



# UNIVERSITY OF WOLVERHAMPTON

## PROJECT AND PROFESSIONALISM

## (6CS007)

## ARTIFACT DESIGN REPORT

## FITNESS COMPANION APPLICATION

Full name : Ananda Neupane

Student ID : 2329810

Group Name : L6CG2

Submission date : February 2, 2024

Supervisor : Bipul Bahadur Pradhan

Reader : Yogesh Bikram Shah

# Table of Contents

# Table of Figures

# 1. Introduction

In today's digital era, technology plays a crucial role in enhancing fitness and wellness experiences. This report presents the design of an automated fitness application aimed at making workouts more personalized, efficient, and engaging. The primary goal is to build a system that simplifies workout tracking, provides real-time exercise feedback, and offers personalized nutrition guidance to help users achieve their fitness goals.

Designing the core functionalities of a system goes hand in hand with its development. In simple terms, artefact design refers to visually mapping out the system's features, user interactions, and workflows before bringing them to life. This report focuses on capturing the architecture, behavior, and core functionalities of the fitness application using various diagrams, such as:

Functional Decomposition Diagram – Breaking down the system's key features.

Use Case Diagram – Showcasing how users interact with the system.

Entity-Relationship (ER) Diagram – Representing the database structure.

Class Diagram – Structuring the application's object-oriented design.

Activity Diagram – Mapping out user interactions and system workflows.

The report begins with a high-level overview of the entire system, explaining how different features come together to create a seamless fitness experience. It then moves into detailed diagrams and wireframes for each functionality, ensuring clarity in system design and user interaction.

The key focus of this artefact is to provide a strong foundation for developing a user-friendly, intelligent, and adaptive fitness application. By integrating real-time pose estimation, workout tracking, and personalized meal planning, this system aims to revolutionize how users train, track progress, and maintain a healthy lifestyle in a structured and effective way.

# 2. Overall System-Level Diagrams

| | |
|---|---|
| **Use Case** | Authentication |
| **Actors** | User, System |

1. Functional Decomposition Diagram (FDD)

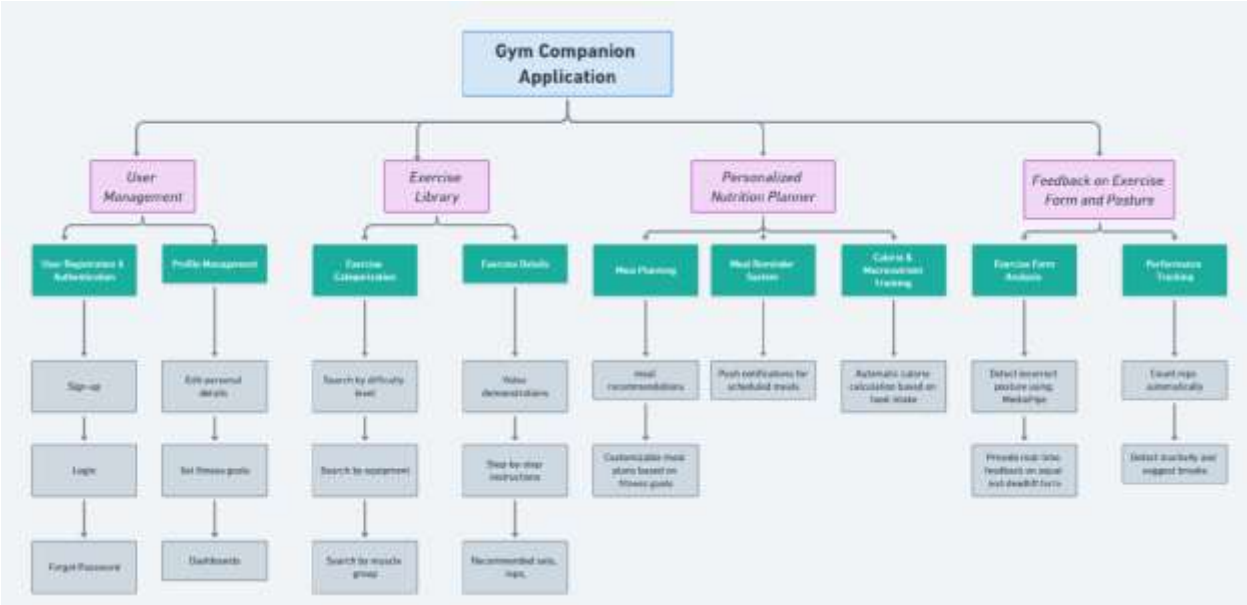



*Figure 1: FDD diagram*

FDD diagram represent high level breakdown of functionalities on the system. My project includes:

a) User Management: It includes User registration and authentication and the profile management and dashboards
b) Exercise Library: It includes browsing exercise, difficulty, exercise details, video tutorials, sets and reps.

c) Personalized Nutrition Planner: It includes generated meal plans, meal reminder system and calorie tracking.
d) Feedback on exercise form and Posture: It includes exercise form analysis and performance tracking like counting sets.

2. Use Case Diagram

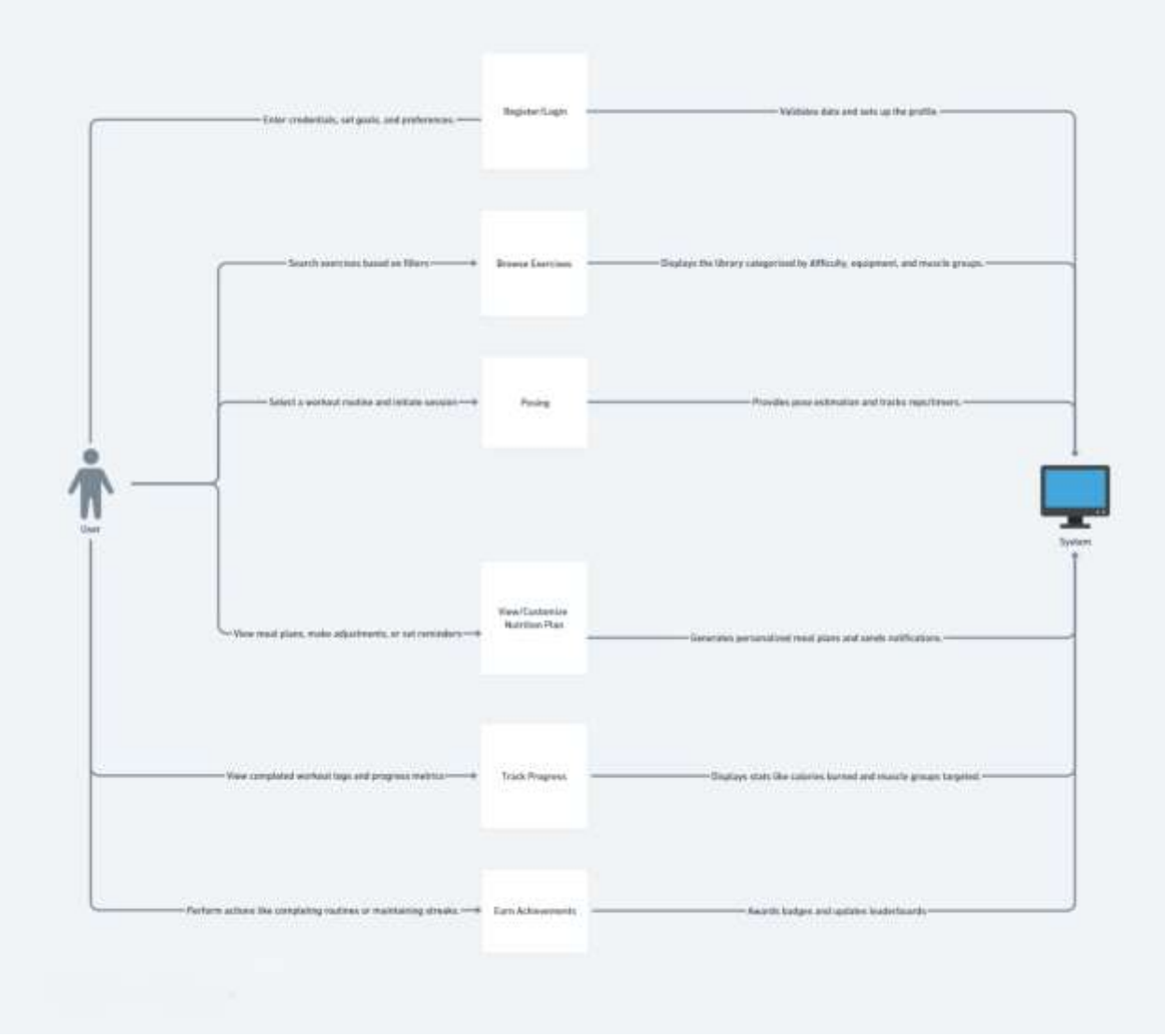



*Figure 2: Use case Diagram*

The Use Case Diagram represents how users interact with the system and its core functionalities:

a) User Management – Users can register, log in, and manage their profiles, including fitness goals, weight, and height.
b) Exercise Library – Users can browse exercises by category (difficulty, equipment, or muscle group) with instructions and video demonstrations.
c) Workout Routine Tracker – Users can create, edit, and track their workout plans with history logs.
d) Real-Time Pose Estimation – Provides form correction and feedback during exercises using MediaPipe.
e) Personalized Nutrition Planner – Suggests diet plans based on user goals and API data.
f) Meal Reminder System – Sends alerts for meal timings to maintain a healthy routine.
g) Progress Dashboard – Displays workout history, body measurements, and progress charts.
h) Exercise Timer – Assists users with timed workouts and rest periods.
i) Beginner & Pro Workout Modes – Allows users to switch between difficulty levels for workouts.
j) Feedback on Exercise Form – Analyzes posture and movement accuracy with AI-based tracking.

3. Entity Relationship(ER) Diagram


*Figure 3: Entity Relationship(ER) diagram*

The ER diagram represents the major entities and their relationships of our fitness application. The entities are as follows:

Entities:

a) CustomUser: Stores user information such as email, first name, last name, age, height, weight, goal, password, and OTP for user verification.
b) OTP: Represents the one-time password (OTP) sent to users for email verification.
c) Workout: Tracks workout details, including the date, duration, calories burned, and custom workout associations.
d) WorkoutExercise: Stores details of the exercises performed in a workout, including the body part targeted, exercise duration, and timestamps.
e) ExercisePerformance: Tracks the performance metrics of each exercise, including set number, reps, and weight lifted.
f) CustomWorkout: Represents custom workout plans created by the user, including workout names and descriptions.
g) CustomWorkoutExercise: Stores details of exercises within a custom workout, including the number of sets, reps, and weight.
h) PoseEstimation: Tracks pose estimation results for a user, including detected pose, accuracy, feedback, and timestamp.
i) NutritionPlan: Contains the user's personalized meal plan based on their goals, including meal types (e.g., Breakfast, Lunch, Dinner), calories, protein, carbs, and fats.
j) A NutritionPlan belongs to a CustomUser and can be linked to a MealLog.
k) CustomMeal: Represents meals customized by the user, including their nutritional content (calories, protein, carbs, fats) and meal type.
l) CheatMeal: Tracks cheat meals consumed by the user, including meal details and timestamp.
m) MealLog: Logs the meals consumed by the user, linking them to the CustomMeal, NutritionPlan, and CheatMeal entities.
n) MealReminder: Stores reminders for users to track their meals (e.g., Breakfast, Lunch, Dinner), including the reminder time and active status.

Key Relationships:

CustomUser to Workout: A user can have multiple workouts, each containing multiple exercises and performance details.

CustomUser to CustomWorkout: A user can create multiple custom workout plans with associated exercises.

CustomUser to PoseEstimation: Each user can have multiple pose estimations based on their workout sessions.

CustomUser to NutritionPlan: A user can have multiple meal plans based on their fitness goals.

CustomUser to CustomMeal: Users can create and log custom meals into their nutrition plan.

CustomUser to MealLog: Users can log their meals, whether from the predefined nutrition plan or custom meals, as well as any cheat meals.

CustomUser to MealReminder: A user can set meal reminders to receive notifications at specific times for meals like breakfast, lunch, and dinner.

4. Class Diagram


*Figure 4: Class Diagram*

Key Classes

a) CustomUser: Manages user-related operations, including registration, authentication, and personal data.
b) Workout: Handles user workout sessions, tracking total time, calories burned, and linking exercises.
c) WorkoutExercise: Stores individual exercises performed in a workout, including duration and body part focus.

d) ExercisePerformance: Records performance details such as sets, repetitions, and weights used.
e) CustomWorkout: Allows users to create personalized workout plans.
f) CustomWorkoutExercise: Stores exercises specific to a user-defined workout plan.
g) PoseEstimation: Analyzes user poses for accuracy and provides feedback.
h) NutritionPlan: Suggests meal plans based on user goals and dietary needs.
i) CustomMeal: Stores user-defined meals, including nutritional values.
j) CheatMeal: Logs meals that deviate from the recommended plan.
k) MealLog: Tracks meals consumed, linking to nutrition plans and custom or cheat meals.
l) MealReminder: Allows users to set reminders for meal times.
m) OTP: Handles one-time password authentication for verification.

Key Relationships

Inheritance

a) CustomUser → OTP: A CustomUser has multiple OTP entries for authentication.

Associations

a) User to Workout (1-M): A single user can log multiple workout sessions.
b) Workout to WorkoutExercise (1-M): A workout consists of multiple exercises.
c) WorkoutExercise to ExercisePerformance (1-M): Each exercise can have multiple performance records.
d) User to CustomWorkout (1-M): A user can create multiple custom workouts.
e) CustomWorkout to CustomWorkoutExercise (1-M): A custom workout consists of multiple exercises.
f) User to PoseEstimation (1-M): A user can perform multiple pose estimations.
g) User to NutritionPlan (1-M): A user can have multiple nutrition plans.
h) NutritionPlan to MealLog (1-M): A nutrition plan can have multiple meal logs.
i) User to CustomMeal (1-M): A user can create multiple custom meals.
j) User to CheatMeal (1-M): A user can log multiple cheat meals.

k) MealLog to CustomMeal (M-1, Nullable): A meal log can reference a custom meal.

l) MealLog to CheatMeal (M-1, Nullable): A meal log can reference a cheat meal.

m) User to MealReminder (1-M): A user can have multiple meal reminders.

5. Activity Diagram

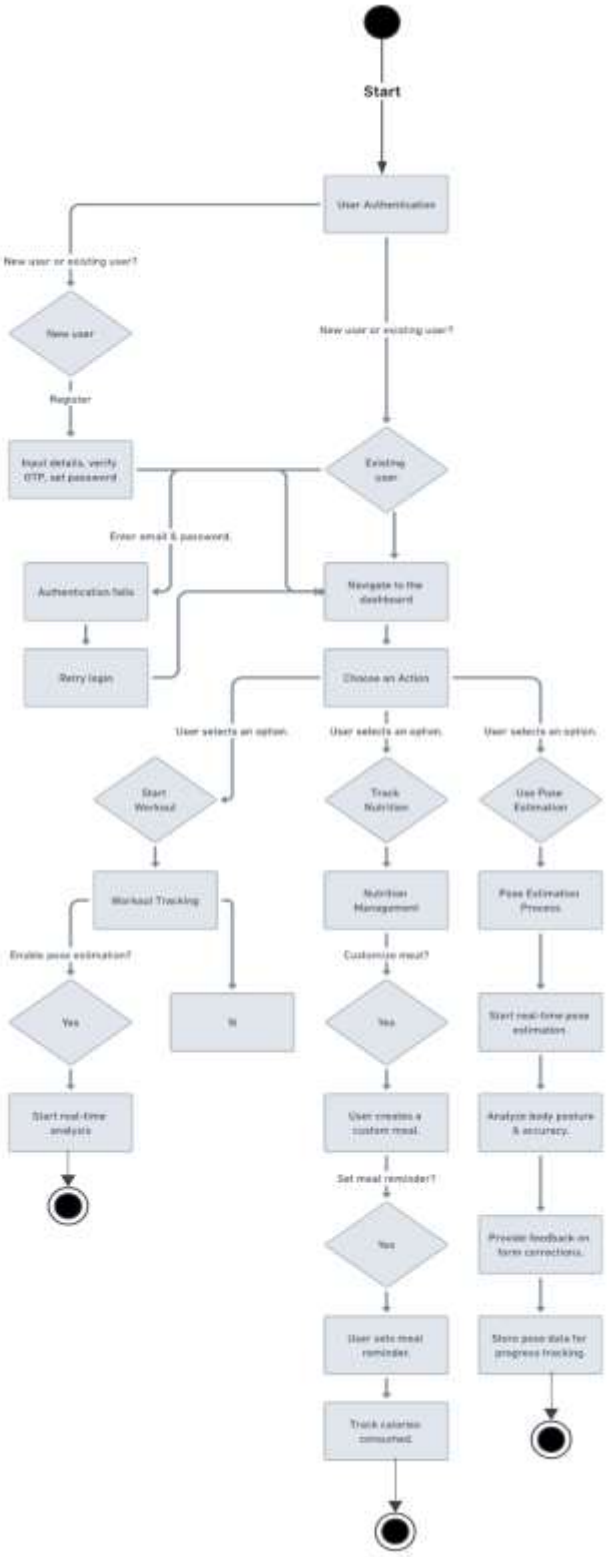



*Figure 5: Activity Diagram*

Generally activity diagram includes the flow of our program. Here is small brief of different symbols and uses that are involved in my figure.

a) Start Node → (Downward/Rightward Arrows) → Actions

The start node (represented by a solid black circle) marks the beginning.

Arrows flow downward or rightward to actions like "User logs in" or "Selects a workout".

b) Decision Points (Diamond Shape) → Multiple Paths

These represent conditional choices (e.g., "Is OTP correct?").

Yes moves in one direction (e.g., "Proceed to login").

No moves in another (e.g., "Resend OTP").

c) Parallel Actions (Fork Node → Two or More Paths)

A fork (thick horizontal bar) splits one process into multiple parallel actions.

Example: "Start Workout" → (Tracks time & monitors calories burned simultaneously).

d) Merging Paths (Join Node → Single Path)

A join (another thick bar) merges parallel tasks back into a single flow.

Example: "Workout Tracking & Pose Estimation" → (Both merge before saving results).

e) Final Node (Solid Black Circle with a Border)

Marks the end of the process, after all tasks are completed.

Example: "Workout session saved" → END.

# 3. Feature Specific Artifacts

## a. Exercise Library

| | |
|---|---|
| **Use case** | Browse Exercise |
| **Actors** | User, System |
| **Descriptions** | The system allows users to browse exercises categorized by the API (e.g., body part, difficulty, equipment). Users can view detailed information, video demonstrations, and progress tracking. Admins can monitor user activity and track exercise history. The system fetches categories and exercise data directly from the API. |

a) Use case diagram

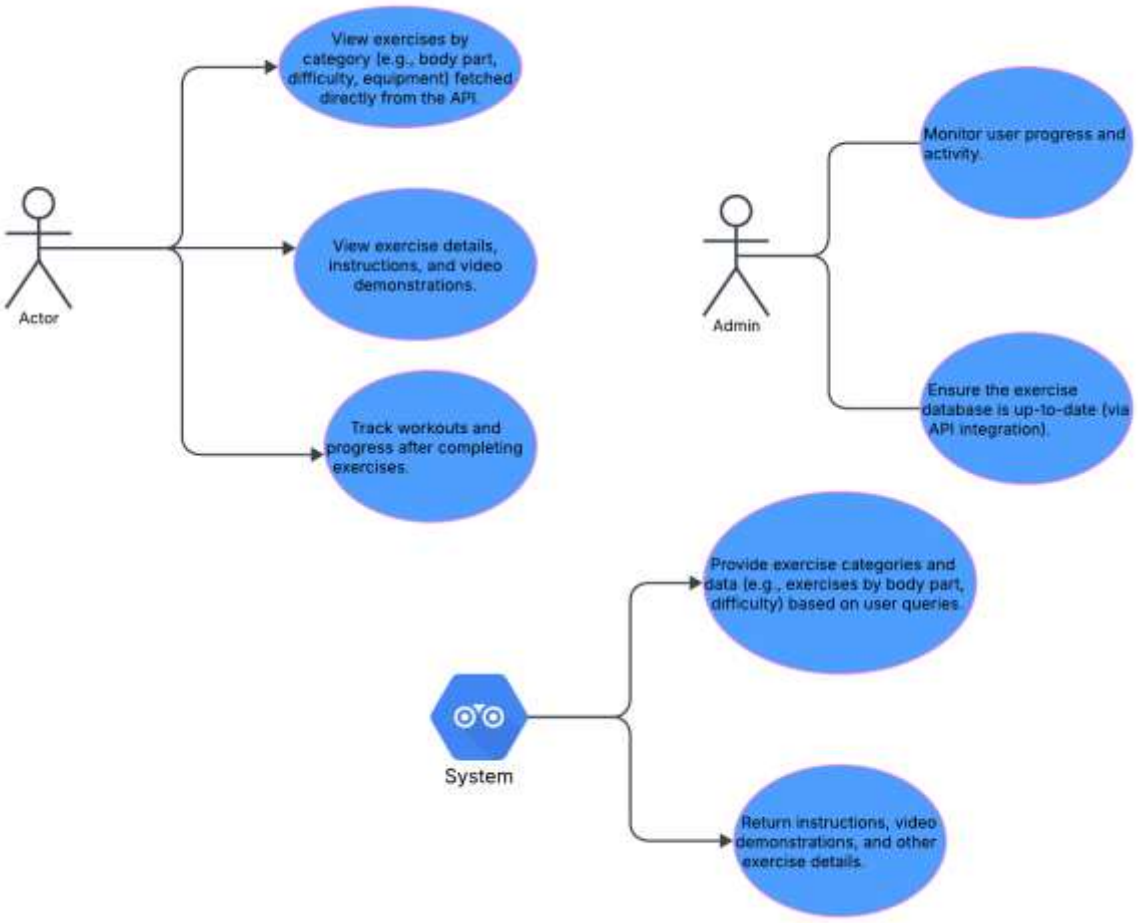



*Figure 6: Use case exercise*

The use case diagram depicts the interaction between the **User**, **Admin**, and **System (API)**. Users can browse exercises, track progress, and view detailed exercise instructions. Admins manage and monitor the exercise database and user activity. The system fetches categorized exercise data from the API, ensuring users can select exercises by category (body part, difficulty, equipment) and access relevant details.

b) Sequence diagram

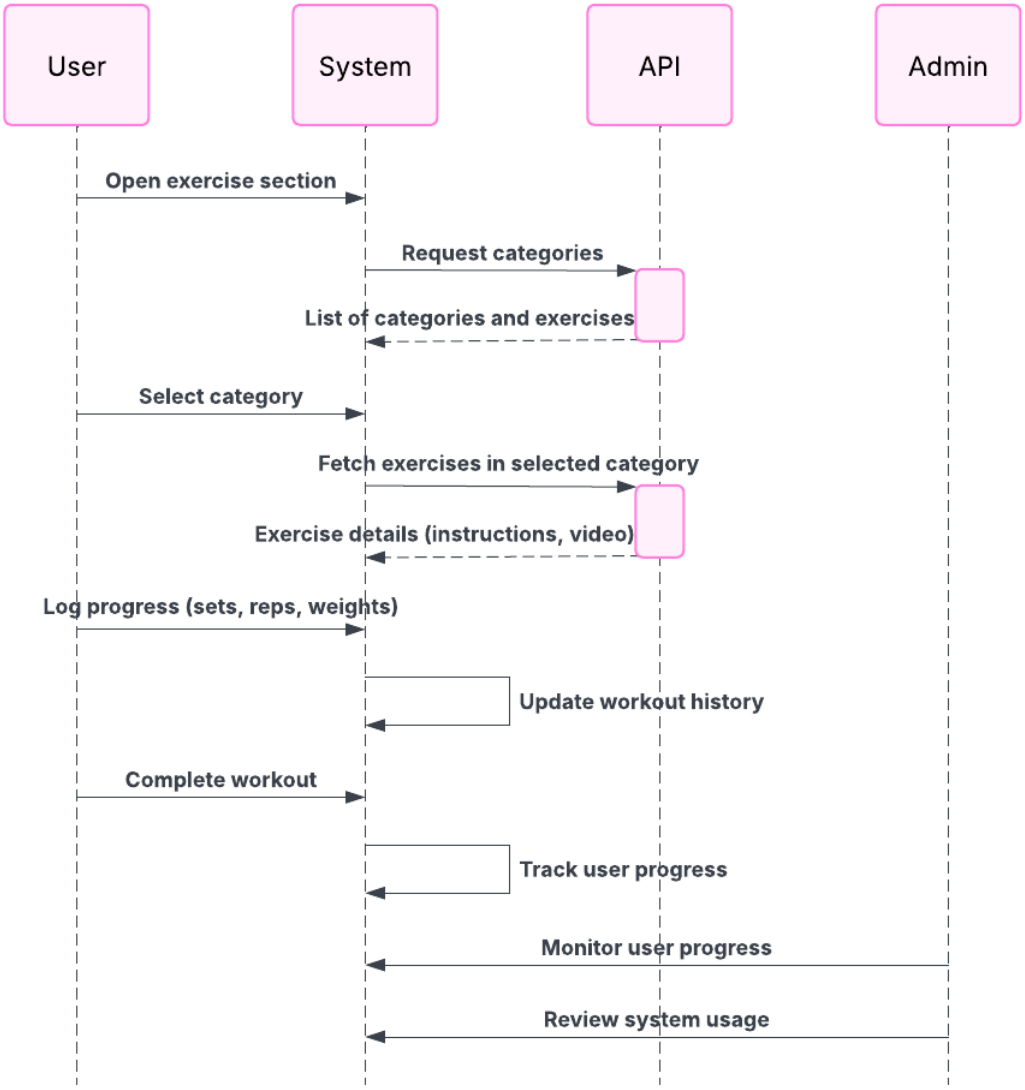



*Figure 7: Sequence Diagram exercise*

The sequence diagram illustrates the key steps in fetching exercises from the API and interacting with the system. The user requests exercises by category, the system fetches the data from the API, and displays exercise details, including instructions and videos. The user performs the exercise, logs progress, and the system updates their workout history. Admins ensure the system provides up-to-date data.

c) Activity Diagram

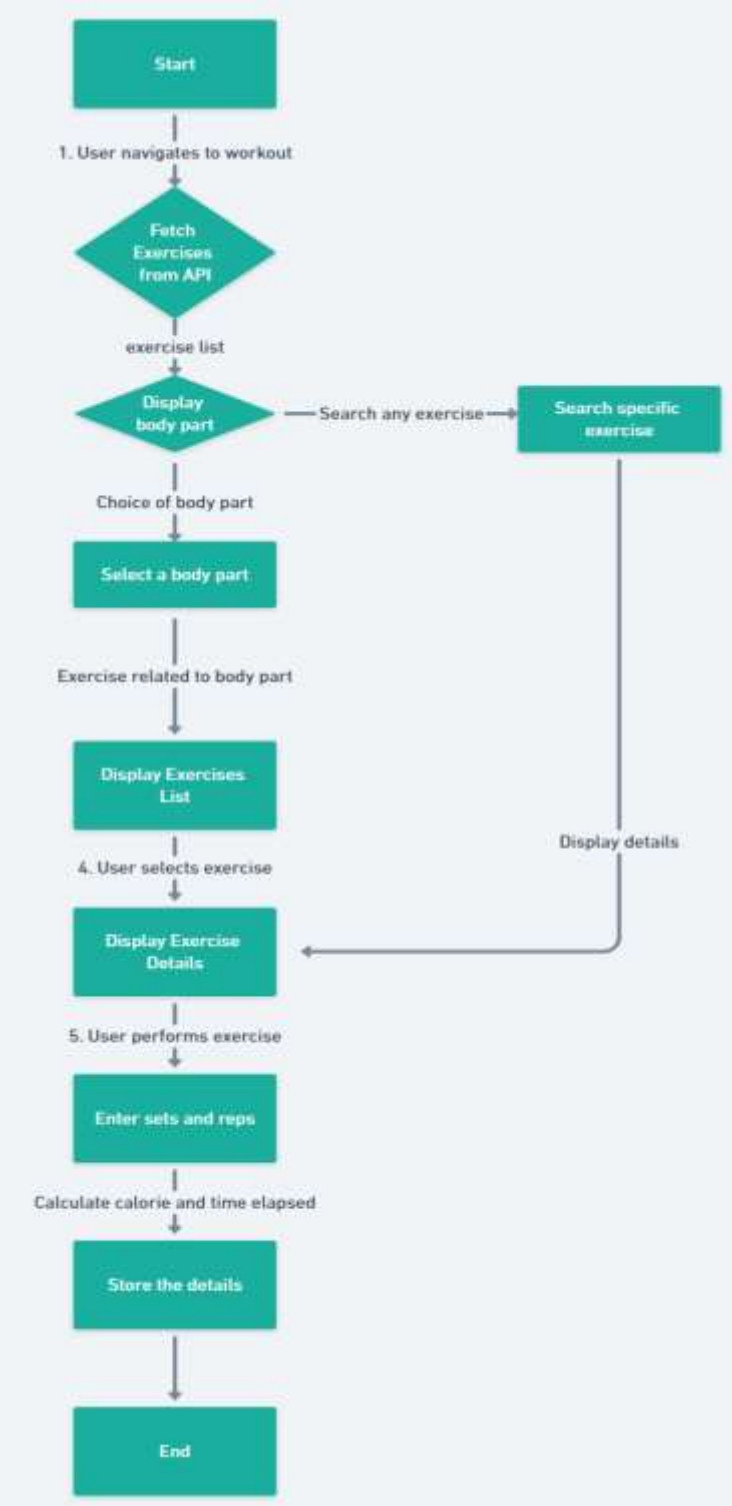



*Figure 8: Activity Diagram Exercise*

This activity diagram outlines the process flow for a user browsing and completing exercises. The user selects a category, the system fetches exercises from the API, and displays them. After performing the exercise and tracking progress (sets, reps, weights), the system updates the user's workout history. Admins monitor system activity and user progress, ensuring data is accurate and current.

### b. Nutrition

| | |
|---|---|
| **Use Case** | The person using the application to receive and track nutrition advice. |
| **Actors** | The individual managing and monitoring the nutrition plans. |
| **Description** | Provides data for diet plans, calories, and nutritional details. |

a) Use case diagram

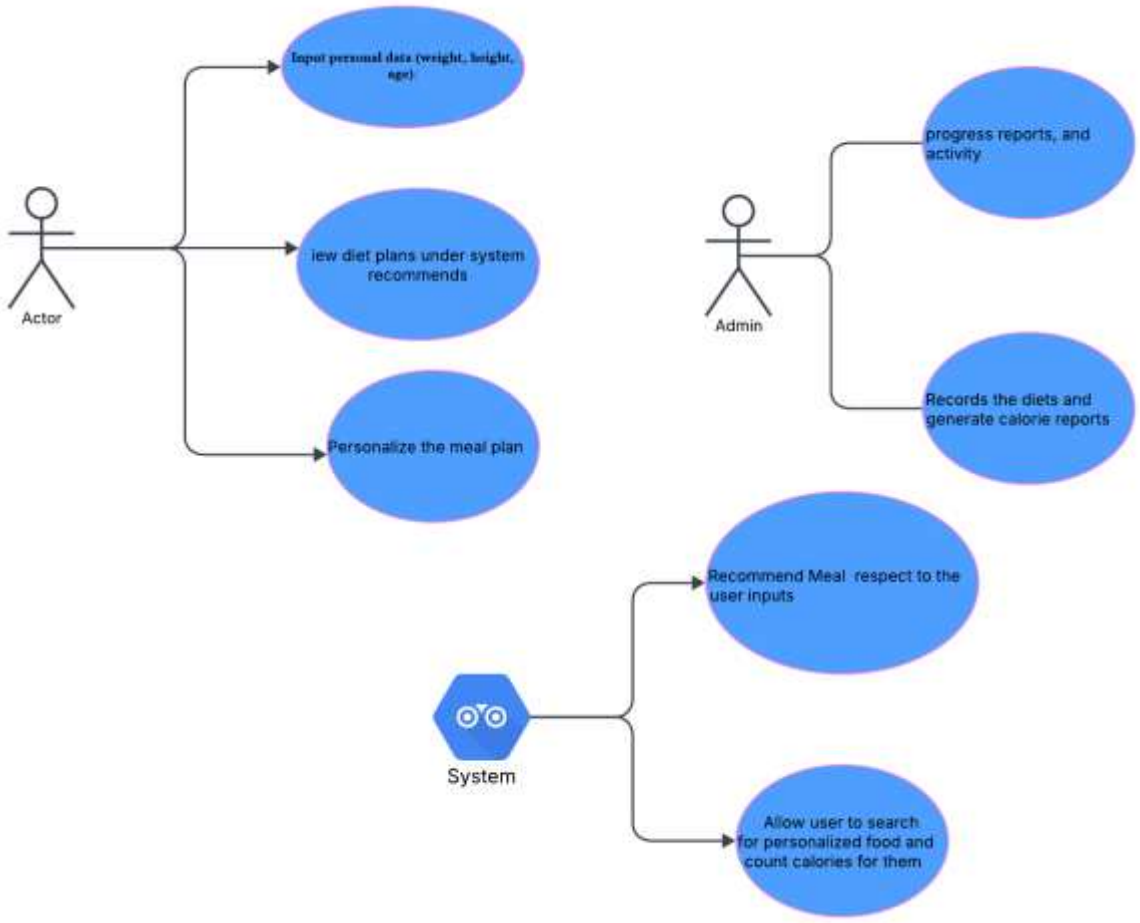



*Figure 9: Use case nutrition*

The Use Case Diagram for the Nutrition Section defines the primary interactions between users, admins, and the system. Users input personal data (weight, height, fitness goals) to receive personalized diet plans, track meals, and monitor their progress. Admins manage user progress and adjust diet plans. The system (API) fetches personalized data, meal information, and progress details, enabling users to follow their nutrition plans effectively.

b) Sequence diagram

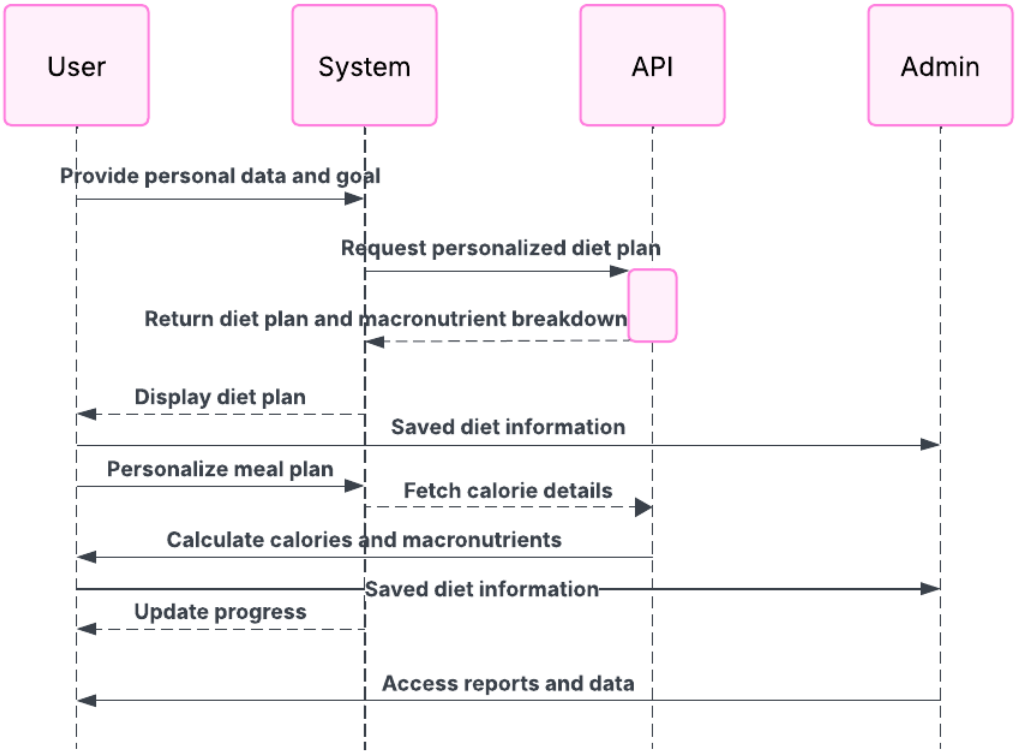



*Figure 10: Sequence diagram nutrition*

The Sequence Diagram illustrates the interaction flow for generating and tracking a user's diet plan. The user inputs personal information, which the system sends to the API for processing. The API responds with a personalized diet plan, which the user reviews. As the user logs meals, the system queries the API for nutritional data, tracks the user's progress, and provides feedback or suggestions based on the user's consumption and progress.

c) Activity diagram

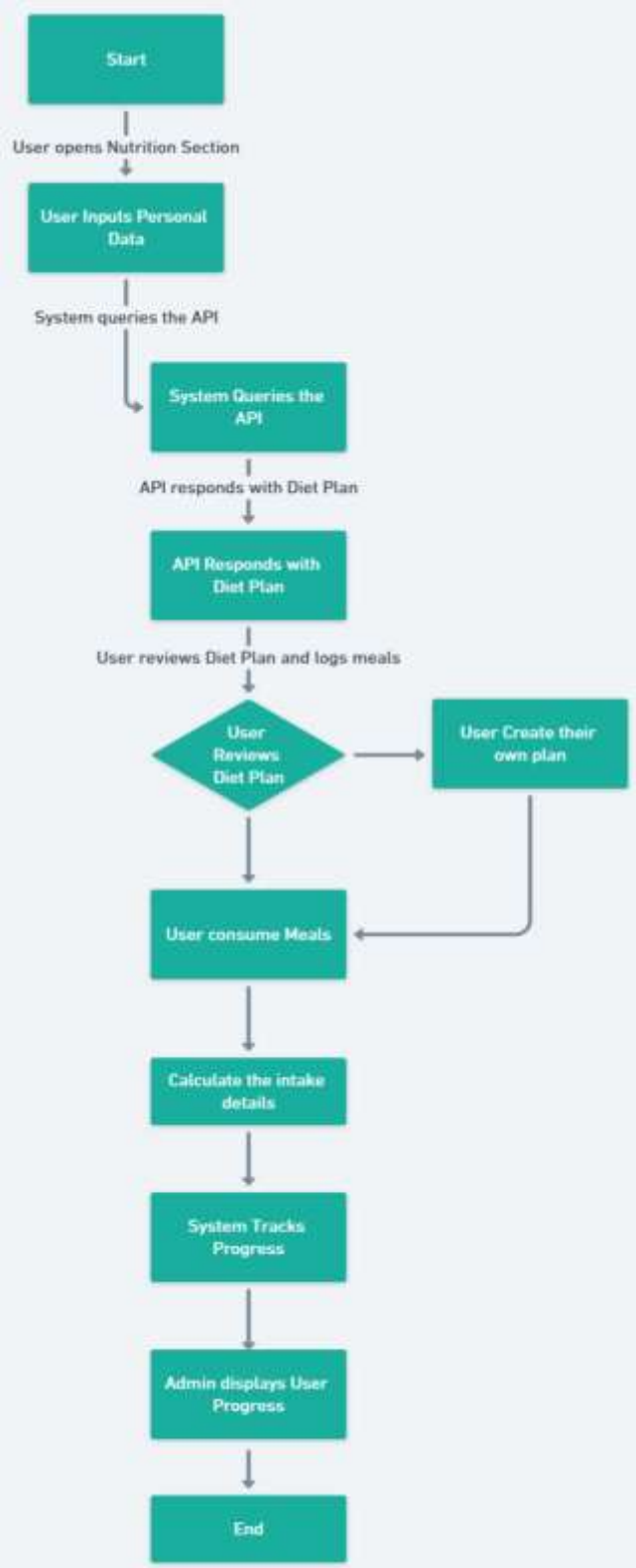


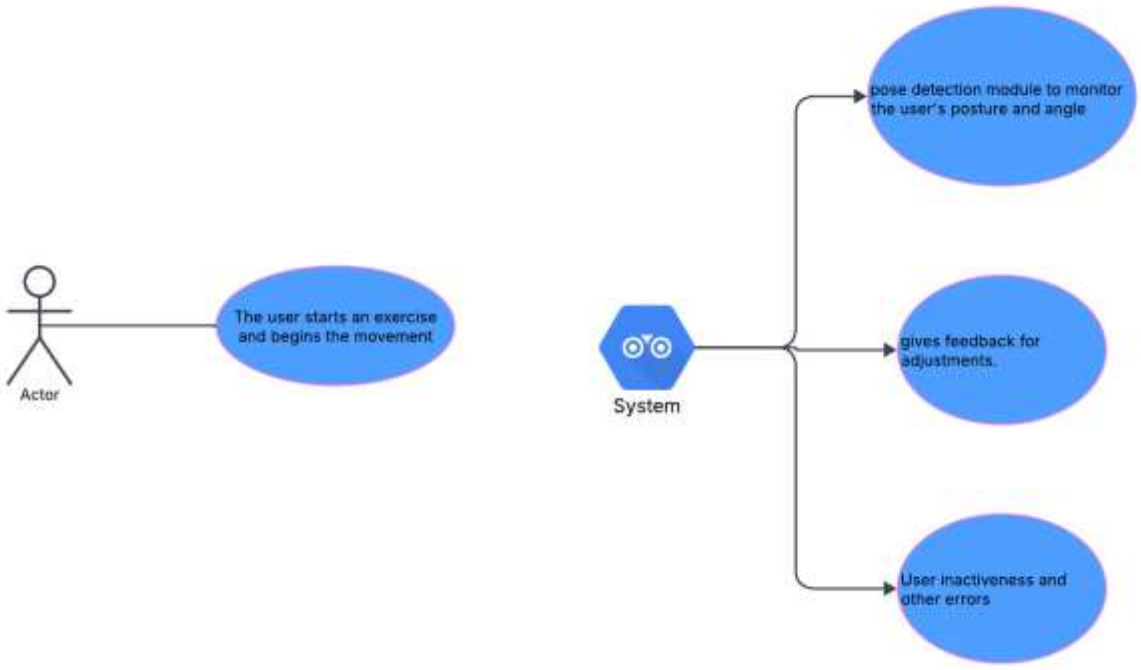

*Figure 11: Activity diagram nutrition*

The Activity Diagram details the process of logging meals and tracking progress in the Nutrition Section. The user inputs personal data, receives a personalized diet plan from the API, and begins consuming meals. The system tracks nutritional progress and provides feedback, suggesting adjustments if goals are not met. Admins monitor user progress and can update diet plans. This flow ensures users stay on track and receive tailored recommendations for their nutrition goals.

### c. Pose estimation

| | |
|---|---|
| **Use Case** | Pose Estimation using Mediapipe |
| **Actors** | User, Admin, Pose Estimation System (MediaPipe). |
| **Description** | The user performs squats and deadlifts, and the system uses pose estimation to track the movements. It provides real-time feedback on posture (e.g., back straightness, knee positioning) and evaluates form quality. The admin monitors user progress and adjusts the system's feedback if necessary. The system helps the user maintain proper form to reduce injury and optimize performance. |

a) Use case diagram




b)

*Figure 12: Use case Pose*

The Use Case Diagram illustrates interactions between users and the system in the pose estimation feature. Users perform exercises (squats and deadlifts) while the system tracks their posture. The system provides real-time feedback on form, and trainers or admins monitor progress. The diagram captures how users engage with the system, receive feedback, and track their form over time for better exercise performance.

b) Sequence diagram

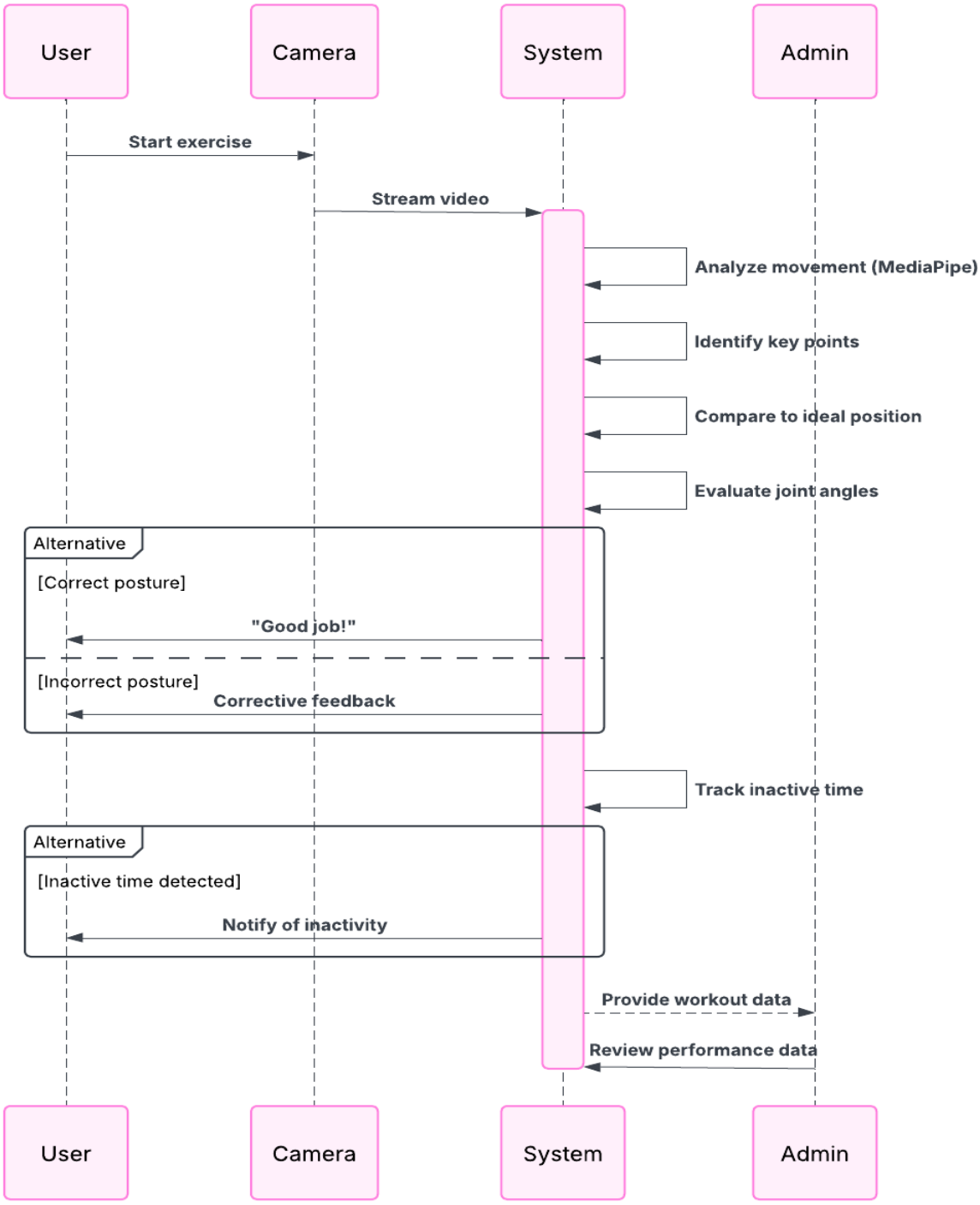



*Figure 13: Sequence diagram Pose*

The Sequence Diagram details the flow of interactions for squat and deadlift tracking. The user starts the exercise, and the system uses pose estimation (e.g., MediaPipe) to track

body movements. The system analyzes the form, provides feedback if there are issues, and updates progress. The user adjusts based on feedback, and the process repeats until the set is complete. This sequence ensures real-time feedback and accurate form tracking.

c) Activity diagram

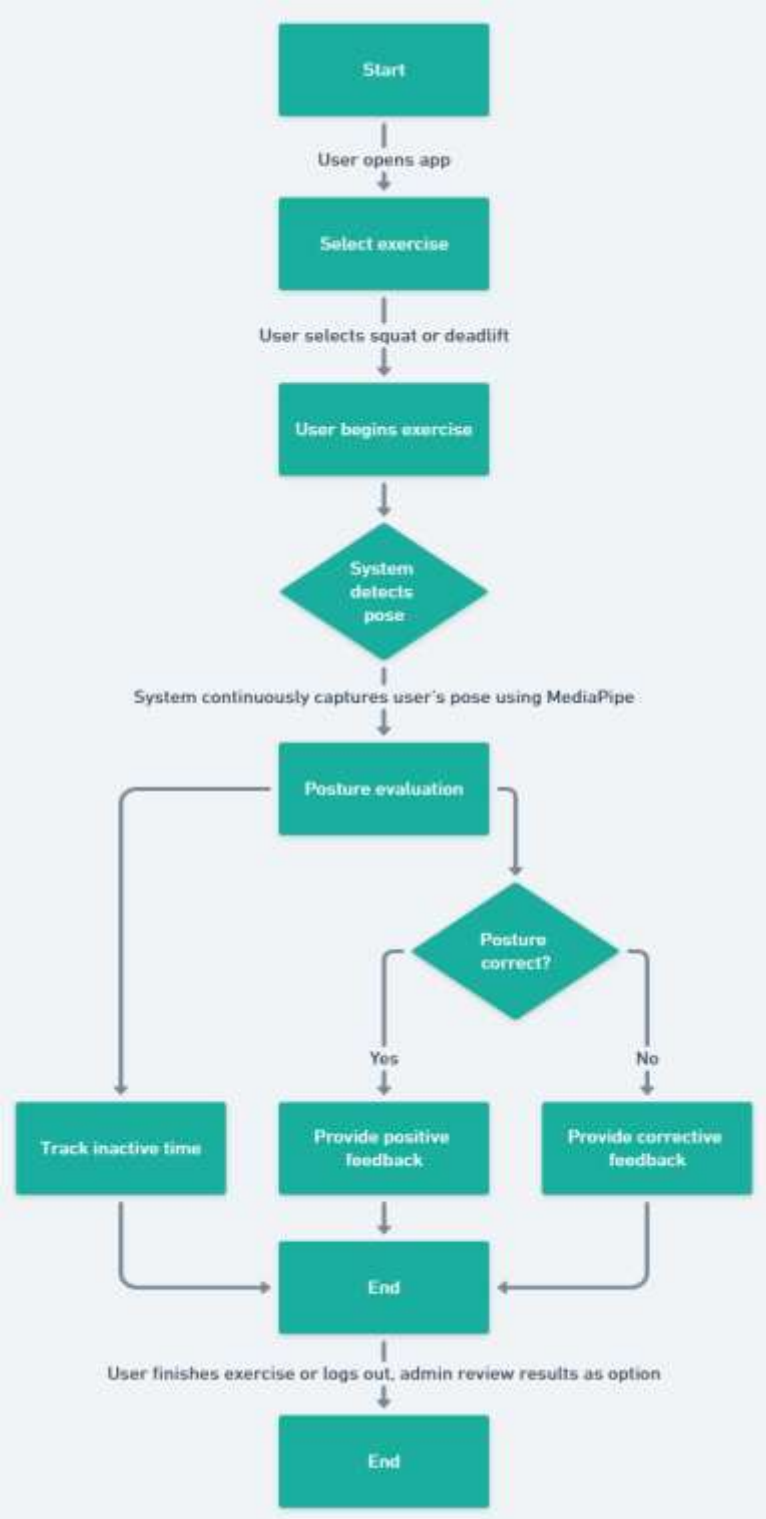



*Figure 14: Activity diagram Pose*

The Activity Diagram outlines the process flow during squat and deadlift exercises. The user starts the activity, and the system tracks their pose using MediaPipe. It then analyzes the form, gives feedback for corrections, and updates progress. The user makes adjustments, and the system continuously evaluates the form. This flow ensures continuous feedback and helps users improve their exercise technique for better results.

### d. Use management

| | |
|---|---|
| Use case | Use registration and dashboard |
| Actors | User, Admin, System |
| Descriptions | The user registers on the platform by entering personal details (name, email, password, etc.), and the system creates an account. After registration, the user can access their dashboard to view progress, track workouts, nutrition, and set goals. Admins manage user data, monitor activity, and make changes to user settings as needed. The system provides secure login and ensures personalized data is accessible in the dashboard. |

a) Use case diagram

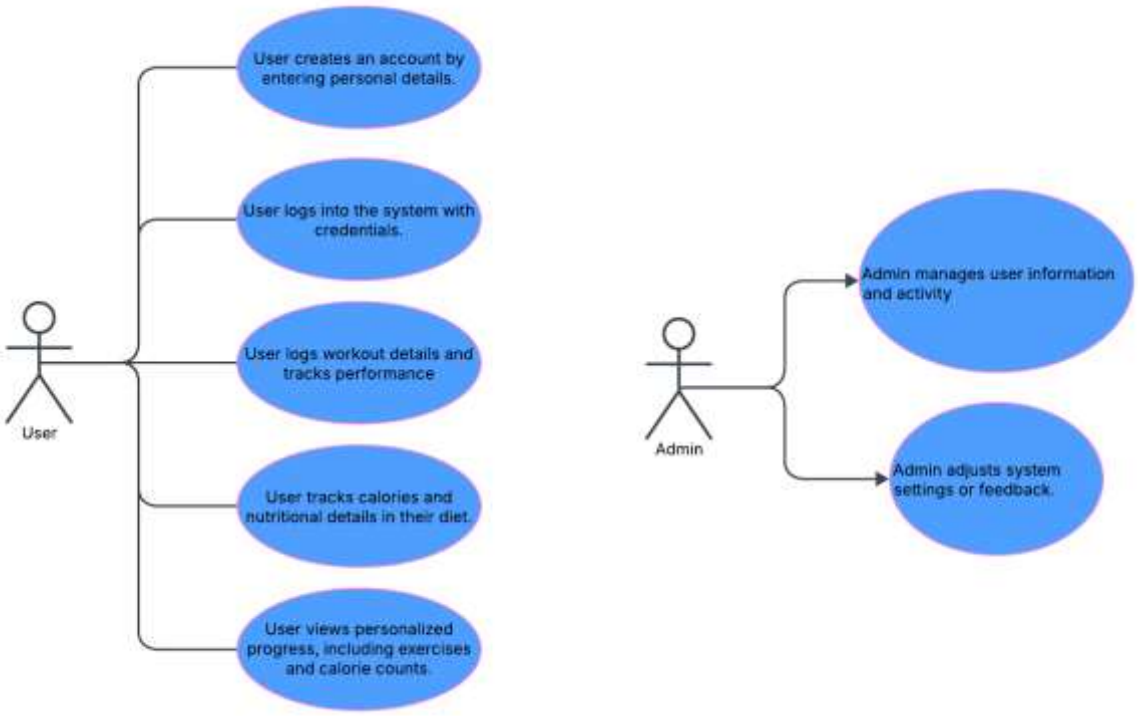



*Figure 15: Use case User management*

The Use Case Diagram for user management and the dashboard illustrates interactions between the User, Admin, and the System. The User registers, logs in, tracks exercises, and monitors their nutrition (including calories). The Admin manages user data, activity, and system settings. The System handles registration, authentication, exercise tracking, calorie monitoring, and dashboard display. The diagram shows how these actors interact to provide a seamless user experience and progress tracking.

b) Sequence diagram

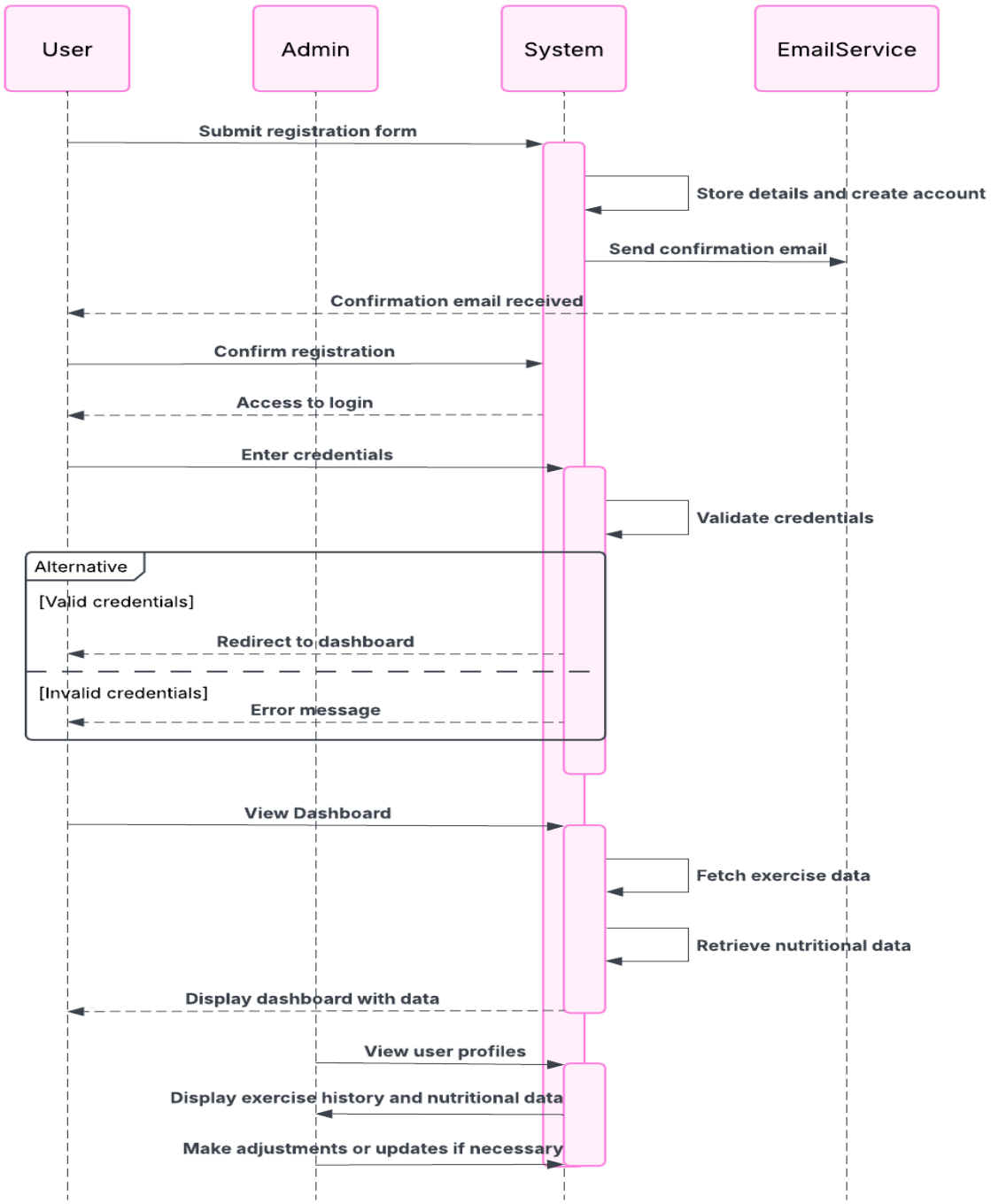



*Figure 16: Sequence diagram user management*

The Sequence Diagram illustrates the process of user registration, login, and dashboard viewing. It starts with the User registering by providing personal details. The System validates and creates an account. Upon login, the System authenticates the credentials.

After login, the User’s dashboard is populated with exercise and calorie data. Admins can monitor and adjust user settings or data. This flow ensures that the user gets personalized progress on exercises and nutrition after authentication.

c) Activity diagram

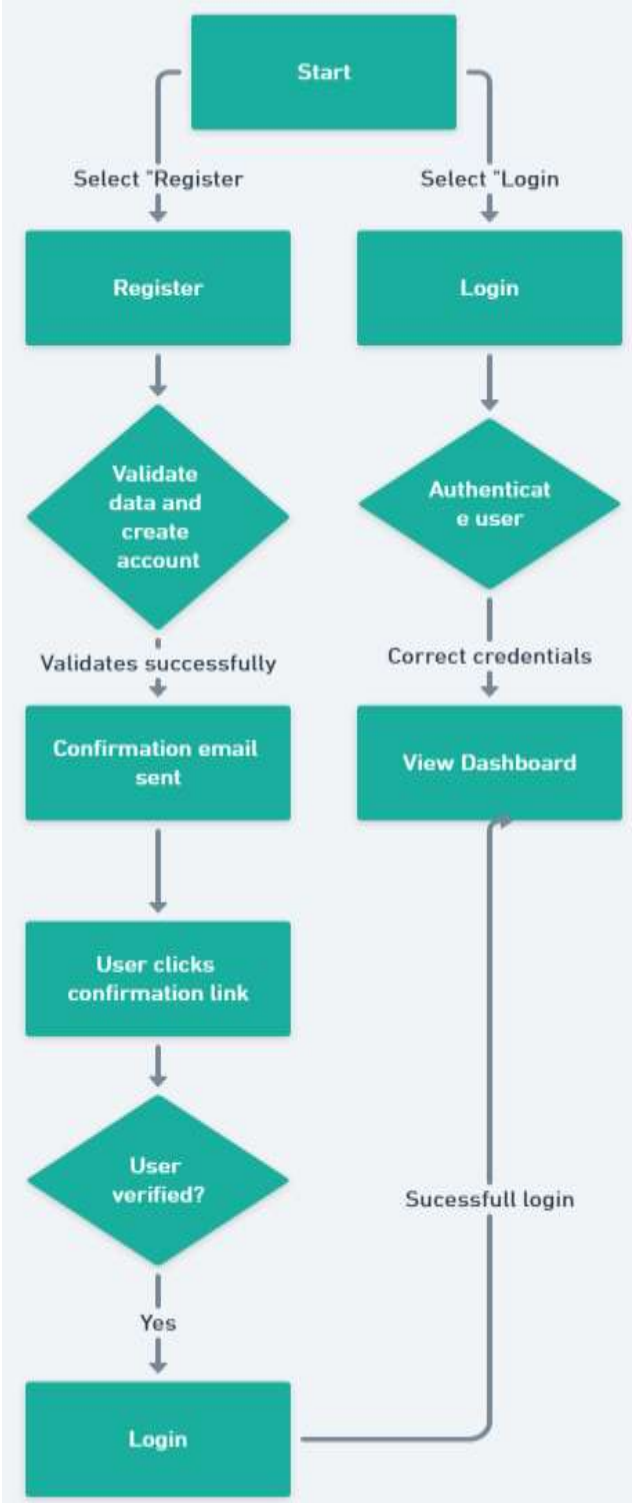



*Figure 17: Activiy diagram User management*

The Activity Diagram defines the step-by-step flow from registration to login and dashboard viewing. The user starts by either registering or logging in. After validation, the System authenticates and redirects the user to their personalized dashboard. The dashboard includes exercise progress and nutritional information (calorie counts). Admins have access to monitor and adjust user data, ensuring effective management of the platform. The diagram outlines the key steps for user management and data access in the app.

## 4. Conclusion

The design of the components discussed in this report has played a vital role in shaping the development phase of the gym application. Each diagram has provided essential insights into the system's structure, its interactions, and the workflow, offering a clear understanding of its primary functions. The Functional Decomposition Diagram has been particularly helpful in breaking down the system into manageable parts, while the Use Case and ER diagrams have clarified user interactions and database relationships. Additionally, the Class Diagram has allowed for an object-oriented approach, and the Activity Diagrams have illustrated step-by-step workflows for various processes.

During the design phase, some challenges arose, such as integrating different parts of the system and optimizing the database structure for efficient data retrieval. These challenges were addressed through iterative refinements, with testing carried out to ensure the system performed well under real-world conditions. As a result, this design phase has been crucial to the overall system development. It not only provides a clear roadmap for building the system but also serves as a solid foundation for future improvements. This phase has been instrumental in delivering the current project and will continue to guide the development of subsequent software projects.

# 5. Gantt Chart

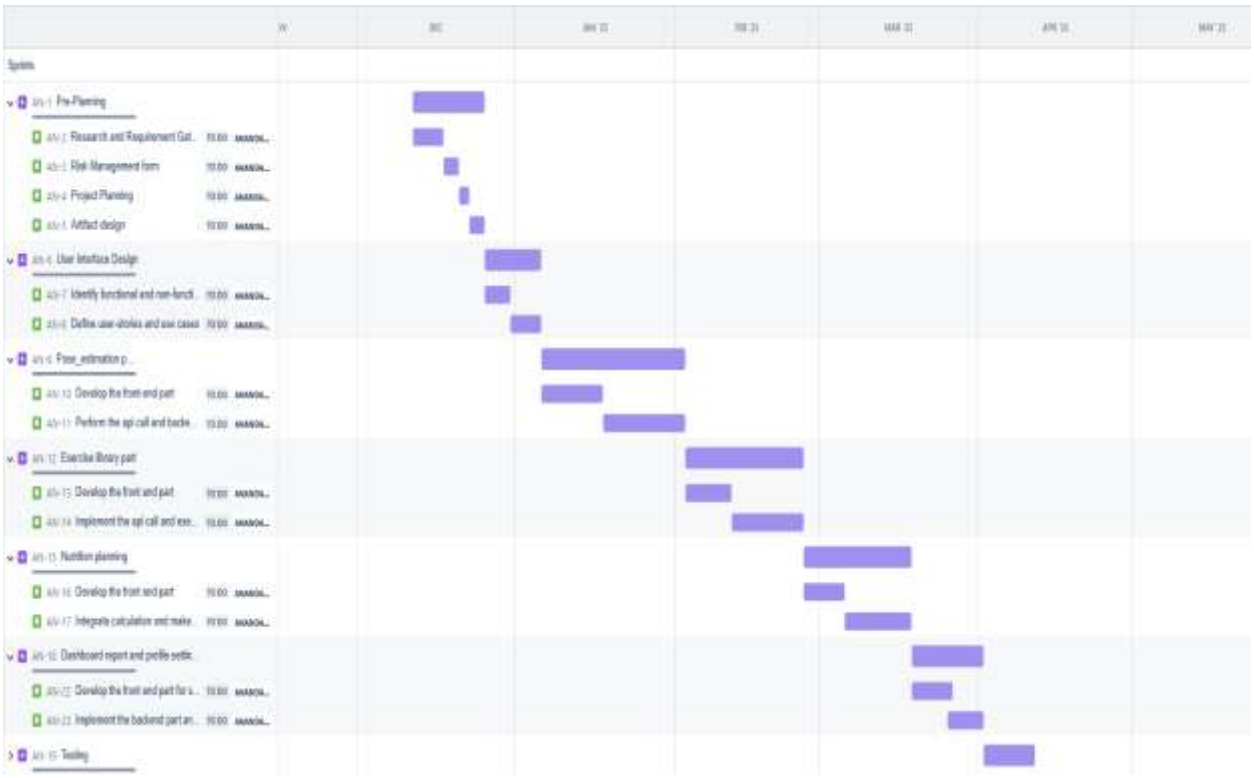


*Figure 18: Gantt Chart*

# 6. Testing plan

| Test ID | Test Scenario | Test Steps | Expected Result | Actual Result | Status (Pass/Fail) | Comments |
|---|---|---|---|---|---|---|
| Authentication Module | | | | | | |
| AUTH-02 | Forgot Password with OTP | 1. Click "Forgot Password."<br>2. Enter email/phone.<br>3. Enter OTP. | OTP sent successfully. Password reset allowed. | | | |
| AUTH-03 | JWT Token Validation | 1. Log in.<br>2. Access protected route with token. | Access granted. Token expires after set time. | | | |
| AUTH-04 | Invalid OTP Handling | 1. Enter incorrect OTP. | Error message: "Invalid OTP." | | | |
| Exercise Library Module | | | | | | |
| LIB-01 | Filter Exercises by Difficulty | 1. Select difficulty level (e.g., Beginner). | Only beginner-level exercises displayed. | | | |
| LIB-02 | Filter Exercises by Equipment | 1. Select equipment (e.g., Dumbbells). | Only exercises using dumbbells displayed. | | | |
| LIB-03 | Calculate Sets/Weight/Reps | 1. Enter weight (50 kg), reps (10), sets (3). | Total volume calculated correctly (1500 kg). | | | |
| LIB-04 | View Exercise Details | 1. Click on an exercise. | Muscle groups, instructions, and video displayed. | | | |
| Workout Logging Module | | | | | | |
| LOG-01 | Save Daily Workout Routine | 1. Create a routine.<br>2. Save routine. | Routine saved and accessible in history. | | | |
| LOG-02 | Track time after workout starts | 1. Add exercise with timer (e.g., 30 seconds). | Timer starts/stops correctly. Notification sent at end. | | | |

*Figure 19: Test Plan 1*

| | | | | | | |
|---|---|---|---|---|---|---|
| LOG-03 | Track Muscle Groups Worked | 1. Complete a workout. | Muscle groups logged and displayed in progress tracker. | | | |
| Exercise Suggestions Module | | | | | | |
| SUG-01 | Recommend Exercises Based on Goals | 1. Set goal. | Strength-focused exercises recommended. | | | |
| SUG-02 | Recommend Exercises Based on Preferences | 1. Set preference (e.g., No Machines). | Only bodyweight exercises recommended. | | | |
| Custom Workout Module | | | | | | |
| CUST-01 | Create Custom Workout | 1. Add exercises to a new routine.<br>2. Save routine. | Custom routine saved and accessible. | | | |
| CUST-02 | Modify Custom Workout | 1. Edit an existing routine.<br>2. Save changes. | Routine updated successfully. | | | |
| Personalized Nutrition Planner | | | | | | |
| NUT-01 | Create Personalized Nutrition Plan | 1. Enter user details (age, weight, goals).<br>2. Generate plan. | Nutrition plan generated based on user inputs. | | | |
| NUT-02 | Adjust Nutrition Plan Based on Preferences | 1. Set dietary preferences (e.g., vegan).<br>2. Regenerate plan. | Plan updated to exclude non-vegan items. | | | |
| NUT-03 | Track Daily Calorie Intake | 1. Log meals.<br>2. View calorie summary. | Calorie intake displayed accurately. | | | |
| NUT-04 | Suggest Recipes Based on Goals | 1. Set goal (e.g., weight loss).<br>2. View recipe suggestions. | Recipes aligned with weight loss goal displayed. | | | |

*Figure 20: Test Plan 2*

| | | | | | | |
|---|---|---|---|---|---|---|
| **Meal Reminder System** | | | | | | |
| MEAL-01 | Set Meal Reminders | 1. Set reminder for breakfast, lunch, dinner. | Reminders trigger at specified times. | | | |
| MEAL-02 | Snooze or Dismiss Reminders | 1. Snooze reminder. 2. Dismiss reminder. | Reminder snoozed for 5 minutes or dismissed permanently. | | | |
| MEAL-03 | Customize Reminder Frequency | 1. Change reminder frequency (e.g., every 2 hours). | Reminders trigger at new frequency. | | | |
| MEAL-04 | Handle Missed Reminders | 1. Ignore reminder. | Missed reminder logged and displayed in history. | | | |
| **Real-Time Pose Estimation** | | | | | | |
| POSE-01 | Detect User Pose During Exercise | 1. Start exercise with camera enabled. | Pose detected and displayed on screen. | | | |
| POSE-02 | Provide Feedback on Incorrect Pose | 1. Perform exercise incorrectly. | Feedback displayed (e.g., "Adjust your posture"). | | | |
| POSE-03 | Handle Low Light Conditions | 1. Perform exercise in low light. | System provides warning: "Low light detected. Improve lighting." | | | |
| POSE-04 | Track Multiple Users in Frame | 1. Two users perform exercises in frame. | Both users' poses detected and tracked separately. | | | |
| **Feedback on Exercise Form and Posture** | | | | | | |
| FORM-01 | Provide Real-Time Feedback on Form | 1. Perform exercise with incorrect form. | Feedback displayed (e.g., "Keep your back straight"). | | | |

*Figure 21: Test Plan 3*

| | | | | | | |
|---|---|---|---|---|---|---|
| FORM-02 | Track Progress on Form Improvement | 1. Perform exercise multiple times. | Progress tracked and displayed (e.g., "Form improved by 20%"). | | | |
| FORM-03 | Handle No User Detected | 1. Start exercise without user in frame. | Message displayed: "No user detected. Please step into frame." | | | |
| FORM-04 | Provide Audio Feedback | 1. Perform exercise with incorrect posture. | Audio feedback provided (e.g., "Straighten your arms"). | | | |

*Figure 22: Test Plan 4*

## 7. Wire frames

See my Figma wireframe designs: Click Here

## 8. References:

I have used following sites:

a) Dbdiagram for entity relationship diagram
b) Lucid chart for the rest of the diagrams
c) Jira for Gantt chart
d) Excel for test plans