Project Title: Health & Fitness Tracker

Name: Venkatesh Golla

Date: 19-03-2025

Batch Name: C1

Instructor: Mr. Naveen Kumar

Table of Contents

Overview of Problem Statement:

This project, **Health & Fitness Tracker**, aims to provide users with a comprehensive platform to log their workouts, track calories, and visualize fitness trends using an intuitive and interactive interface. By leveraging modern web technologies such as **React, Redux, ASP.NET Core, and MS SQL Server**, the application ensures a seamless and efficient user experience.

Project Goals and Objectives

Goals:

1. Develop a full-stack **Health & Fitness Tracker** web application.
2. Provide a user-friendly interface for tracking workouts and calorie intake.
3. Implement **data visualization** to help users analyze fitness trends.
4. Ensure secure and efficient storage of user data using **MS SQL Server**.
5. Improve user engagement with insights based on logged data.

Objectives:

1. **Workout Logging** – Allow users to record details of their workouts, including exercise type, duration, and calories burned.

2. **Calorie Tracking** – Enable users to track daily calorie consumption and expenditure.

3. **View Fitness Trends** – Implement Redux to fetch and display historical fitness data in an interactive manner.

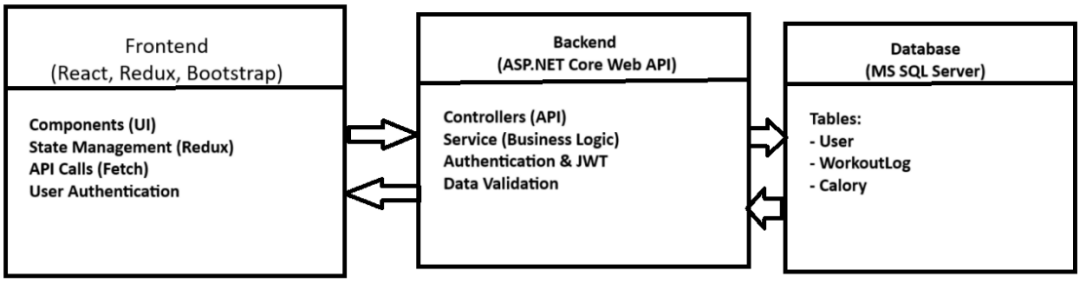
4. **Secure User Authentication** – Provide a secure login and registration system using ASP.NET Core authentication.

Frontend & Backend Architecture

System Architecture Overview

The **Health & Fitness Tracker** follows a **client-server architecture**, ensuring scalability, maintainability, and modularity. The application is structured into three primary layers:

1. **Frontend (Client-Side):** Developed using **React.js**, the frontend provides a dynamic and interactive user interface. It communicates with the backend via RESTful APIs.
2. **Backend (Server-Side):** Built using **ASP.NET Core Web API**, the backend handles business logic, authentication, and database interactions.
3. **Database (Data Storage): MS SQL Server** is used for persistent storage, ensuring data integrity and efficient query handling.



Overview of Chosen Technology Stack

Frontend: React.js

- **React.js** – A fast and efficient JavaScript library for building reusable UI components.
- **Redux** – Manages global state for fitness trends, workouts, and user authentication.
- **React Router** – Enables navigation between pages.
- **Bootstrap** – Provides responsive design and styling for the UI.

Backend: ASP.NET Core Web API

🔹 **ASP.NET Core** – A cross-platform, high-performance framework for building RESTful APIs.
🔹 **Entity Framework Core** – ORM (Object-Relational Mapper) to interact with the SQL Server database.
🔹 **JWT Authentication** – Secure user authentication and authorization.

Database: Microsoft SQL Server

🔹 **MS SQL Server** – A relational database for storing user data, workouts, and calories.
🔹 **EF Core Migrations** – Handles database schema updates efficiently.

Communication & API Handling

✓ The frontend communicates with the backend through **RESTful APIs**.
✓ **Fetch/Axios** is used for making HTTP requests.
✓ **CORS Policies** are implemented to allow secure cross-origin requests.

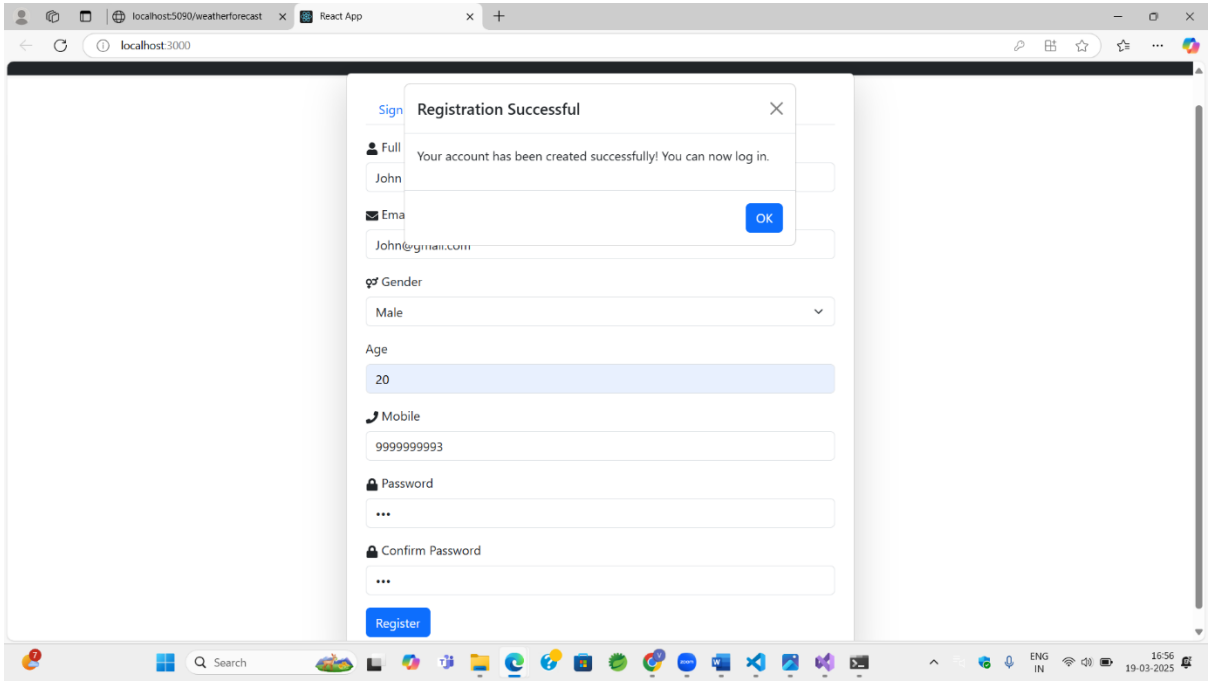
I. Component Breakdown

1. Routing and UI Components (Bootstrap + Custom)

1.1 Sign Up:

Here I will demonstrate how my project flow goes:

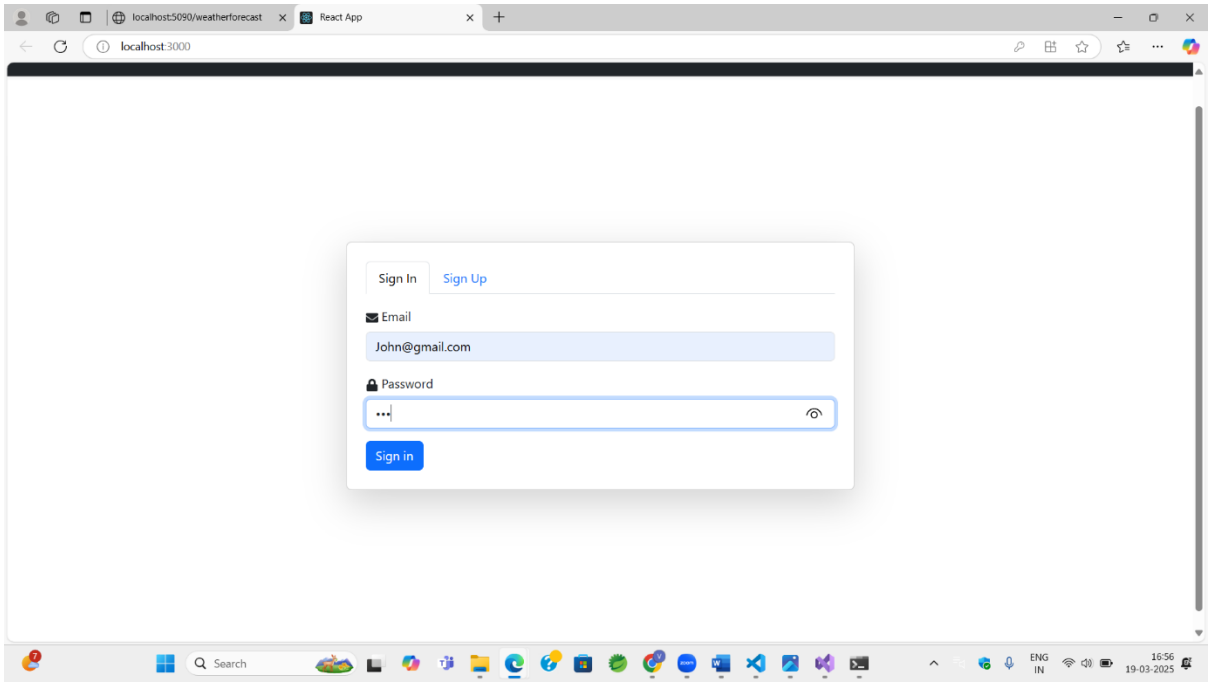
First the user need to sing up into to the website providing the given details. The user details will be stored in User table using dotnet web api core. The password is first converted into hash value using SHA-256 algorithm and then stored in database Users table.



1.2 Sign In:

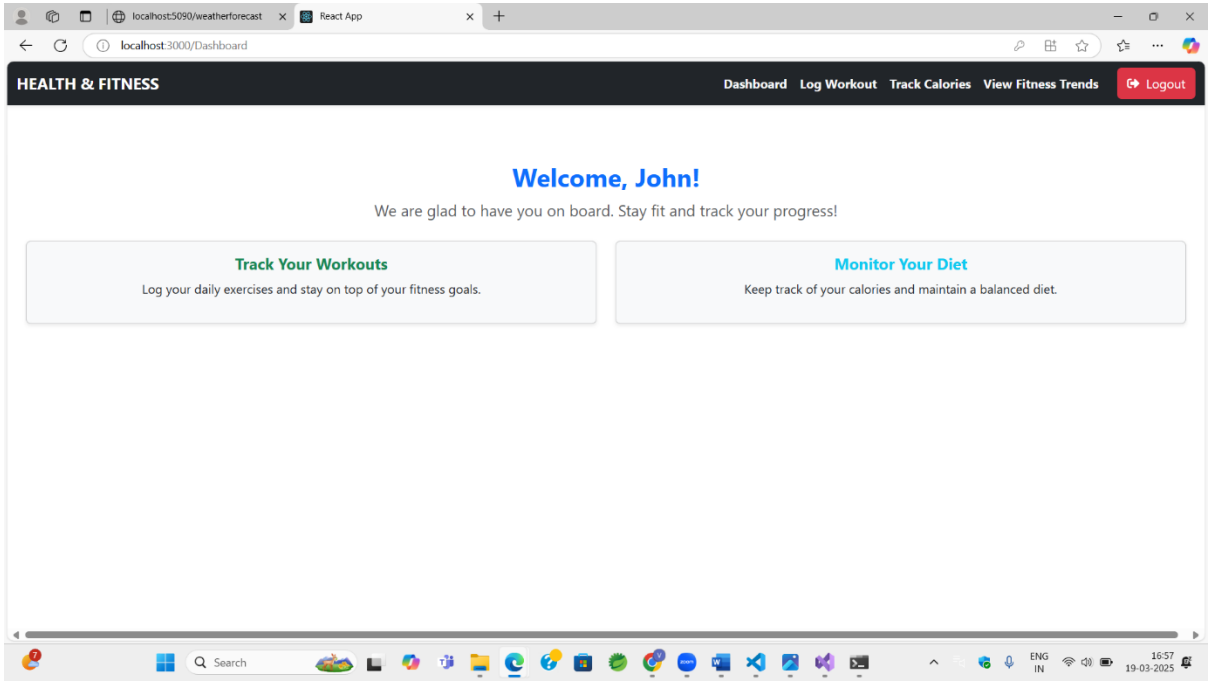
Once registered user can log in into the website. When the user logged in the fetch api gives a authentication token back to the react, that token is stored in local storage, later this token is used for authenticating the user for different kinds of interactions with dotnet apis.

Along with token it stores the user details like id, name for showing at different pages on the website.



1.3 Dashboard Page:

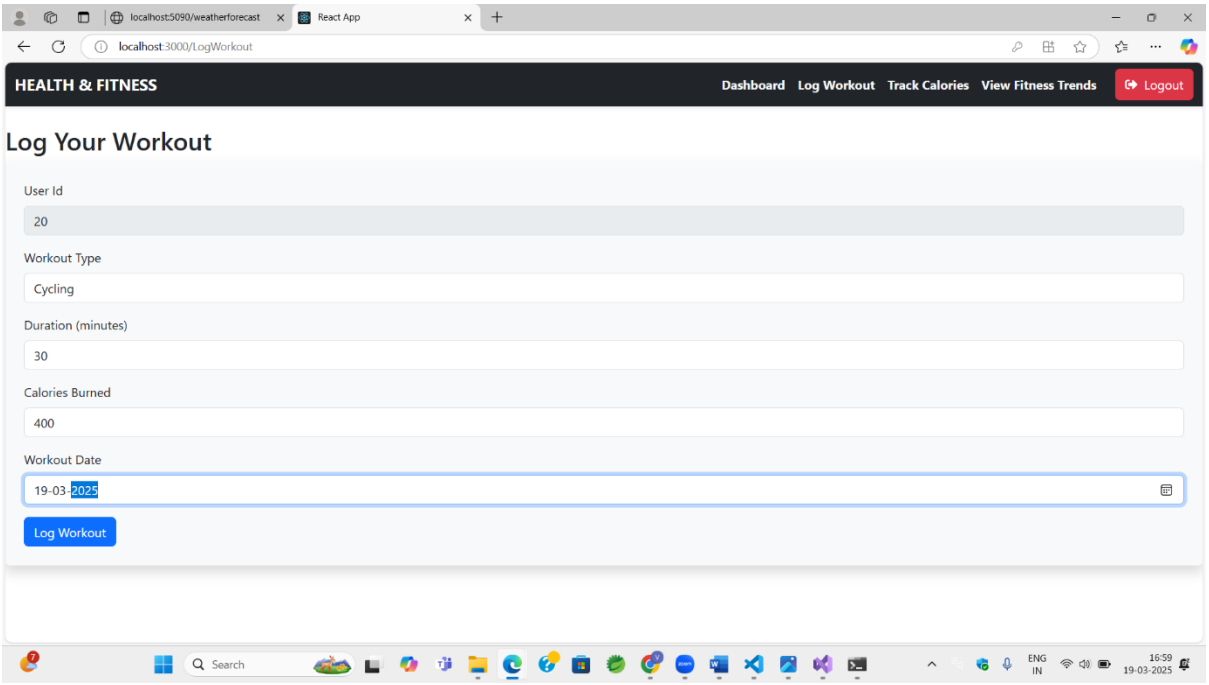
When the user logged in into the website Dashboard Page will be shown with the user name with some welcome message



The user now can navigate through different web pages using the navigator, It uses react protected routers for navigating

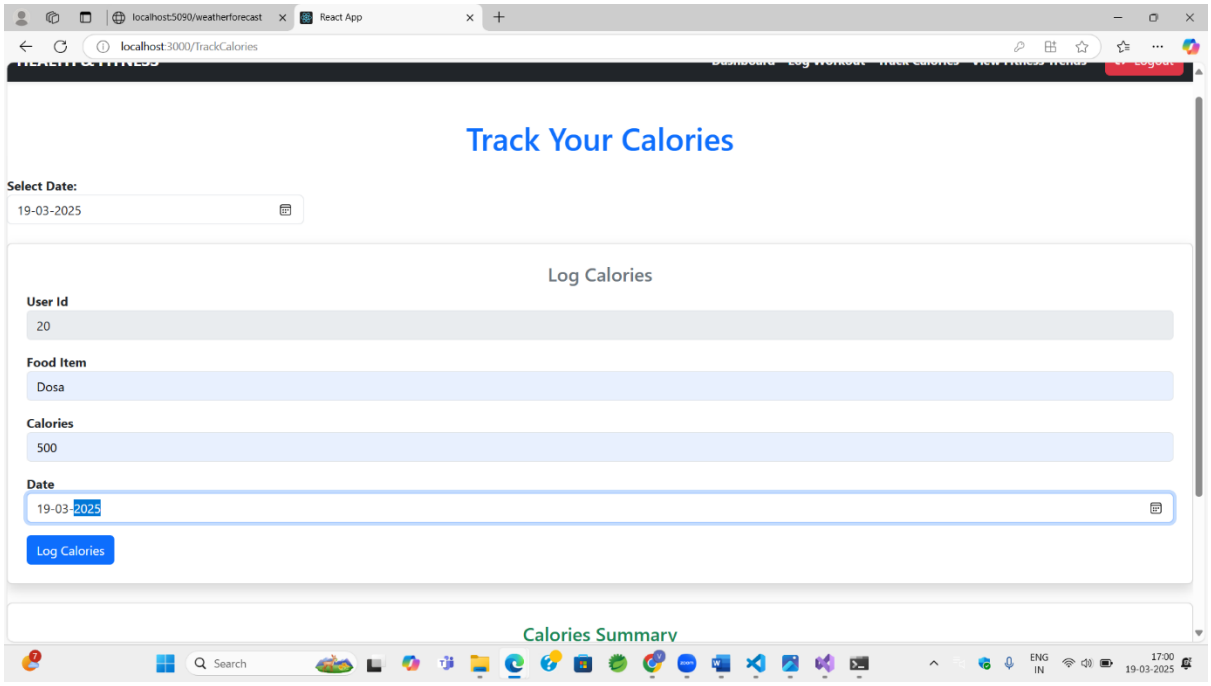
1.4 Log Workout Page:

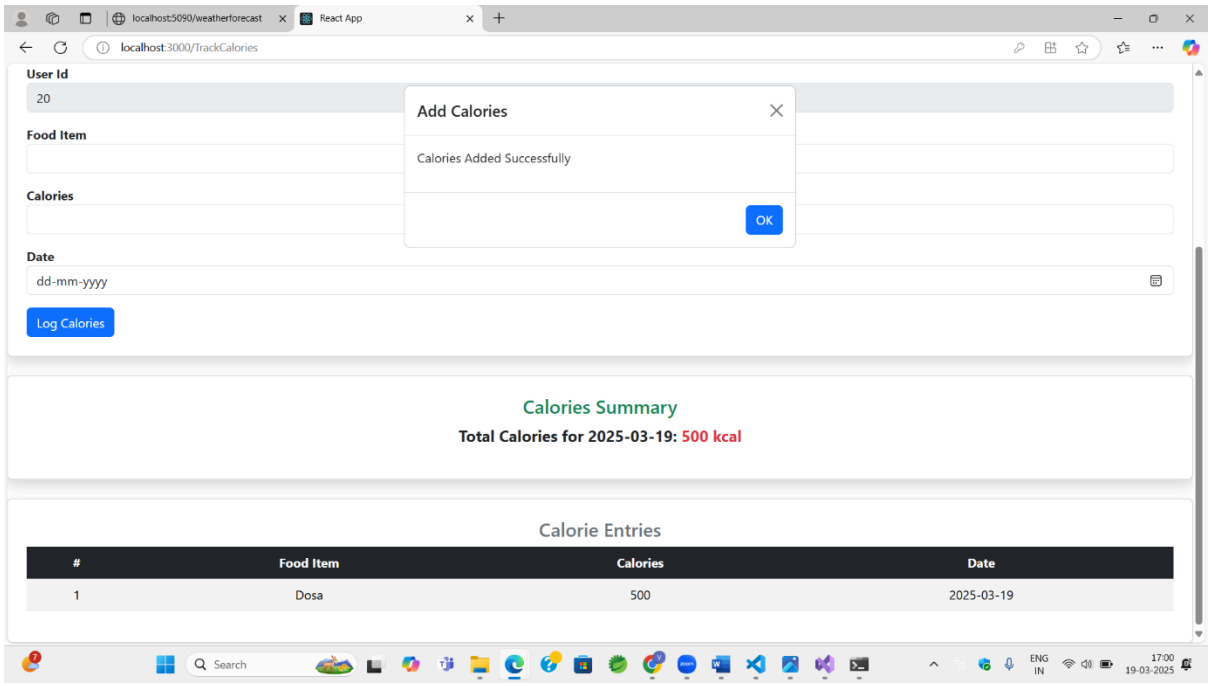
Log workout page allows the user to log the workout details that he/she has done, the details will be stored in WorkoutLog table through post Api call. While storing the details in the table the Jwt authentication is also will be done at dotnet server.



1.5 Track Calories Page:

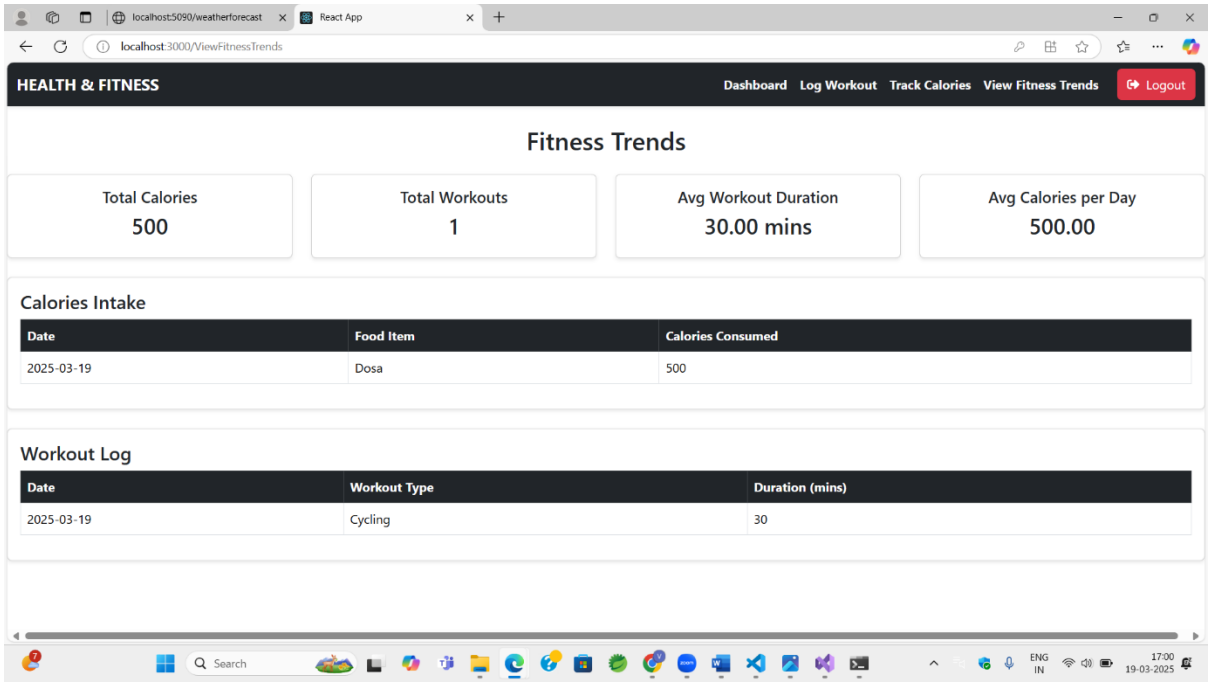
Track Calories page allows user to enter the details of calories intake and also shows the total calories consumed on the selected data along with the data as follows


localhost:5090/weatherforecast
React App
localhost:3000/TrackCalories
Track Your Calories
Select Date:
19-03-2025
Log Calories
User Id
20
Food Item
Dosa
Calories
500
Date
19-03-2025
Log Calories
Calories Summary


localhost:5090/weatherforecast
React App
localhost:3000/TrackCalories
User Id
20
Food Item
Calories
Date
dd-mm-yyyy
Log Calories
Add Calories
Calories Added Successfully
OK
Calories Summary
Total Calories for 2025-03-19: 500 kcal
Calorie Entries
Food Item Calories Date
1 Dosa 500 2025-03-19

1.6 View Fitness Trends Page:

View Fitness Trends page shows the summary of the calories consumed and the calories burned through workouts. The page shows the details taking from redux store.



2. Routing (React Router)

- Handles navigation between pages.
- Routes are protected based on authentication.

3. Redux manages global state such as:

- **Fitness Data:** Stores fetched workout and calorie logs.
- **UI State:** Manages loading states, error handling, etc

II. API Design & Authentication

The backend is built using **ASP.NET Core Web API** and follows **RESTful API principles**.

2.1 API Design and Endpoints

User Authentication

Method	Endpoint	Description
POST	/api/Login	Logs in a user and returns a JWT token.
POST	/api/Users	Registers a new user.

Authentication:

- JWT (JSON Web Token) is used for secure authentication.
- Users must include the token in Authorization: Bearer <token> header for protected routes.

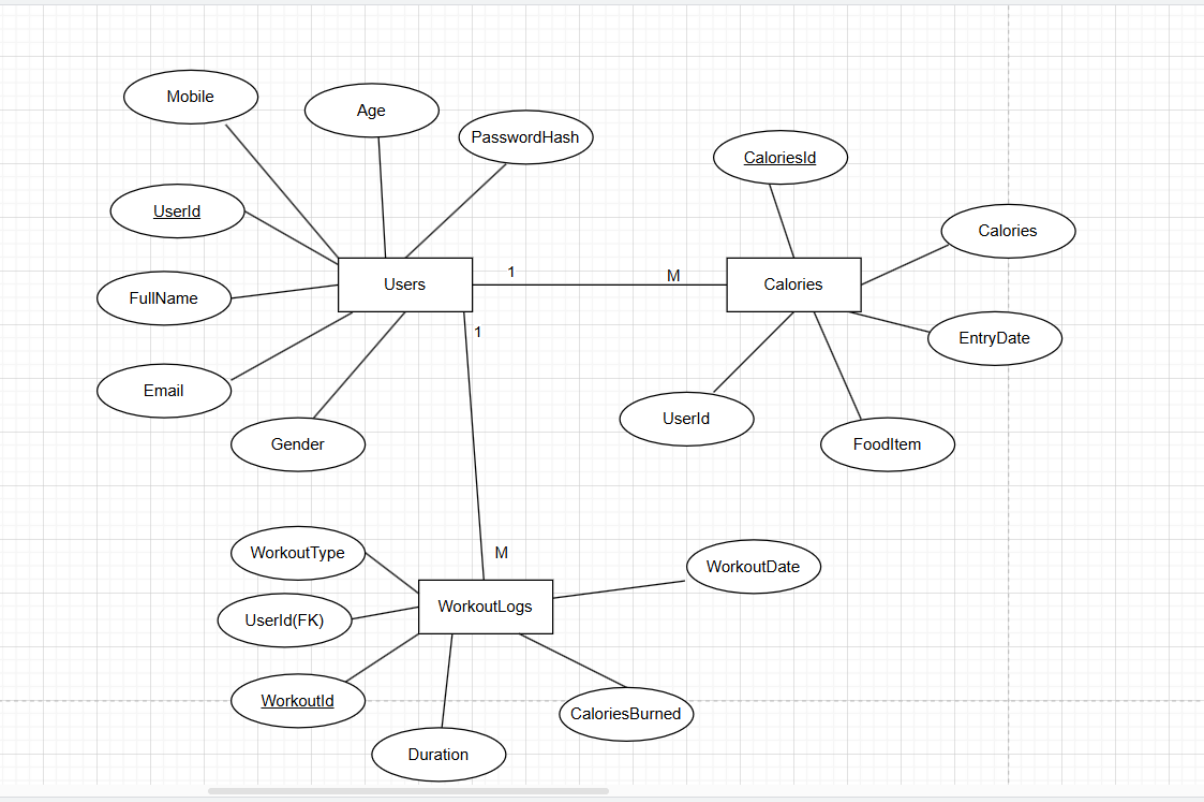
Fitness Data APIs

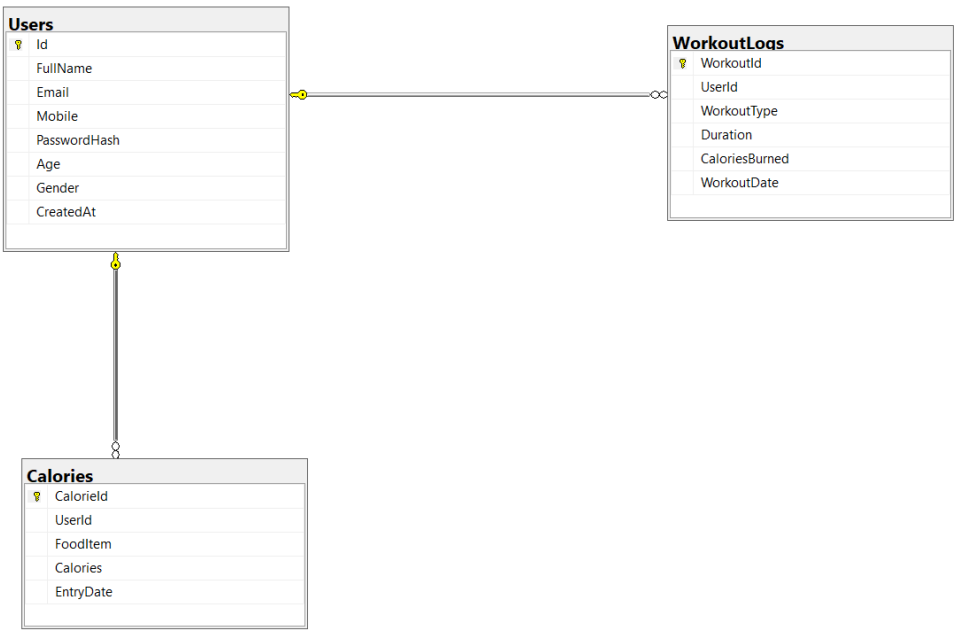
Method	Endpoint	Description
GET	/api/Calories	Retrieves calorie burn records.
POST	/api/Calories	Logs new calorie data.
GET	/api/LogWorkout	Fetches workout history.
POST	/api/LogWorkout	Logs a new workout session.

2.2 Authentication Mechanism

1. **User logs in** → Backend verifies credentials.
2. **JWT token is generated** and returned to the frontend.
3. Token is stored in **localStorage**.
4. For protected API requests, the frontend sends the token in headers.

Database Design & Storage Optimization:





Indexing for Faster Queries

- **Primary Keys**: Indexed automatically.

- **Foreign Keys**: Indexed for faster JOIN operations.
- **Composite Indexing**: (UserID, Date) on Calories and LogWorkout to speed up queries.

Data Normalization

- **Avoids data redundancy** by splitting data into multiple tables.
- **Uses Foreign Keys** to maintain consistency.

Git Repository Links:

Frontend: https://github.com/venkatesh82003/health-fitness-client1

Backend: https://github.com/venkatesh82003/Health-Fitness-Backend