INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

Course Title : Tools and Technologies for Internet Programming (LAB)

Course Code: CSE-3532

Tools and Technologies for Internet Programming Lab
Project Report on Job Seeker Hub using MERN

Submission Date: 09/07/2025

Submitted By:	**Submitted To:**
Nusaiba Ibnath (C223282)	Sara Karim
Semester - 5th (5DF)	Adjunct Faculty
	Computer Science and Engineering (IIUC)

Abstract

This project, titled "Job Seeker Hub", is a job-seeking web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The platform provides a user-friendly interface where job seekers can search and apply for jobs, while employers can post job listings and manage applications. The app also incorporates secure user authentication using JWT to ensure data privacy. Key features include job listing browsing, detailed job descriptions, job filtering by location and type, and a smooth application process. The frontend of the application is built using React and Vite for fast development and responsiveness, with Tailwind CSS used to create a fully responsive design that adapts across devices. The backend is managed using Node.js and Express.js, while MongoDB stores job listings and user data. The project aims to bridge the gap between job seekers and recruiters, offering a streamlined, scalable, and easy-to-use platform to facilitate the job application process.

Introduction

In today's rapidly evolving job market, the need for a digital platform that effectively connects job seekers with employers has become more essential than ever. Traditional job-hunting methods are gradually being replaced by efficient, online solutions that allow real-time interaction, seamless application processes, and broader reach. In response to this growing demand, this project "Job Seeker Hub" was developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) to provide a robust, user-friendly, and scalable job-seeking web application. The project aims to create a platform where users can register as job seekers or employers. Job seekers can search for jobs, view job details, and apply, while employers can post job listings and manage received applications. Security is ensured using JWT-based authentication, and the entire interface is built with React and Tailwind CSS, offering a smooth, responsive experience across all devices. This report presents the development process, objectives, architecture, technologies used, system features, and challenges encountered while building the application. The goal is to demonstrate practical full-stack development skills while delivering a real-world solution that bridges the gap between recruiters and job applicants in a modern, intuitive way.

Objectives

The primary objective of this project is to build a fully functional job-seeking platform using the MERN stack that connects job seekers and employers efficiently. The specific objectives of the Job Seeker Hub project are:

- **Develop a User-Friendly Platform**: To create a responsive, easy-to-use interface that enables job seekers to search for jobs, apply for them, and manage their profiles, while employers can post new job listings and view applications.
- **Enable Secure User Authentication**: Implement JWT-based authentication to ensure secure access for job seekers and employers, enabling user login, registration, and profile management.
- **Job Search & Filter Features**: Allow job seekers to search and filter job listings by criteria such as **location**, **job type**, and **title**, making it easier to find suitable opportunities.
- **Job Application System**: Provide users with the ability to apply for jobs, track their applications, and manage them in a seamless workflow.
- **Job Posting by Employers**: Enable employers (admins) to post new job listings, manage the application process, and engage with potential candidates.
- **Responsive and Scalable Application**: To ensure the application is fully responsive, providing an optimized user experience on both desktop and mobile platforms, and is built to scale as the user base grows.
- **Real-Time Data Management**: Utilize MongoDB to store all job listings, user profiles, and applications, ensuring real-time access and updates across the platform.
- **Improve Web Development Skills**: Gain hands-on experience in full-stack web development using the MERN stack and related tools (React, Node.js, Express, MongoDB) while focusing on creating a seamless frontend-backend interaction.

Tools & Technologies Used

MongoDB (Database Management)

- **Purpose**: MongoDB is a NoSQL database used to store the application's data in the form of collections and documents.
- **Role**: It stores critical data such as job listings, user profiles, applications, and any other dynamic data. MongoDB's flexible schema allows you to store various data types and easily scale as the application grows.

Express.js (Backend Server Framework)

- **Purpose**: Express.js is a minimal and flexible Node.js web application framework that simplifies the creation of robust backend applications.
- **Role**: In this project, Express is used to handle routing, request handling, and middleware for the backend. It ensures that job seekers can send and receive data to/from the frontend securely and efficiently. It also manages the RESTful API endpoints for the application (such as GET, POST, PUT, and DELETE routes).

React.js (Frontend UI Development)

- **Purpose**: React.js is a JavaScript library for building user interfaces, primarily for single-page applications (SPAs).
- **Role**: React is used to create the frontend of the application. It handles dynamic rendering of components, such as job listings, job details, user registration/login forms, and job applications. React's component-based structure allows for reusable UI elements and faster rendering of changes.

Node.js (Server-Side Runtime)

- **Purpose**: Node.js is a JavaScript runtime environment that runs on the server side, enabling JavaScript to be used for both client-side and server-side development.

- **Role**: Node.js is used to run the backend of the application, where the Express.js framework operates. It helps handle asynchronous operations, such as reading from the database or processing multiple user requests, making the application fast and efficient.

Vite (Frontend Build Tool)

- **Purpose**: Vite is a next-generation build tool that significantly improves the development experience for modern web applications.
- **Role**: Vite is used for bundling and fast refreshing of the frontend application during development. Unlike other build tools like Webpack, Vite provides faster startup times and hot module replacement (HMR), making the development process smoother and quicker.

Tailwind CSS (Styling UI Components)

- **Purpose**: Tailwind CSS is a utility-first CSS framework that provides pre-defined classes for building custom designs directly in HTML files.
- **Role**: Tailwind CSS is used to style the frontend UI components such as buttons, forms, layouts, and other visual elements. With Tailwind's utility classes, you can create highly customized and responsive designs without writing a lot of custom CSS.

JWT (JSON Web Token - User Authentication)

- **Purpose**: JWT is a token-based authentication method that securely transmits information between the frontend and backend.
- **Role**: JWT is used to authenticate and authorize users. When a user logs in, the backend generates a token, which the frontend uses to prove that the user is authorized to access certain parts of the application (e.g., applying for jobs, posting jobs). This ensures that secure, token-based communication happens between the server and the client.

Postman (API Testing)

- **Purpose**: Postman is an API testing tool that helps developers test and debug APIs during development.
- **Role**: Postman is used to test the RESTful API endpoints in the backend. It allows developers to send requests (GET, POST, PUT, DELETE) and check the responses, ensuring the backend functions as expected. It also helps validate that data is being correctly sent and received between the frontend and backend.

Git & GitHub (Version Control and Repository)

- **Purpose**: Git is a version control system, and GitHub is a cloud-based repository service for managing code.
- **Role**: Git is used to track changes in the code, allowing developers to collaborate efficiently and avoid conflicts. GitHub stores the project repository, enabling you to push and pull updates to/from the cloud, share code with teammates, and maintain a history of the project's progress. It also allows for version management and code reviews during development.

Table 01: Tools & Technologies Used

Tool/Technology	Purpose
MongoDB	Database management and storage
Express.js	Backend server framework for handling API requests
React.js	Frontend library for building user interfaces
Node.js	JavaScript runtime for backend server-side operations
Vite	Fast build tool for frontend development
Tailwind CSS	Styling UI components using utility classes
JWT	Token-based authentication for secure user access
Postman	API testing tool for validating endpoints
Git & GitHub	Version control and code repository management

System Architecture

The architecture of this web application is divided into two main parts:

Frontend (Client Side): Developed using React.js. Interacts with backend APIs for data retrieval and submission. Features include job listing, job detail view, search filter, user login/signup.

Backend (Server Side): Built using Node.js and Express.js. Manages all database operations. Handles authentication, job posting, and user profile CRUD operations.

Database: MongoDB stores all job listings, user data, and applications.

Authentication: Implemented via JWT (JSON Web Token). Secure login/signup with token-based access control.

Key Features of this Project

Key features of this project is explain step by step below:

User Registration & Login: Secure authentication using JWT and password encryption using bcrypt.

Job Listings: Real-time job feed pulled from MongoDB with filtering capabilities.

Job Details Page: Each job post displays detailed information including title, salary, requirements, etc.

Apply to Jobs: Logged-in users can apply for jobs, with applications stored in the database.

Post New Jobs: Admin/employer role can add new job postings using secure forms.

Responsive Design: Fully responsive layout built using Tailwind CSS to ensure smooth experience across devices.

Profile Management: Users can create, view, and update their personal information.

Admin Dashboard (optional): Access to all user and job management features.

Flowchart

Flowchart of this project is given below

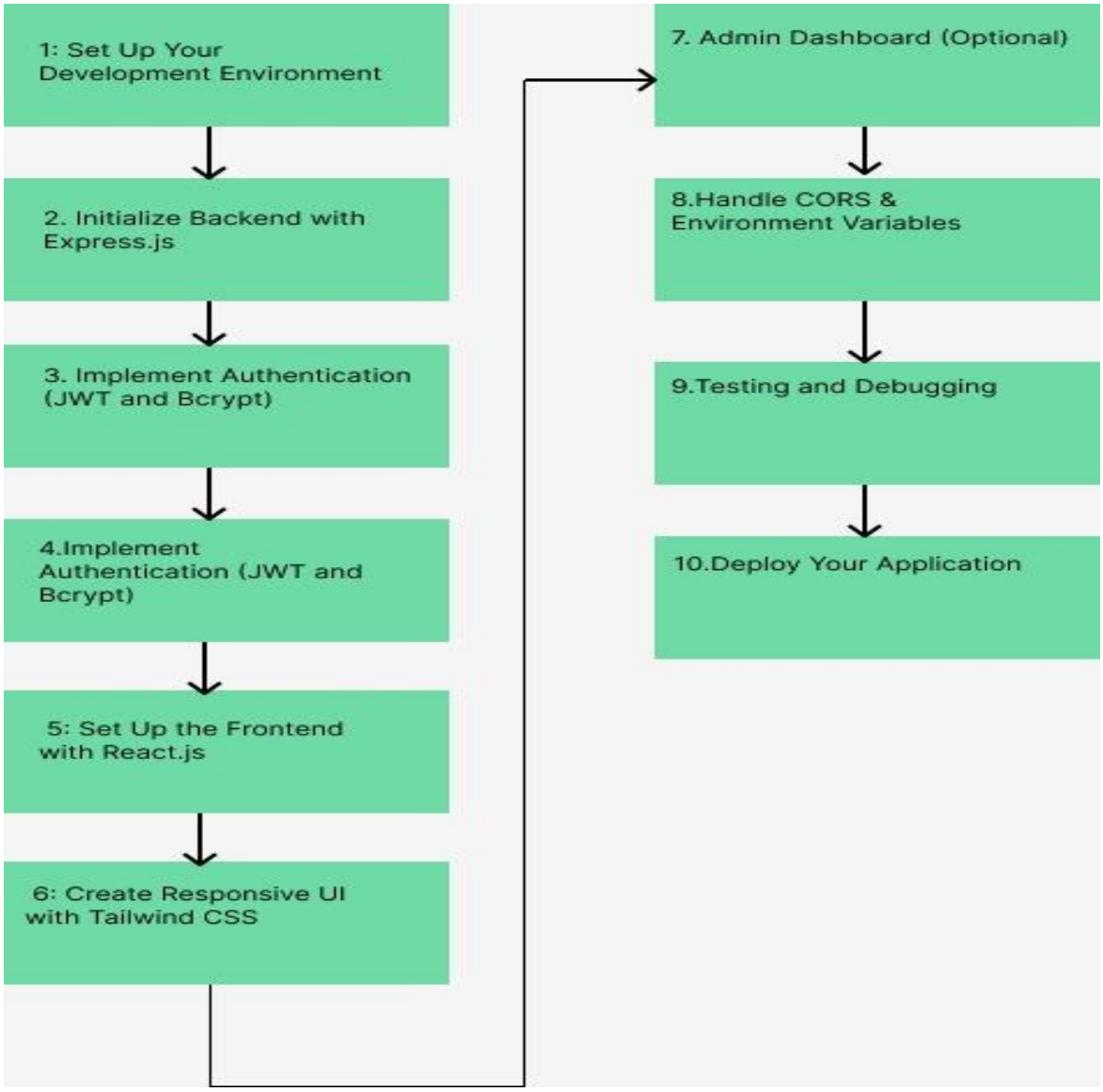

Figure 01: Flowchart of this proposed project

Algorithm

Job Seeker Hub Development algorithm is given below

1. **Set Up Your Development Environment**

- Install necessary tools (Node.js, npm, MongoDB, etc.) and initialize project directories for backend and frontend.

2. **Initialize Backend with Express.js**
    - Set up an Express server to handle requests and manage routes for job posting, job searching, and user authentication.
3. **Implement Authentication (JWT and Bcrypt)**
    - Use JWT for secure user authentication and bcrypt to hash passwords for safe storage in the database.
4. **Create RESTful API Routes for Job Management**
    - Implement CRUD operations (Create, Read, Update, Delete) for job listings, user profiles, and job applications.
5. **Set Up the Frontend with React.js**
    - Set up React.js to build a dynamic, responsive user interface where job seekers and employers can interact with the application.
6. **Create Responsive UI with Tailwind CSS**
    - Use Tailwind CSS to design the frontend, ensuring the app is fully responsive on both desktop and mobile devices.
7. **Admin Dashboard (Optional)**
    - Create an admin dashboard to allow employers (admins) to post jobs and manage applications.
8. **Handle CORS & Environment Variables**
    - Set up Cross-Origin Resource Sharing (CORS) and configure environment variables for database connections and secret keys.
9. **Testing and Debugging**
    - Test all frontend and backend functionality, ensuring correct data flow, user interactions, and performance.
10. **Deploy Your Application**
    - Deploy the backend and frontend to production using services like Heroku, Netlify, or AWS, ensuring that both parts of the app are hosted and accessible online.

Screenshots & Explanation of this Project Step by Step

Home Page – Shows a list of featured job listings with filters.

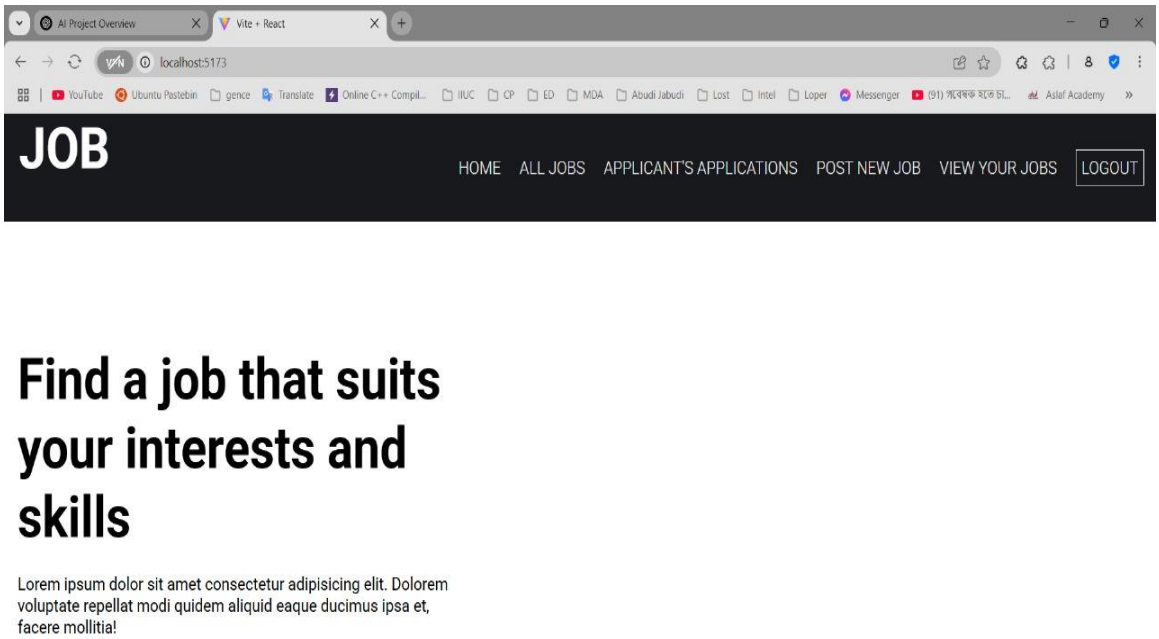

Figure 02: Home page

Login Page: Provides secure user authentication form.

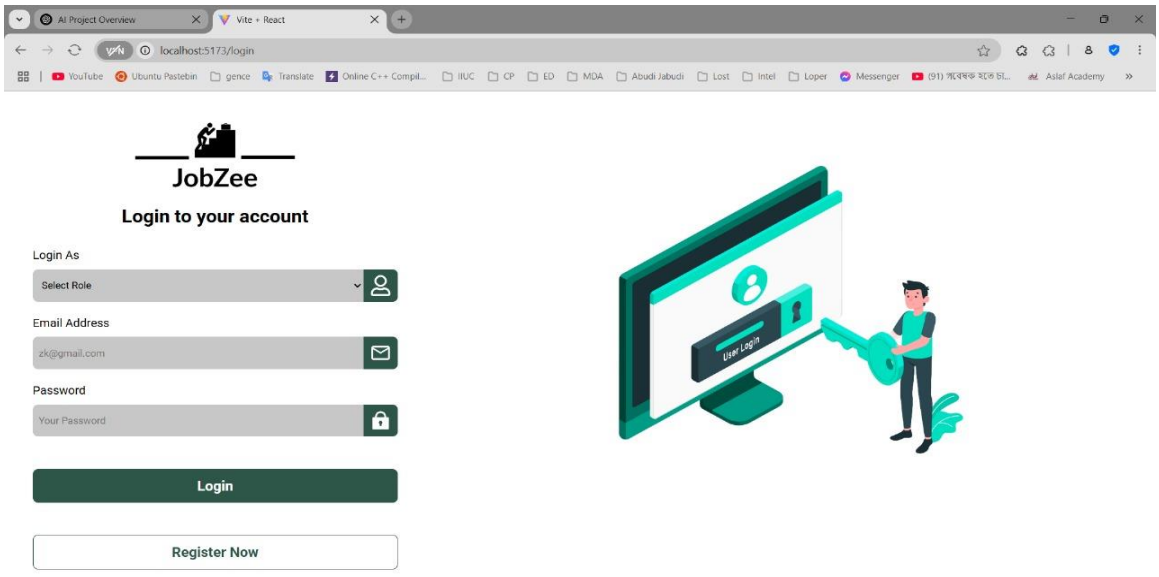

Figure 03: Login Page

Registration Page: Show the registration info.

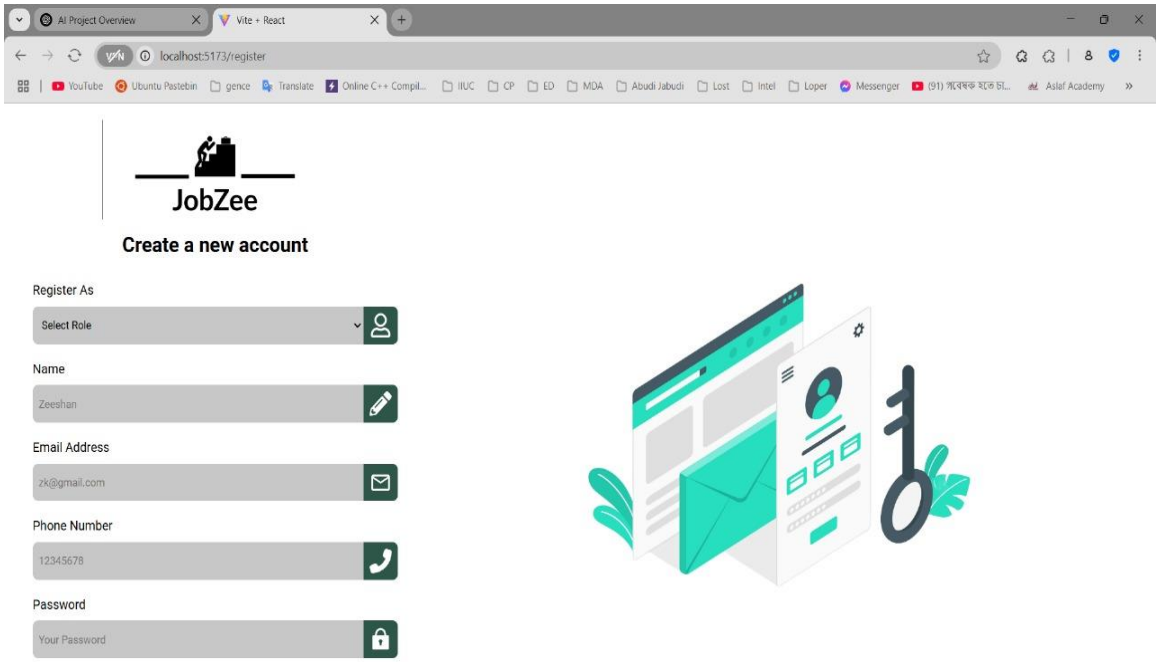

Figure 04: Registration Page

Job Detail Page: Displays complete information about a selected job.

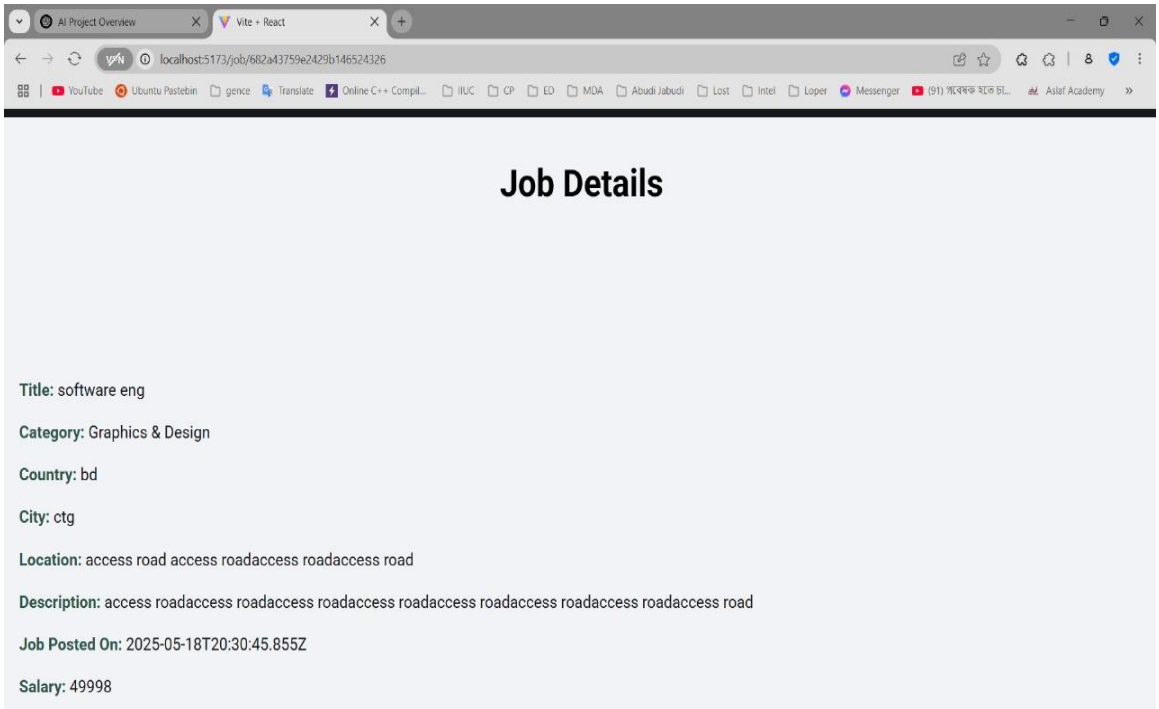

Figure 05: Job Detail Page

Post Job Page: Employer can submit new job listings.

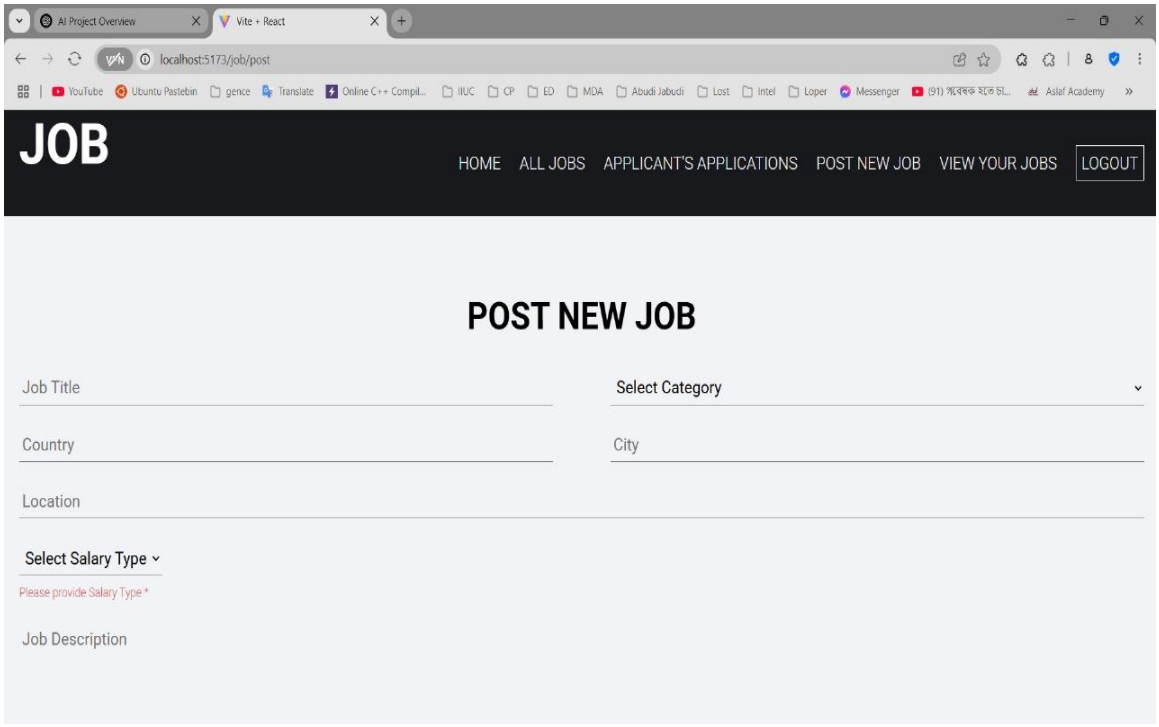

Figure 06: Post Job Page

Job Application Form:

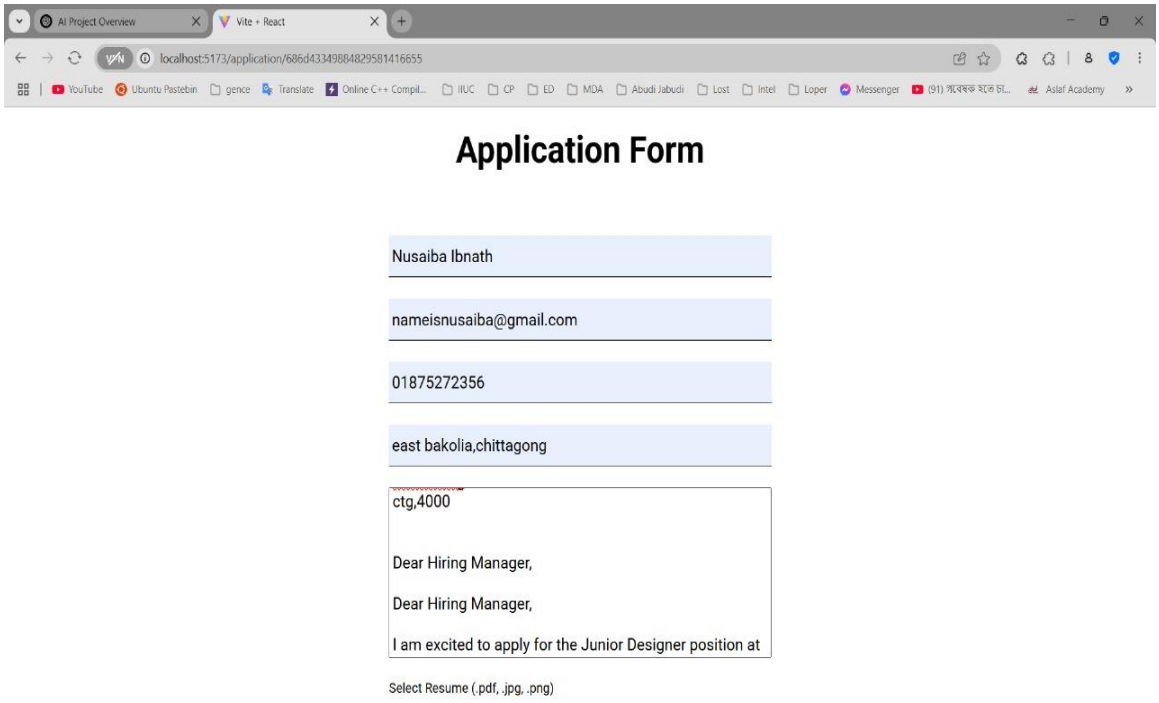

Figure 07: Job Application Form

Posted jobs:

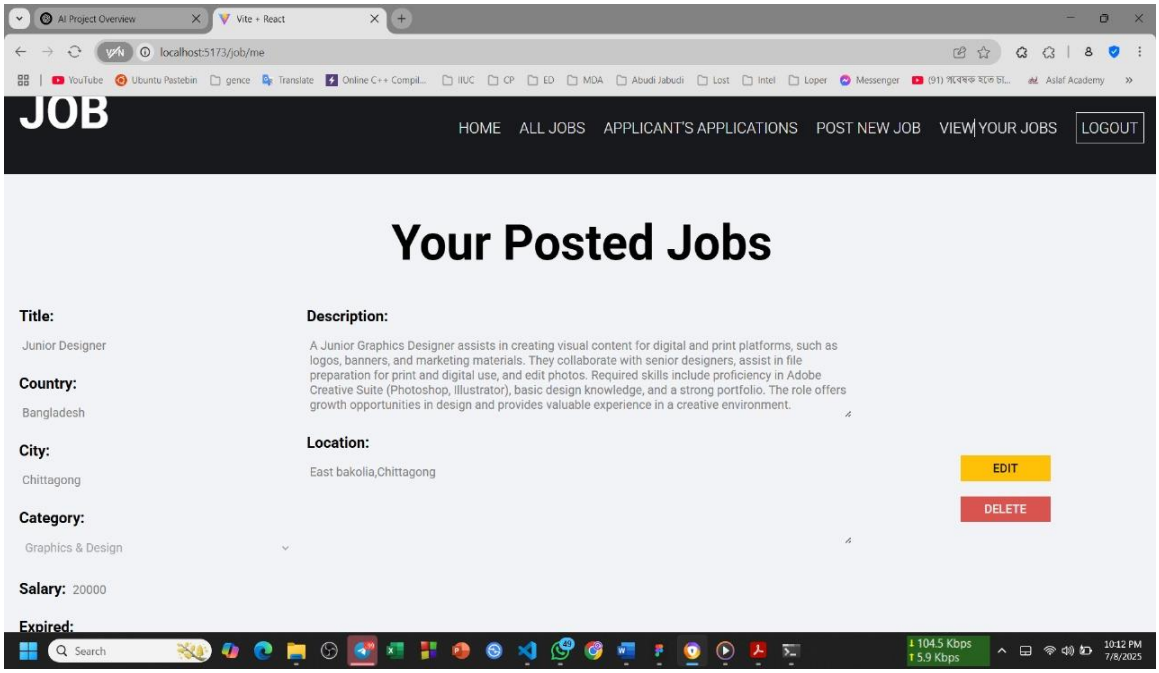

Figure 08: Posted Jobs

Available Job seekers:

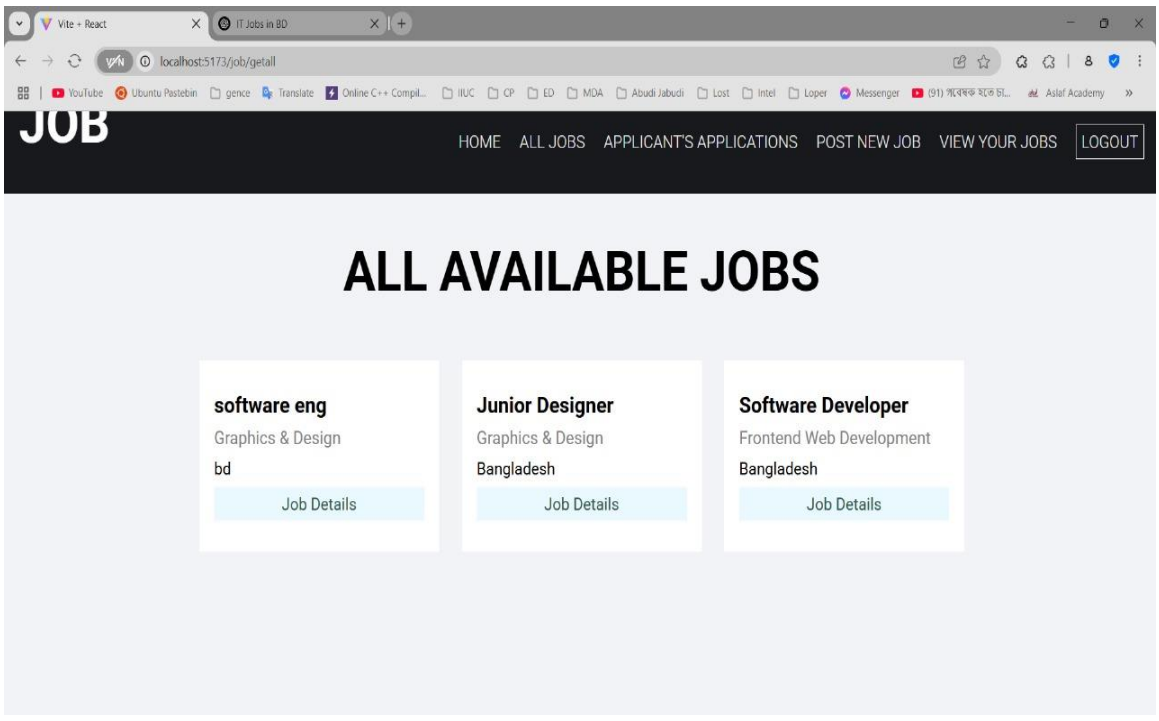

Figure 09: All available jobs

Challenges Faced & Solutions

Frontend-backend communication: Used Axios with proxy setup in Vite config to resolve CORS issues.

Secure authentication: Implemented JWT tokens and bcrypt for encrypted passwords.

UI responsiveness: Leveraged Tailwind utility classes for mobile-first responsive design.

CRUD operations: Used Express routing and Mongoose schema models for efficient database operations.

Error handling: Implemented try-catch blocks and middleware for global error handling.

Outcome of this Project

- Gained real-world experience building a full-stack web application.
- Enhanced understanding of React component structure and state management.
- Built and consumed RESTful APIs with secure data flow.
- Improved knowledge in MongoDB database modeling.
- Learned to manage user authentication and authorization using JWT.

Conclusion

The "Job Seeker Hub" web application demonstrates the power and flexibility of the MERN stack (MongoDB, Express.js, React.js, and Node.js) in building a fully functional, scalable job platform**.** The system serves as a bridge between job seekers and recruiters**,** allowing job seekers to search and apply for jobs, while employers can post job listings and manage applications efficiently. This application highlights the capabilities of the MERN stack in real-world software development, showcasing how each technology contributes to the overall functionality of the system. React.js is used for building the frontend user interface, Express.js and Node.js work together to handle server-side operations, and MongoDB serves as the database for storing job listings, user profiles, and applications.

References

MongoDB Documentation – https://www.mongodb.com/docs/

React.js Documentation – https://reactjs.org/

Express.js Guide – https://expressjs.com/

Vite Official Docs – https://vitejs.dev/

Tailwind CSS – https://tailwindcss.com/