Final Capstone Project Report

Project Title:-	Health and Fitness Tracker
Name:-	Anurag Jadhav
Date of Submission :-	20/03/2025
Batch Name :-	
Instructor Details :-	Mr. Deepak and Mr. Ramesh Nediyadth

1. Introduction

The Health & Fitness Tracker is a full-stack web application developed using React (with Redux) for the frontend and ASP.NET Core for the backend. The project aims to help users track workouts, monitor calories, and analyze fitness trends efficiently.

2. Problem Definition and Objectives

Problem Statement:

Managing fitness data manually can be inefficient and error-prone. Users need a digital platform to log workouts, track calorie intake and expenditure, and analyze trends.

Objectives:

- Provide an interactive interface for users to log workouts and calorie intake.
- Offer fitness insights through graphical visualizations.
- Ensure secure authentication and authorization.
- Optimize data storage for fast and efficient queries.

3. Frontend & Backend Architecture

Technology Stack:

- **Frontend:** React (Redux for state management, React Router for navigation)
- **Backend:** ASP.NET Core Web API
- **Database:** Microsoft SQL Server
- **Authentication:** JWT-based authentication

System Design Diagram:

(A high-level system diagram should be included here.)

4. Component Breakdown & API Design

Frontend Components:

- **Authentication:** Registration, Login pages
- **Dashboard:** Summary of user fitness data
- **Workout Tracker:** Logging workouts
- **Calorie Tracker:** Tracking calorie intake and expenditure
- **Insights:** Graphical visualization of fitness trends

API Design & Endpoints:

Endpoint	Method	Description
/api/user/register	POST	Register a new user
/api/user/login	POST	Authenticate user and return JWT token

/api/workouts	GET	Fetch user workouts
/api/workouts	POST	Add a new workout
/api/calories	GET	Fetch calorie records
/api/calories	POST	Add calorie intake

5. Database Design & Storage Optimization

Optimization Techniques:

- Indexing for faster query execution.
- Proper normalization to eliminate data redundancy.
- Optimized API queries to prevent overloading the database.

6. Project Demonstration

A detailed demonstration of the project, including:

- User authentication
- Adding workouts and calorie records
- Viewing insights and trends
- Testing API endpoints using Swagger/Postman

7. Supporting Files & Deployment

Frontend Code:

- **File Name:** `Frontend_HealthFitnessTracker.zip`
- **Contents:** Complete React source code

```
src/
 ├── assets/              # Static assets like images, icons, etc.
 ├── components/          # Reusable UI components
 │   ├── Auth/
 │   │   ├── Login.jsx
 │   │   ├── Register.jsx
 │   ├── Dashboard/
 │   │   ├── Dashboard.jsx
```

```
│   ├── Workouts/
│   │   ├── WorkoutForm.jsx
│   │   ├── WorkoutList.jsx
│   ├── Calories/
│   │   ├── CalorieForm.jsx
│   │   ├── CalorieList.jsx
│   ├── Insights/
│   │   ├── Insights.jsx
│   ├── Layout/
│   │   ├── Navbar.jsx
│   │   ├── Sidebar.jsx
├── pages/                  # Main pages of the application
│   ├── Home.jsx
│   ├── Profile.jsx
│   ├── NotFound.jsx
├── services/               # API services for backend communication
│   ├── AuthService.js
│   ├── WorkoutService.js
│   ├── CalorieService.js
│   ├── InsightService.js
```

```
│ ——— context/                # Context API for global state management
│   ├——— AuthContext.js
│   ├——— WorkoutContext.js
│   ├——— CalorieContext.js
│ ——— hooks/                  # Custom hooks
│   ├——— useAuth.js
│ ——— styles/                 # CSS files for styling
│   ├——— Global.css
│   ├——— Dashboard.css
│   ├——— Insights.css
│ ——— utils/                  # Utility functions
│   ├——— helpers.js
│ ——— App.js                  # Main React component
│ ——— index.js                # Entry point
│ ——— routes.js               # Defines app routes
│ ——— config.js               # Configuration settings (e.g., API URLs)
```

Backend Code:

File Name:** `Backend_HealthFitnessTracker.zip`

-Structure:-

```
server/
│ ——— FitnessAPI/             # Root folder of the .NET Web API project
│   ├——— Controllers/          # API Controllers to handle requests
```

```
│   │   ├── AuthController.cs
│   │   ├── WorkoutController.cs
│   │   ├── CalorieController.cs
│   │   ├── InsightsController.cs
│   ├── Models/              # Entity models representing database tables
│   │   ├── User.cs
│   │   ├── Workout.cs
│   │   ├── Calorie.cs
│   │   ├── Insights.cs
│   ├── Data/                # Database context and migrations
│   │   ├── FitnessDbContext.cs
│   │   ├── DbInitializer.cs
│   ├── Services/            # Business logic and service classes
│   │   ├── AuthService.cs
│   │   ├── WorkoutService.cs
│   │   ├── CalorieService.cs
│   │   ├── InsightsService.cs
│   ├── Helpers/             # Utility classes (e.g., JWT, validation)
│   │   ├── JwtHelper.cs
│   │   ├── PasswordHasher.cs
```

```
│   ├── DTOs/                 # Data Transfer Objects (DTOs)
│   │   ├── UserDto.cs
│   │   ├── WorkoutDto.cs
│   │   ├── CalorieDto.cs
│   ├── Repositories/         # Repository pattern for database operations
│   │   ├── IUserRepository.cs
│   │   ├── IWorkoutRepository.cs
│   │   ├── ICalorieRepository.cs
│   │   ├── UserRepository.cs
│   │   ├── WorkoutRepository.cs
│   │   ├── CalorieRepository.cs
│   ├── Middleware/           # Custom middleware (if needed)
│   │   ├── ErrorHandlingMiddleware.cs
│   ├── appsettings.json      # Application configuration file
│   ├── Program.cs            # Entry point of the API
│   ├── Startup.cs            # Configuration for services, middleware, etc.
│   ├── FitnessAPI.csproj     # Project file
├── FitnessAPI.sln            # Solution file
├── Database_Schema.sql       # SQL script for database schema
├── README.md                 # Project documentation
```

Database & Configuration Files:

- **Database Schema:** `Database_Schema_HealthFitnessTracker.sql`
- **API Configuration:** `appsettings.json`

Deployment Details:

- The project is deployed on GitHub and kept private.
- Ensure all environment variables are correctly set.

8. Final Check Before Submission

Consistency & Formatting:

- Clear headings and uniform font styles.
- Logical flow of content with page numbers.

File Naming & Submission:

- Ensure all files are named correctly.
- Double-check database scripts and configurations.
- Verify successful submission via the designated platform.