Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** Freelance Finder
- **Team Members:** Aryaman Parashar Behera, Disha M, Adithya PR, Swarna Kamalam

2. Project Overview

- **Purpose:** The purpose of the Freelance Finder project is to create a platform where freelancers can showcase their skills and potential clients can find and hire freelancers for various projects. This platform aims to streamline the process of connecting freelancers with clients, providing an easy-to-use interface and robust features for both parties.
- **Features:** User authentication and authorization
- Freelancer profile creation and management
- Job posting and application system
- Search and filter functionality for finding freelancers and jobs
- Messaging system for communication between freelancers and clients
- Review and rating system for freelancers

3. Architecture

- **Frontend:**The frontend is built using React, providing a dynamic and responsive user interface. Components are organized in a modular fashion, and state management is handled using React's Context API and hooks.
- **Backend:** The backend is powered by Node.js and Express.js, creating a RESTful API to handle client requests. It manages user authentication, job postings, profile data, and communication between users.
- **Database:**MongoDB is used as the database, storing user data, job postings, and messages. The database schema includes collections for users, jobs, messages, and reviews, with appropriate relationships between them.

4. Setup Instructions

- **Prerequisites:**Node.js && MongoDB
- **Installation:** Install dependencies for the frontend and backend:
- Navigate to the `client` directory: `cd client`

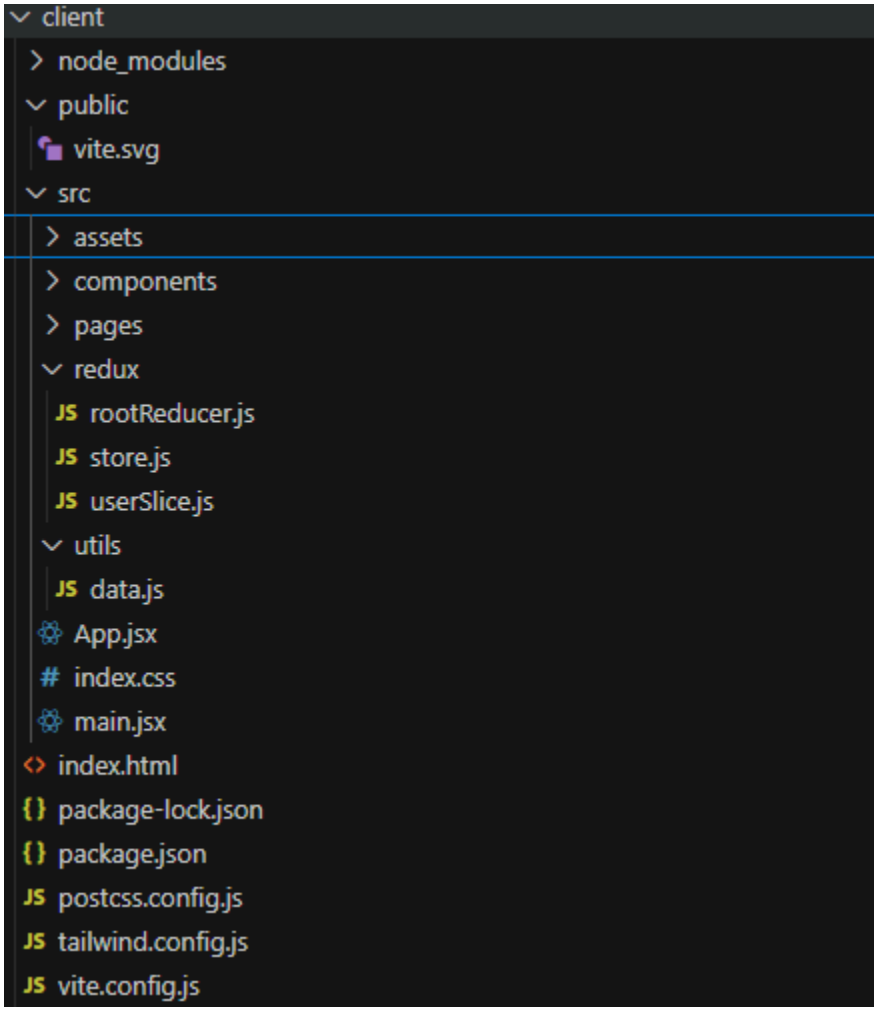
- Install dependencies: `npm install`
- Navigate to the `server` directory: `cd ../server`
- Install dependencies: `npm install`

`SET UP ENVIRONMENT DETAILS:-`

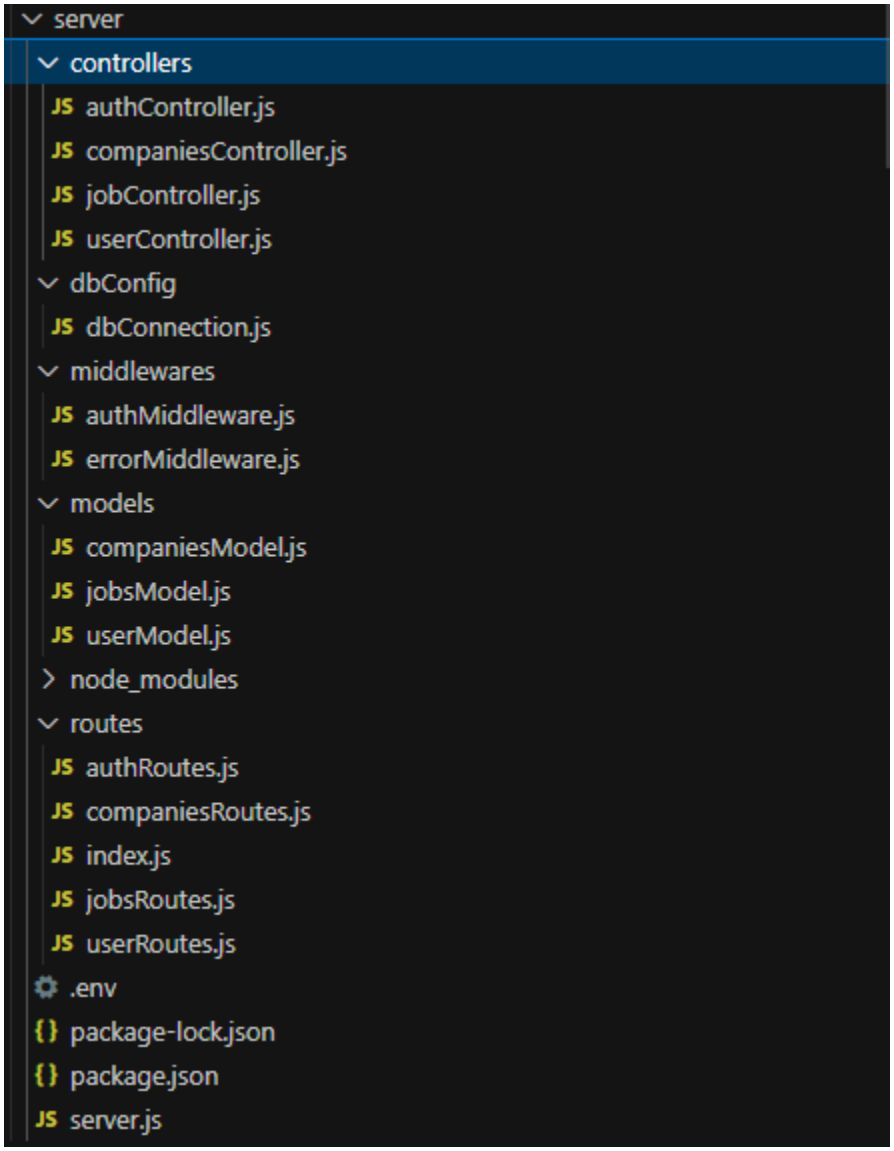
- `MONGO_URI=[your MongoDB connection string]`
- `JWT_SECRET=[your JWT secret key]`

5. Folder Structure

- **Client:**



- **Server:**



6. Running the Application

- Provide commands to start the frontend and backend servers locally.
  - **Frontend:** `npm start` in the client directory.
  - **Backend:** `npm start` in the server directory.

7. API Documentation

User Routes:

- **`POST /api/users/register` - Register a new user**

- **`POST /api/users/login` - Login a user**
- **`GET /api/users/profile` - Get user profile (requires authentication)**

Job Routes:

- **`POST /api/jobs` - Create a new job (requires authentication)**
- **`GET /api/jobs` - Get all jobs**
- **`GET /api/jobs/:id` - Get a specific job**

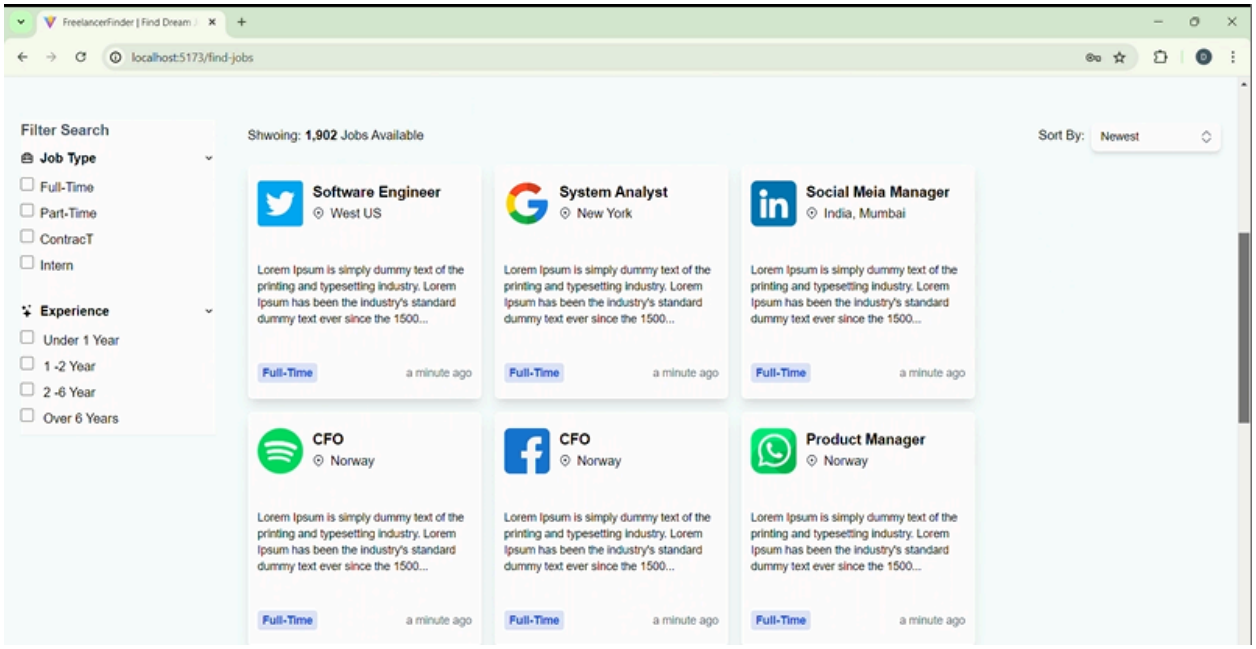
Message Routes:

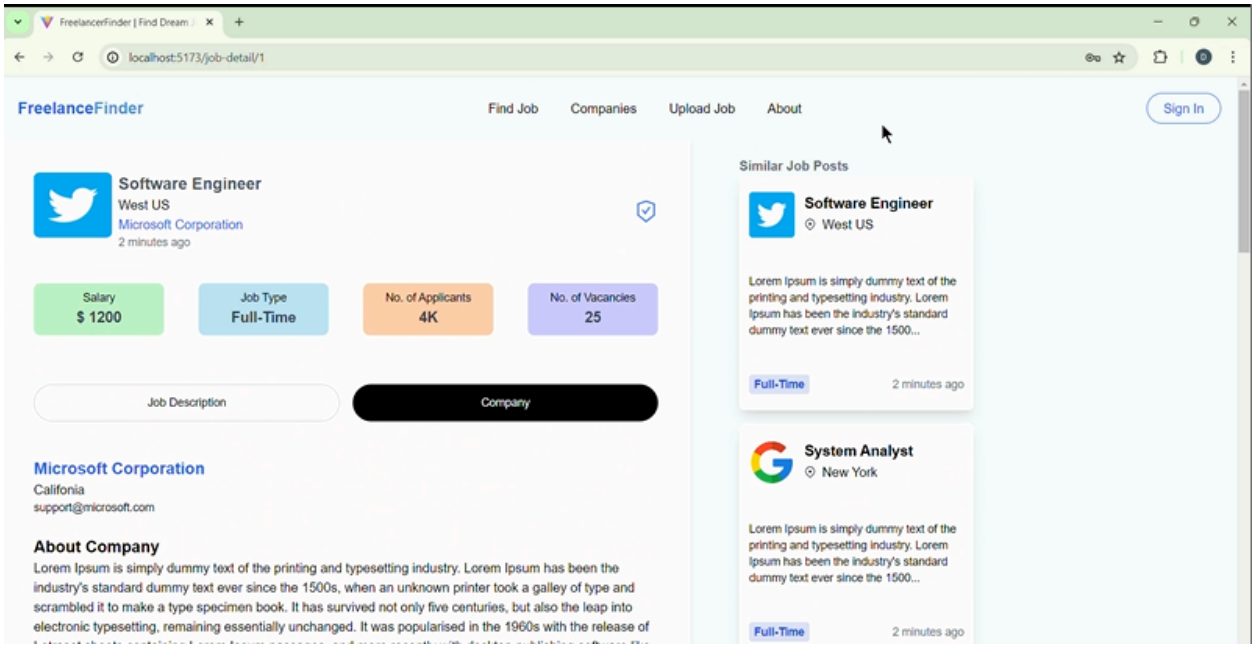
- **`POST /api/messages` - Send a message (requires authentication)**
- **`GET /api/messages` - Get all messages for a user (requires authentication)**

8. Authentication

Authentication is handled using JSON Web Tokens (JWT). Users receive a token upon login, which must be included in the Authorization header for protected routes. The token is used to verify the user's identity and authorize access to protected resources.

9. User Interface





10. Testing

- The testing strategy involves unit tests for individual components and integration tests for API endpoints. Tools used include Jest for JavaScript testing and Supertest for HTTP assertions.

11. Screenshots or Demo

- Demo link : https://youtu.be/1PUTAUeNfsk

12. Known Issues

- Some UI elements may not be fully responsive on smaller devices.

13. Future Enhancements

- Introduce a subscription model for premium features.