Project Title : Freelancing Application Using MERN

Team Members:

Genga Ganesh L IT-A- 210821205029

Jarom Jerwin P IT-A - 210821205033

Sam Kumar S IT-B - 210821205089

Sanjay T IT-B – 210821205093

Purpose:

The application connects freelancers and clients, allowing clients to post projects and freelancers to bid or apply for them. It manages contracts, tracks project progress, and facilitates payments.

Features:

Client Features:

- Post projects with requirements, budgets, and deadlines.
- View freelancer profiles and applications.
- Track project progress.
- Manage payments securely.

Freelancer Features:

- Create a profile with skills, portfolio, and availability.
- Browse and apply for projects.
- Manage project deliverables and milestones.
- Track payment history.

Architecture:

Frontend (React):

- **React Router:** Manage navigation between views (e.g., home, profile, projects).
- **State Management:** Context API or Redux for managing authentication, user profiles, and projects.
- **Styling:** Material-UI or styled-components for a modern, responsive UI.
- **Testing:** Jest and React Testing Library.

Backend (Node.js + Express.js):

- **Authentication:** JWT-based user authentication with bcrypt for password security.
- **RESTful API:** CRUD operations for users, projects, applications, and payments.
- **Database:** MongoDB for flexible schema management (e.g., user profiles, projects).
- **Socket.IO:** Enable real-time chat functionality.

Database Schema:

Users Collection:

- **Fields:** name, email, password, role (freelancer/client), profile, ratings.

Projects Collection:

- **Fields:** title, description, clientId, budget, status (open/closed), bids.

Applications Collection:

- **Fields:** freelancerId, projectId, proposal, status (pending/accepted).

Payments Collection:

- **Fields:** amount, projectId, freelancerId, status (completed/pending).

Messages Collection:

- **Fields:** senderId, receiverId, projectId, message, timestamp.

Folder Structure:

Frontend:

bash

/client

├── /public

│ ├── index.html

│ └── assets (e.g., logos, images)

├── /src

├── /components (UI components like Navbar, Footer)

├── /pages (Dashboard, Project Details, Applications)

├── /services (API handlers)

├── /context (Global state management)

├── App.js

├── index.js

Backend:

bash

/server

├── /config (Database connection, environment variables)

├── /controllers (Business logic for projects, users, etc.)

├── /models (Mongoose schemas)

├── /routes (API endpoints)

├── /middleware (Authentication, error handling)

├── index.js (Entry point)

API Documentation:

Users

POST /api/auth/register - Register a new user.

POST /api/auth/login - Authenticate a user and return a JWT.

Projects

GET /api/projects - Fetch all projects.

POST /api/projects - Create a new project (Client only).

GET /api/projects/

- Fetch project details by ID.

Applications

POST /api/applications - Submit an application for a project.

GET /api/applications/

- Fetch all applications for a project.

Payments

POST /api/payments - Initiate a payment.

GET /api/payments/

- Fetch payment status for a project.

Messages

POST /api/messages - Send a new message.

GET /api/messages/

- Fetch messages for a project.

Prerequisites

Node.js: Install the latest version from Node.js Official Website.

MongoDB: Set up a MongoDB instance:

Use MongoDB Atlas (cloud-based) or install MongoDB Community Edition locally.

Git: For cloning the repository.

Code Editor: Preferably VS Code for development.

Postman/Browser: For testing API endpoints.

Authentication in the Freelancing Application

User Roles:

- Supports freelancers, clients, and admins with role-based access control (RBAC).
- Different routes and functionalities are accessible based on user roles.

JWT Authentication:

- JSON Web Tokens (JWT) are used for secure authentication.
- A token is generated during login and sent to the client.
- The client stores the token (e.g., in local storage) and sends it with each request in the Authorization header.

Secure Passwords:

- User passwords are hashed using bcrypt before storage in the database.
- Prevents plaintext passwords from being exposed even if the database is compromised.

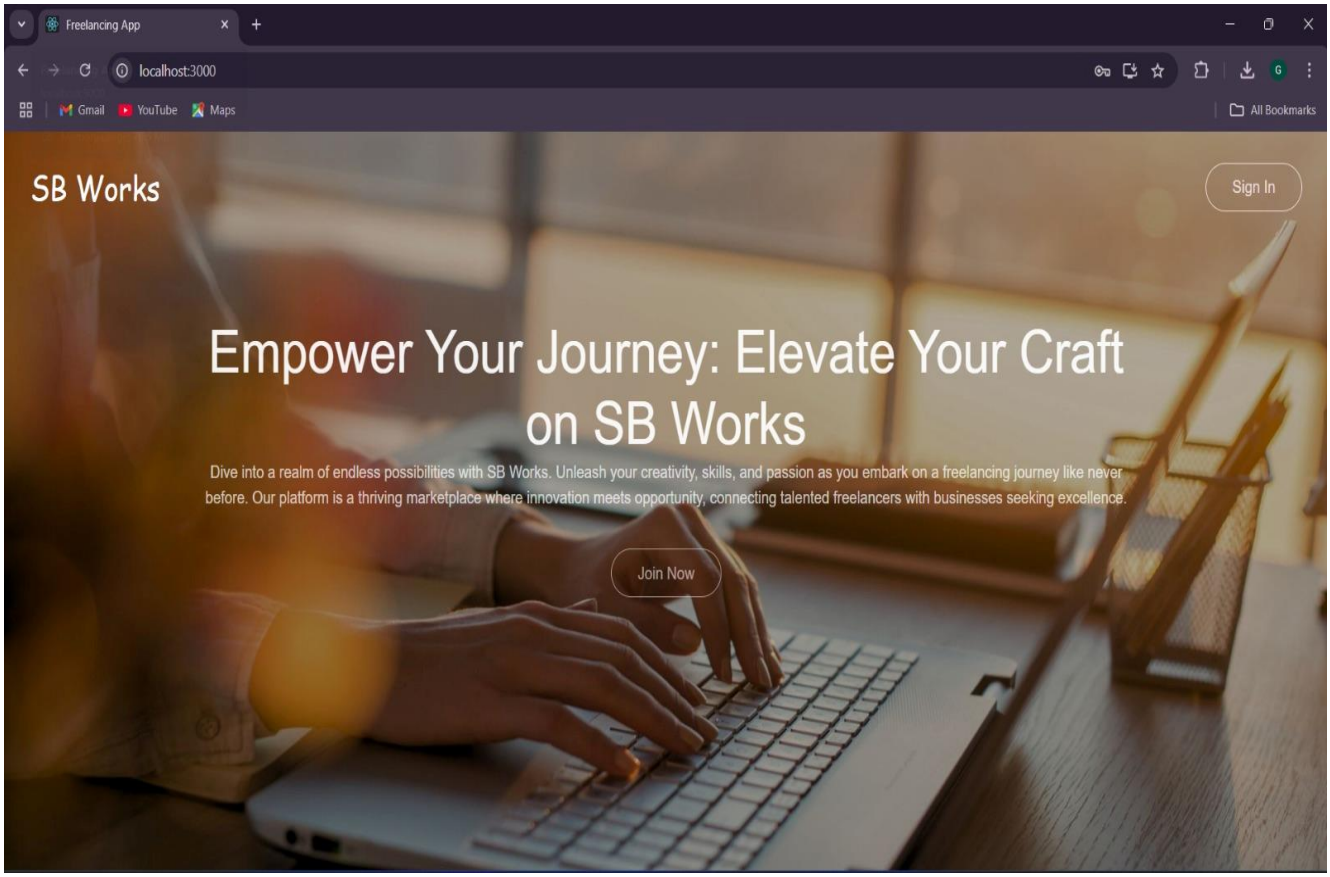
Protected Routes:

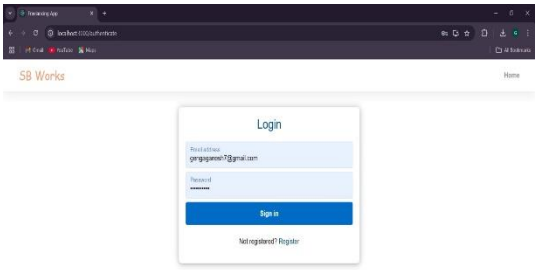
- Certain routes (e.g., project posting, application submission) are protected and require a valid JWT.
- Middleware verifies the token before granting access to these routes.

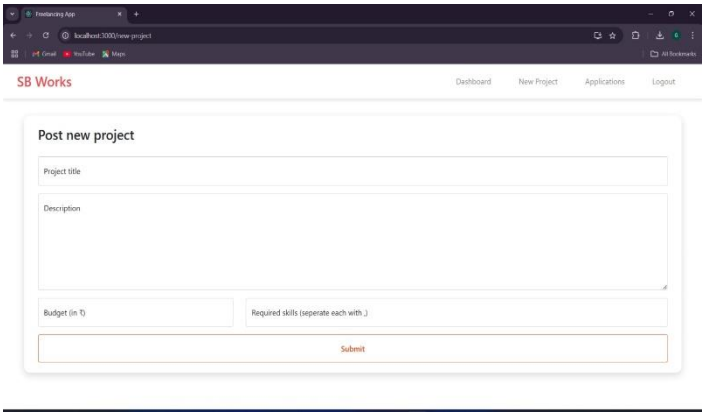
Login & Registration:

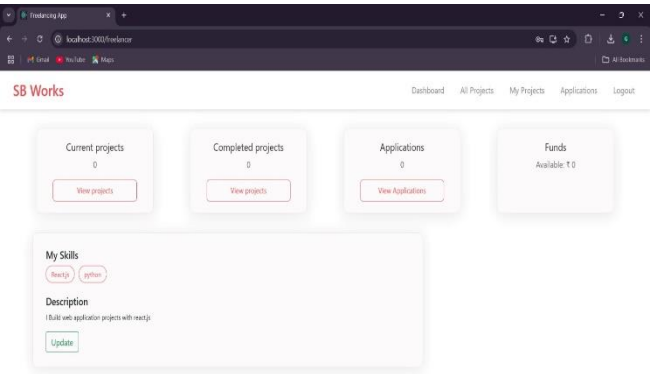
- Registration: Users provide their details (e.g., name, email, password, role).
- Login: Users authenticate with an email and password.
- A valid login returns a JWT for future API requests.

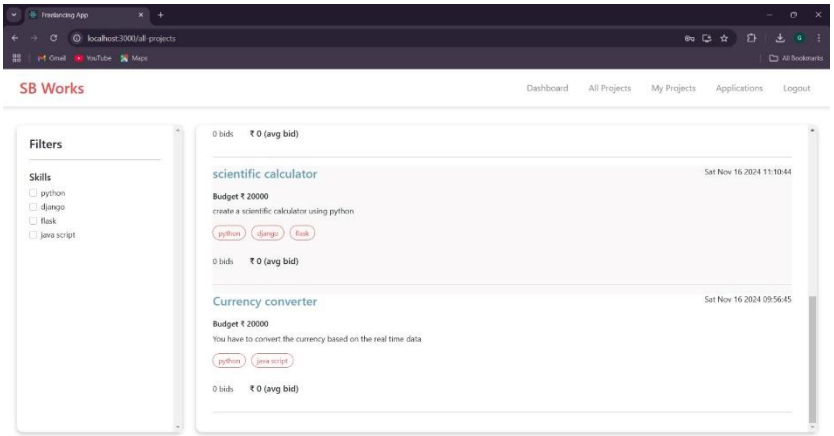
User Interface:

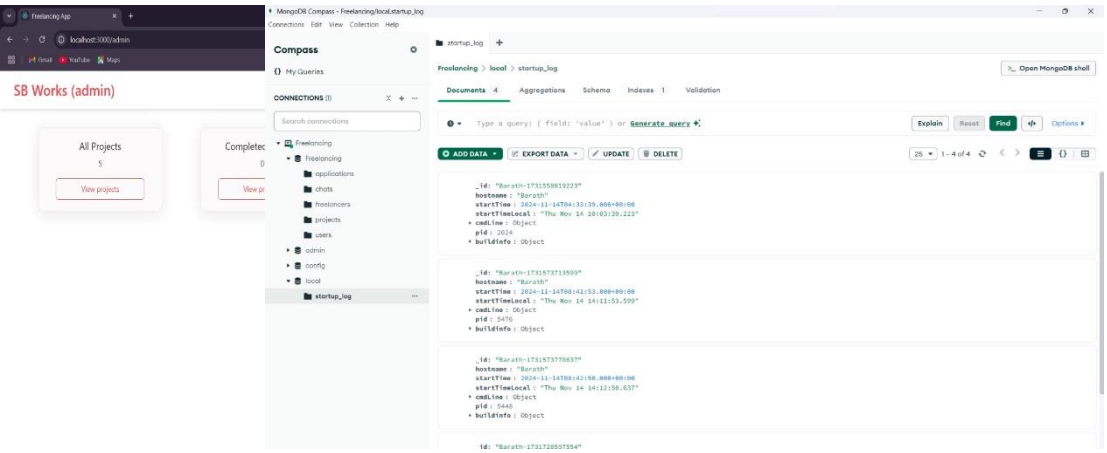












Running the Application:

Start Backend First:

Navigate to the server directory and run:

npm start

This starts the backend server on http://localhost:5000 (default port).

Start Frontend Next:

Navigate to the client directory and run:

npm start

The React app will run on http://localhost:3000.

Database Connectivity:

- Ensure MongoDB is running locally or your connection string to MongoDB Atlas is valid in the .env file.

Access the Application:

- Open http://localhost:3000 in your browser to interact with the application.

Testing API Endpoints:

- Use tools like Postman to test backend APIs (e.g., http://localhost:5000/api/projects).

Real-Time Features:

- Ensure both the backend and frontend are running to test features like real-time chat and notifications.

Hot Reloading:

- Both frontend and backend support hot reloading for changes to reflect instantly during development.

Environment Variables:

- Check .env files for the backend and frontend to ensure the configurations match your setup.

Error Troubleshooting:

- If any service fails, check terminal logs or browser console for specific errors.

- Verify package installations with npm install in the respective directories.

Stopping the Application:

- Press Ctrl+C in the terminal where the servers are running to stop them.

Testing in the Freelancing Application:

Testing Types:

- Unit Testing: Validate individual components, functions, or modules in isolation.
- Integration Testing: Test interactions between different modules (e.g., API endpoints and database operations).
- End-to-End (E2E) Testing: Simulate real-world scenarios to test the entire user flow.
- Manual Testing: Identify edge cases, UI responsiveness, and usability issues.
- Frontend Testing:
- Tools: Use Jest and React Testing Library for testing React components.

Focus Areas:

- Rendering of components (e.g., Navbar, ProjectList).
- Validation of user inputs in forms (e.g., login, registration).
- Testing API interactions with mocked responses.

Future Enhancements for the Freelancing Application:

User Account Management:

- Enhanced profiles for freelancers and clients, including social media links and certifications.
- Detailed dashboards showing project statistics, payment history, and feedback.

Advanced Search & Filtering:

- Implement filters for budget range, project type, freelancer skills, and deadlines.
- Add voice and image search capabilities for specific project needs.

Recommendation System:

- AI-powered suggestions for projects based on a freelancer’s skills and past activities.
- Client recommendations for freelancers based on project history.

Subscription Plans:

- Premium subscriptions for clients and freelancers with benefits like prioritized listing, advanced analytics, or reduced service fees.

Mobile Application:

- Develop native Android and iOS apps for better accessibility.
- Offline capabilities for viewing project details or drafts.

Multi-Language & Currency Support:

- Add localization to support multiple languages for a global audience.
- Allow transactions in multiple currencies with real-time exchange rates.

Improved Payment System:

- Integration with additional payment gateways (e.g., Google Pay, PayPal, cryptocurrency).
- Introduce escrow features with more flexibility in milestone payment releases.

Real-Time Collaboration Tools:

- File sharing, live document editing, and project timeline tracking.
- Video or voice calls directly within the platform.

Gamification Features:

- Add badges, points, or levels to incentivize user engagement and quality contributions.

Enhanced Admin Dashboard:

- Advanced analytics with detailed insights into platform performance.
- Tools for handling disputes, monitoring transactions, and ensuring user compliance.