INTERVIEWXP project overview

InterviewXP

Project Name: InterviewXP

Subtitle: A Social Platform for Sharing Interview Experiences

Dev-Name: ABHISHEK

Tech Stack: MERN (MongoDB, Express, React, Node.js)

➢ Problem Statement

"The Information Gap in Technical Recruitment"

Computer science students and job seekers often face significant anxiety and uncertainty during the recruitment process due to a lack of transparent, accessible information. While generic interview questions are widely available, specific insights into recent interview patterns, company-specific rounds, and real-world experiences are often scattered across fragmented forums or locked behind paywalls. This lack of centralized, authentic peer-to-peer knowledge makes it difficult for candidates to prepare effectively, leading to missed opportunities and inefficient preparation strategies.

➢ Proposed Solution

"InterviewXP: A Community-Driven Interview Experience Platform"

To address this challenge, we developed **InterviewXP**, a full-stack web application designed to democratize access to interview knowledge.

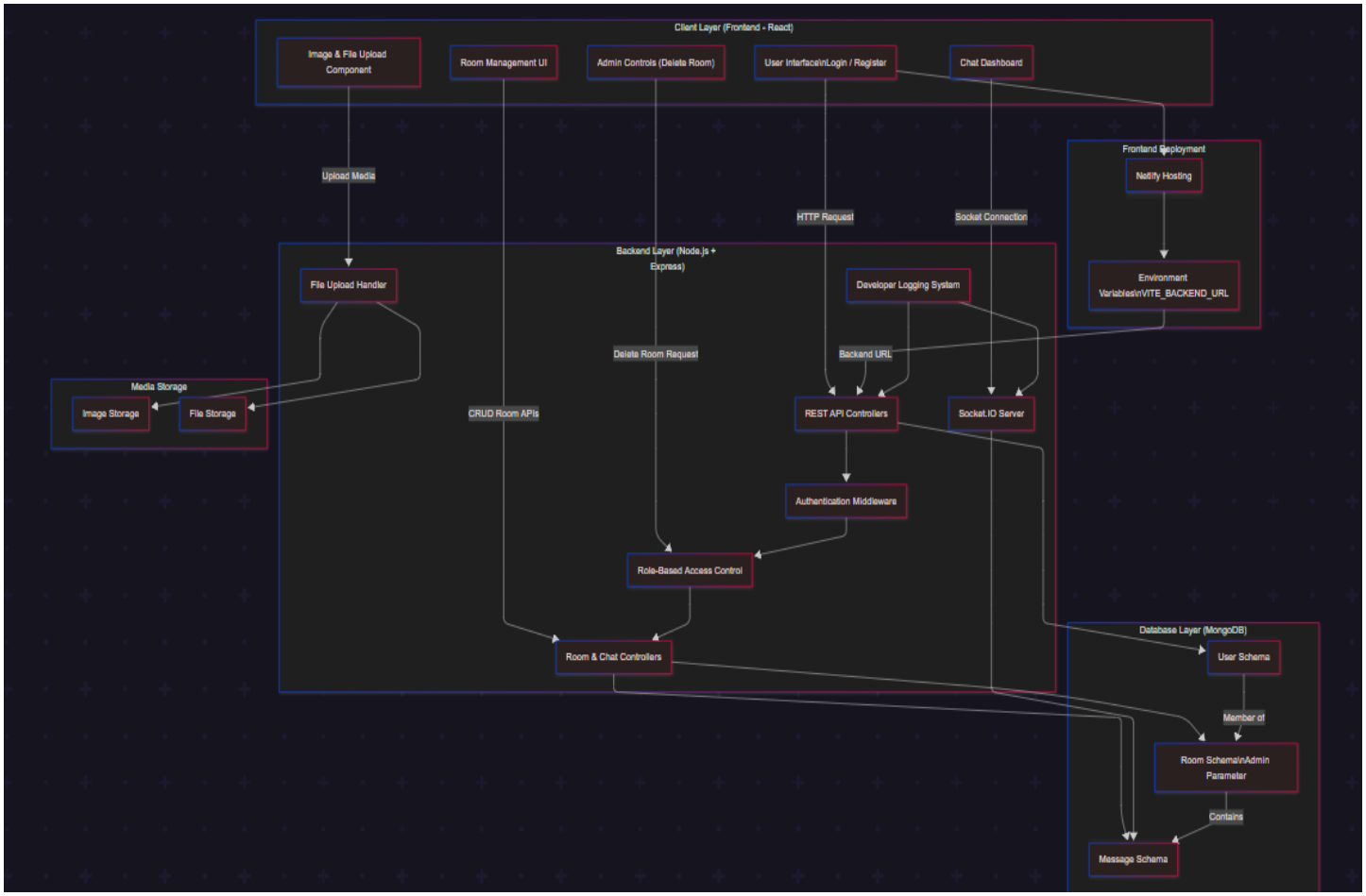
- **Centralized Knowledge Hub:** A dedicated social platform where students and professionals can document and share their detailed interview experiences, including specific questions asked, the difficulty level of rounds, and the overall recruitment process.
- **MERN Stack Architecture:** Built using **MongoDB, Express.js, React, and Node.js**, the solution ensures a responsive and scalable environment for users to create, read, and manage interview stories.
- **Peer-to-Peer Learning:** By enabling users to publish their own experiences, the platform fosters a collaborative community where successful candidates help aspirants navigate the complexities of technical interviews

INTERVIEWXP project overview

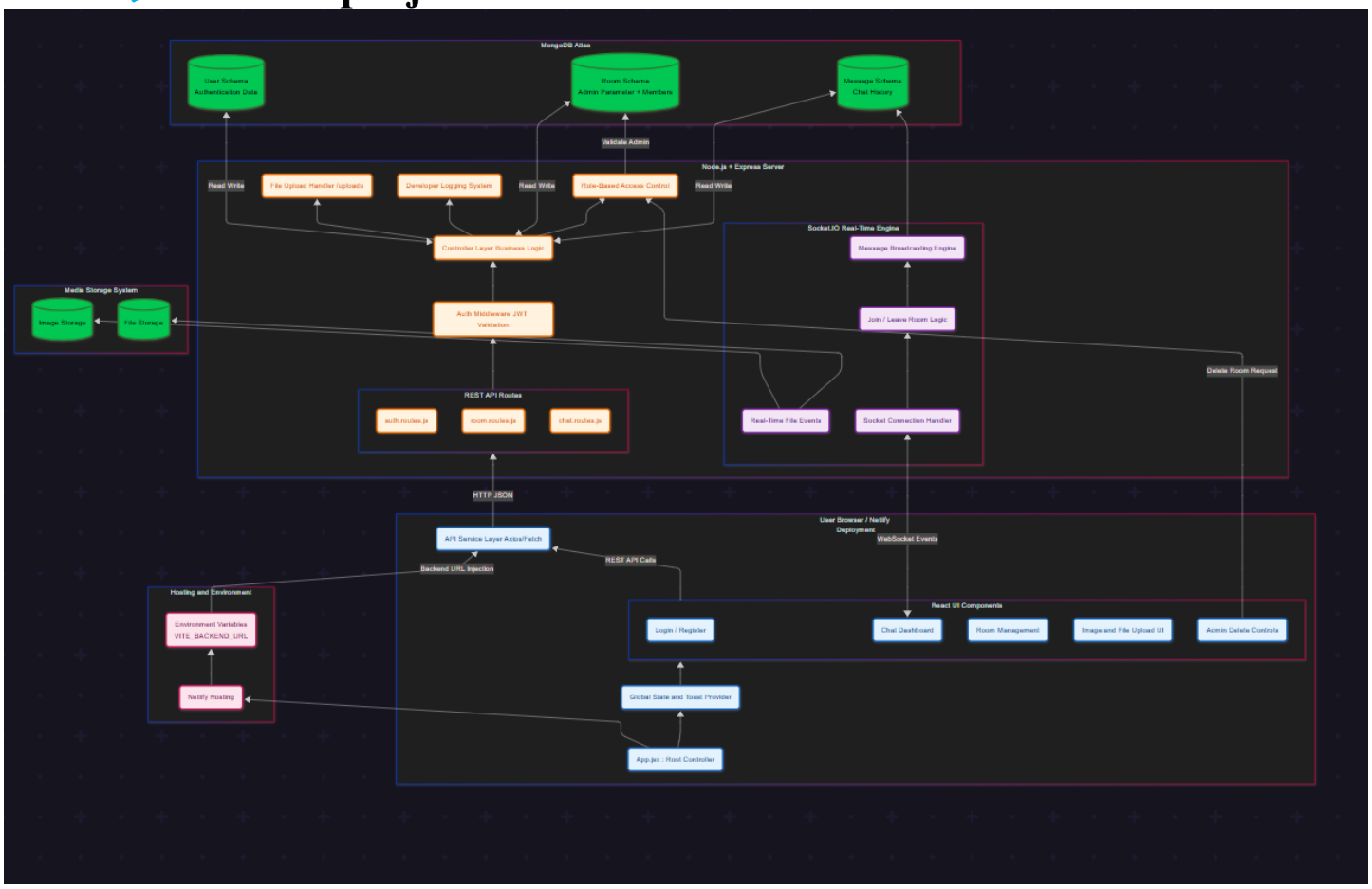
➢ System Architecture

The application follows a **Monolithic MERN Architecture**, ensuring seamless communication between the client-side interface and the server-side database.

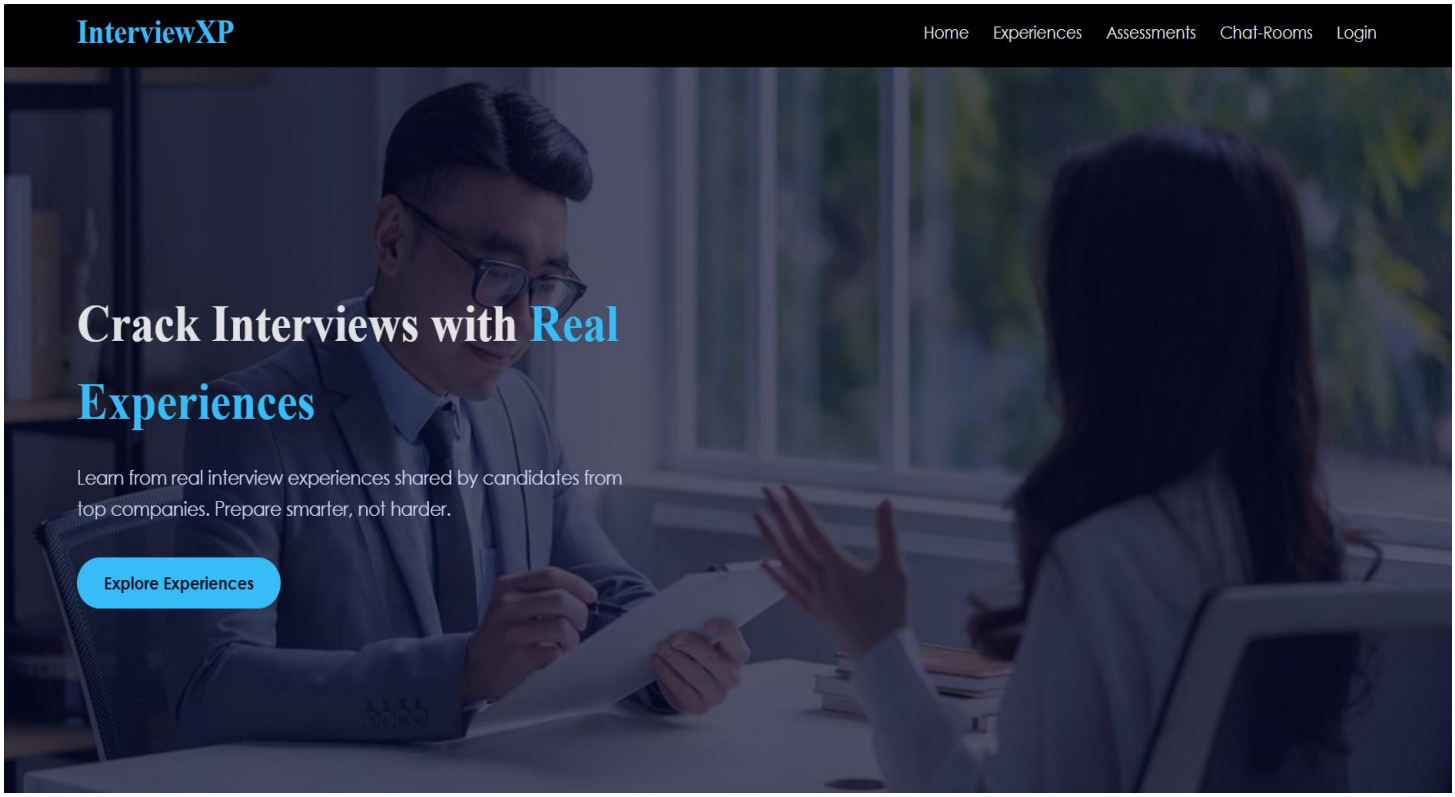
- **Frontend (Client-Side):**
  - **React.js:** Used to build a dynamic, single-page application (SPA) that provides a responsive user interface. Component-based structure ensures code reusability and efficient state management.
  - **Tailwind CSS / CSS Modules:** (Assuming you used one) employed for styling to ensure a modern, accessible, and mobile-responsive layout.
- **Backend (Server-Side):**
  - **Node.js & Express.js:** Serves as the RESTful API layer, handling HTTP requests, routing, and middleware logic. It manages data flow between the frontend and the database.
- **Database:**
  - **MongoDB:** A NoSQL database used to store unstructured data such as user profiles, interview experiences, and comments. Its flexibility allows for scalable data schemas.



INTERVIEWXP project overview

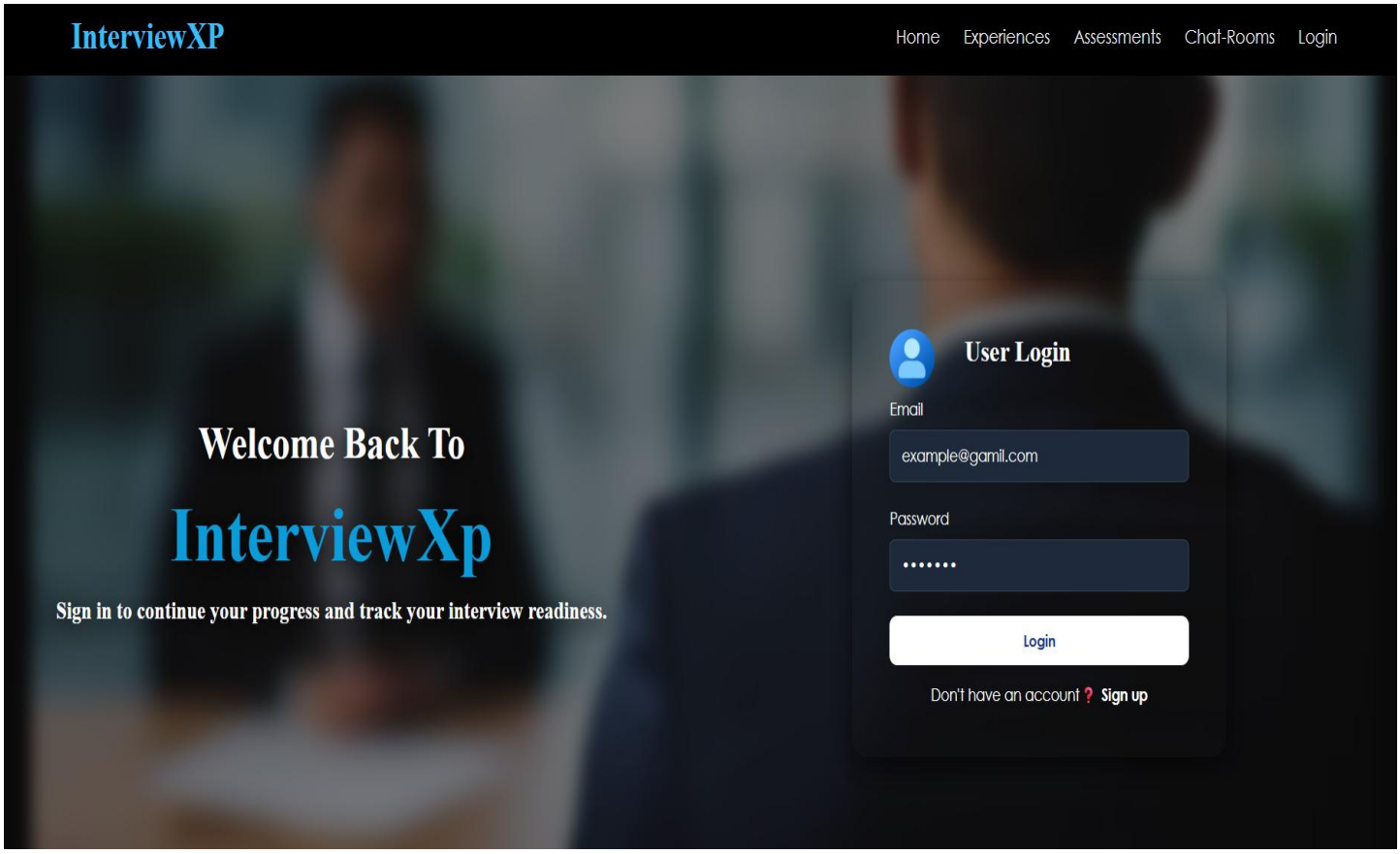


➢ APPLICATION SCREENSHOTS

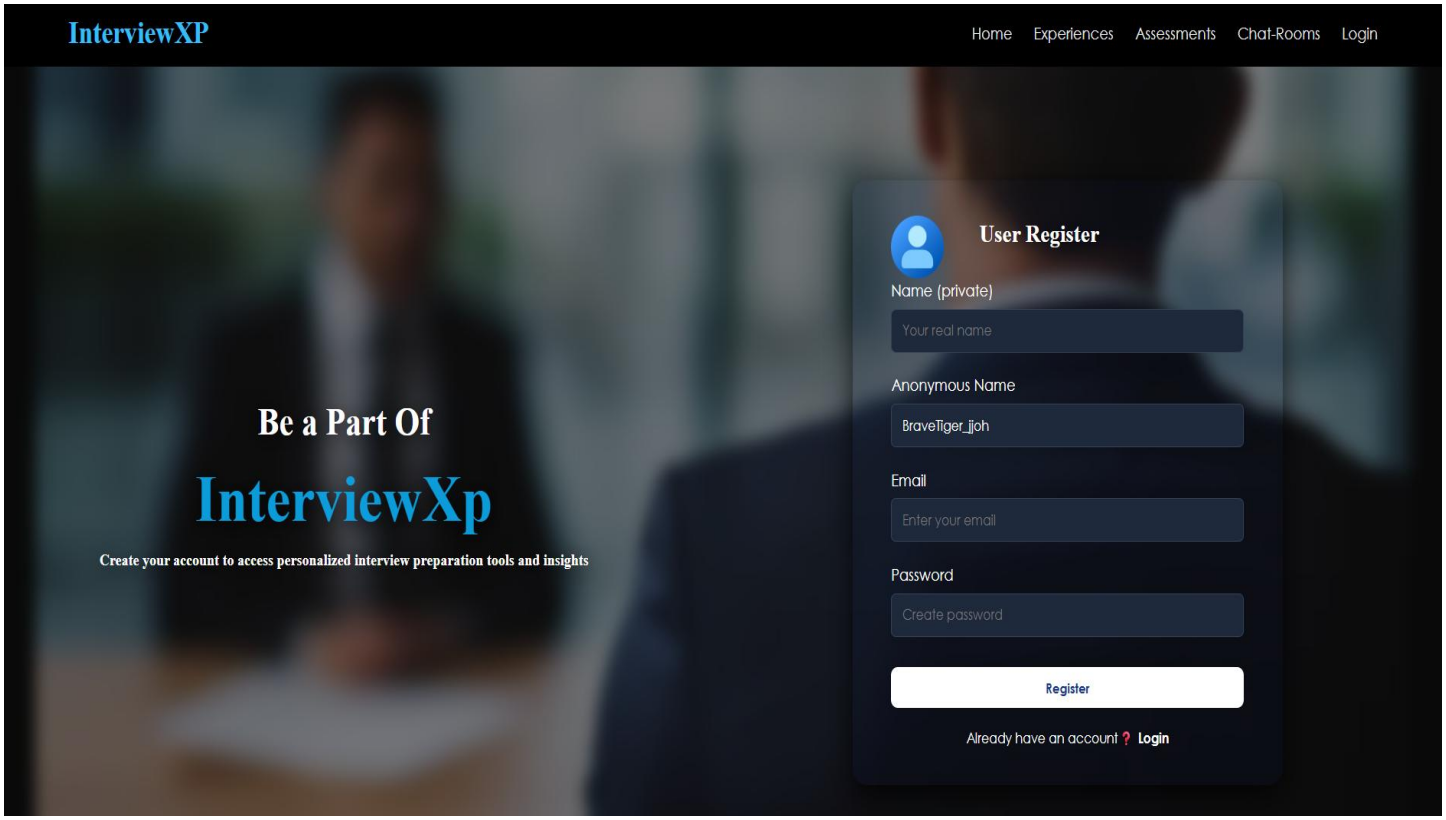


INTERVIEWXP project overview

Login Page:

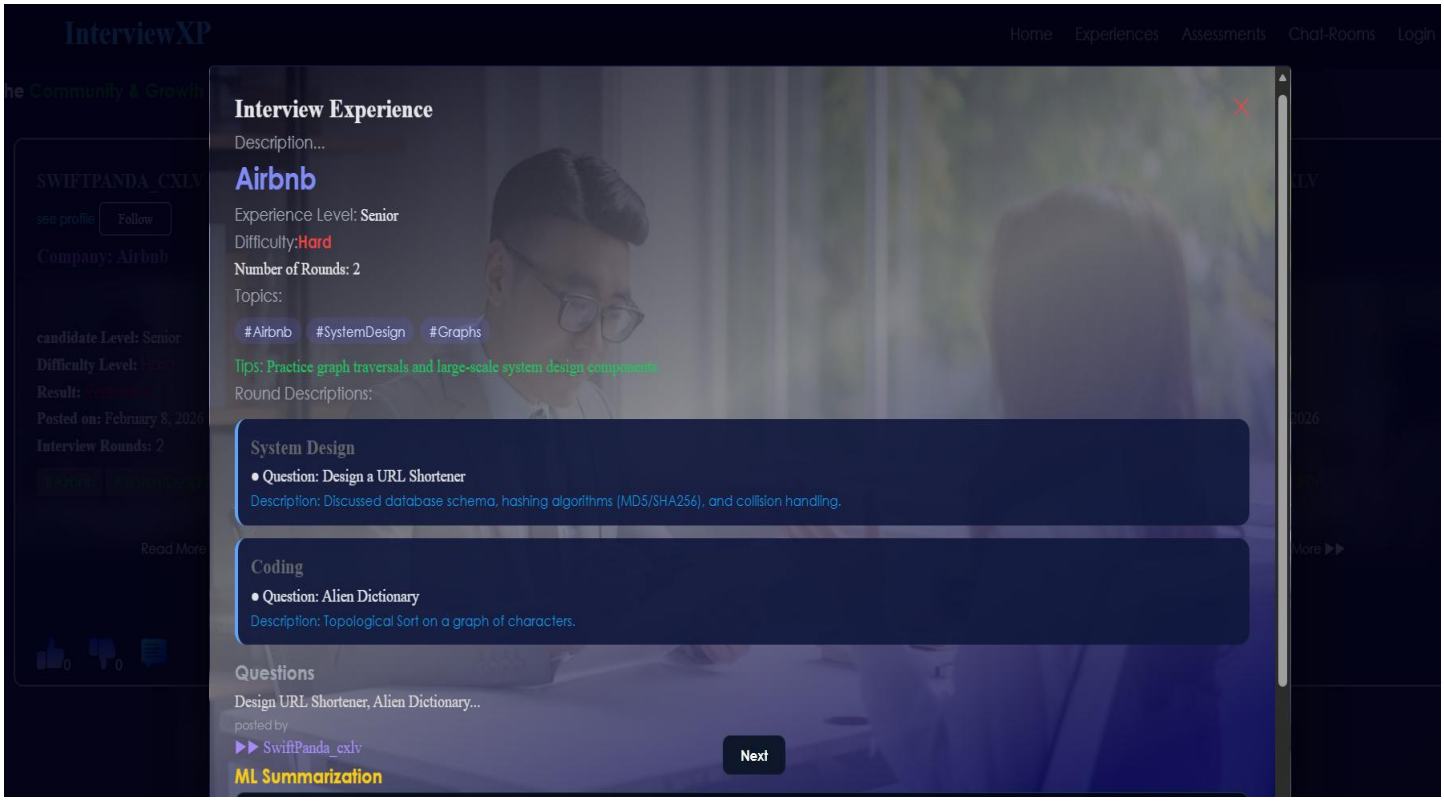


Register Page:

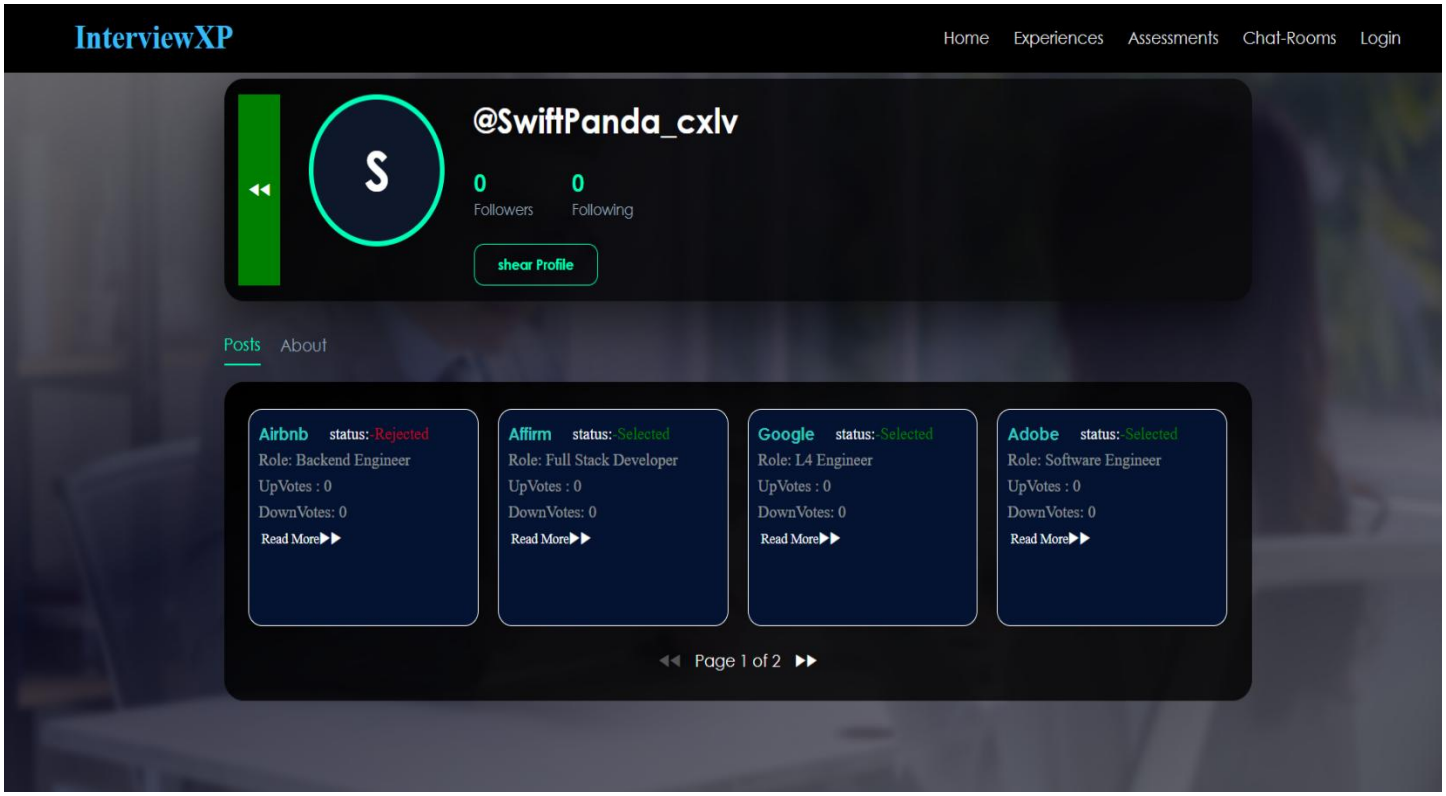


INTERVIEWXP project overview

Interview Experience Description Page

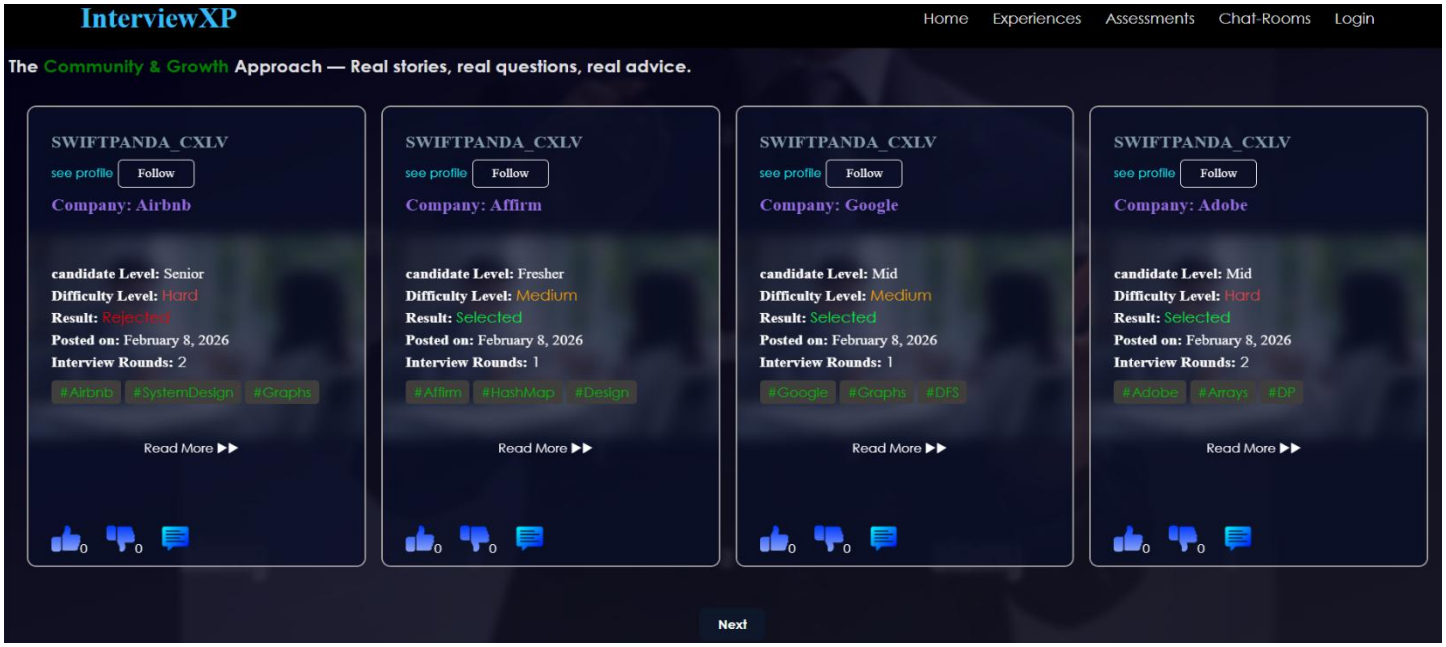


Users Profile Page:

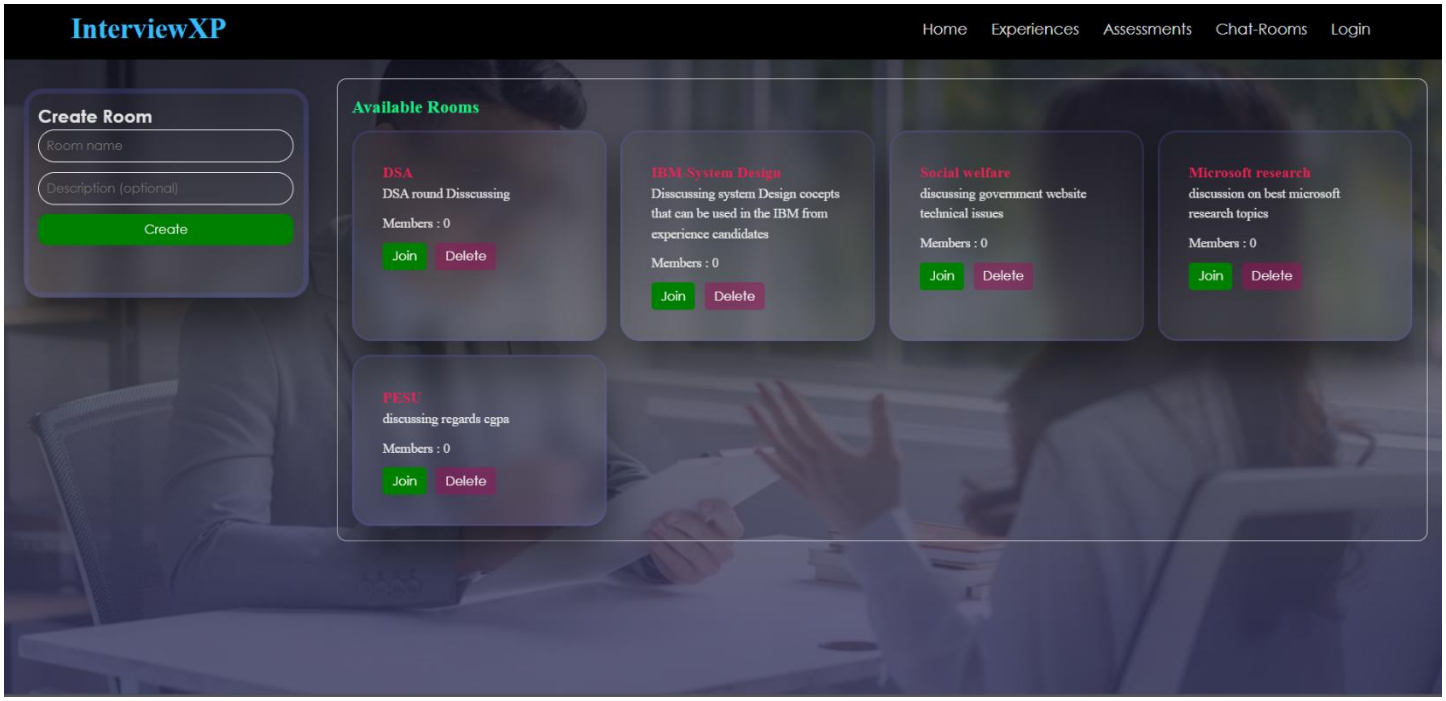


INTERVIEWXP project overview

Experiences section:



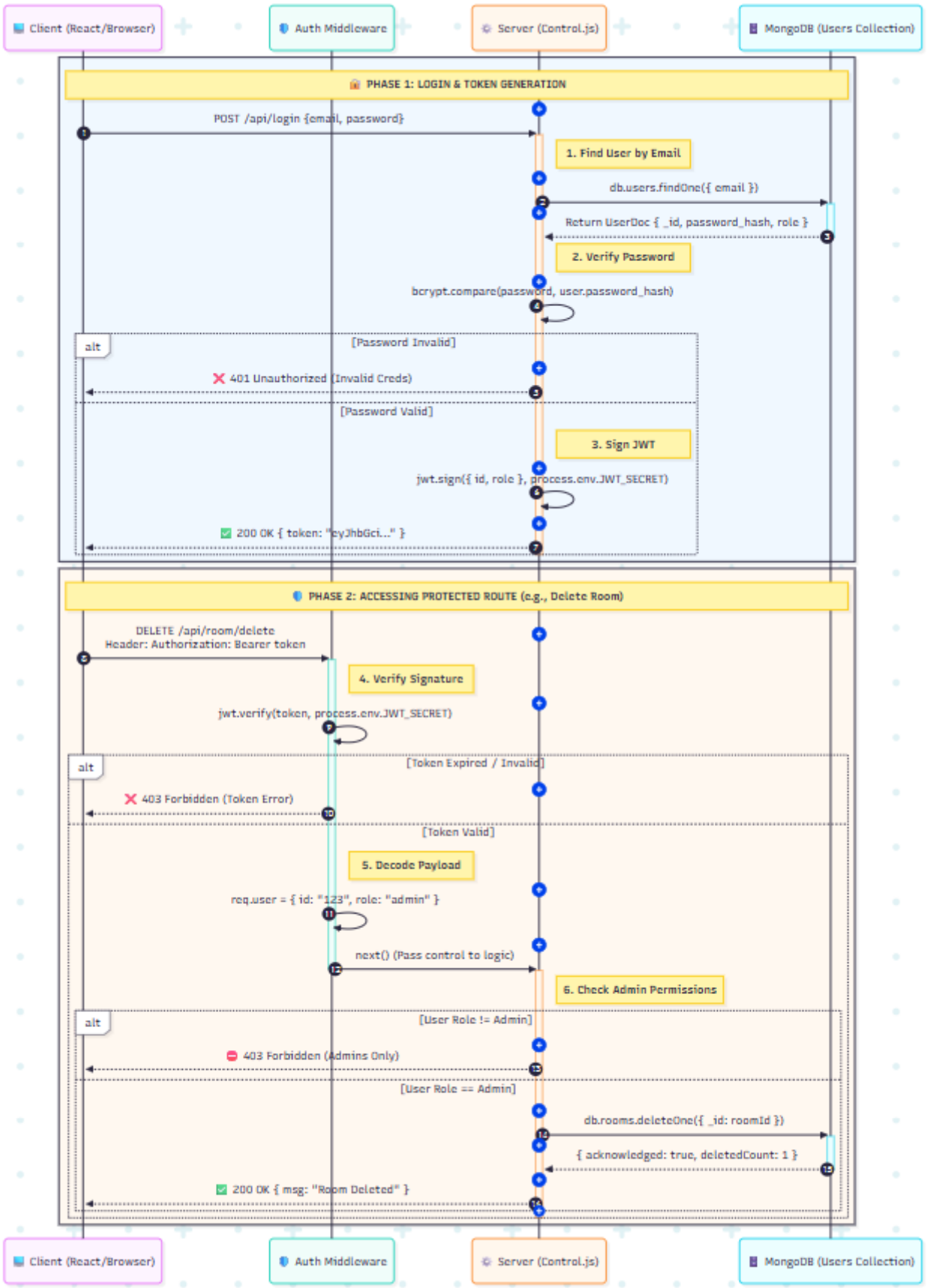
Chat-Rooms Section:



➢ JWT ARCHITECTURE

JSON Web Tokens (JWT) are used to implement secure, stateless authentication by transmitting information between the client and server as a compact, digitally signed JSON object. When a user logs in, the server generates a token containing their encoded identity (User ID) and privileges (e.g., Admin Role), signed with a private secret key. We use JWTs because they eliminate the need for server-side session storage, making the application highly scalable and efficient. This allows the server to verify a user's identity for protected routes—such as deleting chat rooms—simply by validating the token signature in the HTTP header of each request.

INTERVIEWXP project overview



INTERVIEWXP project overview

KEY FEATURES

❖ Authentication System

We implemented a secure user authentication system with Login and Registration functionality. The frontend was built using React, while backend validation ensures proper user verification before access. Production-level CSS and routing issues were resolved to ensure smooth deployment. User-friendly permission popups replaced default browser alerts for better UX.

❖ Real-Time Group Chat System

We developed a real-time room-based chat system using Socket technology for instant communication. Users can join and leave chat rooms dynamically without refreshing the page. Messages are broadcasted to all active members in a room. Production-level socket connection issues were identified and fixed to ensure stable real-time communication.

❖ Image and File Sharing in Chat

The chat system supports image and file sharing within groups. Users can upload media files which are stored and displayed inside the chat interface. This enhances collaboration and simulates real-world communication platforms. Proper backend handling ensures secure file transmission and storage.

❖ Room-Based Architecture & Database Design

We designed a structured MongoDB schema to manage users, rooms, and messages efficiently. Each room maintains its own chat history and member mapping. The backend ensures data consistency and proper relationships between users and rooms. This modular database design makes the system scalable.

❖ Role-Based Access Control (Admin System)

We implemented an admin-based room management system. Only the room creator (admin) has permission to delete the room. A new parameter was added in the MongoDB schema to identify the admin and compare it with the current user. The delete button is dynamically enabled or disabled based on user authorization.

❖ UI/UX Improvements

Multiple UI enhancements were made to improve visual appeal and flexibility. Backgrounds were redesigned and layout glitches were fixed for better responsiveness. Production CSS issues were resolved to ensure consistent design across environments. The interface was optimized for a smoother user experience.

❖ **Deployment & Production Optimization**

The project was deployed using Netlify with proper React Router redirects configuration. Environment variables were configured to manage backend URLs in production. Socket connection errors and missing image issues were fixed during deployment. Conflict resolution and Git handling were managed efficiently during version control.

❖ **Developer Logging & Debugging System**

We added developer-level logging in the backend to monitor server activity and debug issues efficiently. Logs help in identifying errors related to socket connections, database operations, and API requests. This improves maintainability and simplifies troubleshooting in production environments.

❖ **Ethics & Privacy Documentation**

We included ethics and privacy considerations in the project documentation. This outlines responsible data handling and user privacy awareness. It demonstrates an understanding of real-world application standards. Including this section adds professionalism to the project.

➢ Impact & Performance Evaluation

- ❖ Successfully implemented a real-time room-based communication system supporting concurrent users through WebSocket architecture.
- ❖ Reduced UI permission-related errors by replacing browser alerts with structured validation and role-based access control.
- ❖ Achieved stable production deployment by resolving socket connection issues and environment configuration mismatches.
- ❖ Designed a scalable MongoDB schema supporting dynamic room creation, admin-based authorization, and persistent chat history.
- ❖ Integrated file and image sharing within group chats, improving collaboration capability beyond basic text communication.
- ❖ Implemented structured developer-level logging to enhance debugging efficiency and reduce backend issue resolution time.
- ❖ Deployed full-stack architecture using Netlify and environment-based backend configuration for seamless production readiness.

➢ Future Scope & Roadmap

To elevate **InterviewXP** from a social platform to a comprehensive recruitment ecosystem, the following modules are currently under active development:

A. AI-Driven Virtual Interviewer

- **Objective:** To simulate real-world technical rounds.
- **Implementation:** We are integrating Large Language Models (LLMs) to create a virtual interviewer that asks context-aware questions based on the user's resume and provides real-time feedback on their answers.

B. Transition to Microservices Architecture

- **Objective:** To improve scalability and fault isolation.
- **Implementation:** The current monolithic backend will be decoupled into independent services (e.g., *Auth Service*, *Chat Service*, *Assessment Service*), each running in its own container (Docker) to handle high traffic loads efficiently.

C. Secured Assessment Protocol

- **Objective:** To conduct fair and cheat-proof coding tests.
- **Implementation:** A browser-locking mechanism and tab-switch monitoring system will be implemented to ensure the integrity of online coding assessments.

D. User Experience Enhancements

- **Dynamic Tech Feed:** A "Trending in Tech" section that fetches random, latest posts related to the tech world (using external APIs) to keep users updated on industry trends.
- **Advanced Security (Forgot Password):** Implementing secure password recovery flows using **Nodemailer** and short-lived OTP tokens to restore user access safely.

INTERVIEWXP project overview

➢ ENVIRONMENT VARIABLES

FRONTEND:

VITE_BACKEND_URL	http://localhost:port (for local machines)

BACKEND:

JWT_SECRET	Your JWT key
DB_URL	Localhost database URL / atlas URL
PORT	8000 /etc.
NODE_ENV	Production / development
CLIENT_URL	Frontend URL for socket cores

PROJECT STRUCTURE: