IMY 220 Project 2025

Version Control Website

Deliverable 1

Instructions

- This deliverable must be completed by 07:30 on 5 September 2025 as indicated on ClickUP.
- You must demo your deliverable submission in the demo sessions in order to be awarded marks.
- You must demo your deliverable using Docker, not running the project locally will be permitted.
- No late / email submissions will be accepted.

We recommend you constantly refer to the overall spec as well as your individual deliverable specs to make sure your project meets all of the requirements by the end.

Make sure you also continuously test your project as you add new functionality to make sure everything keeps working as expected.

Part 1: Basic React Project

You need to set up a basic React project using webpack and babel as taught under *NodeJs and React* on ClickUP. You can't use a bootstrapping tool like create-react-app or Vite to achieve this. You are required to follow the way that is taught to you.

You do not need to wait for the React NodeJS lectures in order to get started. You have enough knowledge about components to get started on creating those files.

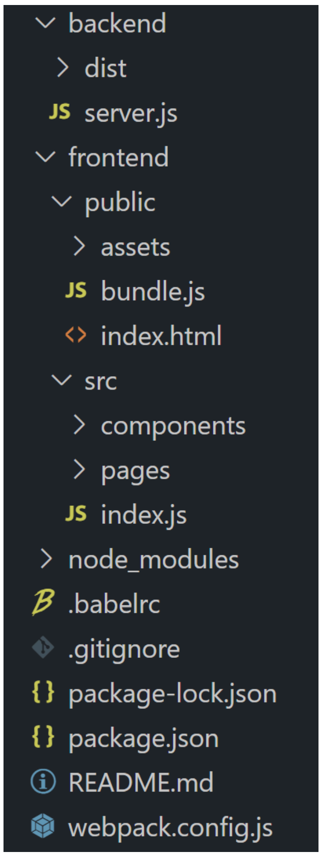
While setting up your project, you should separate your code into various files. You are required to have the following folders and files but you are not limited to these. You are allowed to create other folders to separate your code even more to make it easier for you.

1. In the root of your project, create a folder called *frontend*. Place your public and src folders - with everything currently inside them - in this folder. All your React code will live.
2. In the root of your project, create a folder called backend. Place your server.js in this folder. This is where your *backend* - Express and MongoDB - code will live.

3. Inside your frontend/src folder, create two sub folders called components and pages for your different React components and pages respectively. This will help you better organise your code. Your main index.js file should still remain in the src folder, not in any sub folders.

4. Inside your frontend/public folder, create a folder called assets, and another called images inside of assets. This is where your website images (e.g., background images, etc.) should live.

Your final folder structure should look something like this but would include any other folders you created.



```
backend
  dist
  server.js
frontend
  public
    assets
    bundle.js
    index.html
  src
    components
    pages
    index.js
  node_modules
.babelrc
.gitignore
package-lock.json
package.json
README.md
webpack.config.js
```

Part 2: Frontend Components with Dummy Data

When it comes to components, you want to follow Separation of Concerns. One file for each component and each component does a specific thing, similar to OOP-programming. You will need to decide what goes into each component and how you split up the functionality, but this section provides you with the bare minimum.

This list is not a list of the required content on each page, but rather what content must be encapsulated in a component form rather than just being on the page. You may add extra components if you wish. The theory behind components is that components should be DRY (Don't Repeat Yourself) and reusable. If there is a component that suits the needs of multiple cases, you should use that component.

You should refer back to the main project specification to see the specific requirements for each page and that should guide you on what should be included in each component.

Your HTML should make use of semantic HTML elements (nav, main, aside, article, etc.) where appropriate. This is good practice.

General

- Project Preview component - Should provide an overview of a project, would be useful for the feeds and profile pages.
- Create Project component - Should provide a way for a project to be created.
- Profile Preview component - Displays all basic information on a profile that is viewed in follower / following lists, search results, etc. - i.e., not the entire profile, but a small preview of it.
- Header / Navigation component - Contains your page routing - the ‘navbar’ and must be present on all pages besides the Splash page.

Splash Page

- Login Form - Contains all the information / inputs required to log in, stub our endpoints. [refer to part 5]
- Sign up Form - Contains all the information required / inputs to sign up, stub out endpoints. [refer to part 5]

Home Page (with Feed)

- Feed component - Displays the project preview components, this can be one component that handles the display of both feeds, or a separate component for each.
- Search Input component - Handles search inputs, the search functionality does not need to be implemented yet, you only need to be able to enter a search term based on your planned functionality.

Profile Page

- Profile component - Contains basic profile information.
- Component to edit a profile - Contains all the form information for editing the user’s profile.
- A component to list all the projects that a user has / contributes to.
- Friend components - Displays all users that the current user is friends with.
- Component to create a project - Contains all the form information for adding a project.

Project Page

- Project component - Contains all required information on a single project.
- Files component - Displays all the files in the project.
- Messages component - Displays all the check-in/check-out messages for the project.
- Component to edit a project - Contains all the form information for editing a project.

Part 3: Basic Functionality

All of the components listed above do not need to be functioning yet.

However, some components should have basic client-side functionality. This is to get you started on implementing some of the functionality you need before you are required to implement backend code in Deliverable 2 & 3.

Choose two of your form components you have created (i.e., any components that make use of HTML inputs) and implement basic client-side validation for them. This can either be done with Javascript, HTML input validation (MDN Docs) or a mixture of the two.

- For example, checking if a value filled in is the correct number of characters or has a required symbol (e.g., @), checking if password and repeated passwords match, checking if input fields are not empty, etc.

- You may pick any of your form components, e.g., components with inputs to add / edit a piece of content, a login form, etc.

- You do not need to do any validation that requires a backend. However, you will be required to make use of either Javascript (state) or HTML input validation to ensure that the validation functions correctly (i.e., the form is not able to submit until all required fields are valid).

Part 4: Routing

Set up basic page routing to access each of your main pages via the URL. You do whatever method you prefer, but using *react-router* is advised.

- What is important is that a user is able to navigate to the Splash page, login / sign up forms (if they are not on the splash page), Home page, Profile page and Project page via the URL (i.e., "/", "/home", etc.). You may add any additional routing to any other pages you have if you wish to do so.

- Your Profile page and Project page should make use of dynamic routes. Eventually, you will need to be able to access a specific project or profile through its id in order to render a specific profile using the same component template. This is done by passing an id through the URL. For now, you can render the same profile regardless of which id is passed through the URL, but you should be able to handle passing any id (e.g., “/profile/23532”).

Part 5: Authentication Endpoints Stubbing

Stubbing is when you create endpoints that have no actual functionality, but rather return dummy data. Since you have no database yet, you can’t actually make users and store them somewhere.

With this in mind, you will need to set up your basic ExpressJS server. You do not need to add any database functionality, just get it set up so that you can implement endpoints in the future deliverables. So, similar to how you are setting up your frontend files and folder structure, do the same with your ExpressJS server to save you time later on.

For this deliverable, you should stub out your authentication endpoints. You should have at least the *sign-in* and *sign-up* endpoints stubbed out, returning dummy data. The request should be like the actual request when you come around to this by doing the following:In your frontend, you should implement these endpoints into your frontend, into your sign-in and sign-up forms, so that it actually makes the API calls and handles the response. At this stage, you should also be thinking about how you are going to identify the user on your frontend and backend, e.g, storing some details in local storage or storing some kind of token in cookies.

Part 6: Docker

Write a Dockerfile (and dockerignore) to run your application. Your entire application should run on Docker. Include any commands that you need in a *README.txt* file so that you can easily reference them later on. Your Dockerfile should be able to run your application. The application should be accessible from the port that you specified.

You should test your application thoroughly before Dockerising your application. If something is not working in your local environment, it probably won’t work in Docker. Test your application in both local and in Docker, as there can sometimes be differences between how certain operating systems handle certain things.

Important! You need to ensure that your application works using Docker. If it does not work, you won’t be able to receive any marks on the marking day.

Submission Instructions

Place these in a folder named with your.

- Upload all your files, excluding *node_modules*, and including all relevant files to build and run your Docker container.
- Include a README with the commands that you use to build and run your Docker image.
- Include a *README.txt* file with a link to your GitHub repository. At this point you should have two extra commits (you can commit these to your main branch)

Zip the folder - into a ZIP file called *Position_Surname_D1* - and upload this to ClickUP in the relevant submission slot before the deadline.