A

Project Report On

WorkoutWizard- workout planner

B. Tech. CE Semester – V

Subject: (CE-520) Advanced Technologies

Prepared By:

VEKARIYA VASU PIYUSHBHAI (CE167) - 22CEUOS014

Guided By:

Prof. Preeti Sharma

Assistant Professor

Dharmsinh Desai University

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

Faculty of Technology, College Road, Nadiad – 387001, Gujarat

CERTIFICATE

This is to certify that the term work carried out in the subject of (**CE520**) **Advanced Technologies** and submitted is the bonafide work of **Vekariya Vasu** Roll No.: **CE167** Identity No.: **22CEUOS014** of B.Tech. **Semester V** in the branch of **Computer Engineering** during the academic year 2024-2025.

Prof. Preeti Sharma	Dr. C. K. Bhensdadia
Assistant Professor	Head, CE Department

Table of Contents

Chapter 1 Project Abstract

WorkoutWizard is a web-based application designed to assist users in managing their workouts and meal plans effectively. Developed using the MERN stack (MongoDB, Express, React, and Node.js),

The platform allows users to register, log in, and access a wide range of features, including custom workout plans, a library of exercises, and pre-built routines. Users can log their workouts, *WorkoutWizard* includes functionalities for meal planning and nutrition tracking, enabling users to maintain a balanced diet alongside their workout routine.

The system supports different user roles, including end users who can customize their fitness journey, and administrators responsible for managing the platform's data and maintaining user security. Built with a focus on data security, it leverages JWT for authentication and bcrypt for password encryption, ensuring user data is protected.

WorkoutWizard aims to provide a comprehensive, user-friendly solution for fitness enthusiasts, helping them achieve their fitness goals through well-organized and accessible tools for workout and meal management.

Chapter 2 Introduction

Introduction: WorkoutWizard is designed to digitize and streamline fitness management, providing users with the tools to effectively plan, track, and enhance their workouts. The platform allows users to add custom workout exercises, view their exercise history, and track daily streaks, helping them maintain consistency in their fitness routines.

Users can explore workouts categorized by body parts, such as abs, chest, legs, and more, making it easier to focus on specific areas. Additionally, the app offers short workout videos for each exercise, guiding users through proper techniques. The meal planning feature provides access to basic meal plans, supporting a balanced diet alongside users' exercise routines.

The project aims to enhance the user experience with a comprehensive and intuitive interface, making fitness tracking and planning a seamless part of a healthier lifestyle.

- **Technology/Platform/Tools Used:**

  - **Frontend:** ReactJS , Bootstrap
  - **Backend:** NodeJS , Express.js
  - **Database:** MongoDB
  - **Authentication & Authorization:** JWT
  - **Tools:** Postman, Git, GitHub
  - **Deployment:** Render

Chapter 3 System Analysis

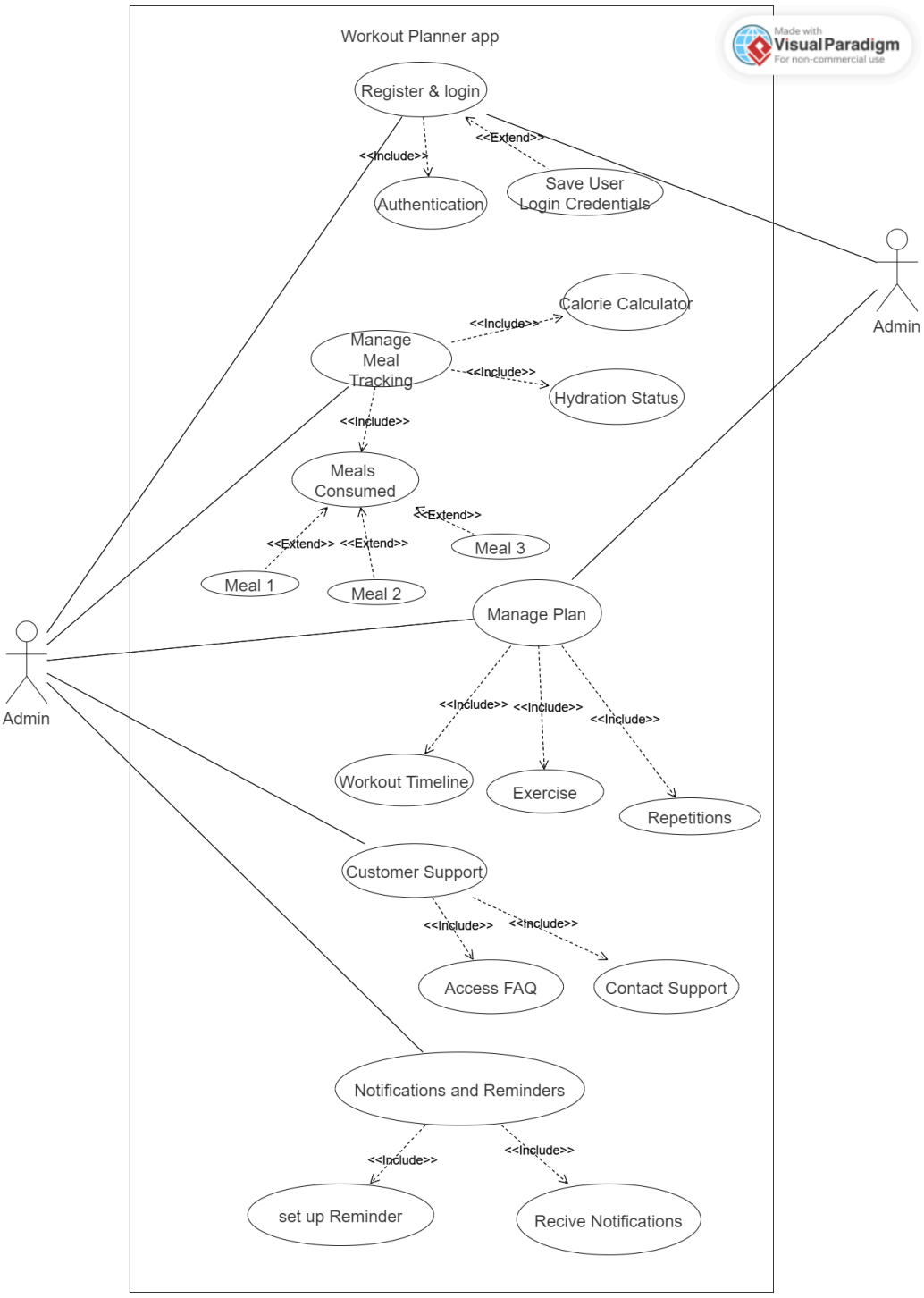

Fig 3.1.1 Use Case Diagram

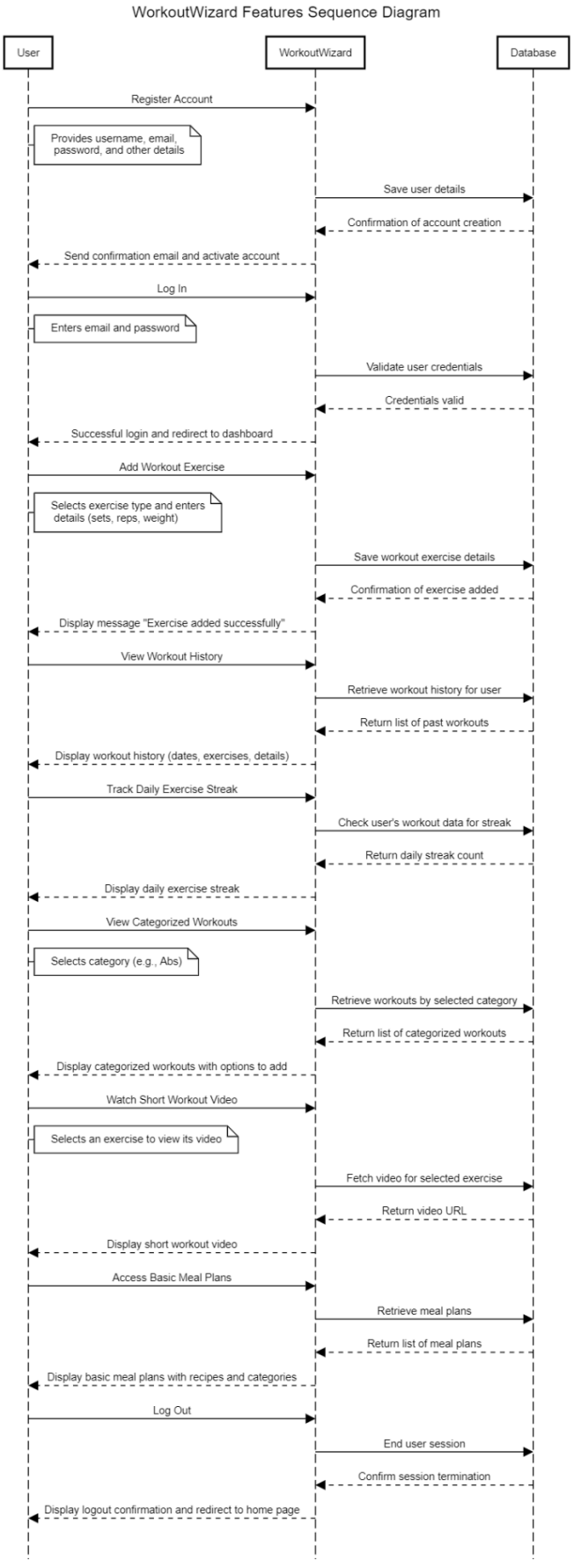

Fig 3.1.2 Sequence Chart

Chapter 4 Software Requirement Specification (SRS)

Introduction

Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the *WorkoutWizard* application. This document outlines the system's functionalities, features, and constraints, including the ability for users to add workout exercises, view their workout history, track daily exercise streaks, access categorized workout routines, watch short exercise videos, and view basic meal plans. It serves as a comprehensive guide for developers, testers, and stakeholders, ensuring a clear understanding of the application's requirements and guiding its development and implementation.

Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

Intended Audience and Reading Suggestions

- **Project Managers and Stakeholders**: To understand the system requirements and the scope of the *WorkoutWizard* application, ensuring alignment with project goals.
- **Developers**: To use as a reference for implementing the system's features, such as workout exercise management, history tracking, and video integration.
- **Testers**: To develop test cases and validate functionalities like exercise tracking, video playback, and daily streak updates, ensuring a seamless user experience.
- **End Users and Technical Writers**: To understand the system's functionalities, including how to use the categorized workout plans, track progress, and access meal plans.

Product Scope

WorkoutWizard is a comprehensive fitness management application designed to streamline workout planning and meal tracking efficiently and effectively. Its purpose is to facilitate users in managing their fitness routines, allowing them to customize workouts, monitor progress, and access meal plans, ultimately making it easier for fitness enthusiasts to achieve their health goals.

For Users:

- Access to a wide range of customizable workout exercises.
- Ability to track workout history and monitor daily exercise streaks.
- Tools to view workouts categorized by body parts for focused training.
- Short workout videos for proper exercise guidance.
- Access to basic meal plans to support nutrition alongside fitness routines.

Objectives:

- To provide an intuitive and user-friendly interface that enhances user engagement.
- To ensure a secure and reliable platform for managing fitness data and meal tracking.
- To empower users with insights and progress tracking tools for better fitness outcomes.

WorkoutWizard aligns with the goals of promoting a healthier lifestyle by leveraging modern technology to simplify and enhance the fitness management process.

Overall Description

Product Perspective

WorkoutWizard is a new, self-contained web application designed to facilitate fitness management for users seeking to enhance their workout routines and meal planning. It is not a follow-on member of an existing product family nor a replacement for any existing system. This application is developed from the ground up to provide a comprehensive, user-friendly platform that empowers fitness enthusiasts to customize workouts, track progress, and access nutritional information effectively. By leveraging modern technologies, *WorkoutWizard* aims to create an engaging and motivating experience for users on their fitness journeys.

Product Functions

- **Add Workout Exercises**: Users can easily add and customize workout exercises to their routines based on personal fitness goals.
- **View Workout History**: Users can access their workout history to review past activities and monitor progress over time.
- **Track Daily Exercise Streaks**: The application allows users to track their daily exercise streaks, encouraging consistency and motivation in their fitness journey.
- **Categorized Workouts**: Users can view workouts categorized by body parts, such as abs, legs, and chest, facilitating focused training.
- **Short Workout Videos**: Users can watch short videos demonstrating exercises, ensuring proper technique and enhancing their workout experience.
- **Basic Meal Plans**: The application provides access to basic meal plans, helping users maintain a balanced diet alongside their fitness routines.

User Classes and Characteristics

The various users of this app are classified into three main types:

- Administrator
- End Users:

Operating Environment

- Web browsers (Chrome, Firefox, Safari, Edge)
- Node.js server environment
- MongoDB database

Design and Implementation Constraints

WorkoutWizard is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) for both frontend and backend services. The application is designed to comply with data protection standards, ensuring user privacy and security. It operates efficiently within the hardware limitations of typical devices, optimizing performance while managing memory and processing power. Key technologies and

tools include React for frontend development, Node.js for backend development, MongoDB for database management, and various third-party APIs for enhanced functionalities. The app follows a modular design, where each feature is encapsulated in separate modules, allowing for easy maintenance, scalability, and integration of future enhancements.

User Documentation

User documentation components such as user manuals, online help, and tutorials will be delivered along with the software to assist users in navigating and utilizing the system effectively.

Assumptions and Dependencies

Assumptions:

- The system will run 24/7 by time.
- The machine should have consistent access to the internet connectivity for better use.
- It requires the version of Windows 7 or above.
- User authentication mechanisms accurately verify user identities.

Dependencies:

- Users should have a proper understanding web application.
- The data will be stored in the database which might be reliable on other web hosting services

External Interface Requirements

User Interfaces

WorkoutWizard features a user-friendly interface designed to enhance the experience of managing fitness routines and meal plans. The key components of the user interface include:

1. **Workout Exercise Addition**: An intuitive interface that allows users to easily add and customize workout exercises based on their fitness goals, providing options for inputting details such as sets, reps, and rest periods.
2. **Workout History View**: A dedicated section where users can access and review their workout history, enabling them to track progress over time and identify trends in their fitness activities.
3. **Daily Exercise Streak Tracking**: An engaging visual representation of daily exercise streaks, motivating users to maintain consistency in their workouts.
4. **Categorized Workout Display**: Users can view workouts categorized by body parts (e.g., abs, legs, chest), allowing for targeted training and easy navigation through different exercise options.
5. **Short Workout Videos**: A section where users can watch instructional videos for each exercise, ensuring they perform movements correctly and safely.
6. **Basic Meal Plan Access**: An interface that provides users with easy access to basic meal plans, helping them to complement their workouts with appropriate nutrition.

Hardware Interfaces

- **User Devices**:
  - Supports personal computers, laptops, tablets, and smartphones. The application is compatible with major web browsers.
- **Web Servers**:
  - Hosted on servers capable of handling high traffic volumes, ensuring optimal performance and reliability.
- **Database Servers**:
  - Utilizes secure servers for storing user data, workout plans, and progress. These servers are configured for efficient data management and retrieval.

Software Interfaces

The system uses the MERN stack (MongoDB, Express.js, React.js, Node.js) for its frontend and backend development. The specific software interfaces include:

- **Frontend:**
    - React.js for the user interface components.
    - Axios for API calls.
    - HTML, CSS, and JavaScript for web page structuring and styling.
- **Backend:**
    - Node.js and Express.js for server-side operations.
    - MongoDB for database management.
    - Mongoose for object data modeling (ODM).
    - JWT (JSON Web Tokens) for secure user authentication and authorization.
- **APIs and External Libraries:**
    - External libraries such as bcrypt for password hashing, moment.js for date and time manipulation, and nodemailer for email services.

Communications Interfaces

The system uses web browsers as the main interface for accessing the application's features. The system communicates with various services using secure protocols:

- **HTTP/HTTPS:**
    - The system communicates with the backend services using HTTPS to ensure secure data transfer.
    - All data exchanges between the client and server are encrypted using SSL/TLS standards to prevent eavesdropping and data breaches.
- **Email and Push Notifications:**
    - The system uses email services for sending notifications and alerts to users.
    - Push notifications are supported through service worker implementations for real-time updates on supported devices.

By ensuring these interfaces are well-defined and secure, i-Medicare aims to provide a reliable and efficient platform for managing hospital operations and patient care.

□ System Features

This section demonstrates i-Medicare's most prominent features and explains how they can be used and the results they will give back to the user.

R.1 Add Workout Exercises

- **State**: The system includes a feature for users to add workout exercises to their routines.
- **Input**: The user provides details such as the exercise name, sets, reps, and rest periods.
- **Processing**: The system validates the entered information and saves the workout exercise to the user's profile.
- **Output**: If the information is valid, the system confirms that the exercise has been added successfully. If the information is invalid, the system prompts the user to correct the input.

R.2 View Workout History

- **State**: The system includes a feature for users to access their workout history.
- **Input**: The user requests to view their workout history.
- **Processing**: The system retrieves past workout data from the database.
- **Output**: The system displays a list of past workouts, including details such as exercises performed, sets, reps, and dates.

R.3 Track Daily Exercise Streaks

- **State:** The system tracks users' daily exercise activities.
- **Input:** The system checks the user's activity over a specified time period.
- **Processing**: The system calculates the number of consecutive days the user has exercised.
- **Output**: The system displays the user's current daily exercise streak, encouraging continued participation

R.4 Categorized Workout Display

- **State**: The system allows users to view workouts categorized by body parts.
- **Input**: The user selects a category (e.g., abs, legs, chest).

- **Processing**: The system retrieves workouts associated with the selected category.

- **Output**: The system displays a list of workouts organized by the chosen body part, with options to view or add them to the user's routine.

R.5 Short Workout Videos

- **State**: The system provides access to instructional videos for exercises.

- **Input**: The user selects an exercise to view the video.

- **Processing**: The system fetches the relevant video from the database or external source.

- **Output**: The system displays the short instructional video demonstrating the selected exercise.

R.6 Access Basic Meal Plans

- **State**: The system allows users to view basic meal plans.

- **Input**: The user requests to view available meal plans.

- **Processing**: The system retrieves meal plans and their associated recipes and nutritional information.

- **Output**: The system displays the basic meal plans, including detailed recipes and nutrition facts.

□ Other Nonfunctional Requirements

Database Requirements

- MongoDB should be used for storing user data, routines, meals, and entries.
- Mongoose should be used for defining schema and interacting with MongoDB.

Environmental Requirements

- Development and testing should be done in a local environment with Node.js and MongoDB installed.
- Deployment should be done on a server with Node.js and MongoDB support.

Security Requirements

- JWT should be used for authentication.
- Passwords should be hashed using bcrypt.
- Sensitive data should not be exposed in the client-side code.

Goal of Implementation

- **Third-Party API Integration:** Connect with APIs like Google Fit, Apple Health, or MyFitnessPal for seamless data sharing and synchronization across platforms.
- **Music Integration**: Allow users to play their workout playlists from Spotify or Apple Music directly within the app to enhance their exercise experience.

Chapter 5 Database Design

User Schema:

```
const UserSchema = new Schema({
  username: {
    type: String,
    trim: true,
    unique: true,
    required: "Username is Required",
  },
  password: {
    type: String,
    trim: true,
    required: "Password is Required",
    minlength: 6,
  },
  email: {
    type: String,
    unique: true,
    match: [/.+@.+\..+/],
  },
  cardio: [{
    type: Schema.Types.ObjectId,
    ref: "Cardio"
  }],
  resistance: [{
    type: Schema.Types.ObjectId,
    ref: "Resistance"
  }],
  meal : [{
    type: Schema.Types.ObjectId,
```

```
    ref: "Meal"
  }]
});
```

Cardio Schema:

```
const CardioSchema = new Schema(
  {
    type: {
      type: String,
      default: "cardio",
      required: true
    },
    name: {
      type: String,
      required: true,
      maxlength: 30
    },
    distance: {
      type: Number,
      required: true,
    },
    duration: {
      type: Number,
      required: true,
    },
    date: {
      type: Date,
      required: true,
    },
    userId: {
      type: Schema.Types.ObjectId,
      ref: 'User',
```

```
    required: true,
   },
  }
);
```

Resistance Schema:

```
const ResistanceSchema = new Schema(
 {
   type: {
    type: String,
    default: "resistance",
    required: true
   },
   name: {
    type: String,
    required: true,
    maxlength: 30
   },
   weight: {
    type: Number,
    required: true,
   },
   sets: {
    type: Number,
    required: true,
   },
   reps: {
    type: Number,
    required: true,
   },
   date: {
```

```
    type: Date,
    required: true,
  },
  userId: {
    type: Schema.Types.ObjectId,
    ref: 'User',
    required: true,
  },
 }
);
```

Meal Schema:

```
const MealSchema = new Schema(
  {
    name: {
      type: String,
      required: true
    },
    author: {
      type: Schema.Types.ObjectId,
      ref: 'User',
      required: true
    },
    recipe: {
      type: String,
      default: ""
    },
    date: {
      type: Number,
      required: true
    },
```

```
    description: {
      type: String
    },
    category: {
      type: String,
      required: true
    }
  }
)
```

Chapter 6 Implementation Details

i) Modules Created and Brief Description of Each Module:

User Management Module
This module handles all aspects of user accounts, including registration, login, and profile management. Users can create unique accounts, secure their profiles with passwords, and manage their personal information such as usernames and email addresses.

Workout Management Module
The Workout Management Module allows users to add and track their workout exercises, including both cardio and resistance training. Users can log workout details, view their workout history, and monitor daily exercise streaks, facilitating an organized approach to fitness.

Cardio Management Module
This module specializes in managing cardio workouts. Users can input details such as the type of cardio, distance, duration, and date of the workout. It provides a comprehensive view of their cardio history, enabling users to track performance and progress over time.

Resistance Training Module
The Resistance Training Module focuses on logging resistance workouts. Users can enter details about the exercises performed, including the type of exercise, weight lifted, sets, reps, and the date. This module allows users to review their resistance training history and track improvements in strength.

Meal Planning Module
This module enables users to create and manage their meal plans. Users can log meals, add recipes, and categorize their meals to support their fitness goals. The Meal Planning Module promotes balanced nutrition alongside exercise routines.

ii) Function Prototypes which implements major functionality:

1. **User Authentication:**

```
// /api/user for user signup

router.route("/").post(createUser)

// /api/user/login for user login

router.route("/login").post(login)

// /api/user/me to get single user data
```

```
router.route('/me').get(authMiddleware, getSingleUser);
```

2. Exercise :

```
router.route("/cardio").post(createCardio);

// /api/exercise/cardio/:id

router.route("/cardio/:id").get(getCardioById).delete(deleteCardio);

// /api/exercise/resistance

router.route("/resistance").post(createResistance);

// /api/exercise/resistance/:id

router.route("/resistance/:id").get(getResistanceById).delete(deleteResistance);
```

3. Meal :

```
// /api/meals

router.route("/meals").post(createMeal);

// /api/meals/:id

router.route(“/meals/:id").get(getMealById).delete(deleteMeal);
```

Chapter 7 Testing

Table 7.1.1 Login Page Test Cases

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
TC01	Check User Login Functionality with valid Credentials	1. Go to the login page 2. Enter a valid email 3. Enter the correct password 4. Click 'Login'	Email: vasu123@email.com Password: 123123	User should be able to log in successfully	Login successful	Pass
TC02	Check User login functionality with invalid credentials	1. Go to the login page 2. Enter an invalid email 3. Enter a wrong password 4. Click 'Login'	Email: wronguser@imedicare.com Password: WrongPass	User should not be able to log in, and an error message should be displayed	Error Message Displayed	Pass

Table 7.1.2 Registration Page Test Cases

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
TC01	Check registration with valid data	1. Go to the registration page 2. Enter a valid name, email, password 3. Click 'Register'	Name: John Doe Email: johndoe@example.com Password: Pass@123	User should be successfully registered and redirected to the login page	Registered successfully	Pass

TC02	Check registration with missing mandatory fields	1. Go to the registration page 2. Leave the 'Email' field empty 3. Enter valid data for other fields 4. Click 'Register'	Name: John Doe Email: [Empty] Password: Pass@123	The system should disable sign up button	sign up button disabled	Pass
TC03	Check registration with invalid email format	1. Go to the registration page 2. Enter an invalid email format 3. Enter valid data for other fields 4. Click 'Register'	Name: John Doe Email: johndoe@invalid Password: Pass@123	The system should display an error message for invalid email format	Error message displayed: "Invalid email format"	Pass
TC05	Check registration with already existing email	1. Go to the registration page 2. Enter an email that is already registered 3. Enter valid data for other fields 4. Click 'Register'	Name: John Doe Email: johndoe@example.com Password: Pass@123	The system should display an error message indicating that the email is already registered	Error message displayed: "Email is already registered"	Pass

Table 7.1.3 add Exercise page Test Cases

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
TC01	add cardio exercise with one or more empty field	1. Go to the 'Exercise' page 2.select cardio 3. Leave the ‘Distance’ field empty 4. Enter valid data for other fields 5. Click ‘Add’	Name : Running Distance : [Empty] Duration : 10 Date : 10/18/2024	system should disable Add button	Add button disabled	Pass
TC02	add cardio exercise with with valid data	1. Go to the 'Exercise' page 2.select cardio 3. Enter valid data for all fields 4. Click ‘Add’	Name : Running Distance : 1000 Duration : 10 Date : 10/18/2024	user able to add cardio and display a message “Cardio successfully added ”	“Cardio successfully added”	Pass
TC03	add Resistance exercise with one or more empty field	1. Go to the 'Exercise' page 2.select Resistance 3. Leave the ‘Reps’ field empty 4. Enter valid data for other fields 5. Click ‘Add’	Name : bench press weight : 100 sets : 10 reps : [empty] Date : 10/18/2024	system should disable Add button	Add button disabled	Pass

TC04	add resistance exercise with with valid data	1. Go to the 'Exercise' page 2.select resistance 3. Enter valid data for all fields 4. Click ‘Add’	Name : bench press weight : 100 sets : 10 reps : 10 Date : 10/18/2024	user able to add resistance and display a message “resistance successfully created”	resistance successfully created	Pass

Chapter 8 Screen-Shots

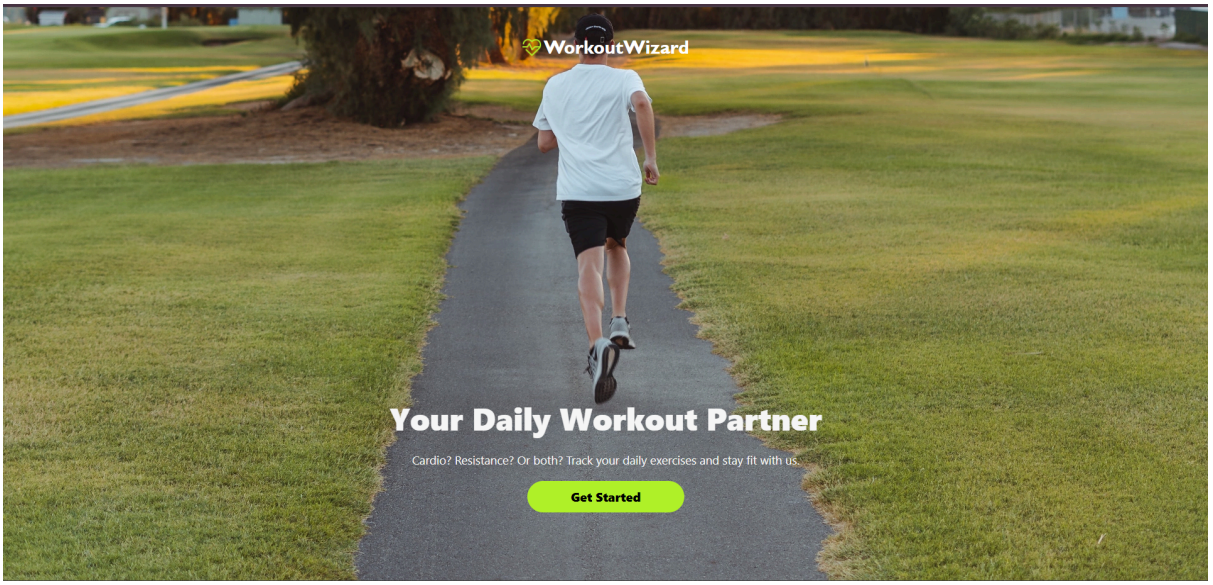

Fig. 8.1.1 welcome page

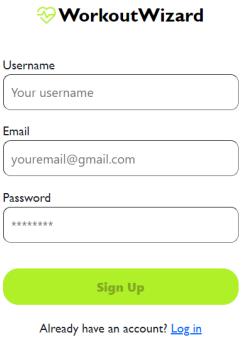

Fig. 8.1.2 sign up Page

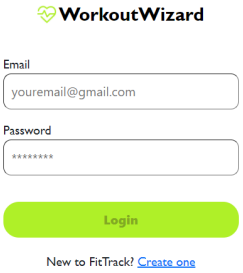

Fig. 8.1.3 login Page

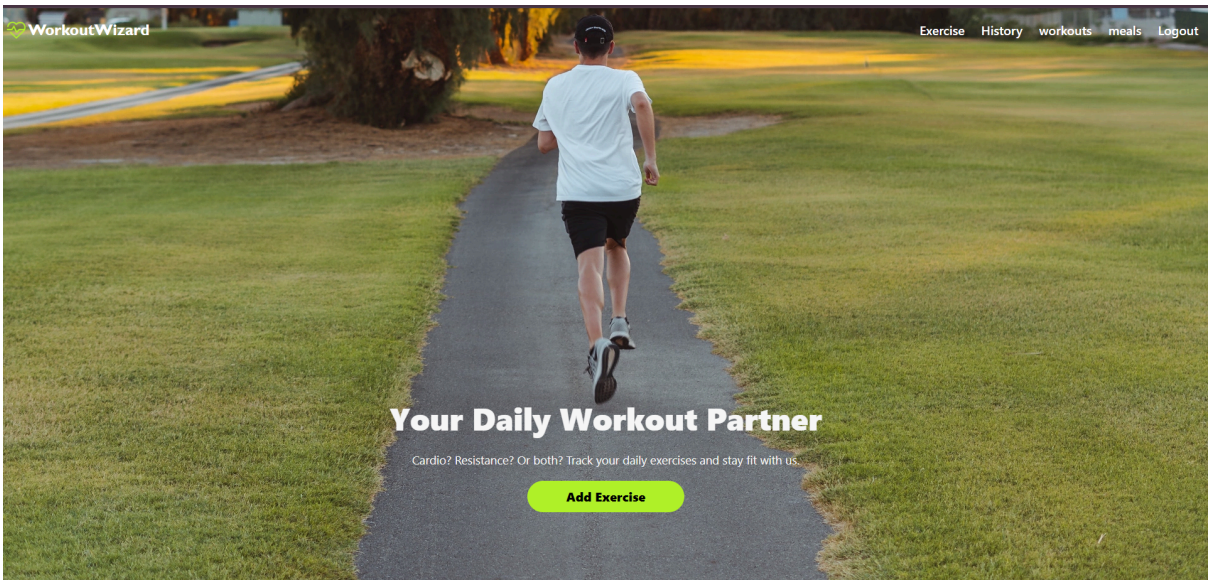

Fig. 8.1.4 home Page

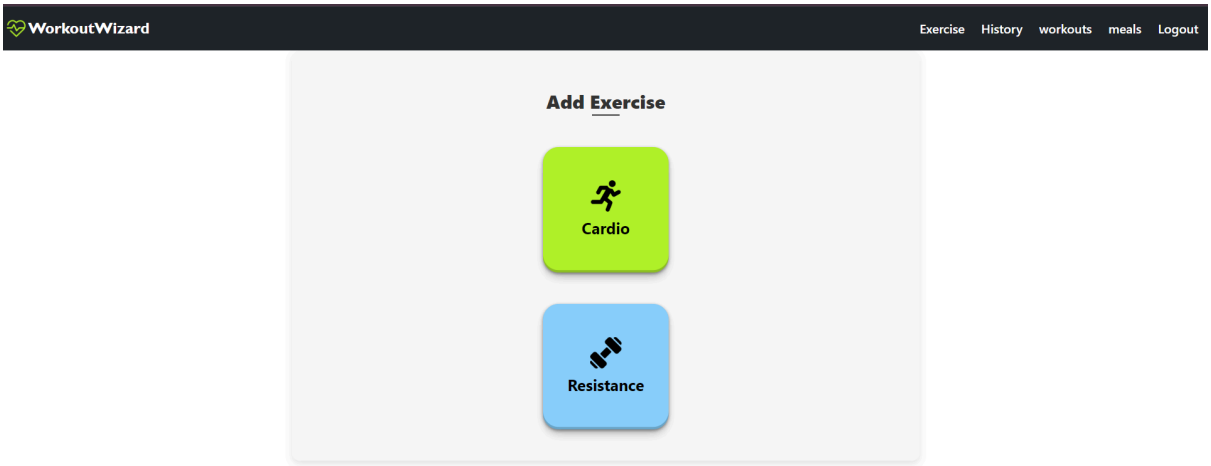

Fig. 8.1.5 Exercise Page

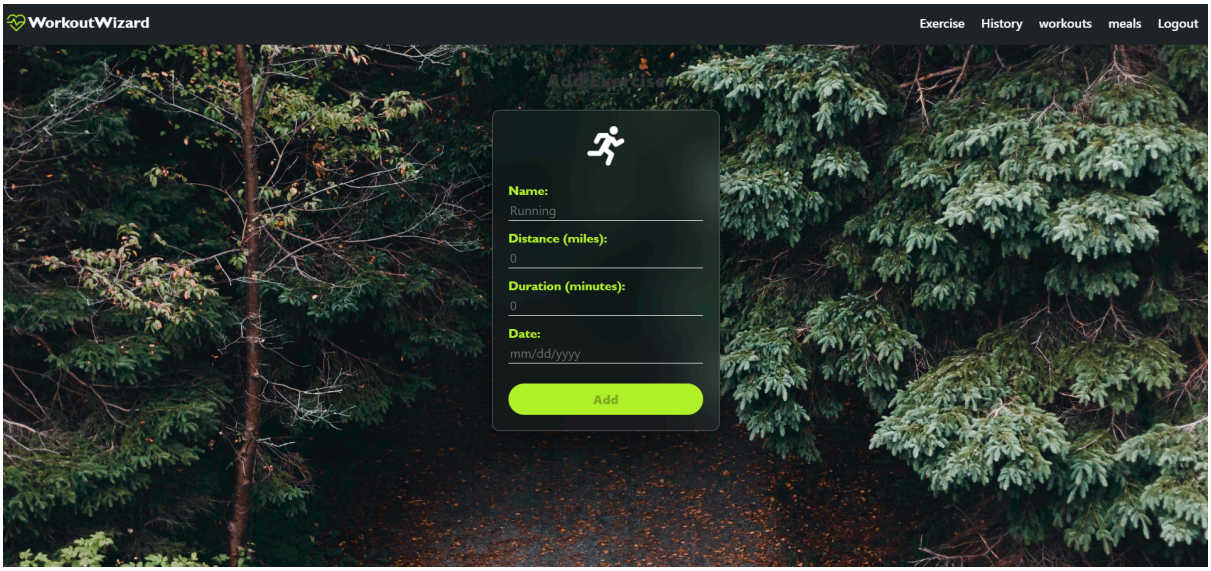

Fig. 8.1.6 Add cardio Page

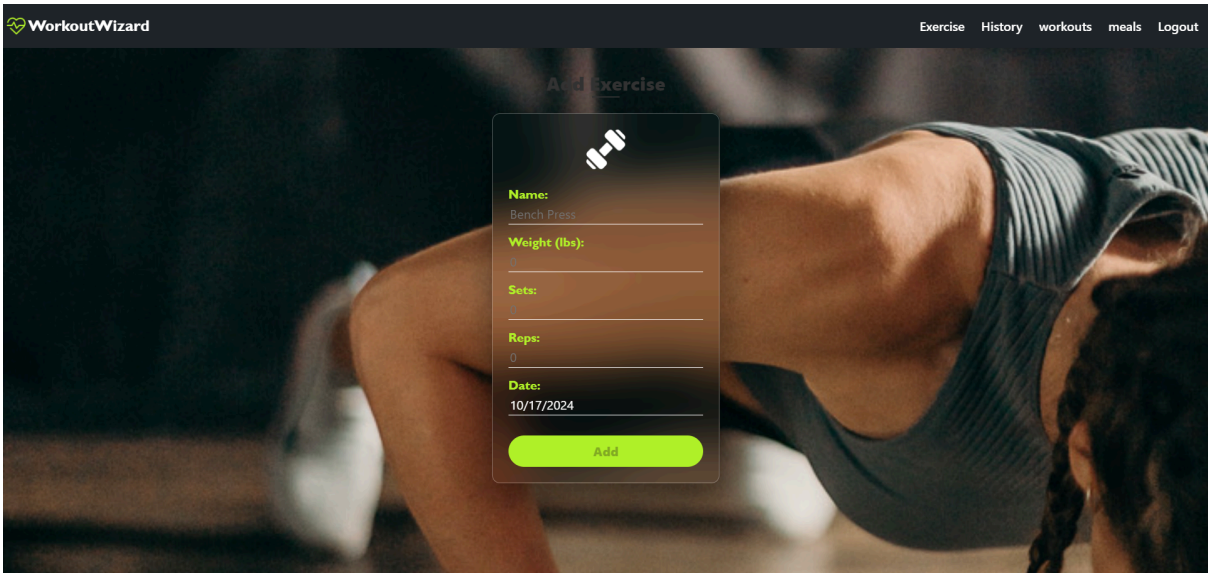

Fig. 8.1.7 Add resistance Page

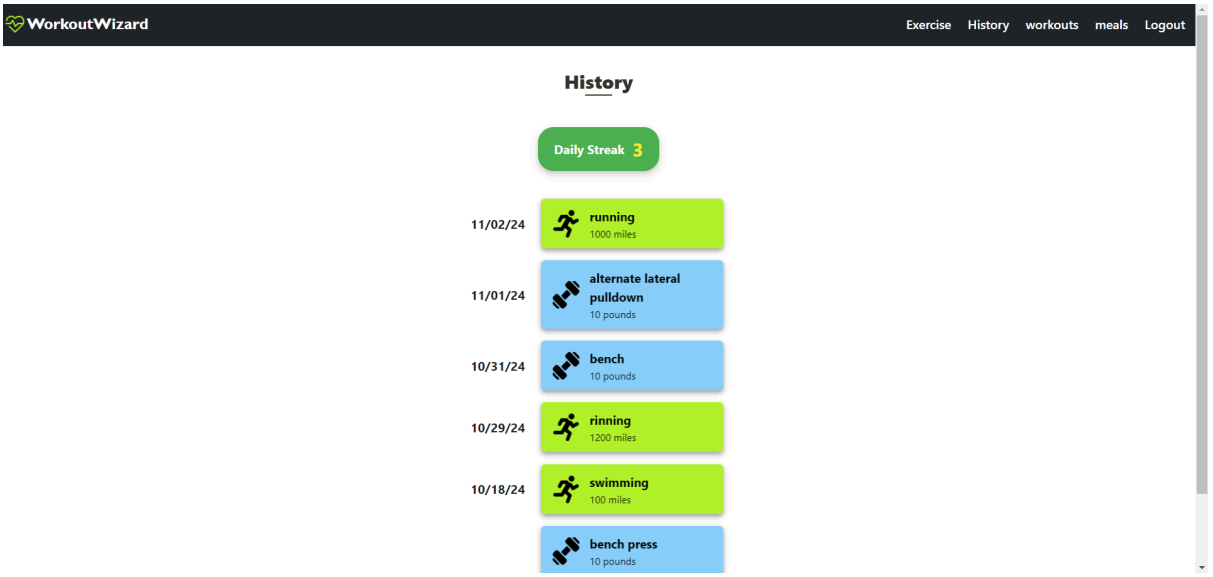

Fig. 8.1.8 history Page

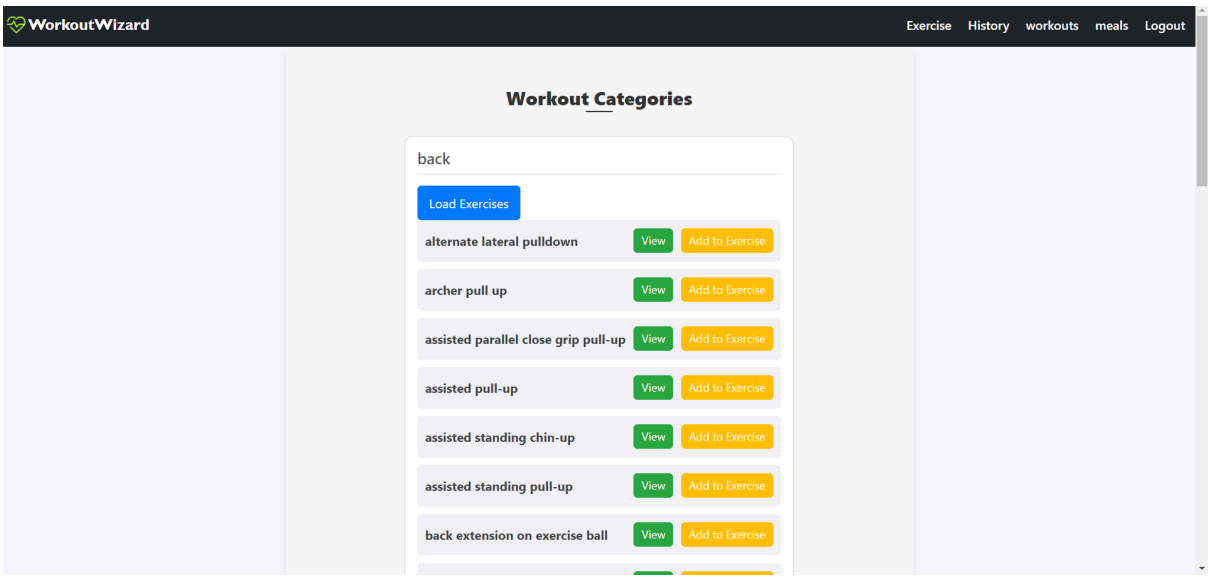

Fig. 8.1.9 workout Page

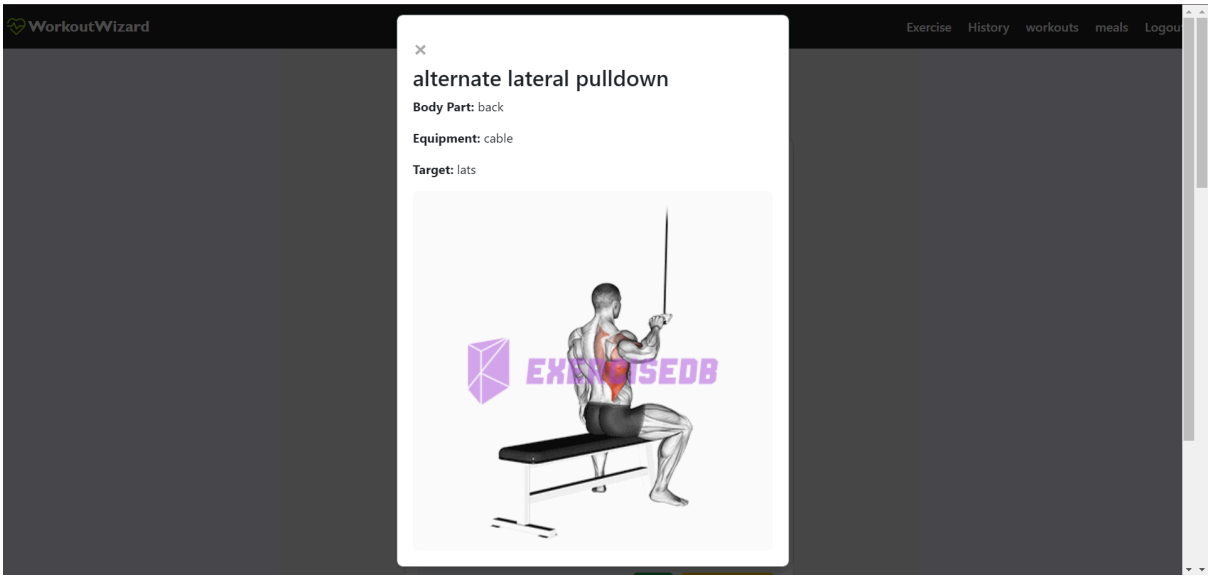

Fig. 8.1.10 workout short video popup

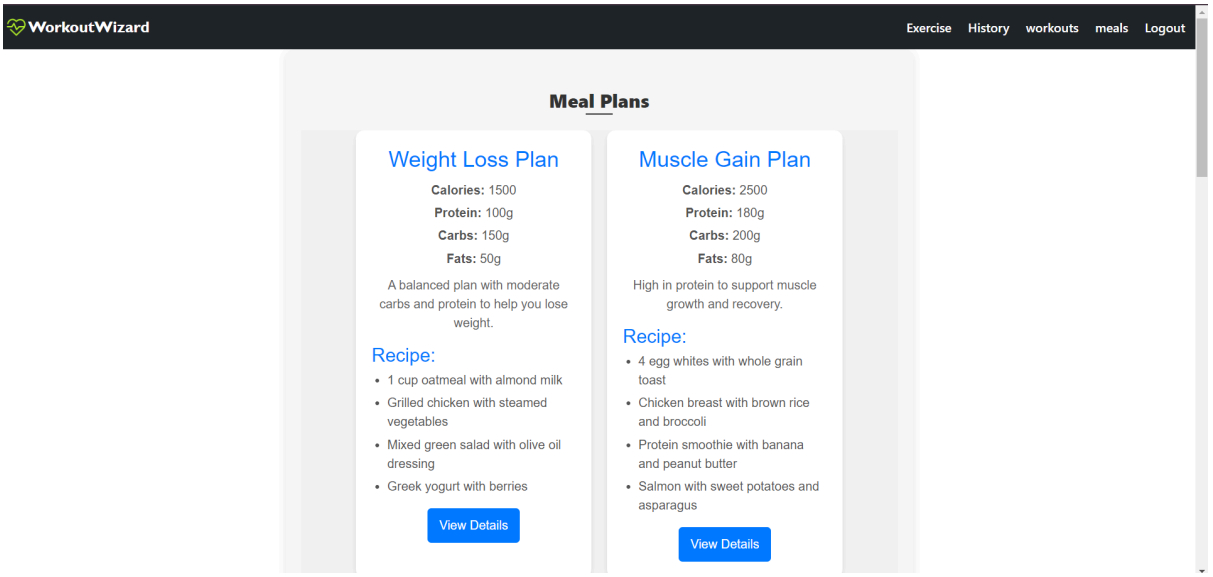

Fig. 8.1.11 Meal page

Chapter 9 Conclusion

The *WorkoutWizard* application successfully implements core features essential for efficient fitness management. These include user registration, workout tracking, meal planning, and instructional video access. Users can easily add and customize their workout exercises, view their workout history, and monitor daily exercise streaks, fostering consistency in their fitness routines. The application also allows users to create and manage meal plans, supporting balanced nutrition alongside their workouts. Additionally, the system enhances user experience with short instructional videos that ensure proper exercise techniques. Robust security measures protect user data through secure authentication. These functionalities provide a comprehensive and scalable solution for managing fitness and nutrition effectively, empowering users to achieve their health goals.

Chapter 10 Limitation and Future Extension

Limitations:

The user interface, while functional, may require further enhancements to improve usability and accessibility for a broader range of users.

The current architecture may face challenges with scalability under heavy load, particularly in managing concurrent user requests and extensive workout and meal data.

Some modules are still under development, including advanced reporting and analytics features for user progress and meal tracking.

Future Extensions:

Complete the remaining modules to provide a full range of functionalities, including advanced meal planning and nutrition tracking.

Incorporate a social feature that allows users to connect and share their fitness journeys for added motivation and support.

Implement integration with third-party fitness tracking devices and applications to provide a comprehensive view of users' health metrics.

Enhance the instructional video library with user-generated content and community feedback, enriching the exercise experience.

Chapter 11 Bibliography

[1] Ant Design - https://ant.design/docs/spec/introduce

[2] Bootstrap - https://getbootstrap.com/docs/5.3/getting-started/introduction/

[3] React JS - https://react.dev/learn/describing-the-ui

[4] Redux Documentation - https://redux.js.org/introduction/ecosystem

[5] JWT Authentication - https://jwt.io/introduction

[6] NodeJS - https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

[7] Express.js - https://expressjs.com/en/guide/routing.html

[8] MongoDB Atlas - https://www.mongodb.com/products/platform/atlas-database