FreelanceFinder: Discovering Opportunities, Unlocking Potential

Team Id: LTVIP2025TMID55793

Name: Kolapalli Gnana Pujitha

Introduction:

Welcome to FreelanceFinder, your dedicated platform for exploring freelancing opportunities and unlocking your full potential in the digital marketplace. Whether you're a budding freelancer looking for your first project or a seasoned expert seeking new challenges, FreelanceFinder connects talent with demand. Our platform is designed with a user-centric approach to empower individuals and businesses through seamless connections and transparent workflows.

Description:

FreelanceFinder isn't just a freelancing portal; it's a comprehensive ecosystem created to revolutionize the way freelancers and clients collaborate. The platform offers a wide array of professional opportunities across categories such as web development, content writing, graphic design, marketing, data science, and more.

Our intuitive interface ensures that users can easily create profiles, browse projects, submit proposals, and get hired quickly. With advanced search filters, recommendation algorithms, secure payment systems, and real-time communication tools, FreelanceFinder removes barriers and enhances productivity for both freelancers and clients.

Whether you're building your portfolio or sourcing top talent for your startup, FreelanceFinder offers the resources, tools, and support you need to thrive.

Scenario Based Case Study:

Meet Riya, an aspiring graphic designer.

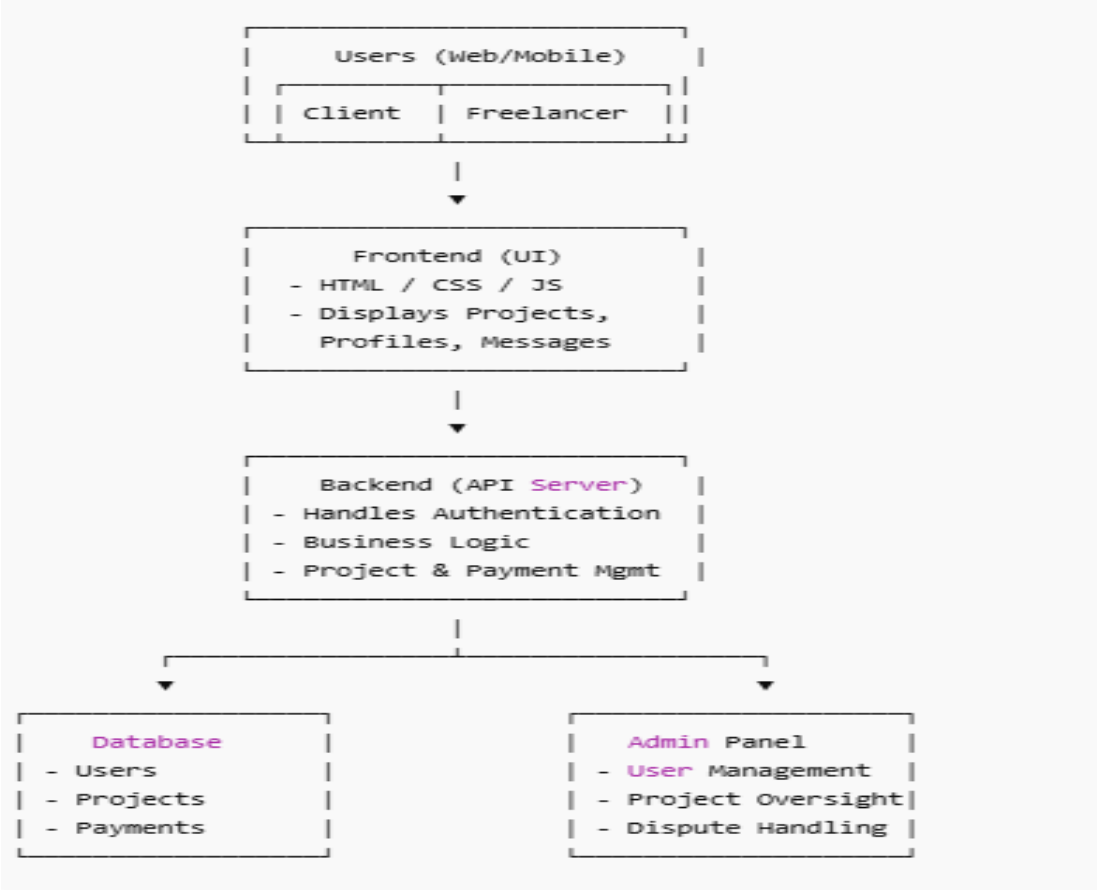
Riya is a creative graduate with a flair for visual storytelling but lacks industry connections to get started. She signs up on FreelanceFinder, completes her profile, and uploads her portfolio.

- **User Registration and Authentication:** Riya registers on the platform, verifies her identity, and sets up her professional profile.
- **Project Browsing:** She explores available gigs using category filters and tags to find design-related projects.
- **Smart Suggestions:** Based on her skills and portfolio, FreelanceFinder recommends projects aligned with her interests.
- **Proposal Submission:** She sends customized proposals and eventually lands her first freelance gig designing a logo for a startup.

- **Secure Payments:** After completing the task, she receives payment securely through FreelanceFinder's escrow-based payment system.
- **Rating & Review:** Riya earns a 5-star rating and a positive review, helping her attract more clients.

Thanks to FreelanceFinder, Riya builds a sustainable freelance career doing what she loves, without worrying about delayed payments or finding clients.

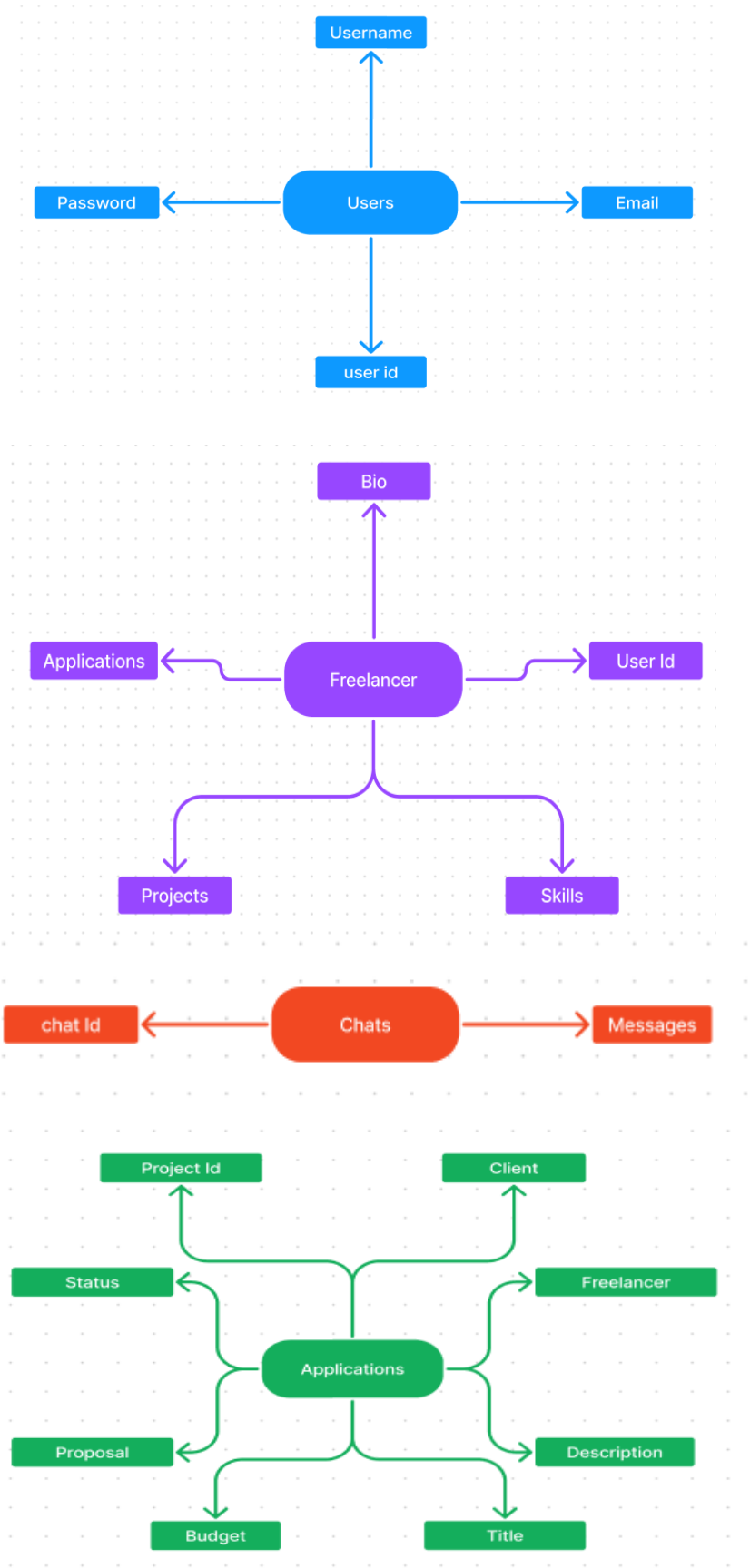
Technical Architecture:-

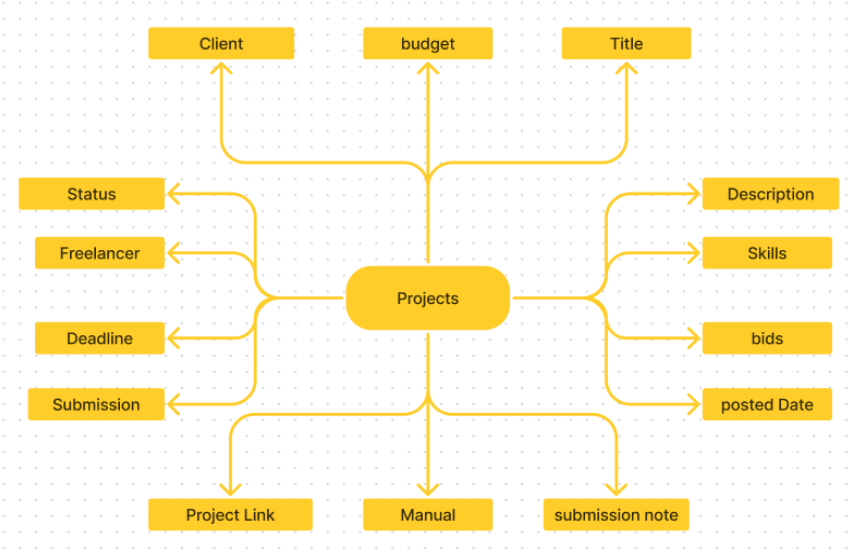


FreelanceFinder follows a modern web architecture based on the MERN stack (MongoDB, Express.js, React.js, Node.js).

- **Frontend:** Built using React.js for a dynamic and responsive user interface.
- **Backend:** Node.js and Express.js handle API logic, user authentication, and job management.
- **Database:** MongoDB stores user data, project listings, messages, and reviews.
- **Communication:** RESTful APIs connect frontend and backend, with WebSocket used for real-time messaging.
- **Security:** JWT-based authentication, encrypted password storage, and secure file uploads.

ER-Diagram:


Username
Password
Users
Email
user id
Bio
Applications
Freelancer
User Id
Projects
Skills
chat Id
Chats
Messages
Project Id
Client
Status
Freelancer
Applications
Proposal
Description
Budget
Title



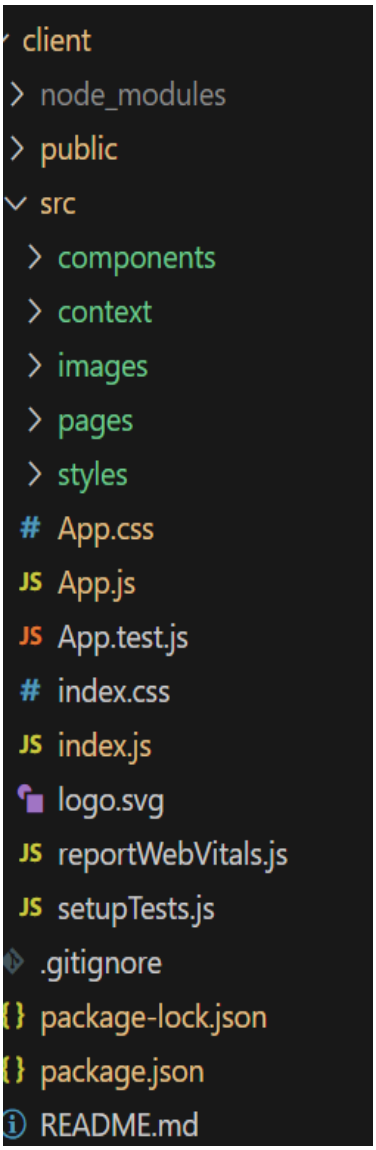
ER-Diagram Overview:

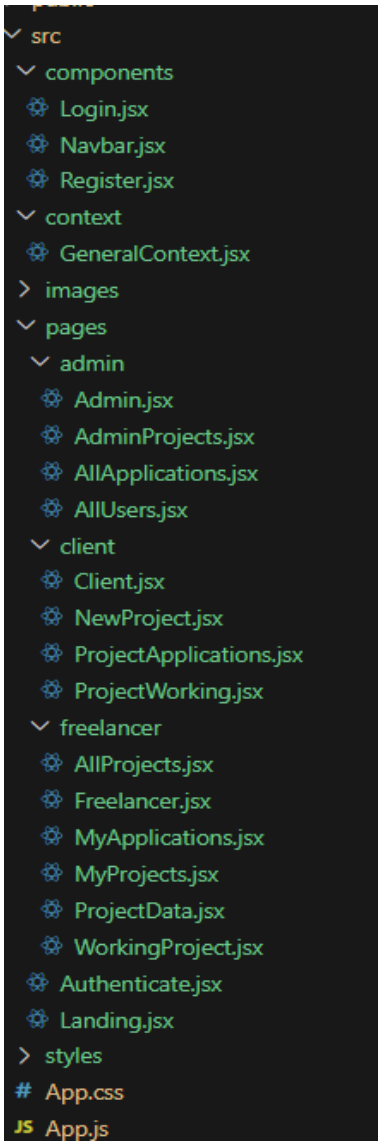
- Users
- Freelancer
- Chats
- Applications
- Projects

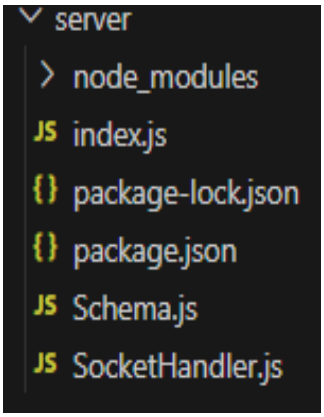
Key Features:

- **Profile Management:** Freelancers and clients can customize and manage their profiles**.**
- **Project Listings:** Clients post projects with detailed requirements, budgets, and deadlines.
- **Smart Matching:** Recommendation engine matches freelancers to relevant jobs.
- **Secure Payments:** Escrow-based payment protection ensures freelancers get paid and clients get quality work.
- **Messaging System:** Real-time chat allows smooth communication and collaboration.
- **Dashboard Analytics:** Clients and freelancers get insights into earnings, performance, and proposals.
- **Admin Panel:** Allows platform moderators to manage users, content, disputes, and reports.

PROJECT STRUCTURE







PRE REQUISITES:

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

Download: https://nodejs.org/en/download/

Installation instructions: https://nodejs.org/en/download/package-manager/

Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.
Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.
Installation: Open your command prompt or terminal and run the following command:
npm install express

MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.
Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community
Installation instructions: https://docs.mongodb.com/manual/installation/

React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.
Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Express Js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Framework: Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
Git: Download and installation instructions can be found at: https://git-scm.com/downloads

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• **Visual Studio Code:** Download from https://code.visualstudio.com/download

Install Dependencies:

• Navigate into the cloned repository directory:
cd freelancer-app-MERN

• Install the required dependencies by running the following commands:

cd client
npm install
../cd server
npm install

Start the Development Server:
• To start the development server, execute the following command:
npm start
• This app will be accessible at http://localhost:3000

You have successfully installed and set up the SB application on your local machine. You can now proceed with further customization, development, and testing as needed.

Roles and Responsibility:

Freelancer Responsibilities:

• Project Submission: Freelancers are responsible for submitting completed and high-quality work for the assigned projects through the platform.
• Compliance: Ensure that the submitted work adheres to client requirements, industry standards, and any specific guidelines outlined by the platform.
• Effective Communication: Actively engage in communication with clients, promptly responding to messages, asking clarifying questions, and providing updates on the project progress.
• Time Management: Manage time effectively to meet project deadlines and deliver work in a timely manner.
• Professionalism: Conduct oneself professionally by maintaining a respectful and cooperative attitude with clients and fellow freelancers.
• Quality Assurance: Deliver work that is accurate, well-executed, and free from errors to maintain client satisfaction.

Client Responsibilities:

• Clear Project Description: Provide a detailed and comprehensive project description, including deliverables, desired outcomes, and any specific requirements.

• Timely Communication: Respond promptly to freelancer inquiries, providing necessary information and feedback in a timely manner.
• Payment Obligations: Fulfill the agreed-upon payment terms promptly and fairly upon satisfactory completion of the project.
• Feedback and Evaluation: Provide constructive feedback and evaluate the freelancer's performance, helping them improve and providing valuable insights.

Admin Responsibilities:

- Data Oversight: As an admin, one of your key responsibilities is to monitor and ensure the integrity and security of all data on the platform
- Policy Enforcement: Admins play a crucial role in enforcing platform policies, guidelines, and ethical standards.
- Conflict Resolution: In the event of disputes or issues within the community, it is the admin's responsibility to address them promptly and impartially
- User Support and Communication: Admins should provide support and guidance to users on the platform
- Platform Maintenance and Improvement: Admins are responsible for the overall maintenance and improvement of the research platform.

PROJECT FLOW:-

Before starting to work on this project, let's see the demo.

Demo link:-
https://drive.google.com/file/d/18-gnCVAZdojBhl7QM0_yShH1JL5A6v32/view?usp=drive_link

Use the code in:-

https://drive.google.com/drive/folders/1gSc8RWmM7yqdvYh525n5DoLjZxS_N-F4

or follow the videos below for better understanding.

Milestone 1: Project Setup and Configuration:

1. Install required tools and software:

- Node.js.
- MongoDB.
- Create-react-app.

2. Create project folders and files:

- Client folders.
- Server folders.

3. Install Packages:

Frontend npm Packages

- Node

- React
- Bootstrap.

Backend npm Packages

- Express.
- Mongoose.
- Body-parser
- Cors.

Milestone 2: Backend Development:

- **Setup express server**
    1. Create index.js file in the server (backend folder).
    2. Create a .env file and define port number to access it globally.
    3. Configure the server by adding cors, body-parser.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Define API Routes:**
    - Create separate route files for different API functionalities such as users orders, and authentication.
    - Define the necessary routes for listing products, handling user registration and login,managing orders, etc.
    - Implement route handlers using Express.js to handle requests and interact with the database.

- **Implement Data Models:**
    - Define Mongoose schemas for the different data entities like products, users, and orders.
    - Create corresponding Mongoose models to interact with the MongoDB database.
    - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Error Handling:**
    - Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

Milestone 3: Database:

1. Configure MongoDB:

- Install Mongoose.
- Create database connection.
- Create Schemas & Models.

2. Connect database to backend:

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.

```
const mongoose = require("mongoose");

const db= 'mongodb://127.0.0.1:27017/grocery'
// Connect to MongoDB using the connection string

mongoose.connect(db, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log(`Connection successful`);
}).catch((e) => {
  console.log(`No connection: ${e}`);
});
```

3. Configure Schema:

Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The schemas are looks like for the Application.

```
import mongoose, { Schema, mongo } from "mongoose";

const userSchema = mongoose.Schema({
    username: {
        type: String,
        require: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    usertype:{
        type: String,
        require: true
    }
})

const freelancerSchema = mongoose.Schema({
```

```
  userId: String,
  skills: {
    type: Array,
    default: []
  },
  description: {
    type: String,
    default: ""
  },
  currentProjects: {
    type: Array,
    default: []
  },
  completedProjects: {
    type: Array,
    default: []
  },
  applications: {
    type: Array,
    default: []
  },
  funds: {
    type: Number,
    default: 0
  },
})

const projectSchema = mongoose.Schema({
  clientId: String,
  clientName: String,
  clientEmail: String,
  title: String,
  description: String,
  budget: Number,
  skills: Array,
  bids: Array,
  bidAmounts: Array,
  postedDate: String,
  status: {
    type: String,
    default: "Available"
  },
  freelancerId: String,
  freelancerName: String,
  deadline: String,
  submission: {
    type: Boolean,
    default: false
  },
  submissionAccepted: {
    type: Boolean,
    default: false
  },
  projectLink: {
    type: String,
    default: ""
  },
  manulaLink: {
    type: String,
    default: ""
  },
  submissionDescription: {
    type: String,
    default: ""
```

```
    },
})

const applicationSchema = mongoose.Schema({

    projectId: String,
    clientId: String,
    clientName: String,
    clientEmail: String,
    freelancerId: String,
    freelancerName: String,
    freelancerEmail: String,
    freelancerSkills: Array,
    title: String,
    description: String,
    budget: Number,
    requiredSkills: Array,
    proposal: String,
    bidAmount: Number,
    estimatedTime: Number,
    status: {
        type: String,
        default: "Pending"
    }
})

const chatSchema = mongoose.Schema({
    _id: {
        type: String,
        require: true
    },
    messages: {
        type: Array
    }
})

export const User = mongoose.model('users', userSchema);
export const Freelancer = mongoose.model('freelancer', freelancerSchema);
export const Project = mongoose.model('projects', projectSchema);
export const Application = mongoose.model('applications', applicationSchema);
export const Chat = mongoose.model('chats', chatSchema);
```

Milestone 4: Frontend Development:

1. Setup React Application:

- Create React application.
- Configure Routing.
- Install required libraries.

2. Design UI components:

- Create Components.
- Implement layout and styling.
- Add navigation.

3. Implement frontend logic:

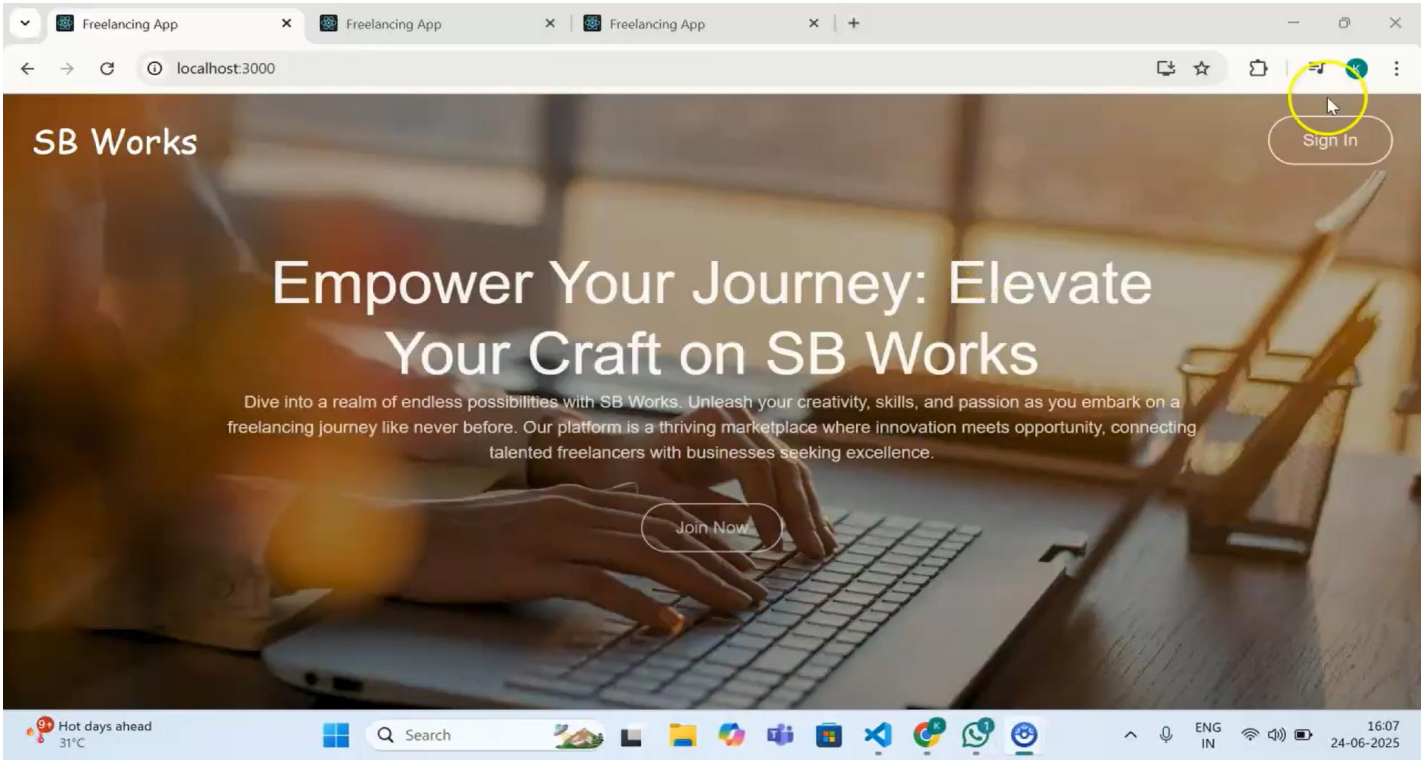
• Integration with API endpoints.

• Implement data binding.

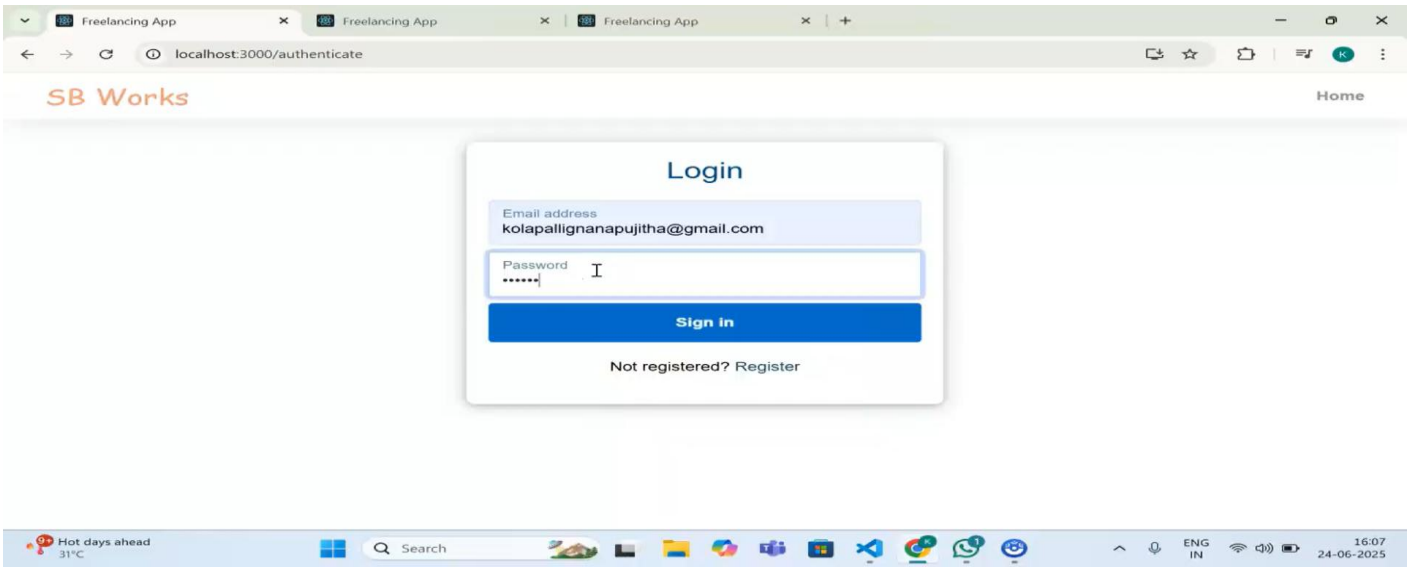
Milestone 5: Project Implementation:

Finally, after finishing coding the projects we run the whole project to test it's working process and look for bugs. Now, let's have a final look at the working of our Darshan Ease.

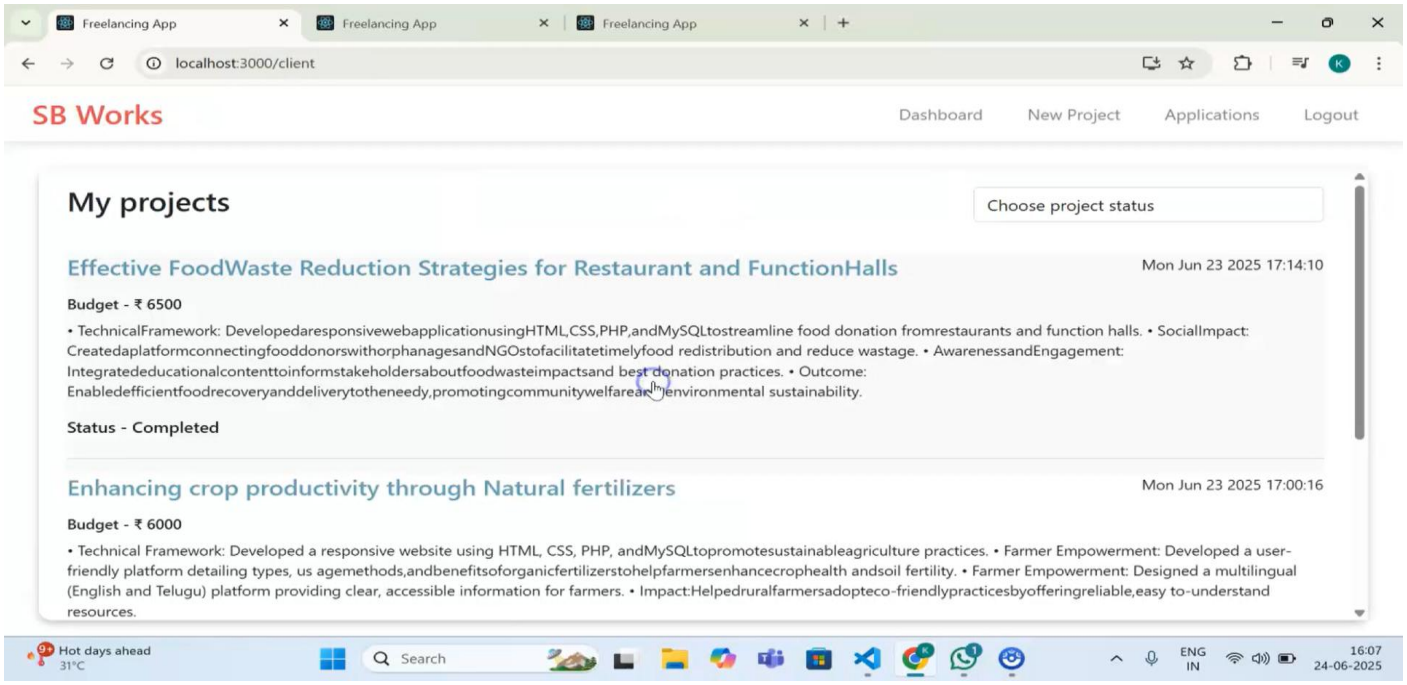
Landing page:-



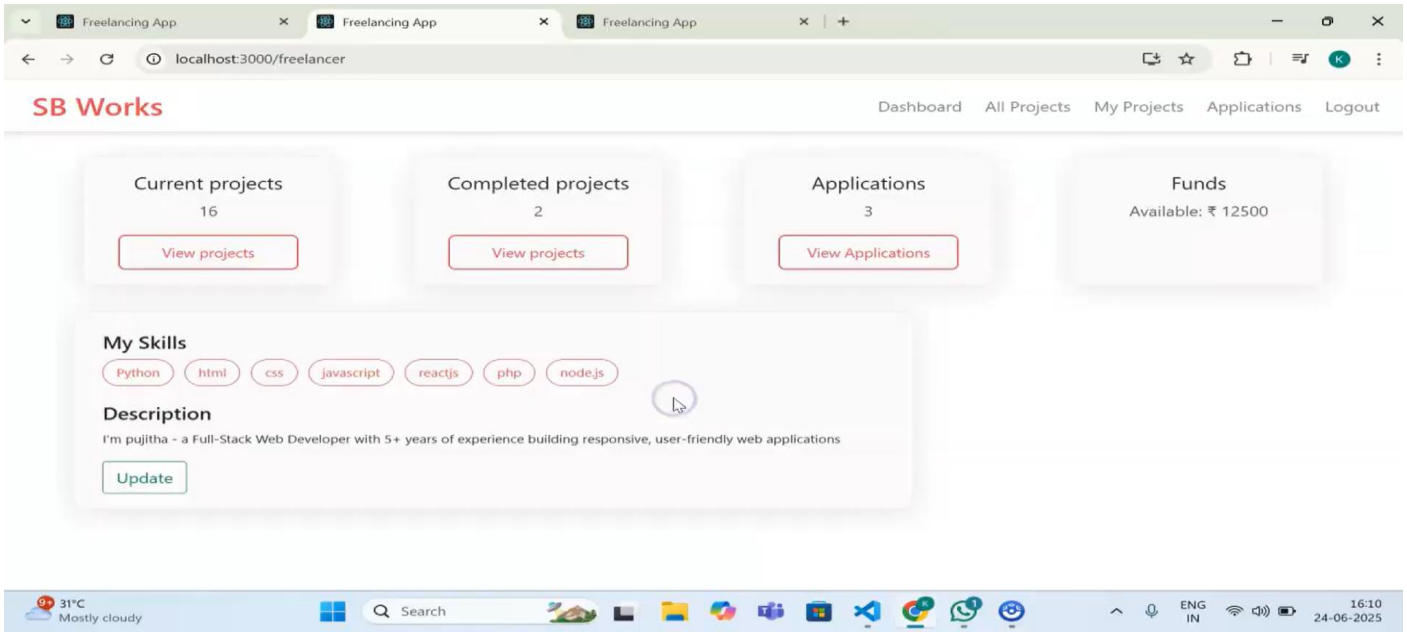
Login Page:-



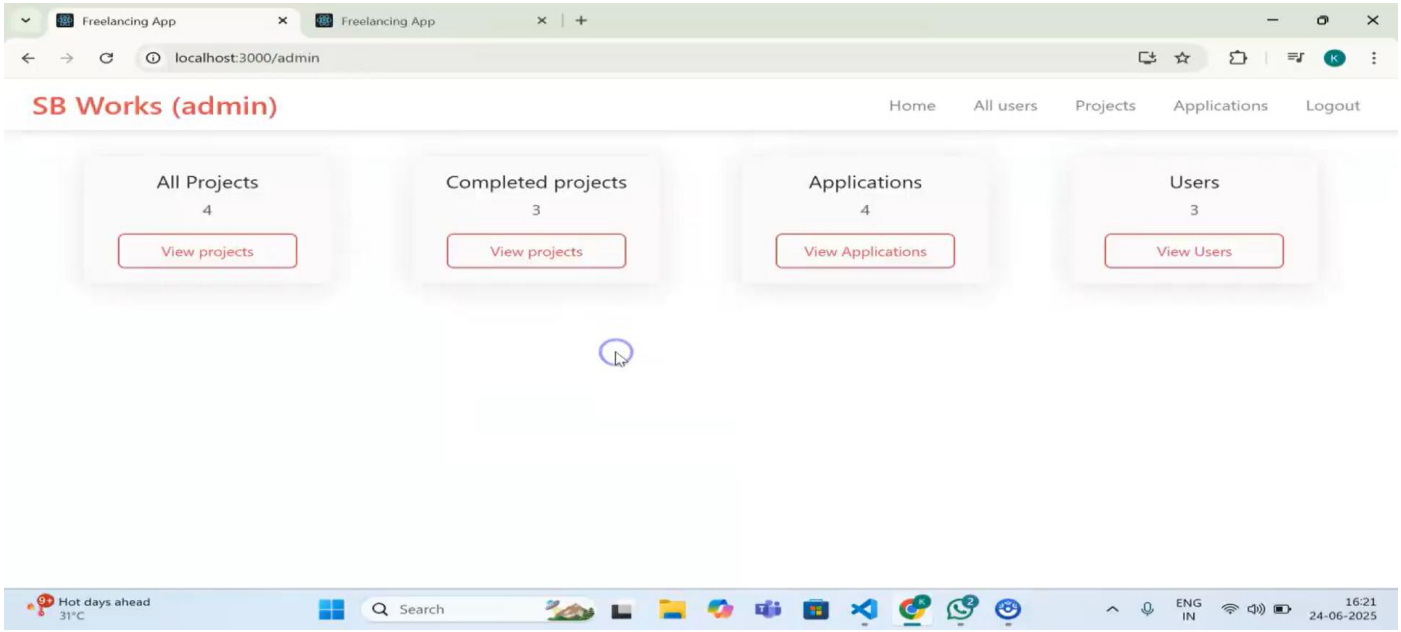
Client page:



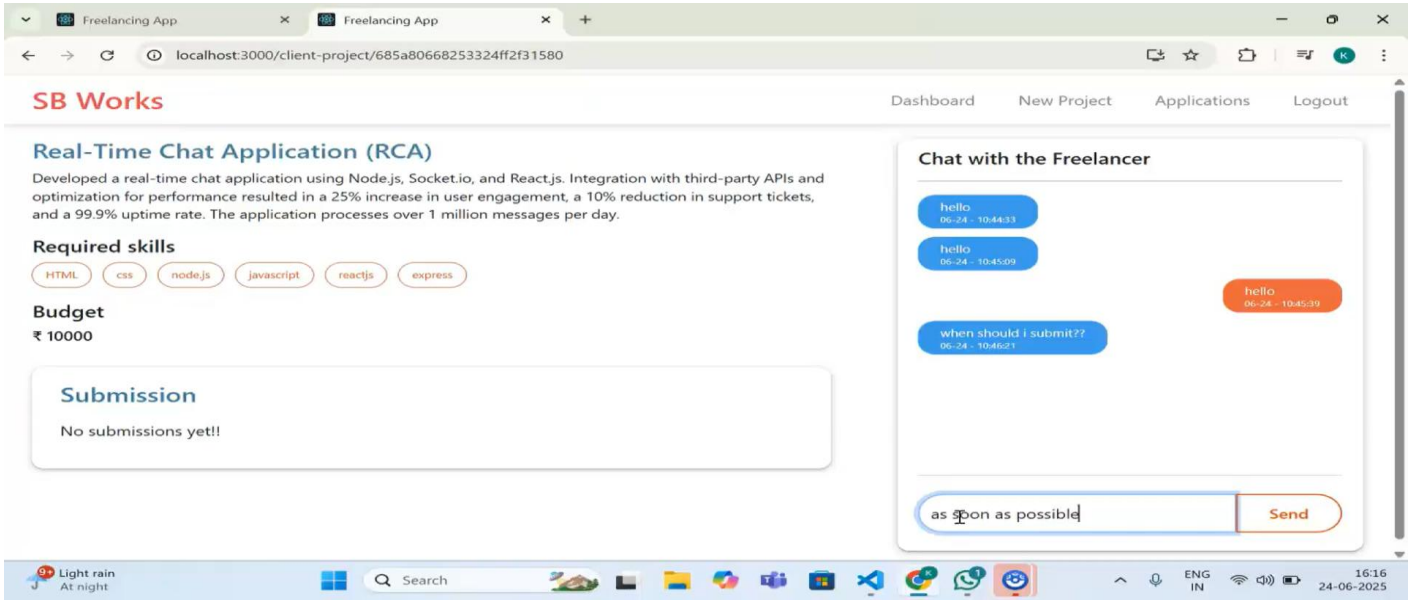
Freelancer page:



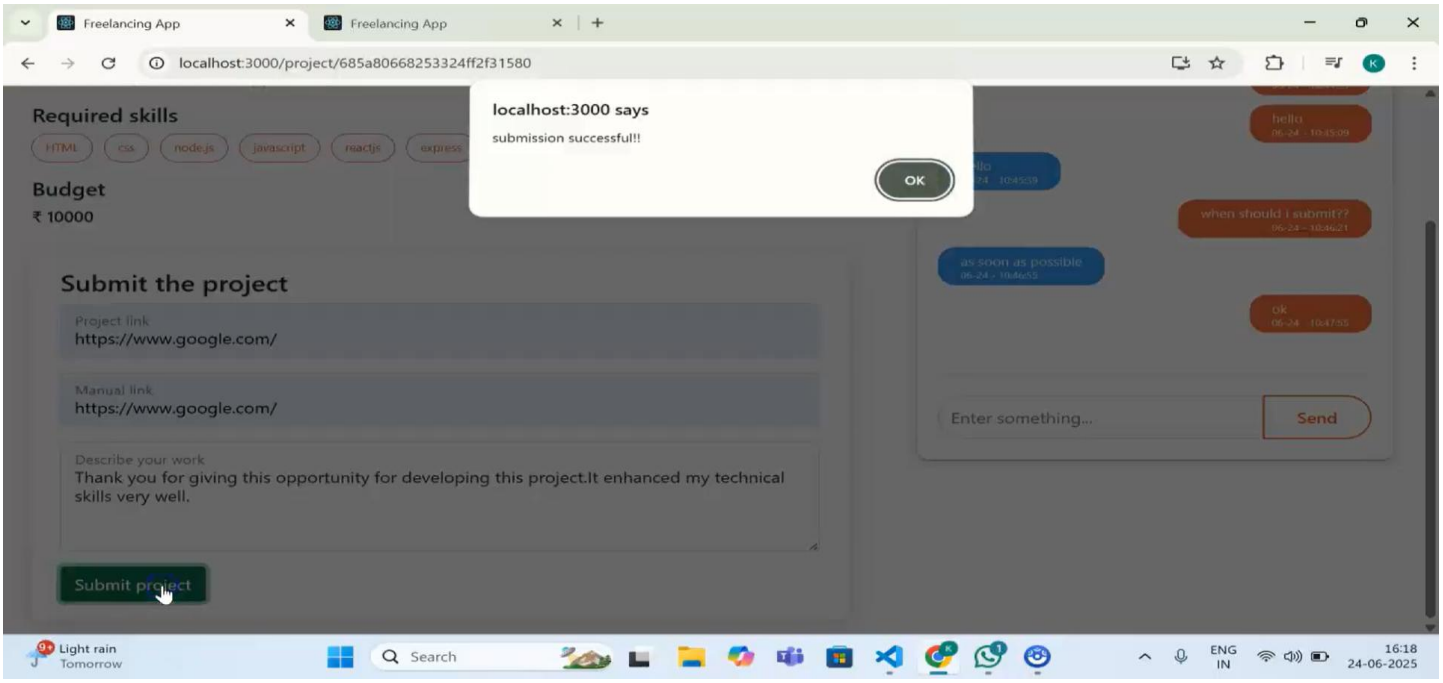
Admin page:



Chat between client and freelancer:



Submission of project by Freelancer:



Conclusion:

FreelanceFinder empowers individuals to take control of their professional journey, one project at a time. By offering an intelligent, secure, and user-friendly platform, we aim to unlock the true potential of freelance work and connect talent with global opportunities.

For any further doubts or help, please consider the code in the drive link given below,

https://drive.google.com/file/d/18-gnCVAZdojBhl7QM0_yShH1JL5A6v32/view?usp=drive_link

The demo of the app is available at:

https://drive.google.com/drive/folders/1gSc8RWmM7yqdvYh525n5DoLjZxS_N-F4

**** Happy Coding ****