

React - Enigma Course Project (handson)

Trainer notes oleh **Muhammad Sulton (@musulton)**

Enigma course project adalah handson yang mengimplementasikan setiap materi dalam silabus pembelajaran react yang hasil akhirnya akan membentuk **course management application**.

Kamu bisa clone dari sini <https://github.com/musulton/r-enigma-course>.

Materi disusun kedalam branch terpisah agar dapat memudahkan trainer membaca alur pembahasan materi/kode. Berikut materi dalam masing-masing branch:

01 Add course form branch

- Membuat reusable component.
- Menggunakan props untuk memberi data pada component.
- Menggunakan state untuk menampung inputan data dari user.
- Menggunakan react-bootstrap sebagai third dependencies untuk component styling.
- Menggunakan styled-component (third dependencies) untuk menerapkan css styling.

FormInput components, komponen ini digunakan untuk membentuk komponen input dengan berbagai type (seperti text, textarea, file, dsb), komponen ini akan membentuk form input dengan styling dan behavior yang kita tentukan seperti akan selalu ada label dan form input pada komponen ini dengan style yang didefinisikan.

The screenshot shows a form with three sections. The first section is labeled 'Title' and contains a text input field with the placeholder text 'Enter course title'. The second section is labeled 'Description' and contains a text area with the placeholder text 'Enter course description'. The third section is labeled 'Course Material' and contains a file input field with a 'Choose File' button and the text 'No file chosen'.

FormSelect components, komponen ini untuk membentuk komponen select / dropdown. Di dalam komponen ini terdapat **values** sebagai props, akan digunakan untuk mendapatkan data yang akan ditampilkan untuk dapat dipilih oleh user, terdiri dari **label** untuk informasi nama yang akan ditampilkan ke user dan **value** (umumnya berisi id) untuk data yang akan di set dan dikirim ke backend, kenapa menggunakan **label**? karena tidak mungkin yang ditampilkan ke user adalah **id** nanti user tidak mengetahui apa yang dia pilih, maka dari itu perlu **label** sebagai informasi yang ditampilkan ke user.

Course Type Id

Function utils, membuat fungsi utilitas untuk event handler karena kemungkinan akan digunakan di banyak komponen.

AddCourse page, halaman tambah course terdiri dari:

- AddCourse(komponen), untuk pengaturan tampilan kita tempatkan disini, terdapat FORM_LIST digunakan untuk membuat konfigurasi form yang nantinya akan ditampilkan dengan cara di mapping / dilakukan perulangan agar kode yang ada di komponen lebih sedikit dan mudah dibaca dan maintenance.
- AddCourse.styled(css styles), menentukan style untuk subkomponen AddCourse menggunakan styled component agar dapat mengirim nilai dinamis untuk properti style menggunakan props.
- useAddCourse(custom hook), memisahkan hooks dari komponen agar tidak menumpuk di file komponen

Hasil

Add Course Page

Title

Description

Course Material

Level

Duration

Course Type Id

02 Course List

- Membuat navigation handler menggunakan state

StyledContainer component, untuk membungkus komponen agar komponen mempunyai jarak antar sisi di browser.

EmptyState component, ketika data tidak ada maka komponen ini akan kita gunakan untuk informasi ke user.

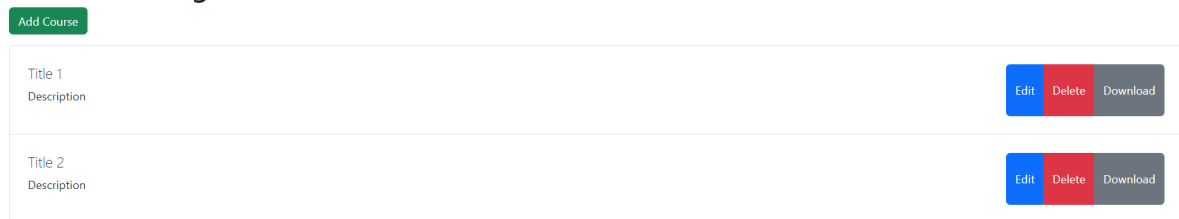
CourseList pages, komponen untuk menampilkan halaman course list, kita mengatur setiap kondisi di dalam komponen ini seperti ketika data kosong akan menampilkan apa, ketika

terdapat data course maka komponen apa yang akan ditampilkan, dan ada tombol add course untuk melakukan navigasi ke antar halaman. Ketika data course ada, maka komponen akan menampilkan **CourseItem component** dengan props yang kita kirim, jumlah item yang ditampilkan tergantung dengan banyaknya data.

App, komponen ini berisi konfigurasi screen. Menggunakan state, props, dan variabel untuk menentukan komponen mana yang seharusnya dimunculkan di browser ketika user melakukan interaksi navigasi.

Hasil

Course List Page



| Title | Description | Edit | Delete | Download |
|---------|-------------|------|--------|----------|
| Title 1 | Description | Edit | Delete | Download |
| Title 2 | Description | Edit | Delete | Download |

03 Data Fetching

- Menggunakan axios untuk data fetching.
- Menggunakan proxy di project client untuk mengatasi cors.
- Menggunakan form data untuk mengirim data course dan upload file material.
- Menampilkan data course saat ini di form input.
- Membuat custom hooks untuk menangani permintaan ke server.
- Menggunakan life cycle untuk menangani permintaan ke server.

Kamu perlu clone **springboot - enigma course** project (springboot handson material) untuk melakukan integrasi dengan **react - enigma course** project.

Kamu bisa clone darisini <https://github.com/musulton/sb-enigma-course>.

Api configs, membuat konfigurasi untuk koneksi ke server, yang kita tentukan disini hanya content-type, sedangkan base url tidak kita definisikan disini tetapi didefinisikan di **package.json** dalam menjadi properti proxy. **Proxy** akan digunakan untuk mengatasi masalah **cors**, dimana nantinya setiap permintaan akan diarahkan ke domain server melalui domain client.

Course services, membuat fungsi query untuk melakukan request ke server, terdiri dari get all course, get course by id, add course, edit course, dan get course type.

UseQuery hooks, digunakan untuk mengambil data ke server, setiap kali melakukan permintaan ke server kita perlu menyediakan 3 state untuk menangani status response nya dan 1 fungsi **refetch**:

- **Loading**, response tidak akan langsung diterima oleh client pasti perlu diproses dulu sisi server maka dari itu kita perlu membuat status loading ketika permintaan masih

di proses oleh server, status ini dapat digunakan untuk menampilkan komponen yang informatif ke user.

- **Error**, tidak selalu permintaan akan sukses diterima oleh server maka dari itu kita perlu membuat status error untuk menampung pesan kesalahan, dan status ini bisa kita gunakan untuk memberitahu user kalau ada yang salah dengan permintaan yang dikirim entah itu karena jaringan, inputan user yang tidak valid, atau mungkin karena server sedang down.
- **Data**, ketika permintaan berhasil diterima umumnya ada data yang dikirim dari server ke client sebagai responsnya, nah data tersebut kita simpan ke dalam state data.
- **Refetch**, digunakan untuk melakukan permintaan ulang ke server. Fungsi refetch akan otomatis dijalankan ketika status reload berubah, karena kita melakukan watching terhadap state reload (fase updating).

Dalam melakukan permintaan ke server kita harus mendeklarasikan fungsi fetching didalam use effect pada fase mounting, ini untuk memastikan komponen dirender terlebih dahulu sehingga ketika data sudah didapatkan tinggal dipasang ke komponennya (aturan dari react).

useMutation hooks, digunakan untuk memanipulasi data ke server, status yang disediakan hampir sama dengan **useQuery**, bedanya disini kita mendefinisikan fungsi untuk menangani permintaan ke server dan perlu dijalankan secara manual / dipanggil seperti di pasang di props onClick.

AddCourse pages, memanggil data course type untuk digunakan pada komponen select / dropdown, dan juga mengganti fungsi onSubmit menjadi permintaan ke server. Karena server menerima request data dari form data (bukan json) maka kita perlu membuat objek form data yang nantinya diisi dengan data2 yang diminta server, dan kita juga mengirim onNavigate sebagai callback ketika data berhasil di submit user akan diarahkan ke halaman course list.

CourseList pages, menambah beberapa permintaan yang dibutuhkan, seperti:

- Mengambil data ke server menggunakan useQuery.
- Menambah fungsi untuk menghapus data, sebelum data dihapus kita perlu memastikan dulu ke user apakah benar benar ingin menghapus course yang dipilih, ini untuk memastikan bahwa user tidak salah klik, dan ketika permintaan dikirim kita juga perlu menjalankan fungsi refetch agar mendapatkan data saat ini dari server dan data yang telah dihapus hilang dari komponen / halaman.
- Menambah fungsi untuk melakukan navigasi ke halaman edit course, fungsi tersebut menerima parameter id untuk dikirim ke halaman edit course.

EditCourse pages, sama dengan AddCourse page, bedanya disini perlu melakukan permintaan untuk mendapatkan data course berdasarkan id dan data course tersebut kita tampilkan pada masing2 formnya sehingga user dapat mengetahui data eksisting pada course tersebut, dan kita juga perlu melakukan permintaan edit course ke server dengan content-typenya adalah json karena di project ini kita tidak mengijinkan perubahan file pada edit course.

04 HOC Pagination

- Membuat pagination menggunakan HOC, agar dapat reusable
- Membuat komponen pagination

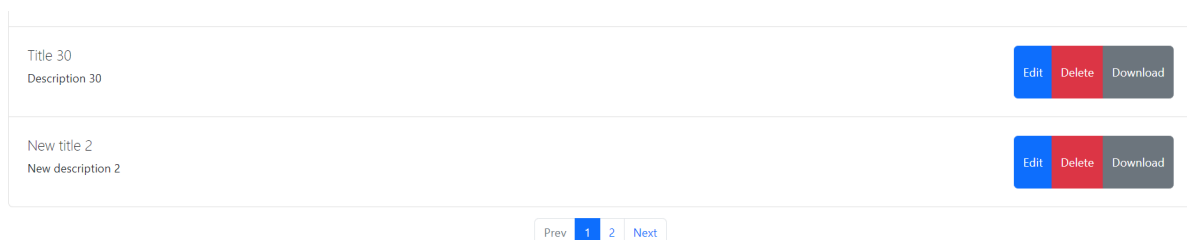
Pagination components, membuat komponen untuk mengatur tampilan paginasi, menerima beberapa props dan terdapat fungsi untuk menghandle paginasi:

- **onNext**, disini kita membuat kondisi ketika sudah berada di akhir halaman maka current page harus tidak bertambah karena kita sudah berada di halaman terakhir.
- **onPrev**, kita membuat kondisi ketika current page adalah 1 maka current page tidak boleh berkurang, karena kita sudah berada di halaman pertama.
- **onPageNumber**, untuk pindah pada page number tertentu, dan untuk item current page harus di disable agar tidak mengirim permintaan percuma ke server.

Jumlah halaman didapatkan dari server dan komponen pagination akan menampilkan nomor halaman sesuai dengan jumlah halaman dari server.

withPagination hoc, pagination akan selalu ada di setiap fitur list, misal course list, course type list, dll, jadi daripada membuat kode yang sama dan berulang pada setiap fitur tersebut lebih baik kita membuat sebuah HOC yang akan menerima component dan konfigurasi terkait pagination di masing2 fitur.

Hasil



05 Download File

- Membuat fungsi download

Course services, membuat fungsi **downloadCourseFile**. Untuk download di browser kita memerlukan trik dengan memanfaatkan fungsi click dari tag anchor html, kita perlu mendapatkan link file dan tentukan sebagai atribut href pada tag anchor kemudian melalui fungsi click tag anchor kita dapat mendownload file tersebut.

CourseList pages, fungsi **downloadCourseFile** dipanggil di halaman ini tapi kita perlu mengambil nama filenya saja maka dari itu kita melakukan konversi string menjadi array dan mengambil item terakhir (karena nama file ada di item terakhir).

06 Navigation

- Menggunakan react router dom (third dependencies) untuk routing (<https://reactrouter.com/en/main>).

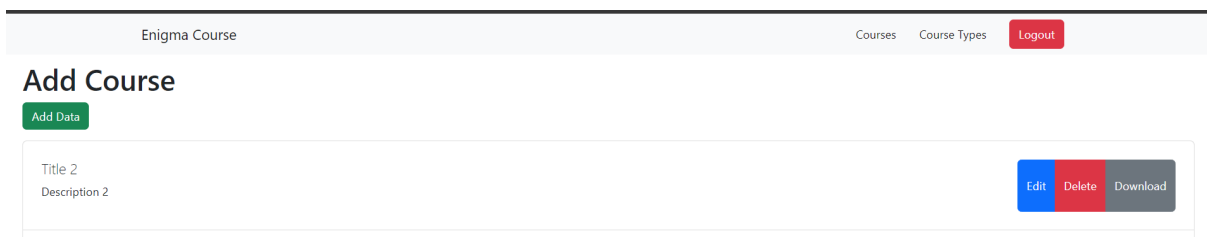
Kita akan menggunakan fungsi yang sering digunakan di project, seperti navigasi, mengambil parameter, mengambil rute saat ini.

AppNavigation navigations, untuk menggunakan react router kita perlu membungkus komponen aplikasi kita di dalam komponen BrowserRouter, dan Router, kemudian mendefinisikan komponen yang ada didalam aplikasi kita.

App, merubah fungsi navigasi menjadi menggunakan useNavigate, maka dari itu kita perlu refactor di komponen App, dan di komponen yang menggunakan fungsi onNavigate.

Navbar component, komponen ini menampilkan menu halaman yang ada di aplikasi, dan juga menampilkan tombol logout.

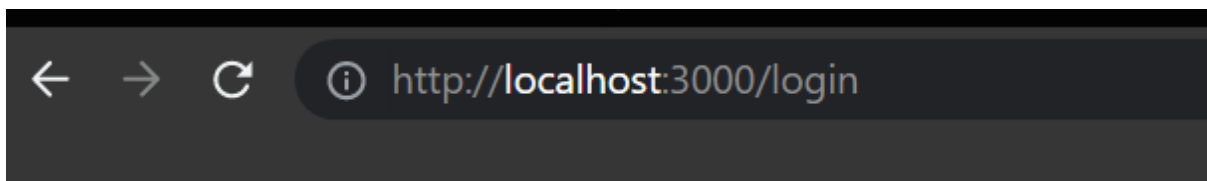
Hasil



07 Protected Routes

- Membuat halaman yang diproteksi

Ketika membuat aplikasi website, halaman akan dapat diakses melalui input url di browser



Perlu diketahui dan diingat bahwa halaman itu ada yang dibuat agar dapat diakses oleh publik atau hanya oleh orang yang punya akses misal admin atau user tertentu, maka dari itu kita perlu memastikan halaman mana yang seharusnya tidak dapat diakses oleh sembarang orang, sehingga kita perlu melakukan konfigurasi untuk proteksi halaman / rute.

Dengan menggunakan react router kita dapat melakukan ini dengan memanfaatkan children route.

AppNavigations navigation, kita membuat sebuah state untuk menampung status login atau belum login dan membuat parent route yang diisi komponen **ProtectedRoutes** dimana nantinya komponen ini akan menentukan menampilkan komponen anak.

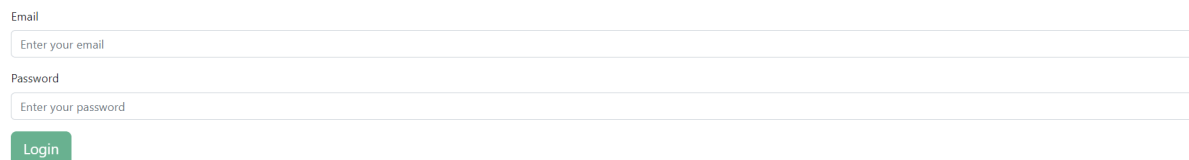
Kita juga menambah konfigurasi rute untuk screen **Login** dan **NotFound**.

ProtectedRoutes component, berisi logika untuk mengatur akses user apakah mempunyai akses pada halaman di aplikasi, kalau tidak maka akan di redirect ke halaman login untuk mendapatkan akses ke dalam aplikasi.

Login page, digunakan untuk mendapatkan akses si user ke dalam halaman aplikasi, di branch ini masih menggunakan state untuk mendapatkan dan menyimpan akses user.

Hasil

Login Form



Email

Password

Login

08 Interceptor

- Menambah konfigurasi untuk interceptor di request axios
- Menyimpan, mendapatkan dan menghapus token di localStorage

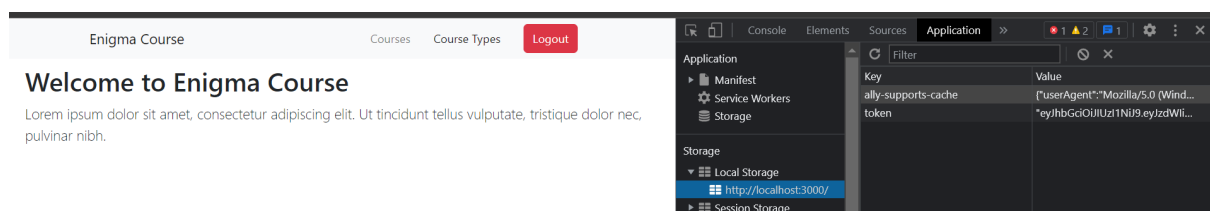
Token utils, fungsi untuk menyimpan, mengambil dan menghapus token dari localStorage. Data yang disimpan di localStorage harus berbentuk string maka dari itu kita perlu merubah token menjadi string menggunakan fungsi JSON dan ketika mengambil kita perlu merubah dari bentuk json menjadi bentuk biasa (misal objek, atau yang lain).

Login page, kita membuat fungsi untuk melakukan permintaan login ke server, ketika berhasil data / response yang diterima (berupa token) disimpan di localStorage karena token ini akan digunakan setiap kali kita mengirim permintaan.

NavBar component, mengatur fungsi logout agar menghapus token di localStorage, dan buat user di redirect untuk login ulang.

ProtectedRoutes component, setiap kali halaman diakses user kita perlu memastikan bahwa user sudah login yaitu dengan cara memeriksa token ada di localStorage atau tidak, kalau ada maka halaman yang dituju dapat diakses kalau tidak ada maka akan diarahkan untuk login terlebih dahulu.

Hasil



09 Unit testing

- Menggunakan `@testing-library/react` untuk testing komponen react, kita melakukan test berdasarkan interaksi / simulasi oleh user.
- Menggunakan `@testing-library/user-event` untuk testing simulasi berdasarkan event input dari user.

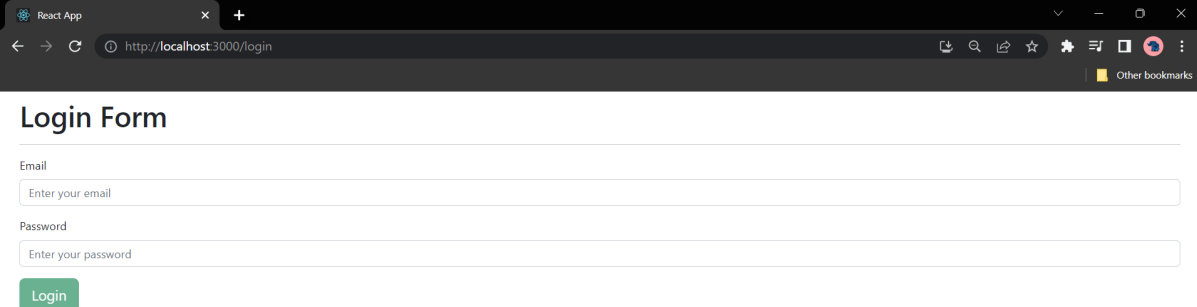
Yang perlu kita test / pastikan adalah outputnya dimana berdasarkan inputan yang diberikan, misal:

- Fungsi utils, test hanya dengan menggunakan jest (test runner). Memastikan apakah yang direturn sesuai ekspektasi apabila dijalankan entah itu diberi parameter atau tidak tergantung fungsinya
- Component, perlu melakukan snapshot terhadap komponen, agar perubahan konten dan style yang di render dapat kita ketahui untuk menjaga perubahan kode / output yang tidak sengaja.

Lebih lengkap baca dokumentasi resmi, <https://testing-library.com/docs/>.

Hasil akhir aplikasi

Login Page

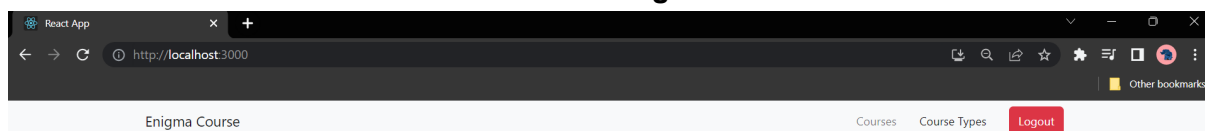


A screenshot of a web browser showing a login form. The browser's address bar displays 'http://localhost:3000/login'. The form is titled 'Login Form' and contains two input fields: 'Email' with the placeholder 'Enter your email' and 'Password' with the placeholder 'Enter your password'. Below the password field is a green 'Login' button.

Not Found Page



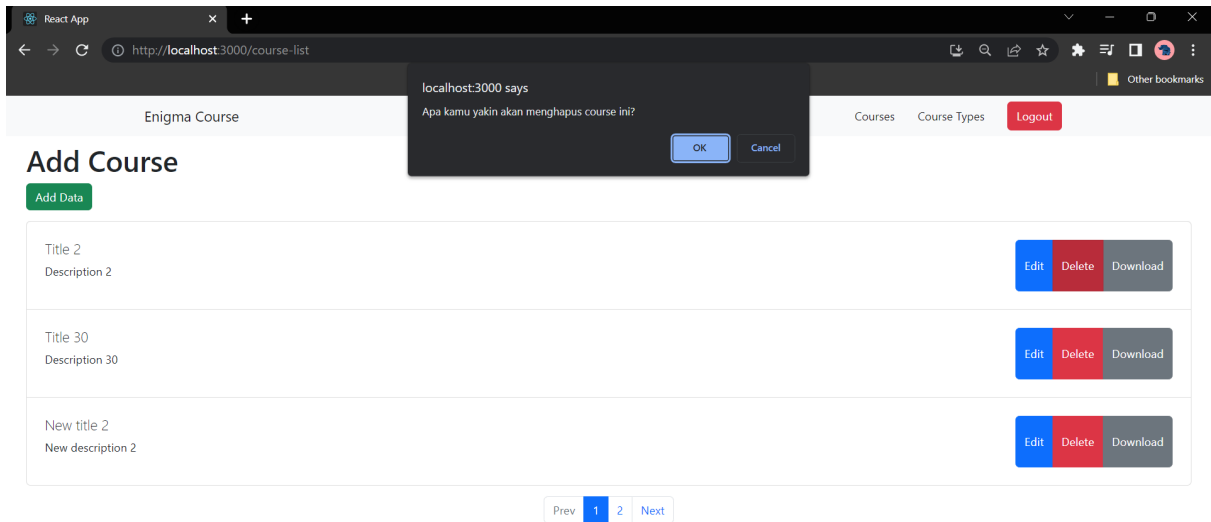
Home Page



Welcome to Enigma Course

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut tincidunt tellus vulputate, tristique dolor nec, pulvinar nibh.

Course List Page dan Popup delete



Add Course dan Edit Course Page

The screenshot shows the "Edit Course Page" with the URL `http://localhost:3000/edit-course`. The header is identical to the previous page. The main content area is titled "Edit Course Page" and contains the following form fields:

- Title:** A text input field containing "Title 2".
- Description:** A text input field containing "Description 2".
- Course Material:** A file upload section with a "Choose File" button and the text "No file chosen". Below it, the file path `D:\learning\java\demo-spring-boot\assets\Challenge - Robot 2.png` is displayed.
- Level:** A text input field containing "Basic".
- Duration:** A text input field containing "5 Jam".
- Course Type Id:** A dropdown menu with "BACKEND" selected.

At the bottom of the form are two buttons: a green "Submit" button and a grey "Cancel" button.