SB Works -Freelancing Platform

Team Members:

Developer 1 (**Mukthapuram Kavya)** – Frontend Developer (React.js, Bootstrap)

Developer 2(**Vaishnavi Bobburi)** – Backend Developer (Node.js, Express.js)

Developer 3 (**K Pujitha)** – Database Designer (MongoDB)

Developer 4(**K Katyaeni)** – UI/UX Designer&Testing

Purpose:

The primary purpose of SB Freelancing is to provide a comprehensive and user-friendly online platform that bridges the gap between clients seeking specialized project support and freelancers offering diverse professional services. As the freelancing economy grows, the need for a robust digital marketplace becomes increasingly important. SB Freelancing is designed to streamline this dynamic interaction by offering tools that simplify project management, foster effective collaboration, and ensure transparent communication.

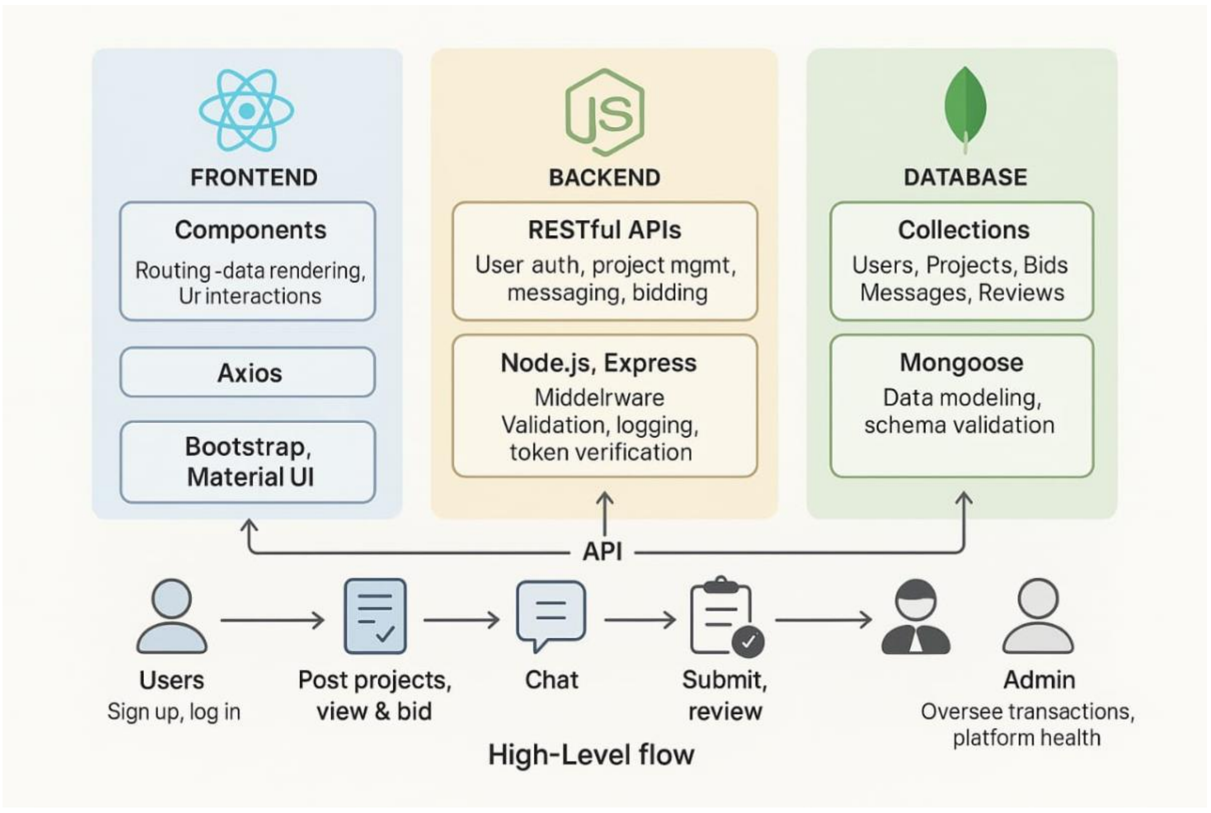
The platform is intended to serve as a centralized hub where clients can post various types of projects—ranging from graphic design and content writing to web development and software engineering. Freelancers can browse these listings, bid competitively based on their skills and experience, and interact directly with clients to clarify project goals and deliverables. This ensures that every engagement is based on mutual understanding and clearly defined expectations.

In addition to supporting core freelancing activities, SB Freelancing prioritizes user trust and safety. The inclusion of an active admin role enables monitoring of user interactions, dispute resolution, and the implementation of security policies to maintain the platform's integrity. Feedback and review systems enhance credibility and encourage a quality-focused environment, where talented freelancers can build strong reputations and clients can consistently find reliable partners.

Features:

- Project posting and bidding system
- Freelancer and client profile management
- Real-time messaging and notifications
- Project tracking and delivery mechanism
- Role-based login (Client, Freelancer, Admin)
- Admin dashboard for user and transaction management
- Feedback and rating system post-project completion
- Secure authentication with JWT

Architecture:


FRONTEND
Components
Routing -data rendering, Ur interactions
Axios
Bootstrap, Material UI
BACKEND
RESTful APIs
User auth, project mgmt, messaging, bidding
Node.js, Express
Middelrware Validation, logging, token verification
DATABASE
Collections
Users, Projects, Bids Messages, Reviews
Mongoose
Data modeling, schema validation
API
Users
Sign up, log in
Post projects, view & bid
Chat
Submit, review
Admin
Oversee transactions, platform health
High-Level flow

Frontend:
Developed with React.js, the frontend consists of reusable components that handle routing, data rendering, and UI interactions. Axios is used for API communication. Bootstrap and Material UI provide consistent design elements and responsive layouts.

Backend:
The backend is built using Node.js and Express.js. It features RESTful APIs for all core operations, including user authentication, project management, messaging, and bidding. Middleware layers ensure validation, logging, and token verification.

Database:
MongoDB is used for its scalability and document-oriented structure. Collections include Users, Projects, Bids, Messages, and Reviews. Mongoose handles data modeling and schema validation.

High-Level Flow:

- Users sign up and log in via secure endpoints.
- Clients post projects; freelancers can view and bid.
- Communication happens through a chat system.
- Completed projects are submitted and reviewed.
- Admin oversees transactions and platform health.

Prerequisites:

- Node.js
- MongoDB
- npm or yarn

Installation:

```
bash
CopyEdit
git clone https://github.com/your-repo/sbworks.git
cd sbworks
cd client
npm install
cd ../server
npm install
```

Set up .env file in server/ with the following:

```
ini
CopyEdit
MONGO_URI=your_mongodb_connection_string
JWT_SECRET=your_jwt_secret
```

6.Folder Structure

- **Client (Frontend):**

```
css
CopyEdit
client/
├── src/
│   ├── components/
│   ├── pages/
│   ├── services/
│   ├── App.js
│   └── index.js
│
```

- **Server (Backend):**

```
pgsql
CopyEdit
server/
├── controllers/
├── models/
├── routes/
├── middleware/
├── server.js
└── config/
```

7.Running the Application

- **Frontend:**

  ```
  bash
  CopyEdit
  cd client
  npm start
  ```

- **Backend:**

  ```
  bash
  CopyEdit
  cd server
  npm start
  ```

7. API Documentation

Example:

- **POST**/api/auth/login
  - **Request Body:**

    ```
    json
    CopyEdit
    {
    "email":"user@example.com",
    "password":"123456"
    }
    ```

  - **Response:**

    ```
    json
    CopyEdit
    {
    "token":"JWT_TOKEN",
    "user":{"id":"123","role":"freelancer"}
    }
    ```

8. Authentication

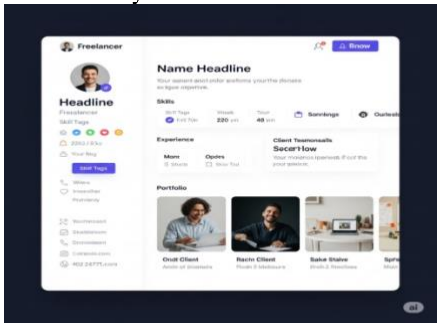
- **Method:** JWT-based authentication
- **Process:**
  - Upon login, a JWT token is issued and stored in the client (localStorage).
  - Protected routes require the token to be included in headers.
  - Middleware verifies token and assigns roles (client, freelancer, admin).

9. User Interface

Freelancer Profile:

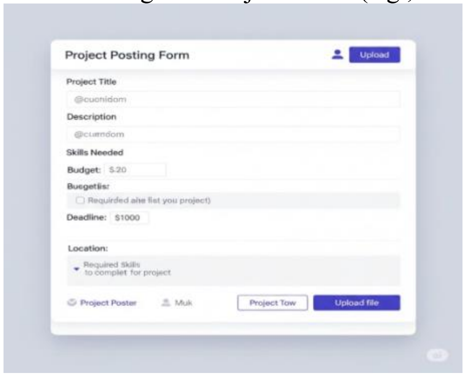
- A digital resume showcasing a freelancer's skills and work.

- A page with their photo, "Web Developer" title, a list of skills like "React, Python," and a gallery of websites they've built.



Project Posting Form:

- How a client tells freelancers what work they need done.
- A form asking for "Project Title" (e.g., "Design New Logo"), "Description," "Budget," and "Deadline."

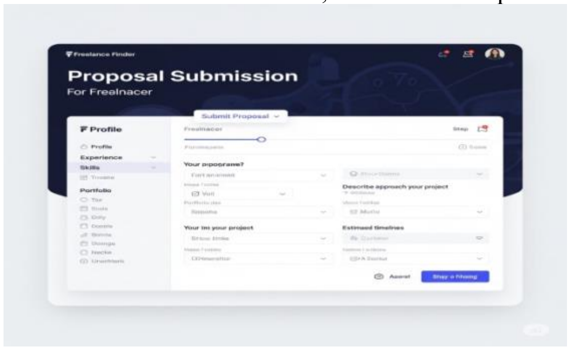


Search Filters:

- Tools to narrow down results when looking for freelancers or projects.
- Dropdowns or checkboxes for "Skill (e.g., Content Writing)," "Rate ($20-$50/hr)," and "Experience Level."

Proposal Submission:

- How a freelancer applies for a project.
- A text box for a "Cover Letter," fields for "Proposed Amount," and "Delivery Time."



In-Platform Chat:

- A secure way for clients and freelancers to talk directly.
- A chat window showing message bubbles, file attachments, and timestamps.

10. Testing

Our testing strategy follows a layered approach to ensure comprehensive quality:

Unit Testing: Developers test individual code components in isolation to catch bugs early.

Integration Testing: Verifies that different modules and services work correctly when combined.

System Testing: Tests the complete application end-to-end against all specified requirements (functional and non-functional).

User Acceptance Testing (UAT): Real users validate the system meets business needs and is ready for release.

Regression Testing: Automated tests continuously confirm that new code changes haven't broken existing functionality.

Specialized Testing: Includes performance (load, stress), security (vulnerability), and usability testing to cover specific quality aspects.

Tools Used:

Code-level Testing (Unit/Integration): Frameworks like JUnit (Java), Pytest (Python), Jest (JavaScript) are used for automated code validation.

Web/API Automation: Selenium WebDriver, Cypress, or Playwright are key for automating browser interactions, while Postman or Rest-Assured are common for API testing.

Performance Testing: JMeter or LoadRunner simulate user load to assess system responsiveness and scalability.

Test Management: Tools like Jira (with plugins like Zephyr or Xray) help organize test cases, track execution, and manage defects.

CI/CD Integration: Jenkins, GitLab CI/CD, or GitHub Actions automate the execution of tests within the development pipeline for continuous feedback.

11.Known Issues

Users and developers should be aware of the following known issues with AI models like me:

Hallucinations: The model can generate factually incorrect or made-up information that sounds plausible. Always verify critical details.

Bias: Responses may reflect biases present in the training data, potentially leading to unfair or stereotypical outputs**.**

Limited Reasoning/Common Sense: Struggles with complex logic, multi-step problems, or nuanced understanding beyond statistical patterns.

Outdated Knowledge: My knowledge is limited to my training data cutoff and I don't have real-time information**.**

Prompt Sensitivity: Minor changes in prompt wording can lead to different or unexpected results.

Security Vulnerabilities: Malicious prompts (prompt injection) can sometimes bypass safety controls or attempt to extract information.

12. Future Enhancements

AI-Powered Matching: More sophisticated algorithms for matching clients with freelancers based on skills, portfolio, experience, and even soft skills or project culture fit.

Skill Verification & Certification: Integrate trusted third-party skill assessments or offer platform-specific certifications to boost freelancer credibility.

Enhanced Portfolio & Showcase Tools: Allow richer media (video, interactive demos), case studies, and customizable portfolio layouts for freelancers.

Integrated Project Management: Built-in tools for task tracking, file sharing, progress updates, and milestone management for both parties.

Financial Management & Tax Tools: Automated invoicing, expense tracking, tax estimations, and potential integration with accounting software.

Learning & Development Hub: Offer courses, workshops, or resources for freelancers to upskill and stay competitive.

Community & Networking Features: Forums, groups, or direct messaging to foster collaboration, mentorship, and support among freelancers.