HELWAN UNIVERSITY
Faculty of Computing and Artificial Intelligence
Information Systems Department

Exercise and diet by AI

A graduation project dissertation by:

Omar Zakaria Mohy ALdeen 20210603

Seif Eldin Ahmed Mohamed 20210434

Ali Ahmed Abdelmawgoud 20210573

Omar Khaled Abdo Hussien 20210599

Ziad Mohamed Abdallah Hassan 20210374

Shehab Tamer Salama 20210458

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computing & Artificial Intelligence at the **Information Systems** Department, the Faculty of Computing & Artificial Intelligence, Helwan University

Supervised by:

Dr. Mostafa Adel

June 2025

Table of Contents

Abstract

This report details the design, development, and implementation of a comprehensive Diet and Exercise Plans Website. The platform aims to provide a robust solution for individuals seeking personalized fitness and nutrition guidance, connecting them with professional coaches and leveraging artificial intelligence for tailored diet recommendations. The system supports two primary user roles: Trainees, who receive customized plans and track their progress, and Coaches, who manage their subscribed trainees and create bespoke exercise programs. Additionally, the platform fosters a vibrant community where users can share experiences and engage with one another. This document outlines the system's architecture, key features, technical specifications, and deployment strategies, providing a holistic view of the project from conception to implementation.

- **Problem:** Traditional approaches often lack the flexibility and individual tailoring required to meet diverse user needs, leading to suboptimal outcomes and disengagement.

- **Objectives:** To develop an innovative Diet and Exercise Plans Website that integrates personalized coaching with AI-driven recommendations and a supportive community environment, empowering users to achieve their health and fitness goals through structured plans, expert guidance, and peer interaction.
- **Methodology:** The project involved the design, development, and implementation of a multi-faceted web-based platform with distinct functionalities for Trainees and Coaches, alongside a shared community feature. Key areas covered include user authentication, AI-powered diet plan generation, coach-assigned exercise plan management, progress tracking, and a community forum.
- **Achievements:** A robust, scalable, and user-friendly application addressing the core needs of its target audience was developed, featuring AI integration for diet recommendations and a dynamic interaction model between Trainees and Coaches.

Keywords

Artificial Intelligence, Diet Plans, Exercise Programs, Fitness, Nutrition, Web Platform

Glossary

AI: Artificial Intelligence, the simulation of human intelligence processes by machines. **FastAPI:** A modern, fast (high-performance) web framework for building APIs with Python 3.7+ based on standard Python type hints. **Spring Boot:** An open-source, microservice-based Java web framework. **Thymeleaf:** A modern server-side Java template engine for web and standalone environments. **MySQL:** An open-source relational database management system. **PostgreSQL:** A powerful, open-source object-relational database system.

Chapter 1: An Introduction

In an increasingly health-conscious world, the demand for personalized fitness and nutrition solutions has surged. Traditional approaches often lack the flexibility and individual tailoring required to meet diverse user needs, leading to suboptimal outcomes and disengagement. This project addresses these challenges by developing an innovative Diet and Exercise Plans Website that integrates personalized coaching with AI-driven recommendations and a supportive community environment. The platform is designed to empower users to achieve their health and fitness goals through structured plans, expert guidance, and peer interaction.

1.1 Overview

The Diet and Exercise Plans Website is a multi-faceted application designed to cater to the evolving needs of fitness enthusiasts and professionals. It serves as a central hub where Trainees can access personalized diet plans generated by an AI, receive customized exercise routines from their subscribed Coaches, and track their progress over time. Coaches, on the other hand, gain a dedicated platform to manage their clientele, create and modify exercise plans, and engage with their trainees. A key feature of the system is its community page, which facilitates interaction, knowledge sharing, and motivation among all users.

1.2 Problem Statement

In the rapidly evolving landscape of health and wellness, individuals increasingly seek personalized and effective solutions to achieve their fitness and nutrition goals. However, traditional approaches to diet and exercise planning often suffer from significant limitations, leading to suboptimal outcomes and widespread user disengagement. These limitations are manifested in several key areas:

- **Lack of Personalization and Adaptability:** Generic diet plans and exercise routines fail to account for individual variations in body type, dietary preferences, health conditions, fitness levels, and lifestyle constraints. This one-size-fits-all approach often leads to plans that are difficult to adhere to, unsustainable, and ultimately ineffective for diverse user needs.

- **Limited Access to Professional Guidance:** Many individuals lack affordable and convenient access to qualified fitness coaches and nutritionists. Traditional coaching models can be expensive, geographically restrictive, and time-consuming, creating a barrier for those who could benefit most from expert guidance.

- **Inefficient Progress Tracking and Feedback:** Users often struggle to effectively track their progress, understand the impact of their efforts, and receive timely, actionable feedback. This lack of clear progress visibility can lead to demotivation and an inability to make necessary adjustments to their plans.

- **Absence of a Supportive Community:** The journey towards health and fitness can be isolating. Traditional methods often do not provide a platform for users to connect with peers, share experiences, seek motivation, or engage in a supportive environment, which is crucial for long-term adherence and success.
- **Static and Non-Responsive Recommendations:** Existing digital solutions frequently offer static recommendations that do not adapt to a user's changing needs, progress, or external factors. This lack of dynamic adjustment, particularly in diet and exercise planning, can hinder continuous improvement and goal attainment.

This project addresses these critical shortcomings by developing an innovative platform that integrates personalized, AI-driven recommendations with professional coaching and a vibrant community, thereby overcoming the limitations of conventional approaches and empowering users to achieve sustainable health and fitness outcomes.

1.3 Scope and Objectives

The scope of this project encompasses the development of a web-based platform with distinct functionalities for Trainees and Coaches, alongside a shared community feature. Key areas covered include user authentication and profile management, AI-powered diet plan generation, coach-assigned exercise plan management, progress tracking, and a community forum for user interaction. The project focuses on delivering a robust, scalable, and user-friendly application that addresses the core needs of its target audience. Future enhancements, such as advanced analytics, real-time chat functionalities, and integration with wearable devices, are considered beyond the current scope but are envisioned for subsequent development phases.

1.4 Report Organization (Structure)

This report serves as a comprehensive documentation of the Diet and Exercise Plans Website project. Its primary purpose is to detail the system’s requirements, architectural design, implementation methodologies. It aims to provide stakeholders, including project managers, developers, and potential users, with a clear understanding of the system's functionalities, technical underpinnings, and strategic objectives. Furthermore, this document will highlight the innovative aspects of the platform, such as the integration of AI for diet recommendations and the dynamic interaction model between Trainees and Coaches.

Chapter 2: Requirements

2. Functional/ Non-functional Requirements

The Diet and Exercise Plans Website is designed to meet the diverse needs of its users, encompassing both functional and non-functional requirements to ensure a robust, scalable, and user-friendly platform.

Functional Requirements

Functional requirements define the specific actions or services the system must perform. Based on the project description and provided documentation, the key functional requirements are:

- User Management:
  - The system shall allow users (Trainees and Coaches) to register and create accounts.
  - The system shall allow registered users to log in securely using their credentials.
  - The system shall allow users to view and edit their profile information, including name, email, and profile photo.
  - The system shall support two distinct user roles: Trainee and Coach.
- Diet Plan Management (Trainee & AI):
  - The system shall allow Trainees to initiate the creation of a diet plan using AI.
  - The system shall prompt Trainees to input personal metrics (e.g., height, weight) for AI-driven diet plan generation.
  - The system shall use AI to generate personalized diet plans based on the Trainee's input metrics and preferences.
  - The system shall allow Trainees to view their generated diet plans.
  - The system shall allow Trainees to accept or request adjustments to the AI-generated diet plan.
- Exercise Plan Management (Trainee & Coach):

  - The system shall allow Trainees to subscribe to a Coach to receive personalized exercise plans.
  - The system shall allow Coaches to view a list of all Trainees subscribed to them.
  - The system shall allow Coaches to create new exercise plans for their subscribed Trainees.
  - The system shall allow Coaches to edit existing exercise plans for their subscribed Trainees.
  - The system shall allow Trainees to view their assigned exercise plans.
  - The system shall allow Trainees to track and update their progress on exercise plans (e.g., weight, repetitions).
- Community Features:
  - The system shall provide a community page accessible to all users (Trainees and Coaches).
  - The system shall allow users to upload posts to the community page.
  - The system shall allow users to view posts uploaded by other users on the community page.
  - The system shall implement content moderation for community posts to ensure appropriate content.
- Communication:
  - The system shall allow Coaches to notify Trainees about new or modified exercise plans.

Non-Functional Requirements

Non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviors. These include:

- Performance:
  - The system shall respond to user requests within an acceptable time frame (e.g., login within 2 seconds, plan generation within 5 seconds).
  - The system shall be capable of handling a large number of concurrent users without significant degradation in performance.
- Security:
  - The system shall ensure secure user authentication and authorization mechanisms.

  - User data, especially personal metrics and plan details, shall be encrypted both in transit and at rest.
  - The system shall protect against common web vulnerabilities (e.g., SQL injection, XSS).
- Usability:
  - The user interface shall be intuitive and easy to navigate for both Trainees and Coaches.
  - The system shall provide clear feedback to users on their actions and system status.
  - The system shall be accessible on various devices (desktop, mobile) with a responsive design.
- Reliability:
  - The system shall be available 99.9% of the time.
  - The system shall have robust error handling and recovery mechanisms.
  - Data integrity shall be maintained through proper validation and database management.
- Scalability:
  - The system architecture shall support future expansion in terms of user base, features, and data volume.
  - The system shall be able to integrate new AI models or coaching tools as needed.
- Maintainability:
  - The codebase shall be well-documented and modular, facilitating easy updates and bug fixes.
  - The system shall adhere to established coding standards and best practices.
- Compatibility:
  - The system shall be compatible with modern web browsers.

2.3 System Analysis & Design

Use Case Analysis

Use case analysis describes the interactions between users (actors) and the system to achieve specific goals. This section details the actors involved, the identified use cases, and their descriptions and scenarios.

Actors

The primary actors interacting with the Diet and Exercise Plans Website are:

- User: The most general actor, representing any individual interacting with the system. Both Trainees and Coaches are specialized types of Users.
- Trainee: A specialized user who seeks fitness guidance, programs, and community engagement. Trainees interact with coaches, follow training programs, and participate in the platform's community.
- Coach: A specialized user who provides training programs, manages trainees, and contributes to the community.

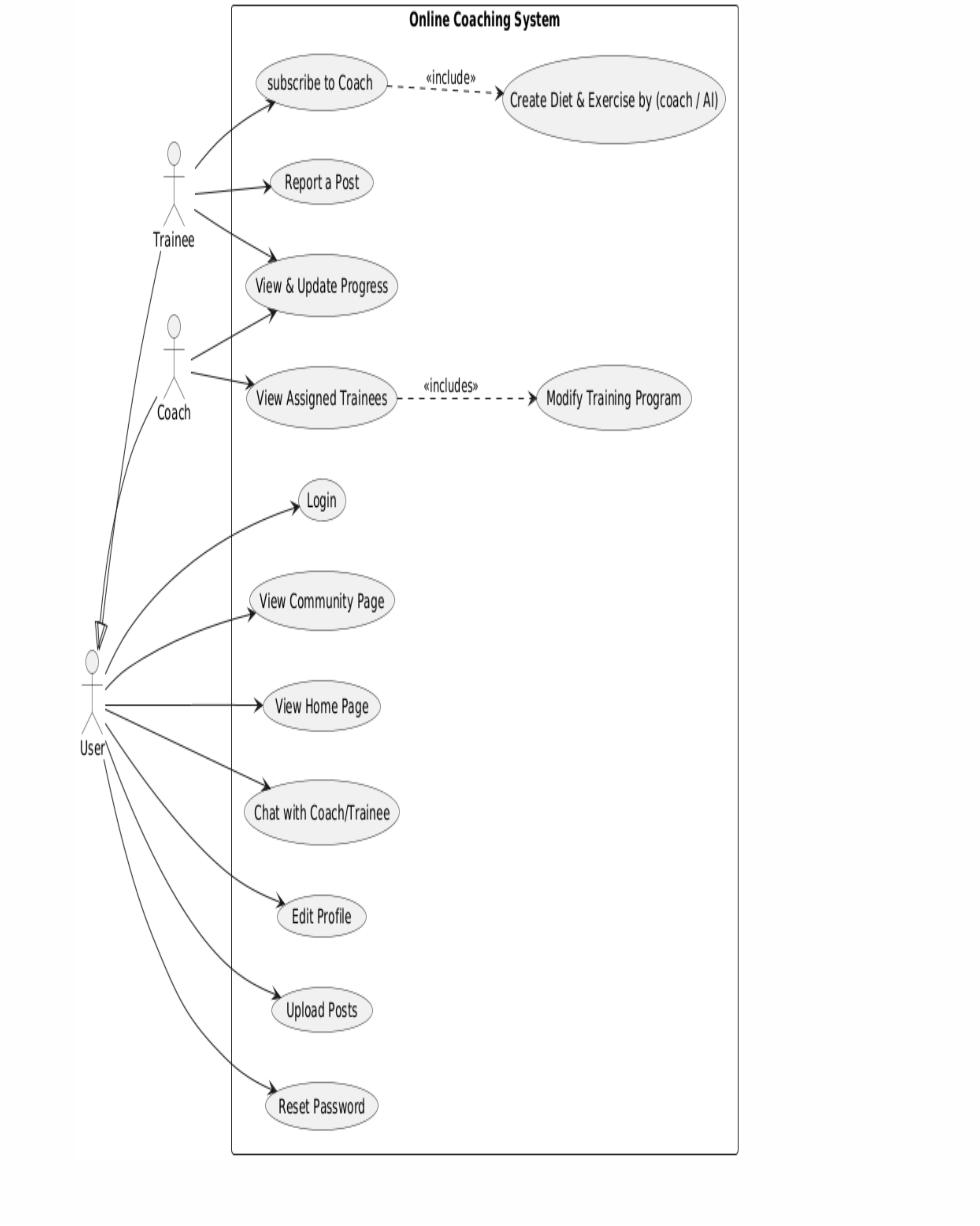
Use Cases

- Login: Allows a registered User (Trainee or Coach) to gain access to the Online Coaching System by providing valid credentials.
- View Community Page: Enables a User to access and browse the community forum, where they can view posts shared by other users.
- View Home Page: Allows a User to view their personalized home page. For Trainees, this includes exploring system features and viewing coach's programs. For Coaches, this includes checking subscribed trainees and managing programs.
- Upload Posts: Enables a User (Trainee or Coach) to create and publish content to the community page.
- Subscribe to Coach: Allows a Trainee to subscribe to a Coach's services, which includes the creation of a personalized diet and exercise plan.
- Create Diet & Exercise by (coach / AI): This is an included use case of 'Subscribe to Coach'. It involves either the Coach or an AI system generating a personalized diet and exercise plan for the Trainee.
- View & Update Progress: Enables a Trainee to view their current progress and update their records within the system.
- View Assigned Trainees: Allows a Coach to view a list of all Trainees assigned to them.

- Modify Training Program: This is an included use case of ‘View Assigned Trainees’. It allows a Coach to adjust or update the training program for an assigned Trainee.

Use Case Diagram

The use case diagram visually represents the relationships between actors and use cases. The provided usecasediagram.png illustrates these interactions:



Use Case Descriptions and Scenarios

Key scenarios include:

- Login: Successful login for Trainee and Coach, failed login due to invalid credentials or account lock.
- View Community Page: Successfully viewing the community page.
- View Home Page: Trainee and Coach viewing their respective home pages.
- Upload Posts: Successful post upload by Trainee and Coach, failed post upload due to content moderation.
- Subscribe to Coach: Successful subscription by a Trainee.
- Create Diet & Exercise by (coach / AI): Coach creating a plan, AI creating a plan.
- View Assigned Trainees: Coach successfully viewing assigned trainees.
- Modify Training Program: Coach successfully modifying a training program.

Detailed Use Case Descriptions:

Use Case: Login

Actors: User (Trainee, Coach)

Preconditions: The User has an account in the Online Coaching System.

Postconditions: The User is successfully logged in and redirected to their respective home page (Trainee Home Page or Coach Home Page).

Main Flow:

1. The User navigates to the login page.

2. The System displays the login form, requesting username/email and password.

3. The User enters their credentials and submits the form.

4. The System validates the credentials.

5. If credentials are valid, the System logs in the User and redirects them to their home page.

Alternative Flows:

Invalid Credentials: If credentials are invalid, the System displays an error message and prompts the User to try again.

Account Locked: If the account is locked due to multiple failed attempts, the System informs the User and provides instructions for unlocking.

Use Case: View Community Page

Actors: User (Trainee, Coach)

Preconditions: The User is logged into the system.

Postconditions: The User can view all posts on the community page.

Main Flow:

1. The User navigates to the Community Page.
2. The System displays all public posts, ordered by recency.
3. The User can scroll through the posts and view their content.

Use Case: View Home Page

Actors: User (Trainee, Coach)

Preconditions: The User is logged into the system.

Postconditions: The User is presented with their personalized home page, displaying relevant information based on their role.

Main Flow (Trainee):

1. The Trainee logs in or navigates to the Home Page.

2. The System displays the Trainee's personalized dashboard, including progress overview, subscribed coach information, and recommended programs.

3. The Trainee can view their current status and access quick links.

Main Flow (Coach):

1. The Coach logs in or navigates to the Home Page.

2. The System displays the Coach's personalized dashboard, including a list of assigned trainees, program management tools, and recent activities.

3. The Coach can manage their trainees and programs from this page.

Use Case: Upload Posts

Actors: User (Trainee, Coach)

Preconditions: The User is logged into the system.

Postconditions: The User's post is successfully published on the community page.

Main Flow:

1. The User navigates to the Upload Post section or initiates a new post from the Community Page.

2. The System provides an interface for creating a new post .

3. The User submits the post. 5. The System processes the post and publishes it to the Community Page.

Alternative Flows:

Content Moderation: If the post contains inappropriate content, the System may flag it for review or prevent its publication.

Use Case: Subscribe to Coach

Actors: Trainee

Preconditions: The Trainee is logged in and has selected a Coach to subscribe to.

Postconditions: The Trainee is subscribed to the Coach, and a diet and exercise plan is created.

Main Flow:

1. The Trainee selects a Coach to subscribe to.

2. The System presents subscription details and confirmation.

3. The Trainee confirms the subscription.

4. The System initiates the 'Create Diet & Exercise by (coach / AI)' use case.

5. The System confirms the successful subscription to the Trainee.

Use Case: Create Diet & Exercise by (coach / AI)

Actors: Coach, AI (system)

Preconditions: A Trainee has subscribed to a Coach.

Postconditions: A personalized diet and exercise plan is generated and assigned to the Trainee.

Main Flow (Coach):

1. Upon Trainee subscription, the Coach is notified.

2. The Coach accesses the Trainee's profile and requirements.

3. The Coach designs and inputs the diet and exercise plan into the system.

4. The System saves and assigns the plan to the Trainee.

Main Flow (AI):

1. Upon Trainee subscription, the System (AI) automatically gathers Trainee data (e.g., goals, preferences, health data).

2. The AI generates a personalized diet and exercise plan based on the collected data.

3. The System assigns the plan to the Trainee.

Use Case: View & Update Progress

Actors: Trainee

Preconditions: The Trainee is logged in and has an assigned diet and exercise plan.

Postconditions: The Trainee's progress is updated and reflected in their profile.

Main Flow:

1. The Trainee navigates to the 'Progress' section.

2. The System displays the Trainee's current progress (e.g., weight, workout completion, diet adherence).

3. The Trainee inputs new progress data (e.g., updated weight, completed workout sessions).

4. The Trainee saves the updates.

5. The System records the new progress data and updates relevant charts/metrics.

Use Case: View Assigned Trainees

Actors: Coach

Preconditions: The Coach is logged in.

Postconditions: The Coach can view a list of all Trainees subscribed to them.

Main Flow:

1. The Coach navigates to the 'My Trainees' or 'Assigned Trainees' section.

2. The System displays a list of all Trainees currently subscribed to the Coach.

3. The Coach can select a Trainee from the list to view their details or modify their program.

4. The System initiates the 'Modify Training Program' use case upon selection.

Use Case: Modify Training Program

Actors: Coach

Preconditions: The Coach is viewing an assigned Trainee's profile.

Postconditions: The Trainee's training program is updated with the Coach's modifications.

Main Flow:

1. The Coach selects a Trainee from their assigned list.

2. The System displays the Trainee's current training program.

3. The Coach makes necessary adjustments to the program (e.g., exercise changes, diet modifications, schedule updates).

4. The Coach saves the modified program.

5. The System updates the Trainee's program and notifies the Trainee of the changes.

Use Case Scenarios:

Use Case: Login

Scenario 1: Successful Login (Trainee)

Preconditions: Trainee 'John Doe' has a registered account with username 'john.doe' and password 'password123'.

Steps:

1. John Doe navigates to the Online Coaching System login page.

2. He enters 'john.doe' in the username field and 'password123' in the password field.

3. He clicks the 'Login' button.

Expected Outcome: John Doe is successfully logged in and redirected to his Trainee Home Page.

Scenario 2: Successful Login (Coach)

Preconditions: Coach 'Jane Smith' has a registered account with username 'jane.smith' and password 'coachpass'.

Steps:

1. Jane Smith navigates to the Online Coaching System login page.

2. She enters 'jane.smith' in the username field and 'coachpass' in the password field.

3. She clicks the 'Login' button.

Expected Outcome: Jane Smith is successfully logged in and redirected to her Coach Home Page.

Scenario 3: Failed Login - Invalid Credentials

Preconditions: User attempts to log in with incorrect credentials.

Steps:

1. A User navigates to the login page.

2. They enter 'wronguser' in the username field and 'wrongpass' in the password field.

3. They click the 'Login' button.

Expected Outcome: The System displays an error message: "Invalid username or password. Please try again." The User remains on the login page.

Scenario 4: Failed Login - Account Locked

Preconditions: User 'john.doe' has attempted to log in 5 times with incorrect passwords, leading to account lock.

Steps:

1. John Doe navigates to the login page.

2. He enters 'john.doe' and an incorrect password.

3. He clicks the 'Login' button.

Expected Outcome: The System displays an error message: "Your account has been locked due to multiple failed login attempts. Please contact support or use the 'Forgot Password' link to unlock your account." The User remains on the login page.

Use Case: View Community Page

Scenario 1: Successfully View Community Page

Preconditions: A User is logged in.

Steps:

1. The User clicks on the 'Community' link in the navigation bar.

Expected Outcome: The Community Page loads, displaying a list of public posts, ordered by recency. The User can scroll through and view the content of the posts.

Use Case: View Home Page

Scenario 1: Trainee Views Home Page

Preconditions: Trainee 'John Doe' is logged in.

Steps:

1. John Doe is redirected to the Home Page after successful login, or clicks the 'Home' link.

Expected Outcome: The Trainee Home Page displays John Doe's personalized dashboard, including his progress overview, information about his subscribed coach, and recommended programs. Quick links to other sections are also visible.

Scenario 2: Coach Views Home Page

Preconditions: Coach 'Jane Smith' is logged in.

Steps:

1. Jane Smith is redirected to the Home Page after successful login, or clicks the 'Home' link.

Expected Outcome: The Coach Home Page displays Jane Smith's personalized dashboard, including a list of her assigned trainees, program management tools, and recent activities. She can access tools to manage her trainees and programs.

Use Case: Upload Posts

Scenario 1: Successful Post Upload (Trainee)

Preconditions: Trainee 'John Doe' is logged in.

Steps:

1. John Doe navigates to the Community Page and clicks 'Create New Post'.

2. He types a message: "Just finished my workout, feeling great!" and attaches a photo.

3. He clicks 'Publish'.

Expected Outcome: The post is successfully published on the Community Page, visible to other users. A confirmation message is displayed.

Scenario 2: Successful Post Upload (Coach)

Preconditions: Coach 'Jane Smith' is logged in.

Steps:

1. Jane Smith navigates to the Community Page and clicks 'Create New Post'.

2. She types a message: "Here are some tips for staying motivated!" and includes a link to an article.

3. She clicks 'Publish'.

Expected Outcome: The post is successfully published on the Community Page, visible to other users. A confirmation message is displayed.

Scenario 3: Failed Post Upload - Content Moderation

Preconditions: A User attempts to upload a post with inappropriate content.

Steps:

1. A User attempts to create a new post.

2. They type a message containing offensive language.

3. They click 'Publish'.

Expected Outcome: The System prevents the post from being published and displays a message: "Your post contains inappropriate content and cannot be published." The User is prompted to revise the content.

Use Case: Create Diet & Exercise by (coach / AI)

Scenario 1: Coach Creates Plan

Preconditions: Trainee 'John Doe' has subscribed to Coach 'Jane Smith'.

Steps:

1. Jane Smith receives a notification about John Doe's new subscription.

2. She accesses John Doe's profile and reviews his goals and preferences.

3. She designs a personalized diet and exercise plan and inputs it into the system.

4. She saves the plan.

Expected Outcome: A personalized diet and exercise plan is created by Jane Smith and assigned to John Doe. John Doe is notified of his new plan.

Scenario 2: AI Creates Plan

Preconditions: Trainee 'Emily White' has subscribed to a Coach, and the system is configured for AI-generated plans.

Steps:

1. Upon Emily White's subscription, the System (AI) automatically collects her data (e.g., age, weight, fitness level, dietary restrictions).

2. The AI processes this data and generates a personalized diet and exercise plan.

3. The System assigns the plan to Emily White.

Expected Outcome: A personalized diet and exercise plan is automatically generated by the AI and assigned to Emily White. Emily White is notified of her new plan.

Use Case: View Assigned Trainees

Scenario 1: Successfully View Assigned Trainees

Preconditions: Coach 'Jane Smith' is logged in.

Steps:

1. Jane Smith clicks on 'My Trainees' from her Coach Home Page.

Expected Outcome: A list of all Trainees currently subscribed to Jane Smith is displayed. Each Trainee's name and basic information are visible. Jane Smith can select a Trainee to view their details or modify their program.

Use Case: Modify Training Program

Scenario 1: Successfully Modify Training Program

Preconditions: Coach 'Jane Smith' is viewing Trainee 'John Doe's profile.

Steps:

1. Jane Smith selects John Doe from her 'My Trainees' list.

2. She navigates to John Doe's training program.

3. She changes a specific exercise from 'push-ups' to 'bench press' and updates the repetitions.

4. She clicks 'Save Program'.

Expected Outcome: John Doe's training program is updated with the new exercise and repetitions. John Doe receives a notification about the program change. A confirmation message is displayed to Jane Smith.

Chapter 3: System Architecture

The system architecture defines the overall structure of the Diet and Exercise Plans Website, outlining its components, their relationships, and how they interact to deliver the system’s functionalities. The platform is designed as a multi-system application, integrating several distinct services to provide a comprehensive fitness coaching solution.

High-Level System Architecture

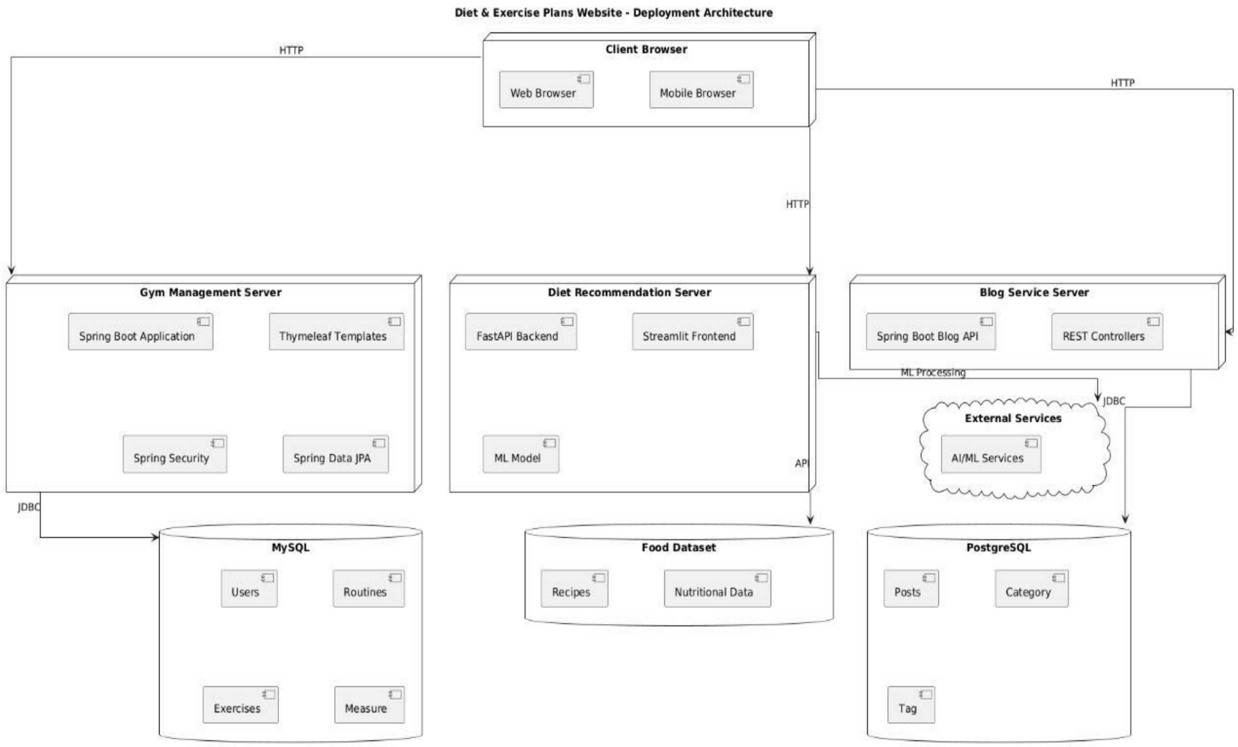
The platform consists of three independently deployable systems:

- Diet Recommendation System: This system is responsible for generating AI-powered diet recommendations. It utilizes a FastAPI backend and integrates with machine learning models to process user metrics and preferences.
- Gym Management System: This system handles core functionalities related to gym management, including user profiles, exercise plans, and progress tracking. It is built using Spring Boot and leverages Thymeleaf templates for its user interface.
- Blog System: This system manages the community page, allowing users to create and view posts. It comprises a Spring Boot backend for API services and an npm-based frontend for user interaction.

These systems work collaboratively to provide a seamless user experience, with data flowing between them to support various features such as personalized plan generation, progress monitoring, and community engagement.

Deployment Diagram

The deployment diagram illustrates the physical deployment of the software components on hardware nodes. The provided DeploymentDiagram.jpg outlines the deployment architecture of the Diet & Exercise Plans Website:

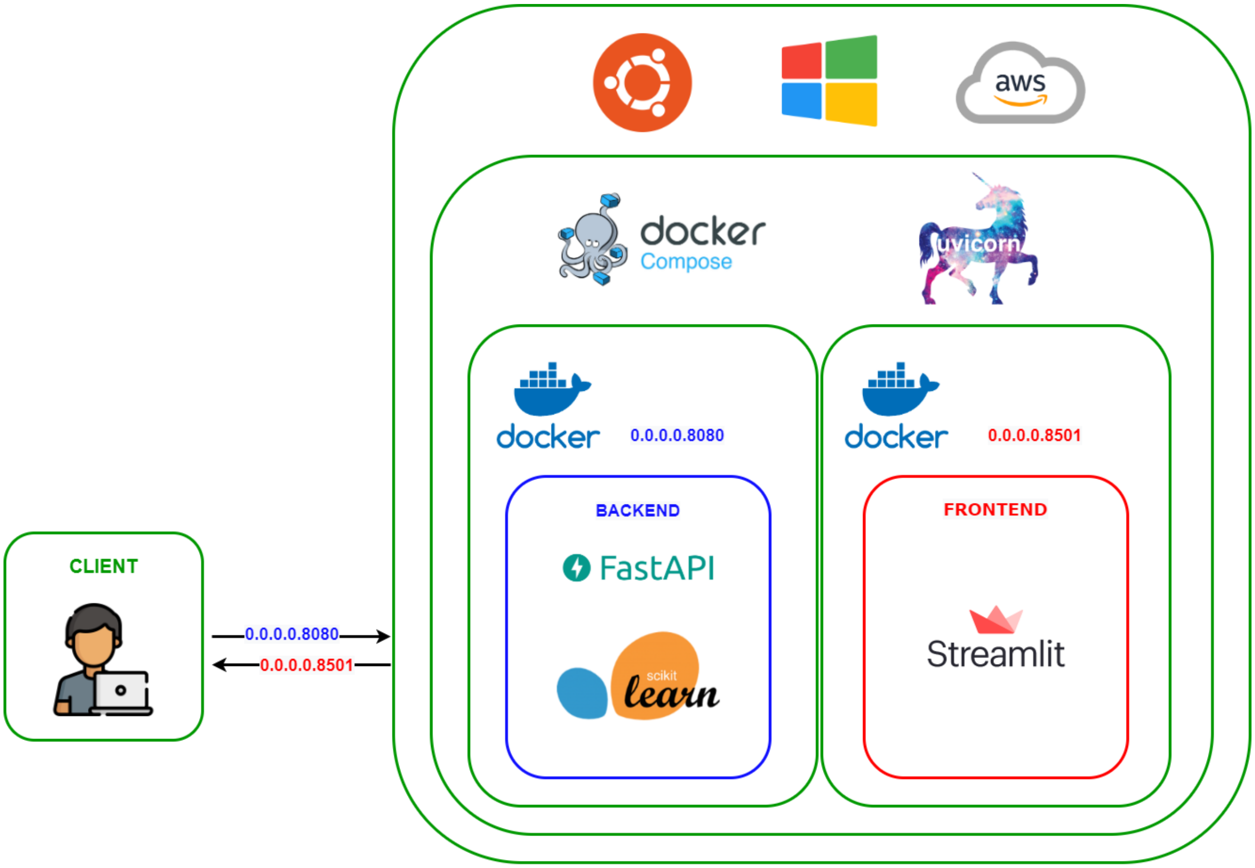


As depicted in the diagram, the system components are deployed across several servers and databases:

- Client Browser: Users interact with the system via web and mobile browsers, communicating with the backend services over HTTP.
- Gym Management Server: Hosts the Spring Boot Application for gym management, utilizing Spring Security for authentication and Spring Data JPA for database interaction. It connects to a MySQL database for storing user, routine, exercise, and measurement data.
- Diet Recommendation Server: Runs a FastAPI Backend with ML Model Integration, serving AI-powered diet recommendations. It interacts with a Food Dataset, which stores recipes and nutritional data.
- Blog Service Server: Manages the community features, running a Spring Boot Blog API with REST Controllers. It connects to a PostgreSQL database for posts, categories, and tags. This server also interacts with external AI/ML Services for potential content moderation or other intelligent features.

This distributed architecture ensures scalability, fault tolerance, and efficient resource utilization, allowing each component to be developed, deployed, and scaled independently.

AI Architecture diagram:



This diagram illustrates the architecture of a machine learning web application deployed using **Docker** and **Docker Compose**.

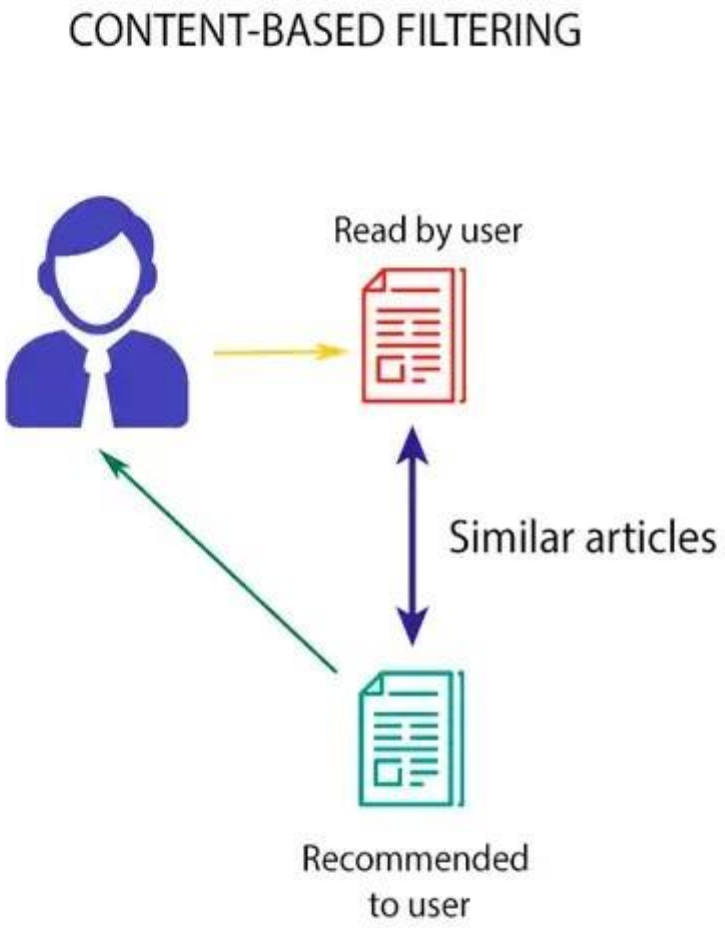
Environment:

- **Compatible OS/Platforms:** Ubuntu, Windows, AWS Cloud (shown by icons).
- **Containerization:** Managed via **Docker Compose**, which handles the setup and coordination of multiple Docker containers.

Components:

1. **Client Side (User):**
    - Interacts with the system via a web interface.
    - Accesses the app through:
        - Port **8080** (likely the backend – FastAPI).
        - Port **8501** (frontend – Streamlit).
2. **Backend Container:**
    - **FastAPI** is used to handle API requests.
    - **Scikit-learn** is used for running machine learning models (e.g., predictions or recommendations).
    - Runs on **port 8080** using **Uvicorn** as the ASGI server.
3. **Frontend Container:**
    - Built with **Streamlit**, a Python-based framework for interactive dashboards.
    - Runs on **port 8501**.
4. **Communication:**
    - Docker containers interact internally.
    - External clients can access services via exposed ports.

Contet-Based Filtering:



This diagram explains a **Content-Based Filtering** recommender system.

How It Works:

- The **user reads an article** (or interacts with any item).
- The system analyzes the **features or content** of that item.
- Based on the **content similarity**, the system finds **similar articles/items**.
- These **similar items** are recommended back to the user.

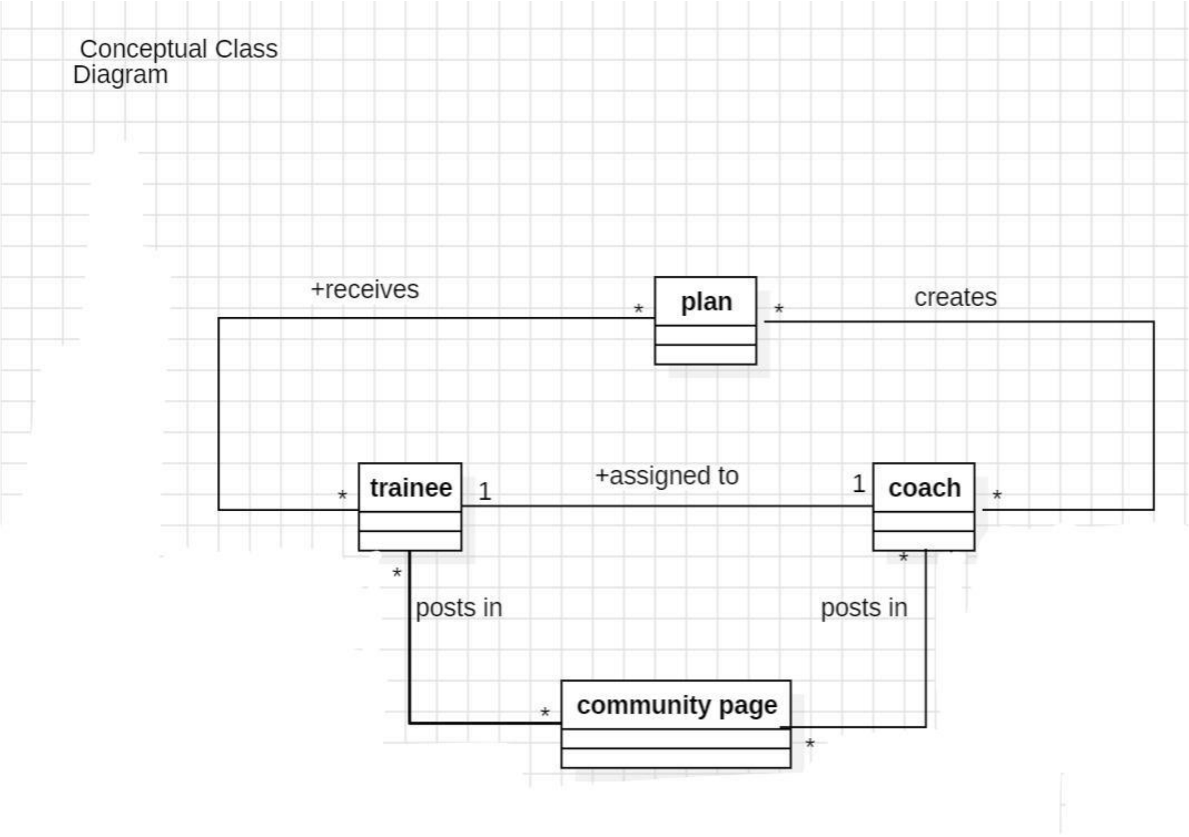
Key Concepts:

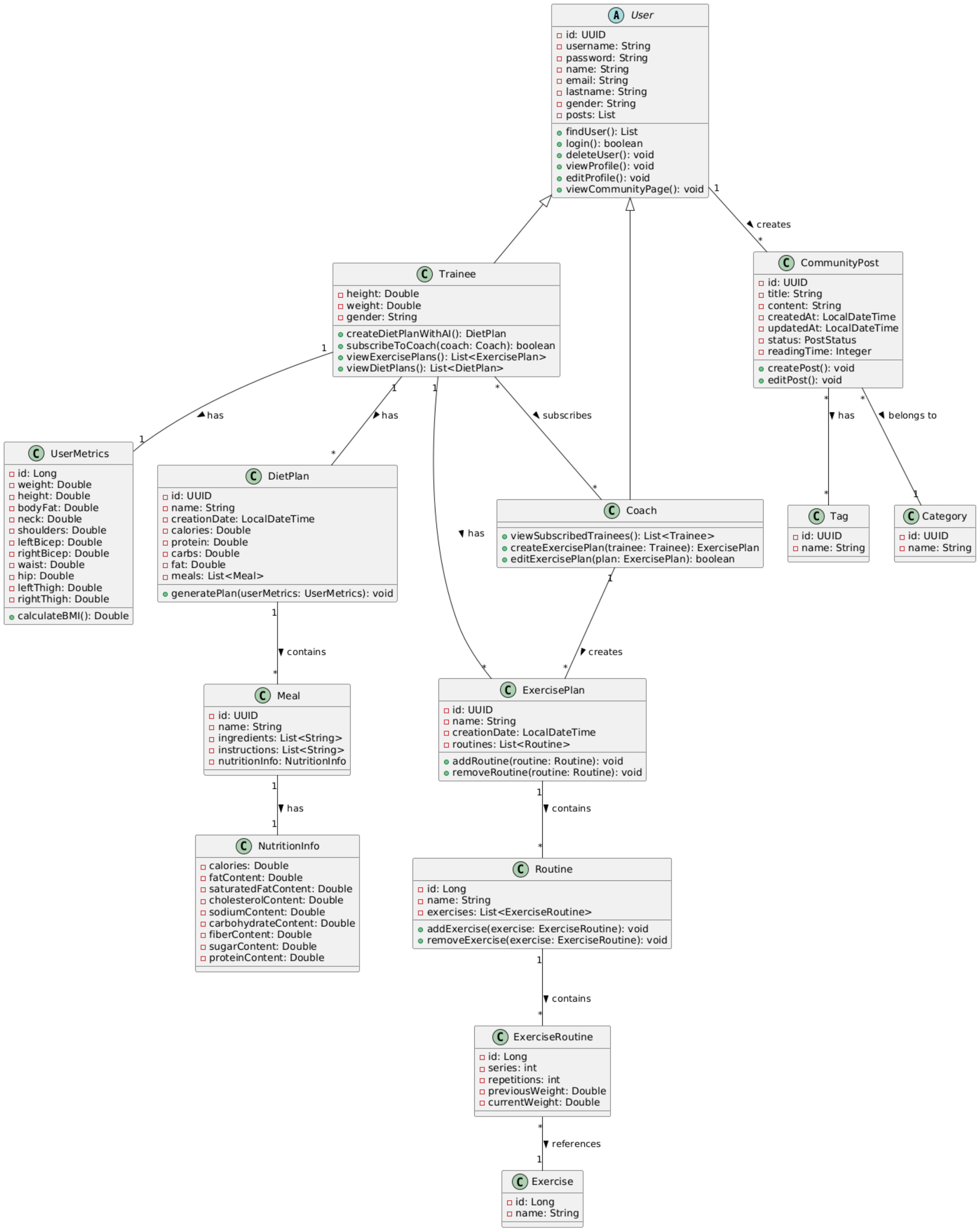
- Recommendations are **personalized based on the user's past behavior** (e.g., articles they read).
- It does **not depend on other users' data**, unlike collaborative filtering.

- Useful when:
    - There’s not much user interaction data.
    - The item content (text, tags, features) is rich.

Class Diagram

The class diagram provides a static view of the system, illustrating the classes, their attributes, operations, and the relationships between them. The provided classdiagram.png and conceptualclassdiagramv3.jpg offer a detailed representation of the system’s domain model:

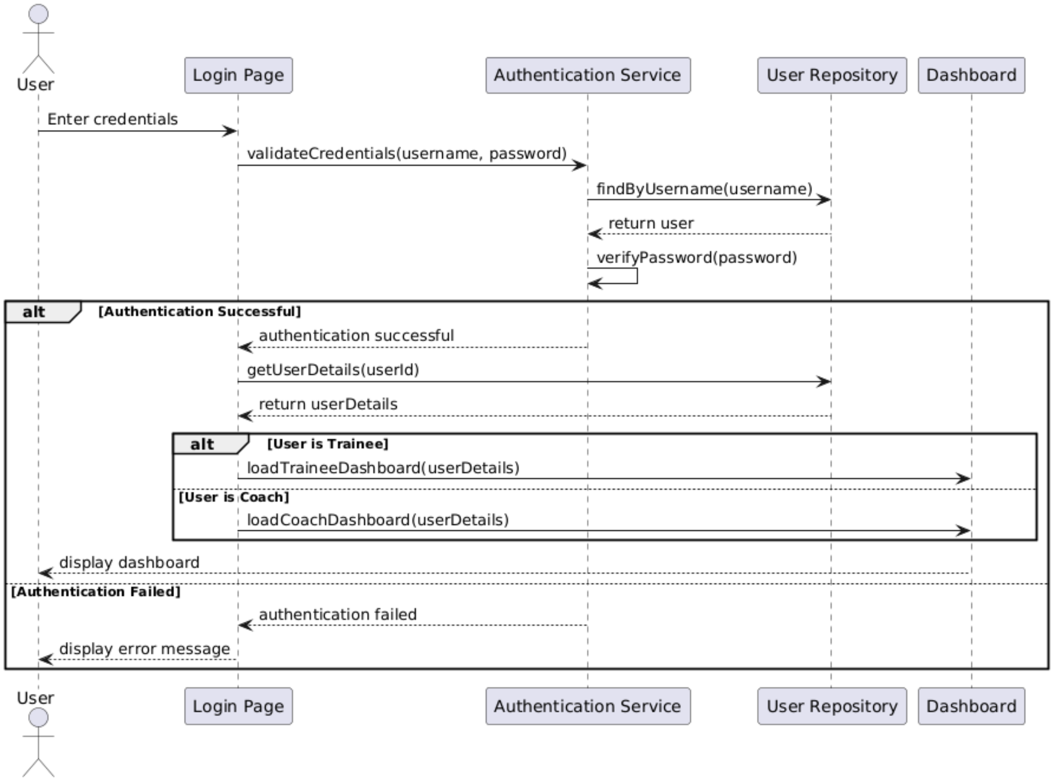



User
id: UUID
username: String
password: String
name: String
email: String
lastname: String
gender: String
posts: List
findUser(): List
login(): boolean
deleteUser(): void
viewProfile(): void
editProfile(): void
viewCommunityPage(): void
1
creates
*
Trainee
height: Double
weight: Double
gender: String
createDietPlanWithAI(): DietPlan
subscribeToCoach(coach: Coach): boolean
viewExercisePlans(): List<ExercisePlan>
viewDietPlans(): List<DietPlan>
CommunityPost
id: UUID
title: String
content: String
createdAt: LocalDateTime
updatedAt: LocalDateTime
status: PostStatus
readingTime: Integer
createPost(): void
editPost(): void
1
has
1
1
has
*
1
*
subscribes
*
*
has
*
*
belongs to
1
UserMetrics
id: Long
weight: Double
height: Double
bodyFat: Double
neck: Double
shoulders: Double
leftBicep: Double
rightBicep: Double
waist: Double
hip: Double
leftThigh: Double
rightThigh: Double
calculateBMI(): Double
DietPlan
id: UUID
name: String
creationDate: LocalDateTime
calories: Double
protein: Double
carbs: Double
fat: Double
meals: List<Meal>
generatePlan(userMetrics: UserMetrics): void
has
Coach
viewSubscribedTrainees(): List<Trainee>
createExercisePlan(trainee: Trainee): ExercisePlan
editExercisePlan(plan: ExercisePlan): boolean
Tag
id: UUID
name: String
Category
id: UUID
name: String
1
contains
*
1
creates
*
*
Meal
id: UUID
name: String
ingredients: List<String>
instructions: List<String>
nutritionInfo: NutritionInfo
ExercisePlan
id: UUID
name: String
creationDate: LocalDateTime
routines: List<Routine>
addRoutine(routine: Routine): void
removeRoutine(routine: Routine): void
1
has
1
1
contains
*
NutritionInfo
calories: Double
fatContent: Double
saturatedFatContent: Double
cholesterolContent: Double
sodiumContent: Double
carbohydrateContent: Double
fiberContent: Double
sugarContent: Double
proteinContent: Double
Routine
id: Long
name: String
exercises: List<ExerciseRoutine>
addExercise(exercise: ExerciseRoutine): void
removeExercise(exercise: ExerciseRoutine): void
1
contains
*
ExerciseRoutine
id: Long
series: int
repetitions: int
previousWeight: Double
currentWeight: Double
*
references
1
Exercise
id: Long
name: String

Sequence and Activity Diagrams

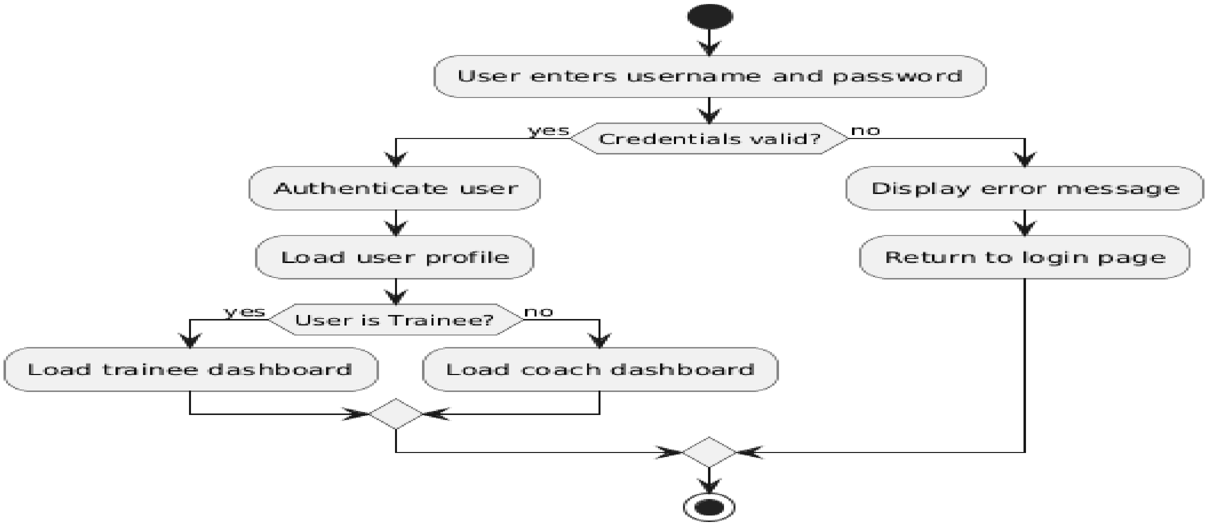
Login Sequence Diagram

The Login Sequence Diagram illustrates the interaction flow when a user attempts to log in to the system:



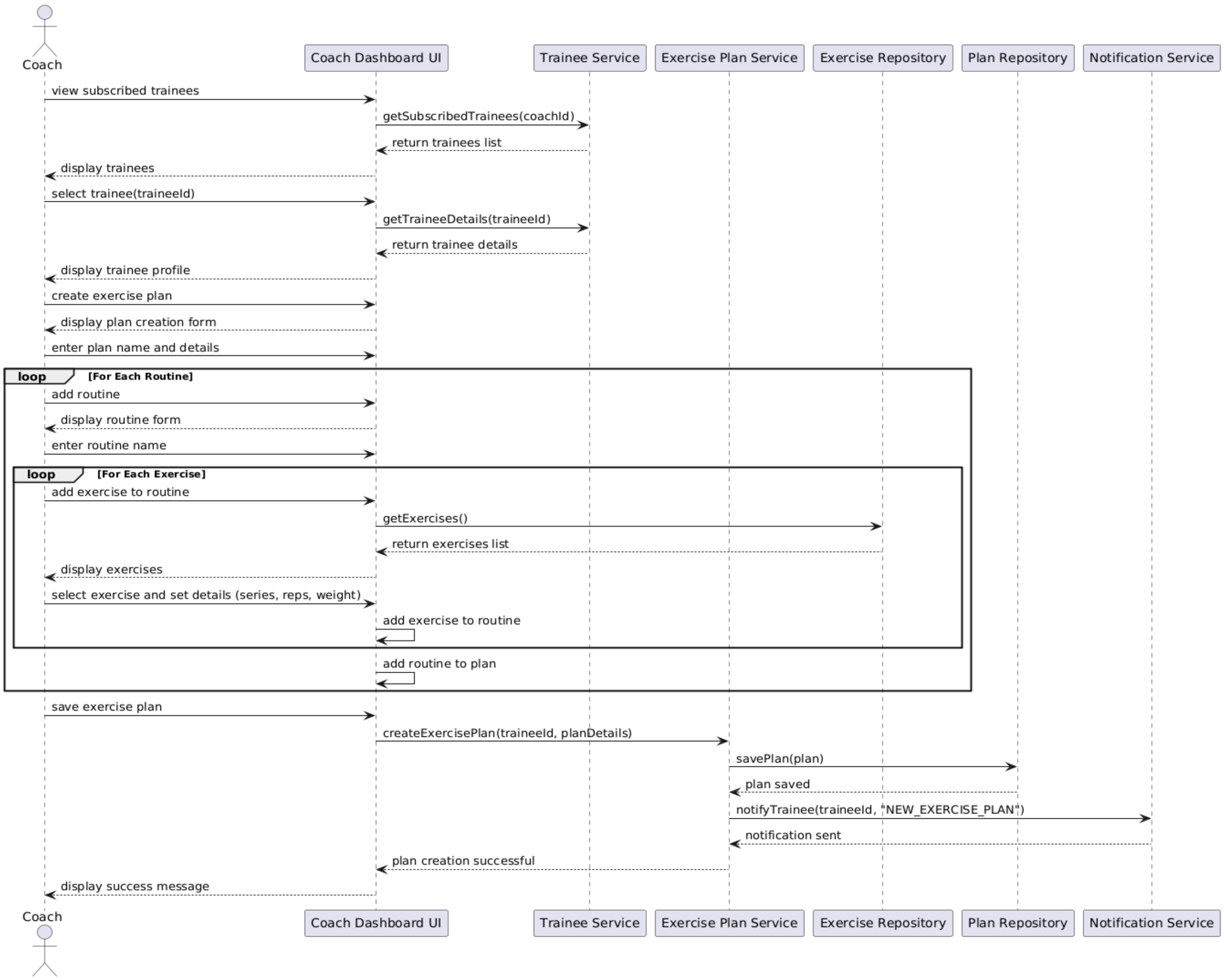
Login Activity Diagram

The Login Activity Diagram depicts the workflow of the login process:

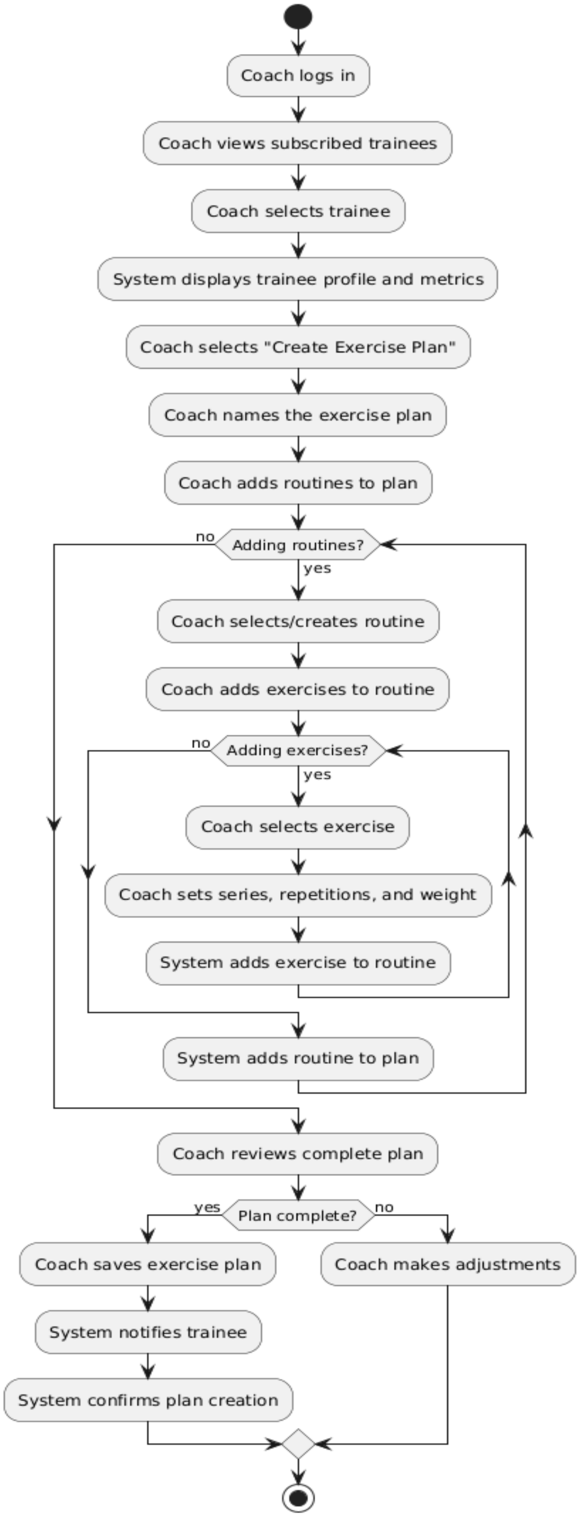


Creating Exercise Plan Sequence Diagram

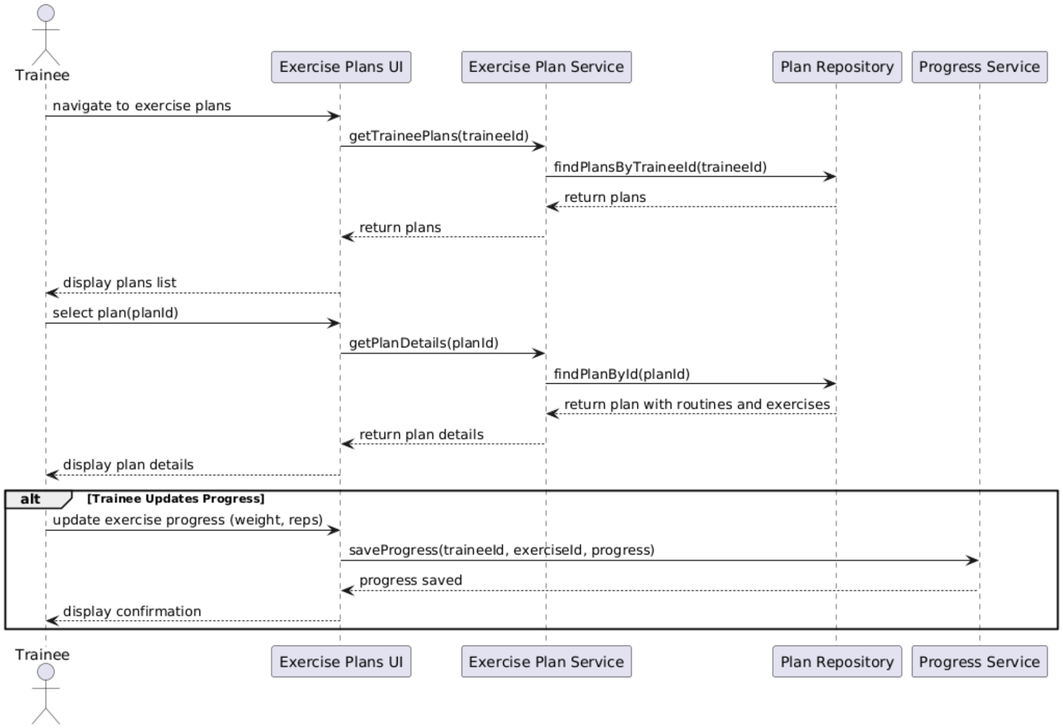
The Creating Exercise Plan Sequence Diagram shows the interactions involved when a coach creates an exercise plan:



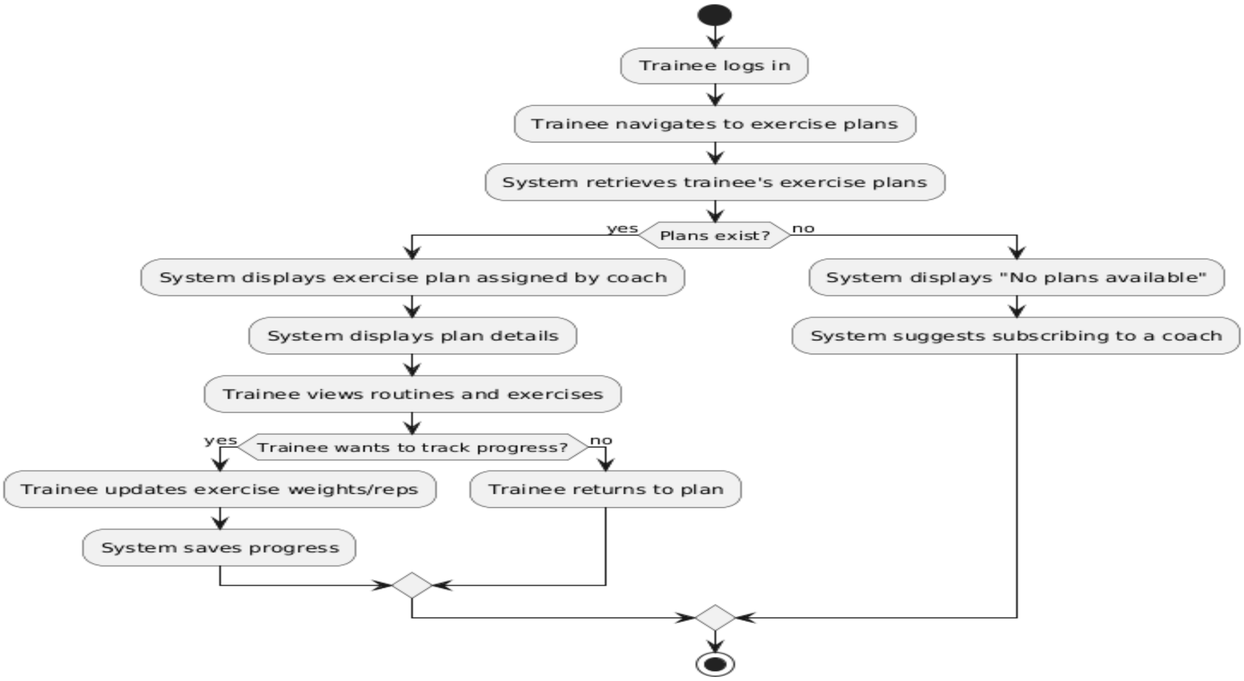
The Creating Exercise Plan Activity Diagram outlines the steps a coach takes to create an exercise plan:


Coach logs in
Coach views subscribed trainees
Coach selects trainee
System displays trainee profile and metrics
Coach selects "Create Exercise Plan"
Coach names the exercise plan
Coach adds routines to plan
Adding routines?
no
yes
Coach selects/creates routine
Coach adds exercises to routine
Adding exercises?
no
yes
Coach selects exercise
Coach sets series, repetitions, and weight
System adds exercise to routine
System adds routine to plan
Coach reviews complete plan
Plan complete?
yes
no
Coach saves exercise plan
Coach makes adjustments
System notifies trainee
System confirms plan creation

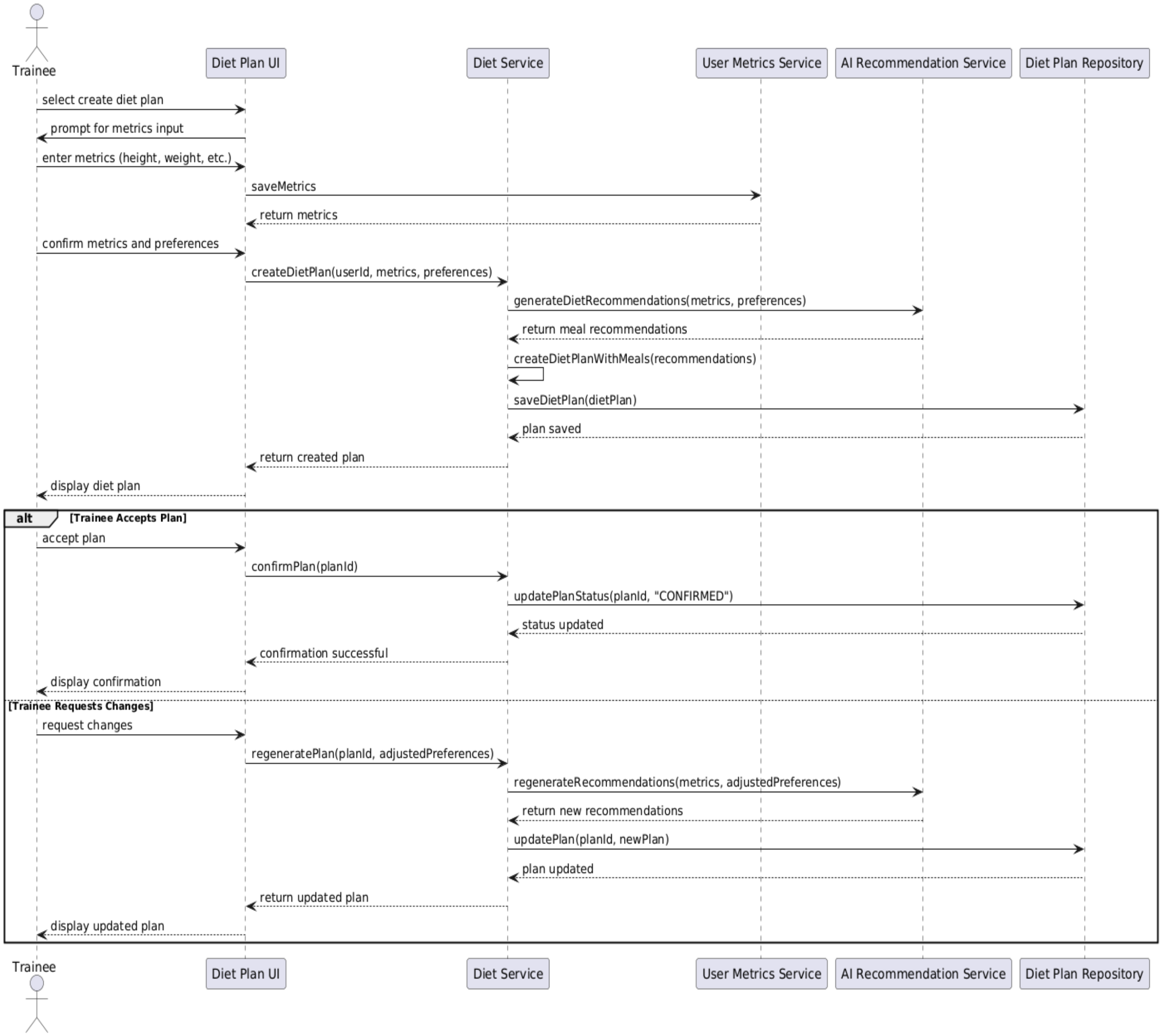
The Viewing Exercise Plan Sequence Diagram illustrates how a trainee views their exercise plan:



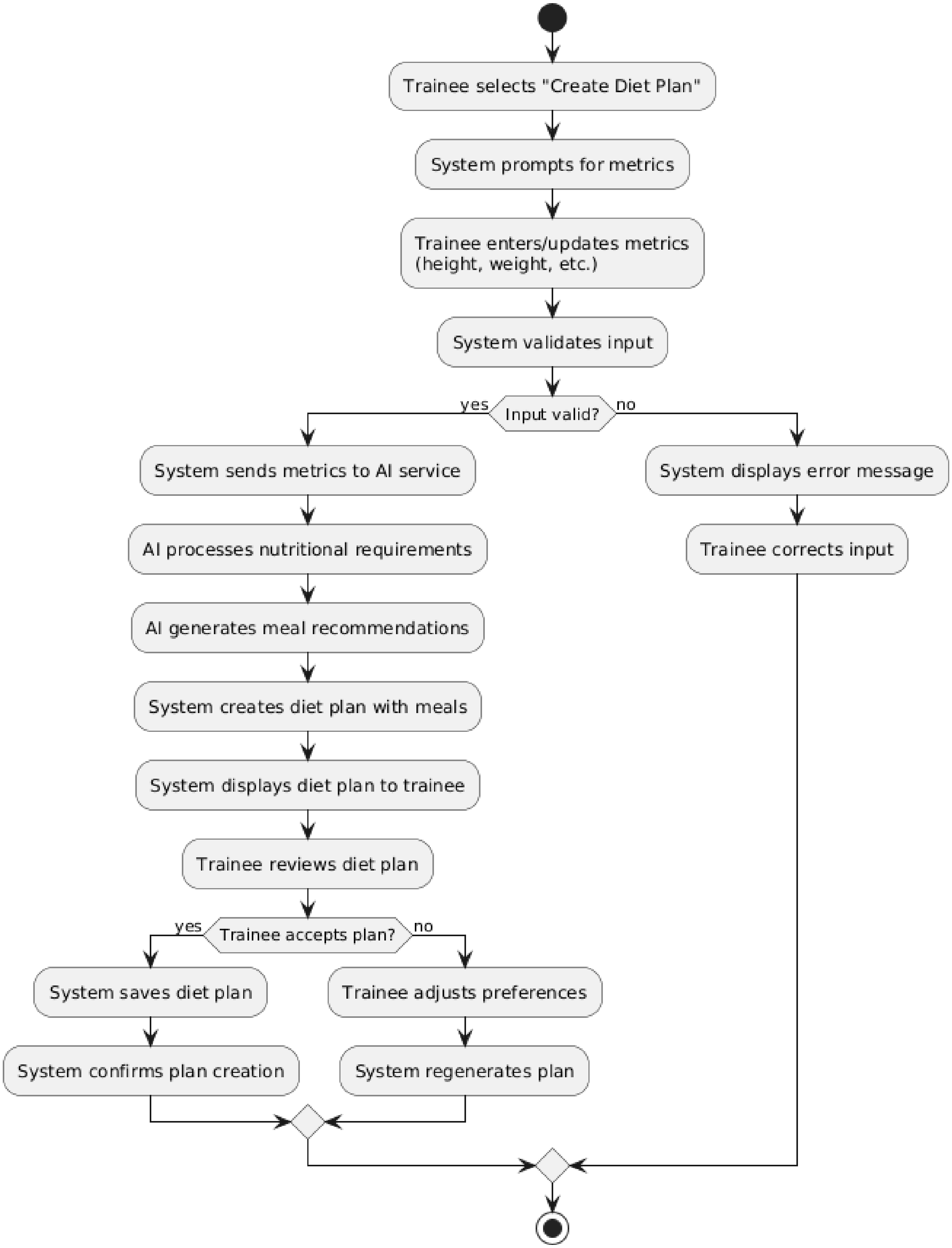
The Viewing Exercise Plan Activity Diagram shows the steps involved in a trainee viewing their exercise plan:


Trainee logs in
Trainee navigates to exercise plans
System retrieves trainee's exercise plans
yes
Plans exist?
no
System displays exercise plan assigned by coach
System displays "No plans available"
System displays plan details
System suggests subscribing to a coach
Trainee views routines and exercises
yes
Trainee wants to track progress?
no
Trainee updates exercise weights/reps
Trainee returns to plan
System saves progress

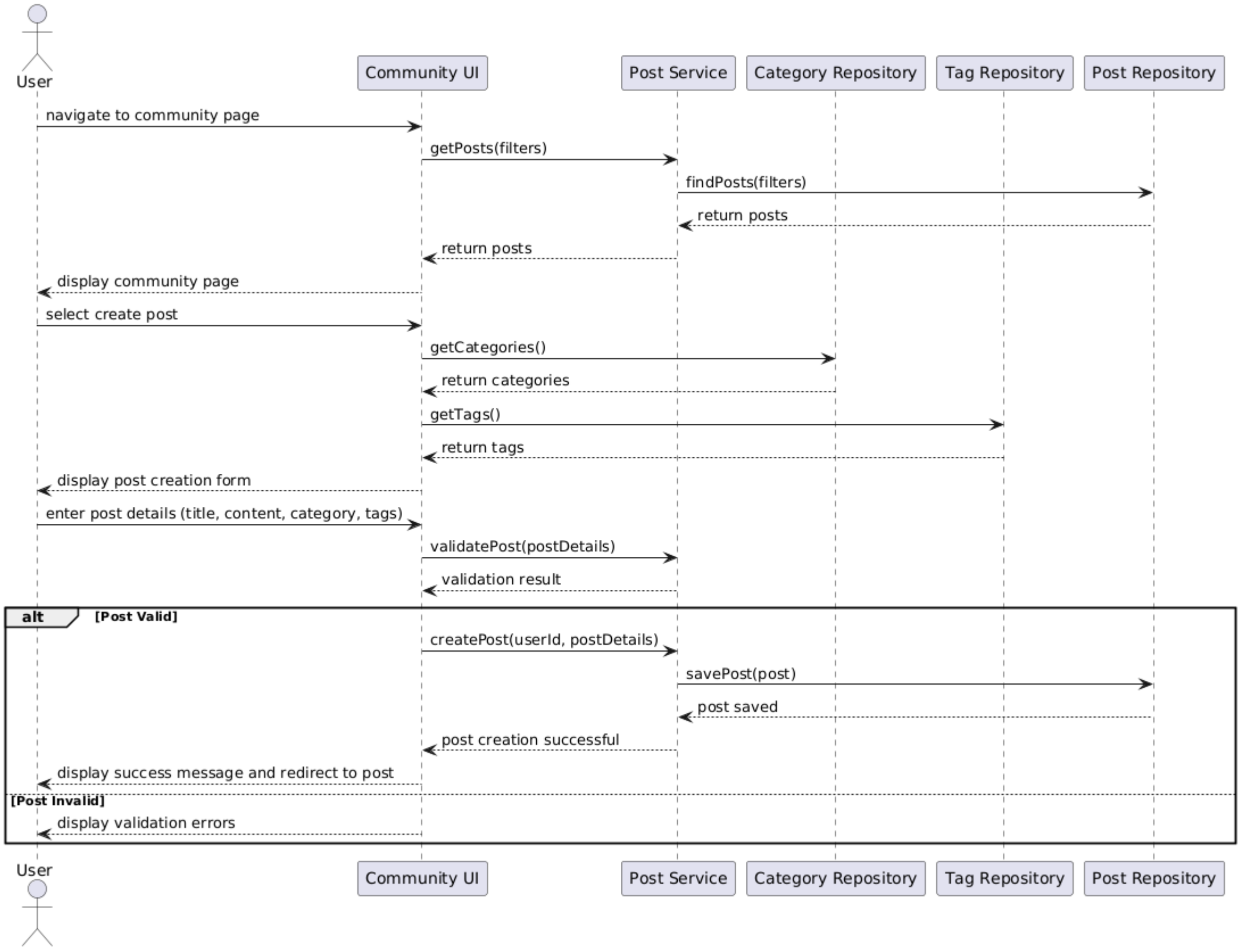
The Creating Diet Plan Sequence Diagram illustrates the interactions involved when a trainee creates a diet plan:


Trainee
Diet Plan UI
Diet Service
User Metrics Service
AI Recommendation Service
Diet Plan Repository
select create diet plan
prompt for metrics input
enter metrics (height, weight, etc.)
saveMetrics
return metrics
confirm metrics and preferences
createDietPlan(userId, metrics, preferences)
generateDietRecommendations(metrics, preferences)
return meal recommendations
createDietPlanWithMeals(recommendations)
saveDietPlan(dietPlan)
plan saved
return created plan
display diet plan
alt
[Trainee Accepts Plan]
accept plan
confirmPlan(planId)
updatePlanStatus(planId, "CONFIRMED")
status updated
confirmation successful
display confirmation
[Trainee Requests Changes]
request changes
regeneratePlan(planId, adjustedPreferences)
regenerateRecommendations(metrics, adjustedPreferences)
return new recommendations
updatePlan(planId, newPlan)
plan updated
return updated plan
display updated plan
Trainee
Diet Plan UI
Diet Service
User Metrics Service
AI Recommendation Service
Diet Plan Repository

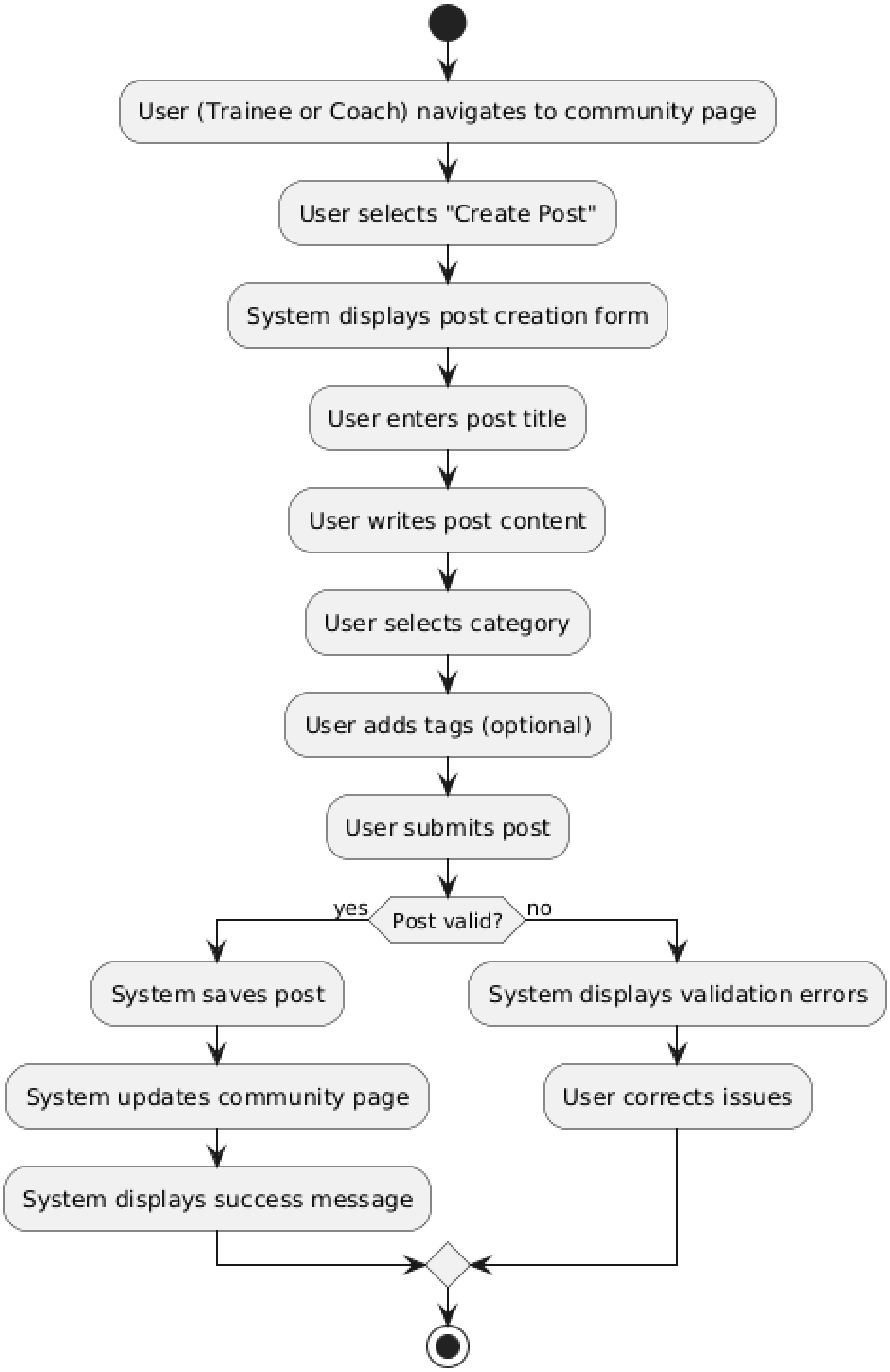
The Creating Diet Plan Activity Diagram outlines the steps a trainee takes to create a diet plan:


Trainee selects "Create Diet Plan"
System prompts for metrics
Trainee enters/updates metrics (height, weight, etc.)
System validates input
Input valid?
yes
no
System sends metrics to AI service
System displays error message
AI processes nutritional requirements
Trainee corrects input
AI generates meal recommendations
System creates diet plan with meals
System displays diet plan to trainee
Trainee reviews diet plan
Trainee accepts plan?
yes
no
System saves diet plan
Trainee adjusts preferences
System confirms plan creation
System regenerates plan

The Creating Community Post Sequence Diagram illustrates the interactions involved when a user creates a community post:



The Creating Community Post Activity Diagram outlines the steps a user takes to create a community post:


User (Trainee or Coach) navigates to community page
User selects "Create Post"
System displays post creation form
User enters post title
User writes post content
User selects category
User adds tags (optional)
User submits post
Post valid?
yes
no
System saves post
System updates community page
System displays success message
System displays validation errors
User corrects issues

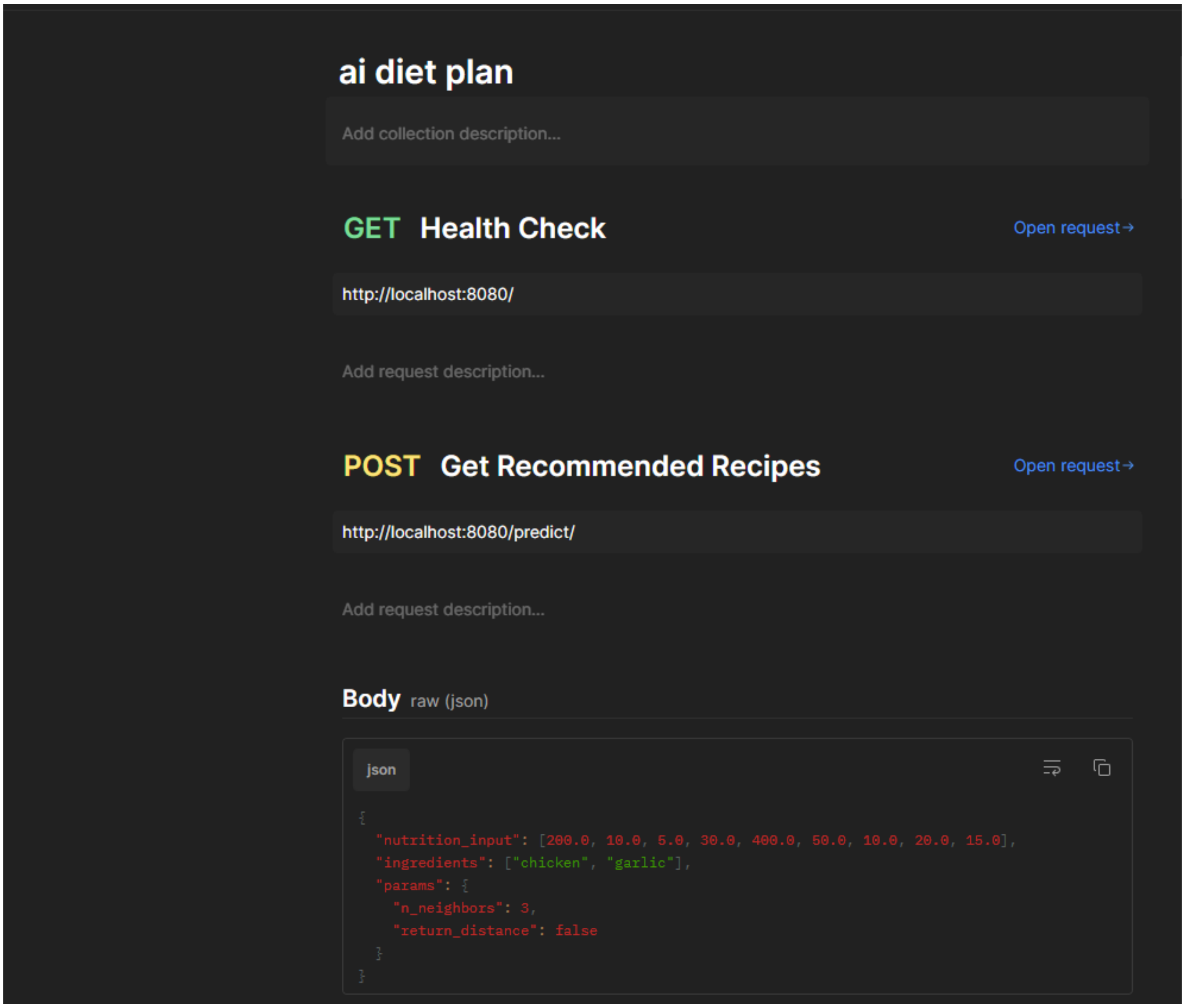
Chapter 4: Implementation

4.1 Implementation Details.

Technology Stack

- **Frontend:**
    - **Languages:** (e.g., HTML5, CSS3, JavaScript/TypeScript)
    - **Frameworks/Libraries:** (e.g., React, Bootstrap, Tailwind CSS)
    - **Build Tools:** (npm)
- **Backend:**
    - **Languages:** (e.g., Python, Java)
    - **Frameworks:** (FastAPI, Spring Boot)
    - **APIs/Protocols:** (RESTful APIs)
- **Database:**
    - **Type:** (e.g., Relational)
    - **Specific Database System:** (e.g., MySQL, PostgreSQL)
    - **ORM/ODM:** (Hibernate)
- **AI/Machine Learning Components:**
    - **Languages/Libraries:** (Python with TensorFlow, PyTorch, scikit-learn)

4.2 API Endpoints.



ai diet plan

Add collection description...

GET Health Check

http://localhost:8080/

Add request description...

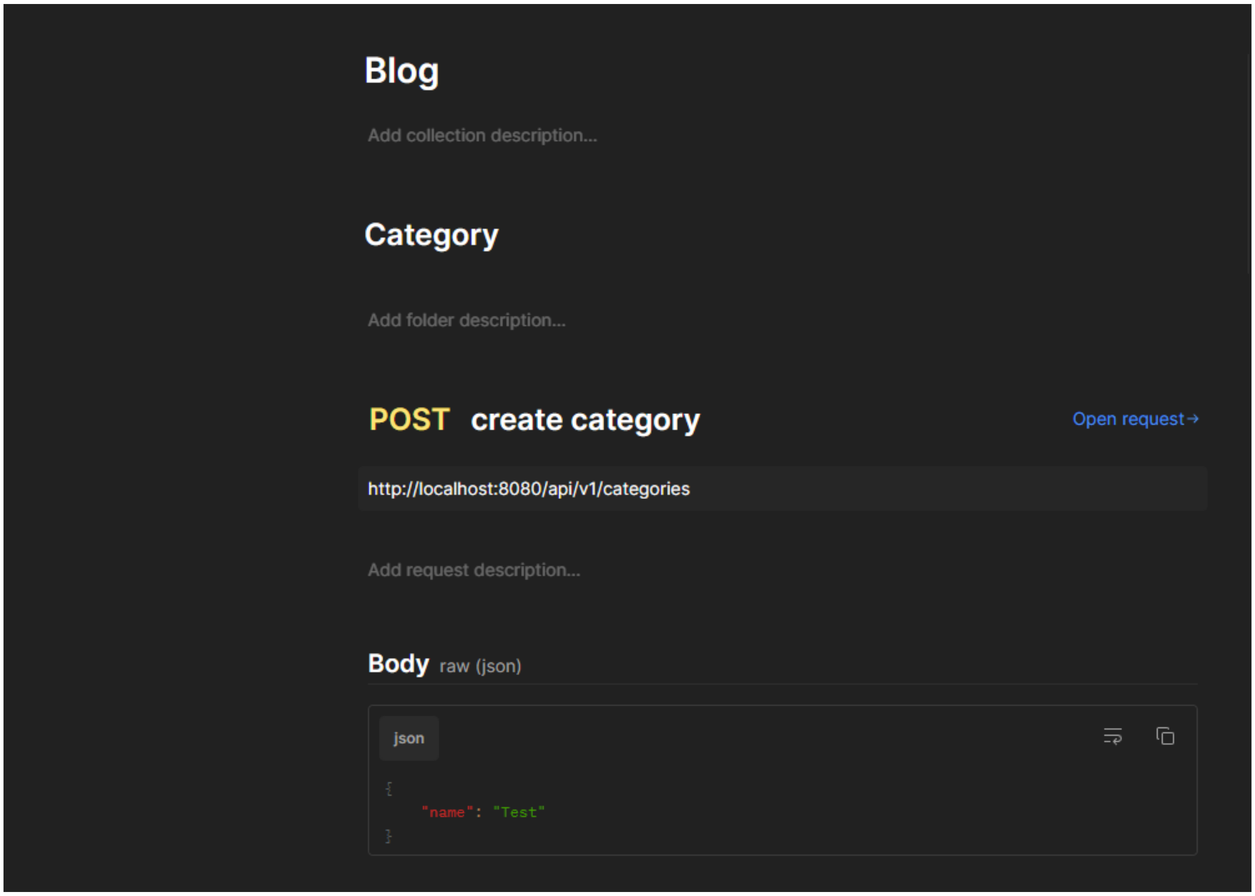
POST Get Recommended Recipes

http://localhost:8080/predict/

Add request description...

Body raw (json)

```json
{
  "nutrition_input": [200.0, 10.0, 5.0, 30.0, 400.0, 50.0, 10.0, 20.0, 15.0],
  "ingredients": ["chicken", "garlic"],
  "params": {
    "n_neighbors": 3,
    "return_distance": false
  }
}
```



Blog

Add collection description...

Category

Add folder description...

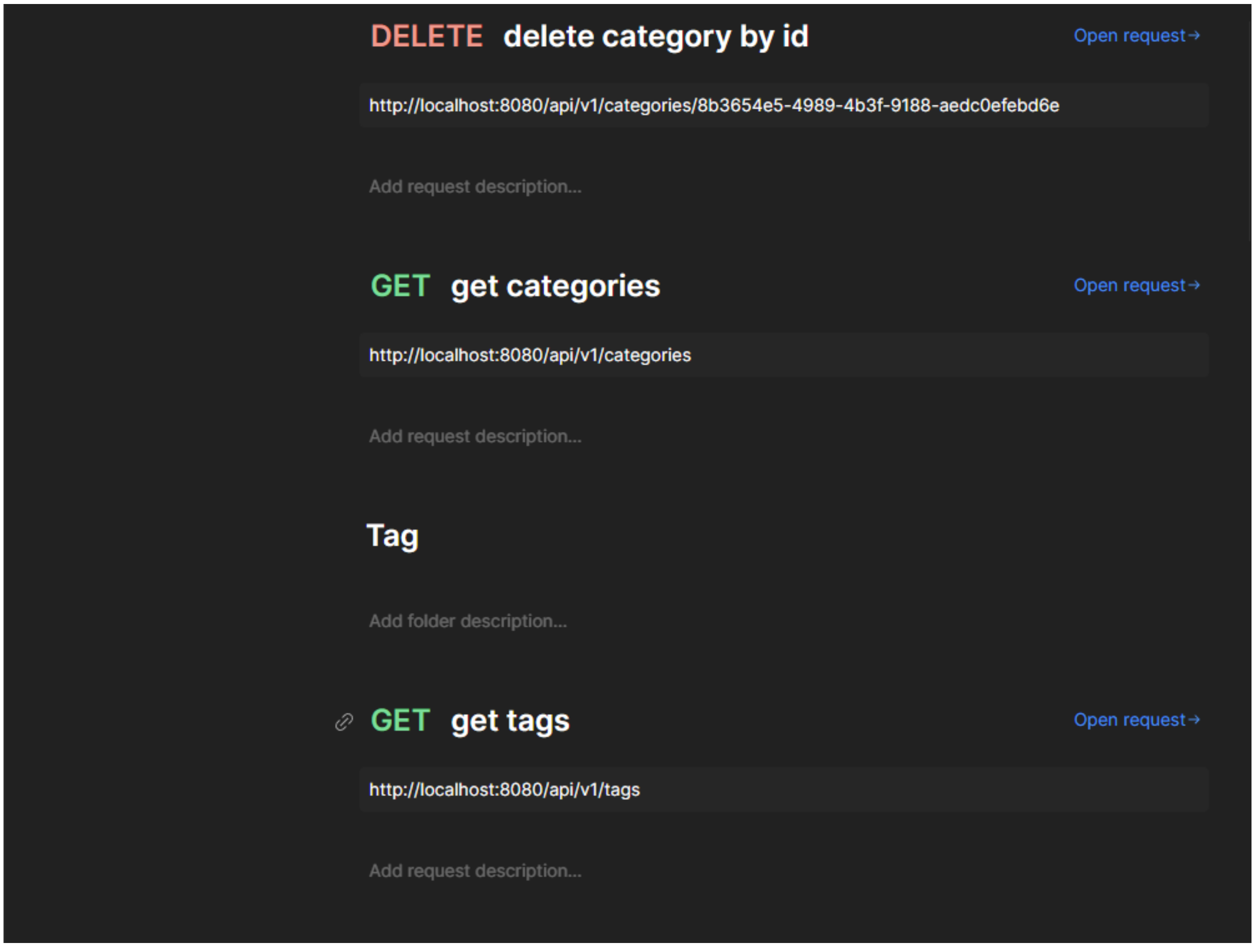
POST create category

http://localhost:8080/api/v1/categories

Add request description...

Body raw (json)

```
{
    "name": "Test"
}
```



DELETE delete category by id

http://localhost:8080/api/v1/categories/8b3654e5-4989-4b3f-9188-aedc0efebd6e

Add request description...

GET get categories

http://localhost:8080/api/v1/categories

Add request description...

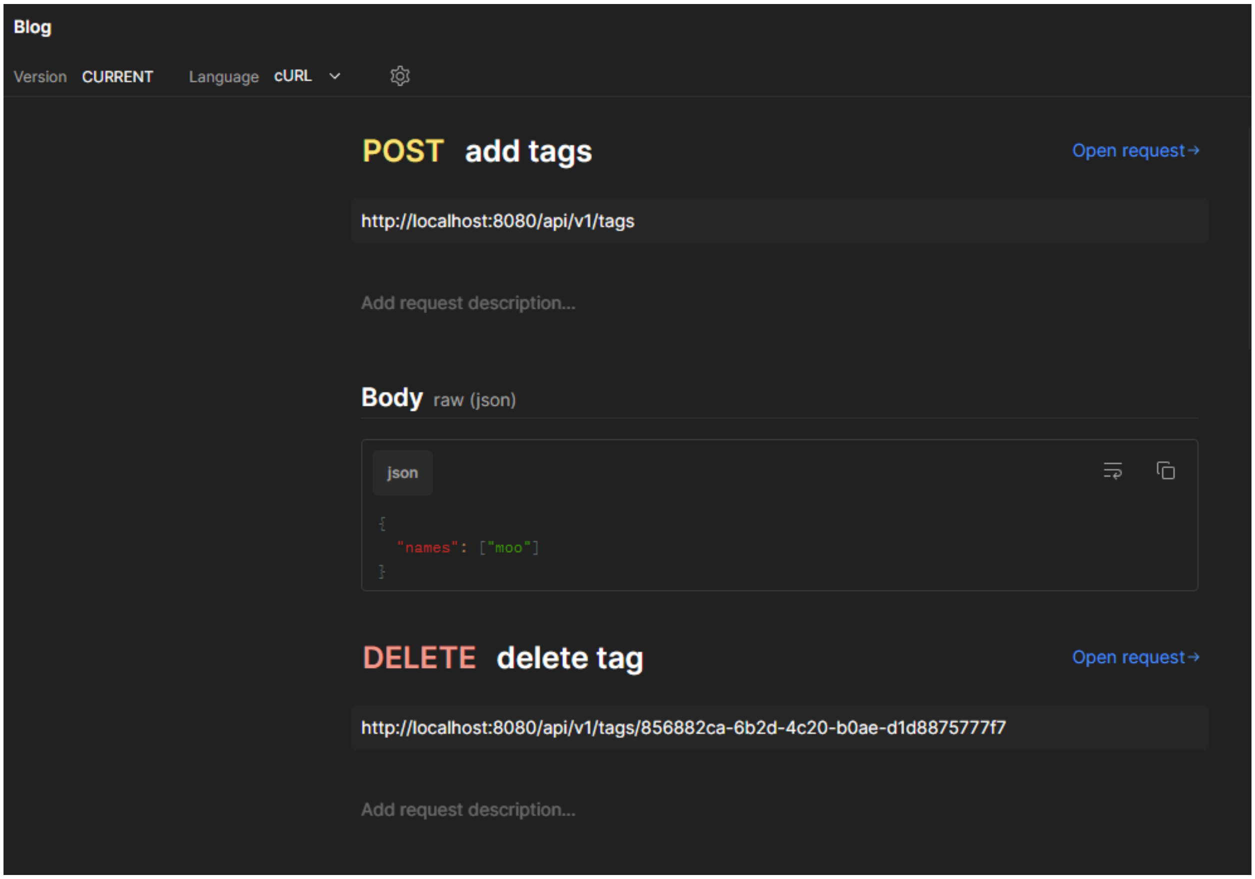
Tag

Add folder description...

GET get tags

http://localhost:8080/api/v1/tags

Add request description...



Blog

Version CURRENT   Language cURL

POST add tags

Open request→

http://localhost:8080/api/v1/tags

Add request description...

Body raw (json)

json

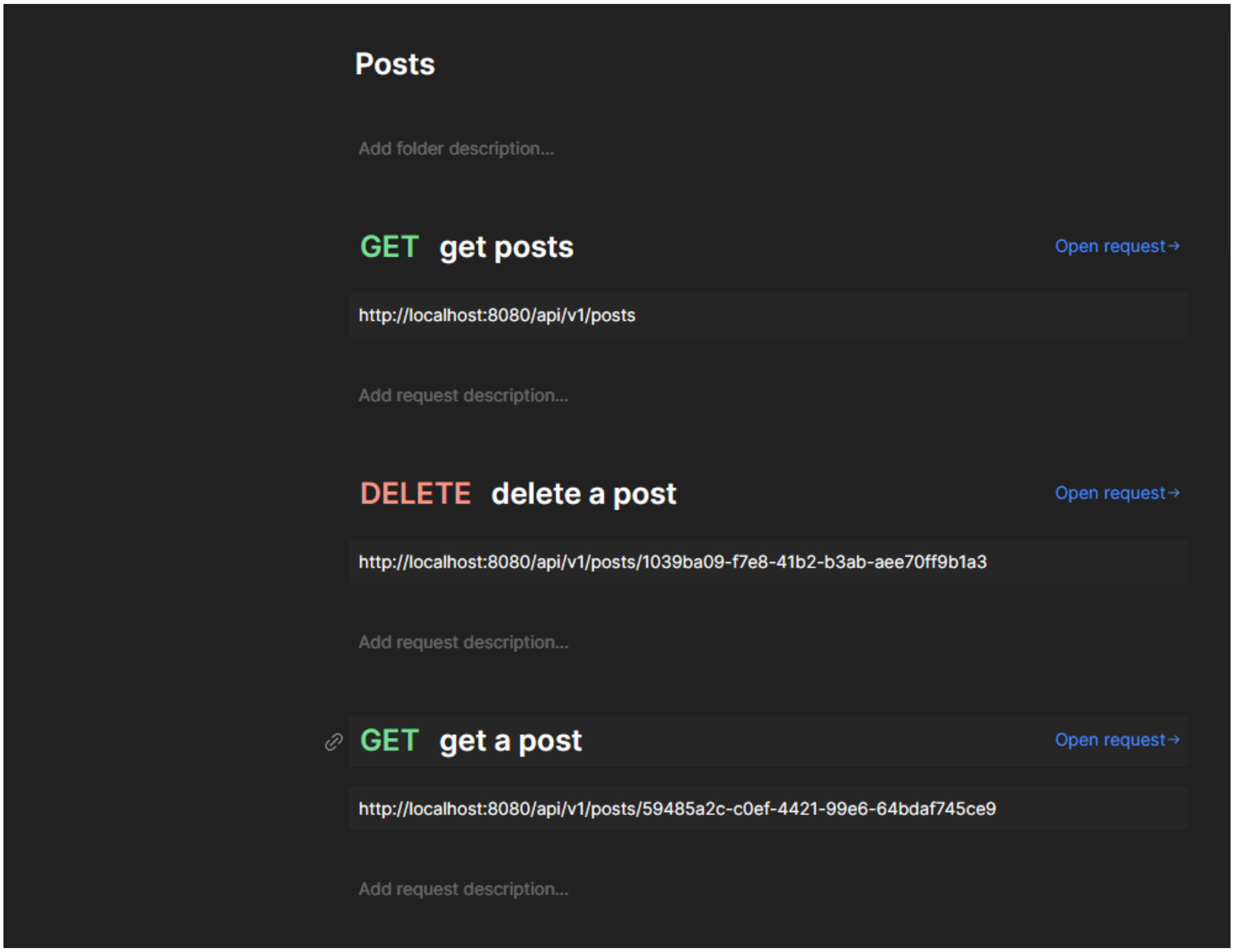
```
{
  "names": ["moo"]
}
```

DELETE delete tag

Open request→

http://localhost:8080/api/v1/tags/856882ca-6b2d-4c20-b0ae-d1d8875777f7

Add request description...



Posts

Add folder description...

GET get posts

Open request→

```
http://localhost:8080/api/v1/posts
```

Add request description...

DELETE delete a post

Open request→

```
http://localhost:8080/api/v1/posts/1039ba09-f7e8-41b2-b3ab-aee70ff9b1a3
```

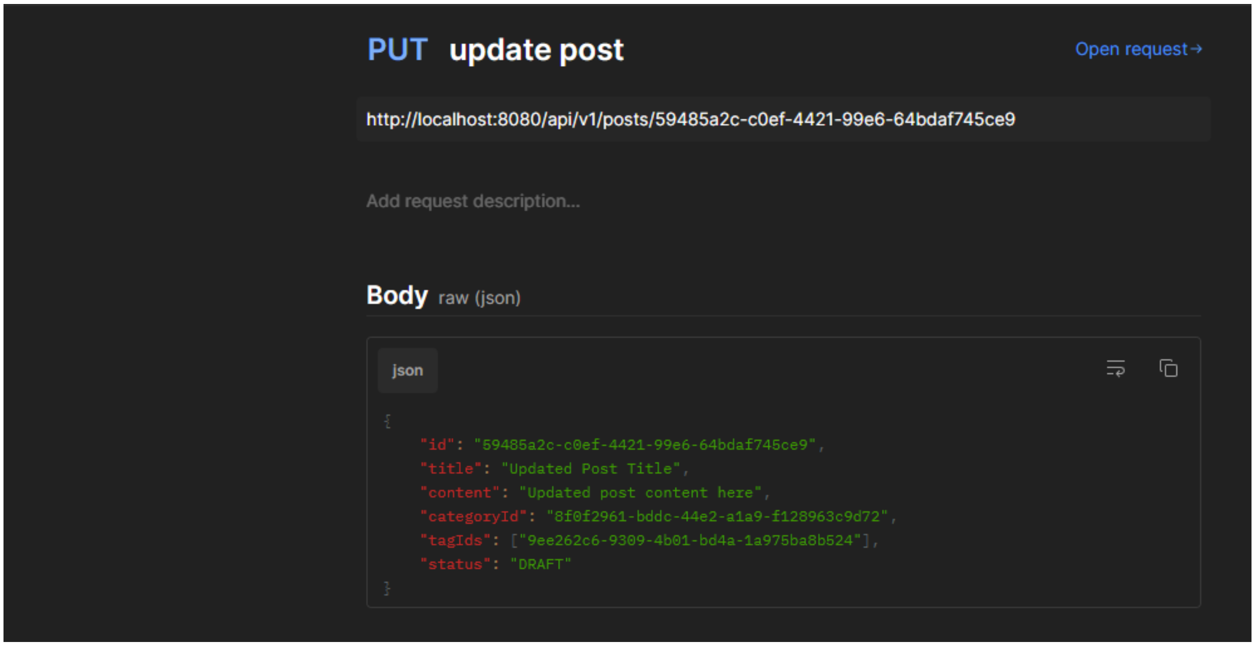
Add request description...

GET get a post

Open request→

```
http://localhost:8080/api/v1/posts/59485a2c-c0ef-4421-99e6-64bdaf745ce9
```

Add request description...



PUT update post

Open request→

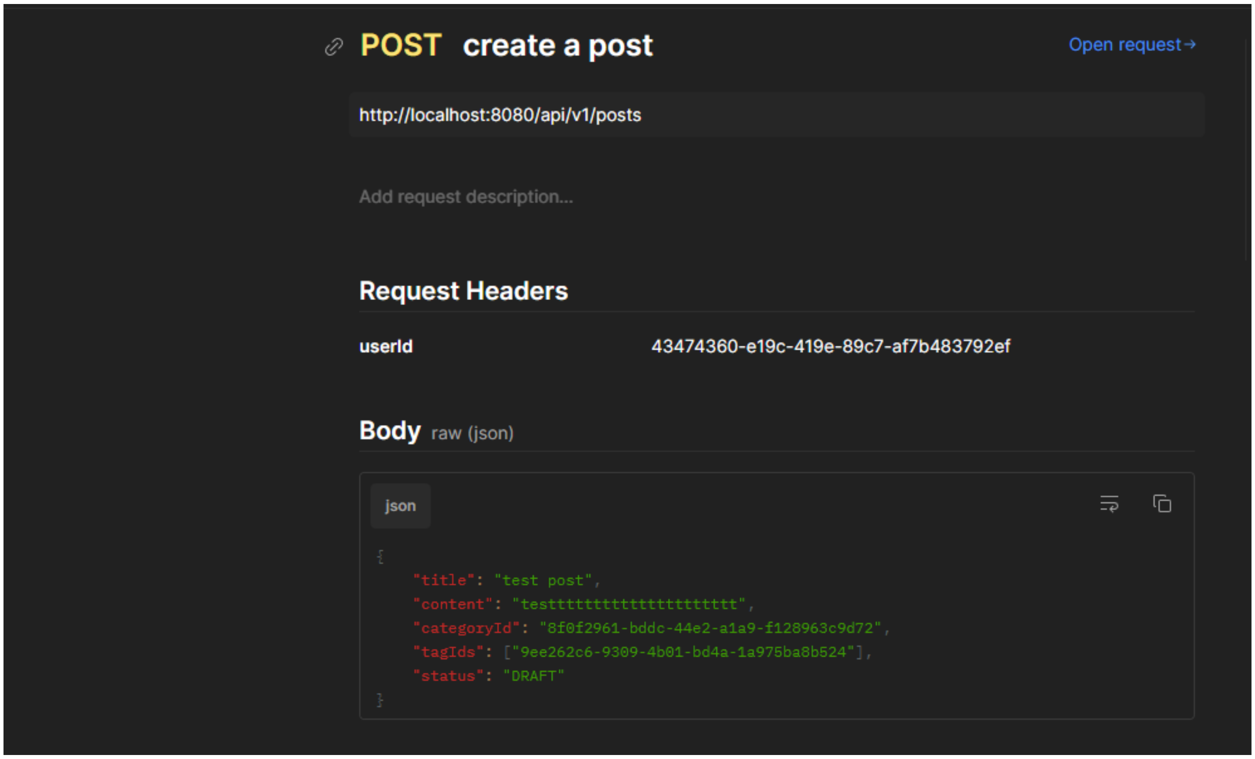
```
http://localhost:8080/api/v1/posts/59485a2c-c0ef-4421-99e6-64bdaf745ce9
```

Add request description...

Body raw (json)

json

```json
{
    "id": "59485a2c-c0ef-4421-99e6-64bdaf745ce9",
    "title": "Updated Post Title",
    "content": "Updated post content here",
    "categoryId": "8f0f2961-bddc-44e2-a1a9-f128963c9d72",
    "tagIds": ["9ee262c6-9309-4b01-bd4a-1a975ba8b524"],
    "status": "DRAFT"
}
```



POST create a post

http://localhost:8080/api/v1/posts

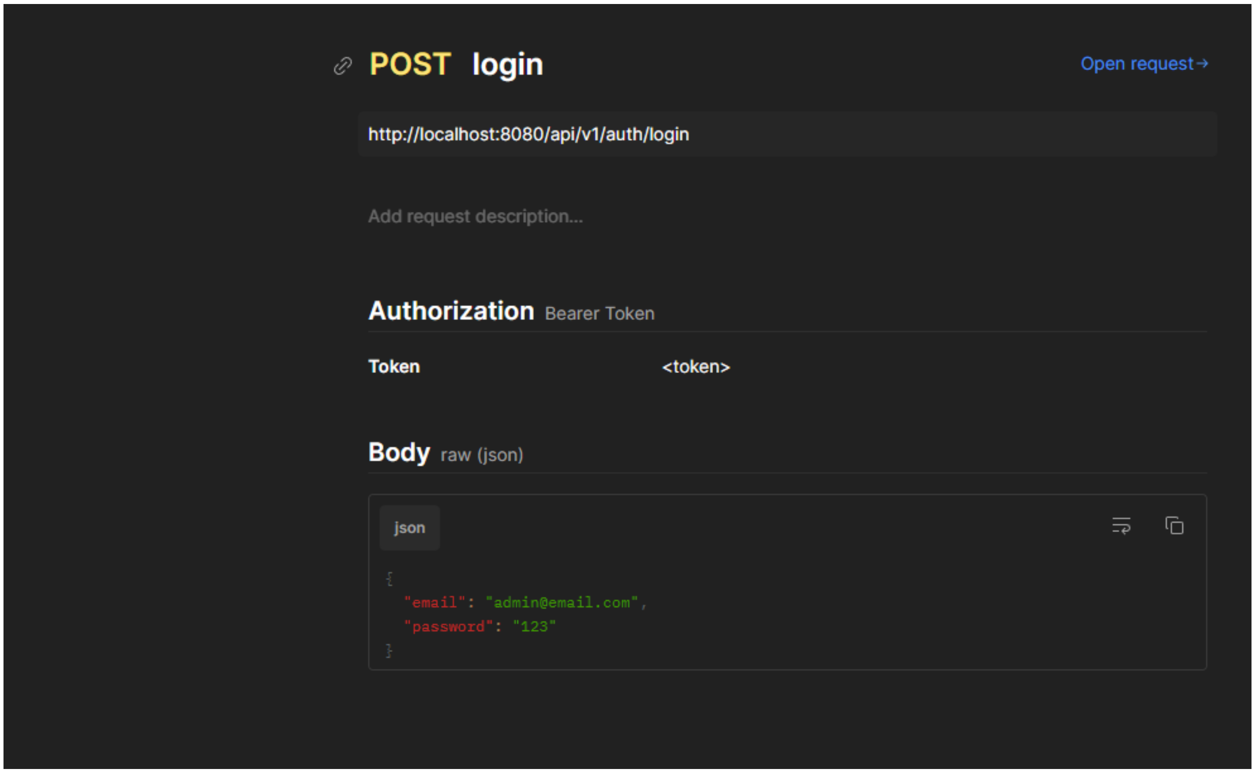
Add request description...

Request Headers

userId 43474360-e19c-419e-89c7-af7b483792ef

Body raw (json)

```json
{
    "title": "test post",
    "content": "testtttttttttttttttttttt",
    "categoryId": "8f0f2961-bddc-44e2-a1a9-f128963c9d72",
    "tagIds": ["9ee262c6-9309-4b01-bd4a-1a975ba8b524"],
    "status": "DRAFT"
}
```



POST login

http://localhost:8080/api/v1/auth/login

Add request description...

Authorization Bearer Token

Token <token>

Body raw (json)

```json
{
  "email": "admin@email.com",
  "password": "123"
}
```

Chapter 5: Conclusions, and Future Work

5.1 Summary & Conclusion

The Diet and Exercise Plans Website successfully delivers a comprehensive platform for personalized fitness and nutrition guidance. By integrating AI-powered diet recommendations, robust gym management features, and a vibrant community forum, the system addresses the diverse needs of Trainees and Coaches. The distributed architecture, leveraging Spring Boot, FastAPI, and various databases, ensures scalability, reliability, and maintainability. The project's adherence to Agile methodologies facilitated efficient development and responsiveness to requirements.

The food recommendation component, in particular, effectively leverages a content-based approach. By employing Standard Scaler for data normalization and Nearest Neighbors with cosine similarity, coupled with a scikit-learn Pipeline for workflow efficiency, the system provides a robust and interpretable recommendation engine based on nutritional profiles and ingredients.

Future enhancements could include advanced analytics for progress tracking, real-time chat functionalities between coaches and trainees, and integration with wearable devices to provide a more holistic health management solution. For the recommendation system, future work could explore hybrid approaches, incorporating collaborative filtering techniques (e.g., user ratings) to enhance serendipity and address the limitations of a purely content-based system. Advanced NLP techniques could also be applied to recipe descriptions for richer feature extraction, further improving recommendation quality. The foundation laid by this project provides a strong base for continuous improvement and expansion in the evolving landscape of digital health and fitness.

5.2 Future Work

1. Enhancements to Existing Features:

- **AI-driven Diet Plans:**
    - **More Granular Customization:** Allow users to specify dietary restrictions (e.g., vegan, gluten-free), allergies, or preferences (e.g., low-carb, high-protein) in more detail for AI generation.
    - **Meal Prep Integration:** Suggest recipes or meal prep ideas based on the generated diet plans.
    - **Dynamic Adjustments:** Implement AI that learns from user feedback and progress to dynamically adjust diet plans over time.
- **Exercise Plan Management:**
    - **Video Demonstrations:** Integrate short video clips for each exercise to guide trainees on proper form.
    - **Workout Tracking Automation:** Explore integration with wearable devices (e.g., smartwatches) for automatic tracking of workouts and biometric data.
    - **Progress Visualization:** Enhance charts and graphs for more intuitive and detailed progress tracking.
- **Community Features:**
    - **Real-time Chat/Messaging:** Implement direct messaging between users, or group chats for coaches and their trainees.
    - **Forum Categories:** Introduce structured categories or sub-forums for different topics (e.g., nutrition, specific workouts, success stories).
    - **Gamification:** Add elements like badges, leaderboards, or challenges to encourage engagement.

2. New Features/Modules:

- **Integration with Wearable Devices:** Allow users to sync data from fitness trackers (e.g., steps, calories burned, heart rate) to provide more accurate insights and personalized recommendations.

- **Advanced Analytics for Coaches:** Provide coaches with more in-depth analytics on their trainees' progress, adherence, and engagement.
- **Payment Gateway Integration:** If not already implemented, a secure system for trainees to subscribe to coaches.
- **Recipe Database:** A searchable database of healthy recipes, potentially linked to diet plans.
- **Educational Content:** A library of articles, videos, or webinars on nutrition, exercise science, and wellness.
- **Goal Setting & Reminders:** Features to help users set specific, measurable goals and receive reminders for workouts or meal times.

3. Technical Improvements & Scalability:

- **Performance Optimization:** Discuss strategies for improving load times, responsiveness, and overall system efficiency, especially as the user base grows.
- **Enhanced Security Measures:** Detail plans for advanced encryption, regular security audits, and protection against emerging threats.
- **Scalability Enhancements:** Outline how the system architecture can be further optimized to handle a larger number of concurrent users and data volume.

4. Research & Development:

- **More Sophisticated AI Models:** Explore integrating more advanced machine learning models for predictive analytics, injury prevention, or highly personalized coaching.
- **Personalized Behavioral Nudges:** Research how to use AI to provide personalized motivational messages or behavioral nudges to improve adherence.
- **User Experience (UX) Research:** Plan for user studies or A/B testing to continually refine the user interface and experience.