**ReadMe file**

**MU Connect – Final Year Project Application Portal**

MU Connect is a role-based web portal developed as part of the Software Engineering course project at Mahindra University. It enables final-year students to apply for faculty-offered projects and facilitates streamlined project management for faculty and administrators.

**How to Run the Project Locally**

**Prerequisites**

* Node.js (v16+)
* MongoDB (local or cloud)
* VS Code (recommended for Live Server extension and running the whole project in the terminal)

**Clone the Repository**

git clone "https://github.com/sahasrapurumandla/my-project"

cd mu-connect

**Backend Setup**

In termina(VScode) run below commands one by one to download the node\_models and packages:

* cd backend
* npm install
* npm init -y
* npm install express mongoose dotenv bcryptjs jsonwebtoken nodemailer
* npm install cors
* npm install nodemon --save-dev
* npm install multer
* npm install mongodb

**Then run the server.j:**

npm start

you will see it is connected to mongodb

server running in port :5000

**Frontend Setup**

Navigate to the frontend folder:

cd /FEWEB

Open login-form/dist/index.html or any role-based dashboard using Live Server or manually open in browser.

Default Login Roles (for testing)

Role Email Password

Admin admin@gmail.com admin123

Faculty faculty@gmail.com faculty123

Student student@gmail.com student123

**Features**

* Role-Based Login for Students, Faculty, and Admins
* Students can view projects, apply with SOP + resume
* Faculty can view applicants and approve/reject them
* Admins manage users, projects, and applications
* Resume Upload (PDF only) using Multer
* Authentication with JWT tokens
* RESTful API following MVC architecture

**Tech Stack**

* Frontend: HTML, CSS, JavaScript
* Backend: Node.js, Express.js
* Database: MongoDB (Mongoose Atlas)
* API Testing: Thunder Client (VS Code)
* Authentication: JSON Web Tokens (JWT)
* File Upload: Multer
* Architecture: MVC Pattern

**Folder Structure**

pp/

│

├── backend/

│ ├── controllers/ # All backend logic (CRUD operations)

│ │ ├── adminController.js

│ │ ├── applicationController.js

│ │ ├── authController.js

│ │ ├── chatController.js

│ │ ├── meetingController.js

│ │ └── projectController.js

│ │

│ ├── middleware/ # Authentication and error handling

│ │ ├── authMiddleware.js

│ │ ├── errorMiddleware.js

│ │ └── uploadMiddleware.js

│ │

│ ├── models/ # Mongoose schema definitions

│ │ ├── Application.js

│ │ ├── Chat.js

│ │ ├── Meeting.js

│ │ ├── Project.js

│ │ └── User.js

│ │

│ ├── routes/ # Route definitions for each feature

│ │ ├── adminRoutes.js

│ │ ├── applicationRoutes.js

│ │ ├── authRoutes.js

│ │ ├── chatRoutes.js

│ │ ├── meetingRoutes.js

│ │ └── projectRoutes.js

│ │

│ ├── uploads/ # Uploaded resume files

│ │ └── resumes/

│ │ └── sample.pdf

│ │

│ ├── .env # Environment config`

│ ├── package.json # Project metadata and dependencies

│ └── server.js # Entry point of backend

│

├── FEWEB/ # Frontend HTML structure

│ ├── admindashboard/

│ │ ├── admindashboard.html

│ │ ├── manage-projects.html

│ │ ├── project-oversight.html

│ │ ├── reports.html

│ │ └── user-management.html

│ │

│ ├── facultydashboard/

│ │ ├── chatfaculty.html

│ │ ├── facultydashboard.html

│ │ ├── facultyschedule.html

│ │ ├── manageproject.html

│ │ └── reviewapplication.html

│ │

│ ├── login-form/

│ │ ├── dist/

│ │ │ ├── index.html

│ │ │ └── style.css

│ │ └── src/ # (if any custom scripts used)

│ │

│ ├── studentdashboard/

│ │ ├── apply.html

│ │ ├── chat.html

│ │ ├── project.html

│ │ ├── schedule.html

│ │ └── studentdashboard.html

**Evaluation Notes**

* Project is fully functional and meets all requirements
* Clear separation of concerns for Student, Faculty, and Admin roles
* Proper route protection and backend validations implemented
* Designed with modular architecture for scalability