# **FITFLEX : YOUR PERSONAL FITNESS COMPANION**

# **INTRODUCTION**

* **Project Title:** Fit Flex
* **Team ID** : NM2025TMID38949
* **Team Leader:**

|  |  |
| --- | --- |
| NAME | MAIL ID |
| ARCHANA | <Archanaarchu3853@gmail.com> |

* **Team Members:**

|  |  |
| --- | --- |
| **NAME** | **MAIL ID** |
| KALPANA A | <Ka7666649@gmail.com> |
| DHARMESH KUMER B | <Dsbb289@gmail.com> |
| YOGESHWARAN K | <Yogesh13waran05@gmail.com> |

# **PROJECT OVERVIEW**

**2.1 Purpose:** Sweat Shine connects clients and freelancers through project postings, bidding, and real-time communication.

The purpose of FITFLEX is to create a **reliable and user-friendly fitness guide** that motivates people to exercise regularly and perform workouts with proper technique. By offering categorized exercises and clear instructions, the platform helps users build healthier lifestyles, reduce the risk of injury, and track steady progress in their fitness journey.

* 1. **Key Features**
* **Exercise Library** – A wide collection of gym exercises organized by muscle groups.
* **Step-by-Step Instructions** – Clear workout guidance to ensure proper form and techniqum
* **Difficulty Levels** – Workouts tailored for Beginners, Intermediate, and Advanced users.
* **Workout Schedules** – Ready-made daily and weekly routines to stay consistent.
* **Training Tips** – Safety advice and effective fitness tips for better results.
  1. **Target Audience**
* **Beginners** – Individuals who are new to fitness and need step-by-step workout guidance.
* **Intermediate Trainees** – People with some gym experience looking for structured routines.
* **Advanced Fitness Enthusiasts** – Users aiming to challenge themselves with harder workouts.
* **Health-Conscious Individuals** – Anyone focused on staying active and maintaining a healthy lifestyle.
* **Gym-Goers** – Regular visitors to the gym who want proper form and exercise variety.

# 3. **ARCHITECTURE**

1. **Frontend – React.js with Bootstrap and Material UI**  
    The frontend of *Sweat Shine* is built using **React.js**, ensuring a highly dynamic and responsive user interface. **Bootstrap** provides a robust grid system and layout framework, while **Material UI** adds modern, pre-designed components for consistency and visual appeal. Together, they deliver a smooth, intuitive, and interactive user experience across devices.
2. **Backend – Node.js and Express.js**  
    The backend is powered by **Node.js** for fast, scalable, and event-driven server operations. **Express.js** is used to handle routing, server logic, and API endpoints, ensuring efficient communication between the frontend and database. This combination supports secure data processing and reliable application performance.
3. **Database – MongoDB**  
    **MongoDB** serves as the primary database, chosen for its flexibility in handling unstructured data. It stores user information, project postings, application details, and chat messages in a secure and scalable manner. Its document-oriented structure makes it well-suited for real-time interactions and large-scale data management.

# 4. **SETUP INSTRUCTIONS**

* 1. **Prerequisites**

Before setting up ***Sweat Shine*,** ensure that the following tools and technologies are installed on your system:

* + **Node.js** – JavaScript runtime environment.
  + **MongoDB** – Database for storing user data, projects, and chat messages.
  + **Git** – For cloning and version control.
  + **React.js** – Frontend framework for building the user interface.
  + **Express.js** – Backend framework for handling APIs and server logic.
  + **Mongoose** – ODM (Object Data Modeling) library for MongoDB.
  + **Visual Studio Code** – Recommended IDE for development.
  1. **Installation steps**
     1. **Node.js**
* **Knowledge**: basic JavaScript (variables, functions, promises/async-await), basic command-line use (terminal / PowerShell), JSON, and git (recommended).
* **System**: Windows / macOS / Linux (x64 or arm64); 2GB+ RAM recommended; internet to download installers.
* **Tools to have ready**: a terminal (CMD/PowerShell/WSL on Windows, Terminal on macOS/Linux), a code editor (VS Code recommended).

**Step-by-step installation**

**Windows — easiest: official installer (or use npm-windows for version management)**

**Official installer (quick)**

1. Visit the official Node.js downloads page and download the Windows **.msi** (choose the **LTS** build).
2. Run the MSI and accept defaults (it will install node and npm and add to PATH).
3. Open PowerShell or Command Prompt and verify:

node -v

npm -v

you’re done. [nodejs.org](https://nodejs.org/en/download?utm_source=chatgpt.com)

**If you want multiple Node versions / safer global installs — use npm for Windows**

1. Install **npm-windows** (download and run the installer from the npm-windows releases).
2. After install, open a new admin PowerShell or CMD and use:

npm install lts

npm list

npm use <version> # e.g. npm use 18.16.0 OR `npm use lts` if supported

node -v

npm-windows lets you switch versions without re-installers. [GitHub+1](https://github.com/coreybutler/nvm-windows?utm_source=chatgpt.com)

npm -v

you’re done.

* + 1. **MongoDB**
  + **Knowledge Prerequisites**
  + Basic **database concepts**: collections, documents, CRUD (Create, Read, Update, Delete).
  + Basic **command line/terminal** usage.
  + JSON knowledge (MongoDB stores data in BSON, similar to JSON).
  + (Optional) Networking basics: ports, IP binding, authentication.

**System Prerequisites**

* + **OS**: Windows (x64/arm64 supported).
  + **Processor**: 64-bit, 1 GHz+ recommended.
  + **RAM**: Minimum 2 GB (4 GB+ recommended for dev; higher for prod).
  + **Disk space**: At least 5–10 GB free (MongoDB stores large amounts of data).
  + **Port**: MongoDB runs on 27017 by default (make sure it’s free/unblocked).

**Software Prerequisites**

* + **Administrator** (to install packages/services).
  + **Package manager** (apt, yum, Homebrew) OR direct installer.
  + **MongoDB Shell (mongosh)** – required to interact with MongoDB.
  + **MongoDB Compass (GUI)** – optional but recommended for beginners.
  + **Docker** (optional) – for containerized setup.

**4.2.3.Git**

**System prerequisites**

* + **OS**: Windows
  + **Permissions**: Administrator (Windows)
  + **Network**: Internet access (for cloning/pushing remote repos like GitHub, GitLab, Bitbucket).

**Software prerequisites**

* + Text/code editor (VS Code recommended).
  + SSH client (optional, for GitHub/GitLab SSH connections).

**Github account creation:**

### 1. **Sign in to GitHub**

* Go to [GitHub](https://github.com).
* Log in to your GitHub account. If you don't have one, you can sign up for free.

### 2. **Create a New Repository**

* Once logged in, click on the **+** icon in the top right corner of the page.
* Select **"New repository"** from the dropdown menu.

### 3. **Fill in Repository Details**

You'll need to fill out the following fields:

* **Repository Name**: Choose a name for your repository (e.g., my-project).
* **Description** (optional): Provide a brief description of your project (e.g., "A cool web app").
* **Public or Private**: Choose whether your repository will be public (anyone can see it) or private (only you and collaborators can access it).
* **Initialize with README**: If you want to add a README file (which is helpful for explaining your project), check this box.
* **Add .gitignore** (optional): You can choose a template for your .gitignore file, depending on your project's language or framework (e.g., Python, Node, etc.).
* **Choose a License** (optional): You can choose a license if you'd like to specify how others can use your code.

### 4. **Create the Repository**

* Once you've filled everything out, click the **Create repository** button.

### 5. **Clone the Repository (Optional)**

* After the repository is created, you'll be taken to the new repository's page.
* To clone it to your local machine, copy the **clone URL** (either HTTPS or SSH) from the "Code" button.
* In your terminal, run the following command:
* git clone <repository-url>

### 6. **Start Adding Files**

* Now you can start adding files to your repository either through the GitHub website or by pushing files from your local machine.

That's it! You've created a new GitHub repository. Let me know if you need help with the next steps, like pushing files to GitHub.

* + 1. **React.js**

**System prerequisites**

* + **OS**: Windows
  + **Node.js & npm**:
  + Install latest **LTS version of Node.js** (includes npm).
  + Verify with:
  + node -v
  + npm -v
  + **Code Editor**: VS Code recommended.
  + **Browser**: Chrome/Edge/Firefox for developer tools.
  + A.Using Create React App (CRA) [Beginner Friendly]
  + Open terminal → navigate to project folder.
  + Run:
  + npx create-react-app my-app
  + (Here npx comes with npm 5.2+, so no extra install needed.)
  + Go into project folder:
  + cd my-app
  + Start development server:
  + npm start
  + 👉 App runs at <http://localhost:3000>.
    1. **Express.js – Mongoose – Visual Studio Code**
  + **Prerequisites (Theory)**
  + **JavaScript & Node.js** → needed since Express runs on Node.
  + **MongoDB knowledge** → Mongoose is an ODM for MongoDB.
  + **REST API concepts** → to design routes/endpoints.
  + **Visual Studio Code** → editor to write and manage the project.
  + **Node.js & npm installed** → to run server and install packages.
  + **MongoDB installed or Atlas account** → for database.
* **Installation Steps:**

# Clone the repository git clone

# Install client dependencies cd client npm install

# Install server dependencies cd

../server npm install

# **FOLDER STRUCTURE**

Sweat Shine -Work/

│

├── client/ # React Frontend

│ ├── components/ # Reusable UI components

│ └── pages/ # Application pages and views

│ └──styles/

│

├── server/ # Node.js Backend

│ ├── routes/ # API endpoint definitions

│ ├── models/ # Mongoose schemas for MongoDB

│ └── controllers/ # Business logic and request handling

This structure ensures a clear separation between the **frontend (React)** and **backend (Node.js + Express)**, making development and maintenance more efficient.

# **RUNNING THE APPLICATION**

The document provides commands to run a frontend and backend application:

* **Frontend**:
  + **cd client**: This command likely navigates to the frontend directory of the project.
  + **npm start**: Starts the frontend application using Node Package Manager (npm).
* **Backend**:
  + **cd server**: Navigates to the backend server directory.
  + **npm start**: Starts the backend server using npm.

**Accessing the Application**:

* The application can be accessed via http://localhost:3000, which is a local address where the frontend application can be viewed in a browser.

# **API DOCUMENTATION**

API documentation serves as a vital part of any application or service, especially when it comes to integrating with backend systems. It provides a clear and structured guide on how to interact with the application through its **API endpoints**. In this case, the API documentation appears to outline how to interact with user management, project handling, chat functionalities, authentication, and routing security. Here's a breakdown:

### **1. API Endpoints:**

The endpoints listed in the documentation represent various functionalities provided by the backend of the application. Each endpoint corresponds to a specific action or resource. Here's a closer look at what the documentation suggests:

#### **User Endpoints:**

* **/api/user/register**
  + **Purpose**: This endpoint is for registering new users. It likely requires some user data (e.g., email, password, name) to create a new account.
  + **HTTP Method**: Typically, a POST request is used here, sending user data to the server to create a new account.
  + **Response**: It could return a success message or an error message if the registration fails (e.g., email already exists).
* **/api/user/login**
  + **Purpose**: This endpoint handles user login. The user submits credentials (email and password), and if the login is successful, a session token (such as a JWT) is returned.
  + **HTTP Method**: A POST request is commonly used here.
  + **Response**: The response typically includes a **JWT token** or a session identifier if the login is successful. If the credentials are invalid, an error message will be returned.

#### **Project Endpoints:**

* **/api/projects/create**
  + **Purpose**: This endpoint allows users to create new projects within the system. It would typically involve sending project details (such as project title, description, and other parameters).
  + **HTTP Method**: A POST request, sending data in the body to create the project.
  + **Response**: Returns a confirmation message or the details of the newly created project.
* **/api/projects/:id**
  + **Purpose**: Fetch details of a specific project based on its ID.
  + **HTTP Method**: A GET request is usually used to retrieve the details of the project.
  + **Response**: It returns the project data (such as title, description, and other attributes) or an error if the project ID doesn't exist.
* **/api/projects/:id/apply**
  + **Purpose**: Allows a user to apply to a project, which means associating the user with the project in some way (e.g., applying for a job or requesting involvement).
  + **HTTP Method**: A POST request would be used to submit the application.
  + **Response**: Confirmation of the application or error if something goes wrong.

# **USER INTERFACE**

The **User Interface (UI)** section of a flavour diary application provides a description of how the user will interact with the app or website. It includes the structure, design elements, and features that allow users to search for recipes, view ingredients, and create a meal plan, among other actions.

**8.1. Landing Page**

**Overview**: The landing page is the first thing users see when they visit the sweat shine app. It should provide a welcoming experience, easy navigation, and access to key functionalities.

**Key Elements**:

* **Logo/Brand Name** – Clear visibility of the app’s name **FITFLEX** and its branding.
* **Search Bar** – Allows users to quickly find exercises by body part, equipment, or difficulty level.
* **Category Navigation** – Displays workout categories like **Arms, Legs, Chest, Back, Core, and Full Body**.
* **Featured Workouts** – A carousel or grid showcasing trending, recommended, or newly added exercises.
* **Workout Schedules** – Easy access to pre-built daily and weekly fitness routines.
* **Training Tips Section** – Quick advice for safe practice, recovery, and effective results.

**8.2 Workout Search Page**

**Overview:** This page allows users to search for workouts by body part, equipment, or difficulty level. It is designed to help users quickly find exercises suited to their needs.

**Key Elements:**

**Search Filters** – Filters for muscle group (e.g., arms, legs, chest, back, core), equipment (dumbbells, machines, bodyweight), duration, and difficulty (Beginner, Intermediate, Advanced).

**Workout Cards** – Each workout is displayed with a title, image/icon, target muscle, and duration. Clicking a workout card takes the user to the workout details page.

**Sort Options** – Sorting options based on popularity, difficulty level, or newly added exercises.

**8.3 Workout Details Page**

**Overview:** This page shows the full details of a selected workout, including step-by-step instructions, muscles targeted, and safety tips.

**Key Elements:**

**Workout Title** – The name of the exercise (e.g., Barbell Squat, Push-ups).

**Image/Video Demo** – A gallery or video demonstrating the correct workout technique.

**Muscles Targeted** – Information on which body parts the exercise focuses on.

**Instructions** – Detailed steps on how to perform the workout safely and effectively.

**Difficulty & Duration** – Information about the workout’s intensity level and estimated time.

**User Feedback** – A section where users can leave ratings, tips, or personal experiences.

**Save to Routine** – Option for users to add the workout to their personal plan or favorites.

### **UI Design Considerations**:

#### **Visual Design**:

* **Colors:** Use energetic and motivating colors like **blue, black, red, and green**, which are commonly associated with strength, fitness, and health.
* **Typography:** Choose bold, modern, and easy-to-read fonts for exercise names and clear instructional text.
* **Images & Media:** Use **high-quality images or videos** of workouts with proper posture to inspire and guide users.

#### **User Experience (UX)**:

 **Responsive Design:** Ensure the interface adapts seamlessly across desktop, tablet, and mobile devices.

 **Navigation:** Provide clear menus for **Search, Categories, Schedules, and Profile**, making it easy to access workouts and plans.

 **Interactive Elements:** Buttons for **Start Workout, Save to Routine, and Mark as Complete** should be highly visible and easy to use.

#### **3. Accessibility**:

* ***I*nclusive Design:** Maintain strong **color contrast** for readability and accessibility.
* **Font Size & Clarity:** Use scalable, readable font sizes for workout instructions and tips.
* **Alt Text & Labels:** Provide descriptive **alt text for images/videos** and labels for interactive elements to support screen readers.

.

## **TESTING**

**Manual testing during milestones**

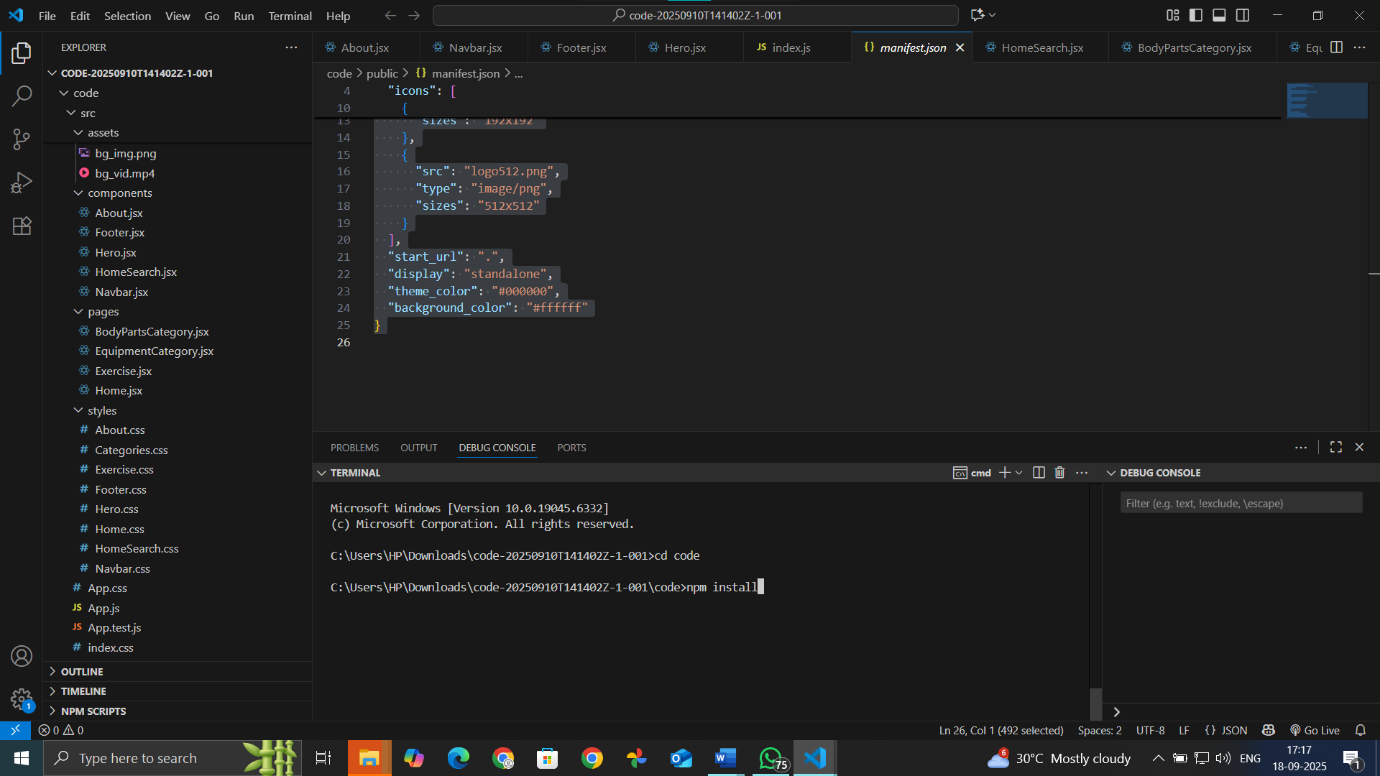
### Manual Testing

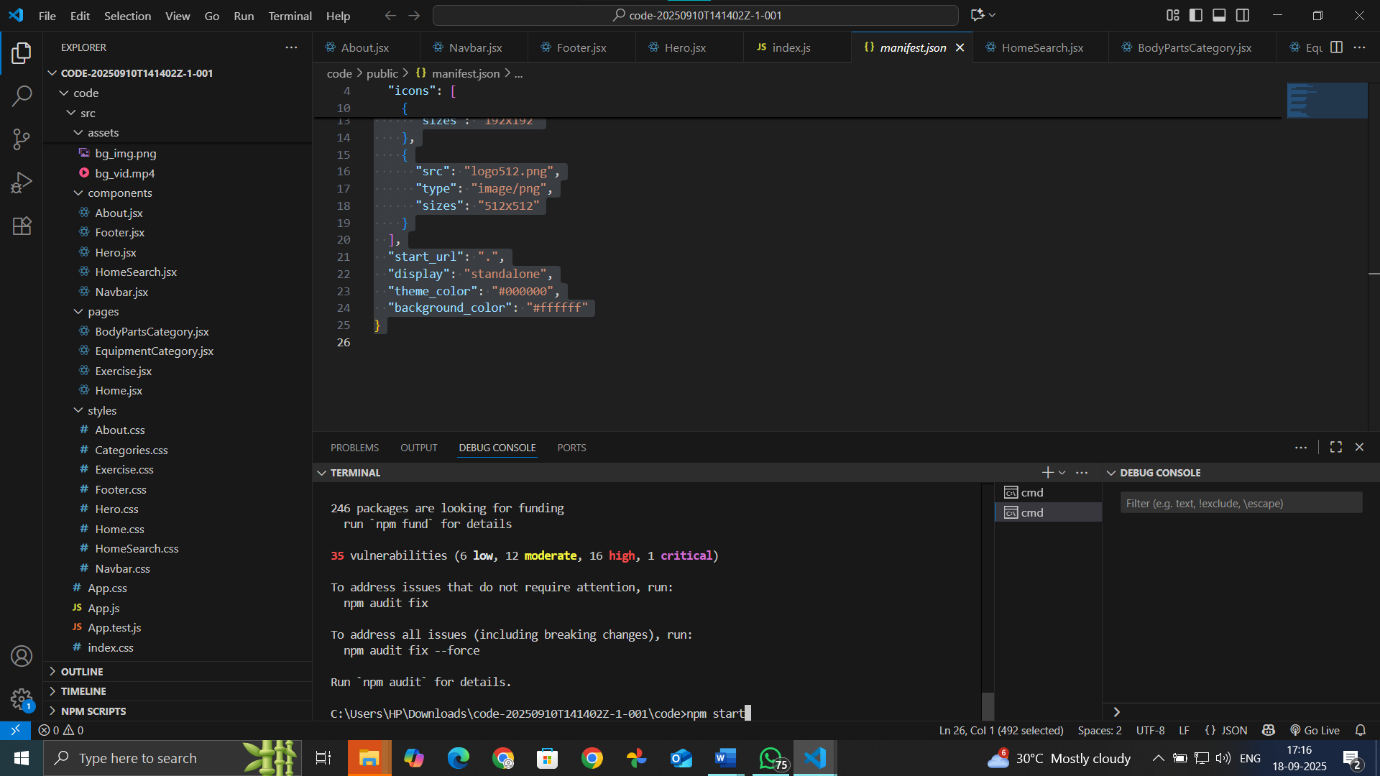
Manual testing is a type of software testing where human testers manually execute test cases to find defects and ensure the software meets its requirements. This process relies on human observation, creativity, and critical thinking.

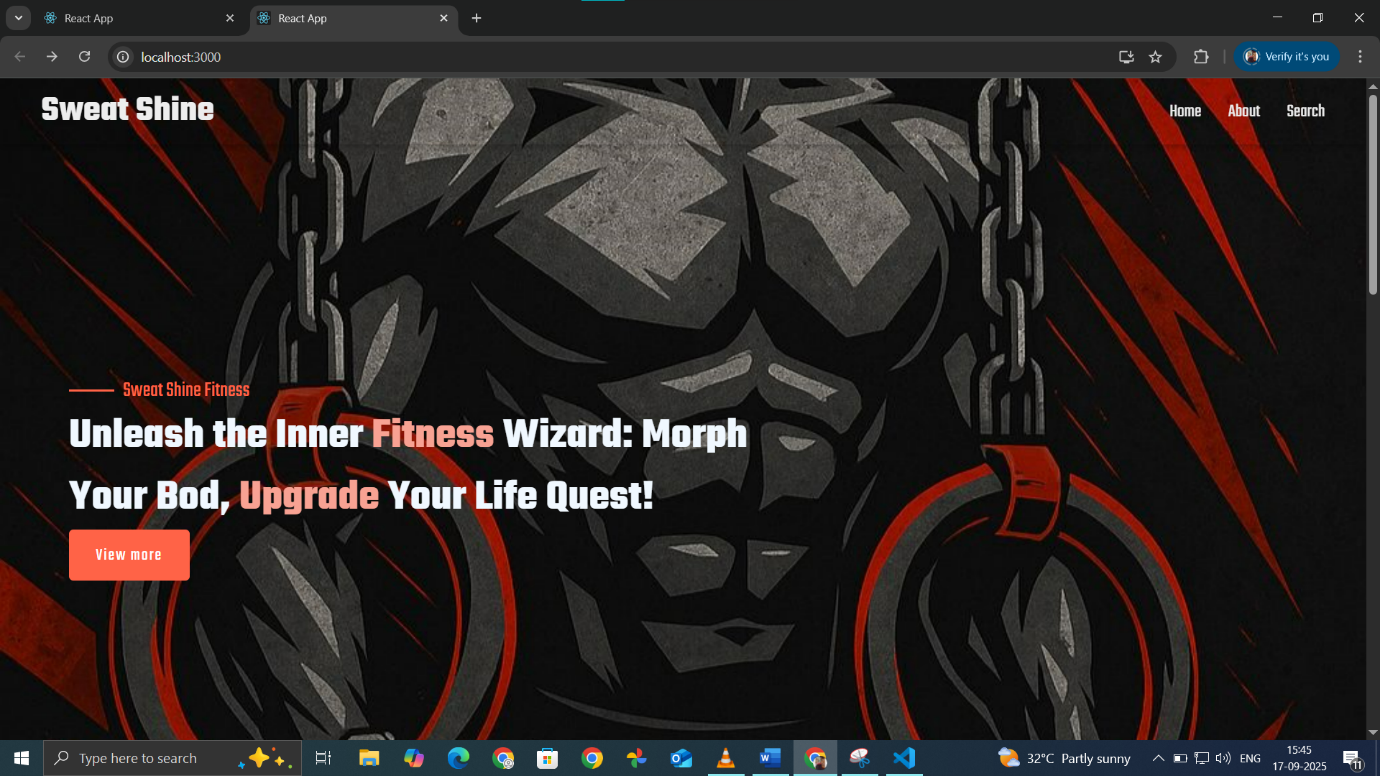
The general steps in manual testing include:

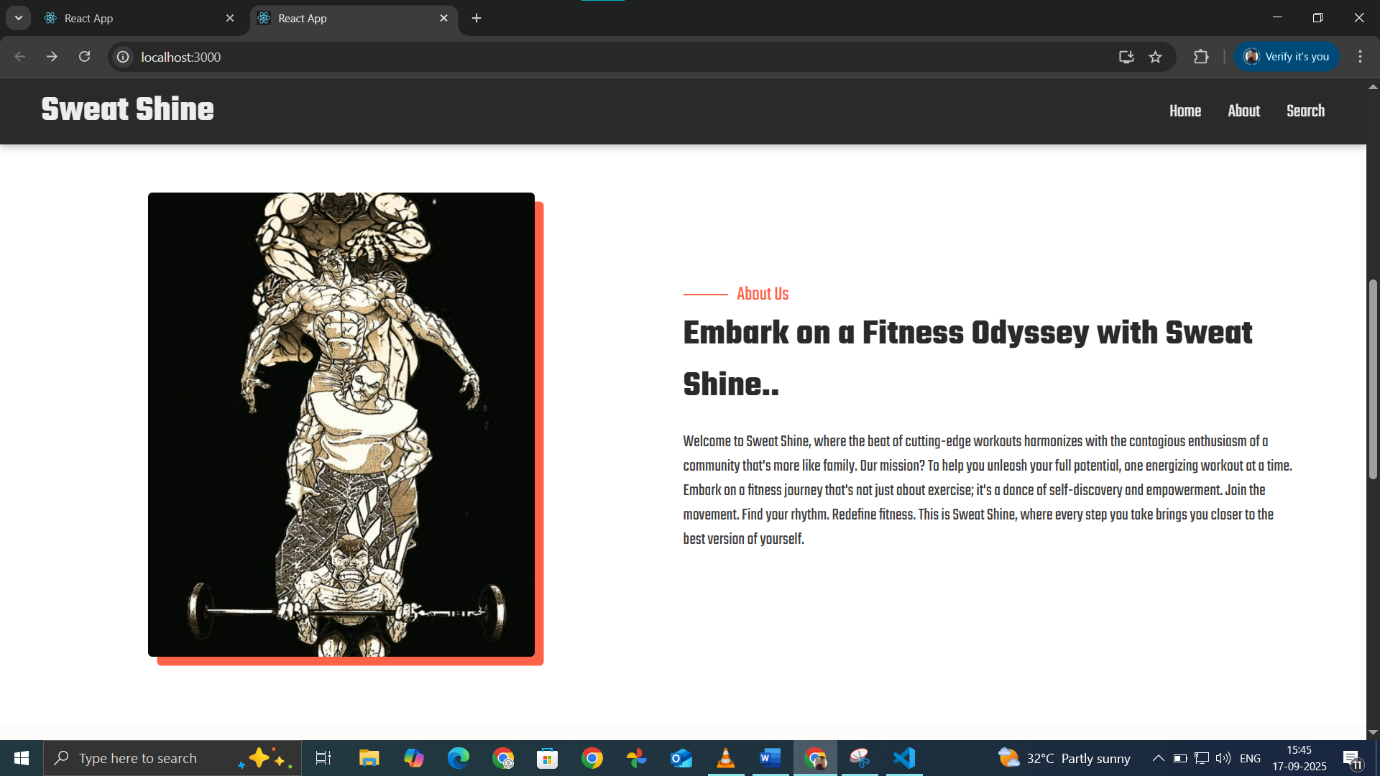
1. **Understanding Requirements**: Reviewing and understanding all software requirements, user stories, and design files.
2. **Creating a Test Pla**n: Developing a roadmap for the entire testing process, including objectives, scope, and resources.
3. **Writing Test Case**s: Creating detailed step-by-step instructions for testing specific features and scenarios.
4. **Setting up the Test Environment**: Installing the necessary software, databases, and other tools to perform the tests.
5. **Executing Test Cases:** Manually interacting with the software to verify it behaves as expected and documenting the results.
6. **Logging Defects:** Reporting any issues or bugs found during testing into a defect tracking tool.
7. **Retesting and Regression Testing:** Re-checking the areas where bugs were fixed and running regression tests to ensure new issues were not introduced.
8. **Analyzing Results and Reporting**: Reviewing the entire process, documenting findings, and creating a final report.

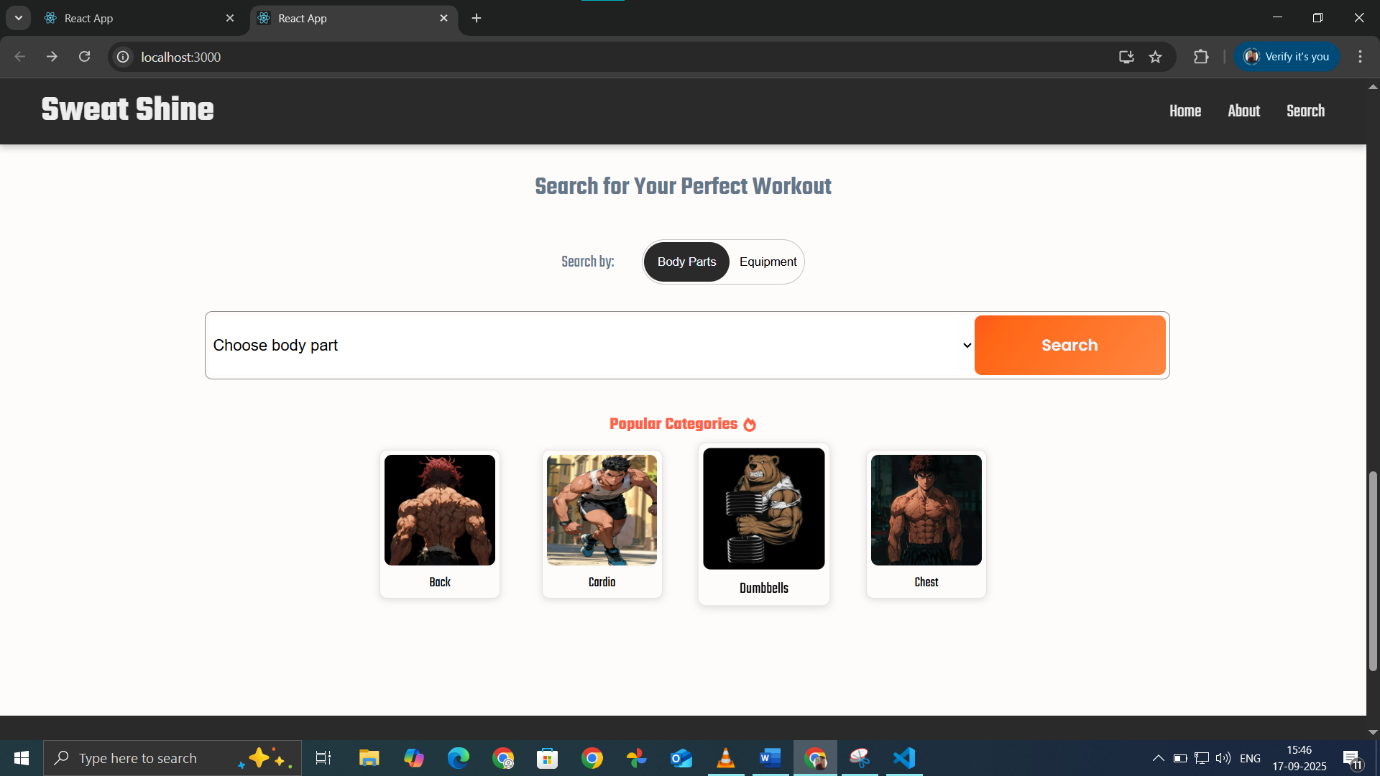
## **SCREENSHOTS**

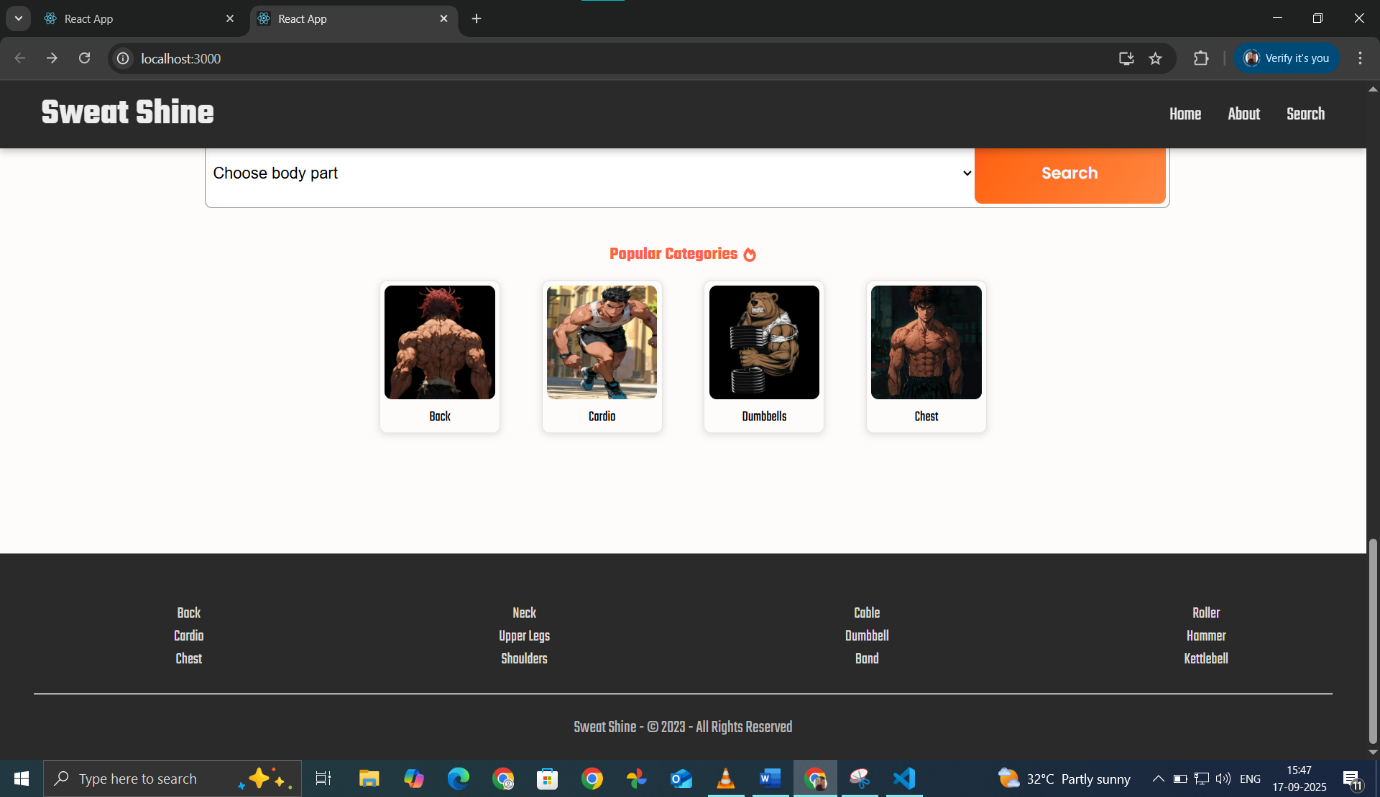


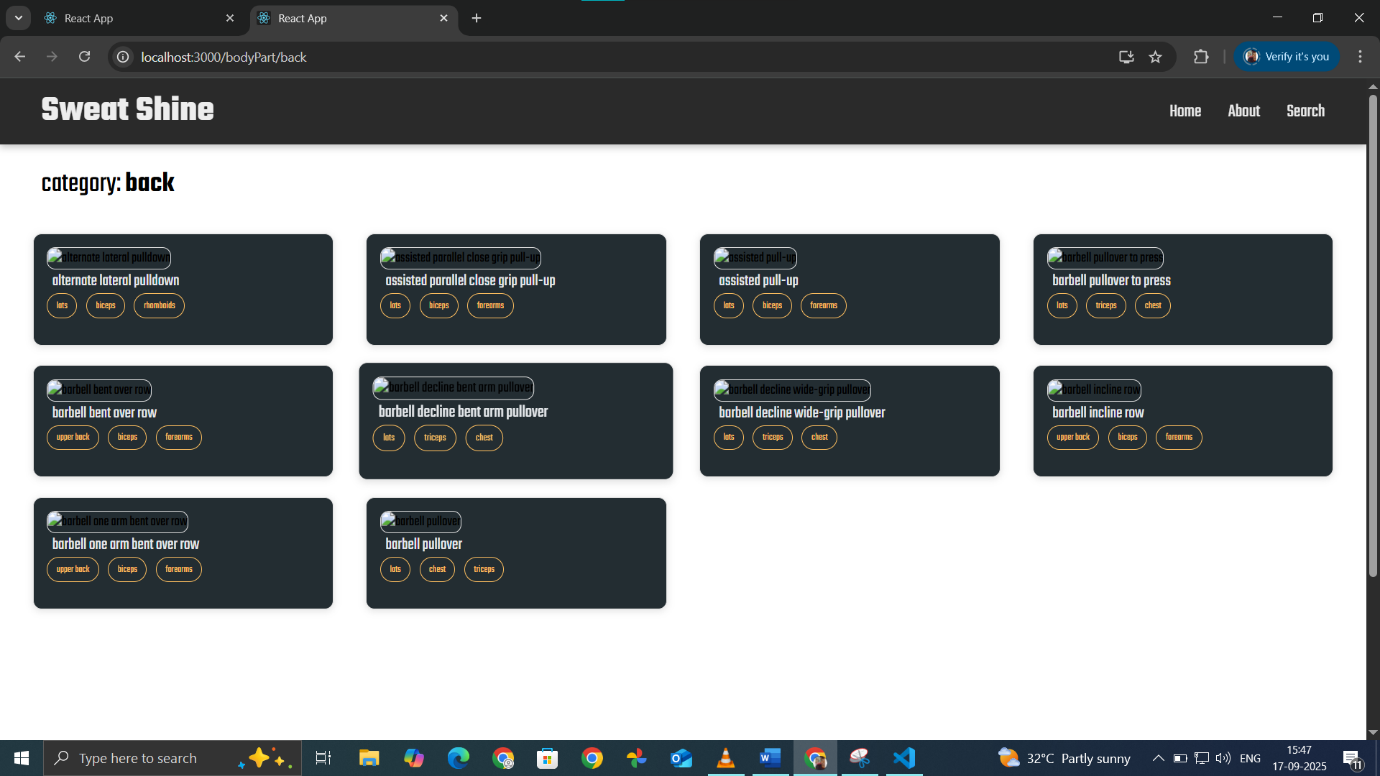


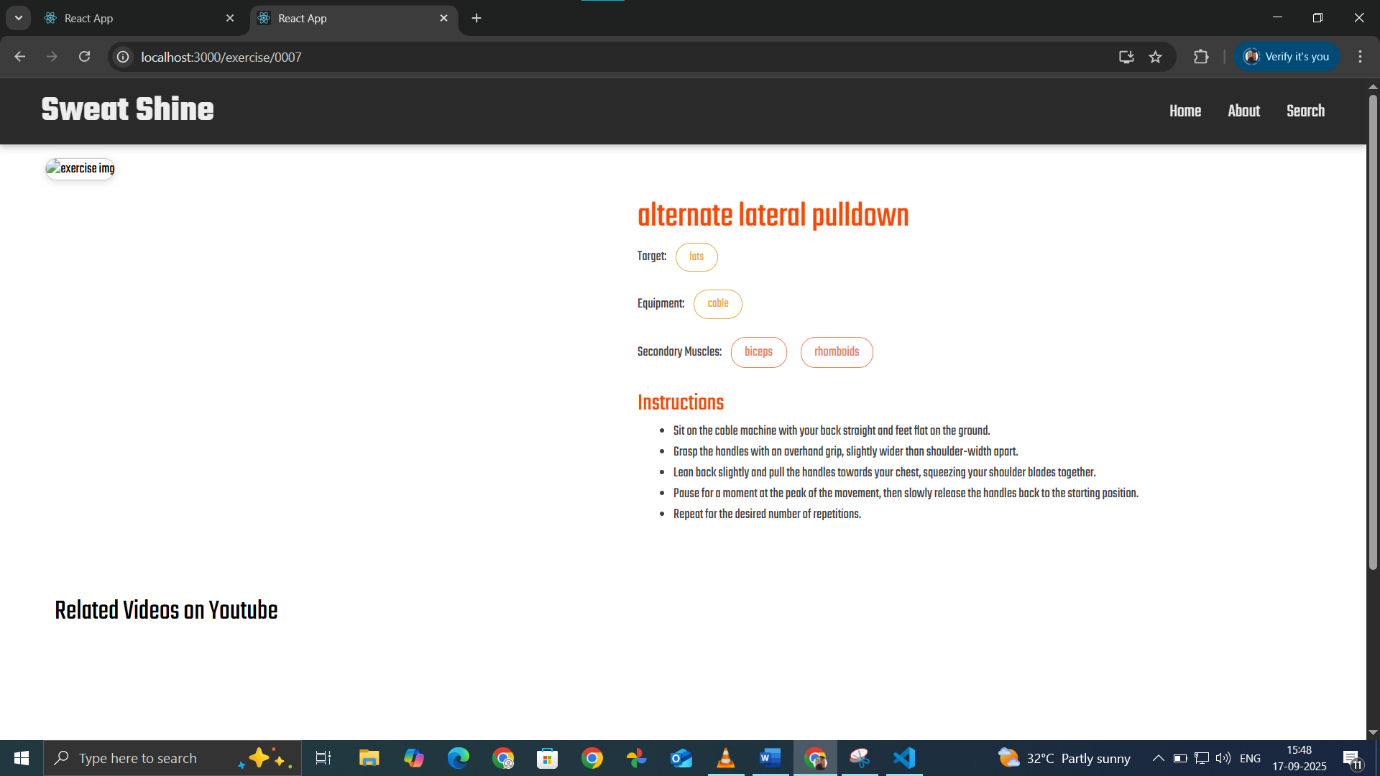


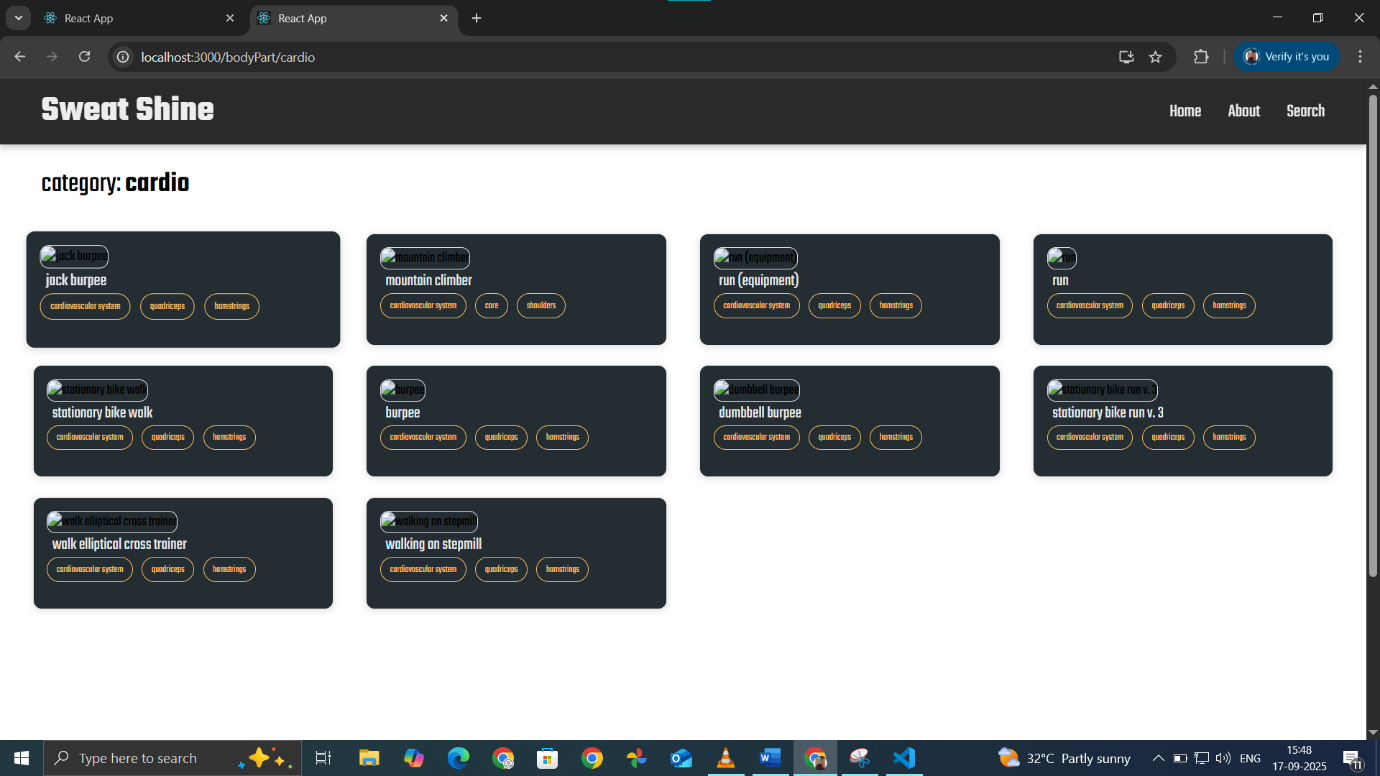


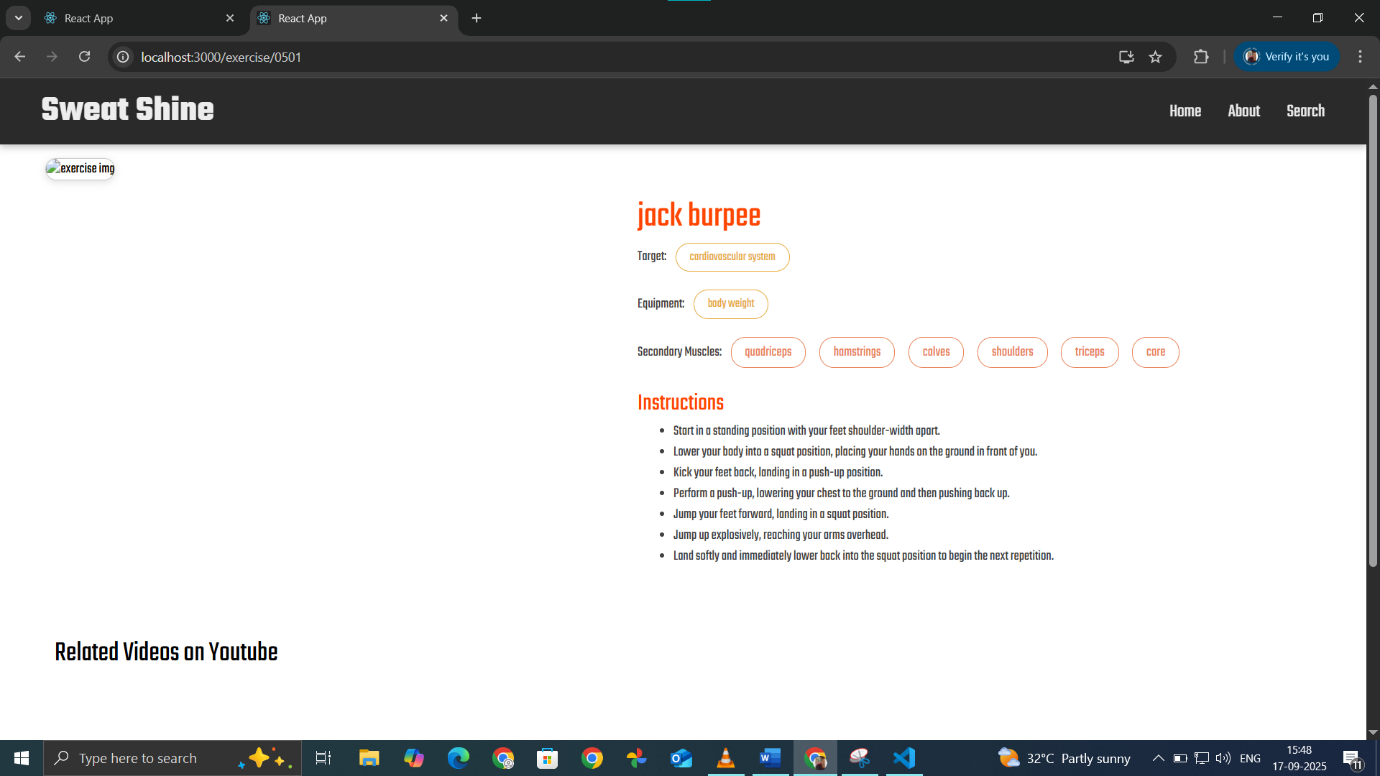


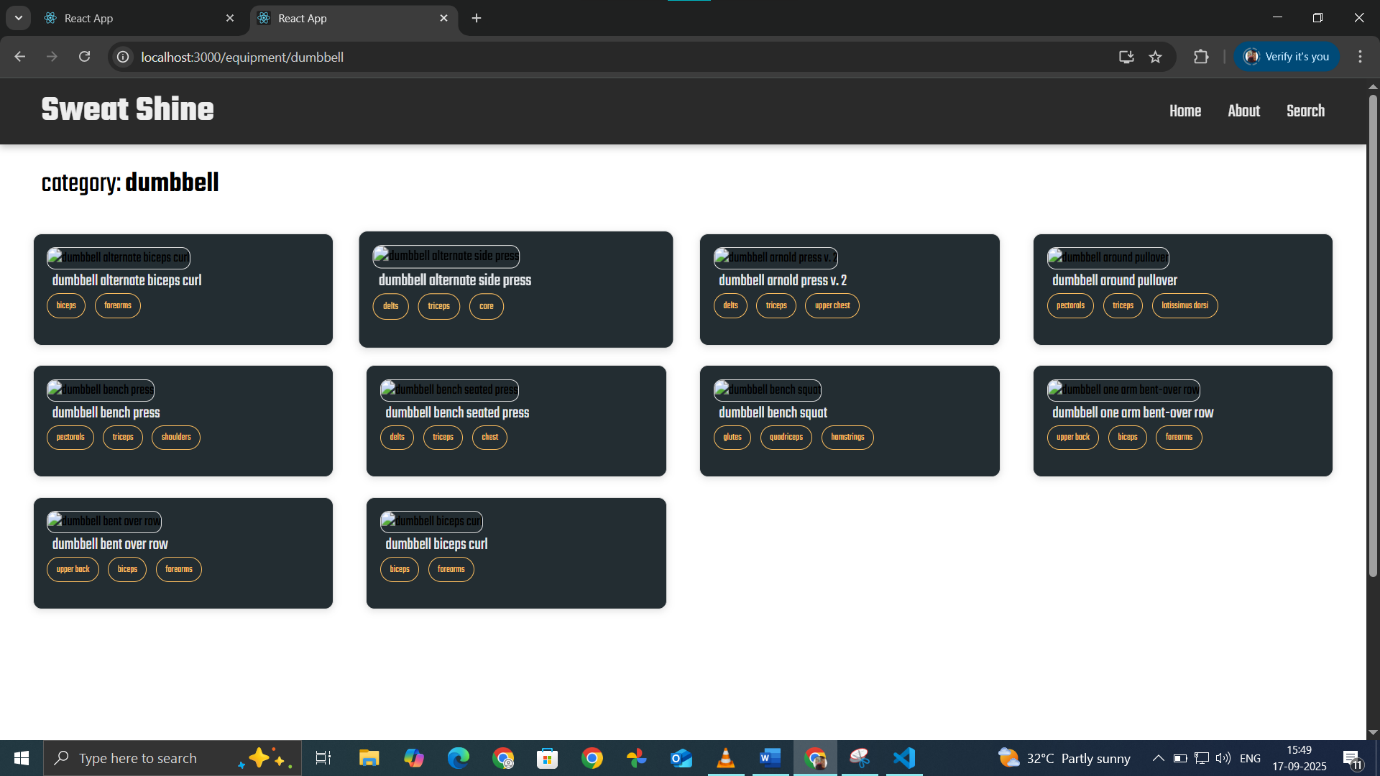


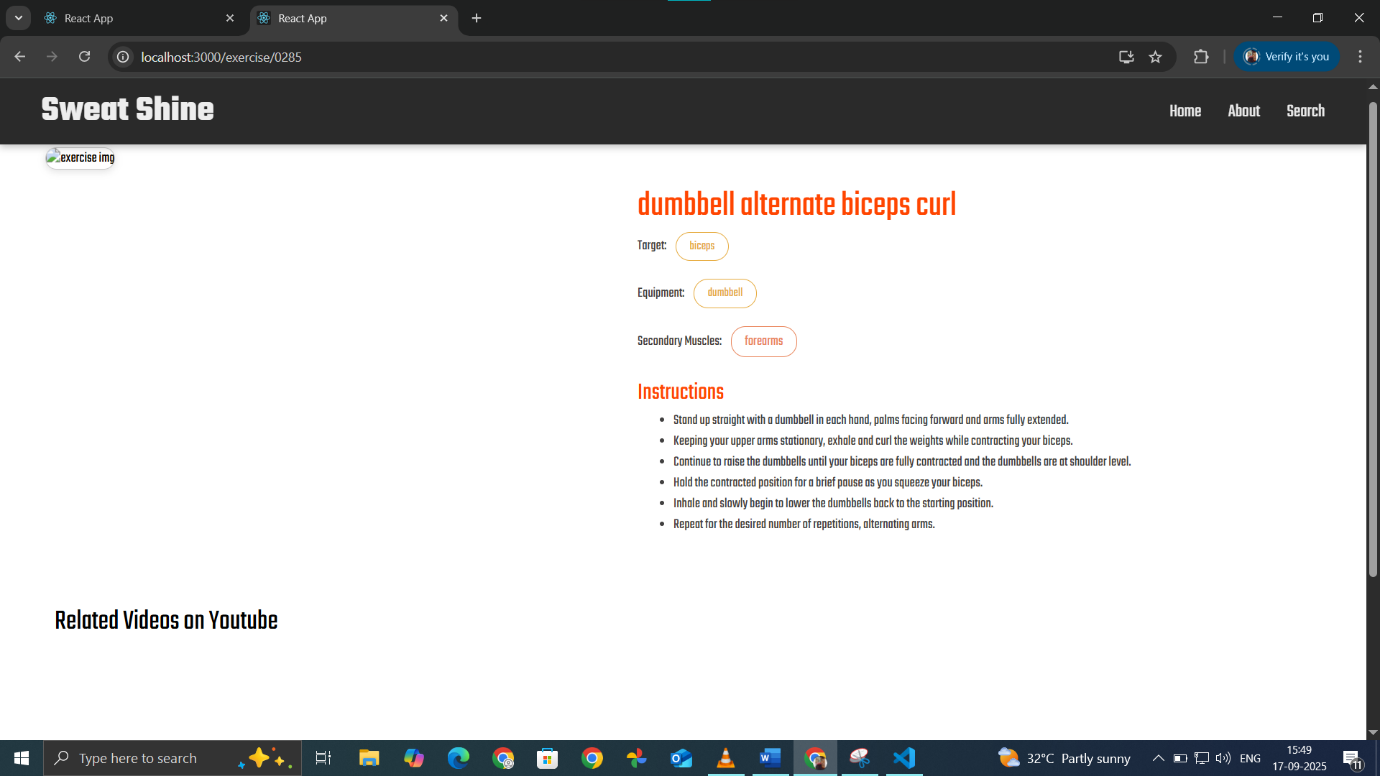


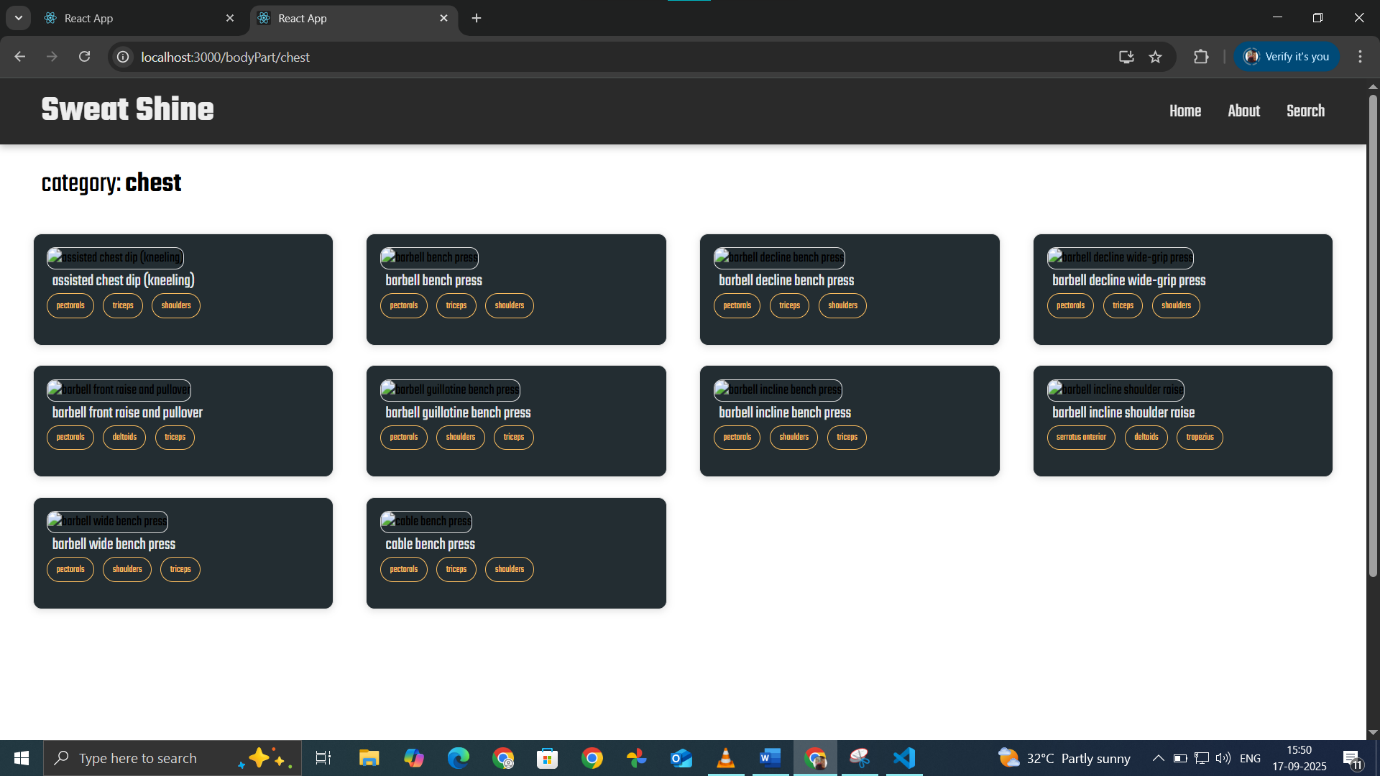


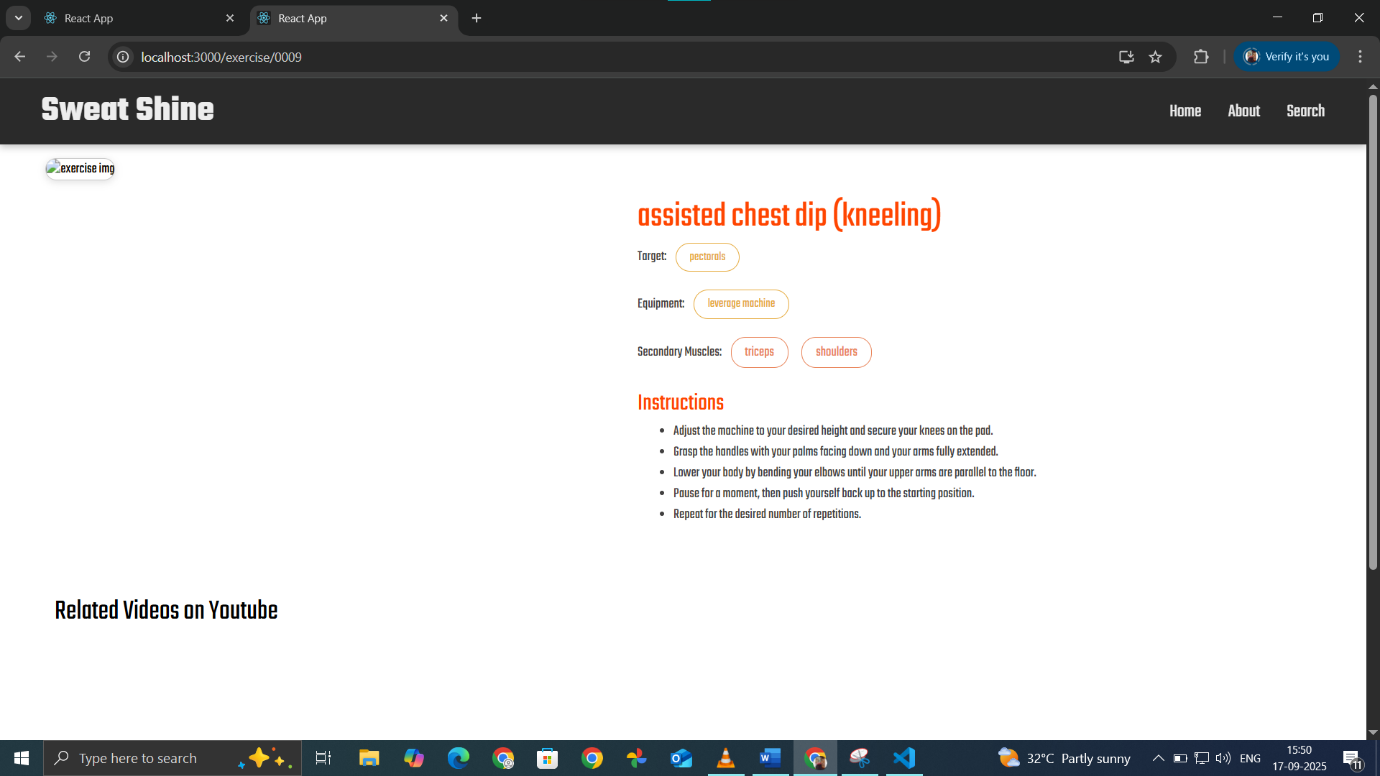












## **KNOWN** **ISSUES**

At the current stage, Sweat Shine has a few limitations that need to be addressed in future updates. The app mainly provides text and image-based exercise instructions, which may not be enough for beginners who prefer detailed video demonstrations. Personalization features such as customized workout recommendations based on user goals and fitness levels are not yet fully implemented, which can limit the user’s training experience. The platform also requires a stable internet connection, as offline access to workouts and schedules is not supported. Additionally, accessibility features such as voice assistance, screen reader support, and adjustable text sizes are still in development. These issues do not prevent basic use of the application but highlight areas for improvement in upcoming versions.

## **12.FUTURE ENHANCEMENT**

In future updates, Sweat Shine aims to provide a richer and more interactive workout experience by introducing **video demonstrations, voice guidance, and animated tutorials**. These features will make it easier for beginners to understand the correct form and technique, reducing the chances of injuries and increasing workout effectiveness. Alongside this, the platform will offer **step-by-step guided sessions**, where users can follow along in real time, making the app feel more like a personal trainer.

The next phase of development will also focus on **personalization and accessibility**. Users will be able to create fitness profiles where they can input their goals, current fitness levels, and available equipment, allowing the app to generate **custom workout plans** tailored to their needs. Accessibility improvements, such as **screen reader support, voice commands, adjustable font sizes, and high-contrast color modes**, will make the platform inclusive for a wider range of users, including those with disabilities.

To improve overall convenience, Sweat Shine will introduce **offline access** so users can download workouts and schedules in advance, ensuring uninterrupted training even without an internet connection. Technical enhancements such as **faster loading times, smoother navigation, and improved device compatibility** will also be prioritized to provide a seamless experience across different platforms.

Finally, Sweat Shine will include features designed to **motivate and retain users over the long term**. This includes **progress tracking dashboards, performance analytics, daily reminders, and gamification elements** like badges, leaderboards, and fitness challenges. By adding these engaging tools, the platform will not only help users stay consistent with their fitness routines but also foster a sense of achievement and community. These enhancements will transform Sweat Shine from a simple workout guide into a **comprehensive digital fitness companion**.

## **13**.**CONCLUSION**

Sweat Shine, developed under the FITFLEX project, is designed to be a comprehensive and user-friendly fitness platform that provides structured workouts, step-by-step exercise instructions, and guidance suitable for all fitness levels. The application empowers users to perform exercises safely and effectively, helping beginners build confidence and advanced users refine their routines. While the current version offers a solid foundation, certain limitations remain, such as the lack of offline access, fully personalized workout plans, and advanced accessibility features. Addressing these limitations will be a priority in future updates to ensure a seamless and inclusive fitness experience.

Looking ahead, Sweat Shine will integrate high-quality video demonstrations, personalized fitness recommendations, offline capabilities, and gamification features like challenges, badges, and progress tracking. These enhancements will make the platform more engaging, motivating, and effective, transforming it from a simple workout guide into a comprehensive digital fitness companion. Ultimately, Sweat Shine aims to promote healthier lifestyles, encourage consistency in workouts, and support users in achieving their fitness goals while providing a safe, enjoyable, and interactive fitness experience.