



Islington college
(इस्लिङ्टन कलेज)

Full Stack development
MERN Stack Project

Submission: Final Submission

Summer Enrichment Project 2025

Student Name: Ajoob Sagar Kansakar

London Met ID: 23056153

College ID: np01cp4s240080@islingtoncollege.edu.np

Assignment Due Date: Friday, August 22, 2025

Assignment Submission Date: Friday, August 22, 2025

Submitted To: Abhishek Raj Jaiswal

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of contents:

1. Introduction:	1
2. Technology implemented:	2
2.1 Backend Technologies:	2
2.2 Frontend Technologies:	3
3. Page features:.....	4
3.1 Registration Page:.....	4
3.2 Login Page:.....	6
3.3 Homepage:.....	7
3.4 Profile page:	8
4. Conclusion:	9

Table of Figures:

Figure 1: Register Page UI.....	4
Figure 2: Login Page UI	6
Figure 3: Homepage UI	7
Figure 4: Profile page UI	8

1. Introduction:

This report outlines the features of a full-stack web application designed as a simplified replica of key LinkedIn features. The primary purpose of the application is to allow users to create a professional profile, log in, view other professionals, and edit their own information.

The project is built using MERN (MongoDB, Express.js, React, Node.js) stack with a clear separation between the backend server logic and the frontend user interface.

2. Technology implemented:

The application leverages a combination of modern programming languages, frameworks, and libraries to deliver a robust and interactive user experience.

2.1 Backend Technologies:

1. JavaScript: The primary programming language used for the backend logic. Its asynchronous nature makes it highly suitable for building efficient web servers
2. Node.js: A JavaScript runtime environment that allows JavaScript code to be executed on the server. It serves as the foundation of the backend.
3. Express.js: A minimal and flexible Node.js web application framework. In this project, it is used to create the RESTful API, define routes, and handle HTTP requests and responses from the frontend.
4. MongoDB: A NoSQL, document-oriented database used to store application data. Its flexible, JSON-like document structure is ideal for storing user profiles with varying information.
5. Mongoose: An Object Data Modeling (ODM) library for MongoDB and Node.js. It simplifies interaction with the database by providing a straightforward, schema-based solution to model application data and enforce validation.
6. JSON Web Tokens (JWT): A standard used to create access tokens for an application. Upon login, a secure JWT is generated and sent to the user, who then includes it in future requests to access protected routes.
7. Bcrypt.js: A library used for securing user passwords. It hashes passwords using a strong cryptographic algorithm before they are stored in the database, ensuring that even if the database is compromised, the actual passwords are not exposed.

2.2 Frontend Technologies:

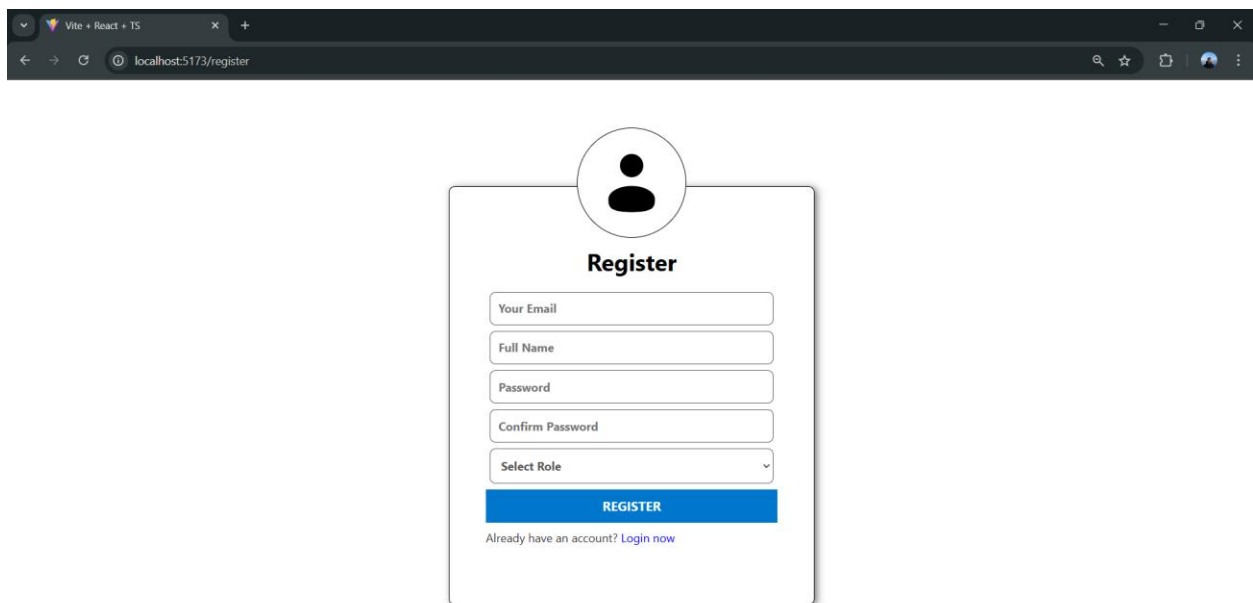
1. TypeScript: A superset of JavaScript that adds static typing. It is used to write the Frontend code, helping to catch errors early in development and improve the overall code quality and maintainability.
2. React: JavaScript library for building user interfaces. The entire frontend is built as a single page application (SPA) using React's component-based architecture, which allows for the creation of reusable and interactive UI element.
3. Vite: A modern frontend build tool that provides an extremely fast development server and bundles the code for production.
4. CSS3: Used for styling the application. It controls the visual presentation, layout, and design of all the pages to create a clean, professional, and responsive user interface.
5. Axios: A promise-based HTTP client used to make requests from the React frontend to backend API. It simplifies the process of sending and receiving data for operations like login, registration, and profile page updates.

3. Page features:

The application is divided into four main pages, each with a distinct set of features designed to provide a seamless user journey.

3.1 Registration Page:

The registration page allows new user to create an account.



The image shows a web browser window with the address bar displaying 'localhost:5173/register'. The page content is a registration form titled 'Register' with a user icon above the title. The form contains five input fields: 'Your Email', 'Full Name', 'Password', 'Confirm Password', and 'Select Role' (a dropdown menu). Below these fields is a blue 'REGISTER' button. At the bottom of the form, there is a link that says 'Already have an account? Login now'.

Figure 1: Register Page UI

The screenshot shows a web browser window with the address bar displaying 'localhost:5173/register'. The page features a registration form with a circular profile icon placeholder at the top. Below the icon is the title 'Register'. The form contains five input fields: an email field with 'ajob.kansakar@gmail.com', a username field with 'Ajob Sagar Kansakar', two password fields (both masked with '*****'), and a role dropdown menu currently showing 'Developer'. A blue 'REGISTER' button is positioned below the dropdown. At the bottom of the form, there is a link that says 'Already have an account? Login now'.

1. User input form: collects essential user information: Email, Username, Password, and Role.
2. Dropdown for Role Section: The “Role” field is implemented as a dropdown menu rather than a text field, ensuring data consistency for roles like “Designer”, “Developer” etc...
3. Password Confirmation: Includes “Password confirmation” field to ensure the user types their intended password correctly.
4. Client-Side Validation:
 - Checks if the password and confirm password field match.
 - Ensures that a role is selected from the dropdown menu.
 - Displays clear, user-friendly error message directly on the form If validation fails.
5. Secure Backend Communication: Sends the registration data to the backend, where the password is securely hashed with bcrypt before being saved to the database.

3.2 Login Page:

The Login page provides a secure way for exiting users to access the application.

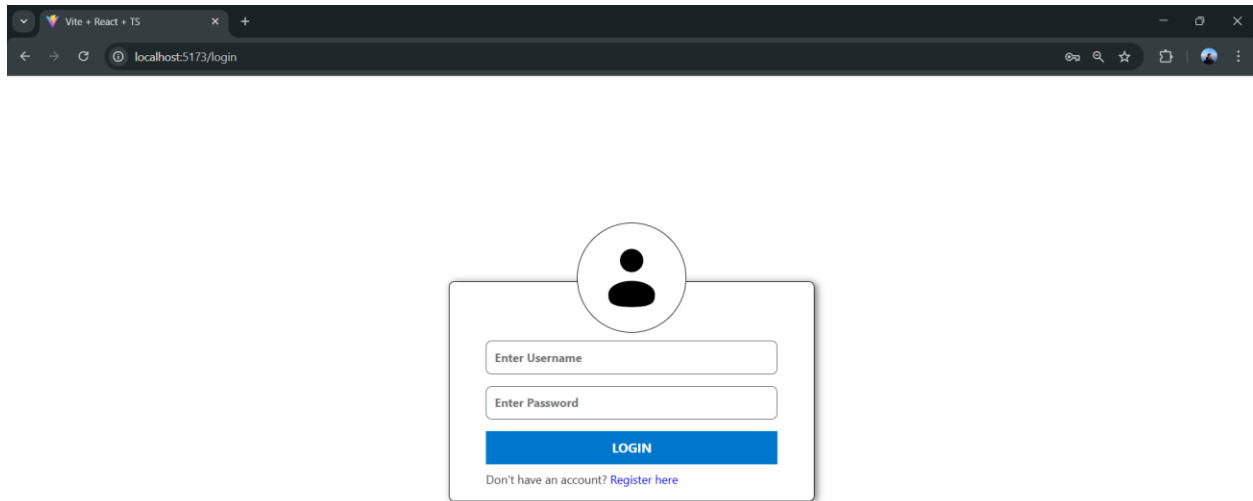


Figure 2: Login Page UI

1. **Real-time Inline Validation:** instantly checks if the input fields are empty and displays an error message as the user interacts with the form.
2. **Secure Authentication:** On submission, it sends the credentials to the backend. The server then verifies the username, retrieves the hashed password from the database, and uses bcrypt to compare it with the submitted password.
3. **Session Management:** Upon a successful login, the backend generates a JWT. This token and the user's data are stored in the browser's LocalStorage to keep the user logged in for subsequent requests.
4. **Error Handling:** If login fails, a clear error message from the server is displayed on the form.

3.3 Homepage:

The central hub of the application, accessible only after a user has logged in.

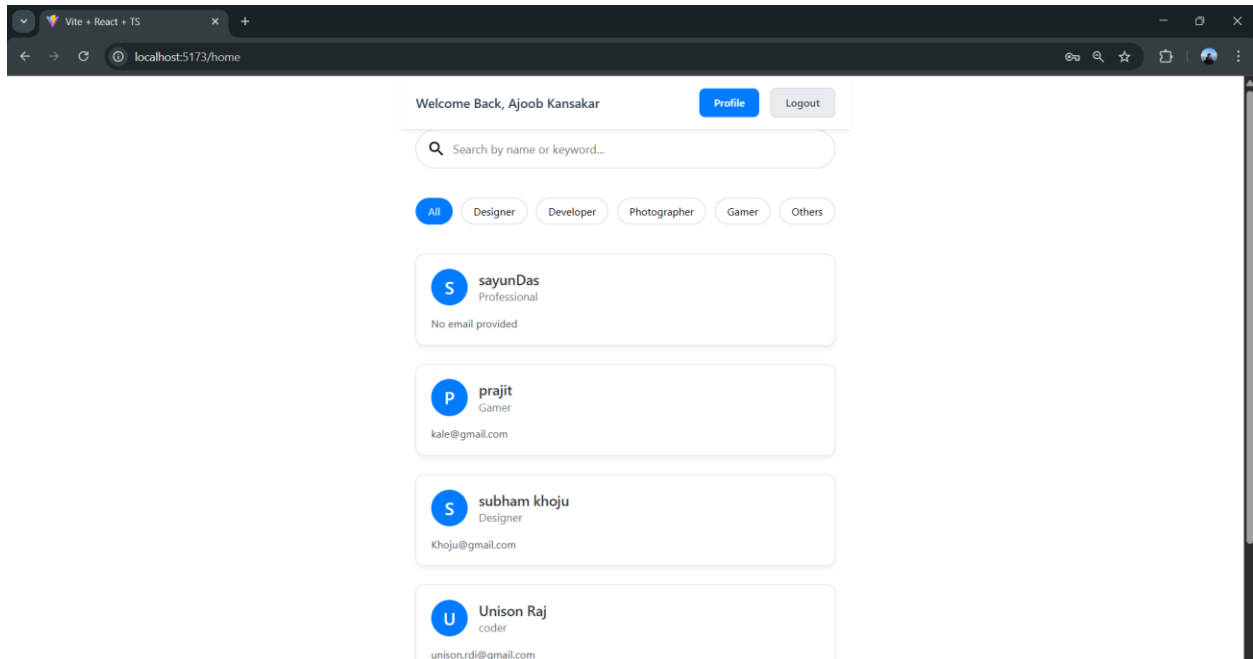


Figure 3: Homepage UI

1. User Authentication: This page is protected route so unauthenticated users are automatically redirected to the login page.
2. Personalized Welcome: Displays a Welcome message that includes the logged-in user's name.
3. Search Bar: A search bar that allows users to search for other professionals by their name or their roles.
4. Role-Based Filtering: A set of clickable tags that filter the displayed list of users based on their professional roles.
5. User Card Display: Each card shows the professional's name, role, and email, with a placeholder avatar generated from their initials.
6. Navigation: Contains clear buttons to navigate to the profile page and to logout.
7. Logout functionality: The logout button securely ends the user's session by clearing the token and user's data from localStorage and redirecting them to the login page.

3.4 Profile page:

This page allows users to view and update their own profile information.

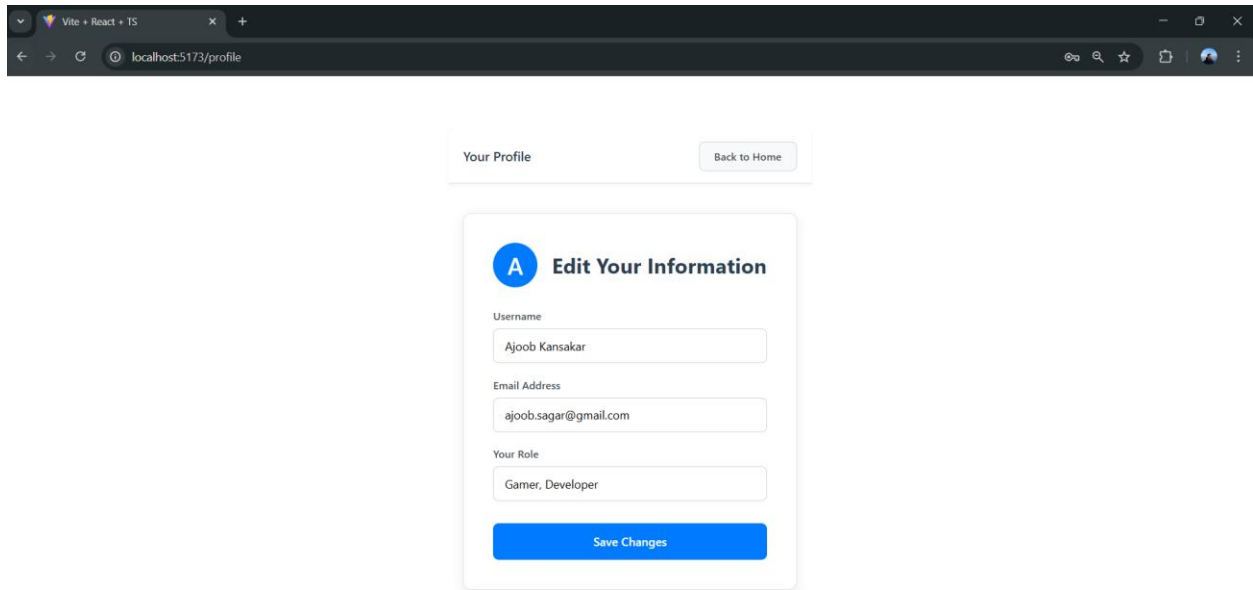


Figure 4: Profile page UI

1. Protected Route: Similar to Homepage, this page is only accessible to logged-in users.
2. Pre-filled form: The form fields for username, email address, and role directly in the form fields.
3. Information Editing: Users can modify their username, email address, and role directly in the form fields.
4. Secure Update Mechanism: When the user clicks “Saves changes” the updated data is sent via a secure PUT request to a protected backend endpoint. The backend uses the user’s JWT to verify their identity and ensure they can only update their own profile.
5. User Feedback: Displays success message or error messages directly on the page after an update attempt.

4. Conclusion:

This project successfully achieved its goal of developing a functional, full-stack Professional Networking Platform. By leveraging a modern technology stack including React, Node.js, and MongoDB, the application provides a complete user journey, from secure registration and login to real-time professional searching and profile management.

The development process was an invaluable practical exercise in building a modern web application, reinforcing key concepts such as RESTful API design, secure user authentication with JWT and password hashing, and component-based frontend development with React. The clear separation between the backend and frontend created a scalable and maintainable architecture.

While the current platform delivers on its core features, there is significant potential for future expansion. Future enhancements could include a user connection system, a real-time messaging feature, and the ability to upload profile pictures.

Overall, this project stands as a comprehensive demonstration of modern web development principles, successfully integrating a secure backend with a responsive and user-friendly frontend to create a complete and functional application.