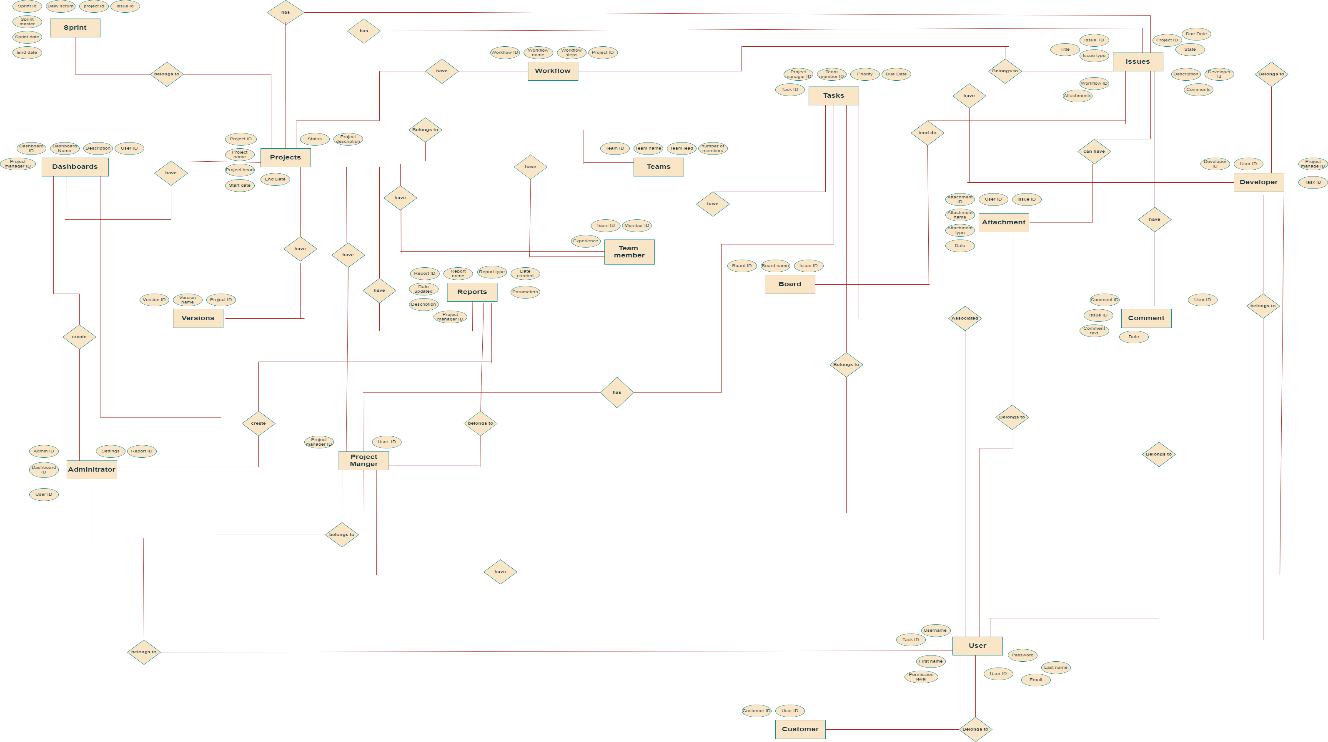
**REPORT FOR DATABASE PROJECT**

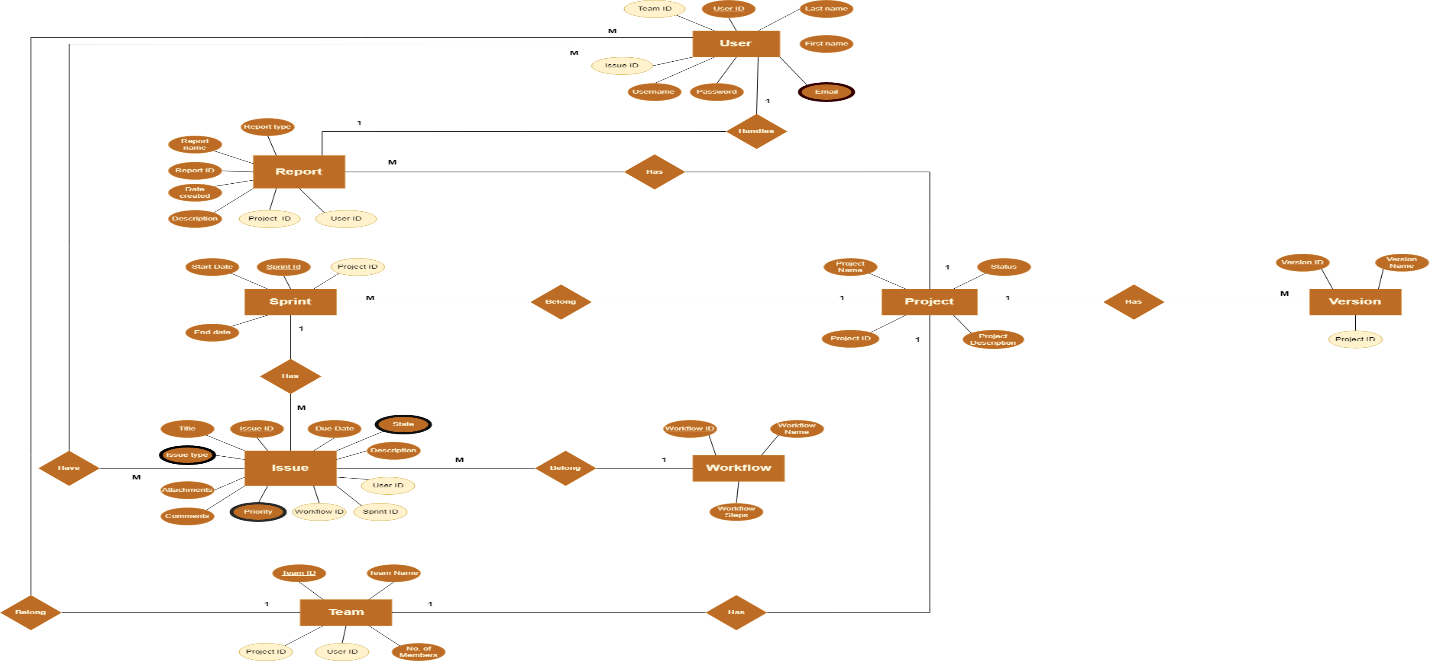
**Eman Tahir i21-1718:**

**Part Attempted: Working with SQL (Back-End)**

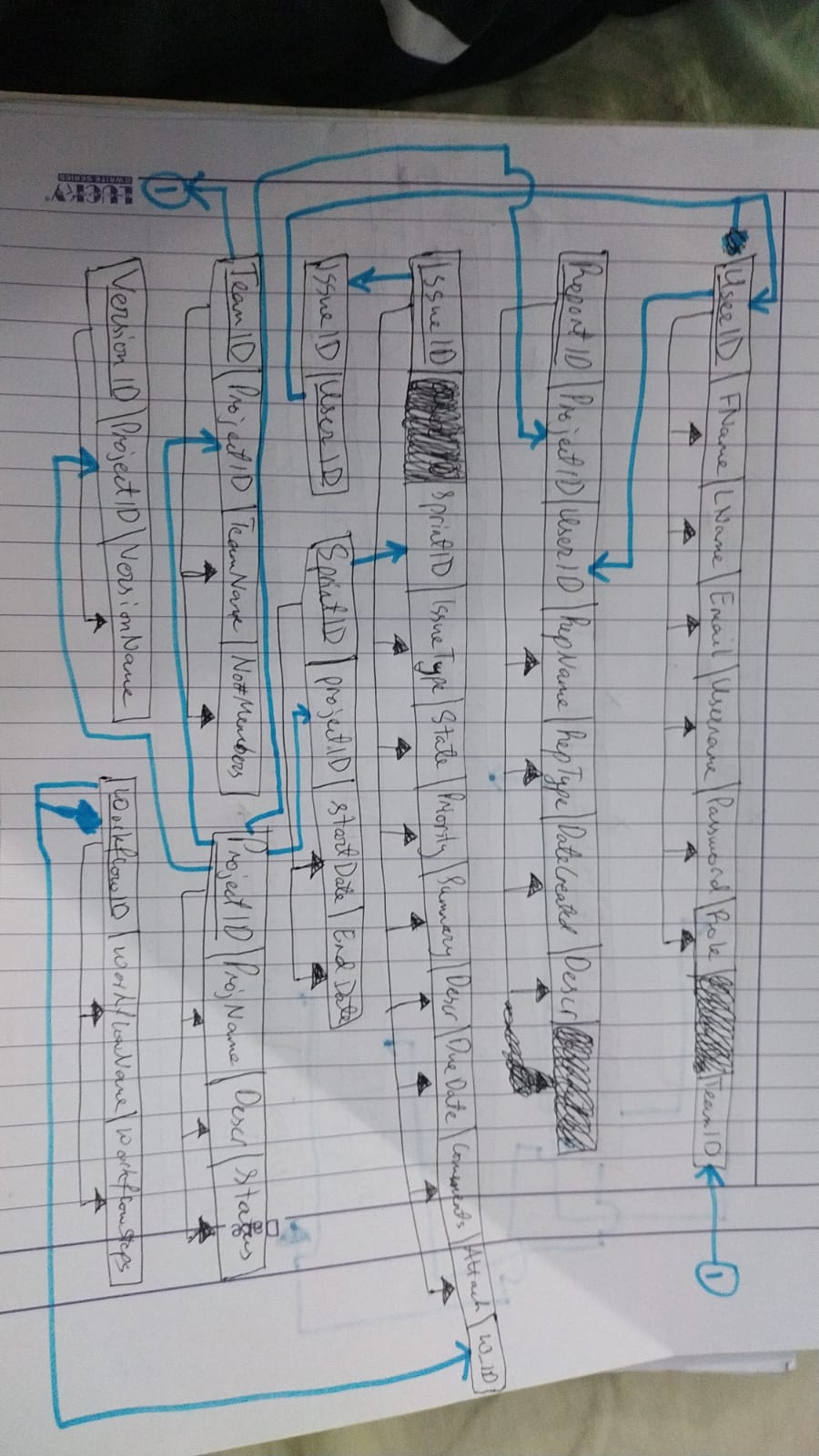
The first challenge that I faced was normalization of the tables. Before normalization, there were so many extra entities which lead to the cluttering up of ERD. A picture is attached for reference.

A word document describing all the entities, their attributes and their cardinalities is also submitted, just to better understand the working of the ERD.

After this, with the help of my group mates, normalization was done. After discussion with the teacher, some extra entities were removed. This aided in a quite easy and simple ERD. It is also attached for reference.

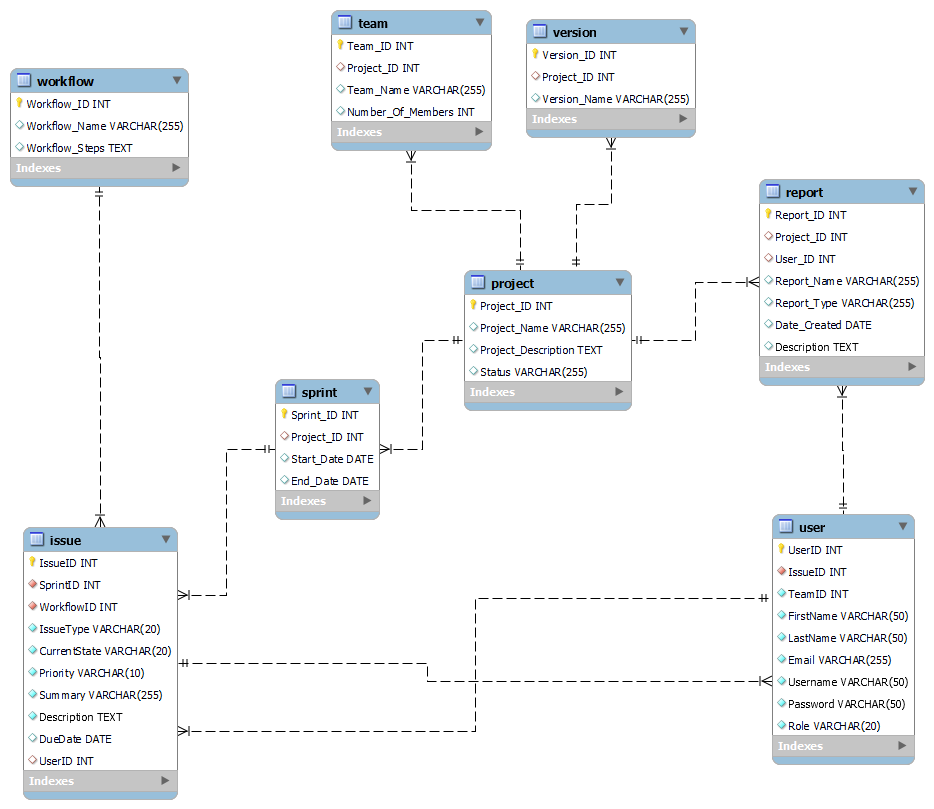
****

Here is how we did normalization (primarily Afaq i21-1700):

****

After this I wrote the sql files for table creation and insertion of dummy values keeping in view the case study.

However, the values that I have inserted are based upon a DATASCIENCE working project (as an example), so it looks more realistic as this is our field. At the end I used the reverse engineering feature of mySQLWorkbench to create the relational schema of our database. A pic has been attached as reference:

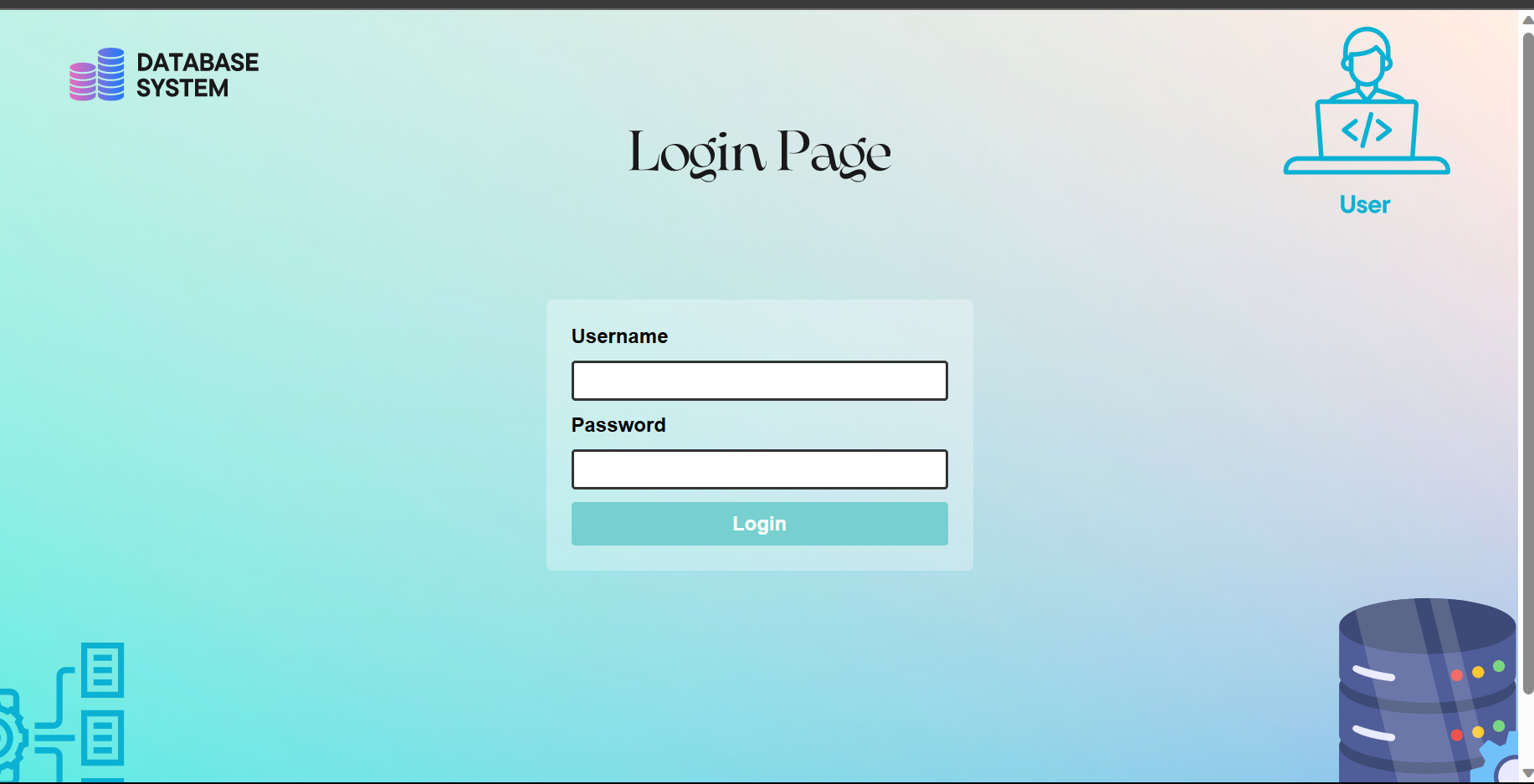


This was all for my part.

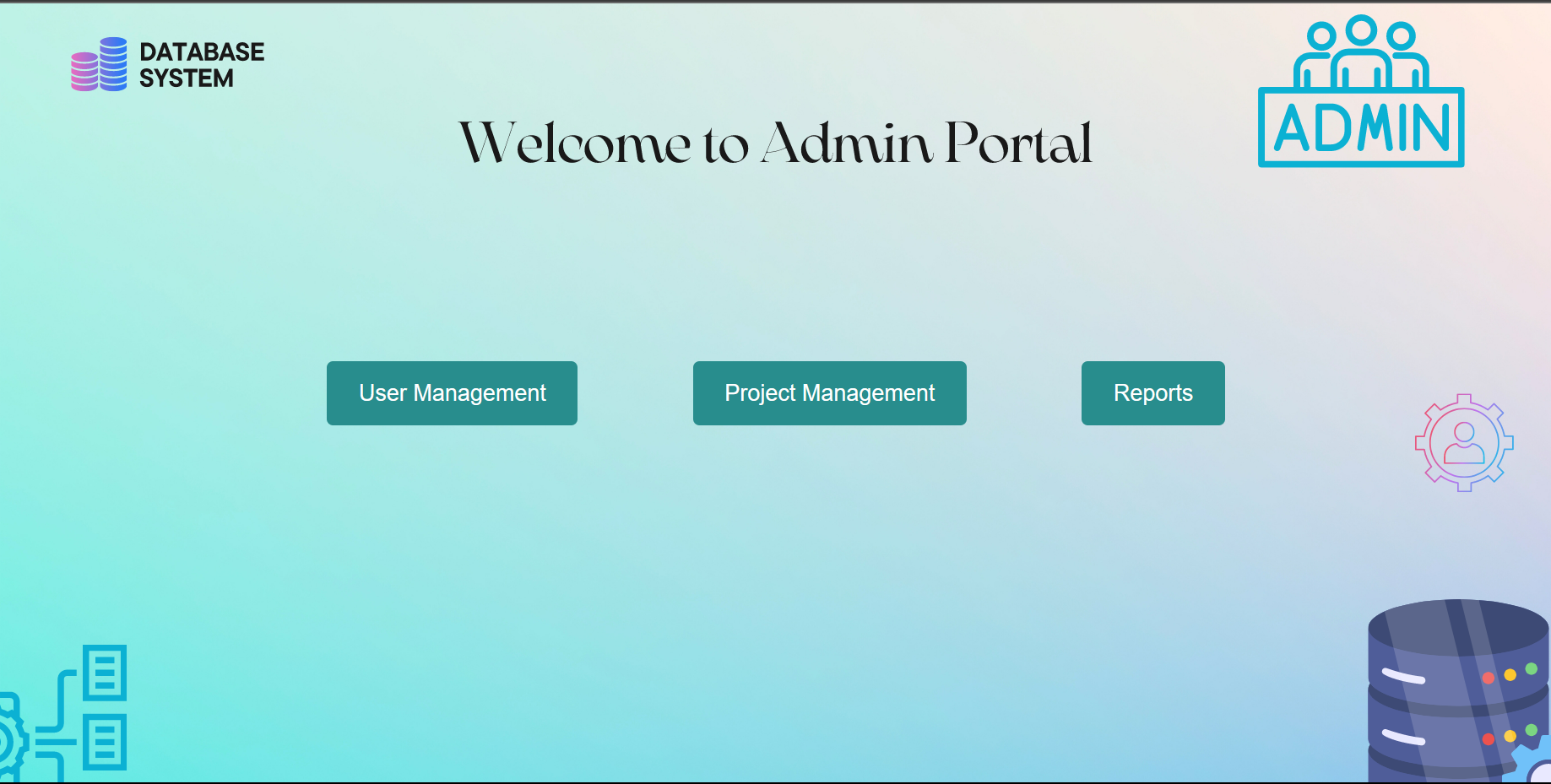
**Hammad Sikandar i21-1684:**

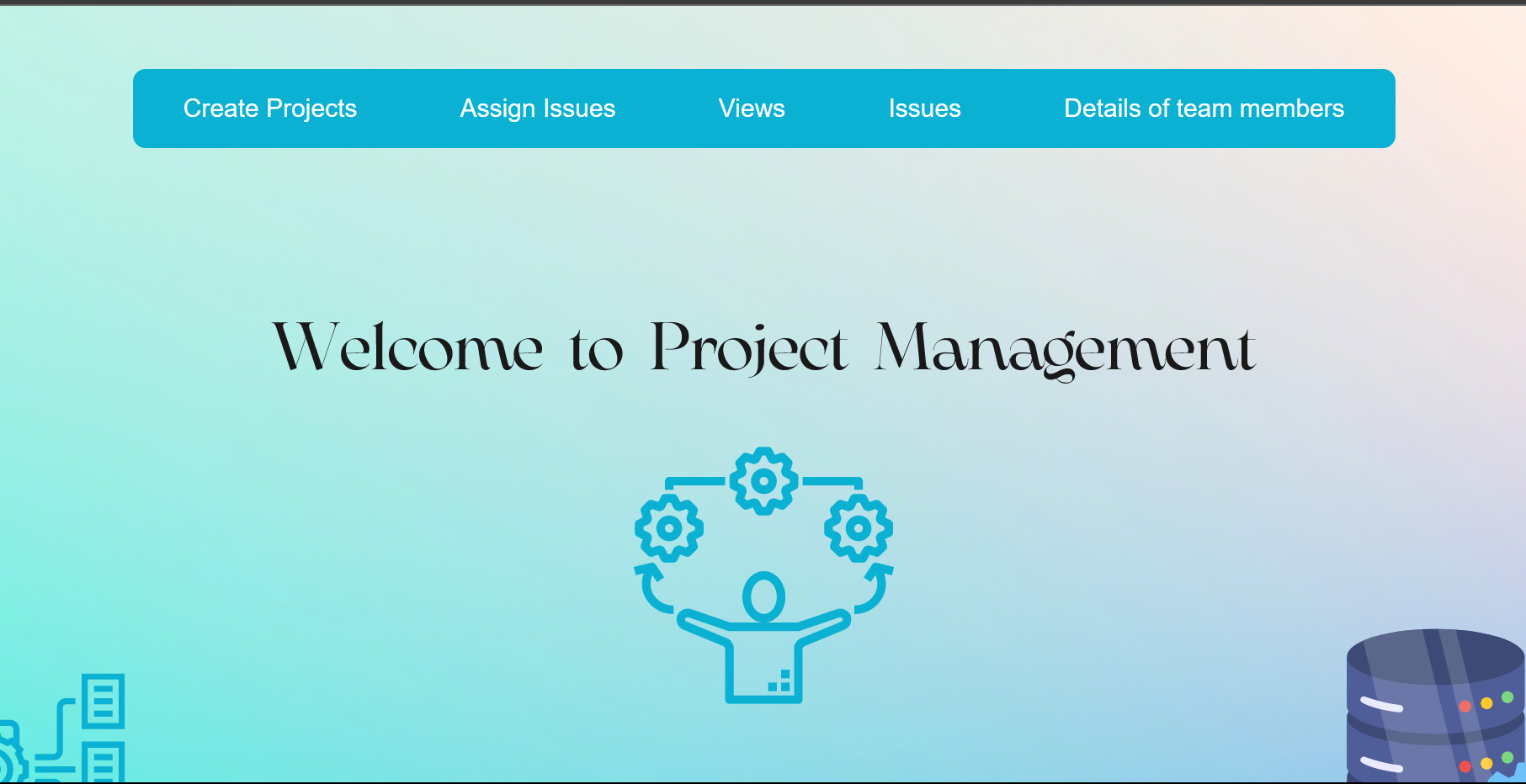
**Part Attempted: Website Development (Front-End)**

My part was to develop a **front-end** of this project, the initial task I worked on was the **login page**, which was designed to authenticate users and route them to their respective portals based on their credentials. This page was pivotal in maintaining the security and integrity of the system by ensuring that users only had access to the features and functionalities of their respective roles.

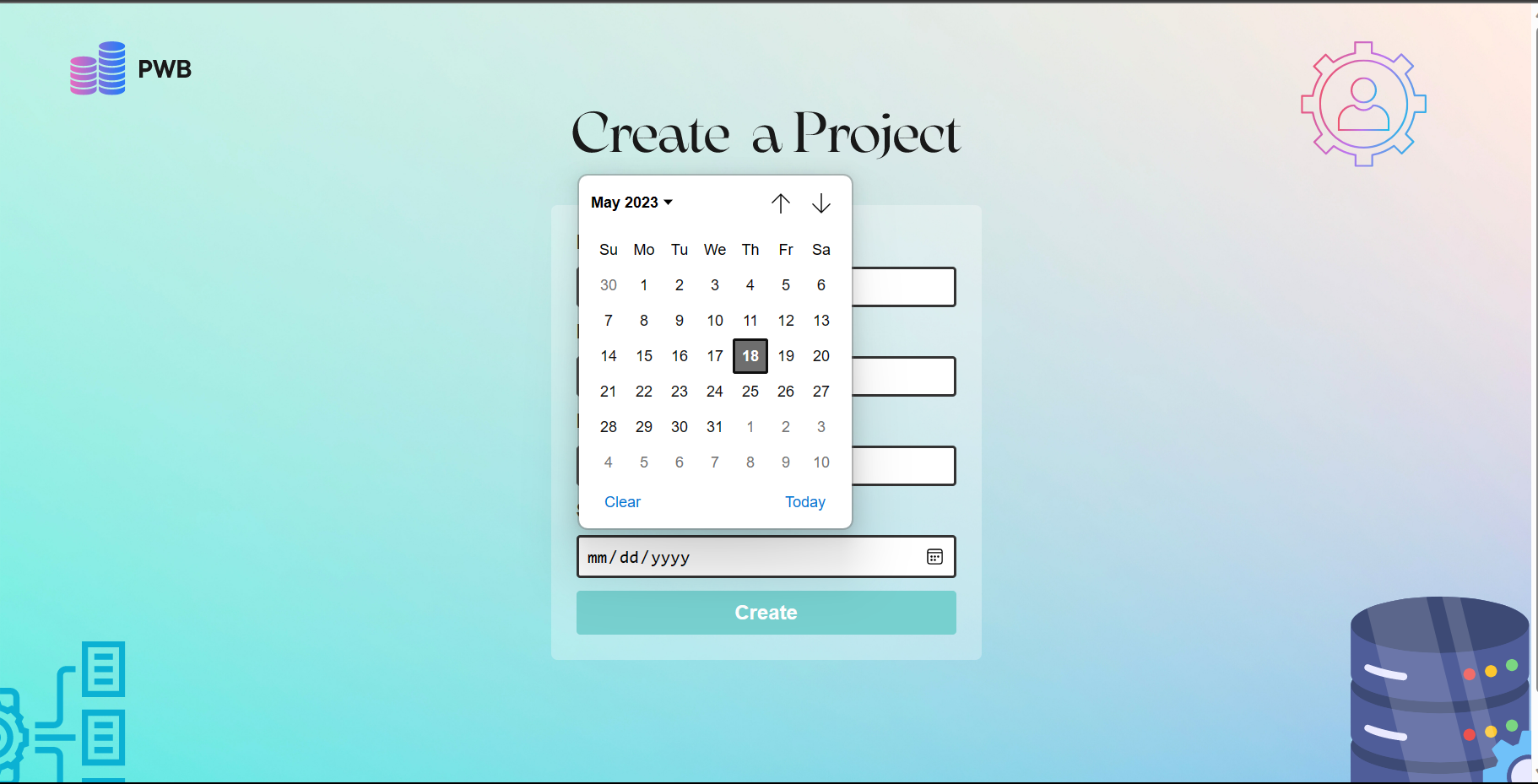


Once authenticated, **Admin** users were directed to an **admin portal**, where they were granted full access to all system functionalities. I designed this portal to be comprehensive yet easy to navigate, ensuring that user management, reporting, and project management features were readily available. These features were encapsulated in a sleek and professional design that underscored the seriousness and authority of the **admin role**.



The **Project Manager portal**, on the other hand, was designed with project management in mind. The user interface I created allowed **Project Managers** to create **new projects, assign issues, view team member details**, and more. This portal was designed to be intuitive and straightforward, enabling Project Managers to effortlessly coordinate their teams and track project progress.

For **Developers**, I designed a portal focused on task management and status updates. This portal provided a clear overview of the tasks assigned to each developer and offered an easy-to-use interface for updating task statuses. This portal was designed to be minimalist and efficient, reflecting the role of Developers in the project.

Lastly, the **Customer** portal was designed to enhance the user experience for customers. **Customers** were provided with functionalities to create new issues, track issues, and report problems. The design of this portal was focused on simplicity and ease-of-use, ensuring that even users with minimal technical knowledge could easily navigate and use it.

**Afaq Alam i21-1700:**

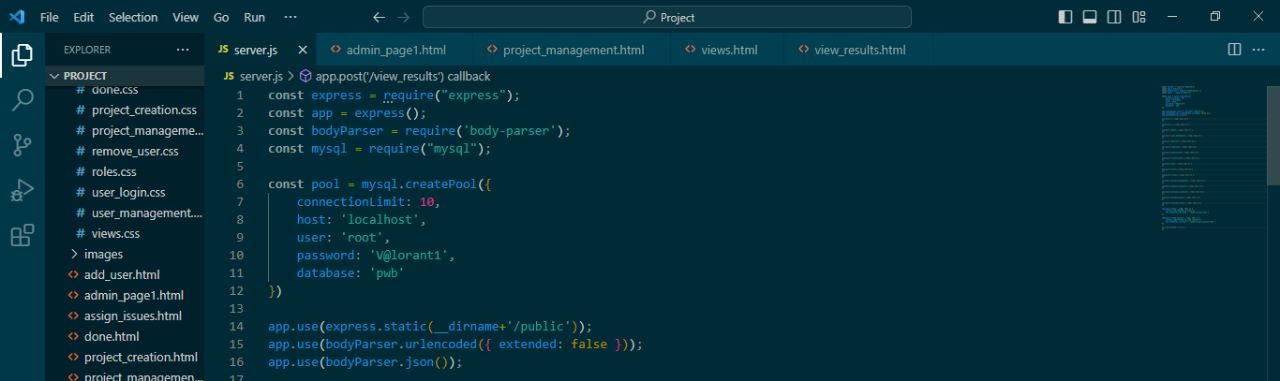
**Part Attempted: Connection between front end and back end with NodeJS**

The first challenge that I faced was that NodeJS executes code asynchronously because of which the queries were executed at the end, basically after the execution of the latter code. As a result, the variables that I made to keep in the fetched value from the database were getting the required results quite late, which of course became a hassle.

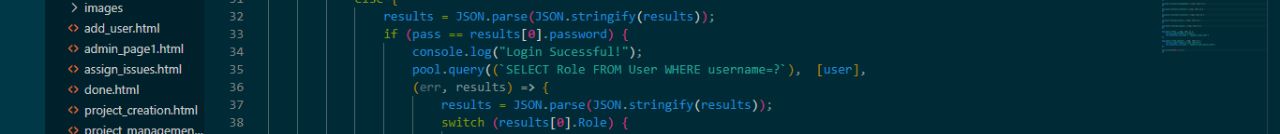
To solve this, I wrote my code block inside a function, which I then placed inside the queries which was again difficult and demanded a lot of time and effort.

The second challenge was that after fetching the required results from the mysql database, named PWB. I was getting a weird error, which said “undefined” as its description.

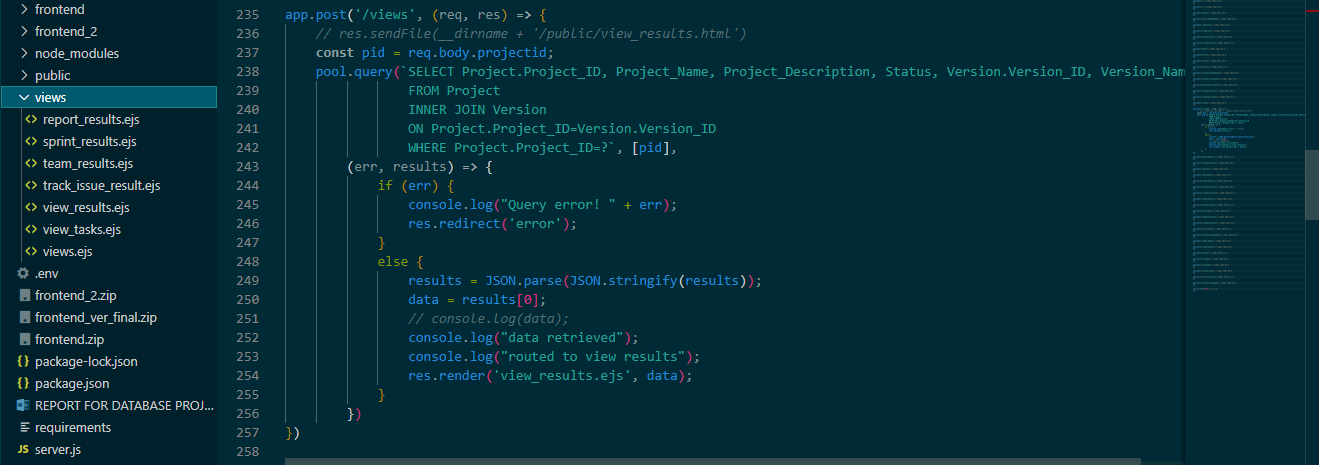
To solve this, I imported “body-parser” module. A picture has been attached for reference.

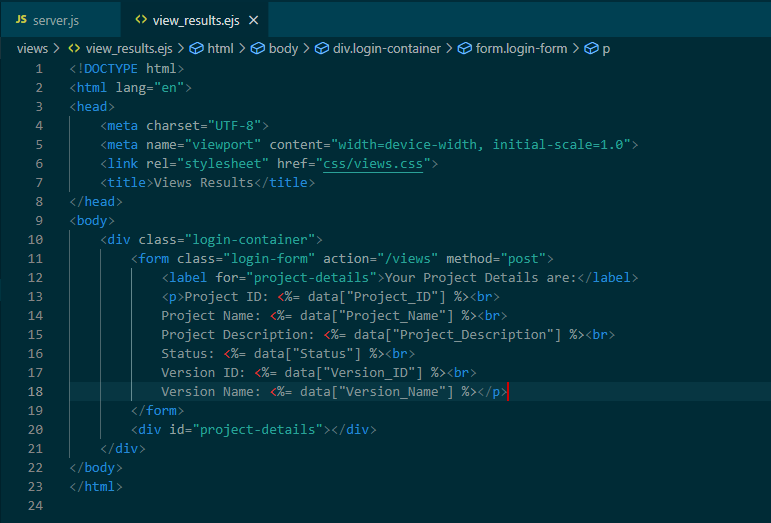


Then I used JSON format to read the values that were previously fetched from the PWB database. One of the code lines where I have used it is attached for reference.



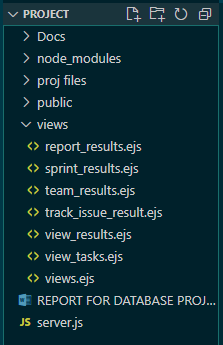
Another hurdle I encountered was how to send data from the JavaScript file to the html file and display it. Sadly, there was no easy solution, so instead I opted to use ejs files, with the help of installing ejs module and ejs tags in my ejs file which is quite like html and the only difference being the ejs tags. I was able to easily display the fetched results onto the webpage. The following are some screenshots for reference.





Lastly, I discovered that for Node.js to see the .ejs files, my folder had to follow a certain hierarchy. Which was as followed:

* *Root folder: This is the main directory of your Node.js application.*
* *node\_modules: This folder contains the dependencies installed via npm (Node Package Manager).*
* *package.json: This file defines the metadata and dependencies for your application.*
* *server.js or app.js: This is the entry point file where your application starts.*
* *public: This folder may contain static files like HTML, CSS, and client-side JavaScript.*
* *views or templates: This folder holds the views or templates used for server-side rendering.*
* *public/assets or static: This folder may contain static assets like images, stylesheets, or client-side JavaScript.*



After this, the path was quite easy going and I didn’t encounter anymore challenges.