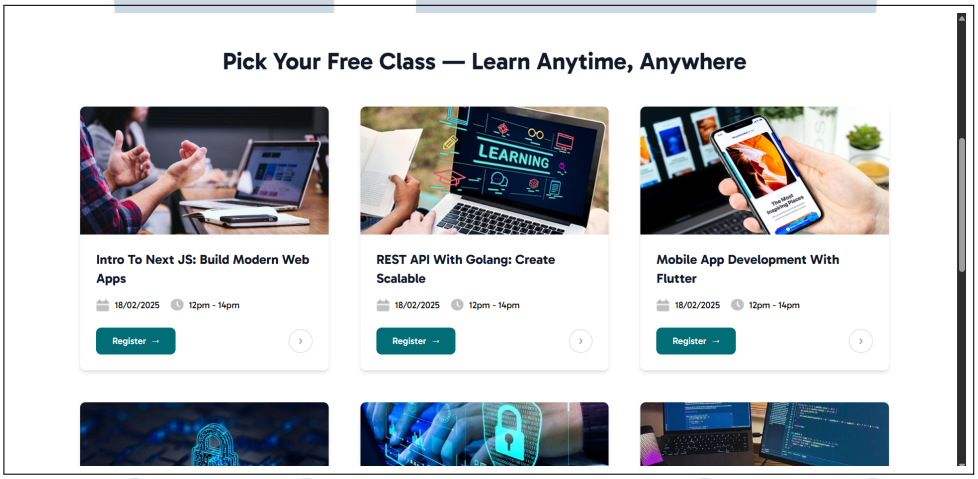
Gambar 3.16. Halaman *Testimonials* Bagian 2

A.4 *Landing Page ”See More Free Class”*

Halaman Landing Page ”See More Free Class” merupakan halaman yang menampilkan daftar lengkap kelas gratis yang tersedia di FlowCamp. Halaman ini menggunakan komponen *HeroSection* dengan konfigurasi khusus yang mengaktifkan *overlay* gelap dan menyembunyikan *right section* dengan *badge* ”Free Class” seperti yang terlihat pada gambar 3.17. Kemudian, pada gambar 3.18 diimplementasikan dengan *grid layout* yang menampilkan 9 kelas per halaman menggunakan komponen *card* dengan data yang menyesuaikan.



Gambar 3.17. Halaman *See More Free Class* Bagian 1



Gambar 3.18. Halaman *See More Free Class* Bagian 2

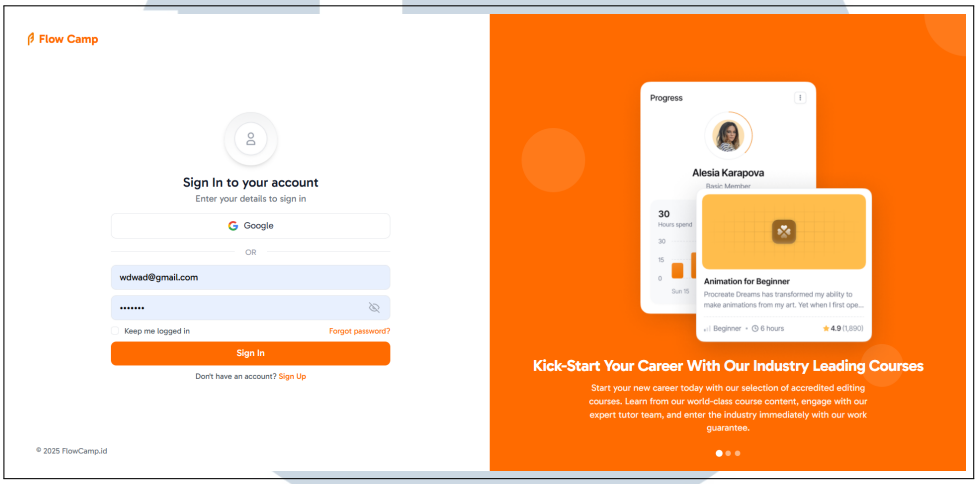
B Tampilan Autentikasi dan *Forgot Password*

B.1 Halaman Login

Gambar 3.19 menunjukkan tampilan *login* yang terdiri dari dua bagian utama yaitu bagian kiri untuk formulir autentikasi dan bagian kanan untuk konten visual promosi. Implementasi *layout* menggunakan sistem *flexbox* dengan kelas `flex-col lg:flex-row` untuk memastikan tampilan responsif antara perangkat *mobile* dan *desktop*. Bagian kiri mencakup logo *FlowCamp*, ikon profil pengguna, formulir *login* dengan validasi *email* dan *password*, tombol masuk melalui *Google OAuth*, serta *link* untuk pendaftaran dan pemulihan *password*.

Bagian kanan halaman *login* dirancang khusus untuk tampilan *desktop* dengan implementasi *hidden lg:flex* yang menyembunyikan konten pada layar kecil. Bagian ini menampilkan latar belakang berwarna oranye dengan elemen

dekoratif berupa lingkaran-lingkaran semi-transparan dan gambar ilustrasi *login*. Sistem *carousel* sederhana diimplementasikan untuk menampilkan teks promosi yang berbeda-beda dengan indikator *pagination dots* di bagian bawah. Penggunaan *utility classes* dari TailwindCSS memungkinkan implementasi yang efisien untuk *spacing*, *typography*, *colors*, dan *responsive breakpoints*, sehingga menghasilkan antarmuka yang konsisten dan sesuai dengan standar desain modern.



Gambar 3.19. Halaman Login

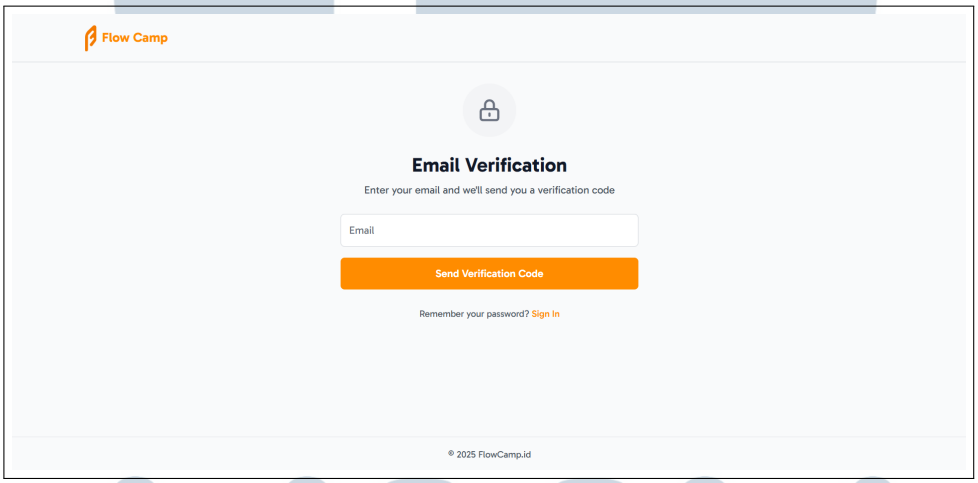
B.2 Halaman OTP Authentication

1. *Email Verification*

   Bagi pengguna yang melupakan *password* atau akun yang digunakan, proses pemulihan harus melalui tahapan *email verification* untuk memastikan identitas pengguna. Tampilan dari proses verifikasi *email* tersebut dapat dilihat pada gambar 3.20. Implementasi halaman verifikasi *email* menggunakan pendekatan *layout* yang sederhana dan berfokus pada *user experience* yang optimal. Struktur halaman terdiri dari tiga bagian utama yaitu *header* dengan logo *FlowCamp*, konten utama yang berisi formulir verifikasi, dan *footer* dengan informasi *copyright*. Implementasi *responsive design* menggunakan sistem *flexbox* dengan kelas `flex flex-col items-center justify-start min-h-screen` untuk memastikan konten terpusat dan dapat menyesuaikan berbagai ukuran layar. Area formulir dibatasi dengan *max-width* menggunakan kelas `max-w-md` untuk menjaga proporsi yang ideal pada layar besar.

   Elemen visual utama pada halaman ini adalah ikon gembok yang

diimplementasikan menggunakan *SVG icon* dengan latar belakang lingkaran abu-abu untuk memberikan penekanan visual terhadap fungsi keamanan. Formulir verifikasi *email* dilengkapi dengan validasi *input* yang memberikan *feedback* visual melalui perubahan warna *border* dan pesan kesalahan. Penggunaan skema warna konsisten dengan *brand identity FlowCamp* yaitu warna oranye `#ff8c00` pada tombol dan *link* navigasi. Implementasi *responsive typography* menggunakan kelas TailwindCSS seperti `text-2xl sm:text-3xl` memastikan keterbacaan optimal pada berbagai perangkat, sementara *spacing* yang konsisten diterapkan melalui *utility classes* untuk *padding* dan *margin* yang responsif.



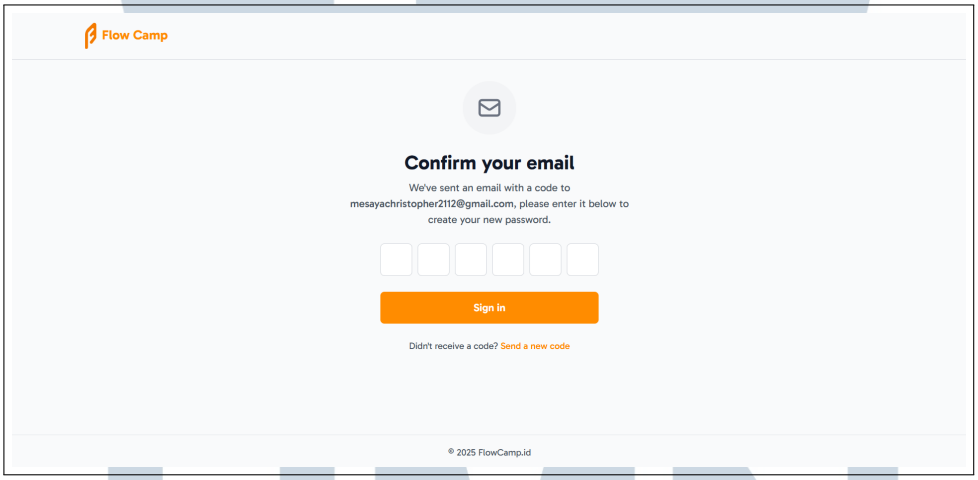
Gambar 3.20. Verifikasi Email

2. Kode OTP

Kemudian, pada gambar 3.21 menggunakan struktur *layout* yang konsisten dengan halaman verifikasi sebelumnya namun dengan komponen *input* yang lebih kompleks. Halaman ini terdiri dari tiga bagian utama yaitu *header* dengan logo *FlowCamp*, area konten utama yang berisi formulir konfirmasi kode, dan *footer* dengan informasi *copyright*. Elemen visual utama menggunakan ikon *email* berbentuk *SVG* dengan latar belakang lingkaran abu-abu untuk memberikan konteks visual yang jelas terhadap fungsi halaman. Implementasi *responsive design* menggunakan sistem *flexbox* dengan kelas `flex flex-col items-center justify-start min-h-screen` untuk memastikan konten terpusat secara vertikal dan horizontal pada berbagai ukuran layar.

Komponen utama pada halaman ini adalah *VerificationInput* yang

menampilkan enam kotak *input* terpisah untuk memasukkan kode verifikasi digit per digit. Implementasi *dynamic content* menggunakan *props* dan *conditional rendering* untuk menyesuaikan teks dan fungsi berdasarkan sumber akses halaman, baik dari proses registrasi maupun pemulihan *password*. Fitur *resend code* dilengkapi dengan sistem *cooldown timer* yang memberikan *feedback* visual melalui perubahan warna tombol dan penampilan *countdown* waktu. Pesan notifikasi sukses dan error ditampilkan menggunakan komponen *alert* dengan latar belakang hijau dan merah yang memberikan *feedback* visual yang jelas kepada pengguna. Konsistensi *brand identity* tetap dipertahankan dengan penggunaan warna oranye `#ff8c00` pada tombol utama dan elemen interaktif lainnya.



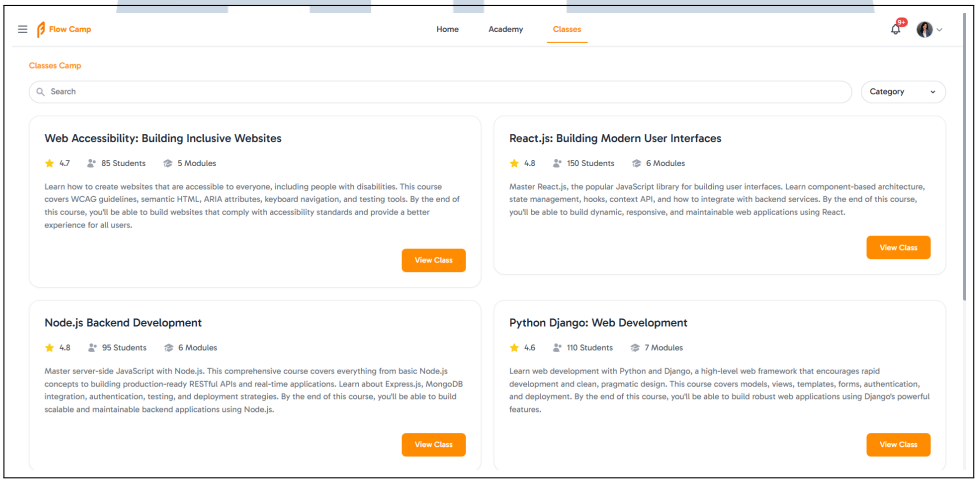
Gambar 3.21. Kode OTP

C Tampilan Dashboard student bagian Available Class

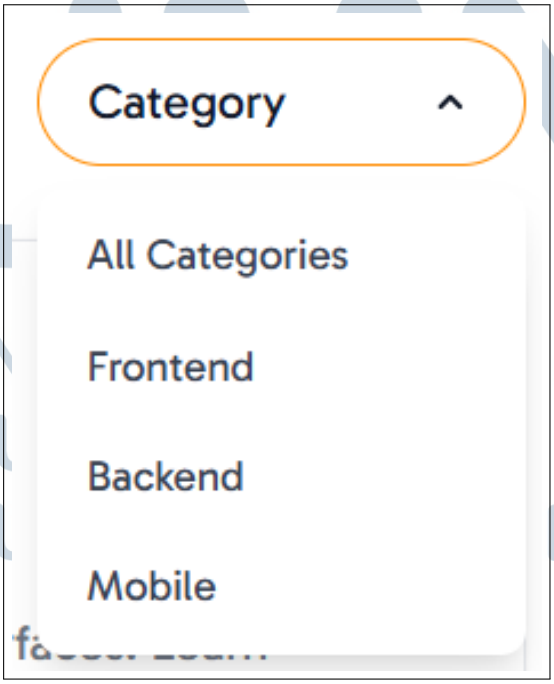
C.1 Halaman *Available Class*

Halaman ini merupakan halaman yang menampilkan daftar kelas yang tersedia untuk diikuti oleh siswa dengan tampilan yang dapat dilihat pada 3.22. Implementasi halaman ini menggunakan *layout* yang kompleks dengan sistem *grid responsive* dan fitur pencarian yang interaktif. Halaman ini menampilkan komponen *breadcrumb* dengan teks "*Classes Camp*" dalam warna oranye untuk navigasi. Area pencarian terdiri dari *input field* dengan ikon pencarian menggunakan *SVG* yang diposisikan secara absolut pada sisi kiri, serta *dropdown* kategori dengan ikon panah yang dapat berputar menggunakan transformasi `rotate-180` seperti pada gambar 3.23. Implementasi *grid system* menggunakan

kelas `grid-cols-1 md:grid-cols-2` untuk menampilkan kartu kelas dalam format satu kolom pada *mobile* dan dua kolom pada *desktop*. Sistem *infinite scroll* diimplementasikan dengan *loading indicator* berupa *spinner* menggunakan animasi `animate-spin` dan *intersection observer* untuk memuat konten tambahan secara dinamis. Konsistensi *brand identity* dipertahankan dengan penggunaan warna oranye pada elemen interaktif seperti *focus states*, *hover effects*, dan *loading indicator*, sementara *responsive design* diterapkan melalui variasi ukuran teks dan *spacing* menggunakan *utility classes* TailwindCSS.



Gambar 3.22. Halaman *Available Classes*

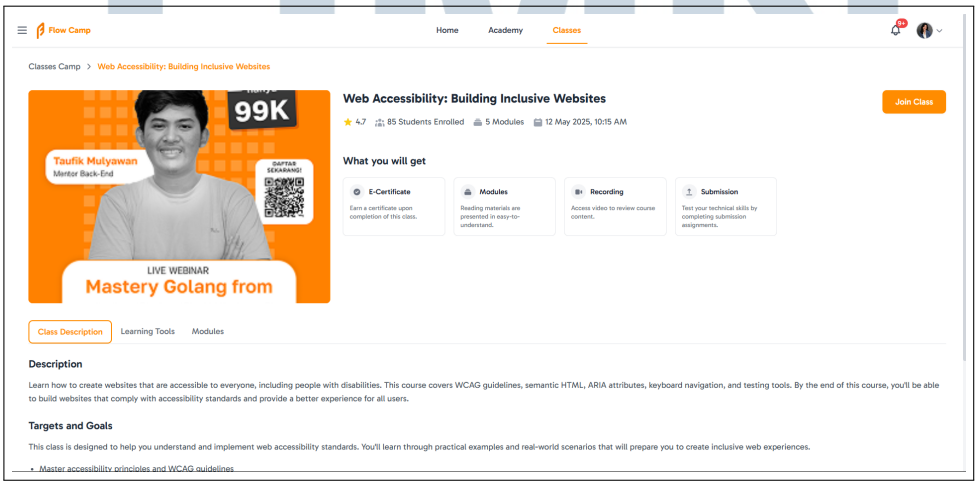


Gambar 3.23. Tampilan Kategori

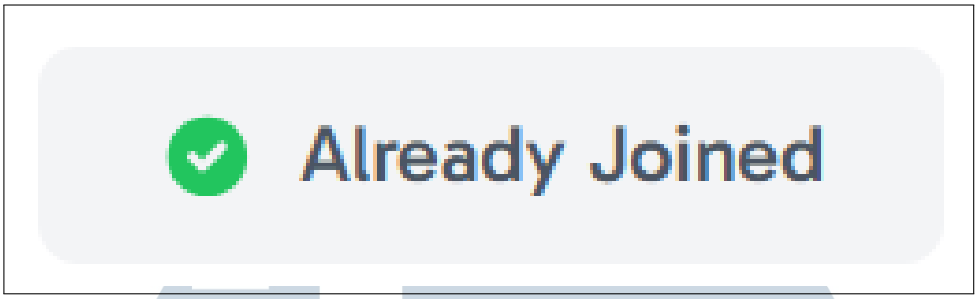
C.2 Halaman Detail *Available Class*

Dari halaman *available class*, pengguna dapat memilih salah satu kelas yang tersedia untuk melihat informasi lebih lanjut. Pilihan tersebut akan mengarahkan pengguna ke halaman *detail available class* yang berisi informasi lengkap mengenai kelas yang dipilih. Selain konten utama, seperti yang terlihat pada bagian atas gambar 3.24 terdapat *navbar student*, lalu bagian *footer student*-nya terlihat di bagian bawah gambar 3.27.

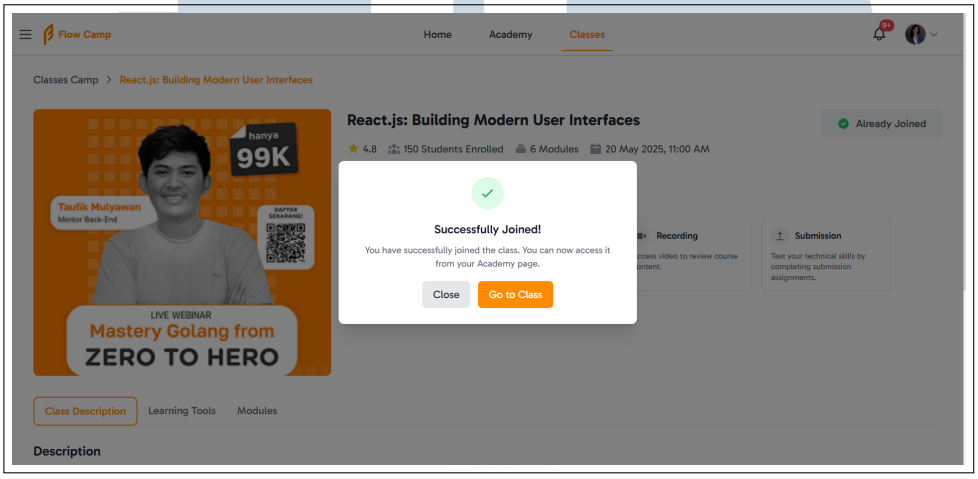
Pada gambar 3.24 menggunakan struktur *layout* yang kompleks dengan sistem *tab navigation* dan *responsive grid layout*. Halaman ini terdiri dari komponen `NavbarStudent` dan `FooterStudent` sebagai *wrapper*, *breadcrumb navigation* dengan ikon panah *SVG*, serta area konten utama yang dibagi menjadi dua kolom menggunakan kelas `flex-col lg:flex-row`. Kolom kiri menampilkan gambar kelas dengan *placeholder loading state* dan *error handling* untuk gambar yang gagal dimuat, sedangkan kolom kanan berisi informasi kelas dengan tombol "*Join Class*" yang diposisikan secara absolut menggunakan `absolute top-0 right-0`. Tombol ini memiliki *dynamic state* yang berubah dari `bg-orange text-white` menjadi `bg-gray-100 text-gray-600` dengan ikon *checkmark* hijau dan teks "*Already Joined*" setelah kelas berhasil di-*join* yang terlihat pada gambar 3.25. *Modal success* ditampilkan menggunakan *fixed overlay* dengan `fixed inset-0 bg-black bg-opacity-50` dan ikon *checkmark* berwarna hijau untuk memberikan *feedback* visual setelah bergabung dengan kelas seperti pada gambar 3.26.



Gambar 3.24. Halaman Detail *Available Classes*



Gambar 3.25. Tombol ”Join Class”

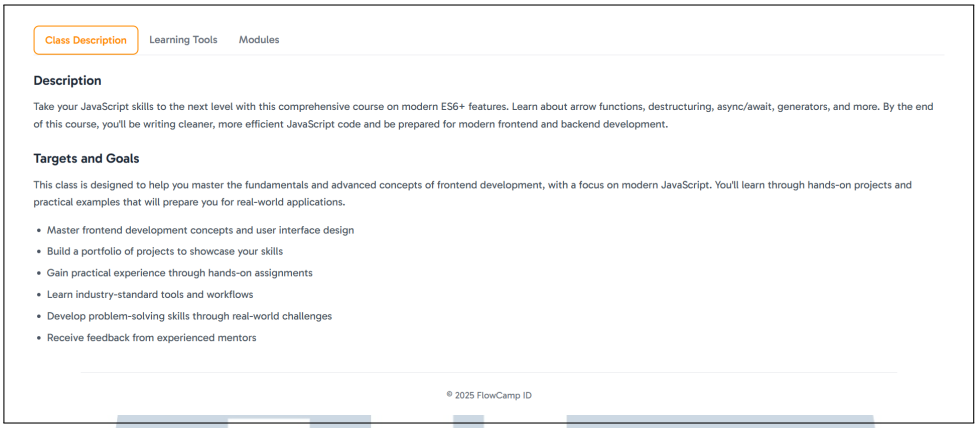


Gambar 3.26. Dialog Box ”*Join Class*”

Bagian selanjutnya yang ada di halaman detail *available class* ini, yaitu sistem *tab navigation* diimplementasikan dengan tiga tab yaitu ”*Class Description*”, ”*Learning Tools*”, dan ”*Modules*” yang menggunakan *conditional styling* dengan kelas `border-2 border-orange` untuk tab aktif. Area konten tab menampilkan informasi yang berbeda-beda dengan *list styling* menggunakan `list-disc` dan *spacing* yang konsisten.

1. *Class description*

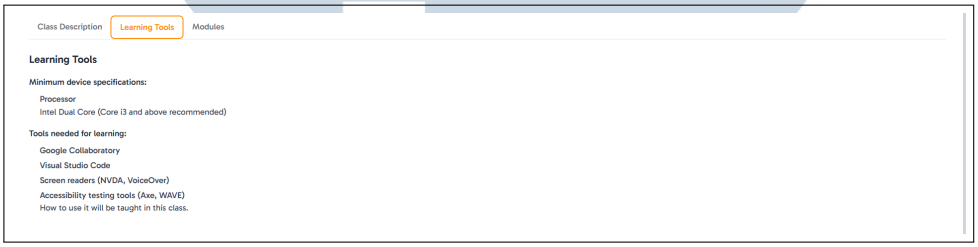
   Class description ini memberikan informasi mengenai deskripsi kelasnya serta target dan tujuan dari kelas ini seperti yang terlihat pada gambar 3.27.



Gambar 3.27. *Class Description*

2. *Learning tools*

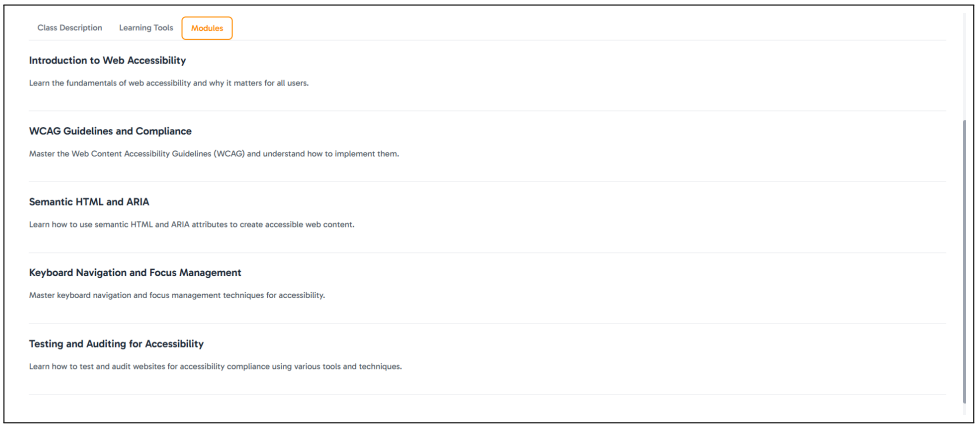
   Seperti pada gambar 3.28, bagian ini menampilkan minimum *device specifications* dan *tools needed* dengan *conditional content*.



Gambar 3.28. Learning Tools

3. *Modules*

   Seperti terlihat pada gambar 3.29, tab ini menampilkan daftar modul yang ada di setiap *available class*-nya dengan memperlihatkan informasi nama modul dan deskripsinya masing-masing untuk setiap kelasnya.

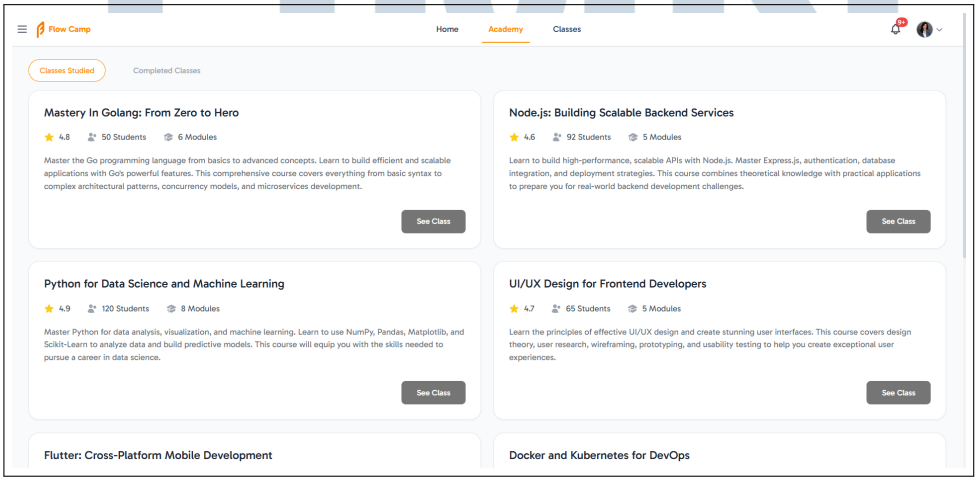


Gambar 3.29. Modules

C.3 Halaman *Class Studied*

Setelah pengguna mempelajari informasi lengkap pada halaman detail *Available Class* dan memutuskan untuk mengikuti kelas tersebut, maka kelas yang telah diikuti akan secara otomatis tercatat dan ditampilkan pada halaman *Class Studied*. Halaman ini berfungsi untuk menampilkan daftar kelas yang saat ini sedang atau sudah diikuti oleh pengguna.

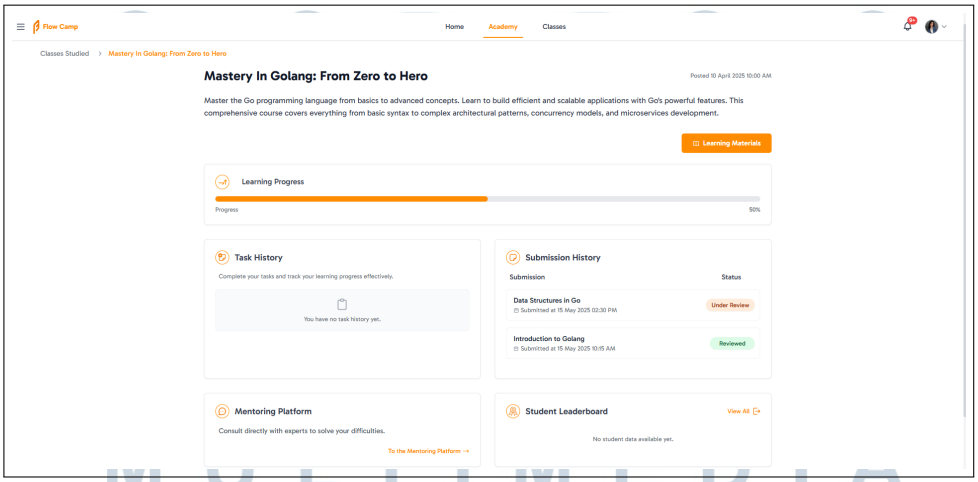
Pada gambar 3.30 terlihat bahwa implementasi halaman ini menggunakan *background* abu-abu melalui *prop* `contentClass="bg-gray-50"`. Sistem *tab navigation* diimplementasikan dengan dua tab yaitu "*Classes Studied*" dan "*Completed Classes*" yang menggunakan *conditional styling* dengan kelas `border border-orange text-orange rounded-full bg-white` untuk tab aktif dan `text-gray-400 hover:text-gray-600` untuk tab tidak aktif. Area konten utama menampilkan kartu kelas dalam format *grid responsive* menggunakan `grid-cols-1 md:grid-cols-2` dengan *spacing* yang konsisten melalui `gap-4 sm:gap-6`. Sistem *infinite scroll* diimplementasikan dengan *loading indicator* berupa *spinner* menggunakan kelas `border-orange border-t-transparent rounded-full animate-spin` dan *intersection observer* untuk memuat konten tambahan secara otomatis. *Empty state* ditampilkan dengan pesan terpusat menggunakan `col-span-1 md:col-span-2 text-center` ketika tidak ada kelas yang ditemukan, sementara *responsive design* diterapkan melalui variasi *padding* dan ukuran elemen menggunakan *utility classes* TailwindCSS seperti `py-4 sm:py-8` dan `w-6 h-6 sm:w-8 sm:h-8`.



Gambar 3.30. Halaman Class Studied

C.4 Halaman Detail *Class Studied*

Ketika pengguna mengklik tombol ”See Class” nya, maka pengguna akan melihat tampilan detail kelasnya seperti yang terlihat pada gambar 3.31. Implementasi halaman ini menggunakan komponen *breadcrumb navigation* melalui `StudentBreadcrumb`. Halaman ini memiliki struktur *layout* yang kompleks dengan judul kelas dan tanggal posting yang diatur dalam *flexbox* menggunakan `flex-col sm:flex-row justify-between items-start sm:items-center`. Area konten utama terdiri dari beberapa komponen kartu dengan *styling* konsisten menggunakan `bg-white border border-gray-200 rounded-lg shadow-sm` dan *padding responsive* melalui `p-4 sm:p-6`. *Progress bar* diimplementasikan dengan *container* abu-abu `bg-gray-200 h-2 sm:h-3 rounded-full` dan *fill bar* berwarna oranye yang menggunakan *dynamic width* berdasarkan persentase kemajuan. Sistem *grid layout* diterapkan dengan `grid-cols-1 md:grid-cols-2 gap-6 lg:gap-8` untuk mengatur komponen *Task History*, *Submission History*, *Mentoring Platform*, dan *Student Leaderboard*. Berbagai *dialog modal* seperti `TaskHistoryDialog`, `SubmissionHistoryDialog`, `FeedbackDialog`, dan `CertificateView` diintegrasikan untuk menampilkan konten tambahan dengan *overlay* dan *transition effects* yang memberikan pengalaman pengguna yang interaktif dan responsif.

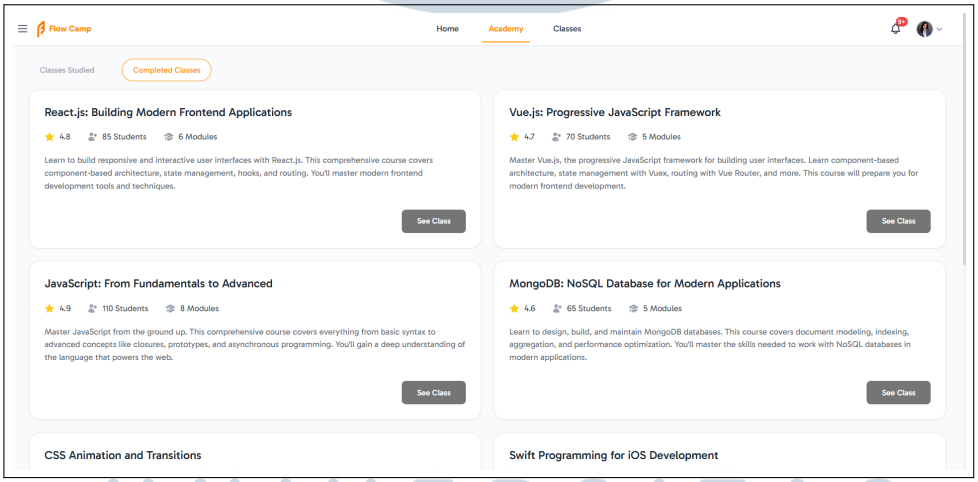


Gambar 3.31. Halaman Detail Class Studied

C.5 Halaman *Completed Classes*

Ketika *progress bar* yang ada pada ”Detail Class Studied” itu sudah mencapai 100% yang menandakan bahwa pengguna sudah menyelesaikan kelasnya,

maka kelas tersebut akan pindah ke halaman tab ”Completed Class” seperti yang terlihat pada gambar 3.32. Implementasi halaman ini menggunakan sistem *tab navigation* yang sama dengan *Classes Studied*, namun dengan fokus pada tab ”*Completed Classes*” yang memiliki *styling* identik menggunakan `border border-orange text-orange rounded-full bg-white` untuk *active state*. Konten utama menampilkan kartu kelas yang telah diselesaikan dalam format *grid responsive* dengan `grid-cols-1 md:grid-cols-2 gap-4 sm:gap-6` dan komponen `ClassCard` yang sama untuk konsistensi visual. Sistem *sorting* khusus diterapkan untuk kelas yang telah selesai berdasarkan tanggal penyelesaian tertua menggunakan *completion date logic* yang mengambil tanggal *submission* dari materi dengan status `reviewed` atau `turned_in`. *Empty state* untuk kelas yang telah diselesaikan menampilkan pesan ”*No completed classes found*” dengan *styling* terpusat menggunakan `col-span-1 md:col-span-2 text-center py-8 sm:py-12`. Sistem *infinite scroll* dan *loading indicator* menggunakan implementasi yang sama dengan tab *Classes Studied*, mempertahankan konsistensi *user experience* melalui *intersection observer* dan *spinner animation* dengan kelas `border-orange border-t-transparent rounded-full animate-spin`.

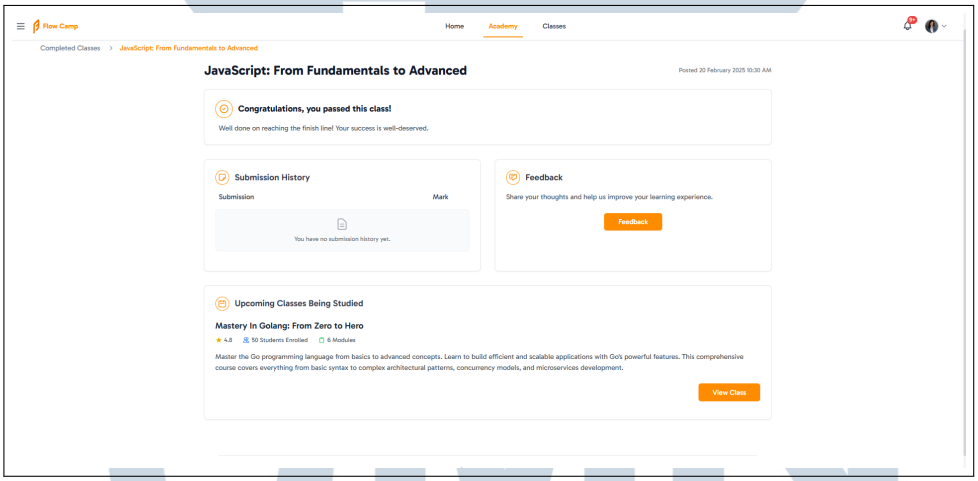


Gambar 3.32. Halaman Completed Classes

C.6 Halaman Detail *Completed Classes*

Ketika user mengklik tombol ”See Class” yang ada pada tab ”Completed Classes”, maka akan terlihat tampilan ”Detail Completed Class” seperti yang terlihat pada gambar 3.33. Implementasi halaman ini menggunakan *conditional rendering* dengan *computed property* `displayAsCompleted` yang mengubah tampilan berdasarkan status kelas. Untuk kelas yang telah selesai, halaman

menampilkan pesan selamat dalam kartu khusus dengan *styling* `bg-white border border-gray-200 rounded-lg shadow-sm p-4 sm:p-6` dan menyembunyikan elemen seperti *progress bar* dan tombol "*Learning Materials*" menggunakan `v-if="!displayAsCompleted"`. Area konten utama menggunakan *grid layout* yang sama dengan `grid-cols-1 md:grid-cols-2 gap-6 lg:gap-8`, namun menggantikan komponen *Task History* dengan *Submission History* yang menampilkan materi dengan status `reviewed`, dan menggantikan *Mentoring Platform* serta *Student Leaderboard* dengan komponen *Upcoming Sessions* yang menggunakan *styling* `border border-gray-200 rounded-lg bg-white shadow-sm p-4 sm:p-6 transition-all hover:shadow-md`. Integrasi *dialog modal* khusus untuk kelas selesai meliputi `FeedbackDialog` dan `CertificateView` dengan *overlay* dan *transition effects*, sementara *breadcrumb navigation* secara dinamis menampilkan "*Completed Classes*" sebagai *path* pertama untuk konsistensi navigasi.



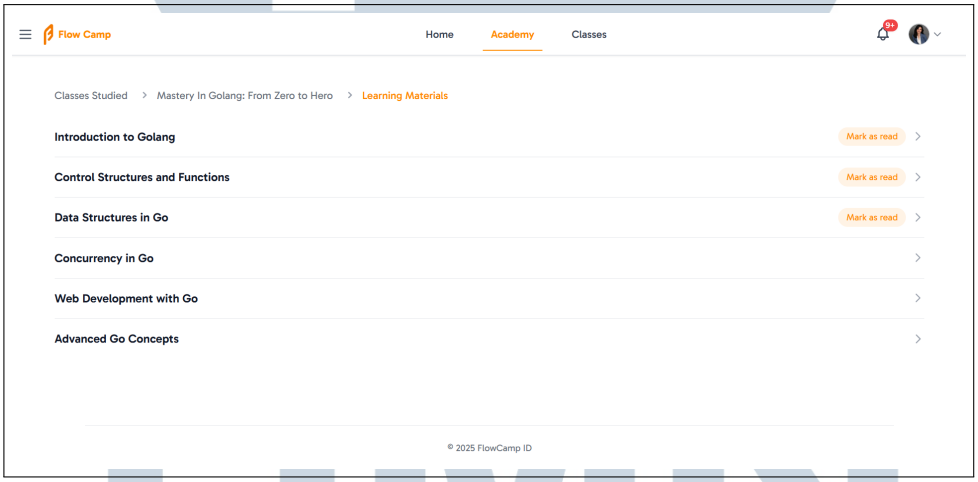
Gambar 3.33. Halaman Detail Completed Classes

D Halaman Pembelajaran Siswa

D.1 Halaman *Learning Materials*

Halaman *Learning Materials* yang berfungsi sebagai *gateway* utama siswa untuk mengakses semua konten pembelajaran dalam sebuah kelas seperti yang terlihat pada gambar 3.34 yang menggunakan komponen *breadcrumb navigation* melalui `StudentBreadcrumb` yang secara dinamis menampilkan *path* berdasarkan status kelas. Daftar materi ditampilkan dalam format *list* dengan setiap item menggunakan *hover effects* yang kompleks, termasuk *background*

`group-hover:bg-orange-50` dengan *positioning* absolut `absolute inset-0 -left-2 sm:-left-6` untuk memberikan efek *highlight* pada seluruh area. Setiap item materi memiliki struktur *flexbox* dengan `flex items-center justify-between` untuk mengatur judul materi dan elemen *status badge* yang menggunakan *styling* `bg-orange/10 text-orange rounded-full px-2 sm:px-3 py-0.5 sm:py-1`. Ikon *chevron* diimplementasikan dengan *SVG* yang memiliki *transform* dan *transition effects* menggunakan kelas `group-hover:translate-x-1 transition-transform duration-200` untuk memberikan animasi *slide* saat *hover*. *Divider lines* ditampilkan menggunakan `border-b border-gray-200` dengan *conditional rendering* `v-if` untuk menghindari garis pada item terakhir, sementara *empty state* menggunakan *styling* terpusat dengan `text-center text-gray-500` dan *responsive padding* melalui `py-6 sm:py-8 px-4 sm:px-6 md:px-8`.



Gambar 3.34. Halaman *Learning Materials*

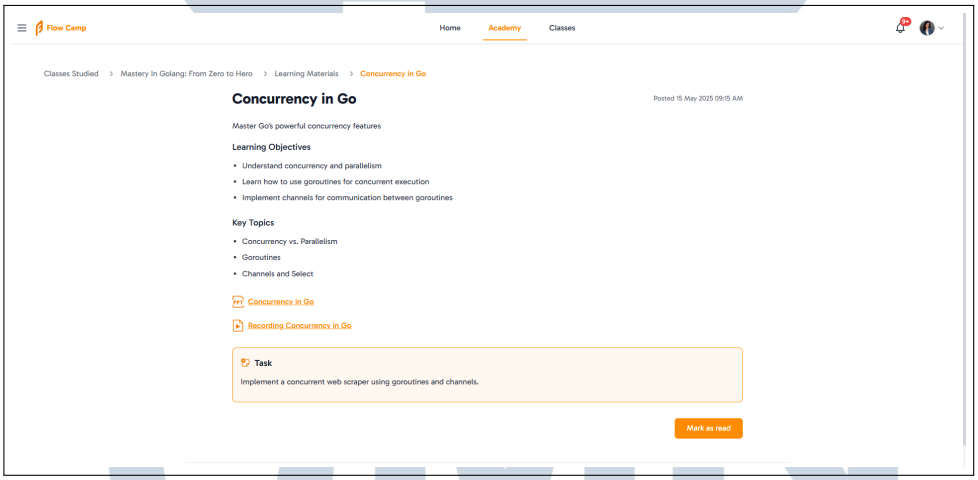
D.2 Halaman *Detail Learning Materials*

Lalu, ketika pengguna mengklik salah satu materialnya, maka sistem akan mengarahkan ke dalam *detail learning material*-nya seperti yang terlihat pada gambar 3.35. Implementasi halaman ini menggunakan komponen *breadcrumb navigation* yang dinamis dan struktur konten yang terorganisir dalam `max-w-5xl mx-auto` untuk pembatasan lebar optimal. *Header section* menggunakan *flexbox responsive* dengan `flex-col sm:flex-row justify-between items-start sm:items-center` untuk mengatur judul materi dan tanggal posting. Konten utama ditampilkan dalam area `prose max-w-none` dengan bagian *Learning Objectives* dan *Key Topics* yang menggunakan *list styling*

`list-disc pl-4 sm:pl-5 space-y-1 sm:space-y-2` untuk organisasi informasi yang jelas. Bagian *Learning Materials* menampilkan file *PPT* dan *video* dengan ikon dan *styling* `text-orange underline cursor-pointer hover:text-orange-dark` untuk memberikan *interactive feedback*. Area *task information* menggunakan *styling* khusus dengan `bg-orange-50 rounded-lg border border-orange` untuk memberikan penekanan visual, sementara tombol "*Mark as read*" diimplementasikan dengan *conditional styling* menggunakan `bg-gray-400 cursor-not-allowed` untuk status sudah dibaca dan `bg-orange hover:bg-orange-dark` untuk status aktif. Integrasi *DialogBox* untuk berbagai *feedback* dan konfirmasi memberikan pengalaman pengguna yang interaktif dengan *modal overlay* dan *transition effects*.

1. Halaman *detail learning materials* jika terdapat *task*

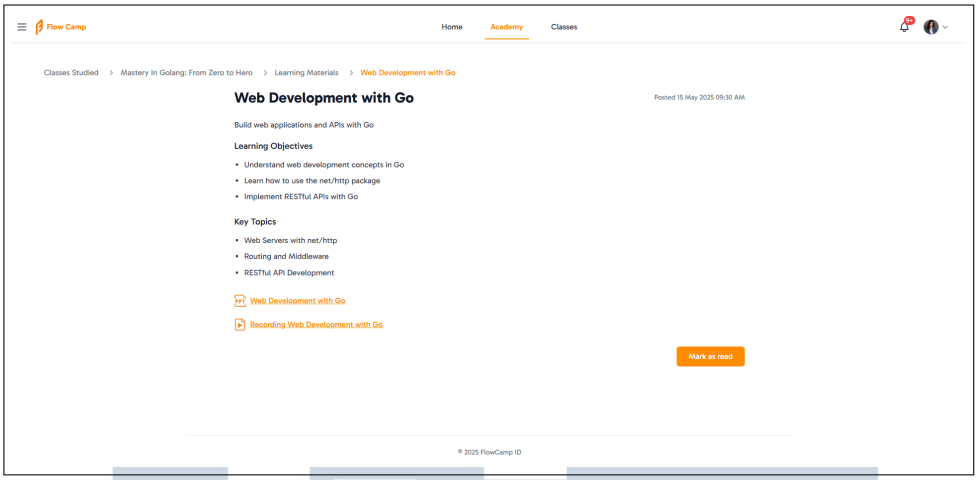
   Gambar 3.35 menunjukkan tampilan jika terdapat *task* pada suatu material.



Gambar 3.35. Halaman *Detail Learning Materials* jika Terdapat *Task*

2. Halaman *detail learning materials* jika tidak terdapat *task*

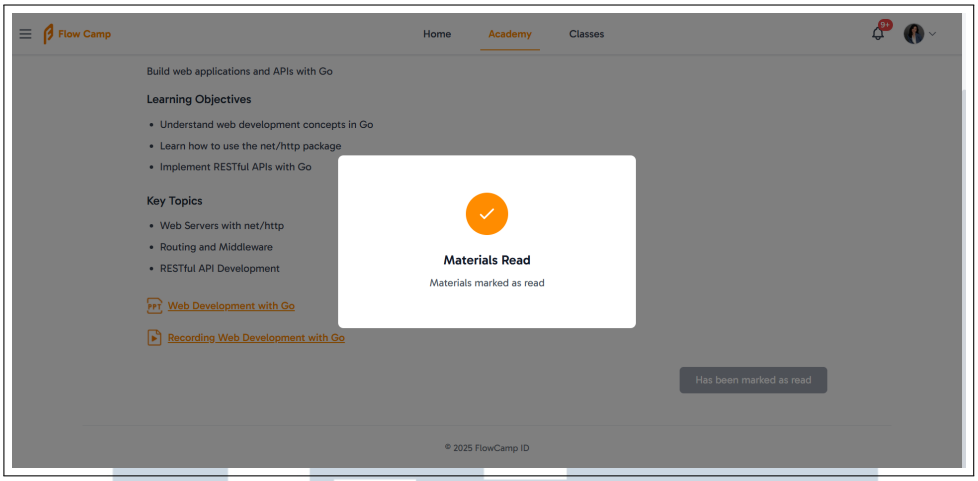
   Ketika suatu material/modul tidak memiliki *task*, maka akan terlihat seperti pada gambar 3.36.



Gambar 3.36. Halaman *Detail Learning Materials* jika Tidak Terdapat *Task*

D.3 *Dialog Box "Mark as read"*

Pada gambar 3.37, implementasi *dialog* "*Mark as Read*" pada komponen DialogBox.vue menggunakan *modal overlay* dengan *fixed positioning* `fixed inset-0 bg-black bg-opacity-50` dan *centering* melalui `flex items-center justify-center z-50`. *Dialog container* menggunakan *styling* `bg-white rounded-lg p-4 sm:p-6 max-w-md w-full` dengan *responsive padding* dan pembatasan lebar maksimal. Area konten utama menampilkan ikon *checkmark* dalam lingkaran oranye menggunakan `rounded-full bg-orange w-12 h-12 sm:w-16 sm:h-16 mx-auto flex items-center justify-center` dengan ikon *SVG* putih berukuran `w-6 h-6 sm:w-8 sm:h-8 text-white`. Teks judul "*Materials Read*" menggunakan *styling* `text-lg sm:text-xl font-semibold mb-2` diikuti dengan deskripsi "*Materials marked as read*" dalam `text-sm sm:text-base text-gray-600`. *Dialog* ini menggunakan *template conditional rendering* dengan `v-if="type === 'markAsRead'"` dan dilengkapi dengan *auto-close timer* yang secara otomatis menutup *dialog* setelah 1 detik menggunakan `setTimeout` untuk memberikan *feedback* visual yang efisien kepada pengguna.



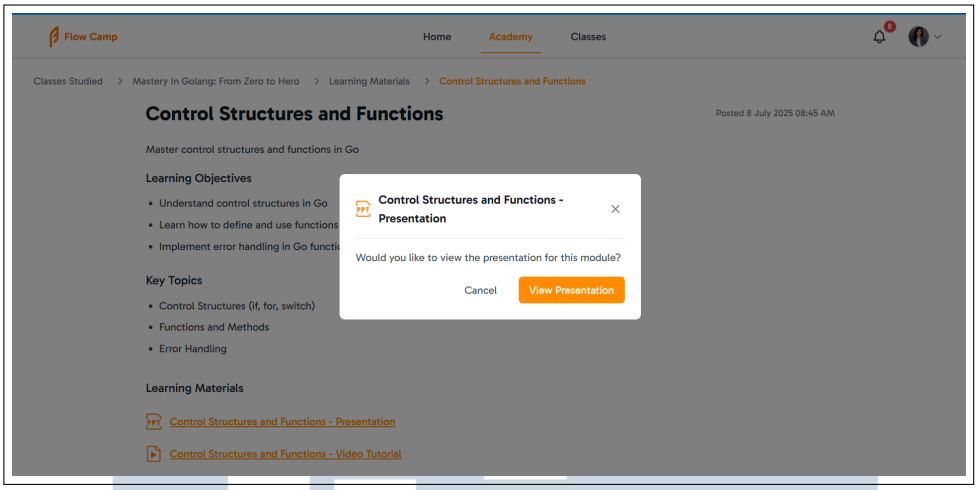
Gambar 3.37. *Dialog Box* "*Mark as read*"

D.4 *Dialog Box* "PPT"

Implementasi *dialog* "*PPT*" pada komponen DialogBox.vue menggunakan *conditional rendering* dengan `v-if="type === 'ppt'"` yang menampilkan dua *state* berbeda berdasarkan ketersediaan konten.

1. *Dialog box* jika terdapat "PPT"

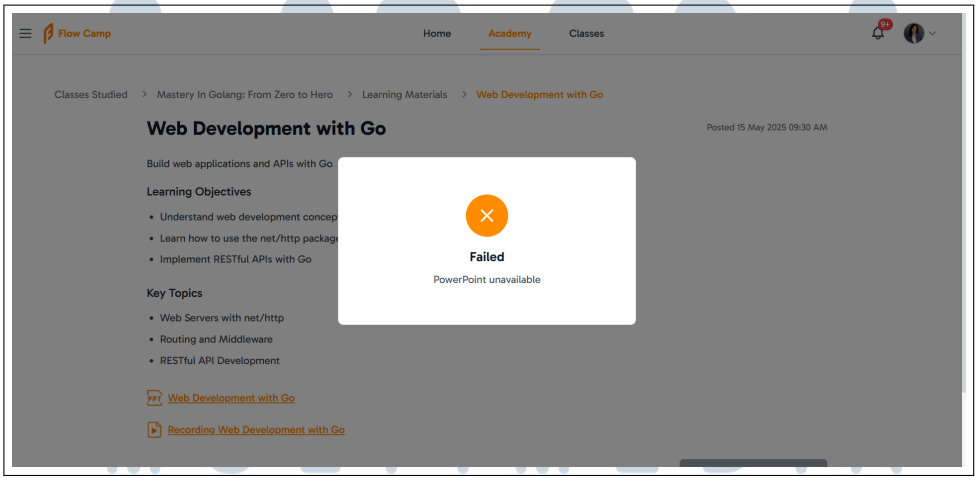
   State tersedia menggunakan *layout* `space-y-3 sm:space-y-4` dengan *header* berisi ikon PPT dari `/ppt.png`, judul dinamis dari `materialInfo?.title`, dan tombol *close* dengan *hover effect* `hover:bg-gray-100`. Area konten menampilkan teks konfirmasi dalam `text-sm sm:text-base text-gray-600` dan *action buttons* menggunakan `flex justify-end space-x-3 sm:space-x-4` dengan tombol "*Cancel*" dan "*View Presentation*" yang menggunakan *styling* `bg-orange text-white` dengan *hover effect* `hover:bg-orange-dark` seperti yang terlihat pada 3.38.



Gambar 3.38. *Dialog Box* jika Terdapat ”PPT”

2. *Dialog box* jika tidak terdapat ”PPT”

   State tidak tersedia menggunakan *centered layout* `text-center py-6 sm:py-8` dengan ikon *X* dalam lingkaran oranye berukuran `w-16 h-16 sm:w-20 sm:h-20`, judul ”*Failed*” dengan *styling* `text-xl sm:text-2xl font-bold`, dan pesan ”*PowerPoint unavailable*” dalam `text-sm sm:text-base text-gray-600` dengan *auto-close timer* 1 detik untuk memberikan *feedback* visual yang konsisten kepada pengguna seperti yang terlihat pada 3.39



Gambar 3.39. *Dialog Box* jika Tidak Terdapat ”PPT”

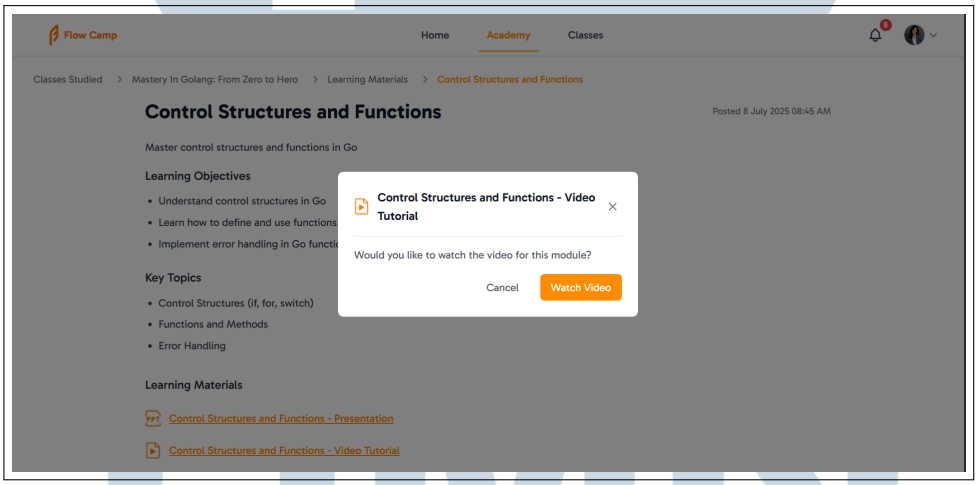
D.5 *Dialog Box ”Video”*

Implementasi *dialog* ”*Video*” pada komponen DialogBox.vue menggunakan *conditional rendering* dengan `v-if="type === 'video'"` yang menampilkan dua

state berbeda berdasarkan ketersediaan konten.

1. *Dialog box* jika terdapat ”Video”

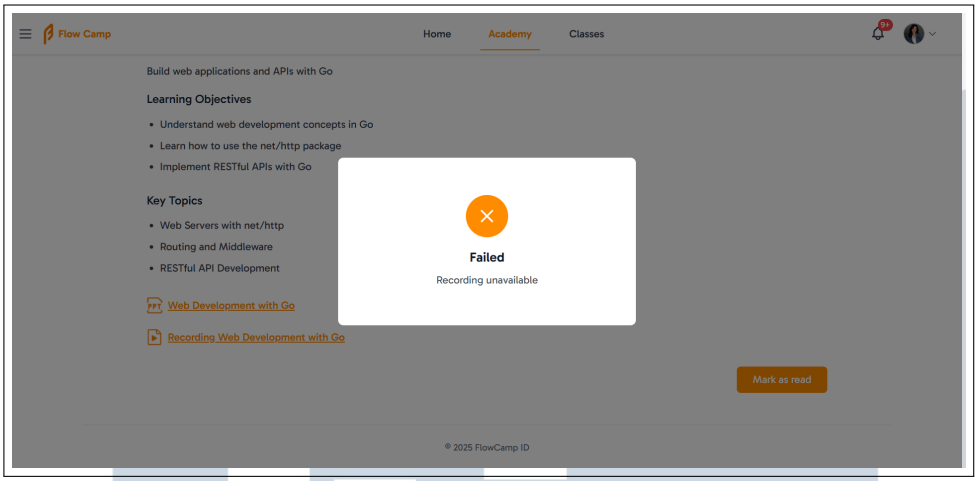
   State tersedia menggunakan *layout* `space-y-3 sm:space-y-4` dengan *header* berisi ikon video dari `/video.png`, judul dinamis dari `materialInfo?.title || 'Watch Video'`, dan tombol *close* dengan *hover effect* `hover:bg-gray-100`. Area konten menampilkan teks konfirmasi ”*Would you like to watch the video for this module?*” dalam `text-sm sm:text-base text-gray-600` dan *action buttons* menggunakan `flex justify-end space-x-3 sm:space-x-4` dengan tombol ”*Cancel*” dan ”*Watch Video*” yang menggunakan *styling* `bg-orange text-white` dengan *hover effect* `hover:bg-orange-dark`. Implementasi tampilannya terlihat pada gambar 3.40 berikut.



Gambar 3.40. *Dialog Box* jika Terdapat ”Video”

2. *Dialog box* jika tidak terdapat ”Video”

   State tidak tersedia menggunakan *centered layout* `text-center py-6 sm:py-8` dengan ikon *X* dalam lingkaran oranye berukuran `w-16 h-16 sm:w-20 sm:h-20`, judul ”*Failed*” dengan *styling* `text-xl sm:text-2xl font-bold`, dan pesan ”*Recording unavailable*” dalam `text-sm sm:text-base text-gray-600` dengan *auto-close timer* 1 detik untuk memberikan *feedback* visual yang konsisten kepada pengguna seperti yang terlihat pada gambar 3.41.



Gambar 3.41. *Dialog Box* jika Tidak Terdapat ”Video”

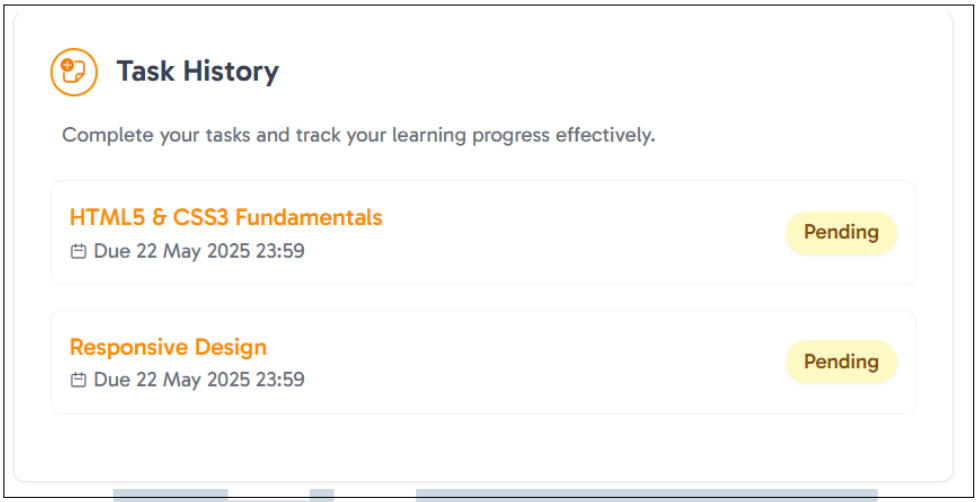
E Halaman Penugasan

E.1 *Task History*

Tampilan ”Task History” ini merupakan tampilan untuk *tracking* dan mengelola tugas-tugas dalam kelas yang sedang berlangsung yang memberikan siswa *overview* yang jelas tentang status *task* dan level urgensinya.

1. Tampilan *task history* saat terdapat *task*

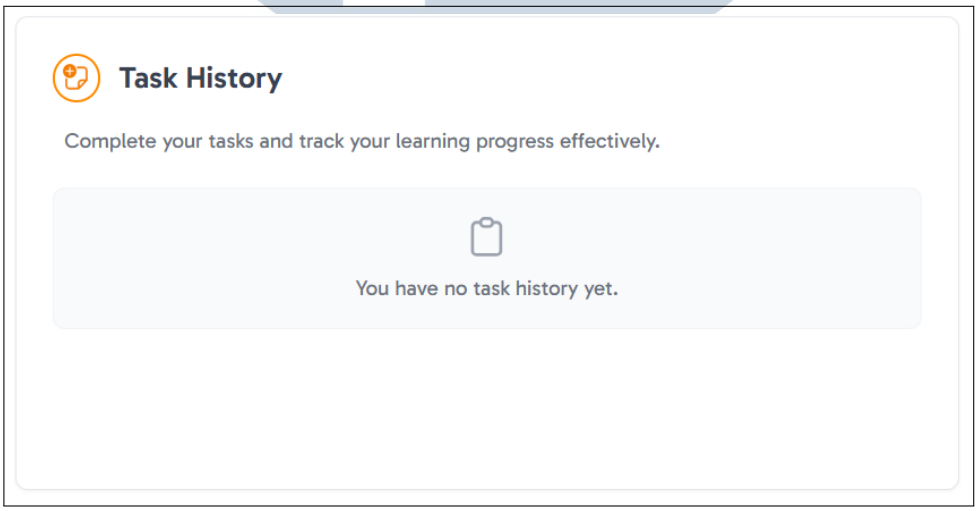
   Ketika pada tampilan detail kelas yang sedang berlangsung dan pengguna memiliki *task* untuk dikerjakan, maka akan terlihat seperti pada gambar 3.42. Untuk menampilkan *task* yang ada disini, dilakukan implementasi *filtering* menggunakan *multiple criteria* untuk menampilkan hanya tugas-tugas yang relevan untuk *student*. Sistem *filtering* menggunakan *conditional logic* yang mempertimbangkan *hasTask=true*, *isRead=true*, dan status *filtering* untuk *pending*. Lalu, pengguna juga dapat mengklik *task* yang ada untuk beralih ke halaman detail *task* nya yang menggunakan *hover* dan transisi.



Gambar 3.42. Tampilan *Task History* saat Terdapat *Task*

2. Tampilan *task history* saat tidak terdapat *task*

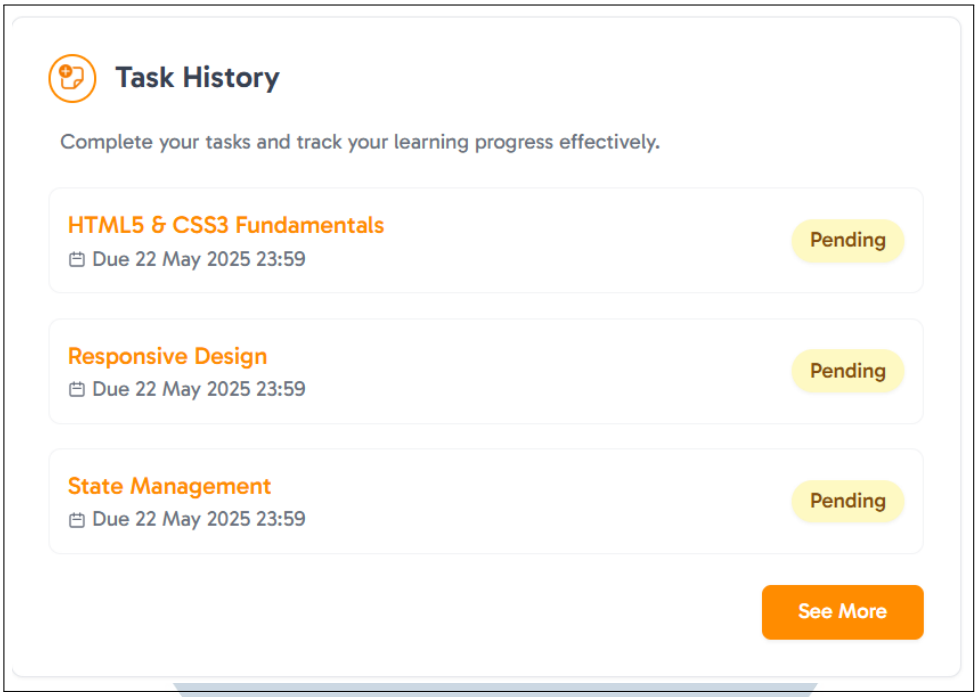
   Ketika pada tampilan *task history* tersebut tidak memiliki *task*-nya, maka akan terlihat seperti pada gambar 3.43.



Gambar 3.43. Tampilan *Task History* saat Tidak Terdapat *Task*

3. Tampilan *task history* saat terdapat lebih dari 3 *task*

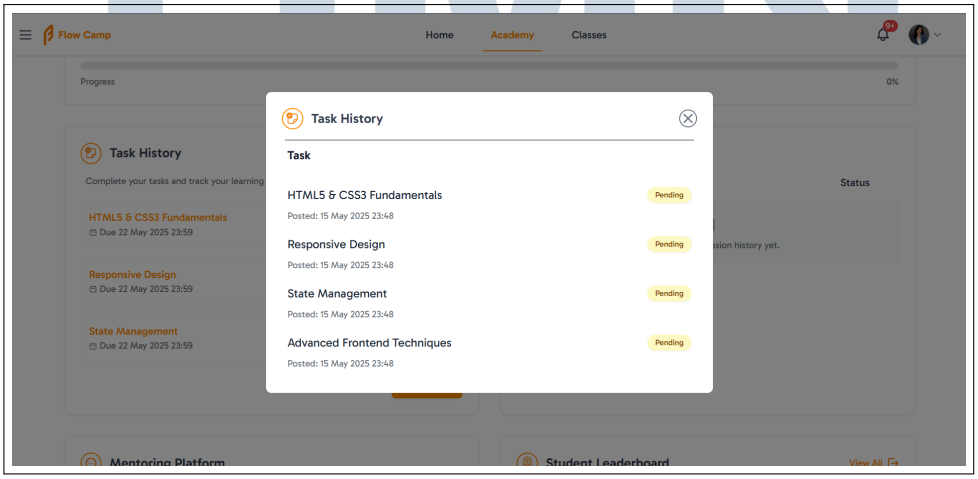
   Pada gambar 3.44 menujukkan bahwa saat jumlah *task* melebihi 3 pada *task history*, maka akan muncul tombol ”See More” dengan menggunakan *hover* yang ketika diklik akan menampilkan komponen *dialog box* yang terlihat pada bagian selanjutnya.



Gambar 3.44. Tampilan *Task History* saat Terdapat Lebih dari 3 *Task*

E.2 *Dialog Box "More Task History"*

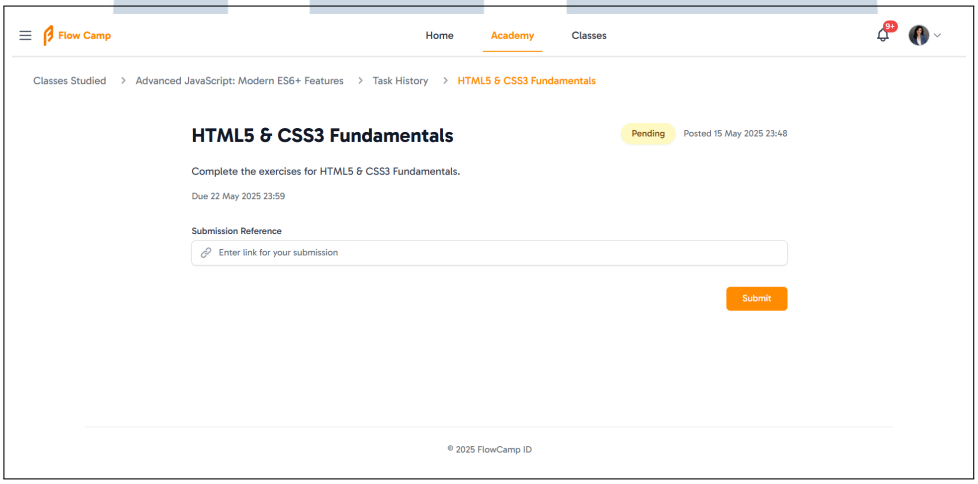
Ketika pengguna mengklik tombol "See More" yang ada di bagian *task history*, maka akan muncul tampilan seperti pada gambar 3.45. Pada bagian ini, akan menampilkan semua *task* yang ada di bagian *task history*-nya berdasarkan waktu *due date* yang terbaru. Pengguna juga dapat mengklik task tersebut yang mengarahkan ke bagian detail *task*-nya.



Gambar 3.45. *Dialog Box "More Task History"*

E.3 Halaman Detail *Task History*

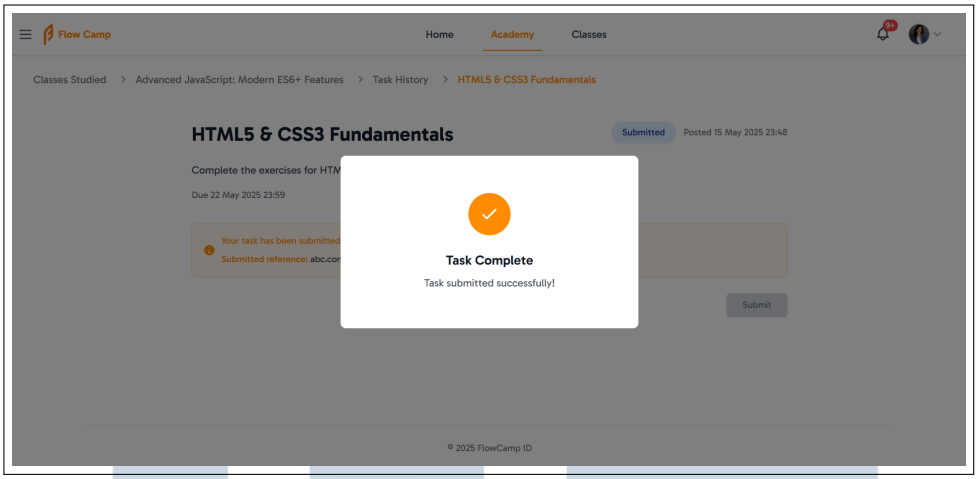
Seperti yang terlihat pada gambar 3.46, tampilan ini merupakan halaman saat pengguna melihat detail *task* yang ada di halaman detail kelas maupun pada *dialog box task history*-nya. Pada bagian ini menampilkan informasi mengenai detail *task*-nya dan memiliki form untuk mengumpulkan hasil *task* penggunanya dengan memasukkan *link* yang kemudian bisa disubmit menggunakan tombol ”submit” yang nanti akan menampilkan *dialog box* dan akan mengubah *status task*-nya menjadi ”*Submitted*”.



Gambar 3.46. Halaman Detail Task History

E.4 *Dialog Box ”Task Complete”*

Ketika pengguna melakukan *submit*, maka *dialog box* akan muncul seperti yang terlihat pada gambar 3.47 yang menunjukkan bahwa *task*-nya sudah terkumpul dengan baik.



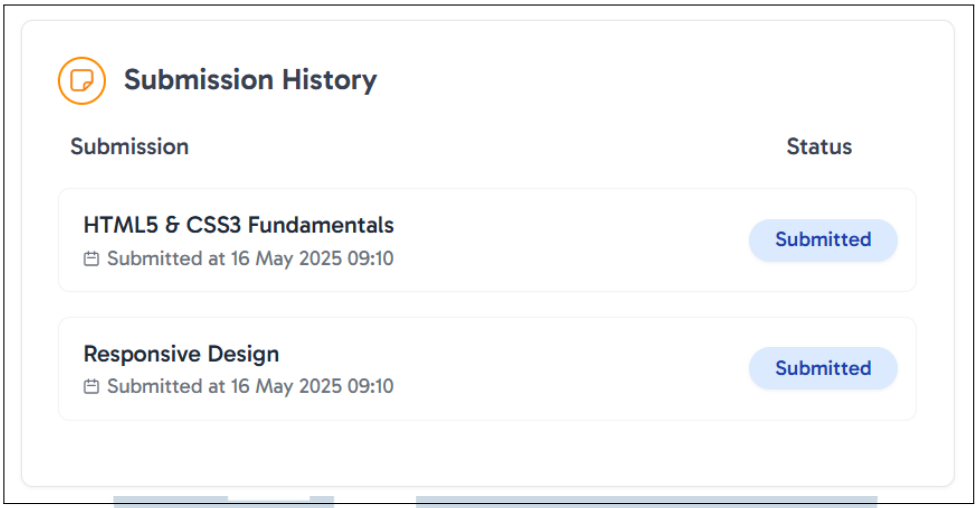
Gambar 3.47. *Dialog Box "Task Complete"*

E.5 *Submission History*

Submission history ini berfungsi sebagai sistem *tracking* untuk menampilkan riwayat pengumpulan tugas yang telah disubmit oleh siswa dalam kelas yang sedang berlangsung dengan implementasi sebagai berikut.

1. Tampilan *submission history* saat terdapat *task*

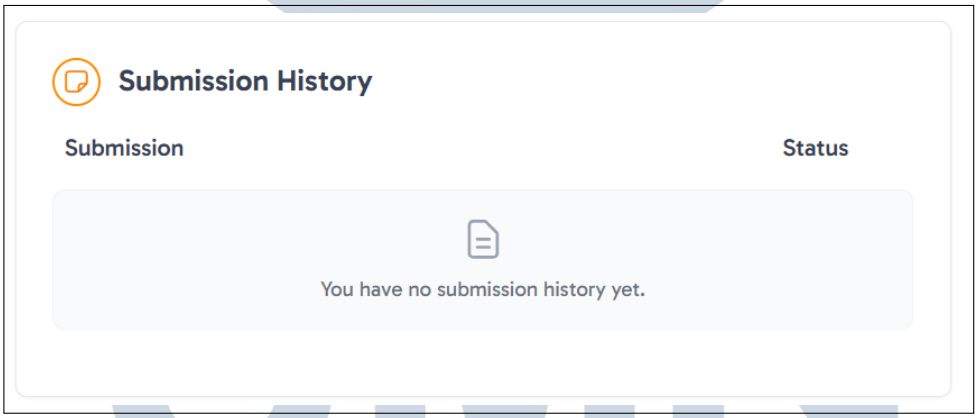
   Pada bagian yang terlihat pada gambar 3.48 memiliki tampilan yang mirip seperti pada *task history*. Namun pada *submission history*, *task* yang ditampilkan difilter berdasarkan status selain *pending*, yaitu "Submitted" "Past Due", "Under Review", dan "Reviewed", dan tampilannya tetap diurutkan berdasarkan waktu *submit* nya. Lalu, untuk yang status *task past due* dilakukan prioritas, sehingga berada di paling atas. *Task* berstatus *past due* ini adalah *task* yang sudah melewati batas pengumpulan, namun siswa masih dapat mengumpulkan *task* tersebut dengan status *task* sebagai



Gambar 3.48. Tampilan *Submission History* saat Terdapat *Task*

2. Tampilan *submission history* saat tidak terdapat *task*

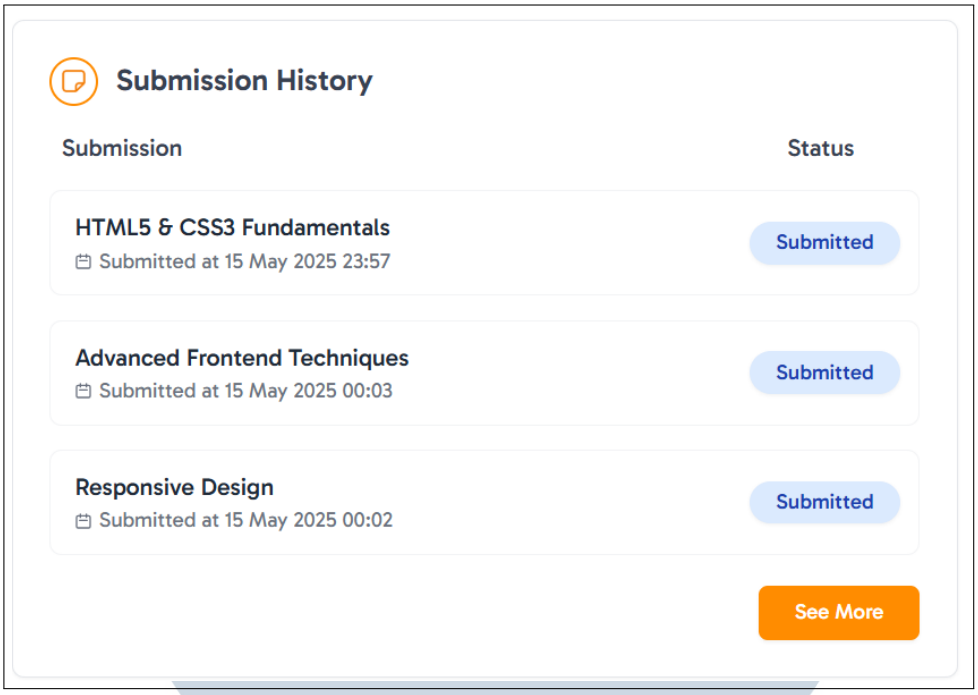
   Ketika pada tampilan *submission history* tersebut tidak memiliki *task*-nya, maka akan terlihat seperti pada gambar 3.49.



Gambar 3.49. Tampilan *Submission History* saat Tidak Terdapat *Task*

3. Tampilan *submission history* saat terdapat lebih dari 3 *task*

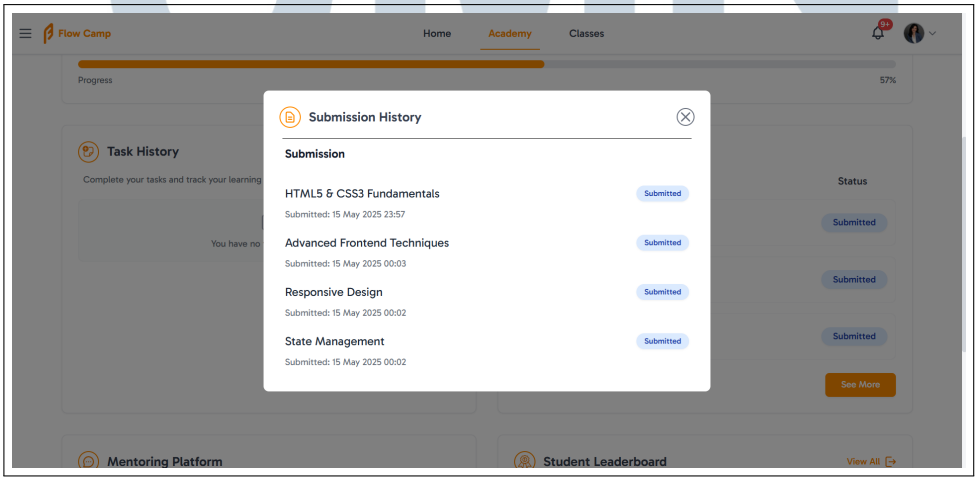
   Pada gambar 3.50 menujukkan bahwa saat jumlah *task* melebihi 3 pada *submission history*, maka akan muncul tombol ”See More” dengan menggunakan *hover* yang ketika diklik akan menampilkan komponen *dialog box* yang terlihat pada bagian selanjutnya.



Gambar 3.50. Tampilan *Submission History* saat Terdapat Lebih dari 3 *Task*

E.6 *Dialog Box "More Submission History"*

Ketika pengguna mengklik *button "See More"* yang ada di bagian *submission history*, maka akan muncul tampilan seperti pada gambar 3.51. Pada bagian ini, akan menampilkan semua *task* yang ada di bagian *submission history*-nya berdasarkan waktu *submit* yang terbaru dengan tetap memprioritaskan pada *task* yang *status*-nya *past due*. Pengguna juga dapat mengklik *task* tersebut yang mengarahkan ke bagian *detail task*-nya.

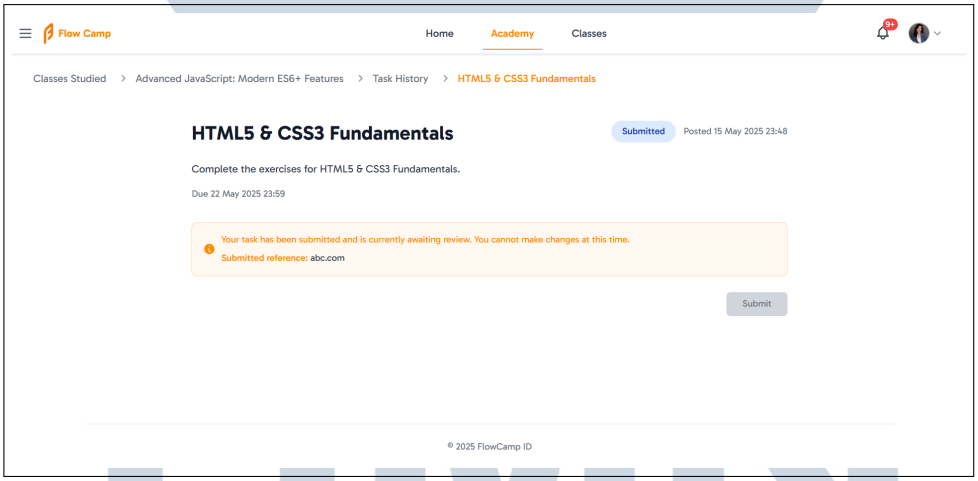


Gambar 3.51. *Dialog Box "More Submission History"*

E.7 Halaman Detail Submission History

Seperti yang terlihat pada gambar 3.52, tampilan ini merupakan halaman saat pengguna melihat *detail task* yang ada di halaman detail kelas maupun pada *dialog box submission history*-nya. Pada bagian ini menampilkan *breadcrumb* yang dinamis berdasarkan sumber halaman yang aktif, lalu terdapat informasi mengenai *task*-nya yang dibedakan berdasarkan status *task*-nya, di mana pada bagian ini merupakan tampilan untuk yang statusnya ”Submitted” yang terlihat pada bagian *badge*-nya, lalu pada bagian ini memiliki informasi bahwa *task*-nya sudah terkumpul dengan baik dengan memperlihatkan *link* yang sudah dikumpulkan oleh siswa, dan *button submit*-nya akan berubah warna menjadi abu-abu dan tidak bisa diklik.

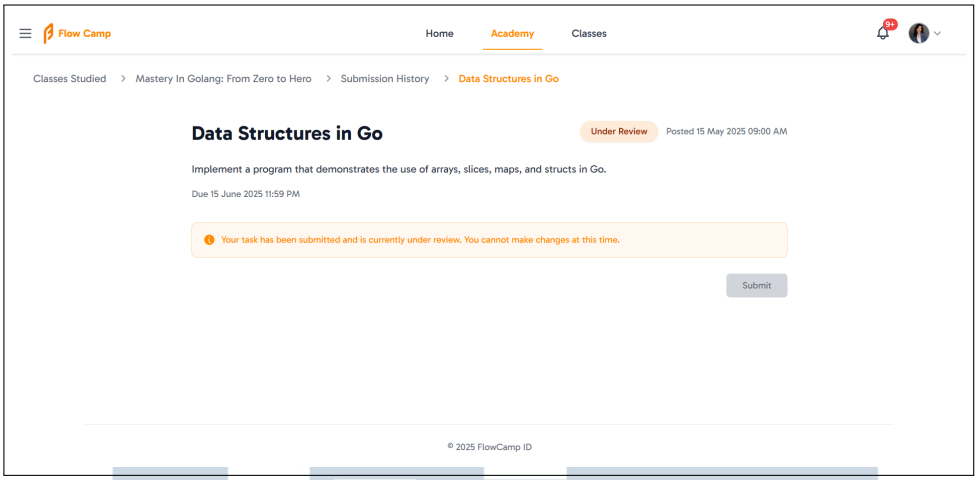
1. Halaman Detail *Submission History Status* ”*Submitted*”



Gambar 3.52. Halaman Detail Submission History Status ”*Submitted*

2. Halaman Detail *Submission History Status* ”*Under Review*”

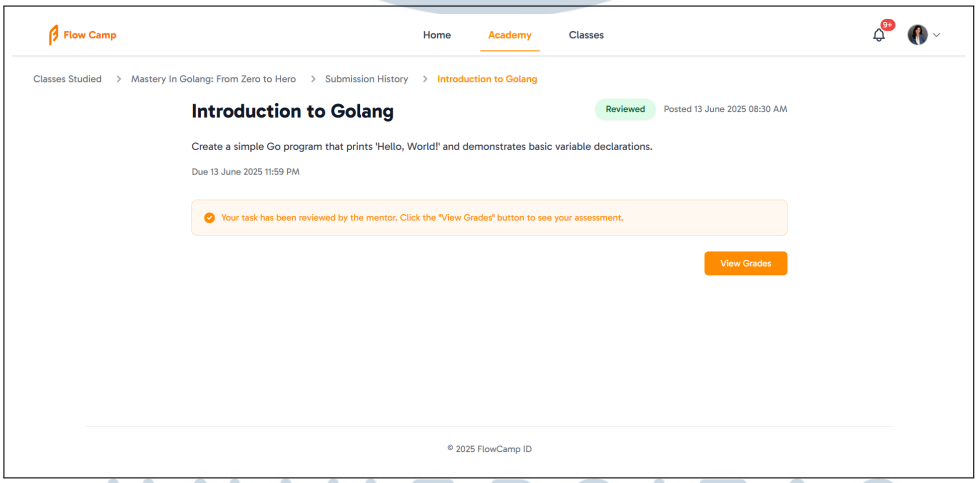
   Yang membedakan halaman detail *submission history* yang statusnya ”Under Review” adalah informasi *task*-nya sedang di*review* dan siswa hanya bisa menunggu *task*-nya dinilai oleh mentornya, serta tombol *submit*-nya berubah warna menjadi abu-abu yang tidak bisa diklik oleh siswanya yang terlihat pada gambar 3.53 berikut.



Gambar 3.53. Halaman Detail Submission History Status ”*Under Review*

3. Halaman Detail *Submission History Status* ”*Reviewed*”

   Lalu, untuk tampilan detail *submission history* berstatus ”Reviewed” yang membedakannya adalah informasi *task*-nya bahwa sudah dinilai oleh mentornya, dan terdapat tombol ”View Grades” untuk melihat nilai *task* tersebut.



Gambar 3.54. Halaman Detail *Submission History Status* ”*Reviewed*

Ketika siswa mengklik tombol tersebut, maka akan menampilkan *dialog box* seperti yang terlihat pada gambar 3.55. *Dialog box* ini menampilkan informasi nilai dengan deskripsi yang dibagi menjadi 5 kondisi, yaitu sebagai berikut.

(a) Nilai lebih dari 90, maka deskripsinya ”Excellent!” dengan tema warna hijau.

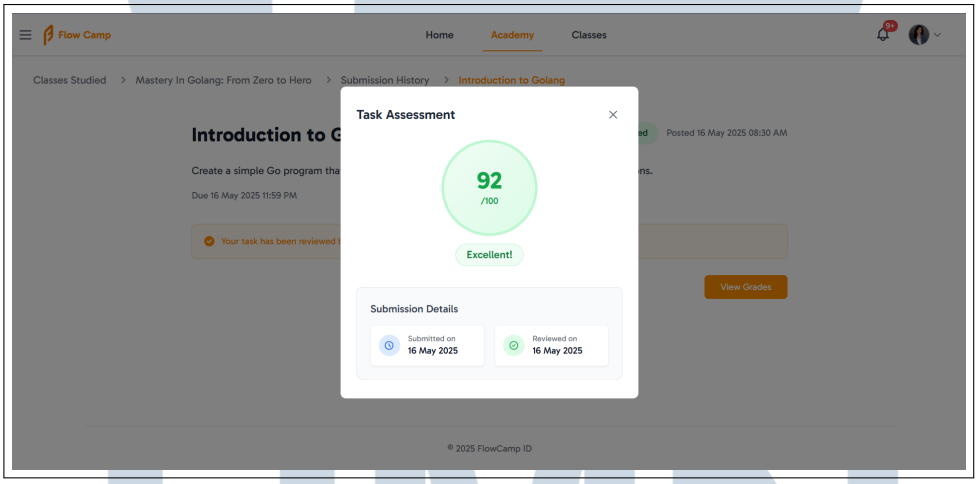
(b) Nilai diantara 80 - 89, maka deskripsinya ”Great Work!” dengan tema warna biru.

(c) Nilai diantara 70 - 79, maka deskripsinya ”Good Job!” dengan tema warna ungu.

(d) Nilai diantara 60 - 69, maka deskripsinya ”Satisfactory” dengan tema warna kuning.

(e) Nilai kurang dari 60, maka deskripsinya ”Needs Improvement” dengan tema warna merah.

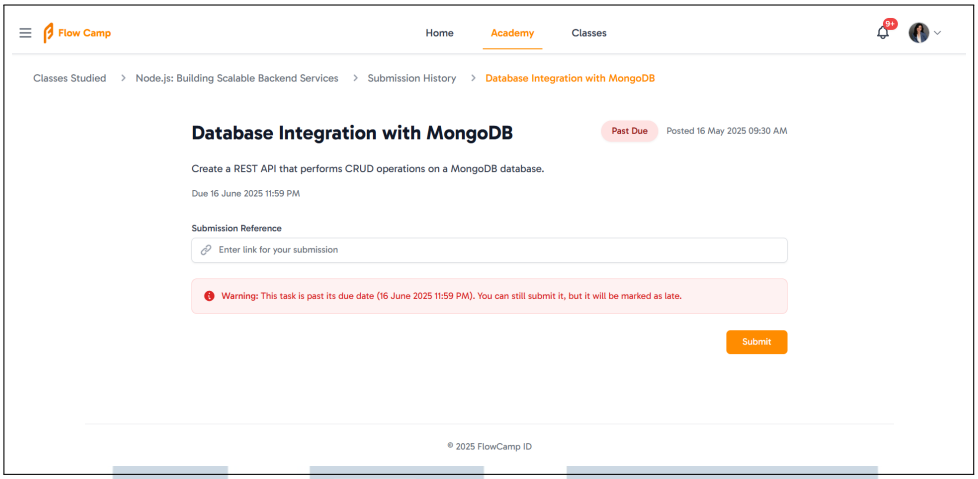
Lalu, terdapat juga informasi tanggal bahwa kapan *task* tersebut disubmit oleh siswa dan di*review* oleh mentornya.



Gambar 3.55. *Dialog Box* ”*View Grades*”

4. Halaman Detail Submission History Status ”*Past Due*”

Ketika siswa melihat *detail task* yang berstatus *past due* menandakan bahwa *task* tersebut sudah melewati batas pengumpulan akan terlihat seperti pada gambar 3.56. Pada bagian ini, terdapat informasi *task* dan terdapat *form* yang dapat diinput *link task* siswa. Meskipun sudah melewati batas pengumpulan, tapi siswa masih dapat melakukan *submit* dengan mengklik tombol ”Submit” nya yang memiliki *hover*. dan ketika sudah di*submit*, maka tampilannya akan menjadi status ”Submitted”.

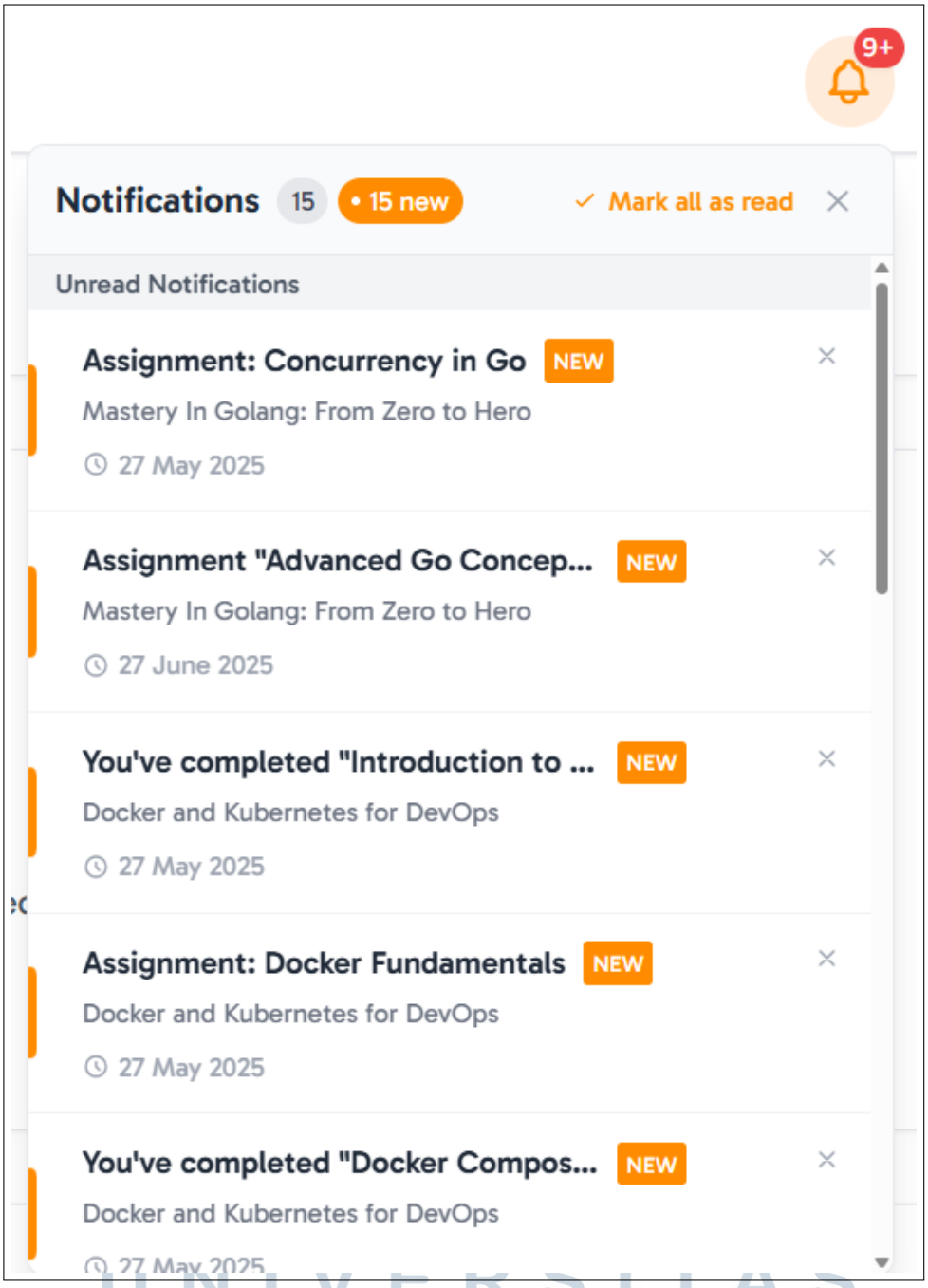


Gambar 3.56. Halaman Detail Submission History Status ”*Past Due*

F Tampilan Notifikasi

Pada gambar 3.57, proses generasi notifikasi hanya mencakup material yang telah dibaca oleh mahasiswa dan memiliki komponen tugas (hasTask === true), memastikan bahwa notifikasi yang ditampilkan relevan. Kemudian, *badge* notifikasi menampilkan visual *indicator* yang dinamis dengan *counter* untuk notifikasi yang belum dibaca. Implementasi menggunakan *computed property*, di mana *badge* secara otomatis *update* ketika ada perubahan status notifikasi. Visual *design badge* menggunakan animasi *subtle-pulse* dan transisi untuk memberikan *feedback* visual. *Badge* menampilkan angka hingga 9, dengan format ”9+” ketika jumlah notifikasi-nya melebihi 9. Lalu, *dropdown* notifikasi mengimplementasikan segmentasi yang jelas antara notifikasi ”unread” dan ”read”. Sistem *sorting* otomatis memastikan notifikasi *unread* selalu muncul di atas, diikuti dengan *sorting* berdasarkan *timestamp*. Setiap *item* notifikasi menampilkan informasi komprehensif termasuk judul tugas, nama kelas, status *badge*, dan *timestamp*-nya. Ketika siswa mengklik notifikasi, sistem langsung mengupdate status *read* dan melakukan *re-sorting* untuk memberikan *feedback* instan. Lalu, terdapat implementasi *processing state* dengan *ID tracking* untuk memastikan tidak ada *race condition* atau *duplicate processing* ketika *user* melakukan *multiple clicks* dalam waktu singkat. Kemudian, siswa juga dapat mengklik data material notifikasinya, sehingga sistem akan mengarahkan ke detail masing-masing materialnya. Setiap navigasi menyertakan *query parameters* yang memberikan konteks tambahan untuk halaman tujuan. Lalu ketika ada perubahan status tugas, sistem secara otomatis meregenerasi notifikasi dan mengupdate *cleared notifications list* jika diperlukan.

Ini memastikan notifikasi selalu *up-to-date* dengan data terbaru tanpa memerlukan manual *refresh* dari *user*. Lalu, pada bagian ini juga diimplementasikan *lazy loading* untuk setiap 100 data yang ditampilkan. Kemudian, siswa juga dapat mengklik ”Mark all as read” yang menandakan bahwa semua notifikasi yang ada di bagian tersebut akan dianggap sudah dibaca semua oleh sistem, sehingga *counter*-nya akan menyesuaikan. Kemudian tanda *cross*-nya menandakan bahwa siswa dapat menghapus salah satu notifikasi maupun jika siswa mengklik tanda *cross* yang berada di samping tulisan ”mark all as read”, maka akan menghapus semua notifikasinya.



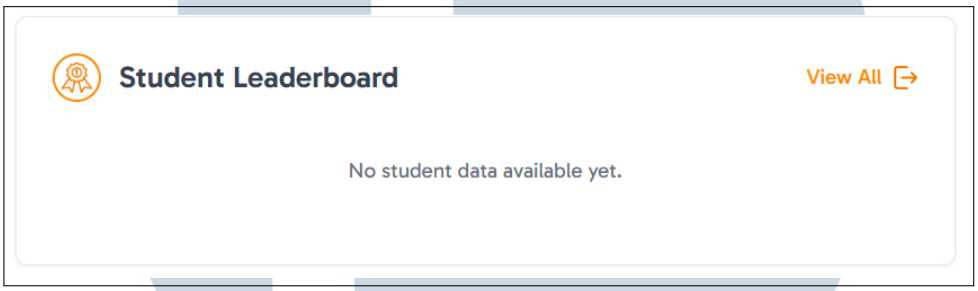
Gambar 3.57. Tampilan Notifikasi

G Tampilan *Student Leaderboard*

Pada halaman detail kelas yang sedang dipelajari, terdapat tampilan dua kondisi dan halaman detail dari *leaderboard* yang dihadirkan pada bagian *student leaderboard* sebagai berikut.

1. Tampilan *student leaderboard* saat tidak terdapat *student*

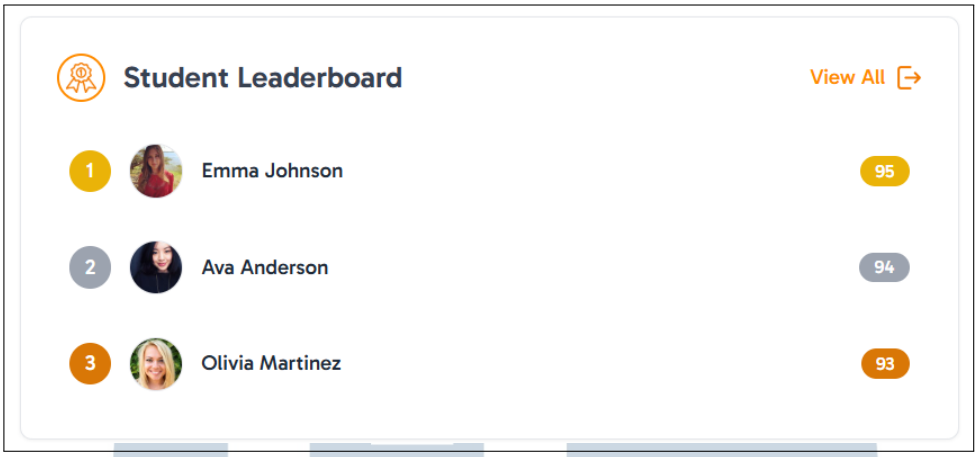
Tampilan *leaderboard* pada detail kelas ketika tidak ada data *student* yang tersedia terlihat pada gambar 3.58. Pada bagian ini, sistem akan menampilkan pesan "*No student data available yet*." Hal ini bisa terjadi jika belum ada siswa yang mendaftar atau belum mengumpulkan tugas. Tampilan ini tetap mempertahankan struktur visual dengan judul "Student Leaderboard" di bagian atas serta tombol "View All" di sisi kanan, tetapi tanpa daftar peringkat di dalamnya.



Gambar 3.58. Tampilan *Student Leaderboard* saat Tidak Terdapat *Student*

2. Tampilan *student leaderboard* saat terdapat *student*

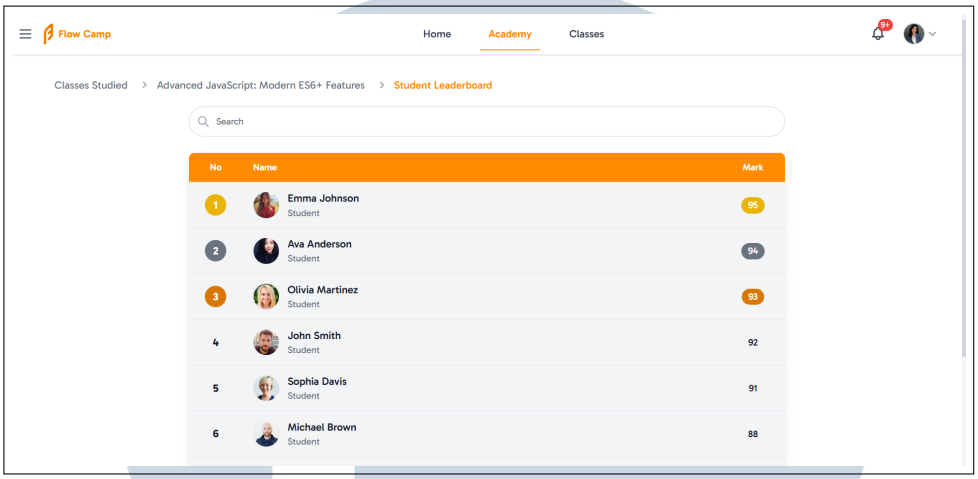
Lalu, tampilan *leaderboard* ketika sudah terdapat siswa yang terdaftar dan sudah memiliki nilai tugas pada task detail kelas tersebut terlihat ditunjukkan pada gambar 3.59. Pada tampilan ini, *leaderboard* menyajikan peringkat tiga teratas yang ditampilkan secara visual dengan mencantumkan foto profil siswa, nama lengkap, urutan peringkat yang ditandai menggunakan angka dengan latar warna berbeda, serta skor atau nilai total yang diperoleh oleh masing-masing siswa. Urutan peringkat disusun secara menurun berdasarkan nilai tertinggi ke nilai terendah. Tampilan ini tidak hanya meningkatkan semangat kompetisi antar siswa, tetapi juga memberikan apresiasi visual terhadap siswa dengan performa terbaik.



Gambar 3.59. Tampilan *Student Leaderboard* saat Terdapat *Student*

Kemudian, ketika siswa mengklik tombol ”View All” yang ada pada detail kelasnya, maka sistem akan mengarahkan siswa ke halaman *leaderboard* dengan informasi yang lebih lengkap, di mana semua siswa yang sudah mendaftar dan *task*-nya sudah dinilai, maka semuanya akan terlihat di *leaderboard ranking* ini seperti yang terlihat pada gambar 3.60. Pada halaman ini, ditampilkan daftar peringkat siswa berdasarkan skor tertinggi yang mereka peroleh selama mengikuti pembelajaran. Setiap baris menampilkan informasi berupa nomor urut, nama lengkap siswa, status siswa, dan skor total yang mereka capai. Tiga siswa dengan skor tertinggi tetap ditandai dengan warna khusus dan ikon medali untuk membedakan peringkat 1 hingga 3, sementara siswa lainnya ditampilkan dalam urutan biasa. Terdapat pula fitur pencarian di bagian atas untuk memudahkan pencarian siswa tertentu dengan menggunakan *filtering* dalam daftar *leaderboard*. Algoritma pencarian ini juga mengimplementasikan *case-insensitive matching* terhadap nama siswa dengan menggunakan *toLowerCase()* dan *includes() methods*. Lalu, terdapat *search state detection* (isSearchActive) yang mempengaruhi visual *styling* di mana *search mode* menghilangkan warna untuk *ranks* 1 sampai 3, dan menampilkan semua *results* dengan *uniform styling* untuk fokus pada *search results*. Lalu, pada bagian ini juga tetap terdapat *breadcrumbs* yang mengikuti halaman aktif siswa.

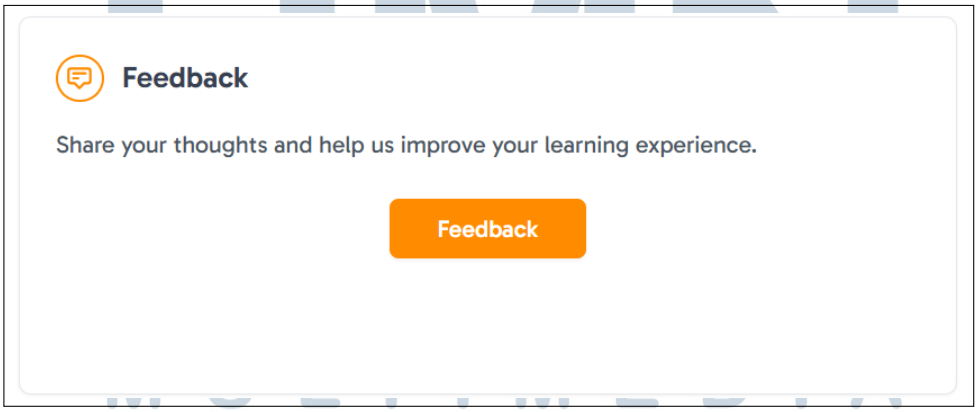
G.1 Halaman Detail *Student Leaderboard*



Gambar 3.60. Halaman Detail *Student Leaderboard*

H Tampilan Umpan Balik/*Feedback*

Tampilan dari bagian *feedback* ini dapat dilihat pada detail kelas setelah siswa menyelesaikan seluruh materi pembelajaran atau pada *completed classes* seperti pada gambar 3.61. Pada tampilan ini, pengguna diperlihatkan tampilan sederhana berisi ajakan untuk memberikan *feedback*, serta tombol berlabel "Feedback" berwarna oranye. Tombol *Call-to-action* tersebut menggunakan *prominent orange styling* dengan *hover effects* dan transisi animasi.

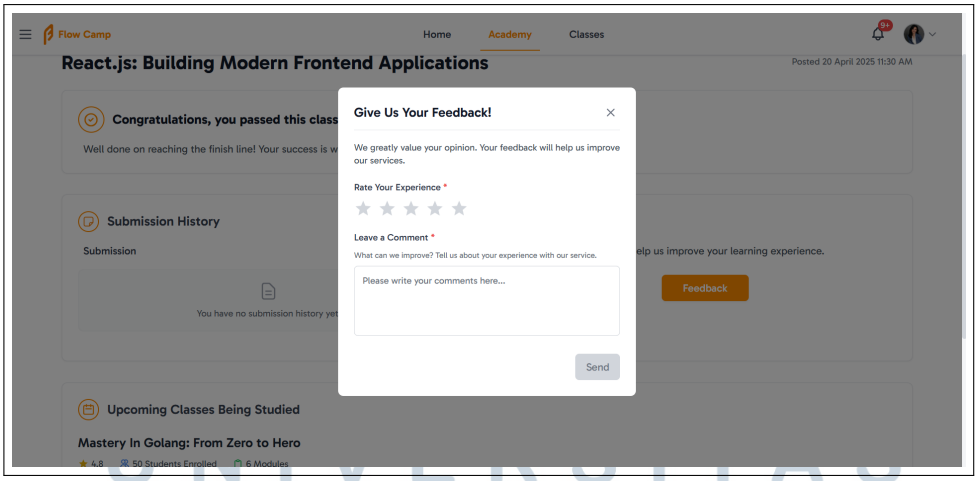


Gambar 3.61. *Tampilan Feedback*

H.1 *Dialog Box* "Konfirmasi *Feedback*"

Ketika pengguna mengklik tombol "Feedback" yang ada pada gambar sebelumnya, maka akan memunculkan *dialog box* yang terlihat pada gambar 3.62.

Pada bagian ini, diimplementasikan modal menggunakan *fixed positioning* dengan *full viewport overlay* (bg-black bg-opacity-50). *Dialog container* menggunakan *centered flexbox layout* dengan *backdrop blur effect* yang mencegah interaksi dengan elemen *background*. Kemudian, Z-index stacking (z-50) juga memastikan modal selalu *appear* di *top layer* dari *UI hierarchy*. Pada *dialog box* tersebut, pengguna dapat memberikan penilaian berupa *rating* bintang dari 1 sampai 5, serta menuliskan komentar secara bebas mengenai pengalaman belajar mereka. Pada bagian ini diimplementasikan *isFormValid computed property* menggunakan *reactive checking* terhadap *rating* (> 0) dan teks komen agar siswa mengisinya terlebih dahulu. Kemudian tampilan validasi erornya menggunakan *conditional rendering* yang muncul hanya setelah *showValidation flag*-nya aktif. Eror tersebut menggunakan warna teks merah dengan teks panduan yang jelas agar siswa mengisi semua *form* yang dibutuhkannya terlebih dahulu. Lalu, ketika pengguna belum mengisi semua *form* maupun bintang pada *dialog box* ini, maka warna dari tombol ”Send”-nya tidak akan berubah menjadi oranye dan tombolnya tidak bisa diklik. Namun, jika sudah mengisi semua kebutuhannya, maka tombol ini akan berubah menjadi oranye dan kemudian bisa diklik. Terdapat *placeholder* juga yang membantu siswa mengetahui hal yang harus diinput oleh siswa tersebut.

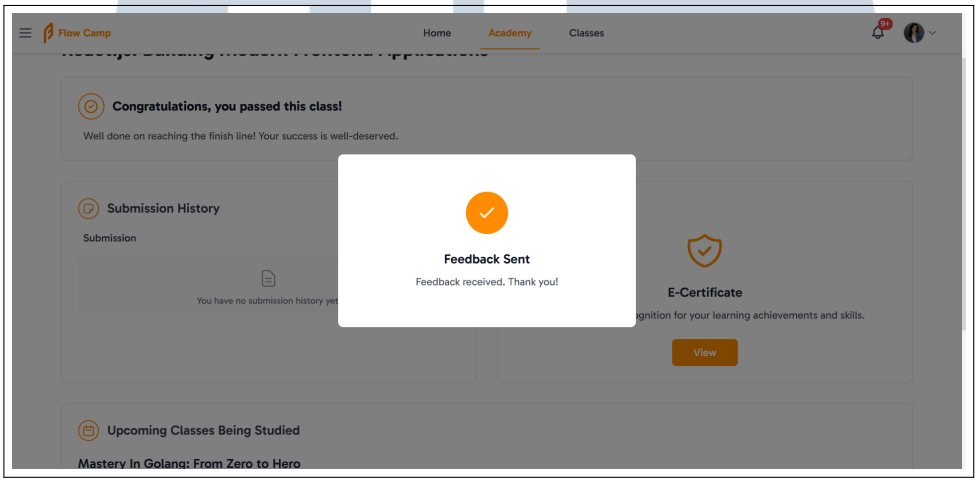


Gambar 3.62. *Dialog Box* ”Konfirmasi *Feedback*”

H.2 *Dialog Box ”Feedback Sent”*

Kemudian, seperti yang terlihat pada gambar 3.63 menunjukkan tampilan komponen *dialog box* dengan pesan “*Feedback Sent*” setelah pengguna berhasil mengirimkan *feedback* dengan mengklik tombol ”Send”. Implementasi *dialog box ”Feedback Sent”* ini menggunakan *success pattern* yang konsisten yang

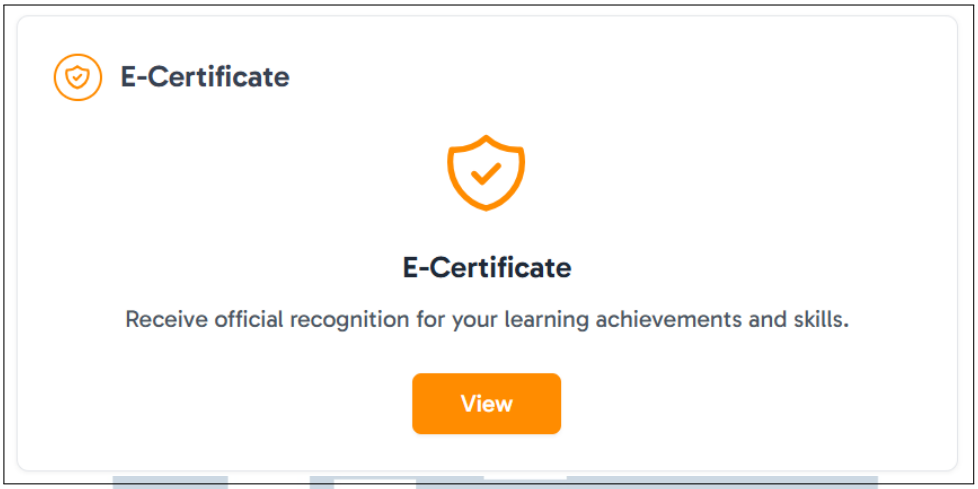
sejalan dengan *dialog success* lainnya. Tampilan ini menjadi penanda bahwa *feedback* tersebut sudah berhasil dikirim. Komponen *dialog box* ini juga akan menghilang/menutup selama 1 detik menggunakan *set timeout*. Implementasi *timer* ini menggunakan *startAutoCloseTimer method* yang dipicu oleh fungsi *watch* ketika *dialog* dengan tipe 'feedbackSent' menjadi *visible*. Pesan ini juga disertai simbol centang berwarna oranye dan teks terima kasih sebagai bentuk apresiasi terhadap partisipasi siswa dalam proses evaluasi kelas. Setelah *feedback* diberikan, *section* secara otomatis bertransformasi menjadi "*E-Certificate*".



Gambar 3.63. Dialog Box "*Feedback Sent*"

I Tampilan *E-Certificate*

Seperti pada gambar 3.64 yang terlihat pada halaman detail kelas yang sudah selesai, tampilan ini menunjukkan tampilan dari fitur E-Certificate yang ditujukan untuk memberikan pengakuan resmi atas pencapaian belajar siswa setelah menyelesaikan seluruh modul dalam sebuah kelas. Di bagian bawah terdapat tombol aksi berwarna oranye dengan label *"View"*.



Gambar 3.64. Tampilan *E-Certificate*

I.1 *Dialog Box "Download E-Certificate"*

Tombol "View" yang ada pada bagian sebelumnya ini akan memicu sistem untuk memunculkan *dialog box* ketika diklik untuk bisa melihat dan mengunduh sertifikat digital. Terdapat dua kondisi pada bagian ini, yaitu sebagai berikut.

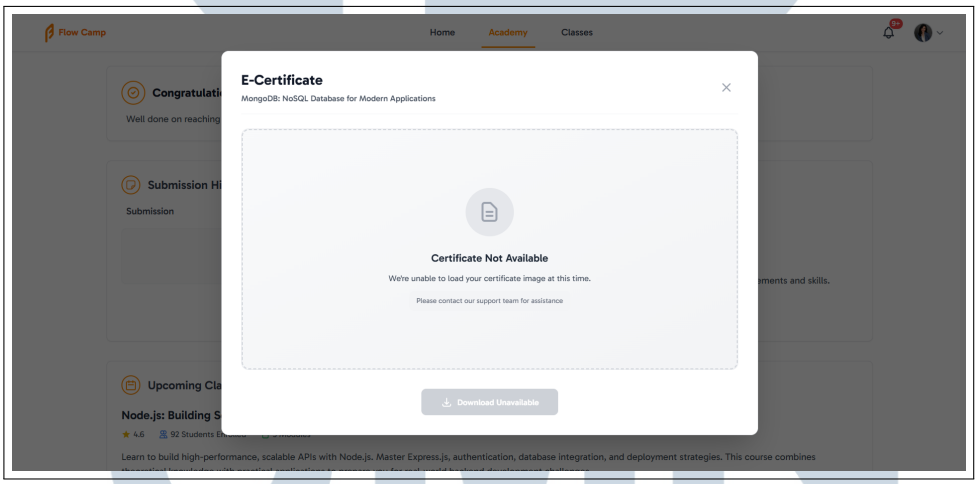
1. *Dialog Box "Download E-Certificate"* jika Terdapat E-Certificate

   Jika sertifikatnya tersedia, maka implementasi tampilannya akan terlihat seperti pada gambar 3.65. Sertifikat ini berbentuk *file* gambar yang dapat diunduh oleh siswa ketika mengklik tombol "Download Certificate" yang memungkinkan pengguna untuk langsung mengunduh sertifikat tersebut. Sertifikat ini juga sebagai tanda bahwa kelas telah diselesaikan secara resmi dan siswa berhak mendapatkan pengakuan atas pencapaiannya.

Gambar 3.65. *Dialog Box "Download E-Certificate"* jika Terdapat *E-Certificate*

2. *Dialog Box* ”*Download E-Certificate*” jika Tidak Terdapat *E-Certificate*

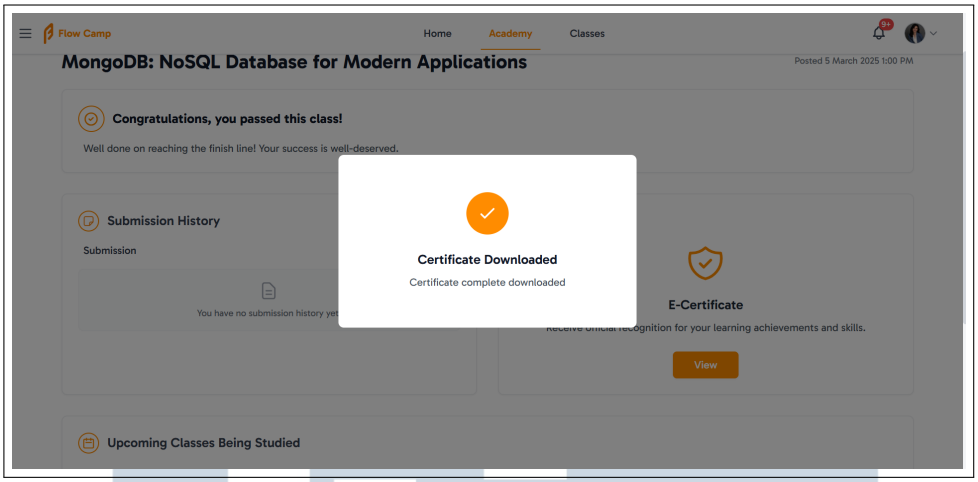
Lalu, jika tidak tersedia sertifikatnya, maka implementasi tampilannya terlihat seperti pada gambar 3.66. Gambar ini menampilkan kondisi ketika pengguna mencoba mengakses atau mengunduh *E-Certificate*, namun sistem belum menyediakan sertifikat tersebut. Di bagian bawahnya terdapat deskripsi tambahan yang menjelaskan bahwa saat ini sertifikatnya tidak ada. Kemudian, terdapat tombol aksi bertuliskan *“Download Unavailable”* yang juga muncul dalam keadaan non-aktif (disabled) sebagai penanda bahwa pengguna tidak dapat melanjutkan proses unduhan sampai terdapat file sertifikatnya. Hal yang menyebabkan tombol tersebut non-aktif karena bagian diimplementasikan validasi proses *download* menggunakan validasi ID kelas, verifikasi ketersediaan gambar, dan pencegahan *download* duplikat. Lalu, terdapat juga ikon silang yang ketika diklik oleh siswa, maka akan menutup *dialog box* tersebut.



Gambar 3.66. *Dialog Box* ”*Download E-Certificate*” jika Tidak Terdapat *E-Certificate*

I.2 *Dialog Box* ”*E-Certificate Downloaded*”

Kemudian, jika tombol ”Download Certificate”nya sudah diklik oleh siswa, maka akan muncul *dialog box* dengan teks “Certificate Downloaded” seperti yang ditampilkan pada gambar 3.67. *Dialog box* ini berfungsi sebagai notifikasi visual yang mengonfirmasi bahwa proses pengunduhan *E-Certificate* telah berhasil dilakukan oleh pengguna. Setelah ditampilkan selama 1 detik, *pop-up* akan menghilang secara otomatis, yang tidak akan mengganggu aktivitas pengguna di halaman tersebut.

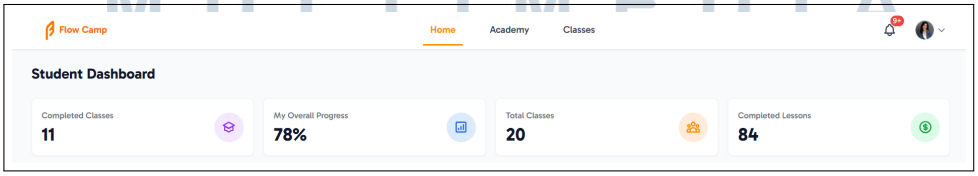


Gambar 3.67. *Dialog Box* "*E-Certificate Downloaded*"

J Tampilan *Home Dashboard* Siswa

J.1 Tampilan Statistik Pembelajaran

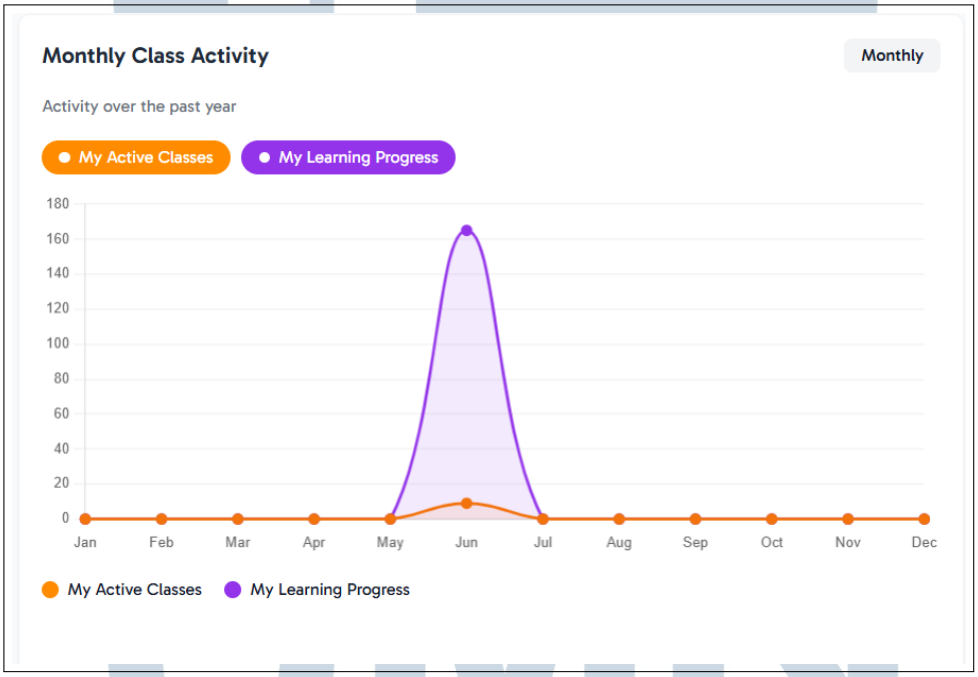
Pada halaman *home dashboard* siswa ini, bagian atasnya terlihat seperti pada gambar 3.68. Tampilan ini menyajikan ringkasan statistik pembelajaran siswa dalam bentuk empat metrik *card* informasi dengan *grid layout*. Keempat kartu tersebut mencakup jumlah *Completed Classes*, persentase *Overall Progress*, jumlah *Total Classes*, serta jumlah *Completed Lessons*. Setiap kartu dilengkapi dengan ikon representatif dan tampilan angka yang memperlihatkan data pencapaian siswa secara kuantitatif. Setiap metrik menggunakan *data processing* dengan *reactive computed properties* yang secara *real-time* menghitung metriknya berdasarkan data saat ini. Lalu, tampilan *card* tersebut menggunakan *styling* dengan efek *hover* dan *semantic color coding*, di mana warna ungu untuk "*completed classes*", warna biru untuk "*overall progress*", warna oranye untuk "*total classes*", dan warna hijau untuk "*completed lessons*". Pada halaman penuh ini juga diimplementasikan *loading states* menggunakan *skeleton screens* dengan animasi *pulse* dalam proses menampilkan data.



Gambar 3.68. Tampilan Statistik Pembelajaran

J.2 Tampilan Grafik *Active Class* dan *Learning Progress*

Kemudian, implementasi tampilan pada bagian *Monthly Class Activity* ditunjukkan pada gambar 3.69 yang menyajikan data aktivitas belajar selama satu tahun terakhir yang dibuat menggunakan *library* Chart.js menggunakan dual dataset. Grafik ini terdiri dari dua kurva yang merepresentasikan dua metrik berbeda, yaitu *My Active Classes* yang ditandai dengan garis oranye dan *My Learning Progress* dengan garis ungu. Visualisasi data ini memudahkan siswa dalam memantau tren aktivitas belajar mereka dari bulan ke bulan, serta mengevaluasi peningkatan atau penurunan progres pembelajaran secara periodik.

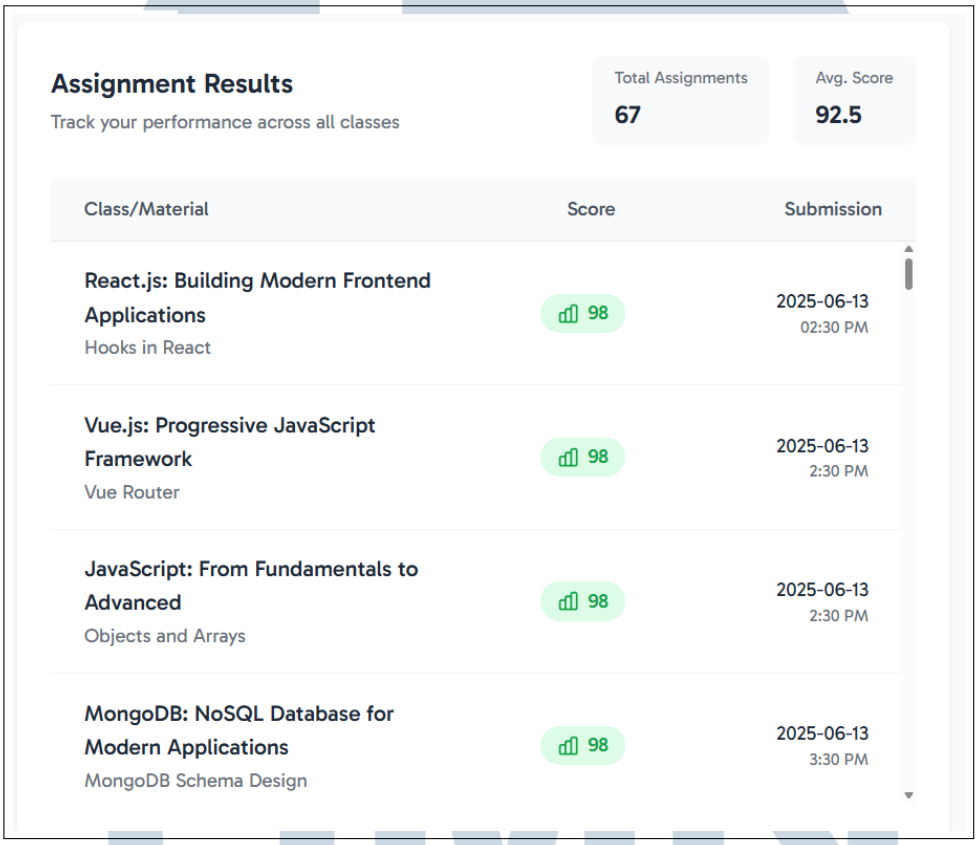


Gambar 3.69. Tampilan Grafik Kelas dan *Student Active*

J.3 Hasil Task Siswa

Selanjutnya, pada gambar 3.70 menampilkan bagian *Assignment Results* yang berfungsi untuk melacak performa siswa dalam menyelesaikan tugas-tugas dari berbagai kelas. Pada bagian atas, terdapat informasi total tugas yang telah diselesaikan (*Total Assignments*) dan rata-rata nilai keseluruhan (*Average Score*). Di bawahnya, tabel menampilkan rincian tugas berdasarkan kelas atau materi, nilai yang diperoleh, serta tanggal dan waktu pengumpulan tugas. Visual pada bagian ”Score”-nya menggunakan *dynamic color-coded badges* dengan *semantic coloring*, yaitu warna hijau untuk score ≥ 90 (excellent), warna oranye untuk

75–89 (good), warna biru untuk 60–74 (satisfactory), dan warna kuning untuk < 60 (needs improvement). Lalu, jika pada *Chart*-nya tidak terdapat data apa-apa, maka tampilannya menggunakan ikon dengan pesan ”No activity data available”. Tampilan *Chart* ini memberikan gambaran menyeluruh tentang kinerja akademik siswa secara terstruktur dan informatif, serta memungkinkan siswa melakukan evaluasi mandiri terhadap capaian pembelajarannya.

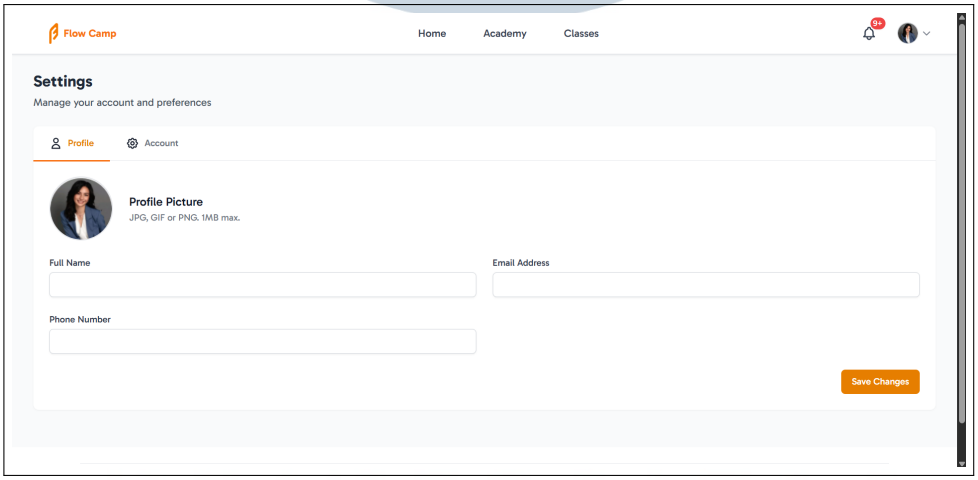


Gambar 3.70. Hasil Task Siswa

K Tampilan *Setting Profile*

Implementasi tampilan untuk bagian *setting profile* terlihat pada gambar 3.71 yang menunjukkan halaman *Settings* pada bagian *Profile* yang digunakan untuk mengelola informasi akun pengguna. Tampilan ini muncul ketika siswa mengklik ikon ”panah ke bawah” yang ada pada navbar, yang akan memunculkan *dropdown*. Ketika siswa mengklik pilihan ”*Settings*”, maka sistem akan mengarahkan pada tampilan ini. Dalam bagian ini, dilakukan implementasi *tab switching* menggunakan *dynamic styling*. *Active tab indicator* menggunakan *animated bottom border* dengan transisi (duration-300). Manajemen *state tab*-nya

menggunakan *Vue watchers*. Kemudian, pada tampilan *setting profile*, pengguna dapat memperbarui foto profil dengan format *JPG*, *PNG*, atau *GIF*. Selain itu, terdapat kolom untuk mengisi atau memperbarui data diri seperti nama lengkap (*Full Name*), (*Email Address*), dan nomor telepon (*Phone Number*). Setelah melakukan perubahan, pengguna dapat menekan tombol "*Save Changes*" untuk menyimpan pembaruan data, dan ketika berhasil maka akan muncul *toast* sukses. Lalu, pada *form* ini terdapat validasi untuk melakukan pengecekan terhadap semua data yang harus diisi termasuk "*Full Name*", "*Email Address*", "*Phone Number*", dan foto profilnya. Namun, ketika siswa tidak mengganti foto profilnya, maka foto yang lama akan diatur sebagai *default*-nya yang membuat siswa dapat mengganti *form* identitas dirinya saja. Pada *form email* juga terdapat validasi *email* menggunakan pencocokan pola *regex* untuk memastikan kesesuaian format dengan konvensi *email* standar. Lalu, untuk *phone number*-nya juga menggunakan input *filtering* yang hanya menerima angka. Pada *form* ini menggunakan *styling* warna oranye sebagai *focus states*-nya dengan *hover* transisi. Lalu, pada bagian *container* foto menggunakan desain *hover overlay* dengan ikon kamera yang muncul saat di *hover*.

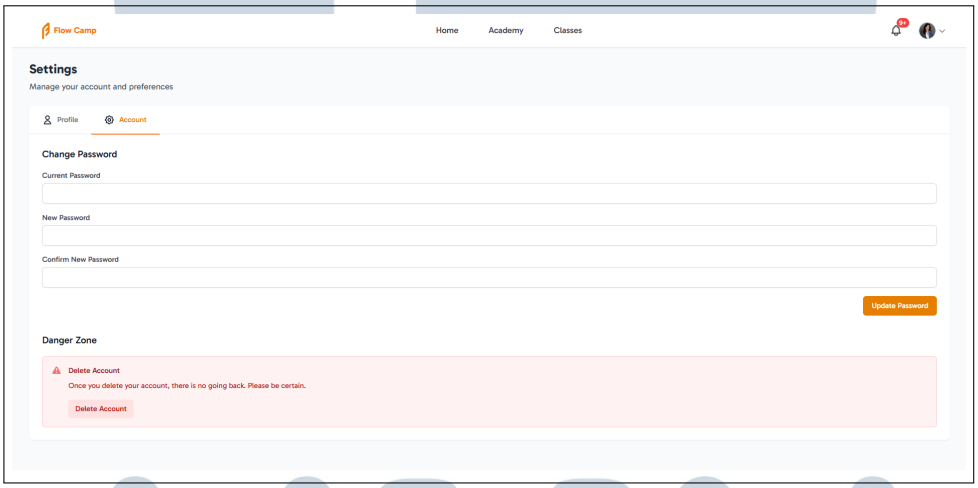


Gambar 3.71. Tampilan *Setting Profile*

K.1 Tampilan *Setting Account*

Halaman *Settings* pada tab *Account* memungkinkan pengguna untuk mengelola keamanan akun mereka dan implementasi tampilannya terlihat pada gambar 3.72. Di bagian atas halaman ini terdapat formulir untuk mengganti kata sandi, yang terdiri dari tiga kolom input yaitu *Current Password*, *New Password*, dan *Confirm New Password*. Setelah pengguna mengisi ketiga kolom tersebut,

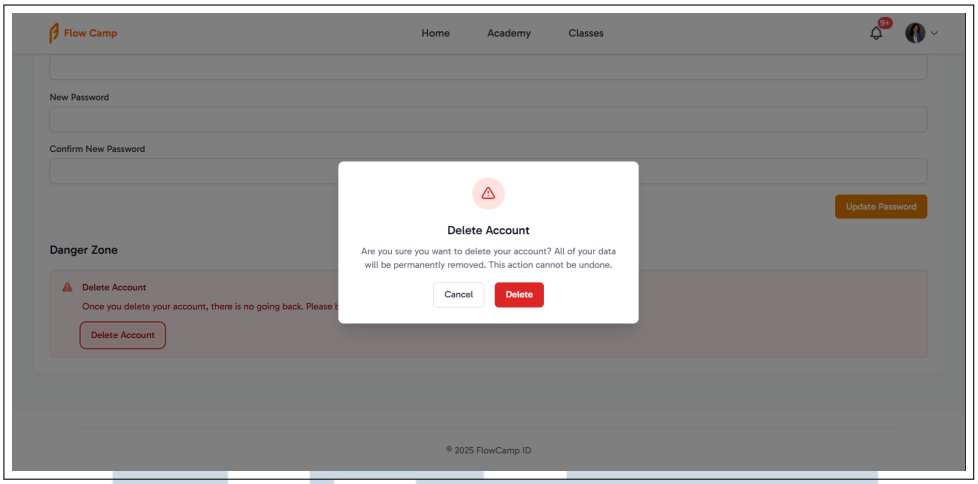
perubahan dapat disimpan dengan menekan tombol "*Update Password*". Pada bagian ini juga diimplementasikan validasi *form* yang mencakup minimum panjang karakternya sebanyak 8 karakter, kemudian *password* tersebut harus terdiri dari huruf *uppercase*, *lowercase*, dan angka. Lalu, sistem juga memiliki validasi lain pada bagian "*New Password*" dengan "*Confirm New Password*" yang mencocokkan *string* yang dimasukkan agar harus sama. Setelah berhasil mengganti *password*-nya, maka akan muncul *toast* sukses selama 3 detik. Kemudian, tombol "*Delete Account*" berwarna merah disediakan untuk memproses permintaan penghapusan akun secara permanen. Ketika siswa menekan tombol tersebut, maka akan diarahkan ke halaman *login*.



Gambar 3.72. Tampilan *Setting Account*

K.2 Tampilan *Dialog Box "Delete Account"*

Tampilan *dialog box* konfirmasi muncul ketika pengguna menekan tombol *"Delete Account"* pada halaman *setting account* seperti yang terlihat pada gambar 3.73. *Dialog box* ini berfungsi sebagai langkah pengaman tambahan untuk memastikan bahwa pengguna benar-benar ingin menghapus akun mereka secara permanen. Pesan yang ditampilkan memberikan peringatan bahwa seluruh data akan dihapus dan tindakan ini tidak dapat dibatalkan. Terdapat dua pilihan yang tersedia, yaitu tombol "*Cancel*" untuk membatalkan proses, serta tombol "*Delete*" berwarna merah sebagai konfirmasi akhir untuk menghapus akun. Tampilan ini dirancang dengan gaya visual yang jelas untuk menghindari penghapusan akun secara tidak sengaja.



Gambar 3.73. Tampilan *Dialog Box "Delete Account"*

L Struktur Komponen

Dalam pengembangan aplikasi web berbasis Vue.js seperti FlowCamp ID, pengelolaan antarmuka secara modular merupakan prinsip utama untuk menjaga keterbacaan, skalabilitas, dan efisiensi pengembangan. Arsitektur berbasis komponen (*component-based architecture*) diterapkan untuk membagi antarmuka menjadi bagian-bagian kecil (komponen) yang dapat digunakan kembali (*reusable*) dan disusun menjadi tampilan halaman yang kompleks. Penamaan komponen merujuk langsung pada *file .vue* yang digunakan dalam proyek dan disimpan di dalam direktori *src/components/*.

Tabel 3.2. Daftar komponen beserta halaman penggunaannya

Nama *File* Komponen (.vue)	Fungsi Utama	Halaman yang Menggunakan
NavbarStudent.vue	Navigasi utama yang muncul di bagian atas halaman setelah pengguna login sebagai siswa.	semua halaman *dashboard student*
FooterStudent.vue	*Footer* yang muncul di bagian bawah setiap halaman *student*.	Semua halaman *dashboard student*
NavbarHome.vue	navigasi pada *landing page*.	Semua halaman *landing page*, yaitu BootcampPage.vue, Testimonials.vue, FreeClass.vue, FreeClassSeeMore.vue, ProgramDetail.vue
Footer.vue	*Footer* yang muncul di bagian bawah setiap halaman *landing page*.	Semua halaman *landing page*, yaitu BootcampPage.vue, Testimonials.vue, FreeClass.vue, FreeClassSeeMore.vue, ProgramDetail.vue
HeroSection.vue	Sebagai bagian atas halaman yang menampilkan elemen visual utama, judul, deskripsi, dan tombol aksi.	Semua halaman *landing page*, yaitu BootcampPage.vue, Testimonials.vue, FreeClass.vue, FreeClassSeeMore.vue, ProgramDetail.vue

ClassCard.vue	menampilkan informasi kelas dalam format kartu.	AvailableClasses.vue, AcademyClassStudied.vue, dan FreeClassSeeMore.vue
StudentCard.vue	Menampilkan informasi siswa dalam format kartu.	Testimonials.vue
StudentBreadcrumb.vue	Sebagai navigasi *breadcrumb* yang membantu pengguna mengetahui lokasi mereka dalam hierarki halaman.	StudentLeaderboard.vue, LearningMaterials.vue, DetailClass1.vue, DetailTaskHistory.vue, DetailLearningMaterials.vue dan ClassDetail.vue
DialogBox.vue	Komponen *dialog* atau *modal box* yang sangat fleksibel dengan berbagai jenis tampilan berdasarkan *type* yang diberikan, seperti konfirmasi material telah dibaca, *dialog* untuk melihat *PPT* atau video, status penyelesaian tugas, dan tampilan nilai.	DetailTaskHistory.vue, DetailLearningMaterials, dan DetailClass1.vue
TaskHistoryDialog.vue	Untuk menampilkan riwayat tugas dan interaksi dengan tugas-tugas.	DetailClass1.vue

Sebagai hasil dari proses *slicing design*, komponen-komponen yang terdaftar pada Tabel 3.2 telah diimplementasikan. Tampilan akhir dari setiap komponen dapat dilihat pada gambar-gambar berikut.

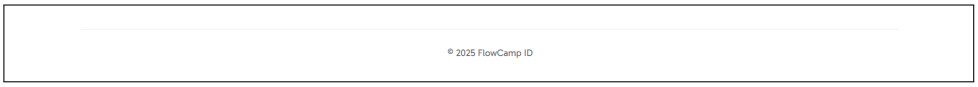
1. NavbarStudent.vue

   Implementasi komponen NavbarStudent.vue seperti yang terlihat pada 3.74 menggunakan *sticky positioning* dengan `sticky top-0 z-50` dan *shadow* `shadow-sm` untuk memberikan efek mengambang. *Layout* menggunakan *flexbox* `flex items-center justify-between h-16` dengan *responsive design* melalui *breakpoints* `md:` untuk tampilan *desktop* dan *mobile*. Komponen ini memiliki *navigation links* yang hanya tampil pada *desktop* menggunakan `hidden md:flex`, area notifikasi dengan *dropdown* menggunakan *transition* `transform opacity-0 scale-95`, dan *profile dropdown* dengan *avatar* menggunakan *hover effects* `hover:bg-orange-50`. *Sidebar toggle button* untuk *mobile* menggunakan `inline-flex items-center justify-center p-2` dengan *active state* `active:scale-95` dan *focus ring* `focus:ring-2 focus:ring-orange-300`. Logo menggunakan *conditional rendering* dengan *opacity transition* `opacity-0 invisible` ketika *sidebar* terbuka, serta *notification badge* dengan *computed properties* untuk menampilkan jumlah notifikasi yang belum dibaca dengan *styling* dinamis berdasarkan *status* tugas.

Gambar 3.74. Tampilan NavbarStudent.vue

2. FooterStudent.vue

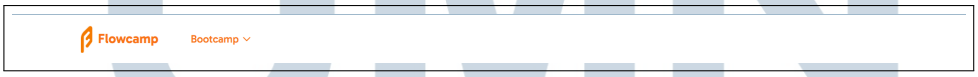
   Implementasi komponen FooterStudent.vue seperti pada gambar 3.75 menggunakan *semantic HTML* `<footer>` dengan *layout* `py-4 sm:py-6 mt-auto` untuk memberikan *padding* vertikal yang responsif dan *margin-top auto* agar selalu berada di bagian bawah halaman. *Container* menggunakan *max-width* `max-w-screen-xl mx-auto` dengan *horizontal padding* responsif `px-4 sm:px-6 lg:px-8` untuk berbagai ukuran layar. Bagian konten utama menggunakan *border-top* `border-t border-gray-200` dengan *padding-top* responsif `pt-4 sm:pt-6` sebagai pemisah visual, serta *text styling* `text-center text-xs sm:text-sm text-gray-500` untuk menampilkan *copyright* "© 2025 FlowCamp ID" dengan *typography* yang responsif dari *extra small* pada *mobile* hingga *small* pada layar yang lebih besar, menggunakan warna abu-abu yang *subtle* untuk memberikan kesan profesional dan tidak mengganggu konten utama.



Gambar 3.75. Tampilan FooterStudent.vue

3. NavbarHome.vue

Implementasi komponen NavbarHome.vue seperti yang terlihat pada gambar 3.76 menggunakan *semantic HTML* `<nav>` dengan *layout* `bg-white shadow-sm py-4 relative` dan *container* `mx-auto px-4` untuk struktur responsif. *Logo section* menggunakan *flexbox* `flex items-center` dengan *router-link* yang menampilkan logo dan teks "Flowcamp" menggunakan *styling* `text-orange-500 font-bold text-lg sm:text-xl`. *Desktop dropdown* menggunakan *absolute positioning* `absolute left-0 mt-1` dengan *styling* `w-72 bg-white rounded-2xl shadow-lg py-4` dan *transition* `transition-transform duration-200` untuk *chevron icon* yang berotasi menggunakan `rotate-180`. Setiap *dropdown item* menggunakan *conditional styling* dengan `bg-orange-100 border border-orange-200` untuk *active state* dan `hover:bg-gray-50` untuk *hover effect*, dilengkapi emoji sebagai *visual indicator*. *Mobile menu* menggunakan *hamburger icon* dengan *transform animation* `transform transition-all duration-300 ease-in-out` untuk efek *cross* ketika dibuka, serta *mobile dropdown* menggunakan *absolute positioning* `absolute top-full left-0 right-0` dengan *opacity transition* `opacity-100 visible` dan `opacity-0 invisible` untuk *smooth show/hide animation*.

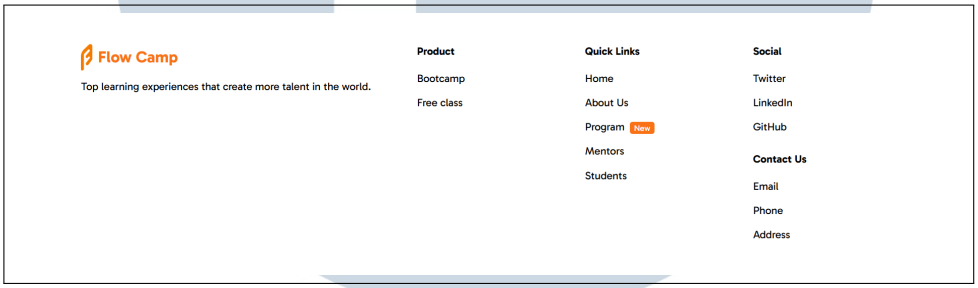


Gambar 3.76. Tampilan NavbarHome.vue

4. Footer.vue

Implementasi komponen Footer.vue seperti yang terlihat pada gambar 3.77 menggunakan *semantic HTML* `<footer>` dengan *layout* `bg-white py-6 sm:py-8 lg:py-12` dan *container* `mx-auto px-4 sm:px-6 lg:px-8` untuk struktur responsif. *Grid system* menggunakan `grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-5 gap-6 sm:gap-8` dengan *column spanning* yang berbeda untuk setiap ukuran layar, dimana *logo section* menggunakan `col-span-1 sm:col-span-2 md:col-span-3 lg:col-span-2` dan bagian *social/contact* menggunakan

`col-span-1 sm:col-span-2 md:col-span-1 lg:col-span-1`. *Logo section* menggunakan *flexbox* `flex items-center` dengan *typography* `text-orange-500 font-bold text-xl sm:text-2xl` dan deskripsi menggunakan *responsive text* `text-sm sm:text-base` dengan *max-width* `max-w-xs sm:max-w-sm lg:max-w-md`. Setiap *section* menggunakan *heading* dengan *styling* `font-bold text-black text-sm sm:text-base mb-3 sm:mb-4` dan *navigation links* menggunakan `space-y-2 sm:space-y-3` dengan *hover effects* `hover:text-orange-500 transition-colors`, serta *badge* "*New*" menggunakan `bg-orange-500 text-white text-xs px-1.5 py-0.5 rounded ml-1` untuk menandai item khusus.

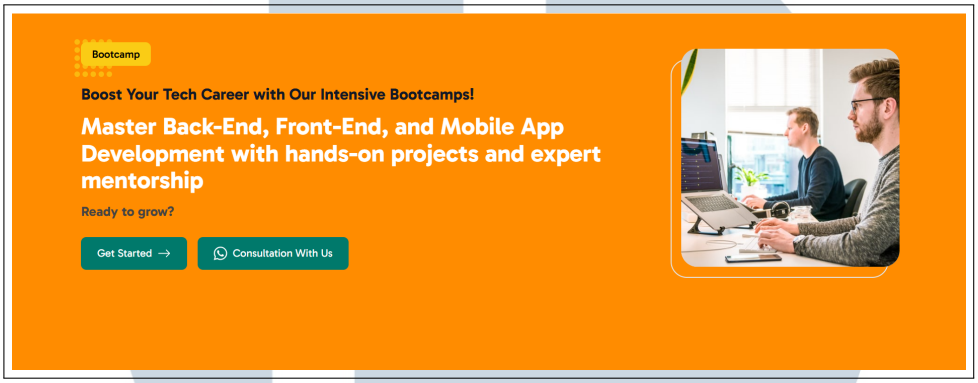


Gambar 3.77. Tampilan Footer.vue

5. HeroSection.vue

Implementasi komponen HeroSection.vue seperti pada gambar 3.78 menggunakan *section* dengan *background color* `bg-[ff8c00]` dan *overflow* `overflow-hidden relative min-h-[500px]` untuk struktur dasar. *Background image overlay* menggunakan *conditional rendering* `v-if="useOverlay"` dengan *absolute positioning* `absolute inset-0` dan *gradient overlay* `bg-gradient-to-l from-[FF9A33]/20 to-black/90` untuk efek visual. *Layout* menggunakan *flexbox* `flex flex-wrap items-center` dengan *conditional class binding* untuk mengatur lebar kolom kiri berdasarkan *props* `hideRightSection`. *Badge decoration* menggunakan *yellow dots* dengan *grid* `grid grid-cols-5 gap-1` dan *positioning* `absolute -top-2.5 -left-2.5 z-0`, sedangkan teks *badge* menggunakan *styling* `bg-yellow-400 text-black font-medium px-4 py-2 rounded-lg`. *Typography hierarchy* menggunakan *responsive text sizes* dengan *conditional color classes* berdasarkan *props* `isDark`, dan *action buttons* menggunakan *flexbox* `flex gap-4` dengan *styling* `bg-[00796B] text-white rounded-lg px-6 py-3` dan *hover effects*

`hover:bg-[00695C] transition-colors`. *Right section image* menggunakan *aspect ratio* `pb-[100%]` dengan *border decoration* `border-2 border-teal-600 rounded-3xl` yang diposisikan menggunakan `absolute top-4 right-4` untuk memberikan efek *layered design*.

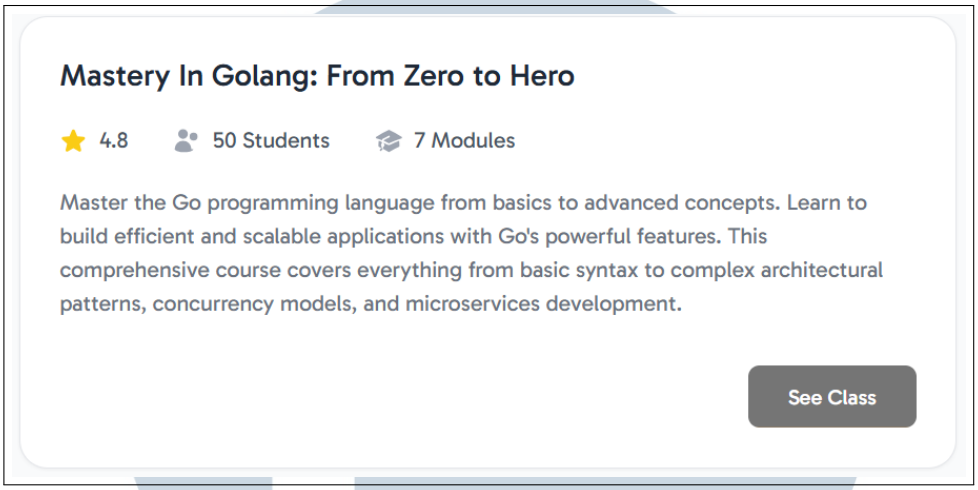


Gambar 3.78. Tampilan HeroSection.vue

6. ClassCard.vue

Implementasi komponen ClassCard.vue seperti pada gambar 3.79 menggunakan *container* dengan *styling* `bg-white rounded-[16px] sm:rounded-[20px] p-3 sm:p-5 md:p-7 shadow-sm border border-gray-200` dan *hover effect* `hover:shadow-md transition-shadow duration-300` untuk interaksi visual. *Card header* menggunakan *responsive typography* `text-base sm:text-lg md:text-xl lg:text-[22px] font-medium text-gray-800` dengan *text truncation* `line-clamp-2` untuk membatasi jumlah baris. *Stats section* menggunakan *flexbox* `flex flex-wrap items-center gap-2 sm:gap-3 md:gap-6 lg:gap-8` dengan *SVG icons* berukuran responsif `w-3.5 h-3.5 sm:w-4 sm:h-4 md:w-5 md:h-5` untuk menampilkan *rating*, jumlah siswa, dan modul. *Description* menggunakan *responsive text* `text-xs sm:text-sm md:text-base` dengan *line clamping* `line-clamp-3 sm:line-clamp-4` untuk konsistensi layout. *Action button* menggunakan *conditional class binding* dengan *styling* `bg-orange hover:bg-orange-dark` untuk *View Class* dan *complex hover animation* menggunakan `bg-[757575] overflow-hidden group hover:text-white` dengan *sliding overlay* `absolute inset-0 bg-orange transform translate-y-full transition-transform duration-300 ease-out group-hover:translate-y-0` untuk tombol

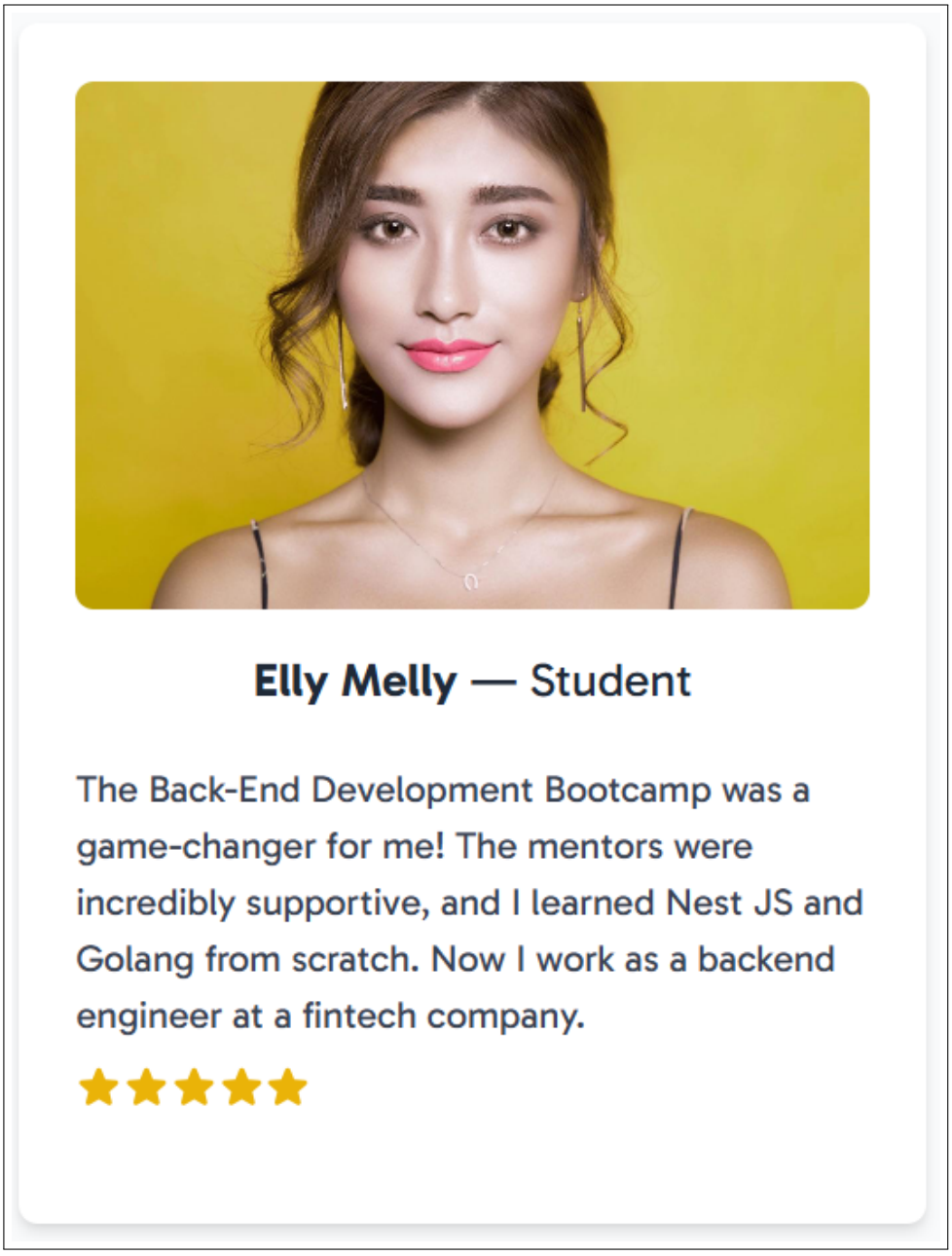
”*See Class*”, dilengkapi dengan *z-index* `relative z-10` untuk memastikan teks tetap terlihat di atas *overlay animation*.



Gambar 3.79. Tampilan ClassCard.vue

7. StudentCard.vue

Implementasi komponen StudentCard.vue seperti pada gambar 3.80 menggunakan *container* dengan *styling* `bg-white rounded-lg shadow-md overflow-hidden transition-transform hover:shadow-lg p-4 sm:p-6` dan *flexbox layout* `flex flex-col items-center` untuk penataan vertikal yang terpusat. *Image section* menggunakan *responsive spacing* `mb-3 sm:mb-4` dengan *container* `w-full overflow-hidden rounded-lg` dan gambar menggunakan *object-fit* `w-full h-48 sm:h-56 object-cover` untuk mempertahankan *aspect ratio*. *Name and role section* menggunakan *text alignment* `text-center mb-3 sm:mb-5` dengan *typography* `text-lg sm:text-xl font-bold text-gray-800` untuk nama dan `font-normal` untuk peran yang dipisahkan dengan *em dash*. *Testimonial text* menggunakan *responsive font size* `text-sm sm:text-base` dengan *styling* `text-gray-700 mb-2 w-full` untuk memastikan teks memenuhi lebar *container*. *Rating stars* menggunakan *flexbox* `flex w-full` dengan *template loop* `v-for="star in 5"` untuk menampilkan 5 bintang menggunakan *SVG icons* berukuran responsif `w-4 h-4 sm:w-5 sm:h-5 text-yellow-500` dengan *fill rule* `fill-rule="evenodd"` dan *clip rule* `clip-rule="evenodd"` untuk rendering yang optimal.

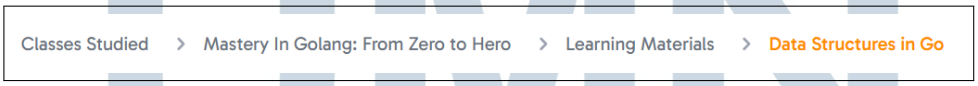


Gambar 3.80. Tampilan StudentCard.vue

8. StudentBreadcrumb.vue

   Implementasi komponen `StudentBreadcrumb.vue` seperti pada Gambar 3.81 dilakukan dengan memanfaatkan *semantic HTML* menggunakan elemen `<nav>` dan atribut `aria-label` untuk meningkatkan aksesibilitas. Struktur tata letak disusun menggunakan kelas `w-full mb-4 flex items-center justify-center sm:justify-start` untuk menciptakan perataan yang responsif terhadap berbagai

ukuran layar. *Container* yang dibuat dapat di *scroll* secara horizontal dengan menerapkan kelas `overflow-x-auto`, serta penyembunyian *scrollbar* diterapkan untuk berbagai browser menggunakan kombinasi utilitas `[scrollbar-width:none]`, `[-ms-overflow-style:none]`, dan `[&::-webkit-scrollbar]:hidden`. Daftar breadcrumb disusun menggunakan *Flexbox* dengan kelas `flex flex-nowrap items-center space-x-1 sm:space-x-2`. Pengulangan elemen dilakukan melalui perulangan `v-for="(item, index) in items"` dan setiap teks diberi kontrol jarak menggunakan `whitespace-nowrap` agar tidak terpotong. Elemen pemisah (separator) di antara *breadcrumb* diterapkan secara kondisional menggunakan `v-if="index > 0"` dan ditampilkan sebagai ikon *chevron* berbasis SVG dengan ukuran responsif melalui kelas `h-4 w-4 sm:h-5 sm:w-5 text-gray-400` serta `flex-shrink-0` untuk mencegah kompresi ukuran. Setiap item *breadcrumb* dibedakan antara *active state* dan *inactive state* menggunakan pengkondisian `v-if` dengan gaya teks `text-orange font-medium` untuk status aktif dan `text-gray-500 hover:text-orange transition-colors duration-200` untuk status tidak aktif. Pemotongan teks panjang dilakukan menggunakan kelas `max-w-[100px] xs:max-w-[150px] sm:max-w-none truncate` serta pengaturan tipografi responsif melalui `text-sm sm:text-[15px]` guna memastikan keterbacaan yang optimal pada berbagai ukuran layar.

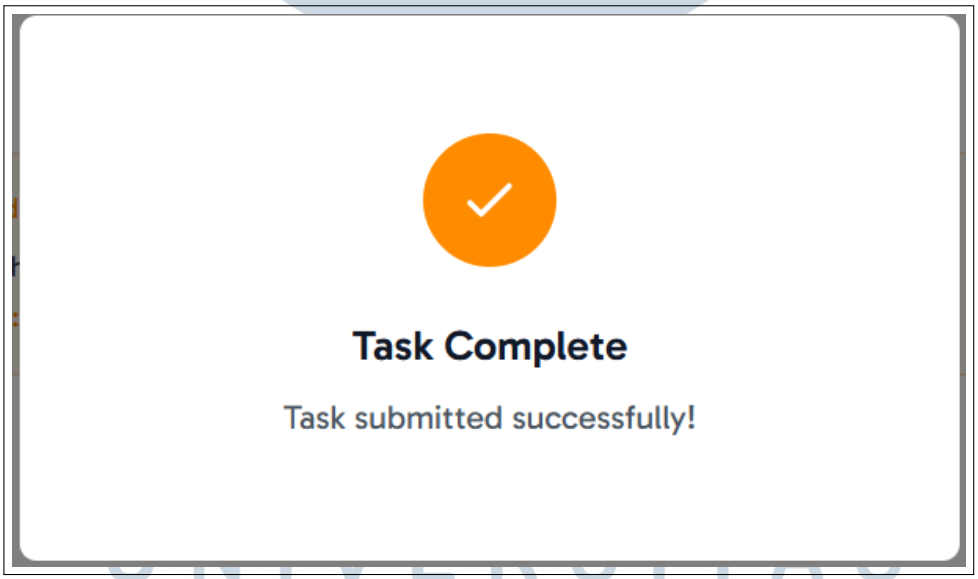


Gambar 3.81. Tampilan StudentBreadcrumb.vue

9. DialogBox.vue

Implementasi komponen DialogBox.vue yang terlihat pada gambar 3.82 menggunakan *modal overlay* dengan *fixed positioning* `fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4` dan *modal container* `bg-white rounded-lg p-4 sm:p-6 max-w-md w-full` dengan *click event stopping* `@click.stop`. Komponen menggunakan *multiple conditional templates* berdasarkan *props type* dengan *validator* untuk tipe `markAsRead`, `ppt`, `video`, `taskComplete`, `viewGrades`, `feedbackSent`, dan `certificateDownloaded`. *Success dialog* menggunakan *centered layout* `text-center py-4 sm:py-8`

dengan *circular icon containers* `rounded-full bg-orange w-12 h-12 sm:w-16 sm:h-16 mx-auto flex items-center justify-center` dan *SVG checkmarks* berukuran responsif `w-6 h-6 sm:w-8 sm:h-8 text-white`. *PPT/Video dialog* menggunakan *conditional content* berdasarkan ketersediaan URL dengan *header section* `flex items-center justify-between border-b pb-4`, *action buttons* menggunakan *flexbox* `flex justify-end space-x-3 sm:space-x-4` dengan *styling* `bg-orange text-white px-3 sm:px-4 py-1.5 sm:py-2 rounded-md hover:bg-orange-dark`, dan *error states* dengan *X icons* `w-8 h-8 sm:w-10 sm:h-10 text-white`. *View Grades dialog* menggunakan *complex layout* dengan *loading state* `animate-spin rounded-full h-8 w-8 border-b-2 border-orange`, *grade circle* dengan *dynamic classes* berdasarkan *computed properties* `w-28 h-28 sm:w-36 sm:h-36 rounded-full bg-gradient-to-br border-4`, dan *submission details grid* `grid grid-cols-1 sm:grid-cols-2 gap-3 sm:gap-4` dengan *card styling* `bg-white rounded-md border border-gray-100 shadow-sm`.

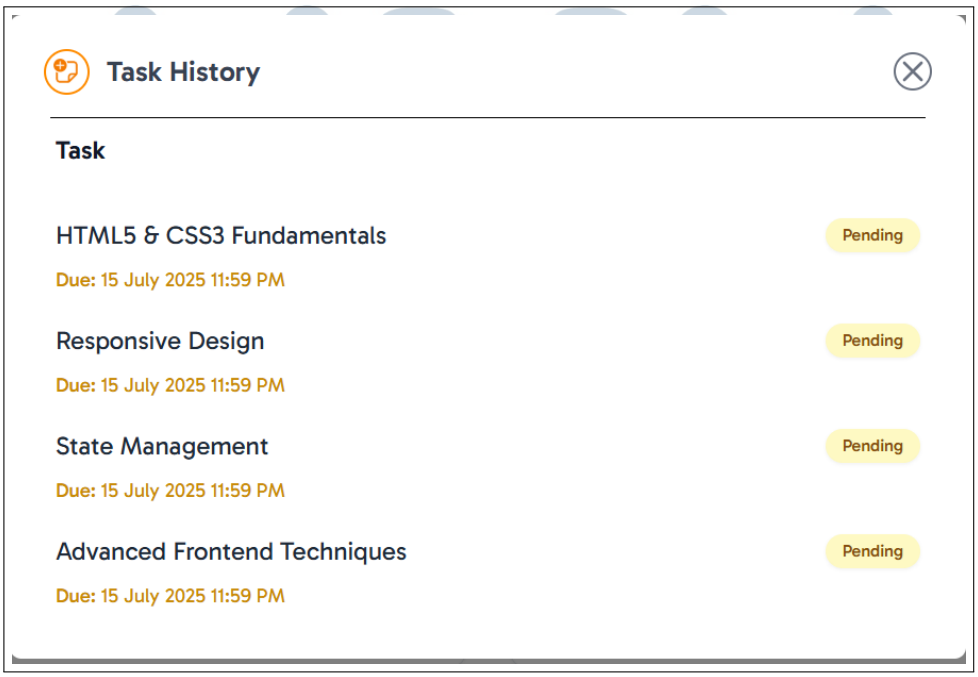


Gambar 3.82. Tampilan DialogBox.vue

10. TaskHistoryDialog.vue

Implementasi komponen TaskHistoryDialog.vue yang terlihat pada gambar 3.83 menggunakan *fullscreen modal* dengan *layered z-index* `z-[999]` dan *backdrop overlay* `absolute inset-0 bg-black/50` dengan *click handler* untuk menutup *dialog*. *Modal container* menggunakan *centered*

positioning `absolute inset-0 flex items-center justify-center p-4` dengan *dialog box* `bg-white rounded-lg p-4 sm:p-6 max-w-2xl w-full max-h-[90vh]` dan *flexbox layout* `flex flex-col relative z-[1000]`. *Header section* menggunakan *flexbox* `flex items-center justify-between mb-3 sm:mb-4 flex-shrink-0` dengan *task icon* dalam *circular border* `w-6 h-6 sm:w-8 sm:h-8 border-2 border-orange rounded-full` dan *close button* dengan *hover effects* `text-gray-500 hover:text-gray-700 border-2 rounded-full border-gray-500 hover:border-gray-700`. *Content area* menggunakan *scrollable container* `flex-1 overflow-y-auto` dengan *divider* `border-b ml-1 mr-1 border-black` dan *empty state* menggunakan `text-center text-gray-500 py-4` dengan *opacity icon* `h-12 w-12 opacity-50`. *Task items* menggunakan *hover effects* `hover:bg-gray-100 hover:rounded-lg transition-colors cursor-pointer` dengan *responsive layout* `flex flex-wrap sm:flex-nowrap justify-between items-center gap-2`, *status badges* menggunakan *dynamic classes* dari `getStatusClass` dengan *styling* `text-xs font-medium px-2 sm:px-3 py-1 rounded-full shadow-sm whitespace-nowrap`, dan *due date* menggunakan `text-xs sm:text-sm text-yellow-600` dengan *font weight* `font-medium` untuk label.



Gambar 3.83. Tampilan TaskHistoryDialog.vue

Komponen-komponen di atas tidak seluruhnya diimplementasikan dalam bentuk komponen terpisah. Beberapa bagian tetap berada di dalam *file* halaman karena arahan internal perusahaan yang tidak mewajibkan komponenisasi penuh. Meski demikian, pendekatan modular tetap diutamakan untuk bagian-bagian yang memiliki potensi penggunaan ulang.

3.5 Kendala dan Solusi yang Ditemukan

3.5.1 Kendala

Selama proses kerja magang berlangsung, terdapat beberapa kendala yang dihadapi sebagai berikut.

1. Mengalami masalah *layout* salah satunya adalah pada saat membuat modal dengan *backdrop blur* menggunakan *backdrop-filter* TailwindCSS, ternyata tidak *support* di semua browser.
2. Desain pada Figma tidak menunjukkan bagaimana tombol harus bereaksi saat di-*hover*, klik, dan *disable* di sebagian besar desain.

3.5.2 Solusi

Dari berbagai kendala yang dihadapi selama proses kegiatan kerja magang, didapatkan solusi sebagai berikut.

1. Menggunakan *bg-black bg-opacity-50* untuk *backdrop effect*, karena kompatibel di semua browser.
2. Melakukan konsultasi dan berdiskusi langsung ke desainer UI/UX. Lalu, karena tidak tersedia pedoman khusus, maka digunakan prinsip-prinsip *user interface* yang umum dan *best practices* dalam pengembangan antarmuka menggunakan TailwindCSS.