Project Documentation

1. Introduction

• Project Title: Your Personal Fitness Companion

• Team ID : NM2025TMID32278

• Team Leader : Srinithi / ssrinithi172@gmail.com

• Team Members:

Sripriya / sripriya572@gmail.com

Sowmiya / sowmiyadharamaraj2007@gmail. Com

Sowmiya/ asowmiya001@gmail.com

2. Project Overview

• Purpose: SB Works connects clients and freelancers through project postings, bidding, and

real-time communication.

• Features:

– Project posting and bidding

– Secure chat system

– Feedback and review system

– Admin control panel

ANS:

SB Works is a platform that connects clients and freelancers. Here's a breakdown:

- purpose : SB Works lets clients post projects and freelancers bid on them. They can communicate in real-time.

- Features :

* Project posting and bidding*: Clients post projects, freelancers bid on them.

* Secure chat system*: Clients and freelancers can message each other securely.

* Feedback and review system*: Both can leave reviews for each other after projects.

* Admin control panel*: Admins manage the platform, users, and projects.

Does that help? Do you want me to explain any of these features more or help with something related to SB Works?

3. Architecture

• Frontend: React.js with Bootstrap and Material UI

• Backend: Node.js and Express.js managing server logic and API endpoints

• Database: MongoDB stores user data, project information, applications, and chat messages 2

ANS:

Here's a breakdown of the architecture:

*Frontend :

* React.js: A JavaScript library for building user interfaces and single-page applications.

* Bootstrap*: A CSS framework for responsive design and styling.

* Material UI*: A popular React component library for implementing Google's Material Design.

*Backend :

* Node.js: A JavaScript runtime environment for building server-side applications.

* Express.js: A Node.js framework for building web applications and managing API endpoints.

*Database :

* MongoDB : A NoSQL database that stores data in a JSON-like format, used for storing:

- User data
- Project information
- Applications
- Chat messages

This tech stack is well-suited for building a scalable and dynamic web application like SB Works. React.js handles the frontend, Node.js and Express.js manage the backend logic and API endpoints, and MongoDB stores and retrieves data efficiently.

4.Setup Instructions

•Prerequisites:

* Node.js

* MongoDB

* Git

* React.js

* Express.js – Mongoose – Visual Studio Code

• Installation Steps:

* Clone the repository git clone

*Install client dependencies cd

client npm install

* Install server dependencies cd

server npm install

ANS ;Here's a breakdown:

1. Node.js: A JavaScript runtime environment that allows you to run JavaScript on the server-side. It's used for building scalable and high-performance server-side applications.

2. MongoDB: A NoSQL database that stores data in a JSON-like format, allowing for flexible and efficient data storage and retrieval.

3. Git: A version control system that helps you track changes in your codebase, collaborate with others, and manage different versions of your project.

4. React.js: A JavaScript library for building user interfaces and single-page applications. It's used for creating reusable UI components and managing state changes.

5. Express.js: A Node.js framework for building web applications and RESTful APIs. It provides a lightweight and flexible way to handle HTTP requests and responses.

6. Mongoose: An Object Data Modeling (ODM) library for MongoDB and Node.js. It provides a simple way to interact with MongoDB, define schemas, and perform CRUD (Create, Read, Update, Delete) operations.

7. Visual Studio Code (VS Code): A lightweight, open-source code editor that provides a range of features, including syntax highlighting, debugging,

and extensions for various programming languages.

These technologies are commonly used together to build modern web applications, with:

* Node.js and Express.js handling the backend

* React.js handling the frontend

* MongoDB storing data

* Mongoose interacting with MongoDB

* Git managing version control

* V Code providing a development environment

Does that help clarify things?

5. Folder Structure

SB-Works :

* client / React frontend

* components

* pages

* server/ Node.js backend

* routes

* models

* controllers

Ans:Here's a breakdown:

Folder Structure: SB-Works

The project is organized into two main folders:

1. Client (Frontend)

- `components/`: Reusable UI components (e.g., buttons, forms, cards) used throughout the application.
- `pages/`: Individual pages of the application (e.g., Home, About, Dashboard).

2. Server (Backend)

* Routes : Defines API endpoints and handles HTTP requests (e.g., `/users`, `/projects`).
* Models : Defines the structure and schema of data stored in the database (e.g., User, Project).
* Controllers : Handles business logic and interacts with models to perform CRUD operations.

This structure separates the frontend and backend concerns, making it easier to manage and scale the application.

Some notable benefits:

- Easy to find and modify code
- Clear separation of concerns (frontend vs. backend)
- Scalable and maintainable architecture

Does that help clarify the structure?

6. Running the Application

• Frontend:

cd client

npm start •

Backend:

cd server npm

start

• Access: Visit http://localhost:3000

Ans:Here's a breakdown:

Running the Application

To start the application, follow these steps:

*1. Frontend :

* cd client`: Navigate to the client folder (React frontend).

* npm start`: Starts the frontend development server.

*2. Backend :

* cd server`: Navigate to the server folder (Node.js backend).

* npm start`: Starts the backend server.

3. Access:

* Visit `http://localhost:3000` in your web browser to access the application.

This will start both the frontend and backend servers, allowing you to access the application locally.

Some notes:

* Make sure you have Node.js and npm installed on your system.

* The frontend and backend servers might run on different ports (e.g., frontend on port 3000, backend on port 8080).

* You might need to configure the frontend to proxy requests to the backend server.

By running both servers and accessing the application, you'll be able to interact with the frontend and backend components.

7. API Documentation

• User:

– /api/user/register

– /api/user/login

• Projects: – /api/projects/create

– /api/projects/:id • Applications: /api/apply

• Chats:

– /api/chat/send

– /api/chat/:userId

Ans;

These are API endpoints for chat functionality:

- *Purpose:* Send a new chat message.

- *HTTP Method:* Typically `POST`.

- *Request Body:* Message content, sender ID, receiver ID, etc.

- *Response:* Success or failure status, possibly with the sent message details.

2. `/api/chat/:userId`

- *Purpose:* Retrieve chat messages with a specific user.

- *HTTP Method:* Typically `GET`.

- *Path Parameter:* `:userId` (ID of the user whose chat messages are being retrieved).

- *Response:* List of chat messages between the authenticated user and the specified user.

These endpoints enable basic chat functionality:

- Sending new messages

- Retrieving conversation history with a specific user

Does that clarify the purpose of these endpoints?

8. Authentication

• JWT-based authentication for secure login

• Middleware protects private routes.Here's a breakdown:

ANS ;

Authentication Features:

- *JWT-based authentication*: JSON Web Tokens (JWT) are used to securely authenticate users. When a user logs in, a JWT token is generated and sent to the client, which then includes it in subsequent requests to authenticate.

- *Middleware protects private routes*: Middleware functions verify the JWT token in incoming requests to protected routes. If the token is valid, the request is allowed to proceed; otherwise, it's blocked, ensuring that only authenticated users can access private routes.

This setup provides:

- *Security*: JWT tokens are digitally signed, making them tamper-proof and secure.

- *Authentication*: Users are verified before accessing protected resources.

- *Authorization*: Access to private routes is restricted to authenticated users.

By using JWT-based authentication and middleware, the application ensures secure and controlled access to sensitive data and features.

9. User Interface

• Landing Page

• Freelancer Dashboard

• Admin Panel

• Project Details Page

Ans ; These are key pages in the SB Works application:

1. Landing Page

- *Purpose:* Introduce the platform, showcase its features, and encourage users to sign up or log in.

- *Content:* Taglines, benefits, call-to-actions, and possibly testimonials or featured projects.

2. Freelancer Dashboard

- *Purpose:* Provide freelancers with a centralized hub to manage their projects, bids, and profile.

- *Features:*

    - View and manage ongoing projects

    - Receive notifications for new project postings and messages

    - Update profile and portfolio

    - Track earnings and performance metrics

3. Admin Panel

- *Purpose:* Allow administrators to manage the platform, users, and projects.

- *Features:*

    - User management (freelancers and clients)

    - Project moderation and monitoring

    - Dispute resolution and support

    - Platform settings and configuration

4. Project Details Page

- *Purpose:* Display detailed information about a specific project.

- *Content:*

    - Project description and requirements

    - Budget and timeline

    - Client information (anonymized or partial, depending on the platform's settings)

    - Bidding or application functionality for freelancers

These pages cater to different user roles and needs, providing a comprehensive experience for freelancers, clients, and administrators.

10. Testing

• Manual testing during milestones

• Tools: Postman, Chrome Dev Tools

Answer :

Testing is carried out through manual checks at each milestone to ensure functionality and quality. Tools like Postman are used for API validation, while Chrome Dev Tools assist in debugging and analyzing frontend performance. This helps in identifying issues early and ensuring smooth progress throughout development.

11. Screenshots or Demo

Answer :

Screenshots or demos are provided to showcase the application’s features and functionality. They help in visually demonstrating the workflows, UI, and key outcomes to stakeholders. This makes it easier to validate progress and gather feedback effectively.

12. Known Issues

Answer :

Known issues are the bugs or limitations identified during testing that are yet to be fixed. They are documented clearly to keep the team and stakeholders informed. This helps in tracking, prioritizing, and planning fixes in future updates or releases.

13. Future Enhancements

Ans ;

Future enhancements include adding new features, improving performance, and refining the user experience based on feedback. These planned updates aim to make the application more efficient, scalable, and user-friendly in upcoming versions.

... Thank you...