Full Stack Development Using MERN Stack

1. Introduction

- **Project Title**: Freelancing Website
- **Team Members**:
  - Dhanush Raja R
  - Shyam Kumar B
  - Sanjeevi prasad R
  - Jerome Joshua M

2. Project Overview

- **Purpose**:

  This website aims to connect freelancers with clients by providing a platform for project postings, applications, and seamless collaboration.
- **Features**:

  Highlight the key features, such as:
  - User registration and authentication
  - Job posting and application management
  - User profiles for freelancers and clients
  - Search and filter functionalities
  - Payment integration

3. Architecture

- **Frontend**:
  - Framework/Library: React.js
  - Key Components: The JobCard component displays job postings in a clean layout.
  - State Management: Redux / Context API / Hooks used, if applicable.
- **Backend**:
  - Technologies: Node.js and Express.js
  - Key Functionality: Describe services like authentication, CRUD operations for job posts, and API design.

- **Database**:
    - Database: MongoDB
    - Schema: Detail schema structure (e.g., users, jobs, applications). Include relationships between collections.

4. Setup Instructions

- **Prerequisites**:
    - Node.js (version X.X or later)
    - MongoDB (local or Atlas setup)
    - IDE (Visual Studio Code or equivalent)

- **Installation**:

1. Clone the repository: git clone [repository URL]
2. Navigate to project folders: cd client and cd server
3. Install dependencies: npm install
4. Set up environment variables

5. Folder Structure

- **Client**:
    - src/components: Reusable React components.
    - src/pages: Screens for different routes.
    - src/store: State management files (if applicable).
    - src/services: API interaction logic.
- **Server**:
    - routes/: Backend API routes.
    - models/: MongoDB schemas and models.
    - controllers/: Business logic for API routes.
    - middleware/: Custom middleware like authentication.

6. Running the Application

- **Frontend**:

  Navigate to the client directory and run:

npm start

- **Backend**:

  Navigate to the server directory and run:

  npm start

7. API Documentation

- Example endpoint documentation:
  - **Endpoint**: /api/jobs
  - **Method**: GET
  - **Description**: Fetch all job postings.
  - **Parameters**: None
  - **Response**:

```
json
[
 {
  "id": "1",
  "title": "Frontend Developer",
  "description": "Build responsive web applications."
 }
]
```

8. Authentication

- **Methodology**:
  - Authentication: JWT tokens.
  - Authorization: Middleware to verify user roles (e.g., admin, client, freelancer).
- **Implementation**:
  - JWT tokens stored in HTTP-only cookies or localStorage.
  - Protected routes using custom middleware in the backend.

9. User Interface

- **Visuals**:

  Provide screenshots or links to UI mockups.

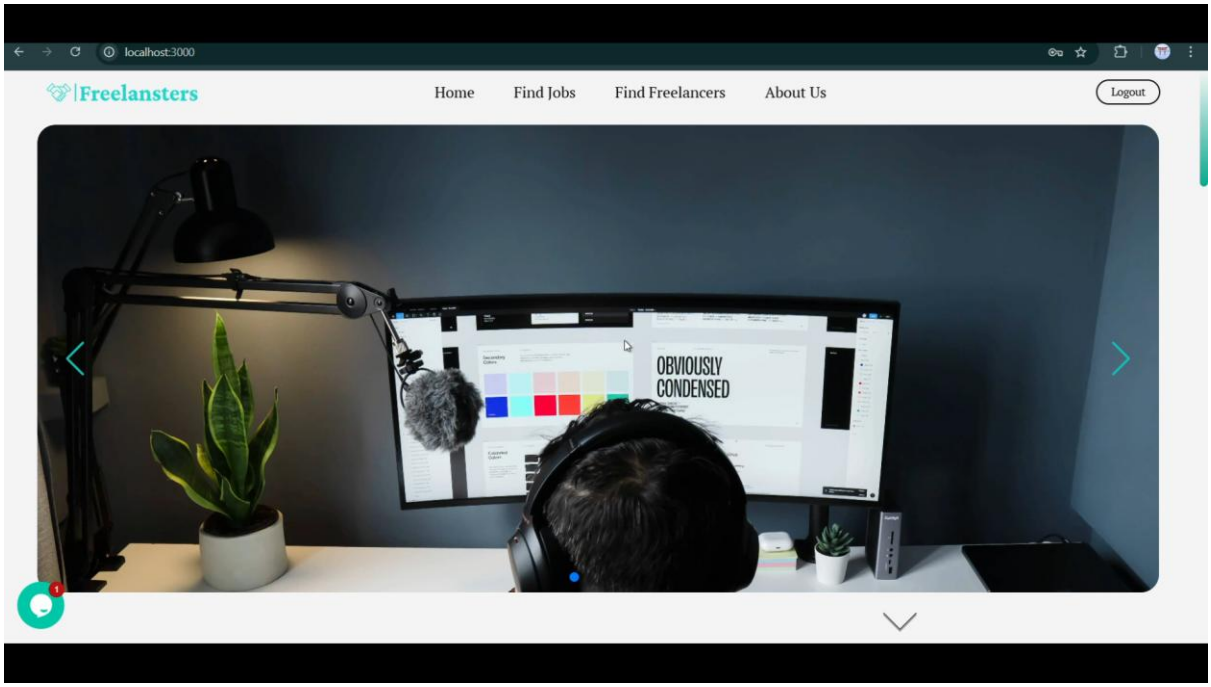
  - Homepage with featured projects
  - Login/Register forms
  - User profile and dashboard

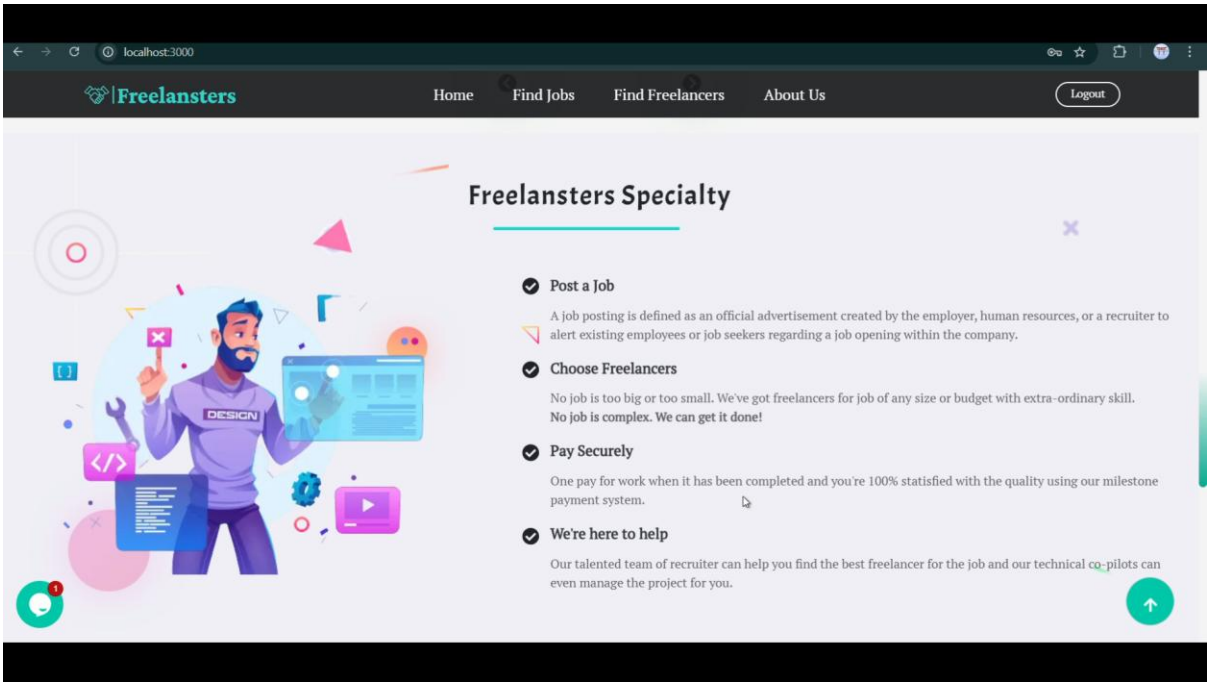
10. Testing

- **Tools Used**:
  - Jest, Mocha, or Cypress for testing.
- **Strategy**:
  - Unit testing for components and backend routes.
  - Integration testing for frontend-backend interactions.

11. Screenshots or Demo

Screenshots:





- **Demo Link**: [https://drive.google.com/file/d/1Dw8U122TDveI0yVVHsosrKX9wrnSbf3R/view?usp=sharing]

12. Known Issues

- Document any unresolved bugs.
    - Search filter delays under high traffic.
    - UI alignment issues on mobile screens.

13. Future Enhancements

- Suggestions for further improvements, such as:
    - Adding a chat system for clients and freelancers.
    - Integration with external payment gateways like Stripe.
    - Advanced analytics dashboard for admin users.