**PROJECT DOCUMENTATION**

PROJECT TITLE:

FITFLEX- YOUR PERSONAL FITNESS COMPANION



TEAM ID:NM2025TMID36554

TEAM LEADER

* NAME:A.ANGEL [24csangel006@gmail.com](mailto:24csangel006@gmail.com)

TEAM MEMBER

* AMRITHA.DN [24csamritha005@gmail.com](mailto:24csamritha005@gmail.com)
* ANUSIYA.K [24csanusiya007@gmail.com](mailto:24csanusiya007@gmail.com)
* ARCHANA.M [24csarchana008@gmail.com](mailto:24csarchana008@gmail.com)

**Project Overview**

**Purpose**

**Fit Flex is a smart fitness marketplace that connects clients with professional trainers. It goes beyond regular fitness apps by letting clients post their fitness goals and trainers bid with personalized plans. The platform includes a secure chat system for direct communication, a feedback and review option for trust and accountability, and an admin panel to ensure fairness and quality. In short,**

**Fit Flex makes fitness services more accessible, transparent, and interactive for everyone. Fit Flex is a smart fitness marketplace that bridges the gap between clients and trainers. Unlike basic fitness apps, it provides a complete interactive platform where clients can post fitness goals, and trainers can bid with customized solutions.**

**Features**

1. **Project posting & bidding:**  **clients can post their fitness needs—such as weight loss, strength training, yoga, or diet planning—by giving details like goals, duration, and preferences. Once posted, trainers can bid on these projects by submitting their plan, pricing, and approach. This creates a competitive environment where clients can compare different offers and choose the trainer who best matches their requirements.**
2. **Secure Chat System: Fit Flex has a built-in real-time chat that lets clients and trainers communicate safely within the platform. It allows sharing progress, clarifying doubts, and making quick adjustments without relying on external apps.**
3. **Feedback & Review: clients can rate trainers and give feedback after each project, ensuring quality, accountability, and trust on the platform.**
4. **Admin Control Panel: Admins oversee platform activities, verify trainers, resolve disputes, and maintain safety, fairness, and quality across the system.**

**Architecture**

**Fit Flex uses a three-tier architecture: frontend, backend, and database, designed for scalability, flexibility, and security.**

**Frontend**

**Built with React.js for dynamic and responsive UI. Bootstrap and Material UI ensure modern, mobile-friendly designs. It serves as the client-facing layer for posting projects, chatting, and overall user interaction.**

**Backend: Developed usingNode.js with Express.js, handling APIs, authentication, project bidding rules, and chat delivery. It ensures smooth, real-time performance with secure communication.**

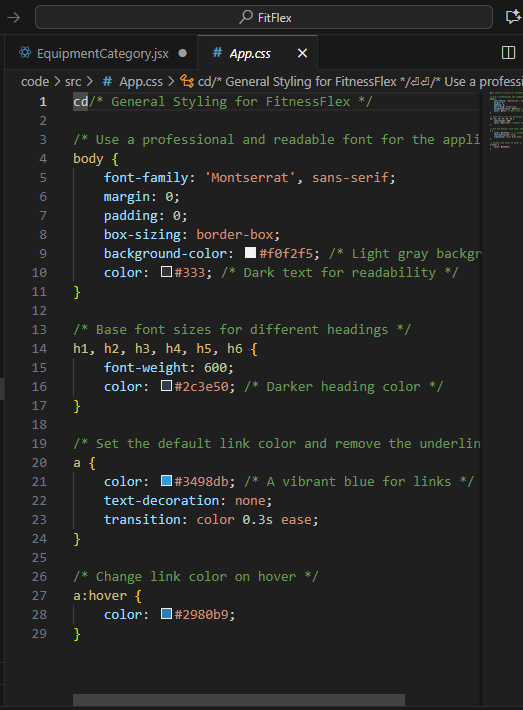
**Database: Powered by MongoDB, a flexible NoSQL database that manages user data, projects, bids, and chat history. Its schema-less design supports scalability and easy future upgrades.**

**Workflow**

**User actions (login, project post, chat) go from frontend → backend → database → back to frontend, ensuring real-time, secure, and efficient processing.**

A screenshot of a computer

AI-generated content may be incorrect.

****

**Setup Instructions**



**Prerequisites**

**Before starting development, ensure that the following tools and technologies are installed and configured properly on your system:**

1. **Node.js**
   * **Node.js is a JavaScript runtime environment built on Chrome’s V8 engine.**
   * **It allows you to run JavaScript on the server side, enabling backend development.**
   * **Install the latest LTS (Long-Term Support) version to ensure stability and compatibility with most libraries.**
   * **Verify installation with:**
   * **node -v**
   * **npm -v**
   * **npm (Node Package Manager) comes bundled with Node.js and is used to install and manage project dependencies.**
2. **MongoDB**
   * **A NoSQL database that stores data in a flexible, JSON-like format (BSON).**
   * **Ideal for projects that require scalability and real-time data handling.**
   * **Can be installed locally or accessed via MongoDB Atlas (cloud database service).**
   * **Verify installation with:**
   * **mongod --version**
   * **mongo --version**
3. **Git**
   * **A version control system to track changes in your codebase.**
   * **Helps in managing project collaboration, branching, merging, and rollback of versions.**
   * **Required for pushing and pulling code from repositories (e.g., GitHub, GitLab, Bitbucket).**
   * **Verify installation with:**
   * **git --version**
4. **React.js**
   * **A frontend JavaScript library used for building dynamic and responsive user interfaces.**
   * **React uses a component-based architecture, making it easier to build reusable UI elements.**
   * **Install via npm or yarn:**
   * **npx create-react-app my-app**
   * **Works seamlessly with modern build tools like Webpack, Babel, and Vite.**
5. **Express.js**
   * **A minimalist web framework for Node.js, designed for building APIs and handling server-side logic.**
   * **Provides powerful features like routing, middleware support, and easy integration with databases.**
   * **Install via npm:**
   * **npm install express**
6. **Mongoose**
   * **An Object Data Modeling (ODM) library for MongoDB and Node.js.**
   * **Simplifies interactions with MongoDB by providing a schema-based solution for data modeling.**
   * **Supports validation, middleware, and easy CRUD (Create, Read, Update, Delete) operations.**
   * **Install via npm:**
   * **npm install mongoose**
7. **Visual Studio Code (VS Code)**
   * **A lightweight yet powerful IDE recommended for development.**
   * **Supports JavaScript, React, Node.js, MongoDB, and Git with extensions.**
   * **Features include debugging, IntelliSense (smart code completion), Git integration, and built-in terminal.**
   * **Recommended extensions:**
     + **ES7+ React/Redux/React-Native snippets**
     + **Prettier - Code formatter**
     + **MongoDB for VS Code**
     + **GitLens**

**Here’s your Installation Steps section expanded and neatly formatted:**

**Installation Steps**

**1. Clone the repository**

**git clone <repository\_url>**

**cd <repository\_name>**

**2. Install client dependencies**

**cd client**

**npm install**

**3. Install server dependencies**

**cd ../server**

**npm install**

**Running the Application**

**Once all prerequisites are installed and dependencies are set up, you can run both the frontend and backend of the application.**

**Frontend (React.js)**

1. **Navigate to the frontend folder:**
2. **cd client**
3. **Install dependencies (only required the first time or after adding new packages):**
4. **npm install**
5. **Start the React development server:**
6. **npm start**
7. **By default, the React app will run at:**
8. **http://localhost:3000**
9. **Key Notes:**
   * **The server will auto-reload whenever you make changes to the source code.**
   * **If port 3000 is already in use, React will prompt to use a different port (e.g., 3001).**
   * **Configuration for proxying API requests to the backend can be set in client/package.json under "proxy".**

**Backend (Node.js + Express)**

1. **Navigate to the backend folder:**
2. **cd server**
3. **Install backend dependencies:**
4. **npm install**
5. **Start the backend server:**
6. **npm start**

**(or, if configured with nodemon for auto-reloading on code changes):**

**npm run dev**

1. **By default, the Express backend runs at:**
2. **http://localhost:5000**
3. **Configuration:**
   * **The server port is typically defined in a config file or environment variable (e.g., .env or server/config.js).**
   * **Example from .env:**
   * **PORT=5000**
   * **MONGO\_URI=mongodb://localhost:27017/mydatabase**
   * **JWT\_SECRET=mysecret**
   * **You can change the port in server/config or by editing the .env file.**

**10.Key Notes:**

* + **If MongoDB is running locally, ensure the MongoDB service is started before running the backend:**
  + **mongod**
  + **Logs will appear in the terminal showing whether the backend successfully connected to MongoDB.**
  + **API endpoints can be tested with tools like Postman or directly from the frontend.**

**Running Frontend & Backend Together**

* **Open two terminal windows/tabs:**
  + **First tab → Run the frontend (cd client && npm start).**
  + **Second tab → Run the backend (cd server && npm start).**
* **Alternatively, you can configure a concurrent run script in the root package.json:**
* **"scripts": {**
* **"start": "concurrently \"npm run server\" \"npm run client\"",**
* **"server": "cd server && npm start",**
* **"client": "cd client && npm start"**
* **}**

**Then simply run:**

**npm start**

**Authentication**

**1. JWT-Based Authentication**

**The application uses JSON Web Tokens (JWT) to secure user login and sessions.**

**Upon successful login, the server generates a JWT token and returns it to the client.**

**The client stores the token (typically in localStorage or cookies) and includes it in the header of every request to protected routes:**

**Authorization: Bearer <token>**

**2. Middleware for Route Protection**

**Backend middleware verifies the JWT token on every protected route.**

**If the token is valid, the request proceeds; otherwise, the server responds with 401 Unauthorized.**

**Example Usage in Express.js:**

**const authMiddleware = (req, res, next) => {**

**const token = req.header("Authorization")?.split(" ")[1];**

**if (!token) return res.status(401).json({ message: "Access denied" });**

**try {**

**const decoded = jwt.verify(token, process.env.JWT\_SECRET);**

**req.user = decoded;**

**next();**

**} catch (err) {**

**res.status(401).json({ message: "Invalid token" });**

**}**

**};**

**User Interface**

**SB-Works provides a clean and intuitive interface for both freelancers and admins. The main UI components include:**

**1. Landing Page**

**First page users see when visiting the platform.**

**Highlights platform features, project categories, and call-to-action buttons (Sign Up / Login).**

**Designed to attract freelancers and clients.**

**2. Freelancer Dashboard**

**Personalized dashboard for freelancers.**

**Shows available projects, applied projects, and chat notifications.**

**Allows freelancers to apply for projects, view project details, and track application status.**

**3. Admin Panel**

**Dashboard for admin users to manage users, projects, and applications.**

**Features include approving projects, monitoring chats, and generating reports.**

**4. Project Details Page**

**Displays full project information: title, description, budget, deadline, and client details.**

**Freelancers can submit applications or start chat discussions with clients.**

**Testing**

**1. Manual Testing**

**Conducted during different project milestones to ensure features work as expected.**

**Key scenarios tested include:**

**User registration and login**

**Project creation and application submission**

**Chat functionality between users**

**Access control for protected routes**

**2. Tools Used**

**Postman – To test all backend APIs (GET, POST, PUT, DELETE) and verify request/response behavior.**

**Chrome Dev Tools – To debug frontend issues, monitor network requests, and inspect UI elements.**

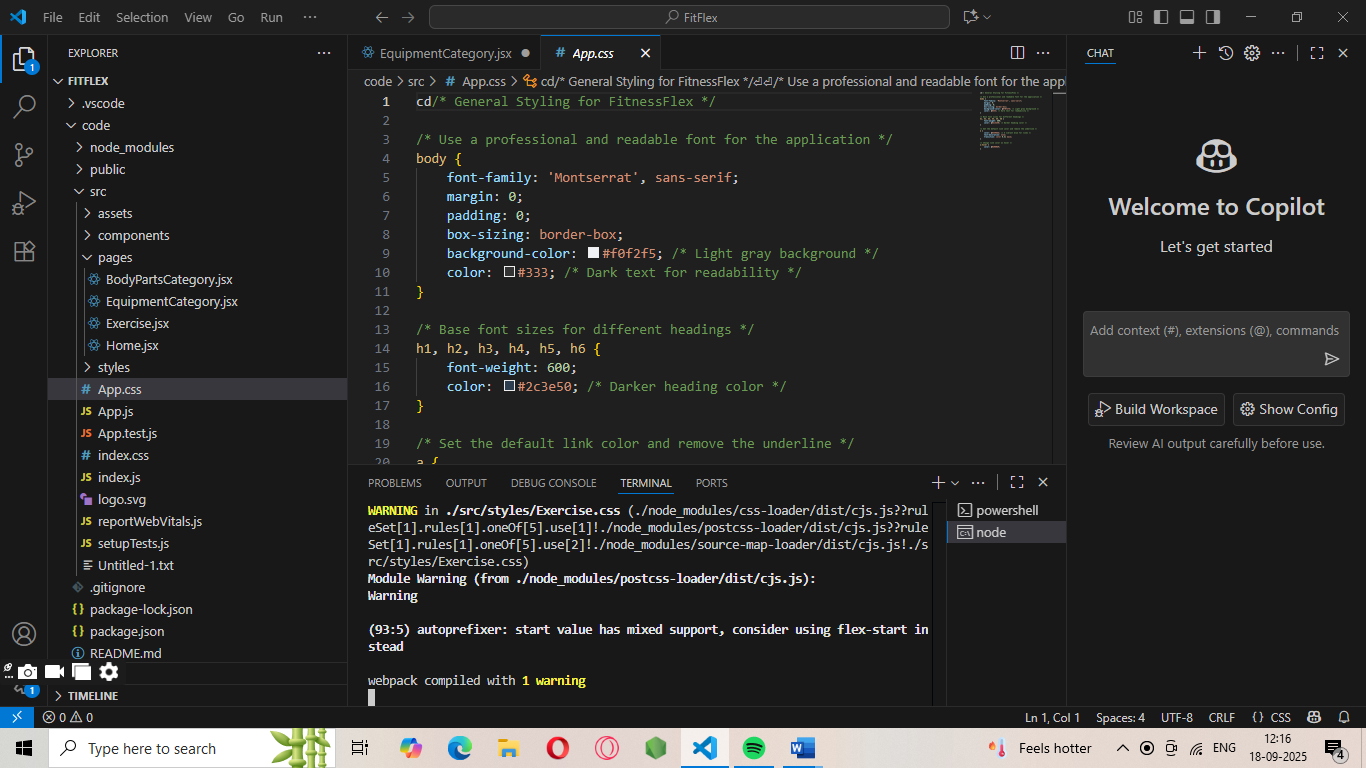
**Screenshots**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.



**Known Issues**

**Real-time chat delays – Messages may experience slight delays under heavy load.**

**Mobile responsiveness – Some UI components may not render perfectly on smaller screens.**

**Error handling – Certain edge cases in form validation may not display user-friendly messages.**

**Database performance – Large numbers of projects or users may slightly slow down query responses.**

**Authentication token expiration – Users may need to log in again if the token expires during long sessions**

**Future Enhancements**

**Push Notifications – Real-time alerts for new messages or project updates.**

**Advanced Search & Filters – For projects based on budget, skills, or deadline.**

**Profile Management – Users can update their skills, portfolio, and profile picture.**

**Mobile App Version – Native mobile applications for iOS and Android.**

**Analytics Dashboard – Admin panel with charts for projects, users, and applications.**

**Payment Integration – Secure payments between clients and freelancers.**

**Improved Chat Features – Read receipts, file sharing, and emojis.**