**JPath - Project Documentation**

**1.Introduction**

JPath is a full-stack application designed to assist users with job-related tasks, including job management, user authentication, skill recommendations, and chatbot interactions. This document outlines the project's architecture, setup process, key features, and code structure.

**2. Project Overview**

**Goals**: - Facilitate job searching and application processes. - Provide user authentication and profile management. - Offer skill recommendations to enhance career development. - Integrate a chatbot for interactive assistance.

**Technology** **Stack**: - **Frontend**: React Native - **Backend**: Node.js, Express.js **Database**: MongoDB (inferred from common MERN stack practices and project structure)

**3. System Architecture**

Client (React Native) <--> Server (Express API) <--> MongoDB

**4. Installation & Setup**

1. **Clone the repository:** bash git clone cd JPath-final
2. **Install Backend Dependencies**: bash cd backend npm install
3. **Install Frontend Dependencies**: bash cd ../frontend npm install
4. **Configure Environment Variables:** Create a .env file in the backend directory and add necessary environment variables (e.g., MONGODB\_URI , JWT\_SECRET )
5. **Start Backend Server:** bash cd backend node app.js
6. **Run Frontend Application:** bash cd frontend npx expo start Follow the instructions in the terminal to run on an emulator or physical device.

**5. Usage Guide**

* **Core Features:** Job Management: Create, view, update, and delete job listings.
* **User Authentication:** Secure user registration, login, and profile management.
* **Skill Recommendations:** Suggest relevant skills based on user data or job roles.
* **Chatbot Interaction:** An AI-powered chatbot to assist with job-related queries.
* **Resume Management:** Functionality to manage and upload user resumes.

**6. Code Structure**

JPath-final/

├── backend/ # Node.js/Express backend

│ ├── Controllers/ # Business logic and request handlers

│ │ ├── ControlJob.js

│ │ ├── ControlJob2.js

│ │ ├── ControlJob3.js

│ │ └── jobapis.js

│ ├── Model/ # Database schemas and models

│ │ └── SchemasJob.js

│ ├── Routes/ # API route definitions

│ │ ├── RouterJop.js

│ │ └── jobapirouter.js

│ ├── app.js # Main backend application entry point

│ ├── package-lock.json

│ └── package.json

│

├── frontend/ # React Native frontend

│ ├── App.js # Main frontend application entry point

│ ├── app.json

│ ├── assets/ # Static assets (images, animations)

│ ├── components/ # Reusable UI components

│ │ ├── certficateValidation.js

│ │ ├── circle.js

│ │ ├── JobDetailsScreen.js

│ │ ├── jobpath\_animated\_logo1.js

│ │ ├── SkillPath.js

│ │ └── skillsRecommand.js

│ ├── index.js

│ ├── package-lock.json

│ ├── package.json

│ └── Screens/ # Application screens/pages

│ ├── ApiCalls/

│ │ └── SignIn\_Api.js

│ ├── BottomBarNavigations/

│ │ ├── BottomBar.js

│ │ ├── HomeScreen.js

│ │ ├── ProfileScreen.js

│ │ └── StatusScreen.js

│ ├── ForgotPassword.js

│ ├── JobPage.js

│ ├── Login.js

│ ├── signup.js

│ └── StackNavigations/

│ ├── JobChatbot.js

│ ├── JobRoleDetails.js

│ ├── JobRoles.js

│ ├── PersonalInfoScreen.js

│ ├── ResumeScreen.js

│ ├── RoleDetailsAndDescription.js

│ ├── SkillsScreen.js

│ └── StackNavigation.js

**7. API Documentation**

* **Auth Endpoints: -** POST /api/auth/register : User registration. - /api/auth/login : User login.
* **Job Endpoints (inferred): -** GET /api/jobs : Retrieve all job listings. - /api/jobs : Create a new job listing. - POST POST PUT /api/jobs/:id : Update an existing job listing. - DELETE /api/jobs/:id : Delete a job listing.
* **Other Endpoints (inferred): -** Endpoints for skill recommendations. - Endpoints for chatbot interactions. - Endpoints for user profile and resume management.

**8. Database Schema**

**Users Collection (inferred): -** \_id : Unique identifier. - name : User's full name. email : User's email address (unique). - passwordHash : Hashed password. (Potentially other fields like skills , resume\_id , etc.)

**9.Testing**

The current project does not include a formally documented or automated testing framework. As such, manual testing has been the primary approach for ensuring the application behaves as expected. Manual tests were performed regularly during the development process to validate functionality, user input handling, UI responsiveness, and navigation flow. These tests helped identify and resolve any issues related to usability, performance, and integration between the front-end and back-end components.

**10.Known Issues**

The Issues we are faced in the development is the API Integration, Data to feedUp, animations in the screen and the restriction’s of the chatbot to specific feed.

The Limitations in the App are the database can’t be changed Dynamically, the backend admin has to be update it every time when the data updates in the real world.