School of Computing

CA400 Year 4 Technical Spec:
Expo +

Genesis Uwumangbe 20459666 & Gideon Amaechi 20364806

Date created: 07/01/24

Table of Contents

1. Overview

The purpose of our project is to assist people in navigating their way through the official DCU project expo by using the DCU Expo+ application. This application aims to replace the physical hanbooks used for the expo which in this day and age of modern technology can only be described as outdated. With an application like this users will be able to upload their projects (if they are a student involved in the expo) and view other projects in a timely and simplistic manner which will be widely accessible through the internet on multiple devices such as phone, laptop or desktop.

The audience in which this project primarily caters for is the viewer which is more often than not companies who are coming to view students projects along with general viewers, with this being in mind and the fact that the expo is heavily technology based it only makes sense for information about it to be optimised for online usage.

The features in this project involve the use of a recommendation system which is used to display projects to the user based on confidence of certain projects that were selected together previously and displays them together based on what we believe the user would share an interest in.

An interactive map which is used to highlight the locations of the selected projects by the user while also containing information about the project and the users who created it.

A preference section upon first sign in which will help to gauge a better understanding of the users preferences in order to show them the projects which they are believed to have the most interest in. Machine learning is used to handle this task.

Opinions of the user in regards to how the application could be improved in the future is accepted through the usage of a feedback firm where users can give their honest feedback and also give an overall rating of the application overall.

In our project, machine learning serves as the backbone of our recommendation system, enabling us to create a tailored and engaging user experience. By analysing user interactions and choices, our system continually evolves its recommendations. We sift through extensive user feedback and encompassing ratings to fuel our machine learning models. These models undergo continuous refinement, adapting to user preferences over time. Employing advanced techniques such as collaborative filtering and content based filtering, we unravel complex patterns within the data. By integrating machine learning across our platform, we not only deliver recommendations aligned with individual tastes but also ensure the seamless evolution of our system, culminating in heightened user satisfaction and engagement.

2. Motivation

The main reason as to why we chose to do this project is because when we were searching through past DCU projects looking for inspiration on what to do for our own, we found that the whole layout of the information available from the past expos was too old fashioned. The expo itself is used to display a multitude of technological projects but yet the booklets used to inform the attendees is poorly organised and makes it difficult for users to find the locations of the projects in which they are looking for. With this in mind we found that this was an area in which we could use our skills that we have been building since our time in college to upgrade this as it is not just a one time usage type of ordeal, but rather it is is something that would actually be of benefit to the school and could quite easily replace the current system at hand. This would be a long overdue yet necessary upgrade for the expo which is why we decided to go down this route.

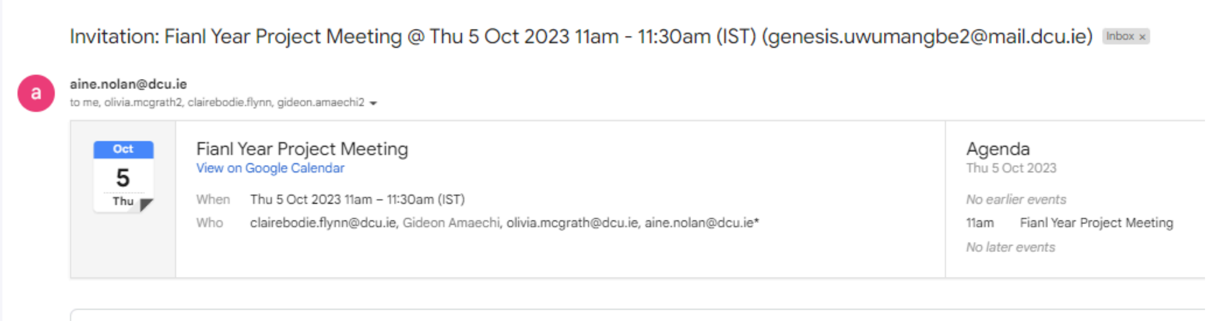
3. Research

This section details the various components of the application that required us to do a lot of research. This allows us for the discovery of multiple new approaches.

3.1 Contacting the Organizers

The Research for this project was done in different phases with each serving a purpose of its own. The phases involved getting information from the ones in charge of the expo in regards to how the expo is organised and what the booklet looks like and finally getting information in regards to the downsides of the current format used to display information.

We Got in contact with the people responsible for running the expo each year and had set up scheduled meetings with them so that we could get as much information about the running of it, this way we could understand in detail what approaches toward this project that we should take. They gave us access to the data for last years expo
 Final Year Project Expo 2023 - (Responses)



Invitation: Fianl Year Project Meeting @ Thu 5 Oct 2023 11am - 11:30am (IST) (genesis.uwumangbe2@mail.dcu.ie) Inbox ×

aine.nolan@dcu.ie
to me, olivia.mcgrath2, clairebodie.flynn, gideon.amaechi2

Oct 5 Thu

Fianl Year Project Meeting
View on Google Calendar

When Thu 5 Oct 2023 11am – 11:30am (IST)
Who clairebodie.flynn@dcu.ie, Gideon Amaechi, olivia.mcgrath@dcu.ie, aine.nolan@dcu.ie*

Agenda
Thu 5 Oct 2023

No earlier events
11am Fianl Year Project Meeting
No later events

Finally before getting started on the project, knowing what some of the main issues that people were having with the expo was key. We found that most people struggled with the time management and not knowing which projects to initially visit as the booklet is overwhelming; they also found the map section very confusing to follow.

We then looked into what frameworks that we would use to achieve our application

3.2 Front end framework

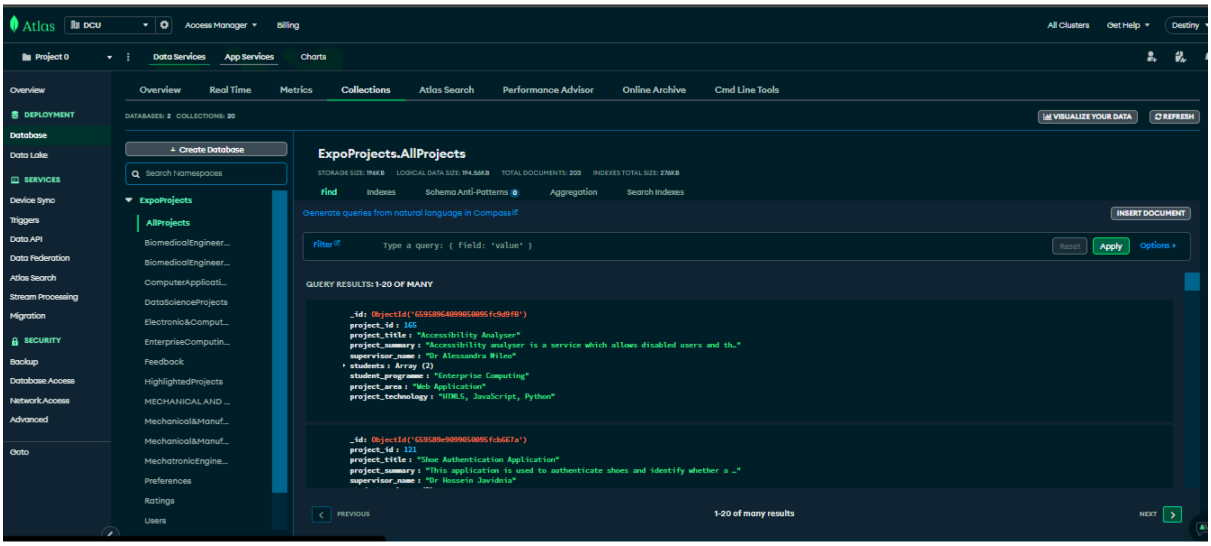
When starting our project we had to decide on what framework we would use for the front-end of our application. When deciding this we had to consider whether we would be making a mobile application or it would be a web browser application that mobile users can access. We originally started with reack.js but then later switched to next.js due to its server side rendering and easy routing, flexibility and many libraries, next.js allows you to develop for both mobile and web simultaneously. This would allow us to provide Expo+ to both mobile and web users, however we did encounter some problems neither of us had much experience with javascript/next.js so developing became a challenge.

3.3 Backend framework

In the development of our project, the selection of Flask as the backend framework was a strategic choice, particularly for its lightweight nature and its seamless integration with machine learning libraries. Flask's minimalist and opinionated framework provides the flexibility needed for rapid development and prototyping, which is essential for integrating and testing machine learning models. Its compatibility with Python's robust ecosystem of machine learning libraries like TensorFlow, Keras, and scikit-learn allows for a smooth transition from data analysis to deploying predictive models in a production environment. Moreover, Flask facilitates the creation of RESTful APIs, which is crucial for serving machine learning model predictions in real-time.

3.4 Database selection

Our project's choice to utilise MongoDB as our primary database was driven by the need for a database that could handle diverse data types and structures with ease. MongoDB, as a NoSQL database, excels in its ability to store unstructured data, offering us the flexibility to quickly iterate and evolve our data schema without the constraints imposed by traditional relational databases. The JSON-like documents created in MongoDB make the process of uploading and manipulating data exceptionally intuitive, mirroring the data structures used in our application's code, thereby streamlining the development workflow. This inherent agility is advantageous when dealing with the varied and complex data that a modern web application encounters, particularly when integrating machine learning outputs



```
_id: ObjectId('65958964099050095fc9d9f0')
project_id : 165
project_title : "Accessibility Analyser"
project_summary : "Accessibility analyser is a service which allows disabled users and th…"
supervisor_name : "Dr Alessandra Mileo"
students : Array (2)
student_programme : "Enterprise Computing"
project_area : "Web Application"
project_technology : "HTML5, JavaScript, Python"
```

```
_id: ObjectId('659589e9099050095fcb667a')
project_id : 121
project_title : "Shoe Authentication Application"
project_summary : "This application is used to authenticate shoes and identify whether a …"
supervisor_name : "Dr Hossein Javidnia"
```

3.5 Account authentication

For account creation and user authentication within our project, we have integrated NextAuth.js, a comprehensive solution tailored for Next.js applications. NextAuth.js simplifies the process of building secure and scalable authentication systems, providing out-of-the-box support for sign-in flows, third-party logins, and session management. Prioritising security, we implemented password hashing with werkzeug.security a python library for security to ensure that user credentials are never stored in plaintext within our MongoDB database. This encryption step is crucial in safeguarding user data against unauthorised access and potential security breaches.

Additionally, our system utilises JSON Web Tokens (JWTs) for maintaining secure user sessions. The use of JWTs offers a stateless method to validate user sessions, where each token contains all the necessary information to verify the user's identity. This negates the need for constant database queries, thus enhancing performance and scalability. JWTs also facilitate secure data transmission between the front end and the Flask backend, maintaining the integrity and confidentiality of the user's authenticated session

3.6 Recommendation Model

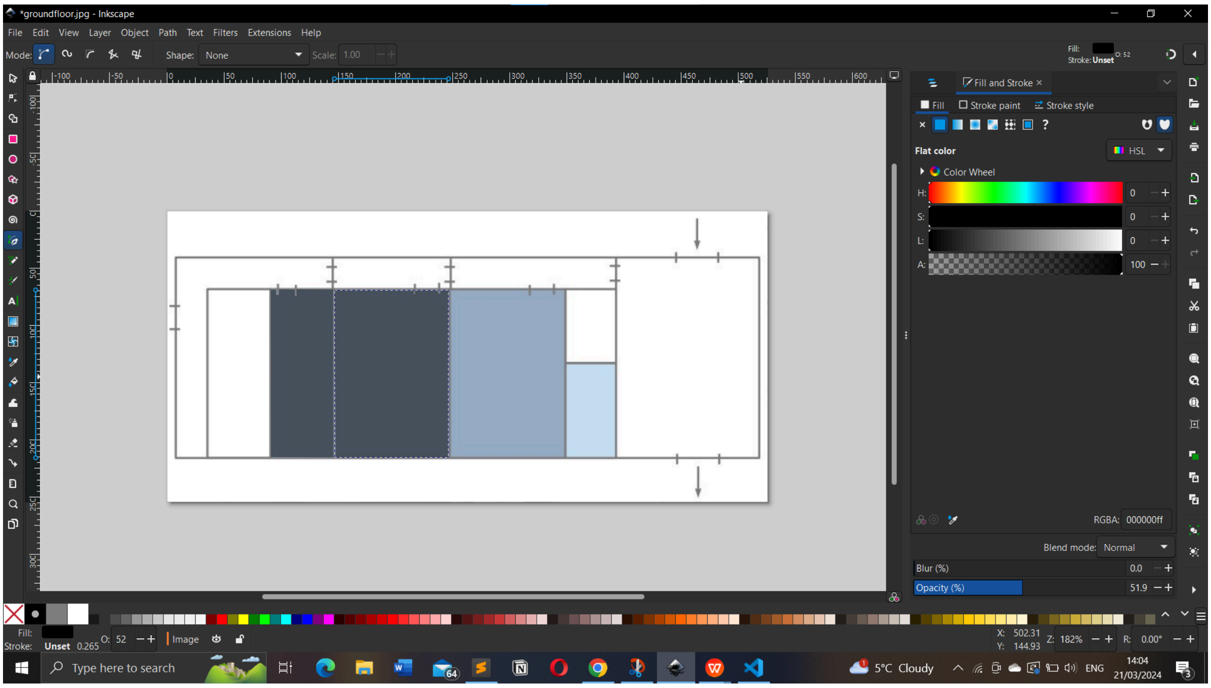
In our project, we've designed a hybrid recommendation system that takes advantage of the strengths of both collaborative filtering and content-based methods. Our project's recommendation system is pivotal in steering users towards projects that best align with their interests and preferences. It was conceived to address the challenge of information overload, where users often face difficulty in sifting through an extensive array of projects to find those that truly resonate with their aspirations and skill sets. By implementing a recommendation system, we aim to streamline the user experience, making it more engaging, efficient, and personalised

The system is designed to recommend projects that users might be interested in, based on:
User Preferences: Projects that align with the user's stated preferences in terms of programming languages, technologies, and areas of interest.

Collaborative Filtering: Projects that users with similar tastes and preferences have rated highly, suggesting a shared interest.
User Interactions: Projects that the user or similar users have interacted with in the past, indicating potential ongoing interest.

3.7 Map Creation

We wanted to use the original design of the class labs that we had access to on the expo booklet. We used it as a reference to how it would look. For the creation and integration of interactive maps within our application, we opted to utilise Inkscape, a powerful and versatile open-source vector graphics editor. This choice was made due to Inkscape's proficiency in converting bitmap images into vector paths, a feature that is incredibly beneficial when designing maps that represent various projects geographically.
Inkscape allows us to transform complex images into a series of paths that can be individually manipulated, enabling the creation of a detailed and interactive map. Each path can correspond to different projects, locations, or areas of interest, providing users with a visual and intuitive means of exploring projects based on their geographical context.
Moreover, Inkscape's capability to export these paths as SVG (Scalable Vector Graphics) images is particularly advantageous for web applications. SVGs are resolution-independent, ensuring that our maps remain sharp and clear on all devices and screen sizes, which is crucial for the responsive design principles followed in modern web development.



H: 0
S: 0
L: 0
A: 100
RGBA: 000000ff
Blend mode: Normal
Blur (%) 0.0
Opacity (%) 51.9
Fill:
Stroke: Unset 0.265
O: 52
X: 502.31
Y: 144.93
Z: 182%
R: 0.00°

3.8 Tailwind

In traditional web development, custom CSS is typically authored to style components. Tailwind CSS, however, enhances the styling process by integrating predefined classes directly within a project's HTML. This approach significantly expedites UI development.

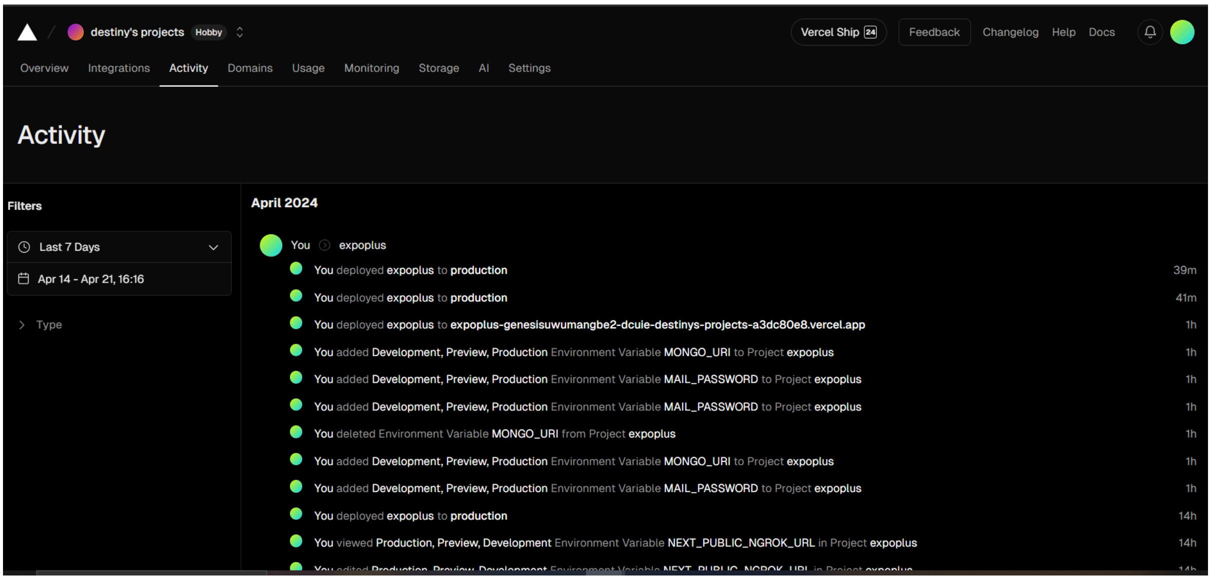
Operating as a utility-first CSS framework, Tailwind negates the need to craft unique CSS classes and styles for each component. Instead, it supplies a suite of utility classes that are assigned directly to HTML elements, facilitating a swift development process. While this method shares similarities with inline styling, Tailwind CSS offers benefits that surpass those of inline styles. Utilising its utility classes means selections are made from a standardised design system, which streamlines the creation of uniform UIs throughout the application. Moreover, Tailwind allows for state targeting—like hover and focus—right within the HTML elements, simplifying the styling workflow.

3.9 Search functionality

We intended on having a search section on our application where users can search for a particular search term and they would be given projects that are related to that search term. This helped enhance user interactivity by allowing them to quickly locate projects of interest. In designing this feature, we took a user-centric approach to ensure that the process is intuitive, efficient, and effective.

3.10 Vercel

We chose to deploy our Next.js application on Vercel due to its deep integration with Next.js, ensuring an optimal hosting environment directly from the creators of the framework. Vercel offers out-of-the-box support for Next.js features, such as server-side rendering and static site generation, enabling us to deploy a fast and secure application effortlessly. Its global Content Delivery Network ensures that our application is readily accessible worldwide with minimal latency. Moreover, Vercel streamlines our development workflow with preview deployments, automatic HTTPS, and seamless scalability. The platform's commitment to developer experience aligns perfectly with our project needs, making Vercel the ideal choice for hosting our Next.js application.



Activity

Last 7 Days

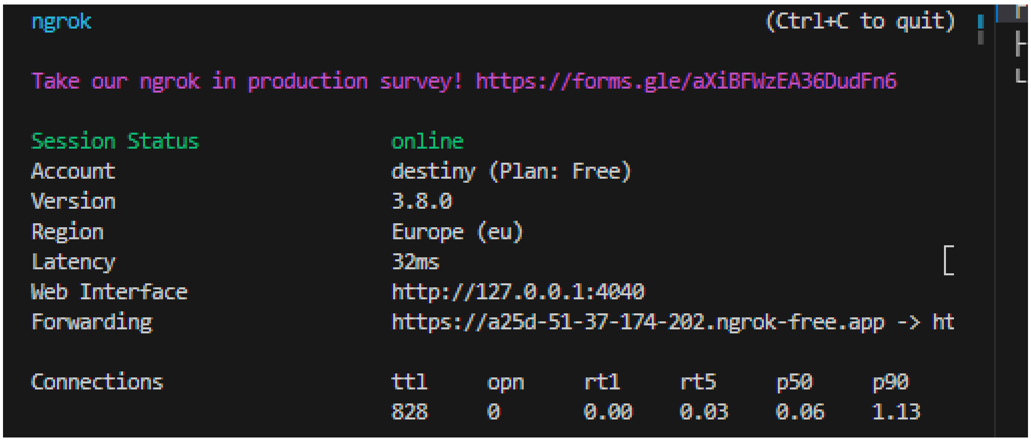
Apr 14 - Apr 21, 16:16

April 2024

```
You  expoplus
  You deployed expoplus to production                                                                              39m
  You deployed expoplus to production                                                                              41m
  You deployed expoplus to expoplus-genesisuwumangbe2-dcuie-destinys-projects-a3dc80e8.vercel.app                  1h
  You added Development, Preview, Production Environment Variable MONGO_URI to Project expoplus                    1h
  You added Development, Preview, Production Environment Variable MAIL_PASSWORD to Project expoplus                1h
  You added Development, Preview, Production Environment Variable MAIL_PASSWORD to Project expoplus                1h
  You deleted Environment Variable MONGO_URI from Project expoplus                                                 1h
  You added Development, Preview, Production Environment Variable MONGO_URI to Project expoplus                    1h
  You added Development, Preview, Production Environment Variable MAIL_PASSWORD to Project expoplus                1h
  You deployed expoplus to production                                                                              14h
  You viewed Production, Preview, Development Environment Variable NEXT_PUBLIC_NGROK_URL in Project expoplus       14h
  You edited Production, Preview, Development Environment Variable NEXT_PUBLIC_NGROK_URL in Project expoplus       14h
```

3.11 Ngrok

For hosting our backend Flask application, we chose ngrok, an ingenious tool that provides a secure way to expose local servers behind NATs and firewalls to the public internet over secure tunnels. ngrok is particularly advantageous for development and testing phases because it does not require complex infrastructure setup or changes to the network configuration. It enables immediate, reliable, and secure public URLs for our Flask application, facilitating real-time sharing and demonstration of our backend services. Moreover, ngrok's introspection tools provide valuable insight into the HTTP traffic, allowing for easy debugging and efficient



```
ngrok                                                   (Ctrl+C to quit)

Take our ngrok in production survey! https://forms.gle/aXiBFWzEA36DudFn6

Session Status                online
Account                       destiny (Plan: Free)
Version                       3.8.0
Region                        Europe (eu)
Latency                       32ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://a25d-51-37-174-202.ngrok-free.app -> ht

Connections                   ttl     opn     rt1     rt5     p50     p90
                              828     0       0.00    0.03    0.06    1.13
```

4. Design

This section is a visual representation of the skeleton of how the project is run and how everything functions. We will speak in detail about each one of the components, and this section will give you an idea of our thought process behind the design/how we went about the making of the design. Diagrams will be used to help highlight how all the components intertwine with each other to help to give a better understanding.

We took insight from dcus home page colours

UI Design & Evaluation

When it came to designing our UI originally, we decided to follow the rules laid

out in Jakob Nielson’s book Usability Engineering. Below are these rules, followed
by an evaluation of how well our web app followed these initially chosen rules:

Visibility of system status. Users should always know where they are and what's going on. Following this rule, every page has titles of the features on the page, helping the User to recognise instantly what page they are on. We believe we have followed this rule perfectly and without exception.This rule was one that did not fully apply to our application, however,where it did was the language we used in the web app. With this in mind,
we made sure to keep all language used simple and easy for someone
knew to programming to understand

Control and freedom. Don't "trap" the user. Support clearly marked exit, undo, and redo functions. Don't force them into a long linear sequence of operations with no escape.
We ensured every page had a way of navigating back to where the user
came from or to move to a different page. The User is never trapped orforced to only move forward through pages in our web app.

Consistency and standards Use objects and phrases consistently. Follow
platform conventions. Here is a checklist of specific items to watch for.
We kept the colour scheme and font the same all across our pages,
allowing for visual consistency for the Users.

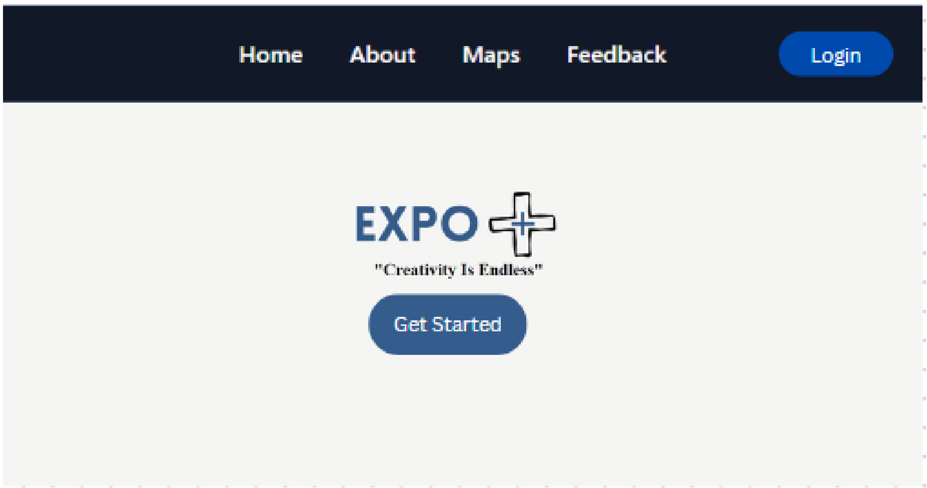
Flexibility and efficiency of use. Accelerators (unseen by novice users) can
speed up interaction for expert users. Allow users to customize frequent
actions whenever possible.

7. **Aesthetic and minimalist design.** Visibility of rarely needed information should be avoided. The more information that appears on the screen, the less
visible each unit of information becomes.
We kept the amount of content on each page to an absolute minimum, and
there is no rarely needed information. All information present is needed.
We have little to no visual clutter in our design.

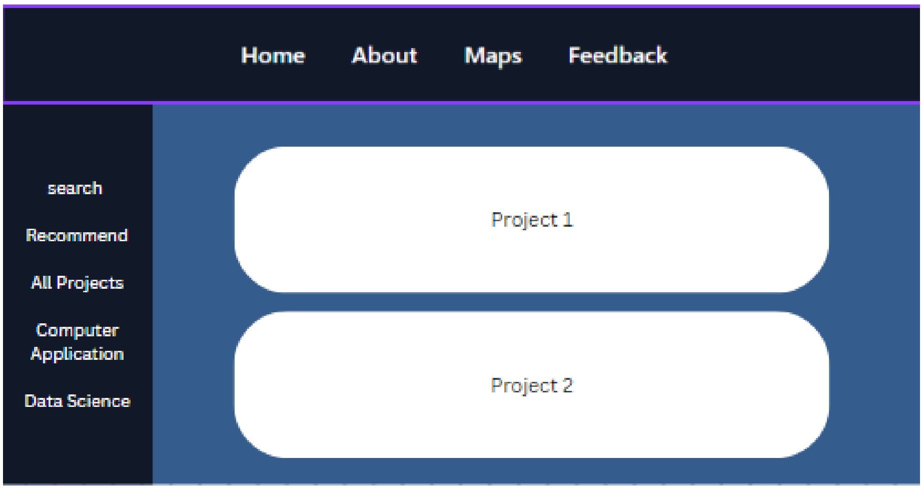
The reason

4.1 Wireframes

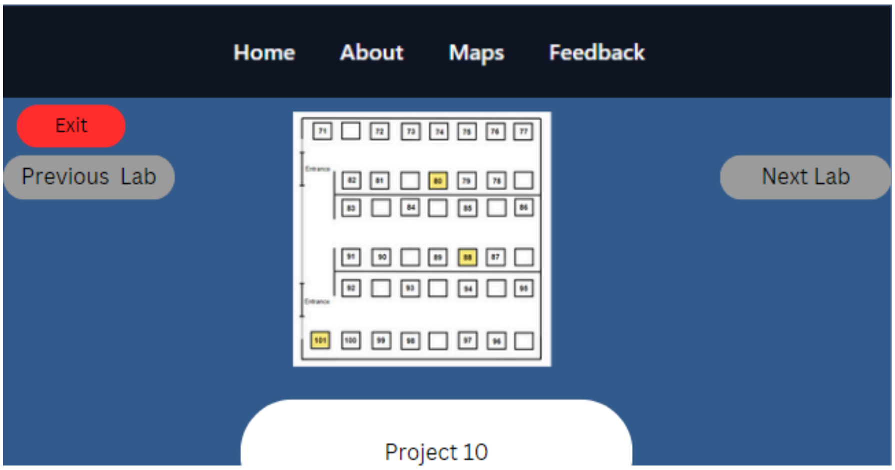
Index page



Home page

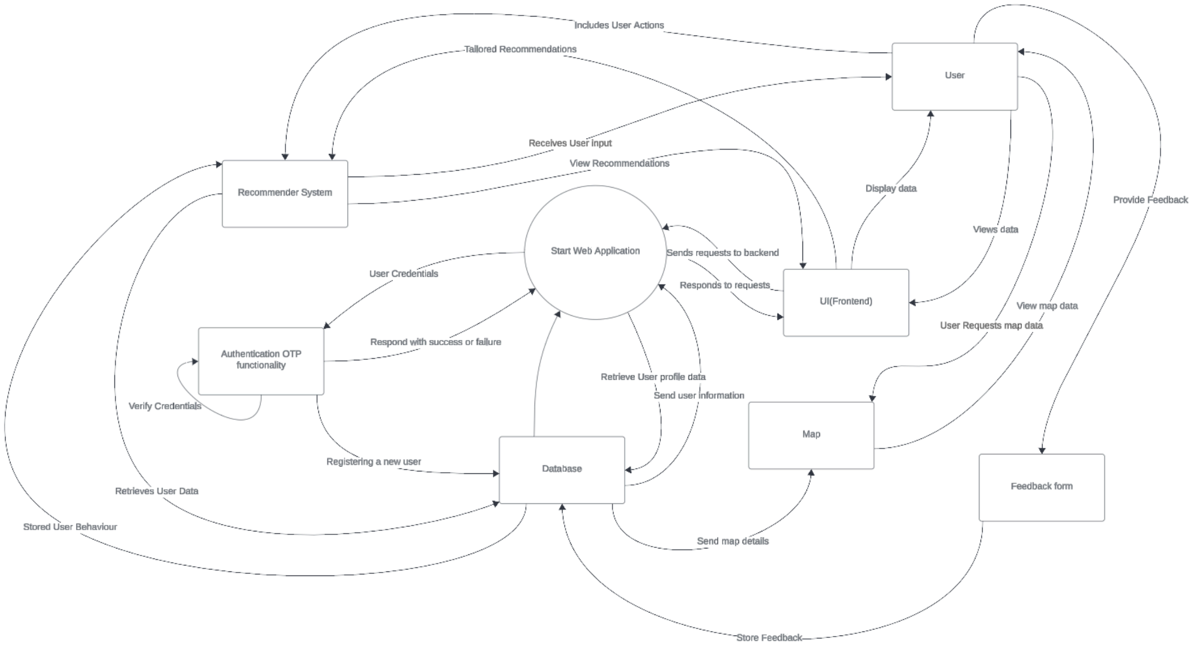


Individual Map page



4.3. Context Diagram

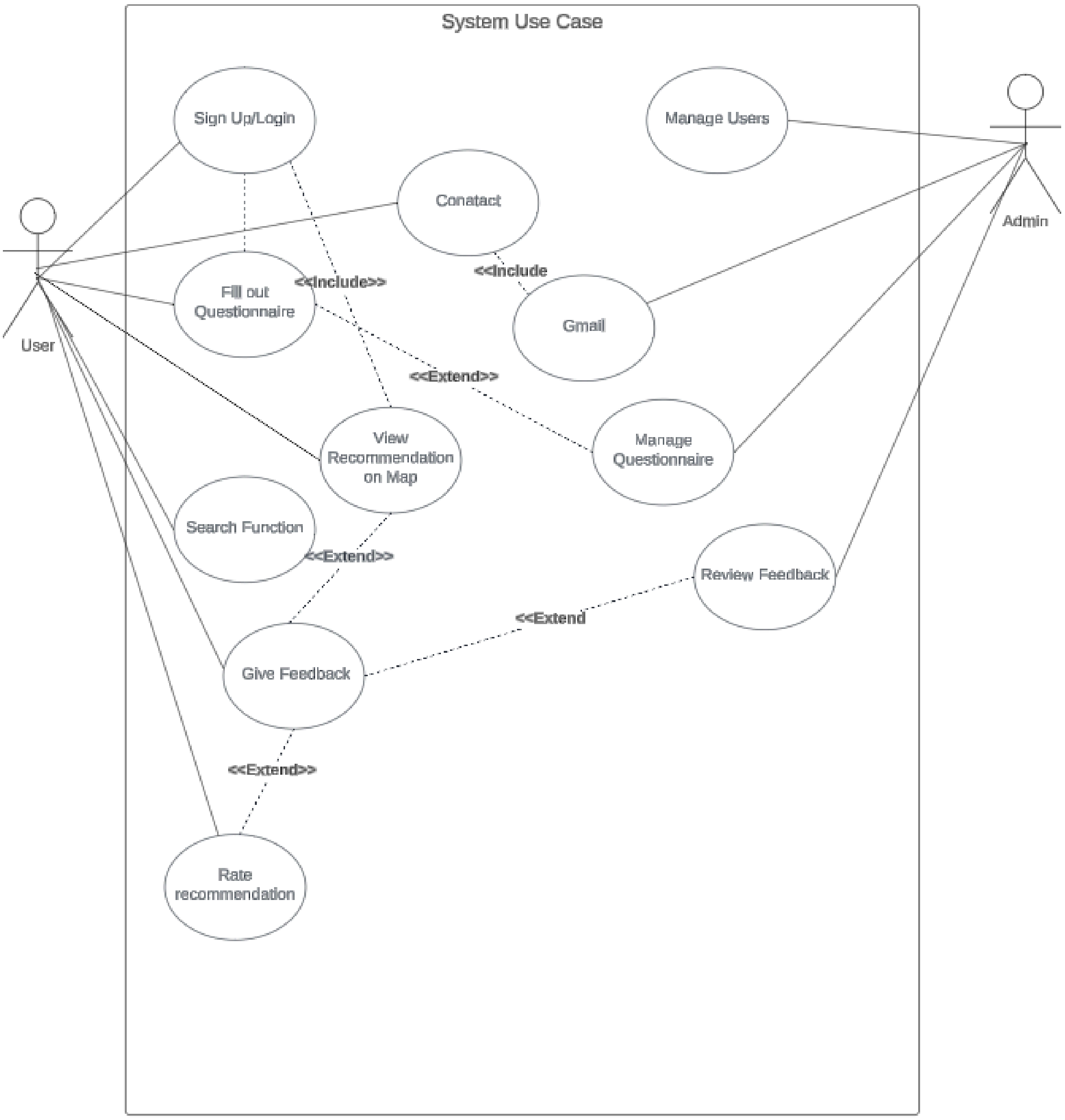
A context diagram is a simple graphical representation that shows the scope and boundaries of a system at a glance. It delineates how the system interacts with external entities like users, other systems, and external data sources. Unlike more detailed diagrams that show processes, a context diagram is high-level and focuses on understanding the system's environment and relationships with external agents.



4.4 Use Case Diagram

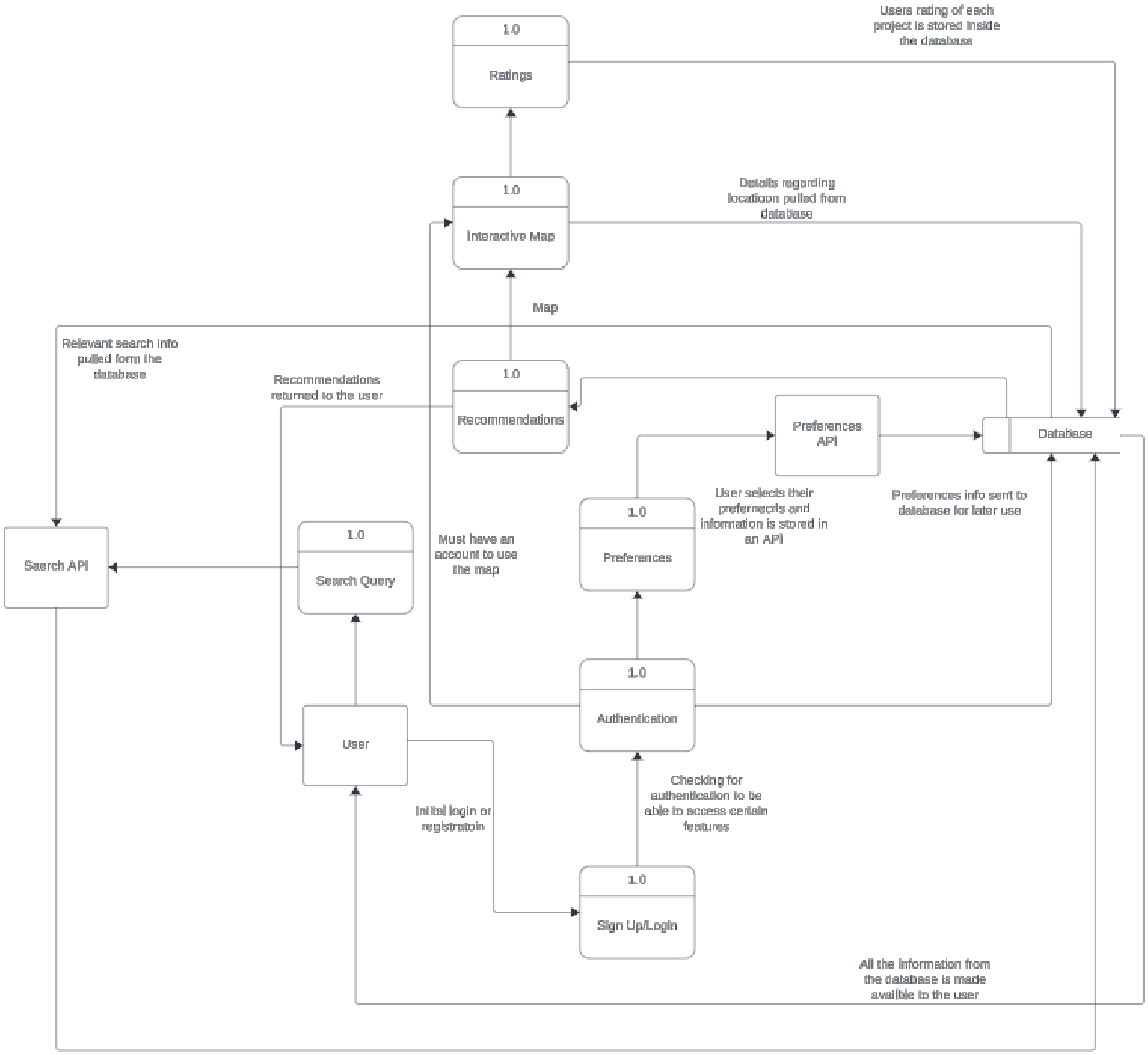
A use case diagram is a visual representation of the interactions between the end-users (known as actors) and a system, designed to capture the system’s functional requirements. It illustrates the various ways that users can interact with the system, highlighting the different functionalities available to different user roles and how they overlap or are related.

In the use case diagram for the Expo+ project, the various interactions that users and administrators can have with the system are mapped out.



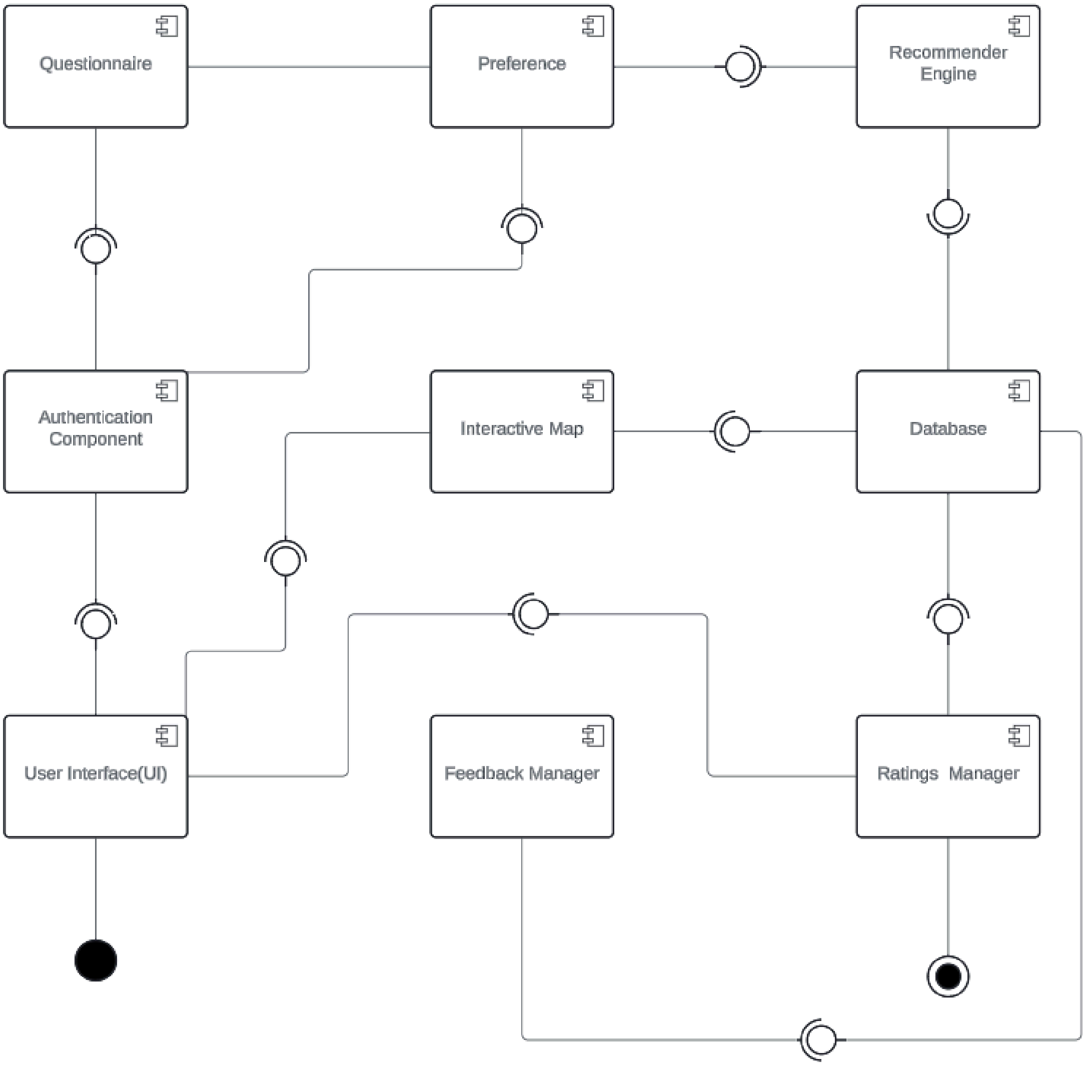
4.5 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation that depicts the flow of data within a system. It maps out how data enters a system, how it's processed, and how it leaves the system, usually in the context of information systems. It's a way of visualising how data moves between different components of a system, such as processes, data stores, and external entities, and how these components interact with each other.


1.0
Ratings
Users rating of each project is stored inside the database
1.0
Interactive Map
Details regarding locatioon pulled from database
Map
Relevant search info pulled form the database
Recommendations returned to the user
1.0
Recommendations
Preferences API
Database
User selects their prefernecds and information is stored in an API
Preferences info sent to database for later use
1.0
Search Query
Search API
Must have an account to use the map
1.0
Preferences
1.0
Authentication
User
Initial login or registratoin
Checking for authentication to be able to access certain features
1.0
Sign Up/Login
All the information from the database is made availble to the user

4.6 Component Diagram

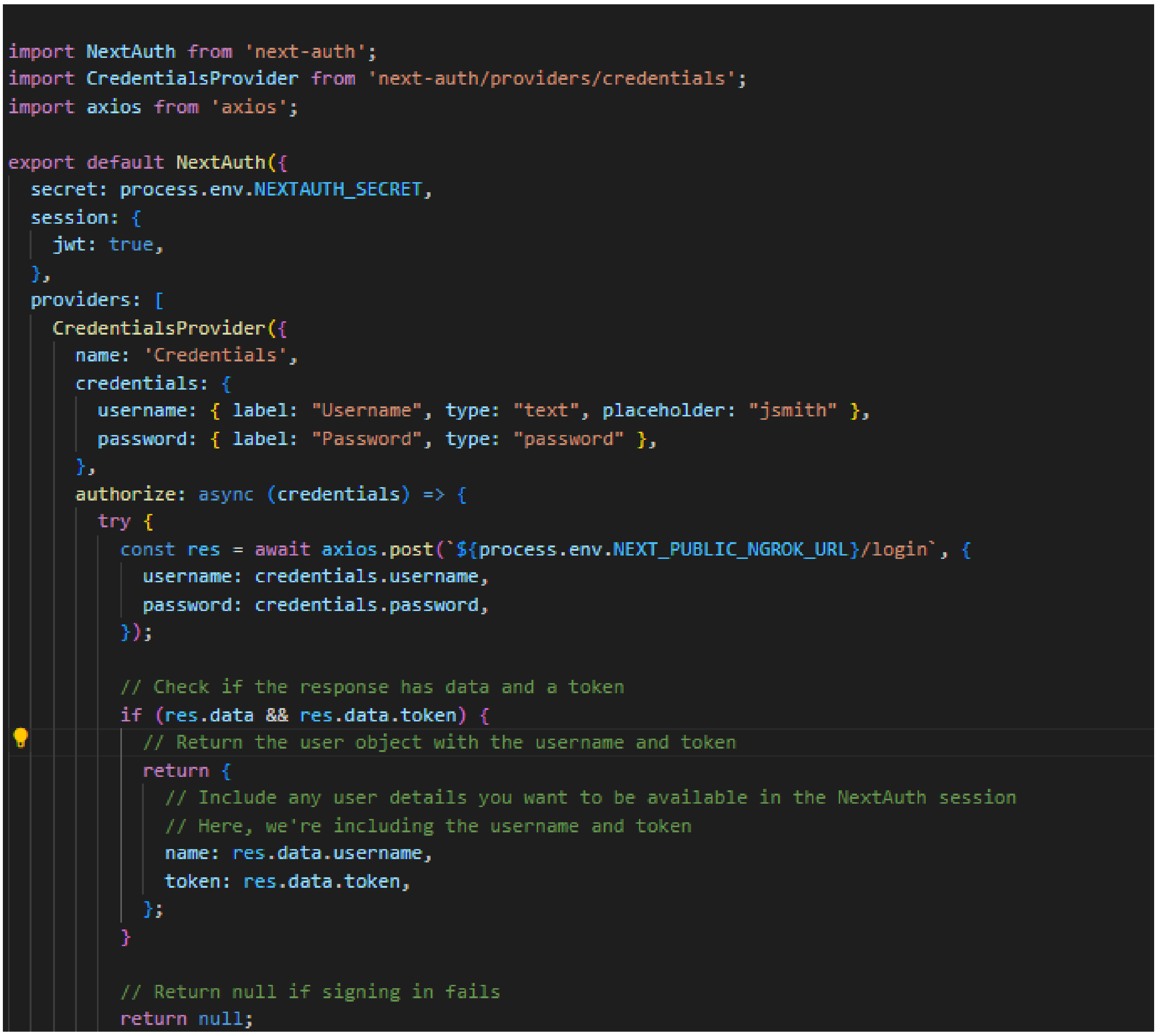
A component diagram, in the context of UML (Unified Modeling Language), is a structural diagram that shows how a software system is split up into components and illustrates the interconnections between these components. Components are modular parts of a system that encapsulate its contents and whose implementation is replaceable within its environment.



5. Implementation/ Sample Code

5.1 User Authentication

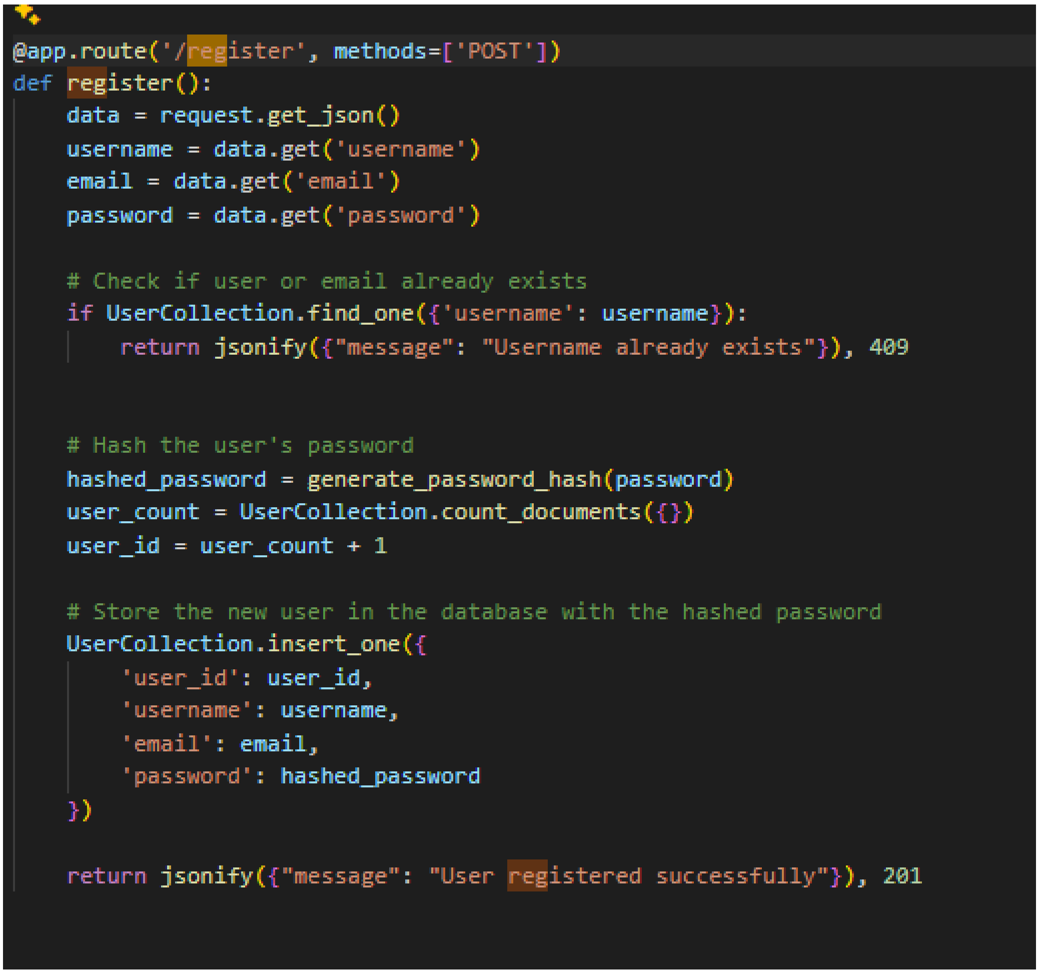
User authentication: within our application registering login and logout is a critical function it ensures account creation and access to user specific content and preferences, which maintain a personalisation within the platform



```
import NextAuth from 'next-auth';
import CredentialsProvider from 'next-auth/providers/credentials';
import axios from 'axios';

export default NextAuth({
  secret: process.env.NEXTAUTH_SECRET,
  session: {
    jwt: true,
  },
  providers: [
    CredentialsProvider({
      name: 'Credentials',
      credentials: {
        username: { label: "Username", type: "text", placeholder: "jsmith" },
        password: { label: "Password", type: "password" },
      },
      authorize: async (credentials) => {
        try {
          const res = await axios.post(`${process.env.NEXT_PUBLIC_NGROK_URL}/login`, {
            username: credentials.username,
            password: credentials.password,
          });

          // Check if the response has data and a token
          if (res.data && res.data.token) {
            // Return the user object with the username and token
            return {
              // Include any user details you want to be available in the NextAuth session
              // Here, we're including the username and token
              name: res.data.username,
              token: res.data.token,
            };
          }

          // Return null if signing in fails
          return null;
```

User Registration:

The registration route (/register) enables users to create accounts by providing a username, email address, and password through our front end UI which will then be fed to our backend route. To prevent duplication, the system verifies that the chosen username is unique. Passwords undergo secure storage using Werkzeug's generate_password_hash function for enhanced protection.



```
@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    username = data.get('username')
    email = data.get('email')
    password = data.get('password')

    # Check if user or email already exists
    if UserCollection.find_one({'username': username}):
        return jsonify({"message": "Username already exists"}), 409


    # Hash the user's password
    hashed_password = generate_password_hash(password)
    user_count = UserCollection.count_documents({})
    user_id = user_count + 1

    # Store the new user in the database with the hashed password
    UserCollection.insert_one({
        'user_id': user_id,
        'username': username,
        'email': email,
        'password': hashed_password
    })

    return jsonify({"message": "User registered successfully"}), 201
```

User Login:

Users are authenticated via the login route (/login) using their username and password. Successful authentication triggers the storage of the username within a session, facilitating personalised content and streamlined session management.

User Logout:

The logout route (/logout) terminates a user's session by clearing session data, effectively logging them out of the application.

Database integration

MongoDB is used to manage our user information once the user registers they will be provided with a unique user id that will be tagged to their activity throughout the database

Security Considerations:

- *Password Hashing*:
  Before storage, user passwords are hashed to safeguard against unauthorised access.
- *Session Management*:
  Flask sessions maintain user login state across requests. Appropriate Safeguards are in place to mitigate session hijacking risks.

5.2 Home Page/ Dashboard

Expo+ homepage (home.js) acts as a central hub, providing access to the side bar which has various links to pages such as Search.js, the individual module pages and recommended projects to visit. Before you can get access to the home page it will check if the user is authenticated. It then has useEffect that fetches all the projects that are stored in our database with the /allprojects route. We applied pagination to the fetch request so 20 projects can be shown per page.

Highlight function
Each project listed has a button which indicates “highlight on map” which communicates with our /api/highlight route in our back end. This route takes the session user's name and project id they wish to highlight. This will then be useful for us to know what projects they wish to visit later on the map after they finished browsing. This feature is added to all the pages which have projects displayed on them this feature was added for easy planning for the users

Visit on map function
Alongside each Project there is a visit button this allows the user to have quick access to view the project on the map

We created this functionality by using react routers to push the user to the correct map page based on the project id.

5.3 Components

Ground Floor Map/First Floor map

We developed a React component that handles SVG images representing the ground floor and the first floor of the McNulty building. Crafted using Inkscape, these images are designed as vector paths, allowing for dynamic manipulation. As part of the interactivity, when a user hovers over any lab on the map, the path colour transitions from blue to black, signalling that the lab is being highlighted. This hover effect is controlled by a useState hook, named HoverPaths, which we have successfully applied to each lab depicted within the maps.

Navbar/Footer/Sidebar

We included the navbar, sidebar and footer in the components folder as we would be calling them in multiple pages and to prevent us from having to write duplicate code we called each component into the pages we desired them to be in.

Starring functionality

In the stars.js file, our rating feature was established, allowing users to evaluate projects on a scale of one to five stars. We initiated the setup by defining default counts, colour schemes, and iconography for the stars. When a user selects a star rating for a highlighted project, the action triggers a request to the api/ratings endpoint of our backend. This request transmits the username, the project's unique identifier, and the chosen rating, which are then recorded in our database. Given its recurring use across different lab pages within our application, this star-rating functionality was encapsulated within a reusable React component.

5.4 Individual Module pages

These pages consist of computerapps.js, datascience.js, mechatronic engineering,each of these pages serves as a functional component in the application. They provide a specialised interface for viewing each module expo project. Here's an overview of its functionality:

State Management and Hooks

The component utilises the useState hook for managing the state of the project list, loading status, errors, and pagination. The useEffect hook is employed to trigger the fetching of project data from the backend when the component mounts or when pagination state changes.The useSession hook from next-auth/react is integrated to check the authentication status of the user, which is required for certain interactive features.

API Integration and Data Fetching

Data fetching is initiated in the useEffect hook, where an asynchronous function fetchProjects is defined and executed.
The function constructs an API request to the specified backend route, passing parameters such as the collection names and pagination details. It uses the fetch API to retrieve project data, which is then stored in the project's state variable.

Interactive Map and Project Highlighting

The highlightProject function is used to send a user's interaction to the backend, marking a project as highlighted on the map it's associated with.

It sends a POST request with the user's username and the project ID to the /api/highlight route.

User Authentication:
It leverages the useSession hook for session management, ensuring certain interactions, like highlighting projects on a map, are limited to authenticated users, thus incorporating an element of access control. Interactive Features: The highlightProject function, which calls the backend API to visually mark projects on an interactive map, enriches the user experience by offering visual cues for project locations.

5.5 Interactive map pages

The files named l101page.js, lg25page.js, l128page.js, and others are part of our application that display projects marked by users. Each file corresponds to a particular page containing an SVG image, which represents a map section, and these SVG files are conveniently located in the public directory of our Next.js application. With distinct paths mapped out for each project on these SVG maps, we utilise the SvgSeatHighlighter() function to visually distinguish the projects that a user has marked. Below the map, there is an enumerated list of these highlighted projects. For each listed project, users have the ability to remove the highlight or toggle the 'visited' status, which changes the project's highlight colour to red on the map, indicating visitation.

State Management:
The component uses React’s useState to track various pieces of data, including a record of visited projects (visitedProjects), loading state (isLoading), and errors. This stateful logic ensures the UI is always up to date with the latest user interactions and data fetch status.

Session and Authentication:
With useSession from NextAuth, the component manages user sessions. Authentication status is checked, and unauthenticated users are prompted to sign in.

Routing:
The useRouter hook from Next.js allows the component to programmatically navigate to different pages within the application, enhancing the user journey through the goToPreviousLab function and other similar navigational actions.

Data Fetching and API Integration:
UseEffect hooks make asynchronous calls to the backend API to retrieve project identifiers (projectIds) and detailed information (projectTitles) related to the specific lab floor, enabling dynamic and real-time updates to the SVG map and project list.

SVG Interaction:
The component uses an iframe to load an SVG representing the lab floor. It references this iframe using useRef for direct DOM manipulation.
A highlightProjects function is defined to colour code the SVG paths, indicating whether a project has been visited based on user interaction. This interactivity is further supported by toggling the visited state of projects with the toggleVisitedState function, allowing users to mark projects as visited and visually reflect this on the SVG map.

Persistent State:
Visited projects are stored in the local storage, ensuring that the state persists across browser sessions.

Error Handling:

Any errors from the fetch requests are captured and managed, updating the error state and presenting feedback to the user.

User Feedback:
Loading states are handled to provide a seamless and informative experience, displaying a loading indicator during data retrieval.

Project Management:
Projects can be removed from the SVG map view, and corresponding updates are made to the backend through the removeProject function, which sends a DELETE request to the API.

UI Components and Layout:
Shared components like NavBar and Footer are included to maintain consistency across the application.
Project titles and additional information are listed in a user-friendly format, allowing for quick scanning and interaction.

Custom Hooks and Event Handlers:
Custom logic handles the colour change of paths within the SVG and updates the UI in response to user actions such as marking projects as visited or navigating to different lab pages.

5.6 User Preferences

The Preferences 'preferences.js page in our React application is a crucial component designed to personalise the user experience by collecting their preferences in terms of programs, technologies, and project areas. Implemented as a functional component with state hooks and interactive UI elements, this page serves as a portal for users to tailor the content they will see on the platform.

Security & Validation

Upon accessing the Preferences page, the component first checks the user's session using the useSession hook from NextAuth. This is a security measure to ensure that only authenticated users can submit their preferences. If the user is not authenticated, the page prompts them to sign in, safeguarding the process and ensuring that preferences are linked to the correct user account.

Form creation

The heart of the Preferences component is the form where users select their preferred programs, technologies, and areas of interest using the react-select library, which offers a user-friendly dropdown selection interface. The selections are managed by individual state variables, allowing the component to track changes in real-time and maintain a responsive and dynamic interface. Upon form submission, the component executes an asynchronous API call to the backend, sending the selected preferences to be stored in the database. This enables the application to provide a customised experience in subsequent sessions, such as content based project recommendations. Upon pressing the submit preferences this will then fetch a post request to our backend storing the users preferences the user will then be redirected to the recommended page.

Data preservation

An essential feature of this component is its ability to remember a user's previously submitted preferences. When a user signs in, the application checks if preferences have already been set. If they have, the user is automatically redirected to the recommendations page, skipping the need to re-enter preferences and streamlining the user experience. This intelligent design choice not only saves time for returning users but also creates a seamless transition from authentication to personalised content.

5.7 Hybrid recommendation system

The details of this model is presented inside the RecommendationModel&Evaluation file located in the Testing Directory

The recommended functional component within our React application is dedicated to showcasing user-personalised project recommendations. Employing a mixture of useState and useEffect hooks, this component maintains a list of projects (projects),

Frontend Implementation:
Upon user authentication, The component initiates a fetch request in useEffect to the backend API endpoint /recommend with the current user's username. This request retrieves a set of projects, determined by the user's previous interactions and ratings, and sets them into the component state. The frontend then dynamically renders these projects, providing options for users to highlight these on an interactive map or visit detailed pages for more information. The Visit function uses the user's selection to navigate to specific project details pages, like /lg27page, depending on the project ID range. Additionally, the highlightProject function interacts with the backend to mark selected projects on the map, indicating user interest.

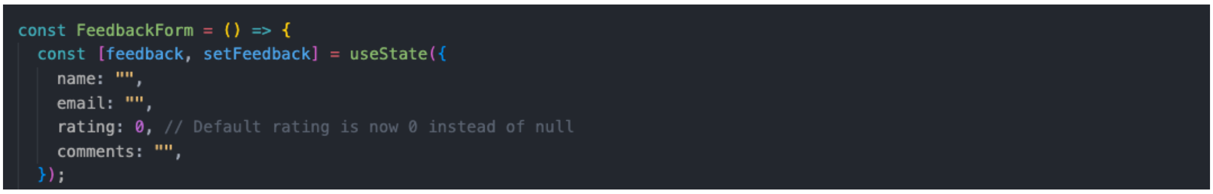
Backend Route Description:
The recommend_projects backend function handles the /recommend API route. It fetches the requesting user's data from the UserCollection and checks their rating history in the RatingsCollection. If a user has a rich history of ratings (5 or more) this helps us eliminate the cold start problem as users will be given project recommendations based on their preferences upon registration. The system employs a collaborative recommendation algorithm, otherwise, it falls back to a default recommendation set. The recommendation logic, whether collaborative or default, retrieves project IDs, fetches the corresponding project details from the AllProjects collection, and returns this structured data as JSON to the frontend. The backend effectively determines whether to use collaborative

filtering based on user activity, thereby providing a more personalised set of recommendations when possible. This intelligent decision-making process is crucial for enhancing user engagement and satisfaction with the application.

5.8 Feedback Form

In our web application, we've successfully integrated a feedback form to enhance our user engagement and obtain valuable insights. Let's discuss how we implemented this feature, highlighting key snippets of the code.

Firstly, we utilised React's functional component structure to manage the state and behaviour of our feedback form. With the useState hook, we defined an object feedback to store user input data, initialising fields for name, email, rating, and comments. The default rating is initially set to 0, an intentional choice to prompt users to actively select a rating:

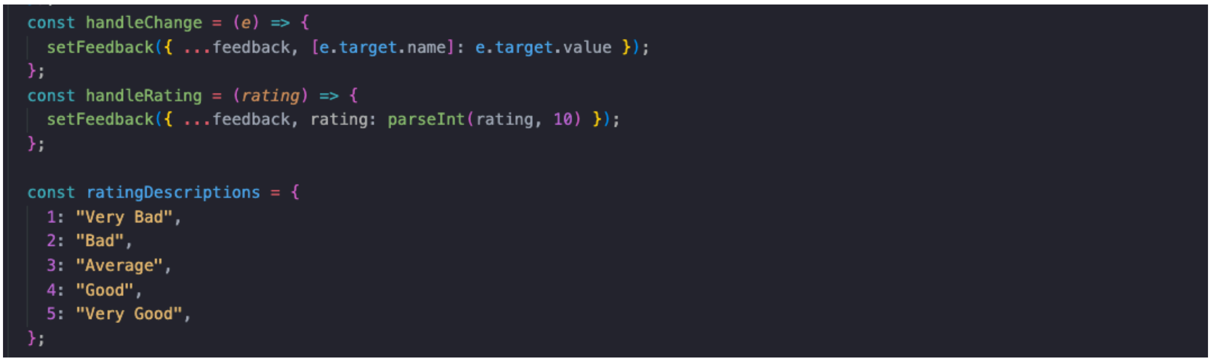


```
const FeedbackForm = () => {
  const [feedback, setFeedback] = useState({
    name: "",
    email: "",
    rating: 0, // Default rating is now 0 instead of null
    comments: "",
  });
```

We employed the handleChange method to update the state whenever a user types into the input fields. This method leverages the flexibility of ES6 computed property names to handle multiple inputs with ease.

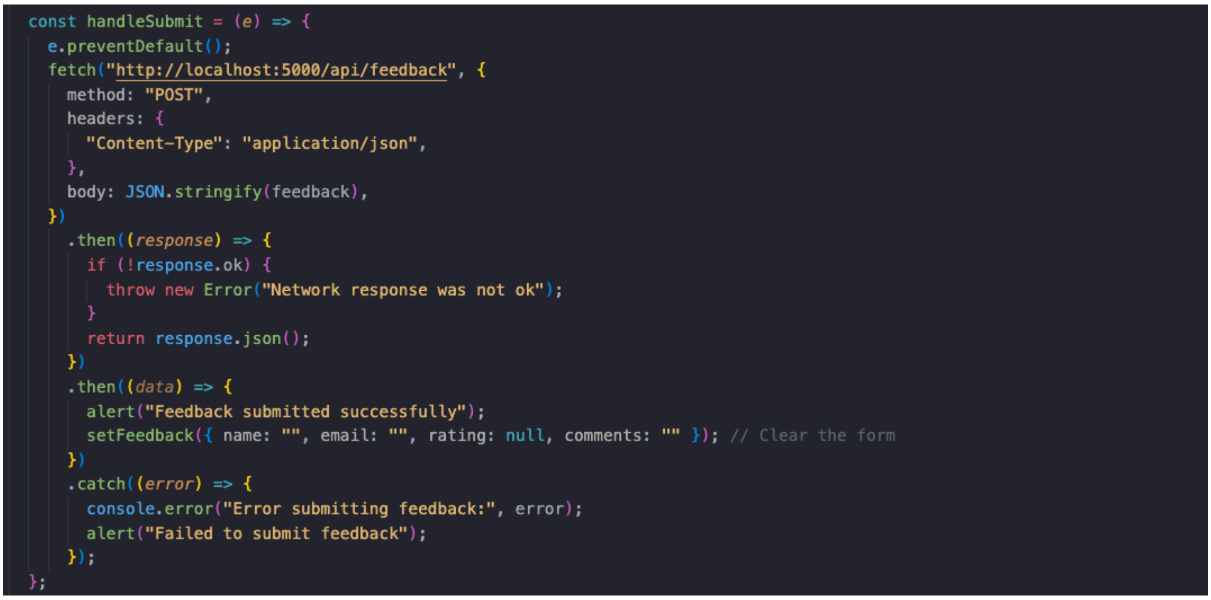
In the case of ratings, we crafted a dedicated handleRating function to parse the rating value to an integer before updating our state. This ensures that the data we collect is of the correct type.

We introduced ratingDescriptions as a simple object to map rating values to descriptive text, enhancing user experience by providing immediate context for the chosen rating



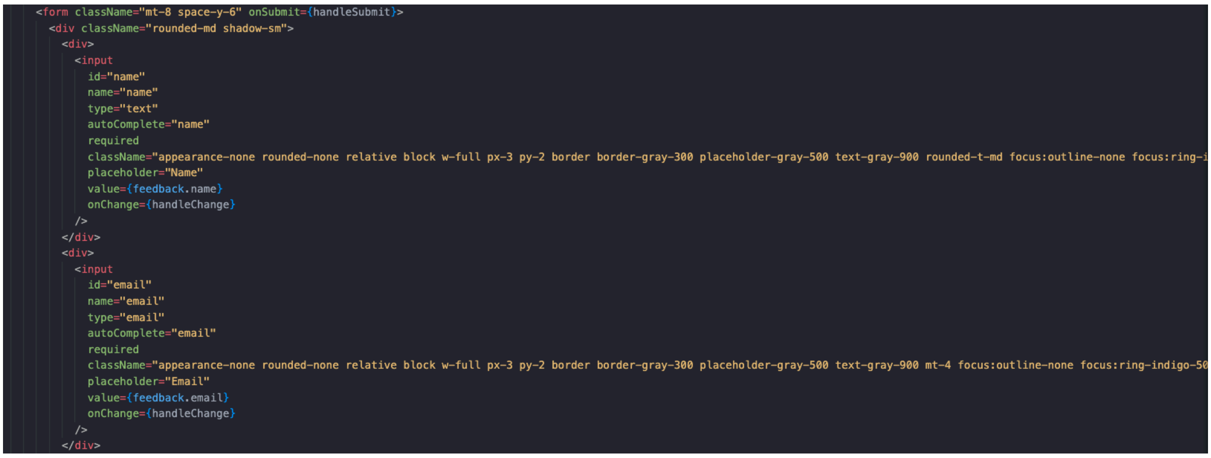
```
const handleChange = (e) => {
  setFeedback({ ...feedback, [e.target.name]: e.target.value });
};
const handleRating = (rating) => {
  setFeedback({ ...feedback, rating: parseInt(rating, 10) });
};

const ratingDescriptions = {
  1: "Very Bad",
  2: "Bad",
  3: "Average",
  4: "Good",
  5: "Very Good",
};
```

The handleSubmit function is pivotal, executing a series of actions upon form submission. It prevents the default form submission, makes a POST request to our server endpoint with the collected feedback data, and handles the promise with a success or error alert to the user.



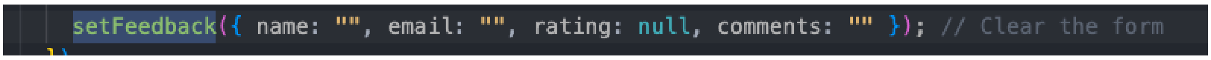
```
const handleSubmit = (e) => {
  e.preventDefault();
  fetch("http://localhost:5000/api/feedback", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(feedback),
  })
    .then((response) => {
      if (!response.ok) {
        throw new Error("Network response was not ok");
      }
      return response.json();
    })
    .then((data) => {
      alert("Feedback submitted successfully");
      setFeedback({ name: "", email: "", rating: null, comments: "" }); // Clear the form
    })
    .catch((error) => {
      console.error("Error submitting feedback:", error);
      alert("Failed to submit feedback");
    });
};
```

For the form's UI, we've put the feedback form within a div that centres it on the page, alongside a NavBar component for easy navigation. The form itself is constructed with input fields for name and email, a dynamic rating button group, and a text area for comments. Each input field is controlled, with its value tied to our feedback state and onChange handlers to ensure state updates with user input.



```
<form className="mt-8 space-y-6" onSubmit={handleSubmit}>
  <div className="rounded-md shadow-sm">
    <div>
      <input
        id="name"
        name="name"
        type="text"
        autoComplete="name"
        required
        className="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-t-md focus:outline-none focus:ring-i
        placeholder="Name"
        value={feedback.name}
        onChange={handleChange}
      />
    </div>
    <div>
      <input
        id="email"
        name="email"
        type="email"
        autoComplete="email"
        required
        className="appearance-none rounded-none relative block w-full px-3 py-2 border border-gray-300 placeholder-gray-500 text-gray-900 mt-4 focus:outline-none focus:ring-indigo-50
        placeholder="Email"
        value={feedback.email}
        onChange={handleChange}
      />
    </div>
```

Finally, upon a successful submission, we clear the form fields by resetting the feedback state, readying the form for another input cycle.



```
  setFeedback({ name: "", email: "", rating: null, comments: "" }); // Clear the form
```

Through this implementation, we've created a user-friendly and responsive feedback form that not only gathers user insights but also maintains a clean and manageable state, thereby exemplifying our commitment to a seamless user experience.

Recover password

We implemented a recover password section that uses OTP and sends the pin to their email.

6.Problems and Solutions

6.1 Problem:

Finding a framework that best suits our application style

Solution: We started off by using react.js as our front end framework but after extensive research we chose to switch to next.js Next.js provides built-in support for server side rendering which can significantly improve performance and Search Engine Optimization of our application. This means that our pages can be pre-rendered on the server and served as fully-formed HTML to the client, rather than relying on client-side rendering alone. Next allows us to create API routes alongside our regular pages, making it easy to build full-stack applications with serverless functions. This can simplify your codebase and improve scalability.

6.2 Problem:

Manipulating the paths for each project on the map.

Solution: Our class maps were initially embedded within an iframe, crafted as SVG images with distinct paths linked to project IDs. Consequently, we encountered limitations in manipulating the paths within the SVG image, as Next.js lacked access to them. Subsequently, we opted to encapsulate the SVG images within a React component and integrate that component into the respective map pages. This approach grants our Next.js application access to the paths, enabling seamless manipulation

6.3 Problem:

Deploying our front end on Vercel

Solution: We encountered an issue during the deployment of our Next.js app on Vercel, wherein certain functionalities were restricted. Notably, users were unable to sign in or access the home page. After thorough investigation, we discovered that we needed to provide Vercel with our NextAuth environment variables. Additionally, deploying our backend on a server was necessary to facilitate communication between Vercel and our backend services

6.4 Problem:

Uploading our backend on a server was difficult. We came across many issues where it wouldn't be published.

Solution: We initially intended to deploy our backend on Heroku, but encountered issues where it did not recognize our environment settings. As an alternative solution, we opted to deploy our backend on Ngrok, a tool for creating secure tunnels to localhost, enabling temporary exposure of our local development server to the internet. However, we plan to transition back to Heroku in the future once we have dedicated time to debug and resolve the issues.

6.5 Problem:

The passwords are exposed when submitted to login and sent via the API to the server where a man in the middle can steal the user’s password.

Solution:

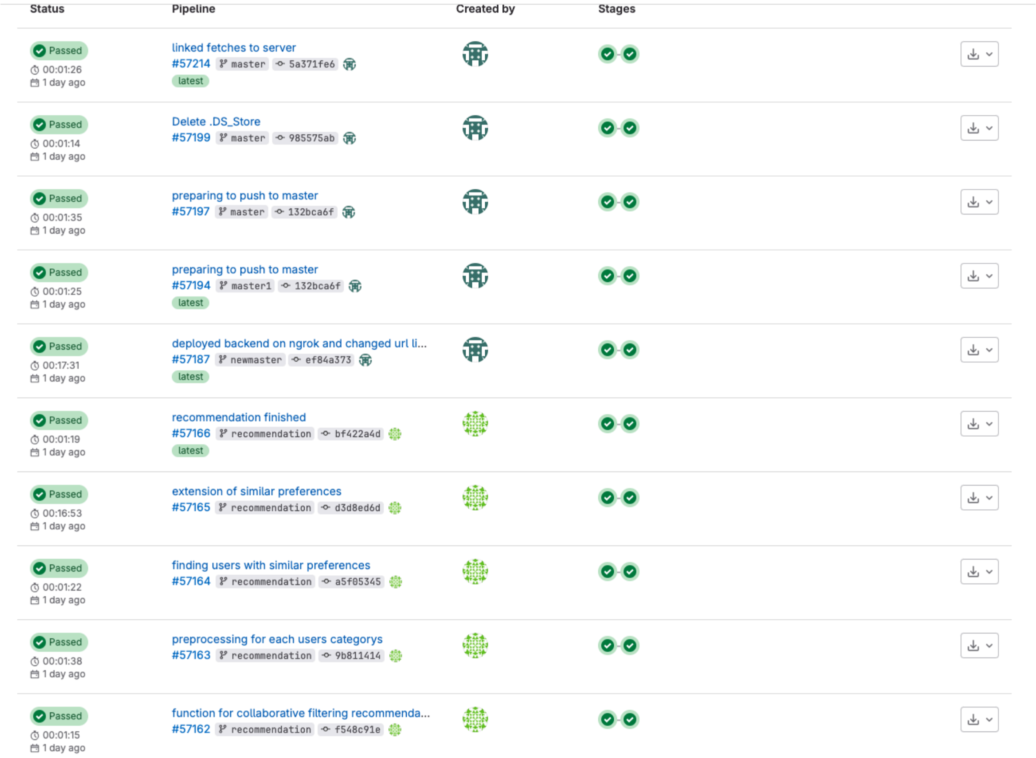
Symmetric encryption was employed to securely transmit the password from the frontend to the backend via API by encrypting it using a secret key, and subsequently, the password is then hashed into the database with added salt rounds.

7. Testing

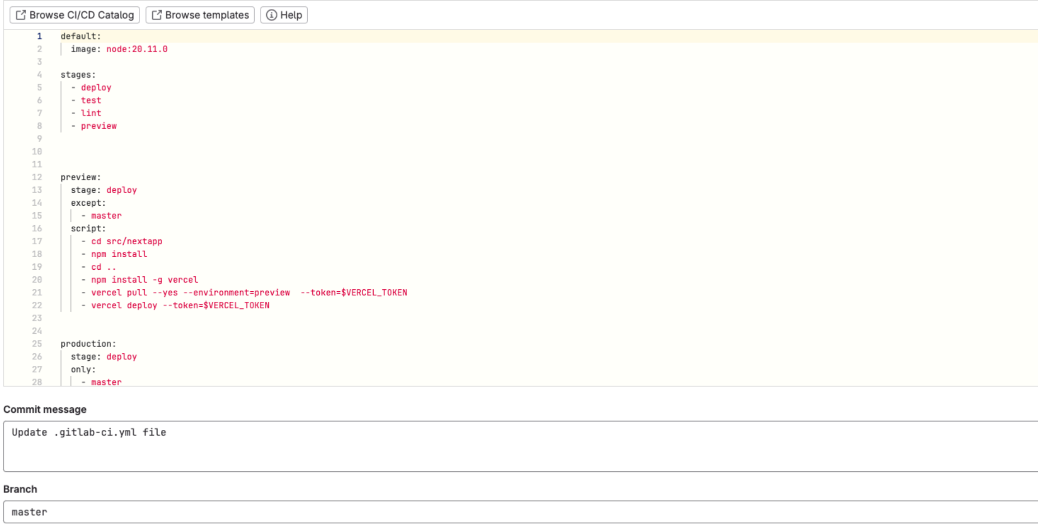
Most of our testing is robust so we created a testing folder to store them please take a look at that folder to see the grounds of our other tests

7.1 Regression testing

In our project, we incorporated GitLab’s robust CI/CD pipelines to automate our regression tests, which are essential for maintaining code quality and ensuring that new changes do not adversely affect existing functionalities. This continuous integration and deployment pipeline allows us to run tests automatically with each commit, providing immediate feedback on the impact of code changes. The pipeline's configuration is designed to execute a suite of regression tests, enabling us to identify and address any issues early in the development cycle. This automation not only streamlines the development process but also reinforces the reliability of our application by ensuring that all features remain stable and functional after each integration.



Status	Pipeline	Created by	Stages
Passed 00:01:26 1 day ago	linked fetches to server #57214 master 5a371fe6 latest		✓ ✓
Passed 00:01:14 1 day ago	Delete .DS_Store #57199 master 985575ab		✓ ✓
Passed 00:01:35 1 day ago	preparing to push to master #57197 master 132bca6f		✓ ✓
Passed 00:01:25 1 day ago	preparing to push to master #57194 master1 132bca6f latest		✓ ✓
Passed 00:17:31 1 day ago	deployed backend on ngrok and changed url li... #57187 newmaster ef84a373 latest		✓ ✓
Passed 00:01:19 1 day ago	recommendation finished #57166 recommendation bf422a4d latest		✓ ✓
Passed 00:16:53 1 day ago	extension of similar preferences #57165 recommendation d3d8ed6d		✓ ✓
Passed 00:01:22 1 day ago	finding users with similar preferences #57164 recommendation a5f05345		✓ ✓
Passed 00:01:38 1 day ago	preprocessing for each users categorys #57163 recommendation 9b811414		✓ ✓
Passed 00:01:15 1 day ago	function for collaborative filtering recommenda... #57162 recommendation f548c91e		✓ ✓



```yaml
default:
  image: node:20.11.0

stages:
  - deploy
  - test
  - lint
  - preview



preview:
  stage: deploy
  except:
    - master
  script:
    - cd src/nextapp
    - npm install
    - cd ..
    - npm install -g vercel
    - vercel pull --yes --environment=preview  --token=$VERCEL_TOKEN
    - vercel deploy --token=$VERCEL_TOKEN


production:
  stage: deploy
  only:
    - master
```

Commit message

Update .gitlab-ci.yml file

Branch

master

In our project, we integrated the use of branches as a fundamental part of our workflow, adhering to best practices in software development to ensure that each feature is developed, tested, and reviewed thoroughly before it is integrated into the main codebase. This branching strategy, typically known as the Feature Branch Workflow, plays a crucial role in our project management and version control systems.

8. Future work and Improvements

If given more time in the future to enhance the Expo+ application, we could incorporate several new technologies to elevate its usage and enrich the user experience. One such addition could be the integration of augmented reality (AR) features, allowing users to visualise virtual guides and exhibits overlaid onto real-world environments through their smartphone cameras. This immersive AR experience could provide visitors with interactive tours and detailed information about various exhibits, enhancing their engagement and understanding. Additionally, implementing location-based services (LBS) using technologies like beacon technology or GPS could enable personalised recommendations and navigation assistance tailored to each user's preferences and location within the expo venue. Furthermore, integrating voice recognition technology could offer users hands-free interaction, allowing them to access information and navigate the app using voice commands, thereby improving accessibility and convenience. These enhancements would not only make the Expo+ application more innovative and unique but also contribute to a more enriching and enjoyable experience for expo attendees.

With more time dedicated to design, we could make the Expo+ app more visually appealing and easier to use. We'd start by talking to users to understand what they like and what they find confusing. Then, we'd use that feedback to make the app's layout simpler and more intuitive, so people can find what they need quickly. Another thing we'd do is we'd make sure the app is easy to use for everyone, including people with disabilities, by using colours and text that are easy to see, and by adding options for people who can't use the app in the usual way.