Full Stack Development with MERN

Project Documentation

Freelancer Finder using MERN

1. Introduction

• Project Title: Freelance Finder – Discovering Opportunities, Unlocking Potential
• Team Members: Velagala Rama Devi, Kavikondala Sai Durga Rupasri, Kimidi Anusha, Lohith Naga Durga Pavan Kumar Musireddy

2. Project Overview

• Purpose: Freelance Finder is a web-based application developed to bridge the gap between freelancers and clients. The purpose of this project is to build a system where clients can post job opportunities and freelancers can search and apply for those jobs easily. The platform ensures secure authentication and role-based access so that both freelancers and clients have separate dashboards and functionalities. The project focuses on creating a smooth and responsive user experience while maintaining data security and efficient backend processing.

Features:The main features of the system include user registration and login, role-based authorization, job posting functionality for clients, job application functionality for freelancers, profile management, and secure API communication. The system also allows users to view available job listings and manage their activities effectively. By implementing this system, the project demonstrates real-world application of full-stack development concepts.

3. Architecture

• Frontend: The architecture of the project follows the MERN stack structure. The frontend of the application is developed using React.js, which provides a dynamic and component-based user interface. The frontend is designed to be responsive and user-friendly, ensuring smooth interaction across different devices.
• Backend: The backend of the application is built using Node.js and Express.js. It follows a RESTful API structure to handle client requests and send appropriate responses. The backend manages user authentication, job postings, and application data.
Database: MongoDB is used as the database for storing application data. It is a NoSQL database that stores data in the form of collections and documents. Mongoose is used to define schemas and manage database

interactions. The main collections in the database include clients, and Applications. These collections store information related to registered users, job postings, and job applications respectively.

4. Setup Instructions

• Prerequisites: To run the project successfully, certain prerequisites must be installed on the system. These include Node.js, MongoDB, Git, and a code editor such as Visual Studio Code. After installing the required software, the project repository can be cloned from GitHub. Once the repository is cloned, dependencies for both the frontend and backend must be installed using npm install in their respective directories.

Installation: An environment file (.env) must be created in the server directory to store the MongoDB connection .After configuring the environment variables, both backend and frontend servers can be started locally. The backend typically runs on port 5000, while the frontend runs on port 3000. Once both servers are running, the application can be accessed through a web browser.

5. Folder Structure

• Client: The client folder contains the React frontend application, including components, pages, routing files, and styling files. It handles the user interface and communicates with the backend APIs.
• Server: The server folder contains the backend logic, including models, and the main server configuration file.

6. Running the Application

• Provide commands to start the frontend and backend servers locally.
  o Frontend: npm start in the client directory.
  o Backend: npm start in the server directory.

7. API Documentation

•The backend exposes several RESTful API endpoints to manage authentication, job postings, and applications. The authentication APIs include user registration and login endpoints.
• The job-related APIs allow clients to post new jobs and

existing job listings. Freelancers can use applications to apply for

jobs by submitting proposals. The APIs follow standard HTTP

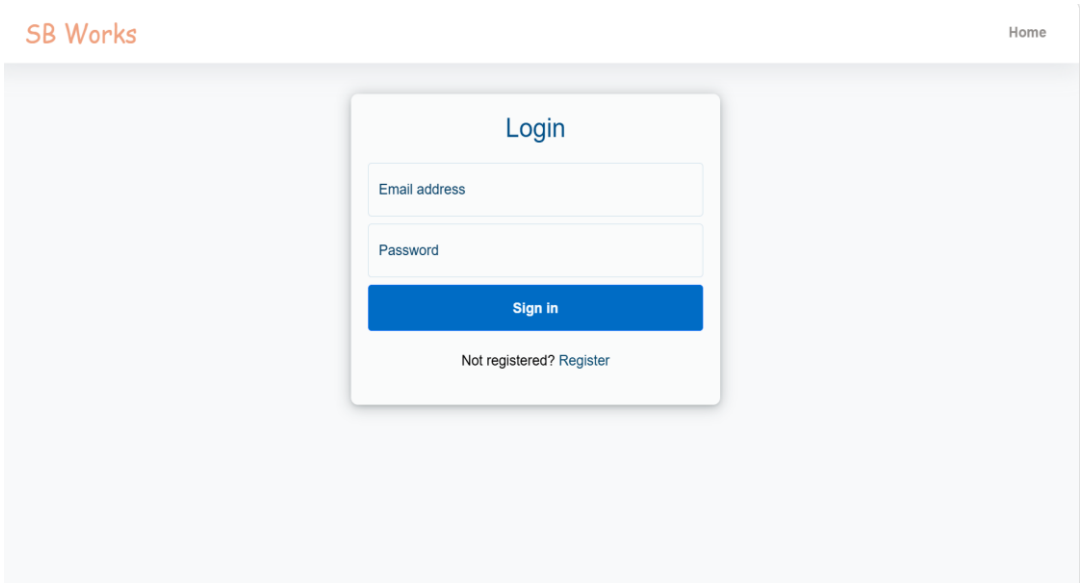
methods such as GET, POST, PUT, and DELETE to perform various

operations.

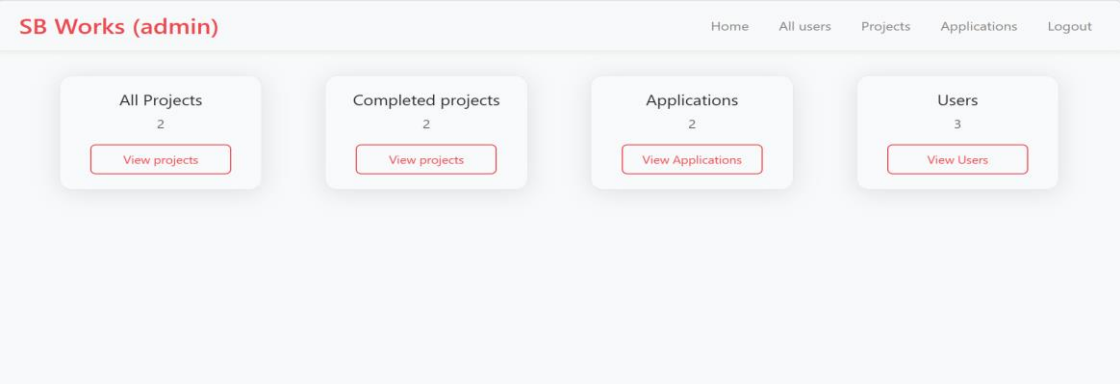
8. User Interface

Registration page:

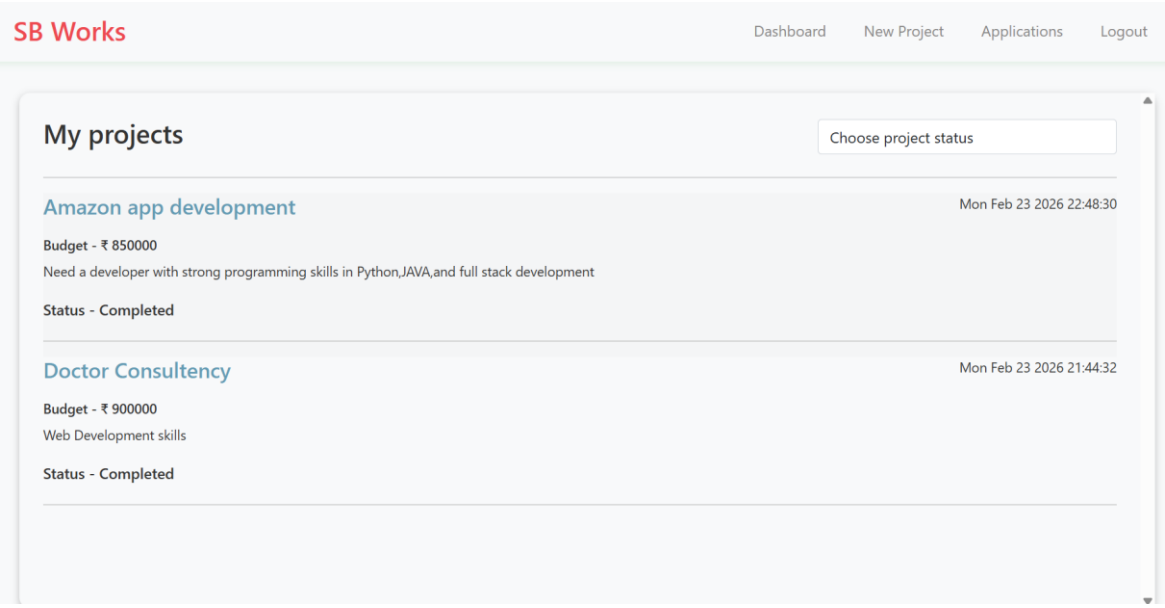
Login page:



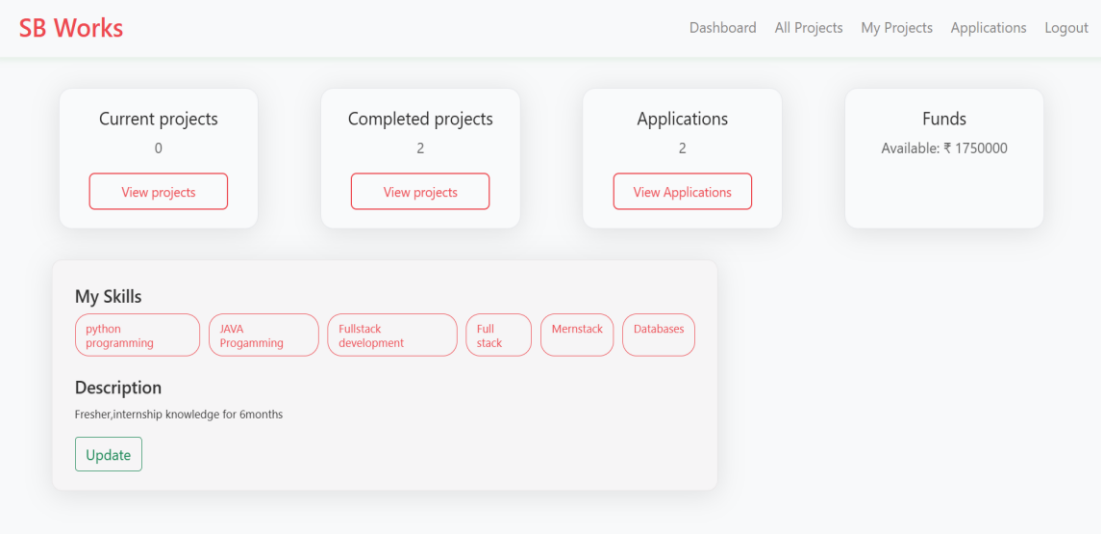
Admin Dashboard:



Client Dashboard:



Freelancer Dashboard:



9. Screenshots or Demo

Demo link:

[Demo link]