NutriLift

A Fitness and Nutrition Tracking Mobile Application with Community and Gamification Features

Academic Year	Module	Assessment Number	Assessment Type
2025/2026	Project and Professionalism	AD1	Artefact Design

Full Name: Luja Ratna Manandhar

Student Number: 2407087

Course: BSc. (Hons) Computer Science

University Email: L.R.Manandhar@wlv.ac.uk

Supervisor: Johan Tandukar

Reader: Yogesh Bikram Shah

Date of Submission: December 26, 2025

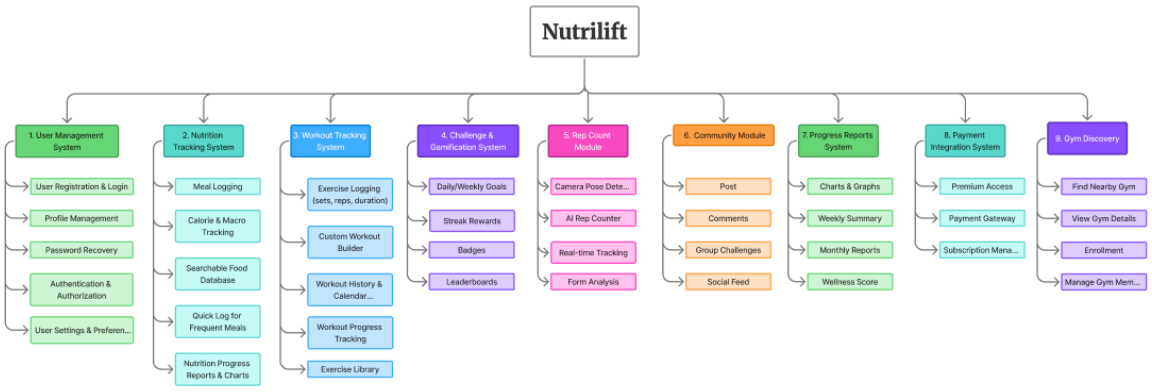
Introduction:

NutriLift is a mobile application designed to help people maintain a healthier lifestyle by combining fitness tracking, nutrition logging, gamification and a small online community in a single platform. This report presents the design artefacts produced during the current phase of development, including sprint planning, product backlog, data models and UML diagrams for each subsystem of the application. Together, these artefacts show how the initial proposal has been translated into a structured, testable system that is ready to move into implementation and formal evaluation.

Table of Contents

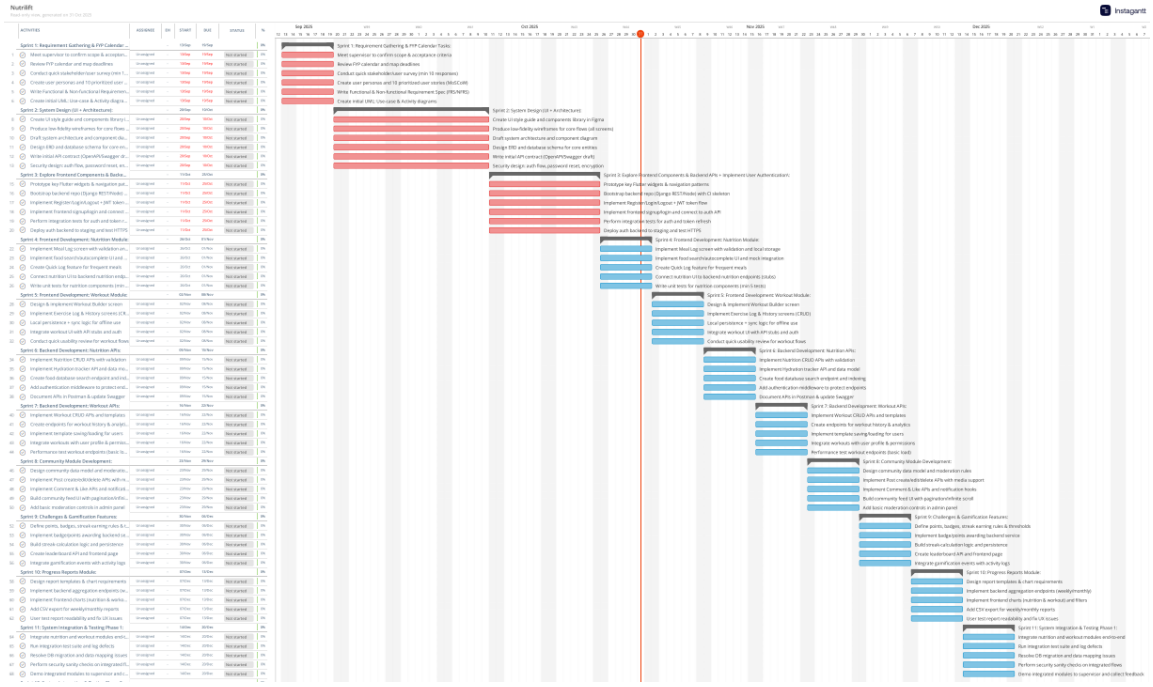
Table of Figures:

1. FDD Diagram:



FDD 1

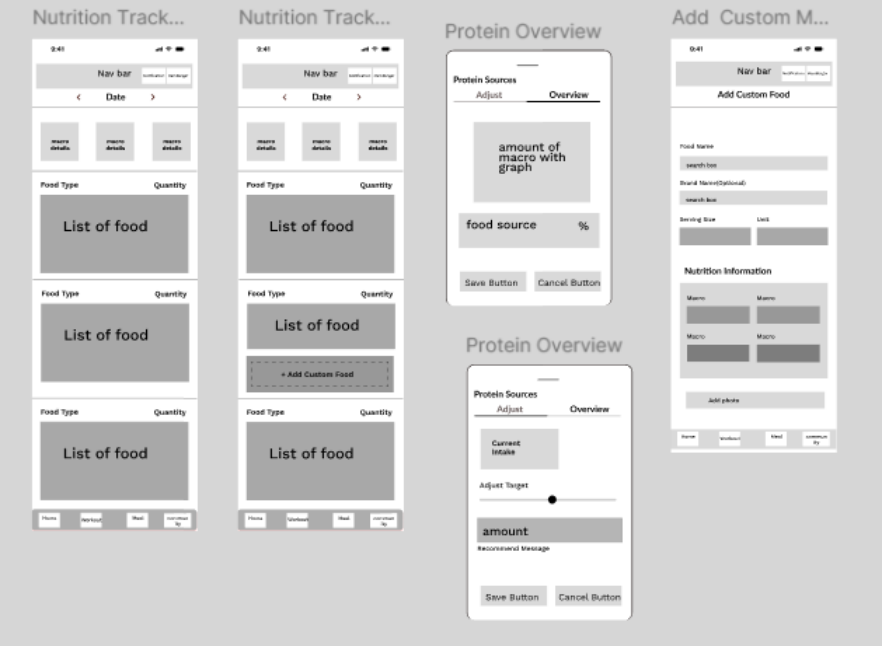
2. Planning:



Planning 1

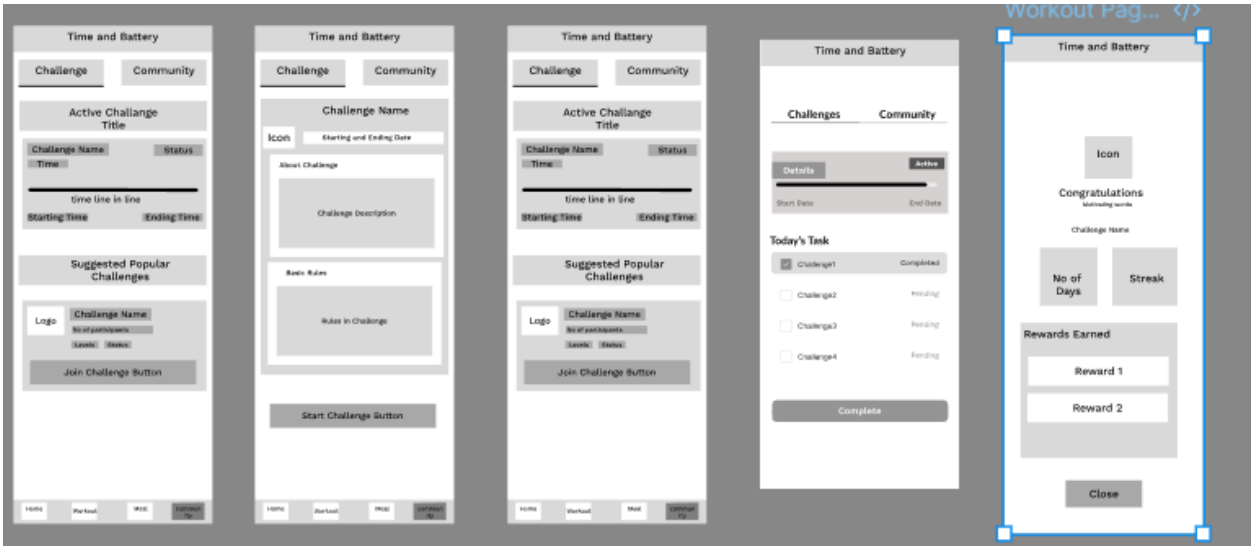
3. Wire Frame:

Wireframe 1

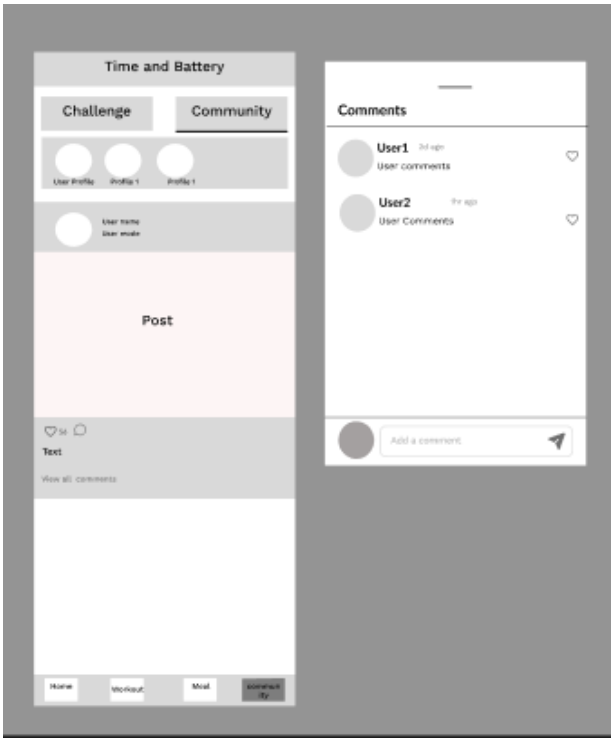


Wireframe 2

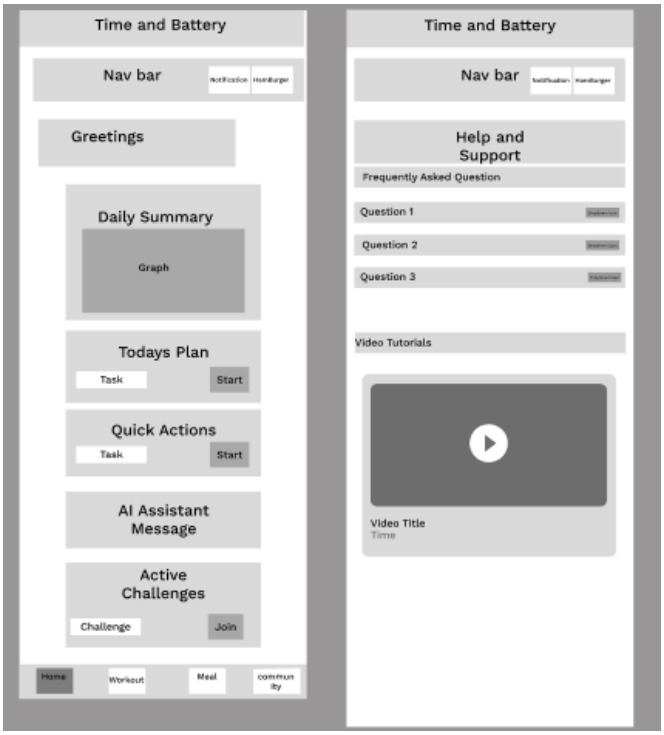
Wireframe 3



Wireframe 4



Wireframe 5



Wireframe 6

4. User Management:

This subsystem looks after user accounts, including registration, login, logout and password reset. It also stores profile details and preferences so other parts of the system can tailor calculations and behaviour to each user.

4.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
UM-F-1.0	As a user, I want to register with email and password so that I can create an account.	User Management	Registration works with valid email, unique records created, invalid inputs rejected.
UM-F-2.0	As a user, I want to log in and log out securely.	User Management	Valid credentials open session, invalid ones denied, logout ends session safely.
UM-F-3.0	As a user, I want to reset my password via email.	User Management	Token email sent, reset link updates password successfully.
UM-F-4.0	As a user, I want to edit my profile (age, weight, goal).	User Management	Saved changes reappear correctly after reopening profile.
UM-F-5.0	As a user, I want to set dietary preferences.	User Management	Preferences saved and available for food filtering.
UM-NF-1.1	Registration data must be transmitted securely.	User Management	All credentials transferred via HTTPS only.
UM-UR-1.1	Password fields should hide/unhide easily on mobiles.	User Management	Toggle icon works intuitively on touch devices.

4.2. Data Dictionary:

Entity: USER

Attribute Name	Data Type	Description
id	UUID	Unique identifier for each user.
email	String	Login email address.
password_hash	String	Hashed password.
name	String	Full name.
age	Integer	Age in years.
weight	Float	Current weight.
height	Float	Height.
gender	String	Gender value.
fitness_goal	String	Main fitness goal.
dietary_preferences	JSON	Dietary preferences/allergies.
is_premium	Boolean	Whether user has active premium.
subscription_expires_at	Datetime	Premium expiry timestamp.
is_active	Boolean	Account active or deactivated.
created_at	Datetime	Record creation time.

Attribute Name	Data Type	Description
updated_at	Datetime	Last update time.

Entity: USER_SESSION

Attribute Name	Data Type	Description
id	UUID	Session identifier.
user_id	UUID	References USER(id).
jwt_token	String	JSON Web Token used for authentication.
expires_at	Datetime	Token expiry datetime.
device_info	String	Device or platform information.
ip_address	String	IP address used.
created_at	Datetime	Session creation time.

Entity: PASSWORD_RESET

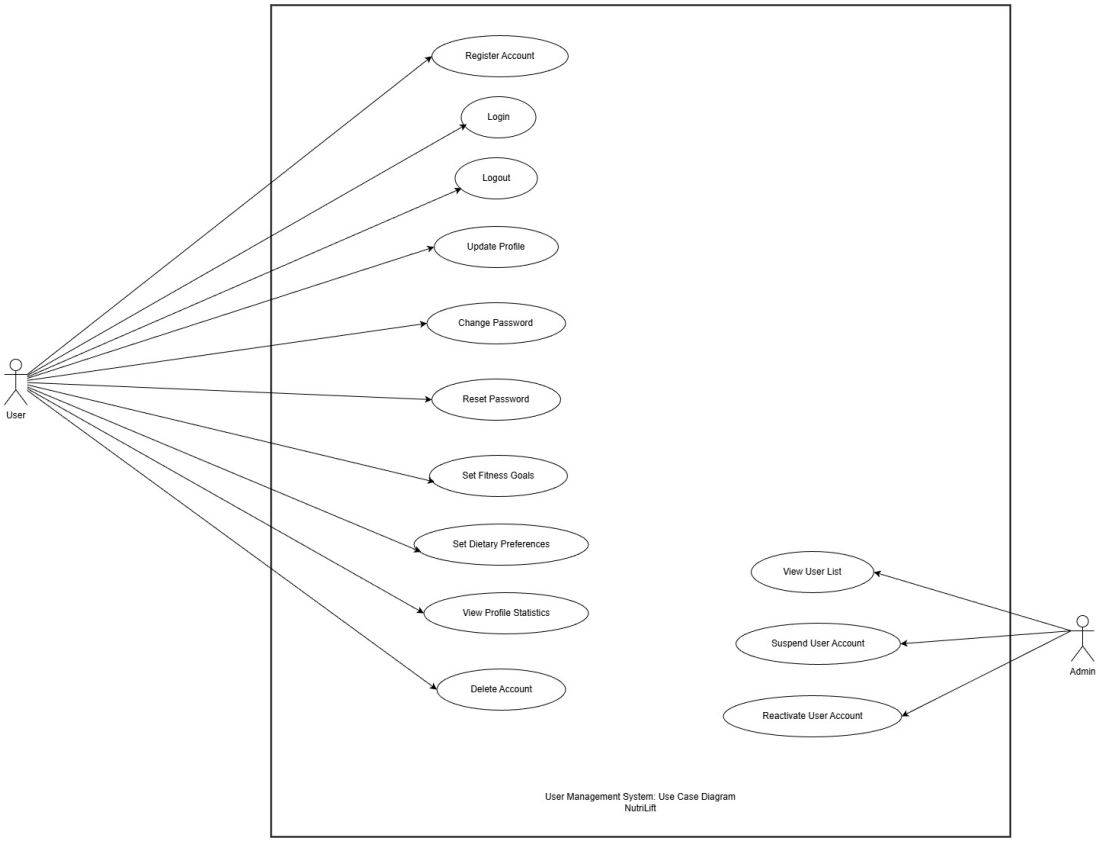
Attribute Name	Data Type	Description
id	UUID	Password reset request id.
user_id	UUID	References USER(id).
reset_token	String	Secure reset token.

Attribute Name	Data Type	Description
expires_at	Datetime	Token expiry time.
is_used	Boolean	Whether token has been used.
created_at	Datetime	Request creation time.

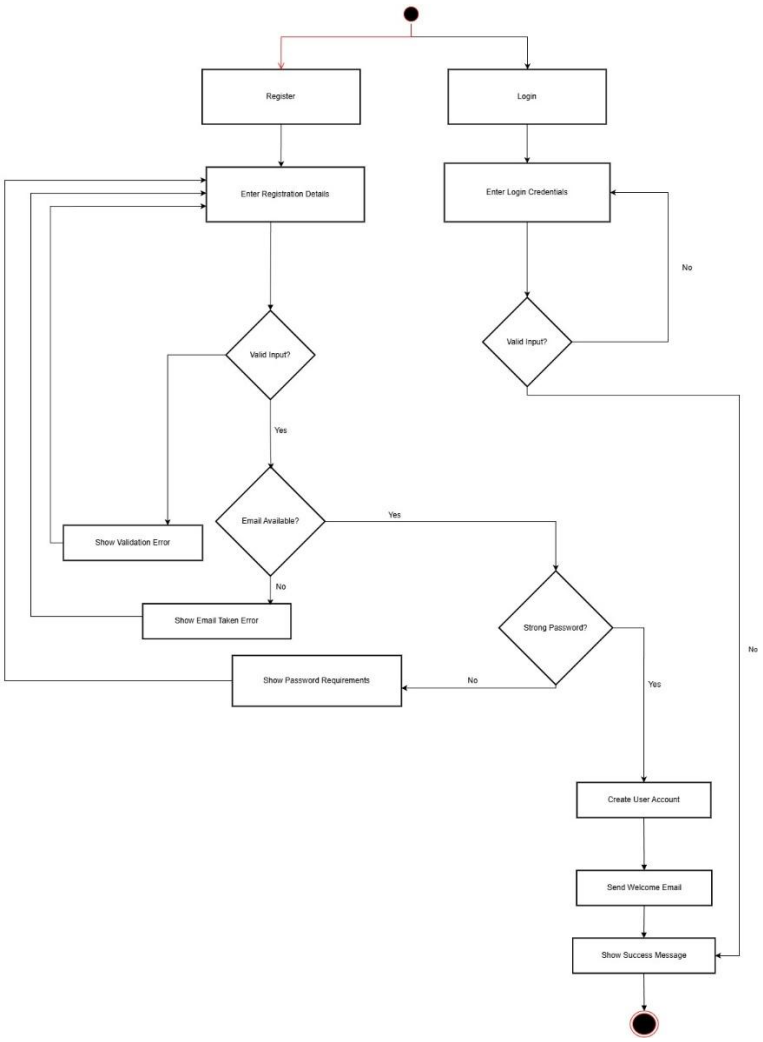
Entity; USER_PREFERENCES

Attribute Name	Data Type	Description
id	UUID	Preferences id.
user_id	UUID	References USER(id).
notification_settings	JSON	Notification configuration settings.
language	String	Preferred language code.
timezone	String	Time zone of the user.
privacy_settings	JSON	Privacy options.
updated_at	Datetime	Last update time.

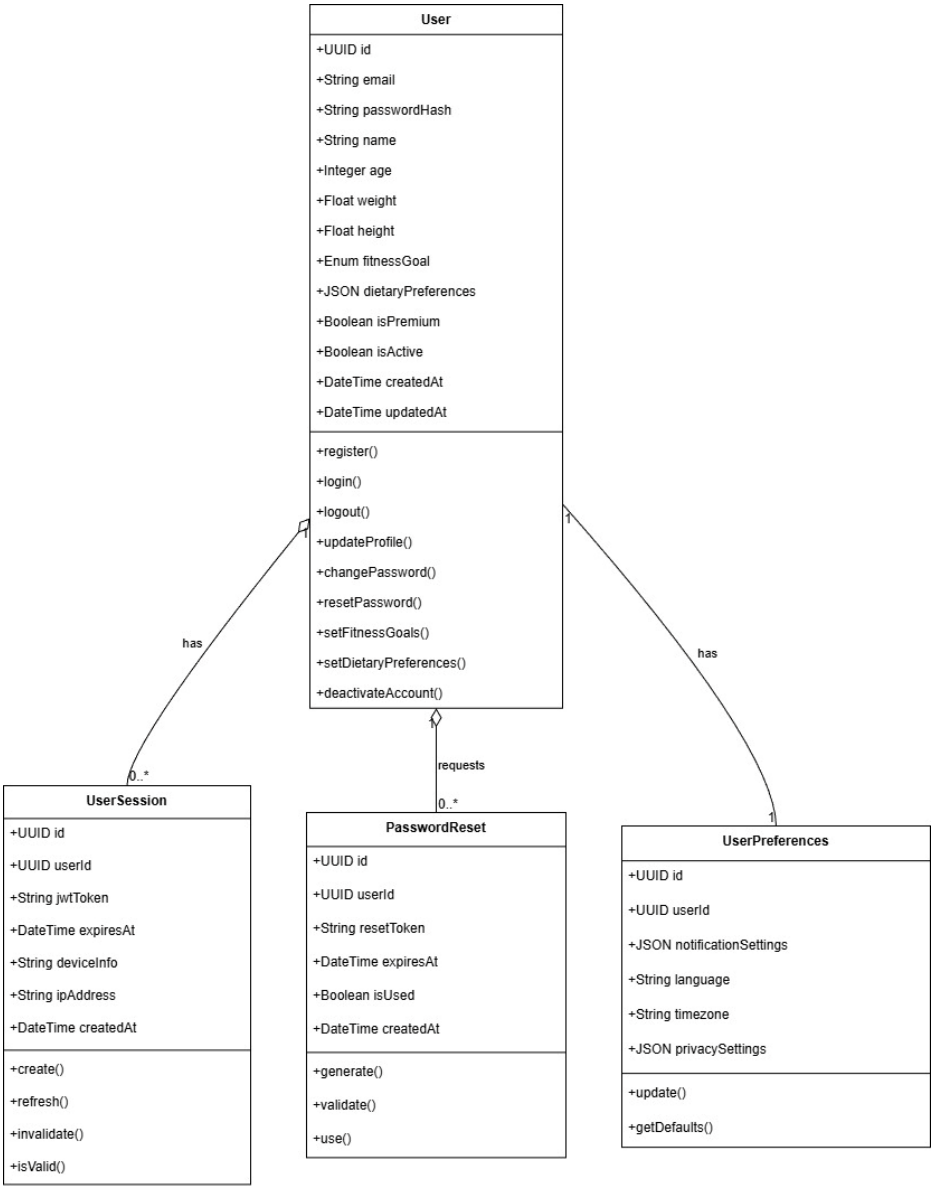
4.3. Diagrams:



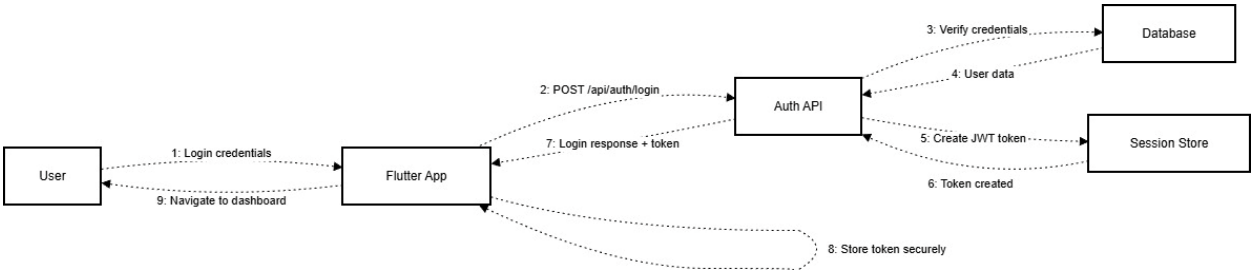
User Management 1: Use Case Diagram



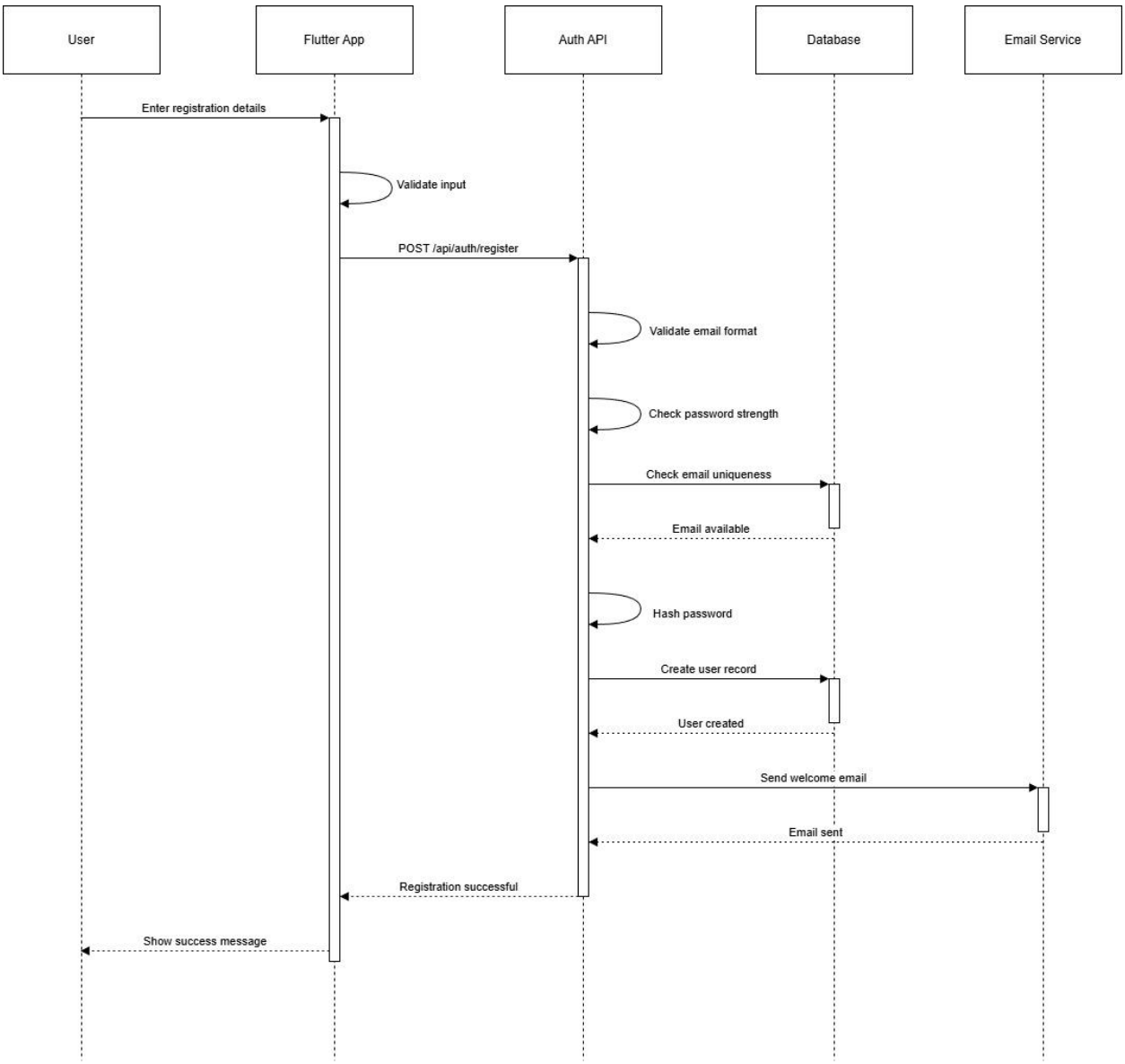
User Management 2: Activity Diagram



User Management 3: Class Diagram



User Management 4: Collaboration Diagram



User Management 5: Sequence Diagram

5. Nutrition Tracking

This subsystem records what the user eats and drinks in a day and calculates calories and macros. It helps the user stay on target by showing daily totals, goals and frequently used meals for quick logging.

5.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
NT-F-1.0	As a user, I want to log meals with calories and macros.	Nutrition Tracking	Meal saved, totals update instantly.
NT-F-2.0	As a user, I want to search foods from a database.	Nutrition Tracking	Search returns matches; custom foods allowed.
NT-F-3.0	As a user, I want to add custom food entries.	Nutrition Tracking	Custom item created and reusable in logs.
NT-F-4.0	As a user, I want to log water intake.	Nutrition Tracking	Water entry updates total and progress bar.
NT-F-5.0	As a user, I want to set daily nutrition goals.	Nutrition Tracking	Targets saved and shown beside daily totals.
NT-F-6.0	As a user, I want to view a daily nutrition summary.	Nutrition Tracking	Summary card displays calorie and macro totals.
NT-F-7.0	As a user, I want quick access to frequent meals.	Nutrition Tracking	Recent meals list appears for faster logging.
NT-F-8.0	As a user, I want to edit or delete logged meals.	Nutrition Tracking	Edits recalculate totals; deletes remove entries.

5.2. Data Dictionary:

Entity: FOOD_ITEM

Attribute Name	Data Type	Description
id	UUID	Food item id.
name	String	Food name.
barcode	String	Barcode string.
calories_per_100g	Float	Energy per 100g of food.
protein_per_100g	Float	Protein per 100g.
carbs_per_100g	Float	Carbohydrates per 100g.
fats_per_100g	Float	Fats per 100g.
fiber_per_100g	Float	Fiber per 100g.
sugar_per_100g	Float	Sugar per 100g.
is_custom	Boolean	Indicates if created by user or from global DB.
created_by	UUID	References USER(id) for custom foods.
created_at	Datetime	Food creation time.

Entity: INTAKE_LOG

Attribute Name	Data Type	Description
id	UUID	Intake log id.
user_id	UUID	References USER(id).
entry_type	String	Type of entry: meal, snack, drink, water.
food_item_id	UUID	References FOOD_ITEM(id) if a defined food is selected.
description	String	Free-text description of intake.
quantity	Float	Amount consumed.
unit	String	Unit such as g, ml, piece.
logged_at	Datetime	When intake was logged.
calories	Float	Calculated calories for this entry.
protein	Float	Calculated protein.
carbs	Float	Calculated carbs.
fats	Float	Calculated fats.
created_at	Datetime	Record creation time.

Entity: HYDRATION_LOG8

Attribute Name	Data Type	Description
id	UUID	Hydration entry id.
user_id	UUID	References USER(id).
amount	Float	Volume of water consumed.
unit	String	Unit such as ml or L.
logged_at	Datetime	Time of hydration entry.

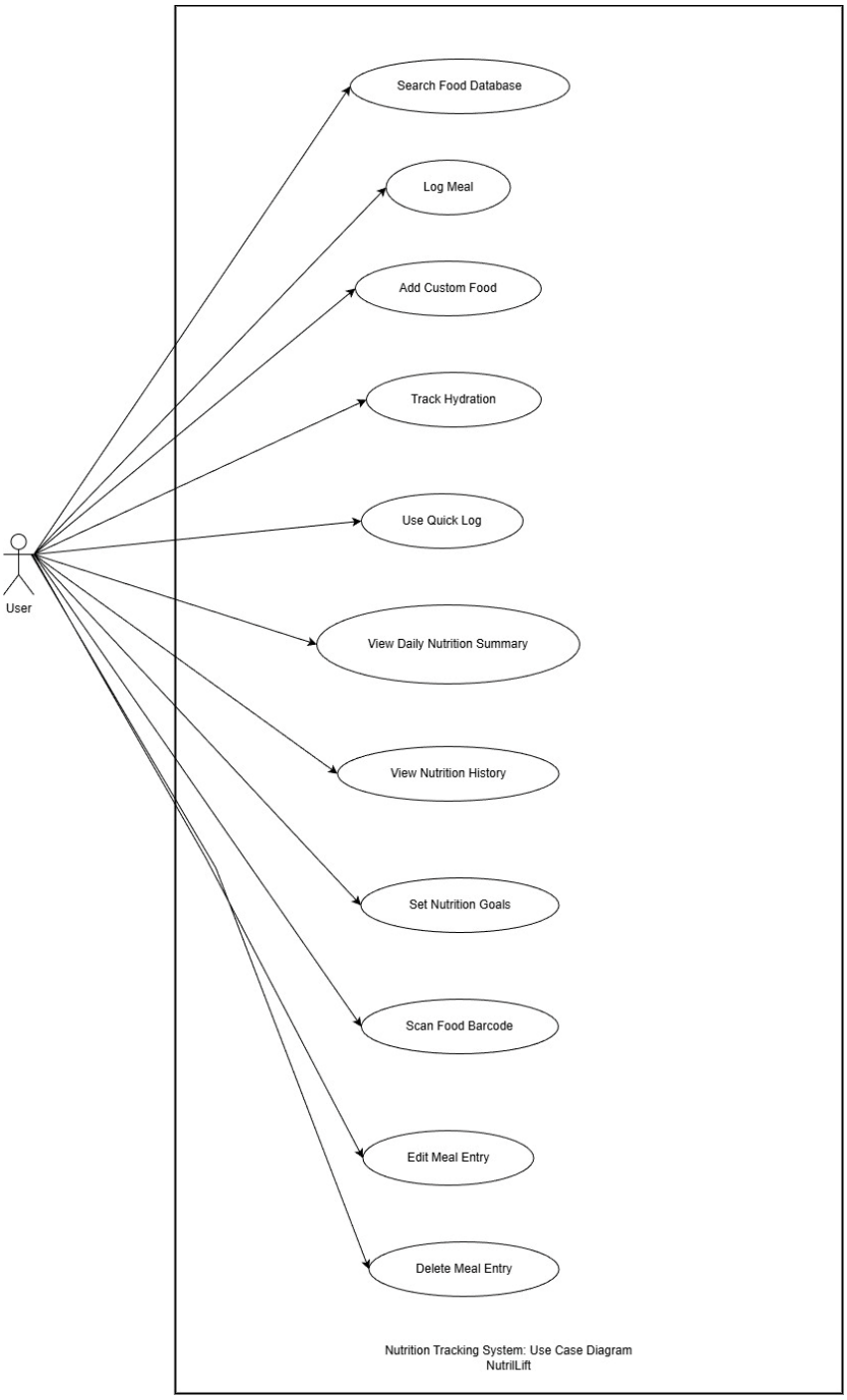
Entity: NUTRITION_GOALS

Attribute Name	Data Type	Description
id	UUID	Nutrition goals id.
user_id	UUID	References USER(id).
daily_calories	Float	Daily target calories.
daily_protein	Float	Daily target protein.
daily_carbs	Float	Daily target carbs.
daily_fats	Float	Daily target fats.
daily_water	Float	Daily target water intake.
updated_at	Datetime	Last update time.

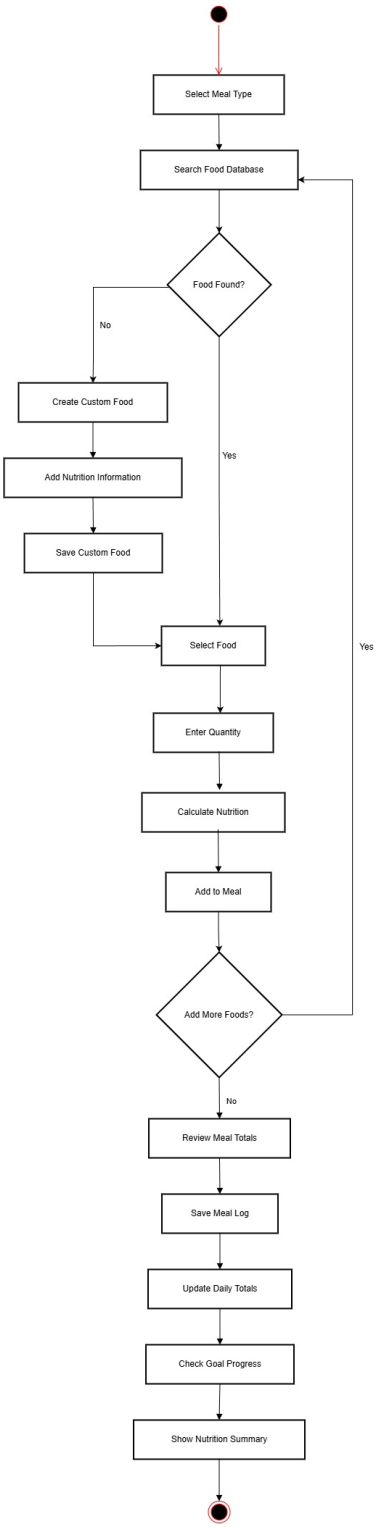
Entity: QUICK_LOG

Attribute Name	Data Type	Description
id	UUID	Quick log id.
user_id	UUID	References USER(id).
frequent_meals	JSON	Cached frequent meals and templates.
updated_at	Datetime	Last update time.

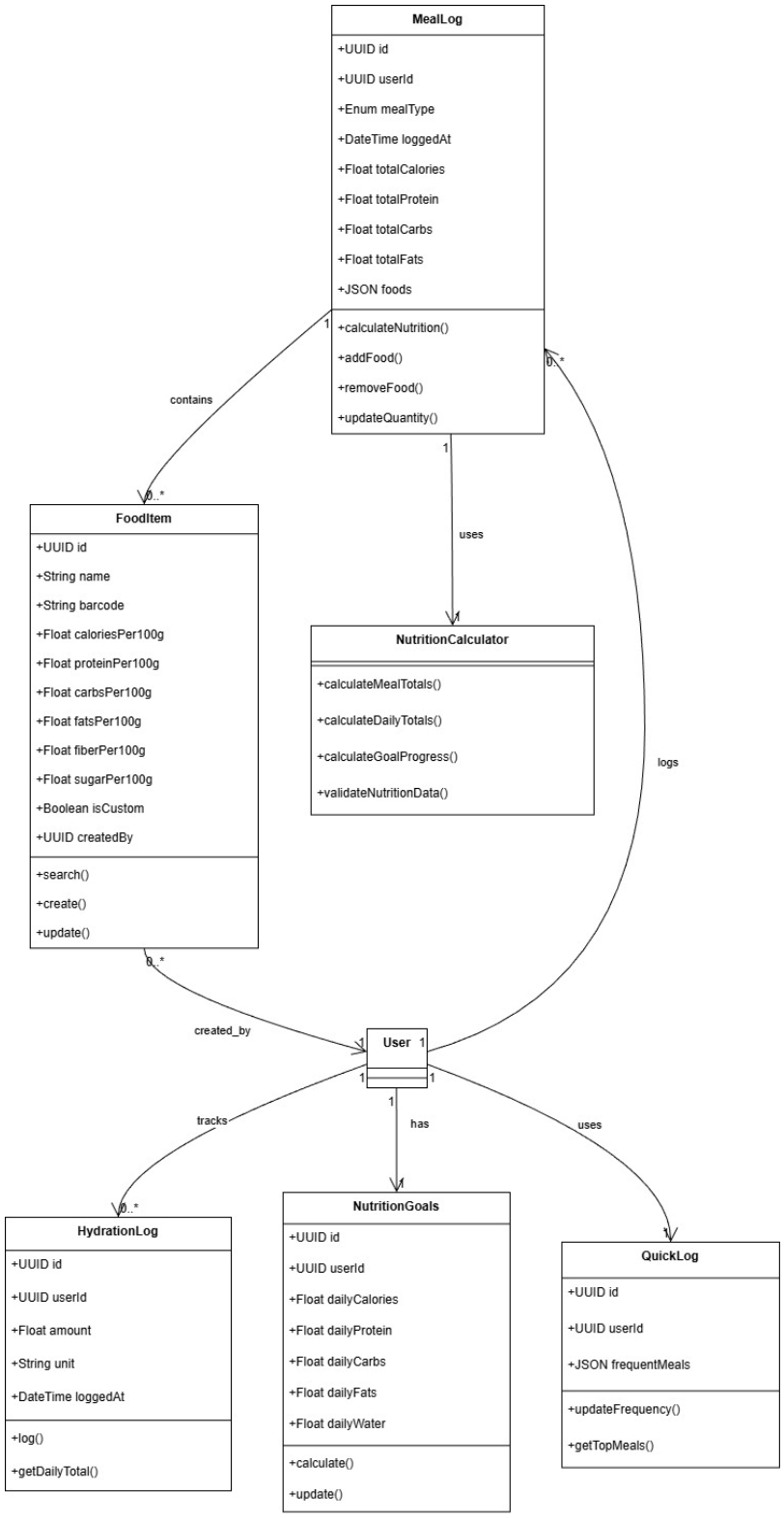
5.3. Diagrams:



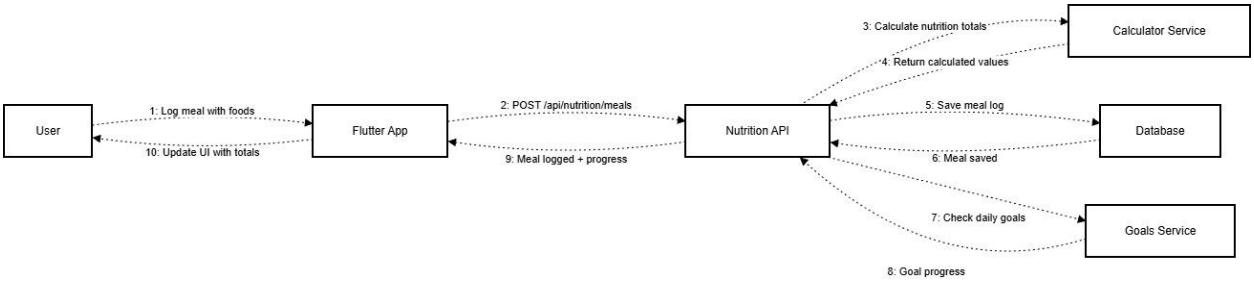
Nutrition Tracking 1: Use Case Diagram



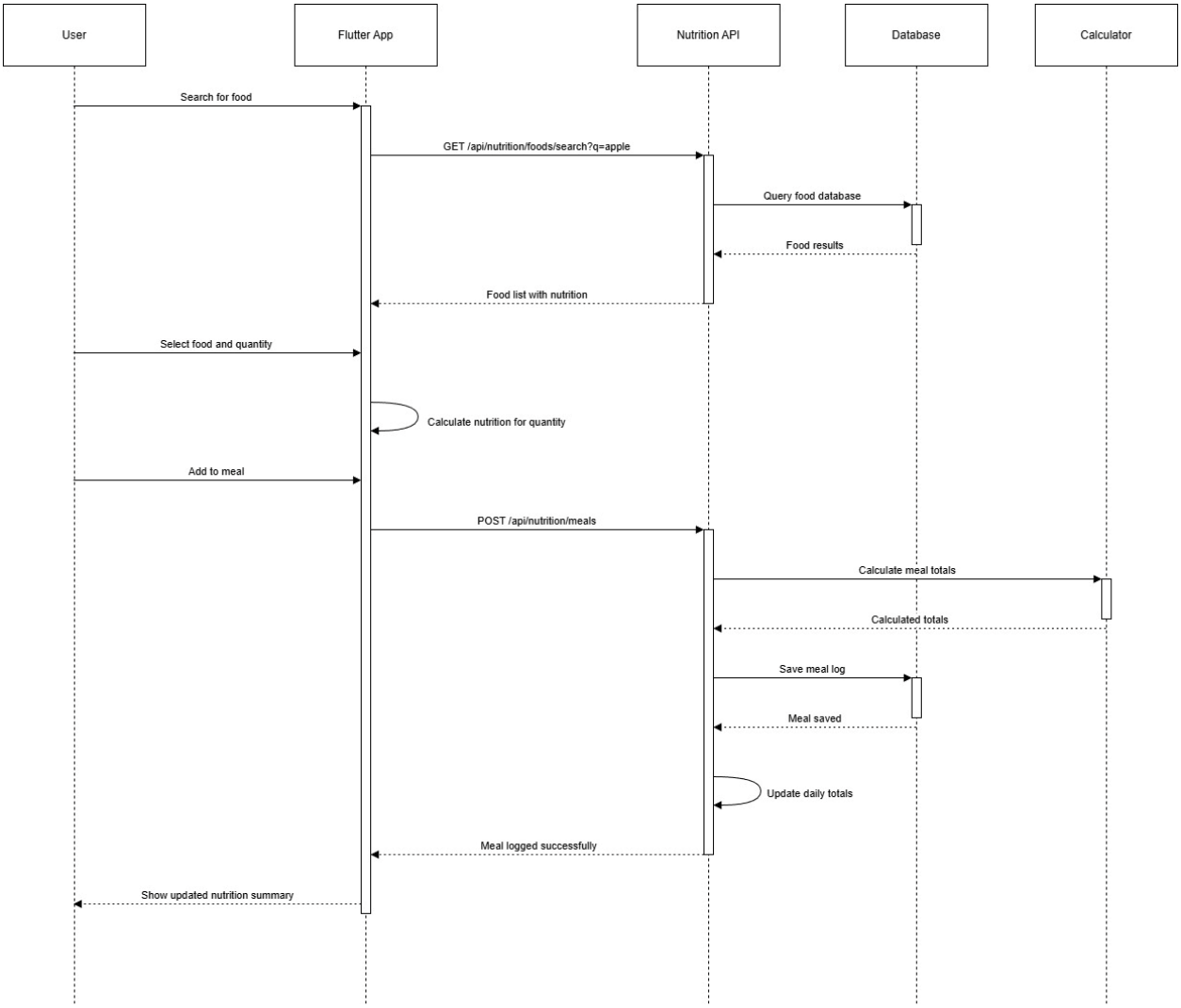
Nutrition Tracking 2: Activity Diagram



Nutrition Tracking 3: Class Diagram



Nutrition Tracking 4: Collaboration Diagram



Nutrition Tracking 5: Sequence Diagram

6. Workout Tracking

This subsystem captures gym and home workouts with exercises, sets, reps and weights. It keeps a history of past sessions and supports templates so users can repeat routines and see how their training changes over time.

6.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
WT-F-1.0	As a user, I want to log workouts with exercises and reps.	Workout Tracking	Records save correctly and visible in history.
WT-F-2.0	As a user, I want to browse the exercise library.	Workout Tracking	List shows exercises with category and muscles.
WT-F-3.0	As a user, I want to view my workout history.	Workout Tracking	History lists previous workouts in date order.
WT-F-4.0	As a user, I want reusable workout templates.	Workout Tracking	Templates save, edit, and load into new sessions.
WT-F-5.0	As a user, I want calorie estimates for workouts.	Workout Tracking	Workout record includes estimated calories burned.
WT-F-6.0	As a user, I want to see personal bests per exercise.	Workout Tracking	Best weight/reps updated automatically.
WT-F-7.0	As a user, I want to link workouts to a gym.	Workout Tracking	Gym field optional; saved location shown in history.

6.2. Data Dictionary:

Entity: EXERCISE

Attribute Name	Data Type	Description
id	UUID	Exercise id.
name	String	Exercise name.
category	String	Category such as strength or cardio.
muscle_groups	JSON	Target muscle groups.
instructions	Text	How to perform the exercise.
video_url	String	Tutorial video link.
image_url	String	Exercise image link.
difficulty	String	Difficulty level.
equipment	String	Required equipment.
created_at	Datetime	Creation time.

Entity: WORKOUT_LOG

Attribute Name	Data Type	Description
id	UUID	Workout log id.
user_id	UUID	References USER(id).

Attribute Name	Data Type	Description
workout_name	String	User-visible workout name.
logged_at	Datetime	When workout was completed.
duration_minutes	Integer	Length of workout in minutes.
calories_burned	Float	Estimated calories burned.
exercises	JSON	Per-exercise sets, reps and weights.
notes	Text	User notes about the workout.
gym_id	UUID	References GYM(id) if workout done at a gym.
created_at	Datetime	Record creation time.

Entity: CUSTOM_WORKOUT

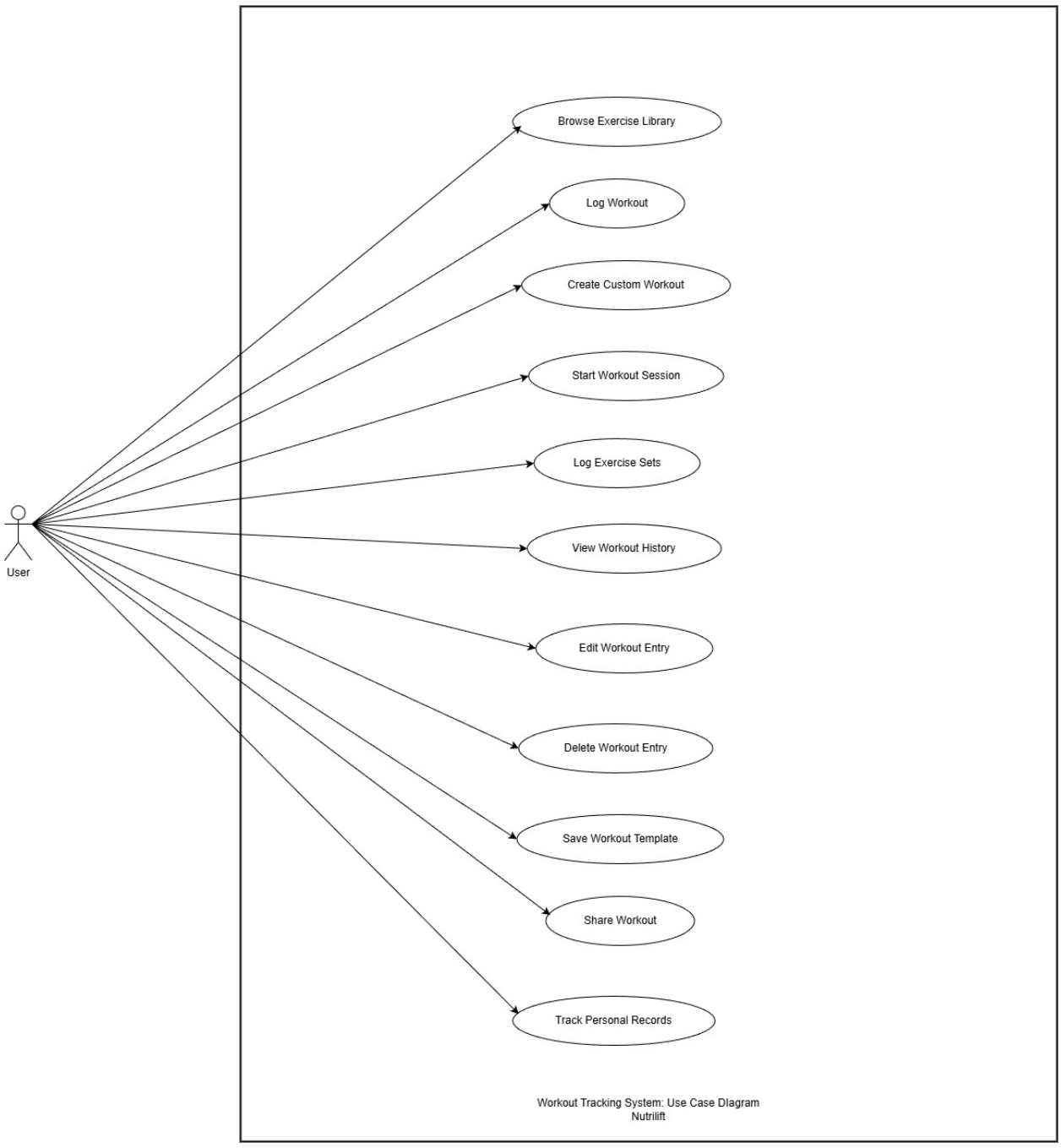
Attribute Name	Data Type	Description
id	UUID	Custom workout id.
user_id	UUID	References USER(id).
name	String	Custom workout name.
description	Text	Description of the routine.
exercises	JSON	List of exercise references with structure.
estimated_duration	Integer	Estimated duration in minutes.

Attribute Name	Data Type	Description
is_public	Boolean	Indicates if workout template is shareable.
created_at	Datetime	Creation time.
updated_at	Datetime	Last update time.

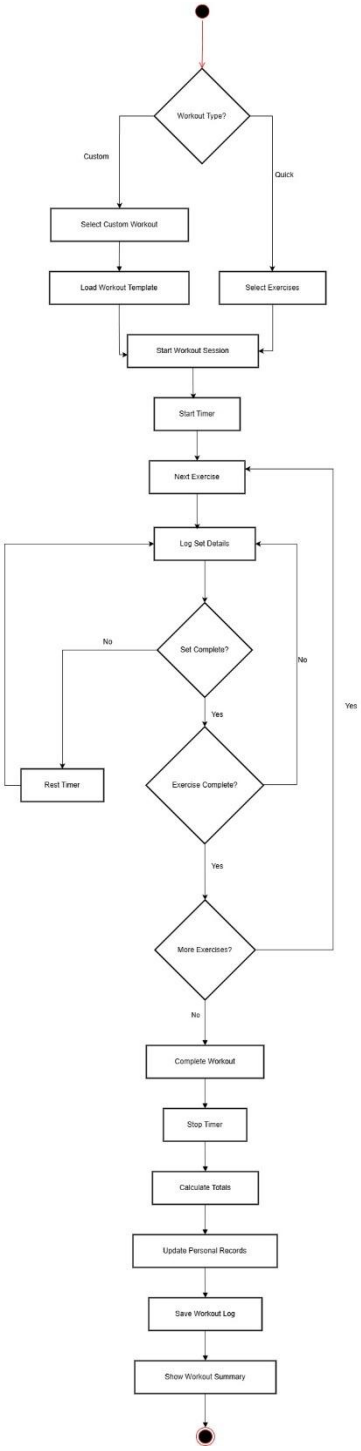
Entity: PERSONAL_RECORD

Attribute Name	Data Type	Description
id	UUID	Personal record id.
user_id	UUID	References USER(id).
exercise_id	UUID	References EXERCISE(id).
max_weight	Float	Maximum weight lifted.
max_reps	Integer	Maximum repetitions performed.
max_volume	Float	Total volume lifted for best set or session.
achieved_at	Datetime	Time when record was achieved.

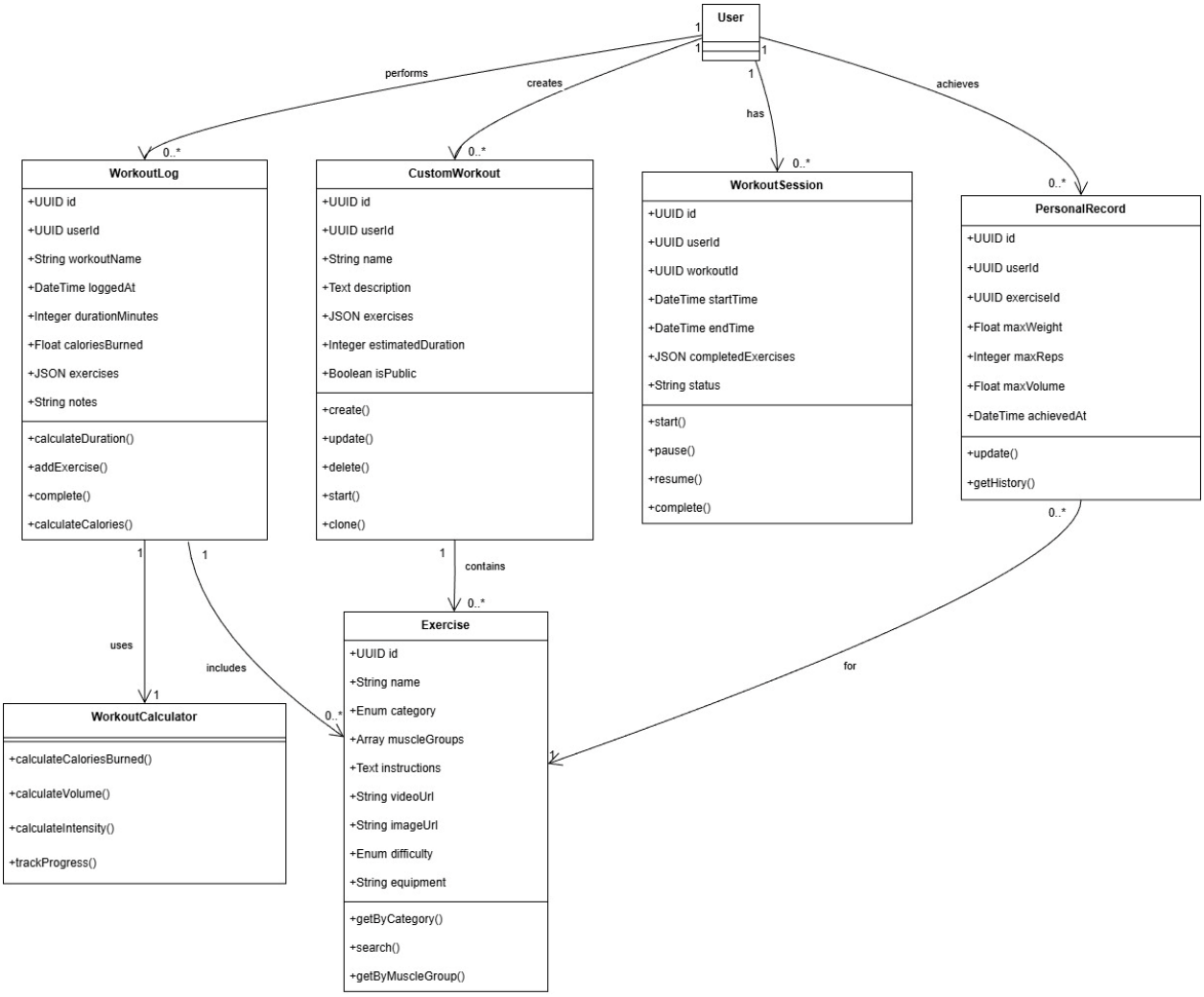
6.3. Diagrams:



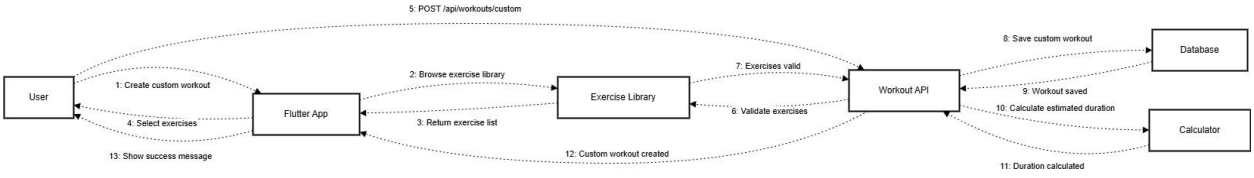
Workout Tracking 1: Use Case Diagram



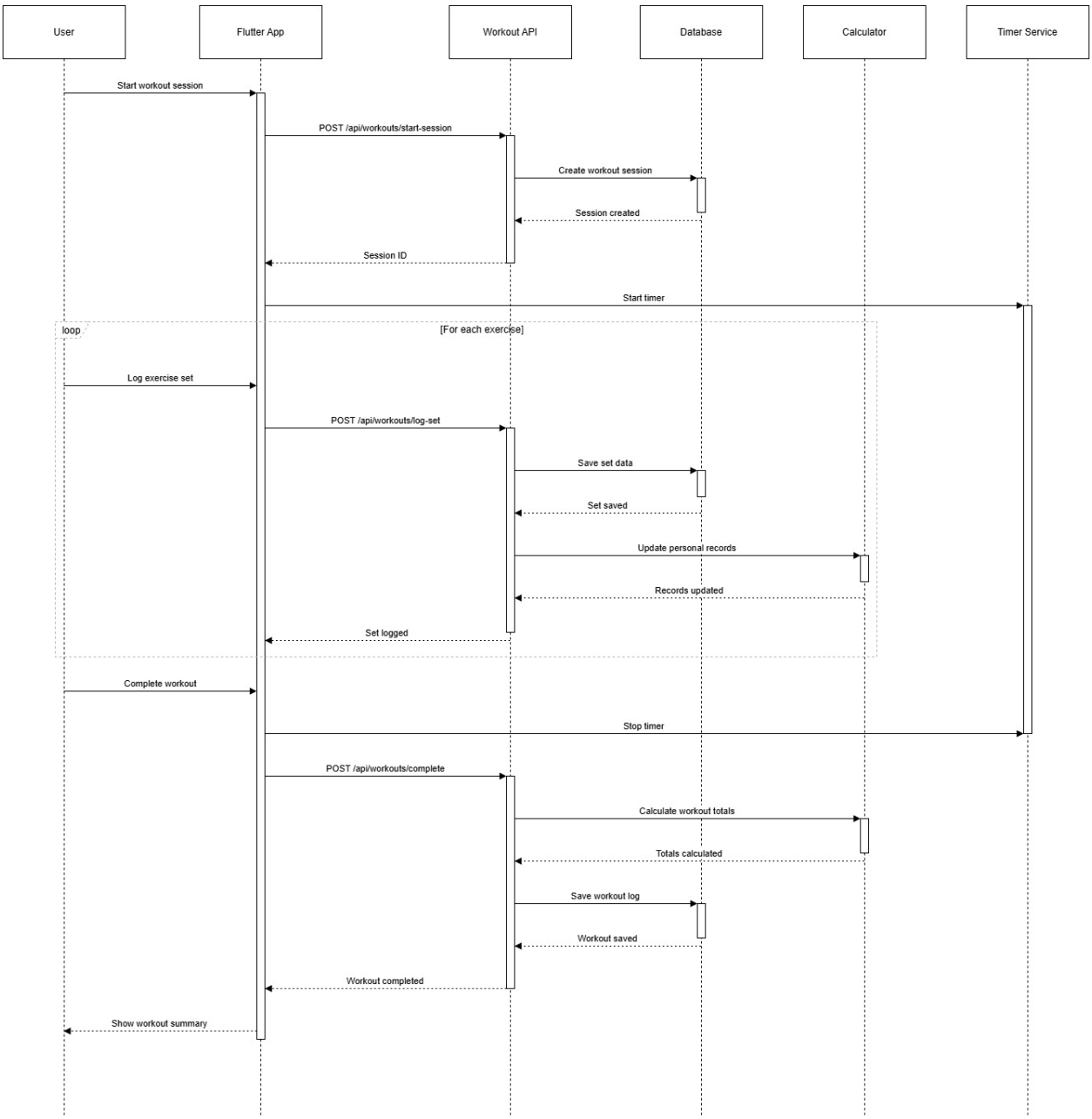
Workout Tracking 2: Activity Diagram



Workout Tracking 3: Class Diagram



Workout Tracking 4: Collaboration Diagram



Workout Tracking 5: Sequence Diagram

7. Rep Count Module

This subsystem uses the device camera to count repetitions for supported exercises automatically. It shows a live rep counter during the session and saves basic session details for later review.

7.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
RC-F-1.0	As a user, I want the app to automatically count reps using the camera so that I do not have to count manually.	Rep Count Module	Rep session uses camera and counts about 10 reps with small error. SRS_NutriLift.docx
RC-F-2.0	As a user, I want to see a live rep counter during the exercise so that I know the system is working.	Rep Count Module	Current rep number is visible and increases as reps are detected. SRS_NutriLift.docx
RC-F-3.0	As a user, I want rep sessions to be saved so that I can review them later.	Rep Count Module	Finished sessions appear in a history with exercise, reps and time. SRS_NutriLift.docx
RC-F-4.0	As a user, I want to know when tracking quality is low so that I can adjust my position.	Rep Count Module	App shows a simple warning when angle or light is bad. SRS_NutriLift.docx

7.2. Data Dictionary:

Entity: REP_SESSION

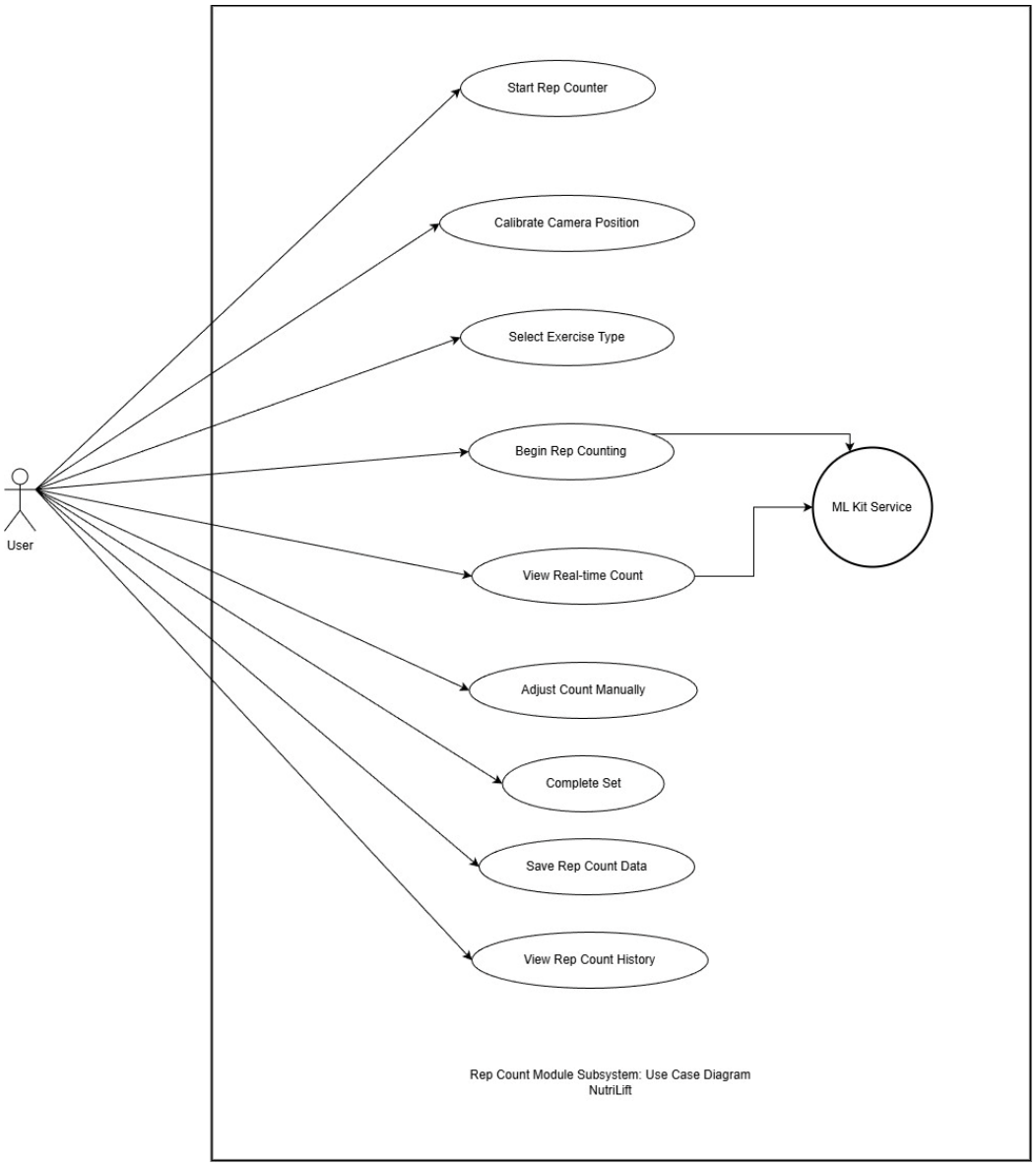
Attribute Name	Data Type	Description
id	UUID	Rep counting session id.
user_id	UUID	References USER(id).
exercise_id	UUID	References EXERCISE(id).
start_time	Datetime	Session start time.
end_time	Datetime	Session end time.
total_reps	Integer	Total repetitions detected by the model.
source_video	String	Optional video file path or URL.

Entity: REP_EVENT

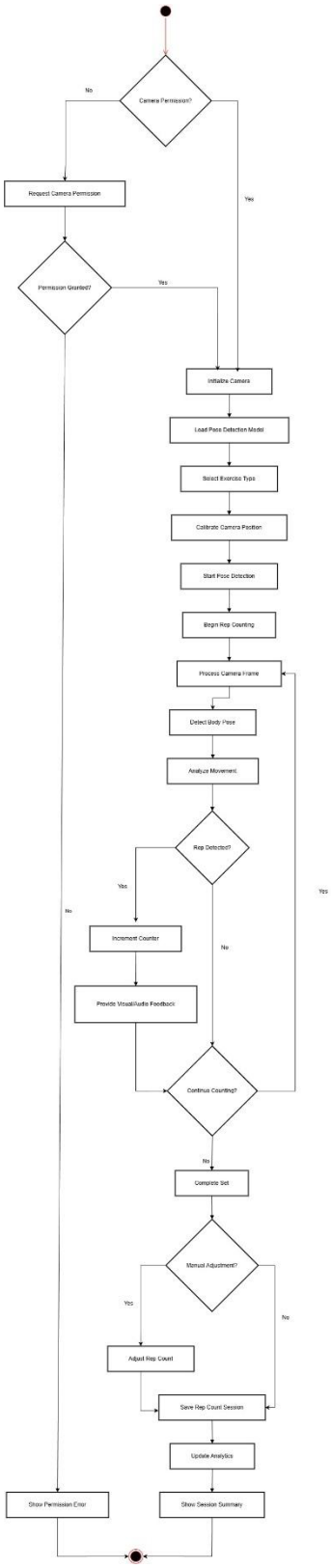
Attribute Name	Data Type	Description
id	UUID	Rep event id.
session_id	UUID	References REP_SESSION(id).
rep_number	Integer	Sequential rep count within the session.

Attribute Name	Data Type	Description
timestamp	Datetime	Time when rep was detected.
confidence	Float	Model confidence score for this rep.

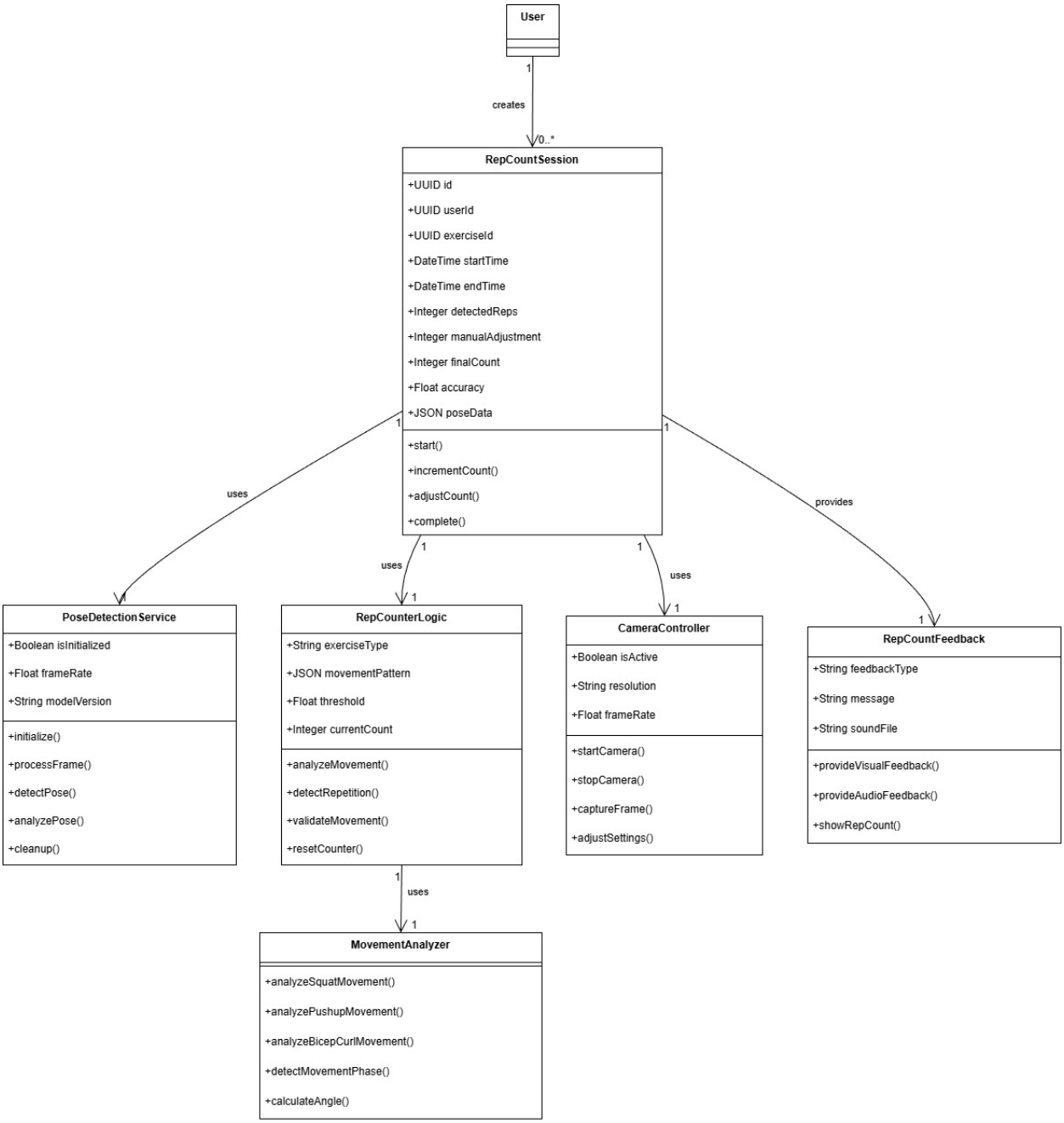
7.3. Diagrams:



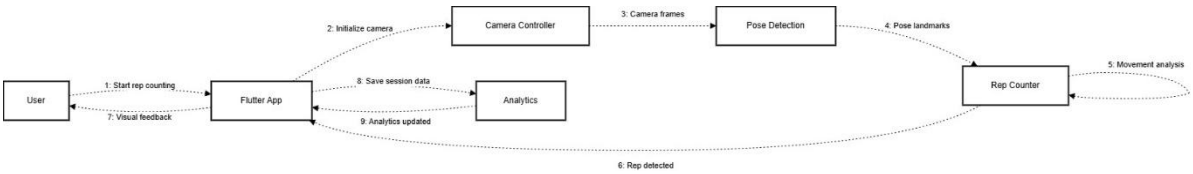
Rep Count 1: Use Case Diagram



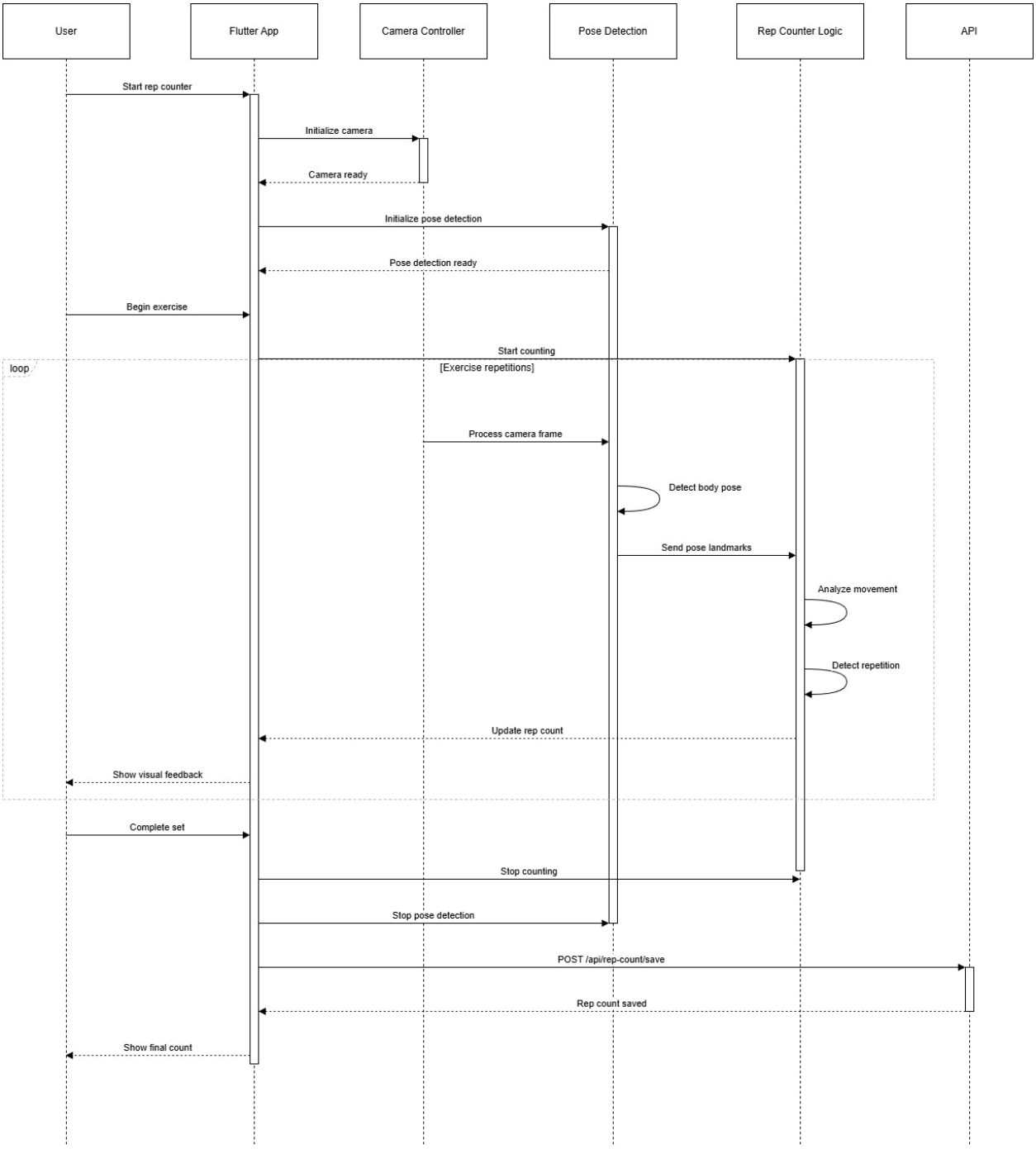
Rep Count 2: Activity Diagram



Rep Count 3: Class Diagram



Rep Count 4: Collaboaration Diagram



Rep Count 5: Sequence Diagram

8. Challenge and Gamification

This subsystem provides challenges, streaks, badges and leaderboards to make consistent activity more rewarding. It updates progress using existing nutrition and workout data so users do not have to enter anything twice.

8.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
CG-F-1.0	As a user, I want to browse and join active challenges so that I stay motivated.	Challenges & Gamification	Active challenges list shows name, metric, goal and dates; join status is stored. SRS_NutriLift.docx
CG-F-2.0	As a user, I want my challenge progress to update automatically from my nutrition and workout logs so that there is no double entry.	Challenges & Gamification	Logging food or workouts updates joined challenge progress. SRS_NutriLift.docx
CG-F-3.0	As a user, I want to see a clear start and end date for each challenge so that I know how long I have.	Challenges & Gamification	Detail page shows start, end and remaining time; challenge closes after end date. SRS_NutriLift.docx
CG-F-4.0	As a user, I want streaks for consecutive active days so that I am rewarded for consistency.	Challenges & Gamification	Current and longest streak values change when days are active or missed. SRS_NutriLift.docx
CG-F-5.0	As a user, I want badges for specific milestones so that my achievements are visible.	Challenges & Gamification	Hitting a milestone adds a badge that appears in a badge list on profile. SRS_NutriLift.docx
CG-F-6.0	As a user, I want a simple leaderboard so that I can compare my performance with others.	Challenges & Gamification	Leaderboard shows top users and the logged-in user’s rank. SRS_NutriLift.docx

8.2. Data Dictionary:

Entity: CHALLENGE

Attribute Name	Data Type	Description
id	UUID	Challenge id.
name	String	Challenge name.
description	Text	Description and rules.
challenge_type	String	Type such as nutrition, workout, mixed.
goal_value	Float	Target metric to achieve.
unit	String	Unit such as kcal, reps, days.
start_date	Datetime	Challenge start date and time.
end_date	Datetime	Challenge end date and time.
created_by	UUID	References USER(id) as creator.
is_active	Boolean	Indicates if challenge is currently active.
created_at	Datetime	Creation time.

Entity: CHALLENGE_PARTICIPANT

Attribute Name	Data Type	Description
id	UUID	Challenge participant id.
challenge_id	UUID	References CHALLENGE(id).
user_id	UUID	References USER(id).
progress	Float	Current progress towards goal.
completed	Boolean	Completion status of the challenge.
joined_at	Datetime	Time user joined the challenge.
completed_at	Datetime	Time challenge was completed by the user.
rank	Integer	User rank on the challenge leaderboard.

Entity: BADGE

Attribute Name	Data Type	Description
id	UUID	Badge id.
name	String	Badge name.
description	Text	Explanation of the badge.
icon_url	String	Badge icon path or URL.

Attribute Name	Data Type	Description
criteria	JSON	Rules or thresholds for awarding the badge.
points_reward	Integer	Points awarded when badge is earned.
is_active	Boolean	Indicates if badge is active.
created_at	Datetime	Creation time.

Entity: USER_BADGE

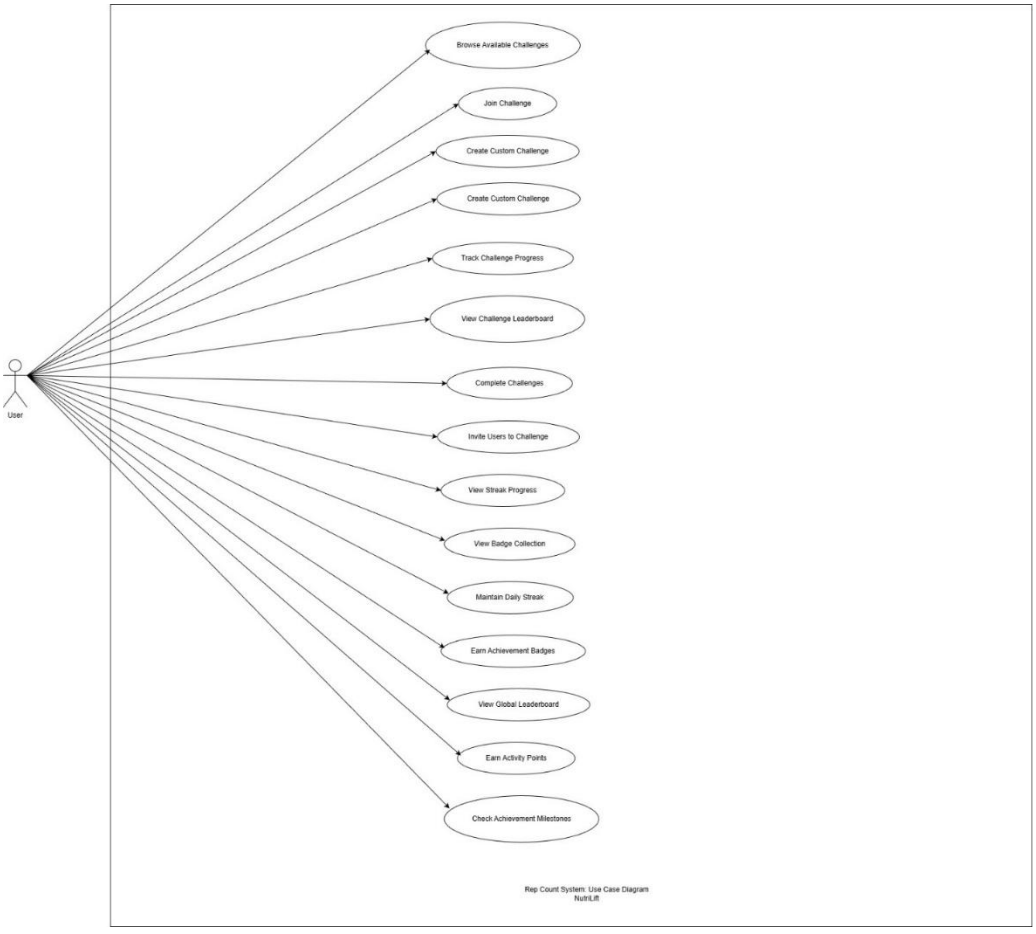
Attribute Name	Data Type	Description
id	UUID	User badge id.
user_id	UUID	References USER(id).
badge_id	UUID	References BADGE(id).
earned_at	Datetime	Time when user earned the badge.

Entity: STREAK

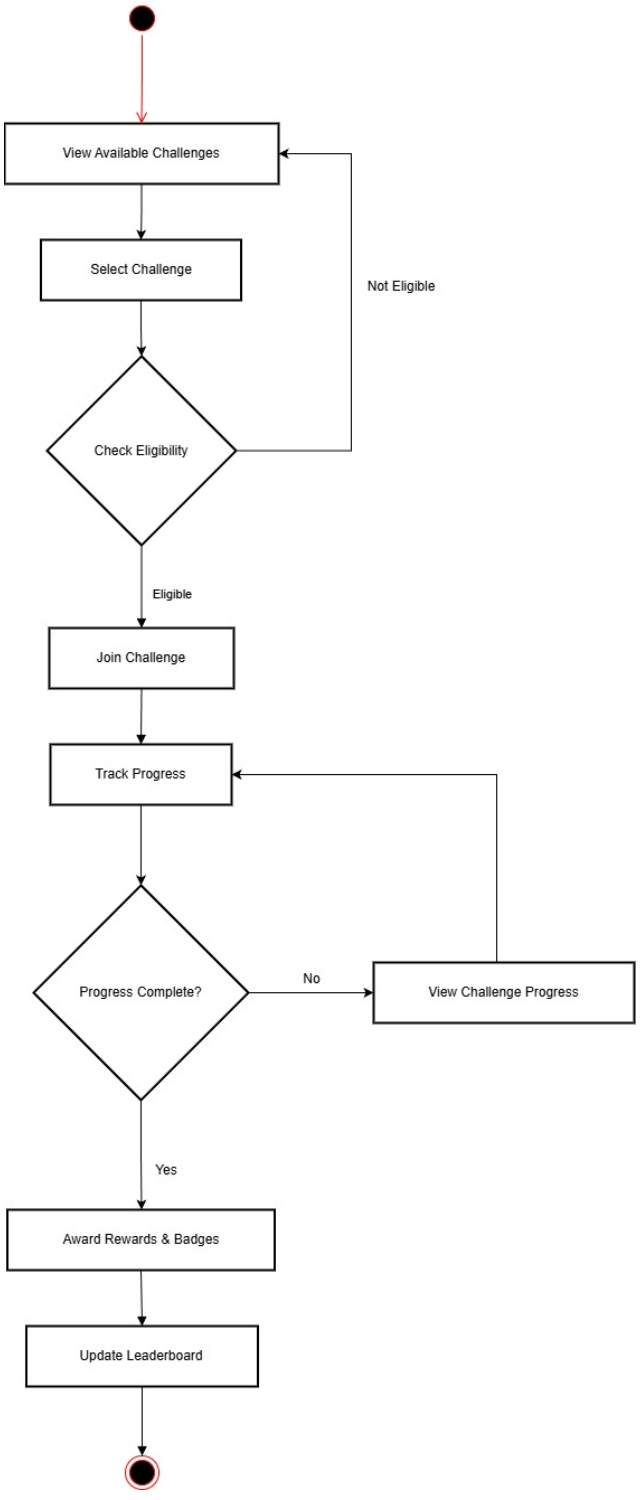
Attribute Name	Data Type	Description
id	UUID	Streak id.
user_id	UUID	References USER(id).

Attribute Name	Data Type	Description
current_streak	Integer	Current consecutive active days.
longest_streak	Integer	Longest streak achieved.
last_active_date	Date	Last date user was active.
updated_at	Datetime	Last update time.

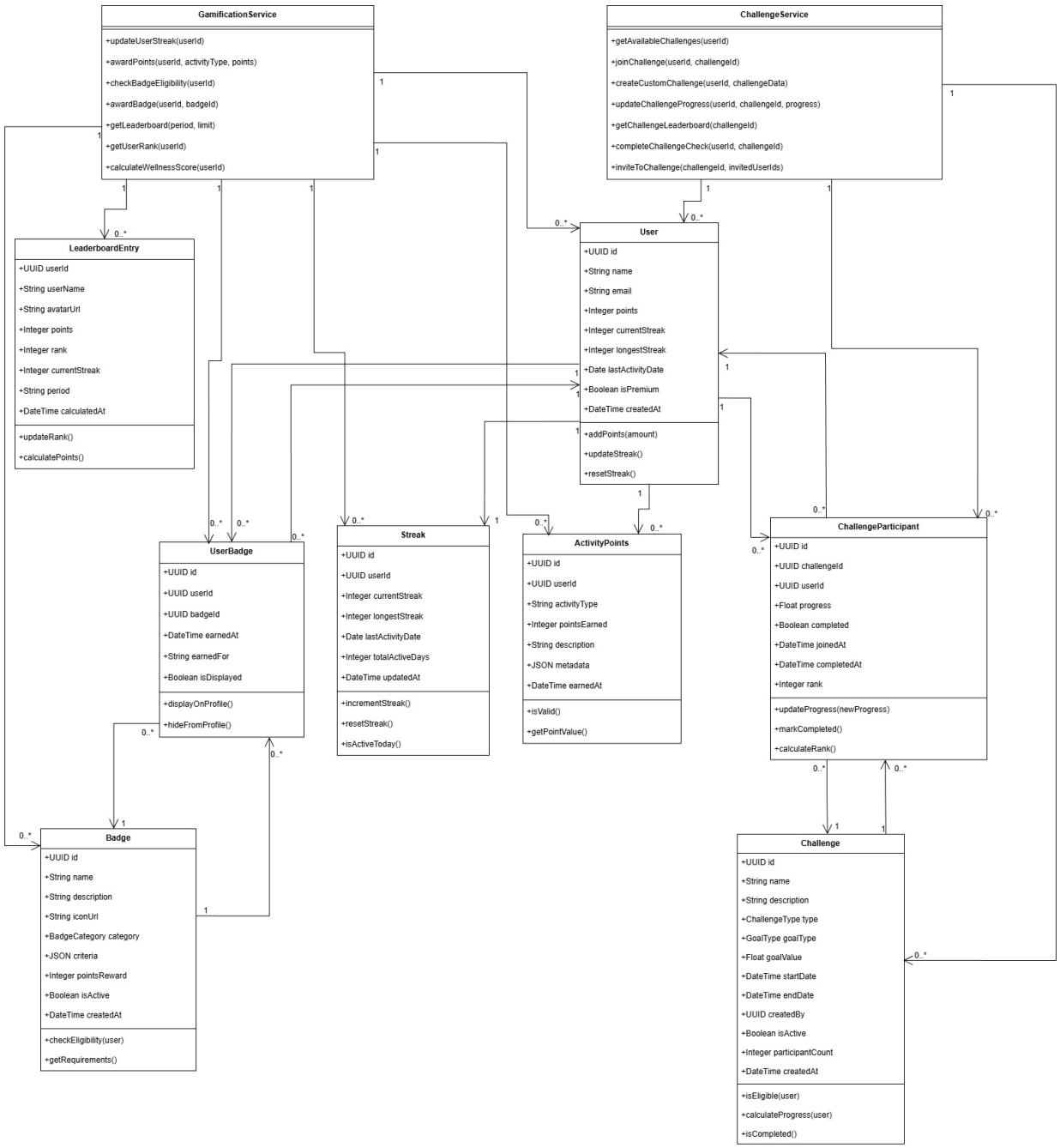
8.3. Diagrams:



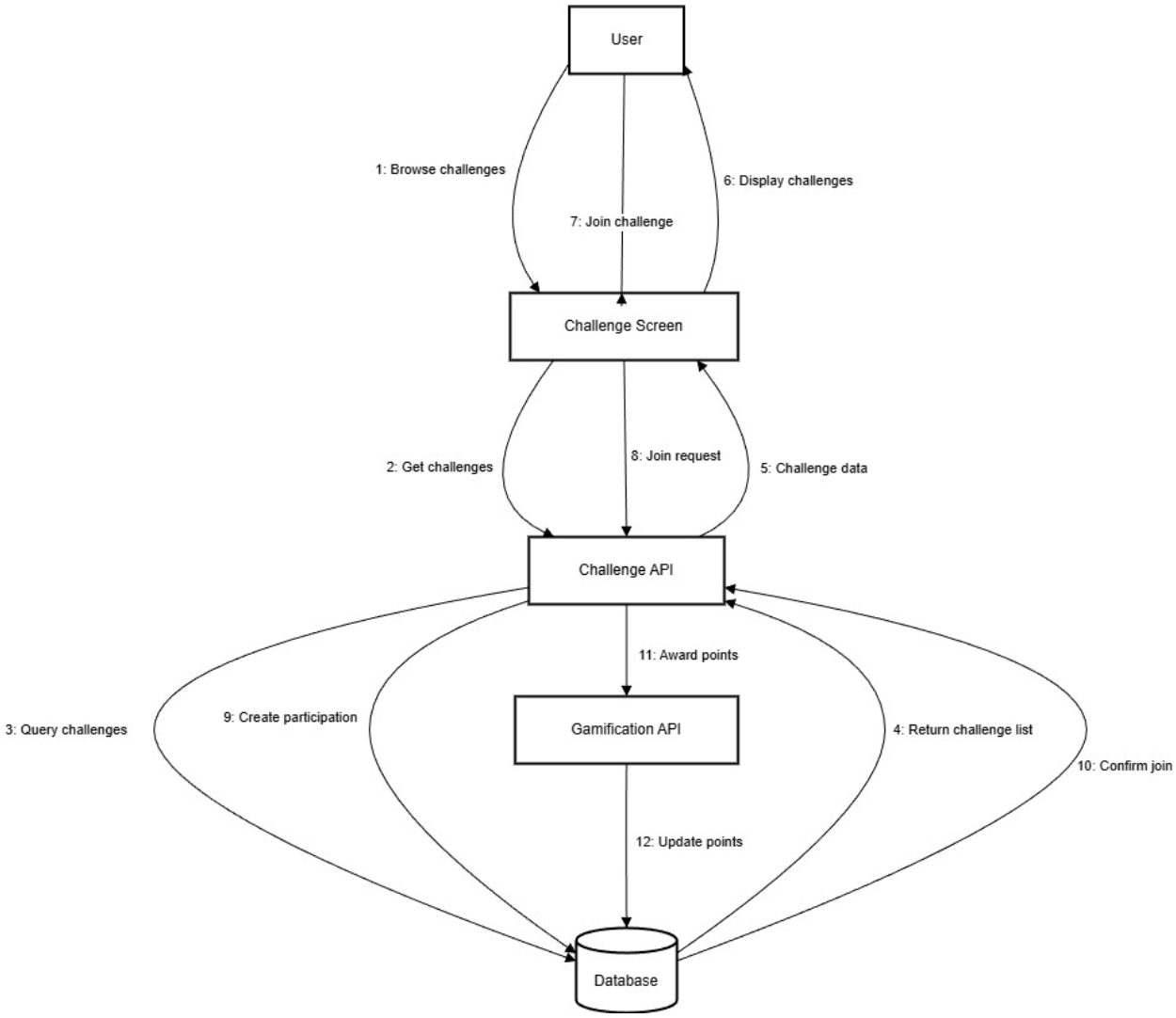
Challenge&Game 1:Use Case Diagram



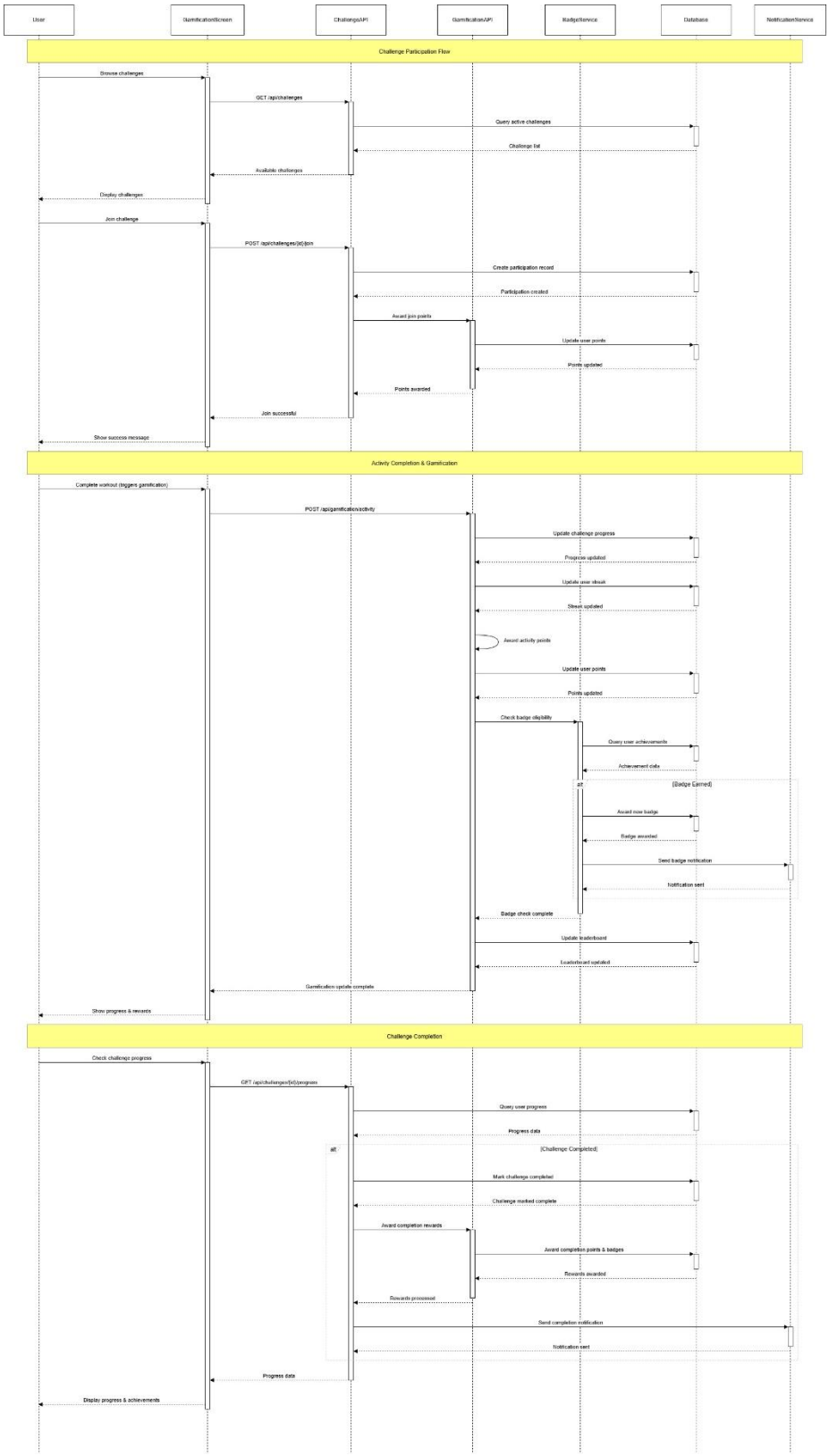
Challenge&Game 2: Activity Diagram



Challenge&Game 3: Class Diagram



Challenge&Game 4: Collaboration Diagram



Challenge&Game 5: Sequence Diagram

9. Community Module

This subsystem offers a simple social space where users can post updates, like and comment on content, and follow others. It also includes reporting and moderation so that inappropriate posts can be reviewed and removed.

9.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
CM-F-1.0	As a user, I want to post updates with text and images so that I can share my progress.	Community Module	New post appears in feed with correct text, image and time. SRS_NutriLift.docx
CM-F-2.0	As a user, I want to like and unlike posts so that I can support others.	Community Module	Like button toggles and like count updates without duplicates. SRS_NutriLift.docx
CM-F-3.0	As a user, I want to comment on posts so that I can interact with the community.	Community Module	Added comment shows under the post with author and time. SRS_NutriLift.docx
CM-F-4.0	As a user, I want to follow and unfollow other users so that my feed feels relevant.	Community Module	Follow button switches between Follow / Following; followed users appear more in feed. SRS_NutriLift.docx
CM-F-5.0	As a user, I want to report inappropriate posts so that the feed stays safe.	Community Module	Reporting saves a record with reason; post is flagged for review. SRS_NutriLift.docx
CM-F-6.0	As an admin, I want to moderate reported posts so that I can remove harmful content.	Community Module	Admin can mark a post removed; it disappears from normal feeds. SRS_NutriLift.docx

9.2. Data Dictionary:

Entity: POST

Attribute Name	Data Type	Description
id	UUID	Post id.
user_id	UUID	References USER(id).
content	Text	Text content of the post.
image_urls	String	Comma-separated or JSON list of image URLs.
like_count	Integer	Cached count of likes.
comment_count	Integer	Cached count of comments.
is_reported	Boolean	Flag indicating reported content.
is_removed	Boolean	Flag indicating moderation removal.
created_at	Datetime	Post creation time.
updated_at	Datetime	Last update time.

Entity: Comment

Attribute Name	Data Type	Description
id	UUID	Comment id.
post_id	UUID	References POST(id).
user_id	UUID	References USER(id).
content	Text	Comment text content.
created_at	Datetime	Comment creation time.

Entity: LIKE

Attribute Name	Data Type	Description
id	UUID	Like id.
post_id	UUID	References POST(id).
user_id	UUID	References USER(id).
created_at	Datetime	Time when like was created.

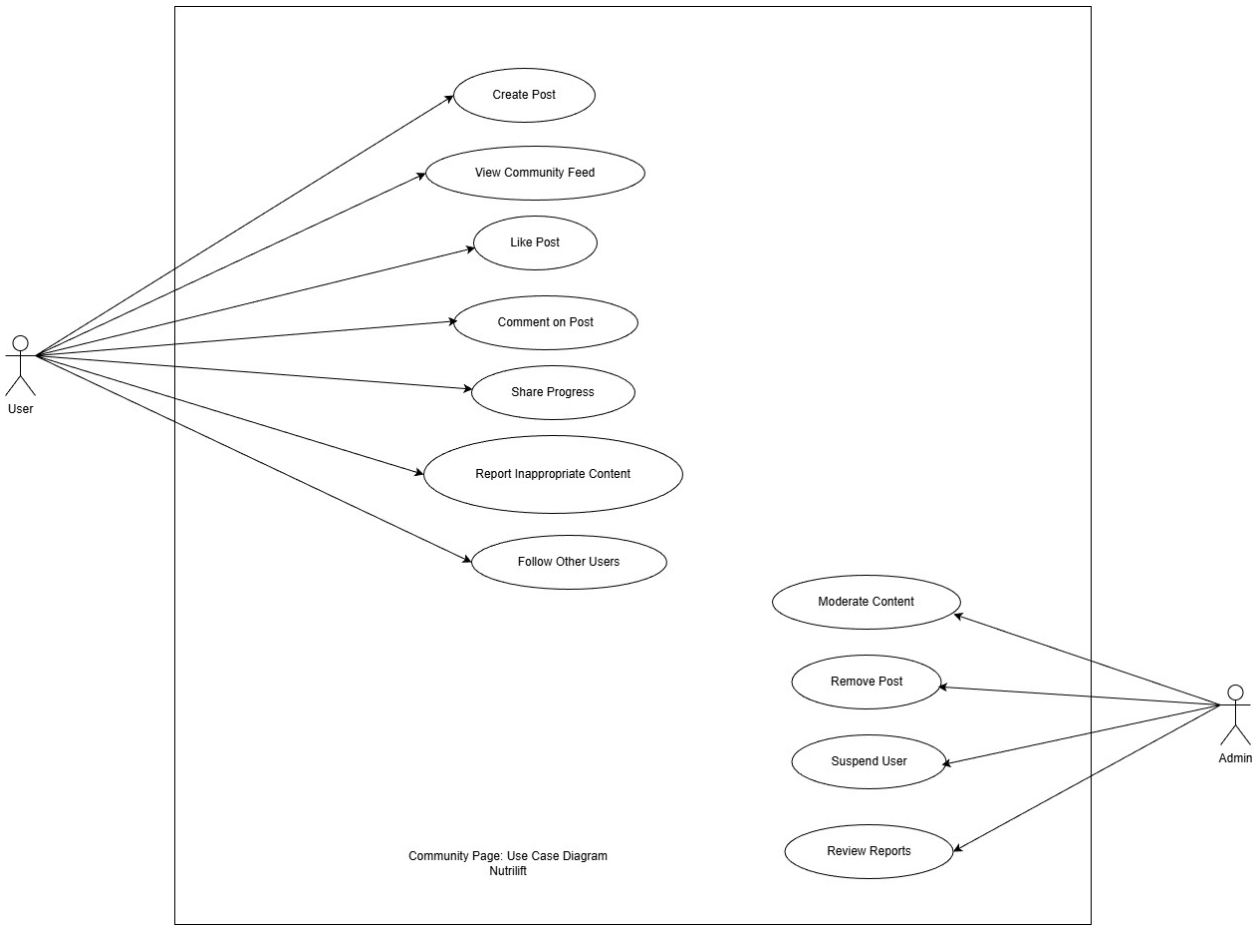
Entity: REPORT

Attribute Name	Data Type	Description
id	UUID	Report id.
post_id	UUID	References POST(id).
reported_by	UUID	References USER(id) who reported.
reason	Text	Reason for reporting the post.
status	String	Status such as pending, reviewed, actioned.
created_at	Datetime	Report creation time.
reviewed_at	Datetime	Time report was reviewed.

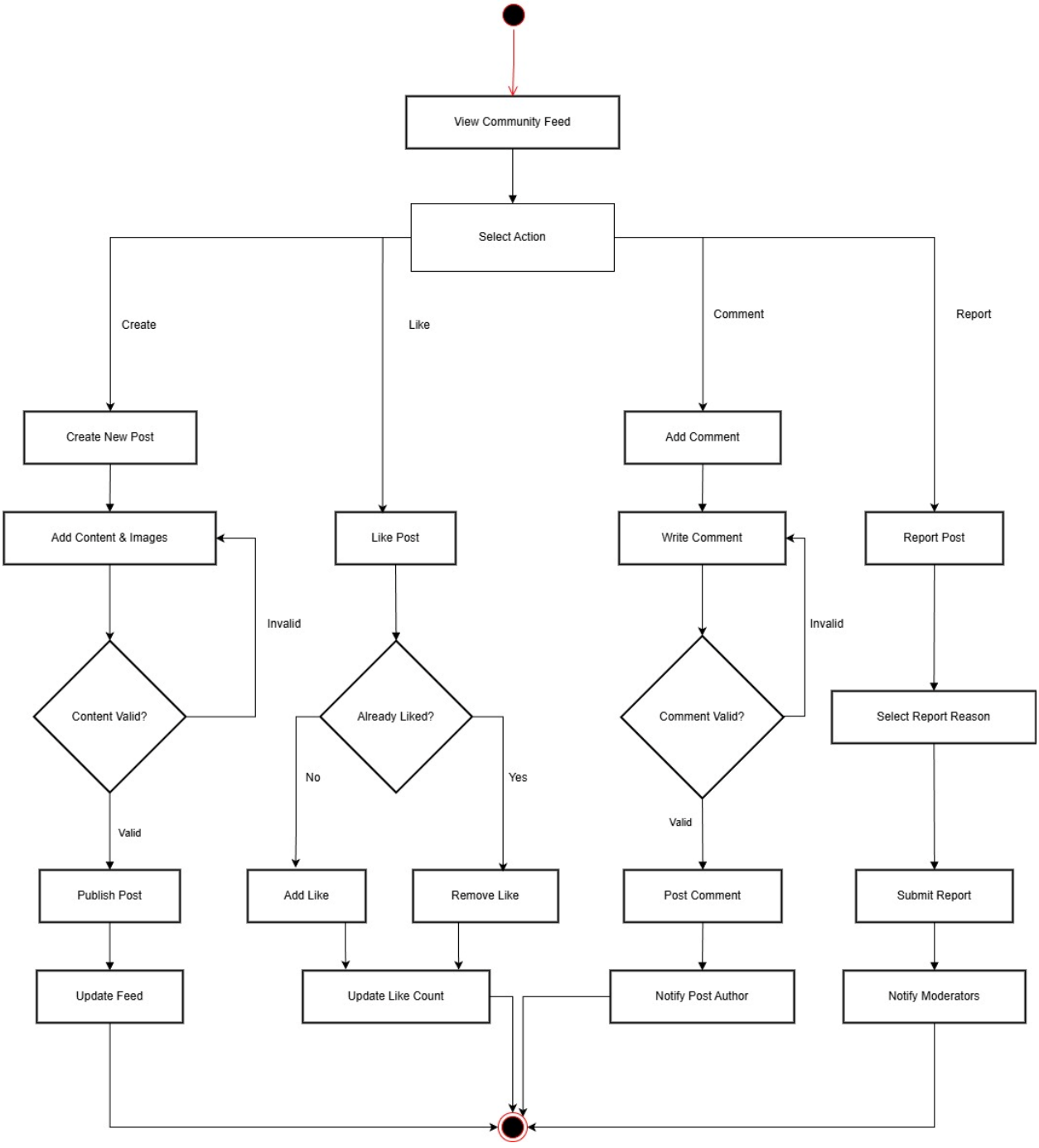
Entity: FOLLOW

Attribute Name	Data Type	Description
id	UUID	Follow relationship id.
follower_id	UUID	User who follows another user.
following_id	UUID	User being followed.
created_at	Datetime	Time when follow started.

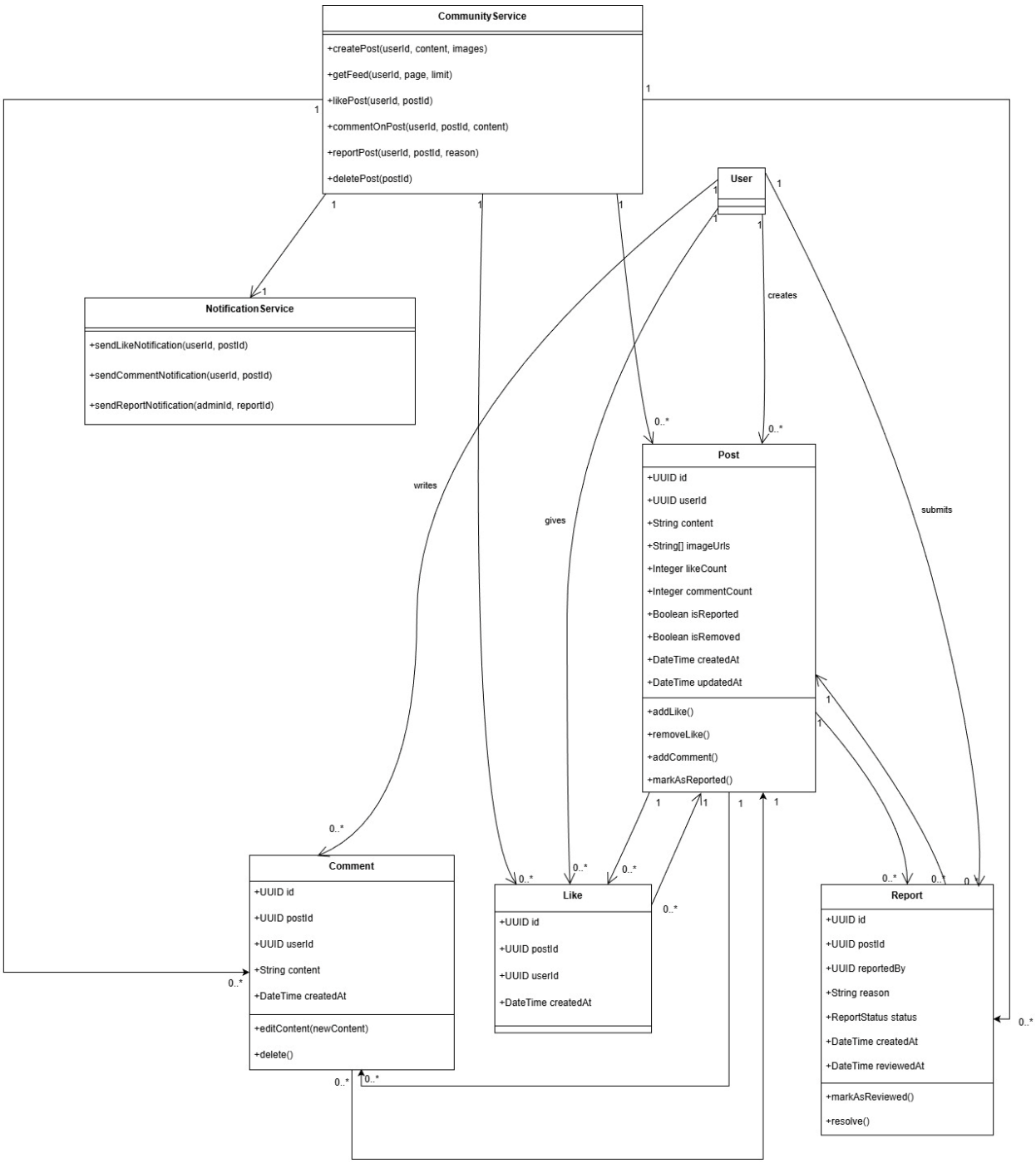
9.3. Diagrams:



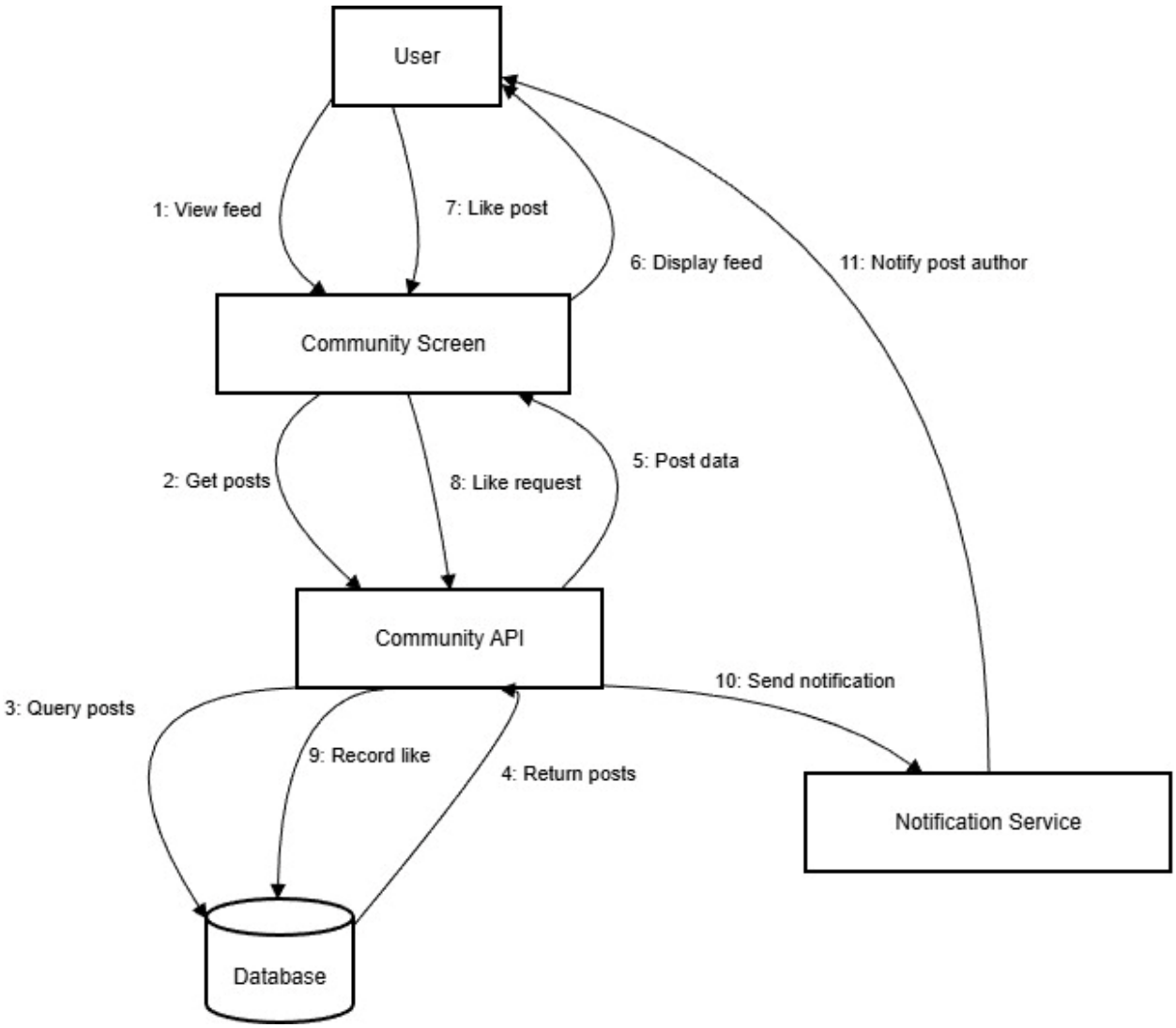
Community Page 1: Use Case Diagram



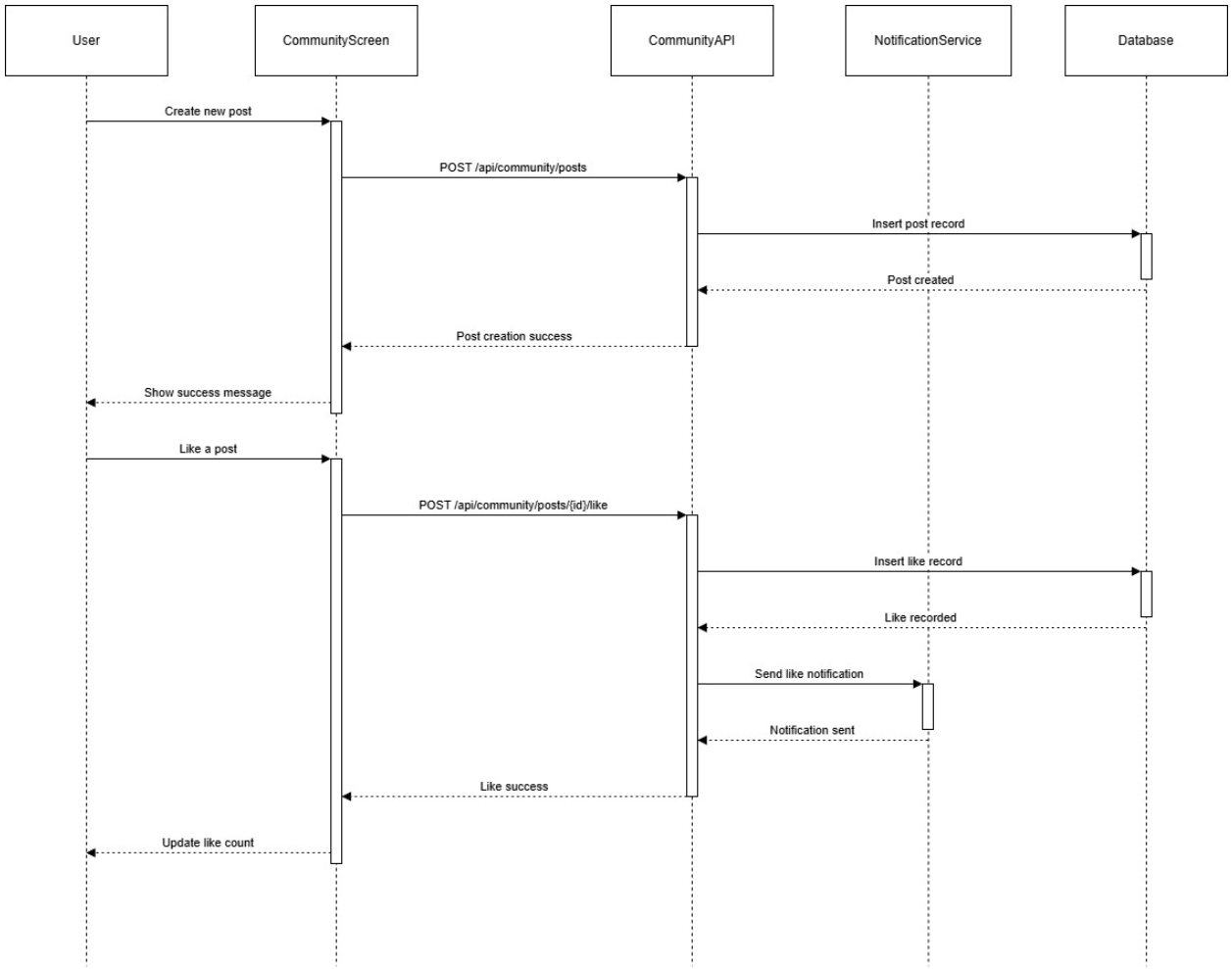
Community Page 2: Activity Diagram



Community Page 3: Class Diagram



Community Page 4: Collaboration Diagram



Community Page 5: Sequence Diagram

10. Progress Reports and Analytics

This subsystem turns raw logs into weekly and period-based summaries with charts and a wellness score. It lets users quickly see trends in their eating and training instead of interpreting numbers on their own.

10.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
PR-F-1.0	As a user, I want weekly graphs of calories and workouts so that I can see trends over time.	Progress Reports & Analytics	Weekly charts show daily calories and workouts matching stored logs. SRS_NutriLift.docx
PR-F-2.0	As a user, I want a wellness score for a chosen period so that I get a simple summary of my habits.	Progress Reports & Analytics	Selected period shows a score and grade that change when data changes. SRS_NutriLift.docx
PR-F-3.0	As a user, I want to filter reports by date range so that I can review specific periods.	Progress Reports & Analytics	Choosing start and end dates reloads summaries and charts for that range. SRS_NutriLift.docx
PR-F-4.0	As a user, I want to see both nutrition and workout summaries in the same place so that I can compare them.	Progress Reports & Analytics	Report screen shows nutrition and workout cards side by side. SRS_NutriLift.docx
PR-NF-1.1	Reports should load quickly for normal data sizes.	Progress Reports & Analytics	Weekly reports appear within a few seconds and UI does not freeze. SRS_NutriLift.docx

10.2. Data Dictionary:

Entity: NUTRITION_PROGRESS

Attribute Name	Data Type	Description
id	UUID	Nutrition progress id.
user_id	UUID	References USER(id).
progress_date	Date	Date for which data is summarized.
total_calories	Float	Total calories consumed that day.
total_protein	Float	Total protein consumed.
total_carbs	Float	Total carbohydrates consumed.
total_fats	Float	Total fats consumed.
adherence_percentage	Float	Percentage of adherence to nutrition goals.
calculated_at	Datetime	Time when summary was calculated.

Entity: WORKOUT_PROGRESS

Attribute Name	Data Type	Description
id	UUID	Workout progress id.
user_id	UUID	References USER(id).
progress_date	Date	Date for which workout is summarized.
workout_count	Integer	Number of workouts performed.
total_duration	Integer	Total workout duration in minutes.
calories_burned	Float	Total calories burned.
exercise_count	Integer	Number of exercises performed.
consistency_score	Float	Consistency metric for workouts.
calculated_at	Datetime	Time when summary was calculated.

Entity: WELLNESS_SCORE

Attribute Name	Data Type	Description
id	UUID	Wellness score id.
user_id	UUID	References USER(id).
score_date	Date	Date of wellness score.

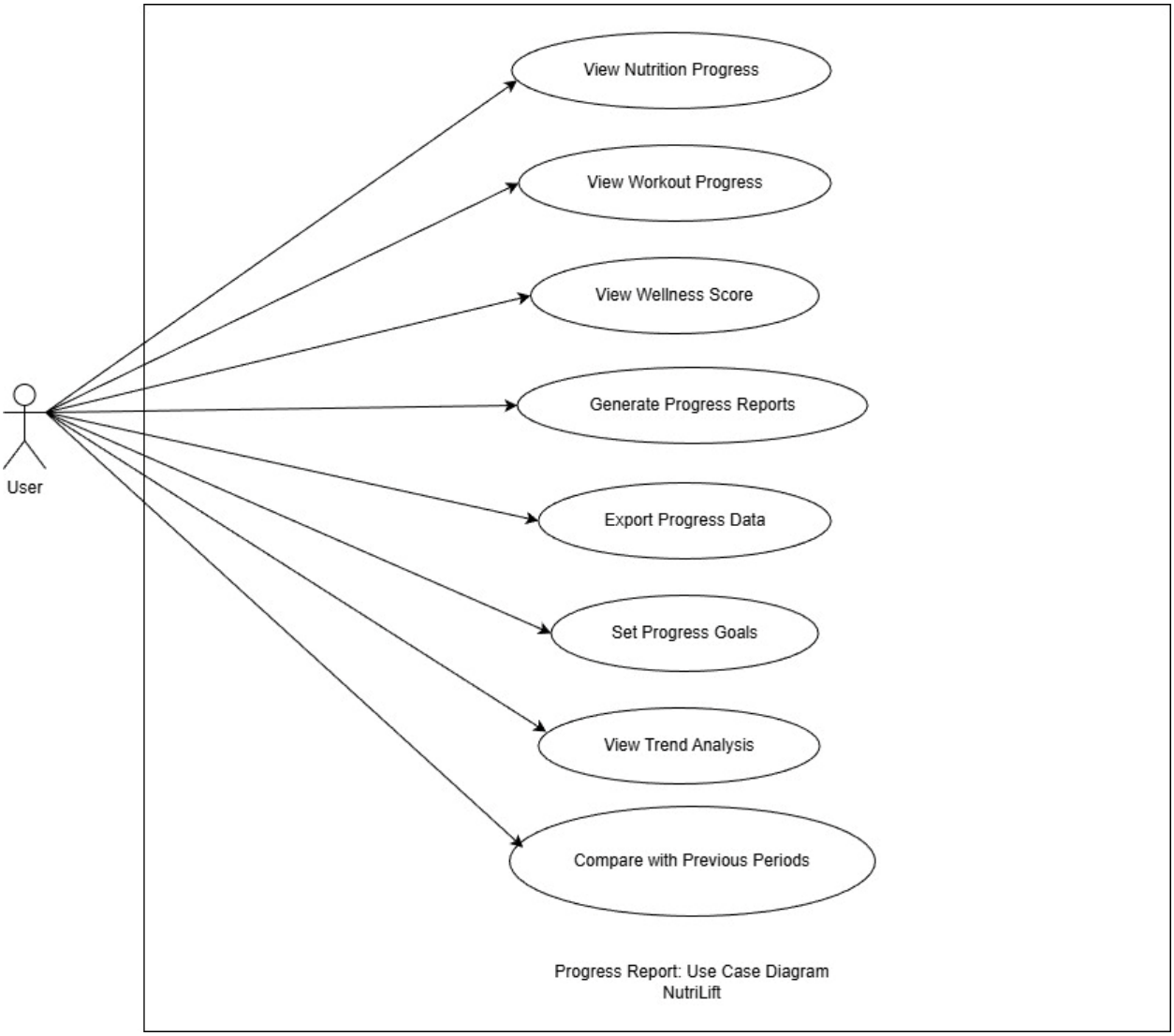
Attribute Name	Data Type	Description
nutrition_score	Float	Nutrition component of wellness.
fitness_score	Float	Fitness component of wellness.
consistency_score	Float	Consistency component of wellness.
overall_score	Float	Combined overall score.
score_grade	String	Grade such as A, B and so on.
calculated_at	Datetime	Time when score was calculated.

Entity: PROGRESS_REPORT

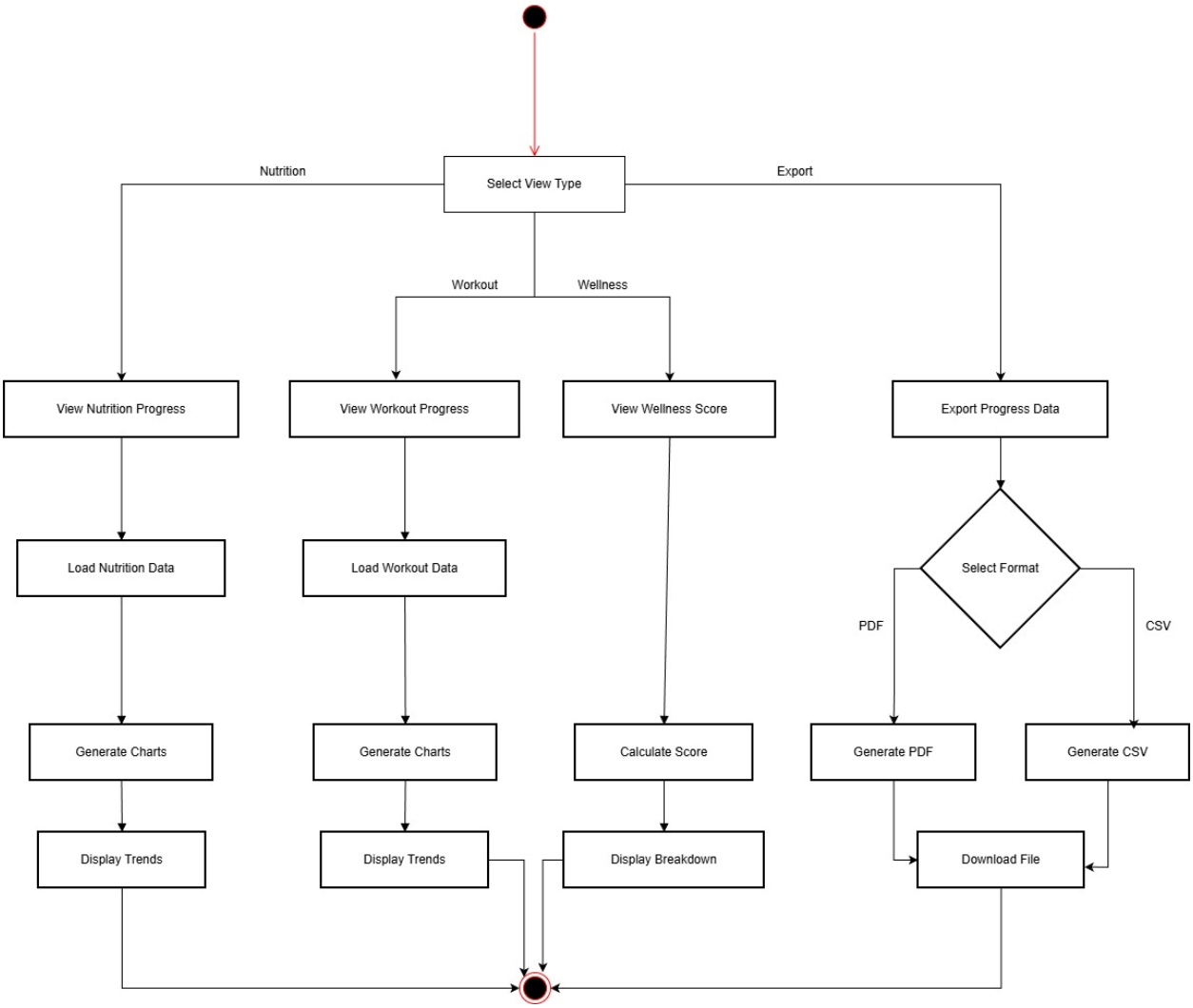
Attribute Name	Data Type	Description
id	UUID	Progress report id.
user_id	UUID	References USER(id).
report_type	String	Type such as weekly, monthly, custom.
start_date	Date	Start date of report period.
end_date	Date	End date of report period.
report_data	JSON	Aggregated report data.
format	String	Export format such as pdf, csv, json.

Attribute Name	Data Type	Description
generated_at	Datetime	Time when report was generated.

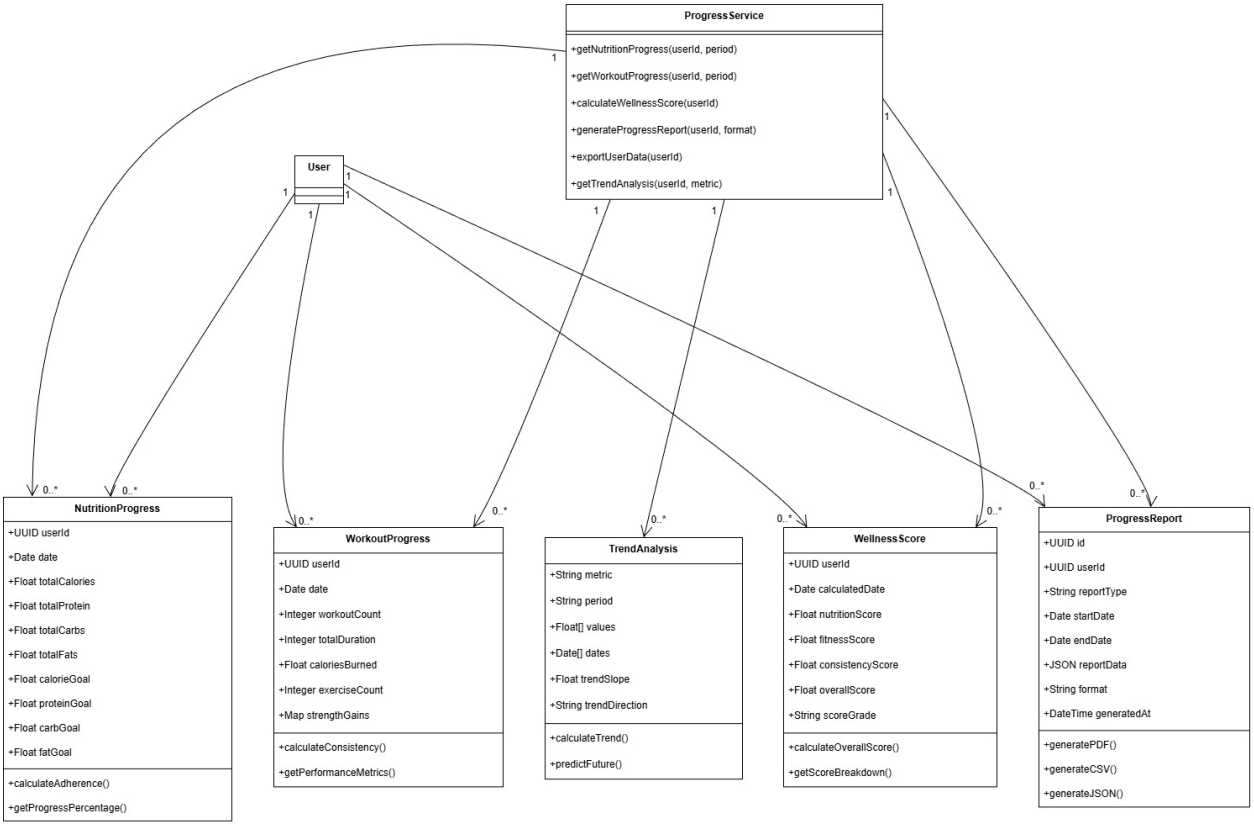
10.3. Wireframe:Diagrams:



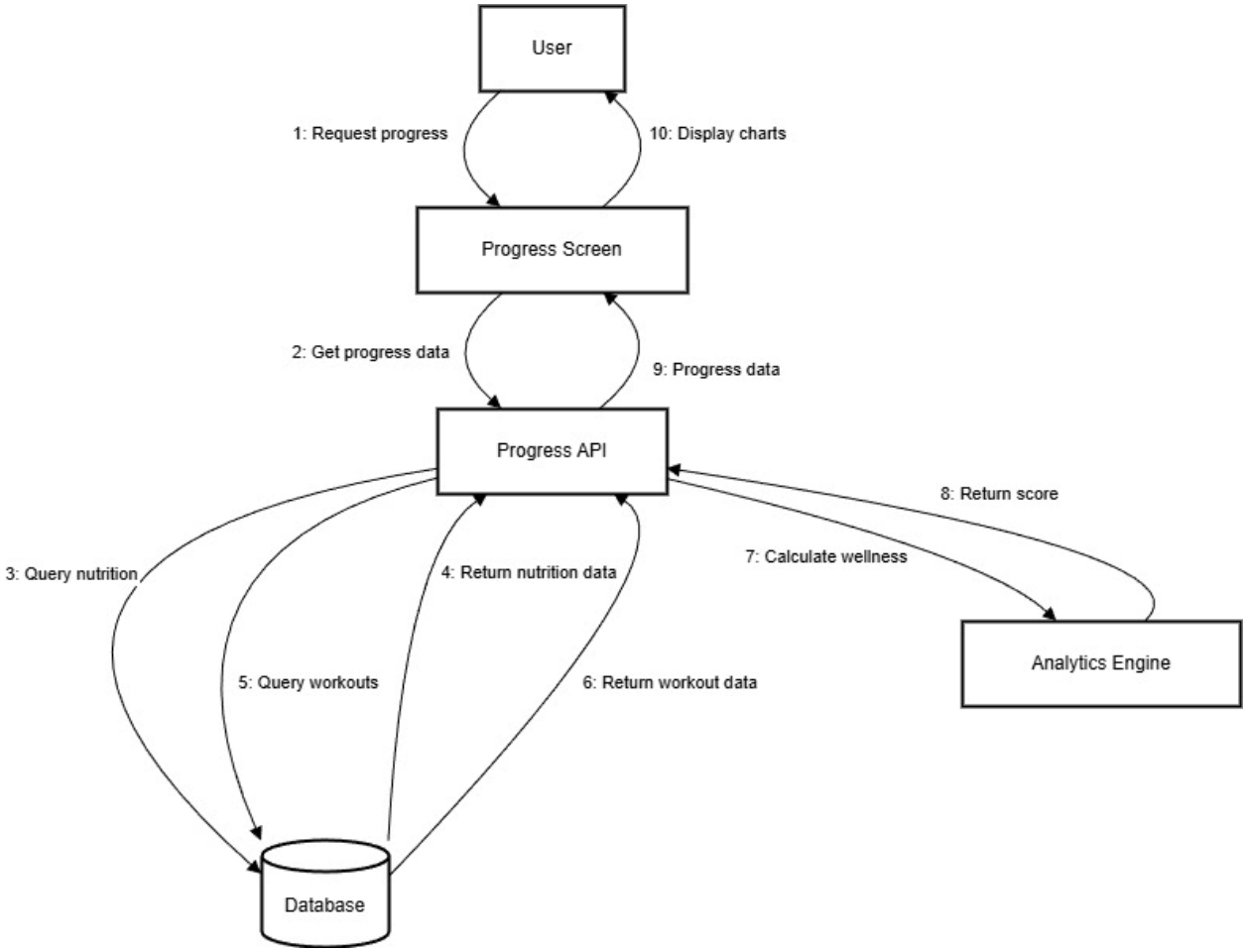
Progress Report 1: Use Case Diagram



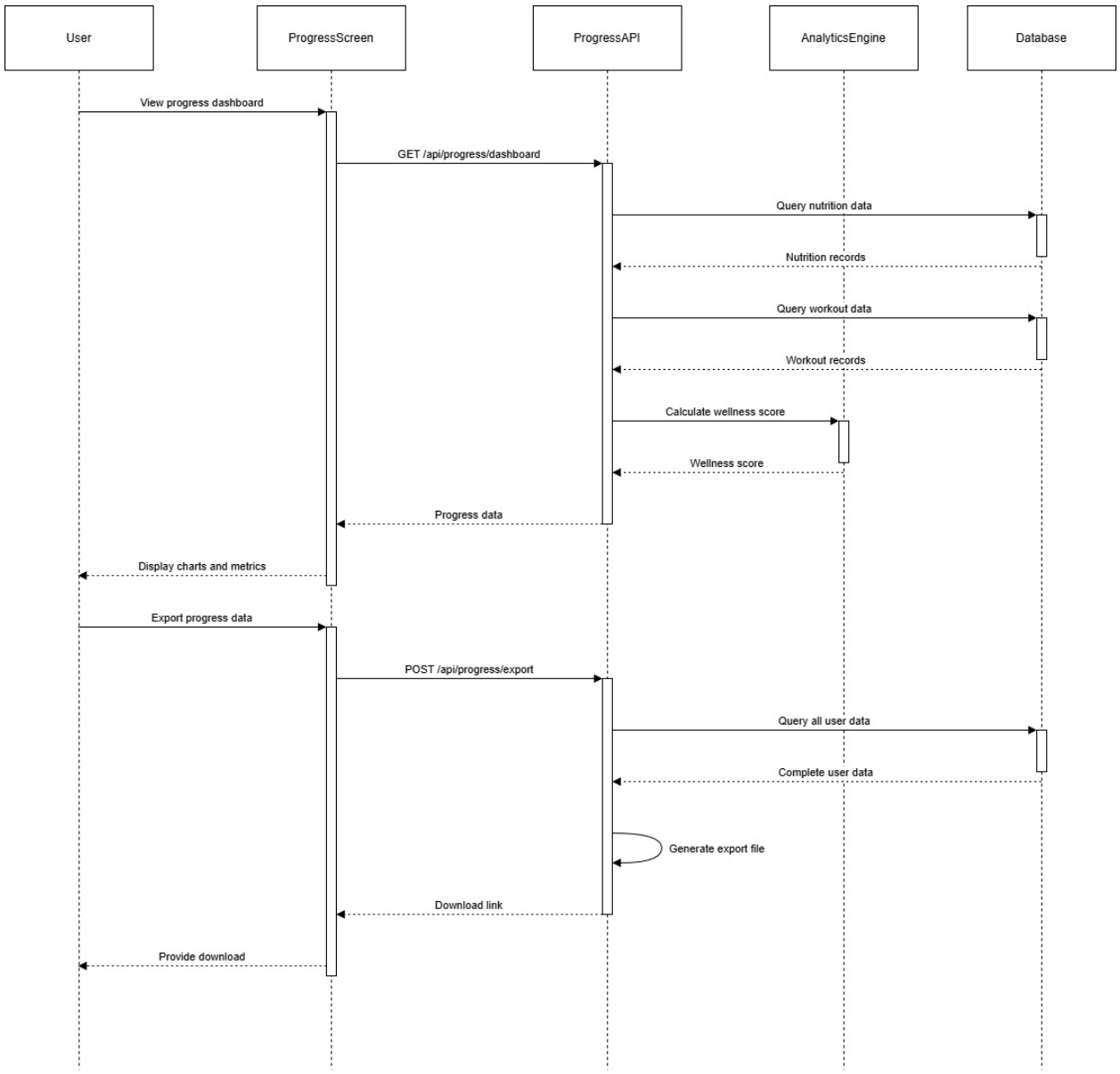
Progress Report 2: Activity Diagram



Progress Report 3: Class Diagram



Progress Report 4: Collaboration Diagram



Progress Report 5: Sequence Diagram

11.Payment Integration

This subsystem manages premium plans and payments through a secure gateway. It keeps track of transactions and subscription status so the app knows when to unlock or remove premium features.

11.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
PY-F-1.0	As a user, I want to see available premium plans so that I can choose one that suits me.	Payment Integration	Premium screen lists plans with name, duration and price. SRS_NutriLift.docx
PY-F-2.0	As a user, I want to pay securely for premium so that my card details are safe.	Payment Integration	Payment uses gateway flow; app never stores raw card data; success activates premium. SRS_NutriLift.docx
PY-F-3.0	As a user, I want the app to remember my premium status so that I do not pay again unnecessarily.	Payment Integration	After restart or re-login, premium features stay unlocked while active. SRS_NutriLift.docx
PY-F-4.0	As a user, I want to view my payment history so that I can keep track of charges.	Payment Integration	History lists transactions with date, plan, amount and status. SRS_NutriLift.docx
PY-NF-1.1	Payment processing must be secure and consistent with the gateway.	Payment Integration	All payment calls use HTTPS and handle success/failure without duplicate charges. SRS_NutriLift.docxppl-ai-file-upload.s3.amazonaws

11.2. Data Dictionary:

Entity: SUBSCRIPTION_PLAN

Attribute Name	Data Type	Description
plan_id	String	Subscription plan code.
name	String	Plan name.
description	Text	Description of the plan.
monthly_price	Float	Monthly subscription price.
yearly_price	Float	Yearly subscription price.
features	JSON	Included features for this plan.
is_active	Boolean	Indicates if plan is active.
created_at	Datetime	Creation time.

Entity: SUBSCRIPTION

Attribute Name	Data Type	Description
id	UUID	Subscription id.
user_id	UUID	References USER(id).
plan_id	String	References SUBSCRIPTION_PLAN(plan_id).

Attribute Name	Data Type	Description
subscription_type	String	app_premium or gym_membership.
gym_id	UUID	References GYM(id) for gym membership.
payment_gateway	String	Payment gateway name such as stripe or khalti.
transaction_id	String	Last payment gateway transaction id.
amount	Float	Recurring subscription amount.
currency	String	Currency code.
status	String	Status such as active, cancelled, expired.
starts_at	Datetime	Subscription start time.
expires_at	Datetime	Subscription expiry time.
created_at	Datetime	Creation time.

Entity: PAYMENT_TRANSACTION

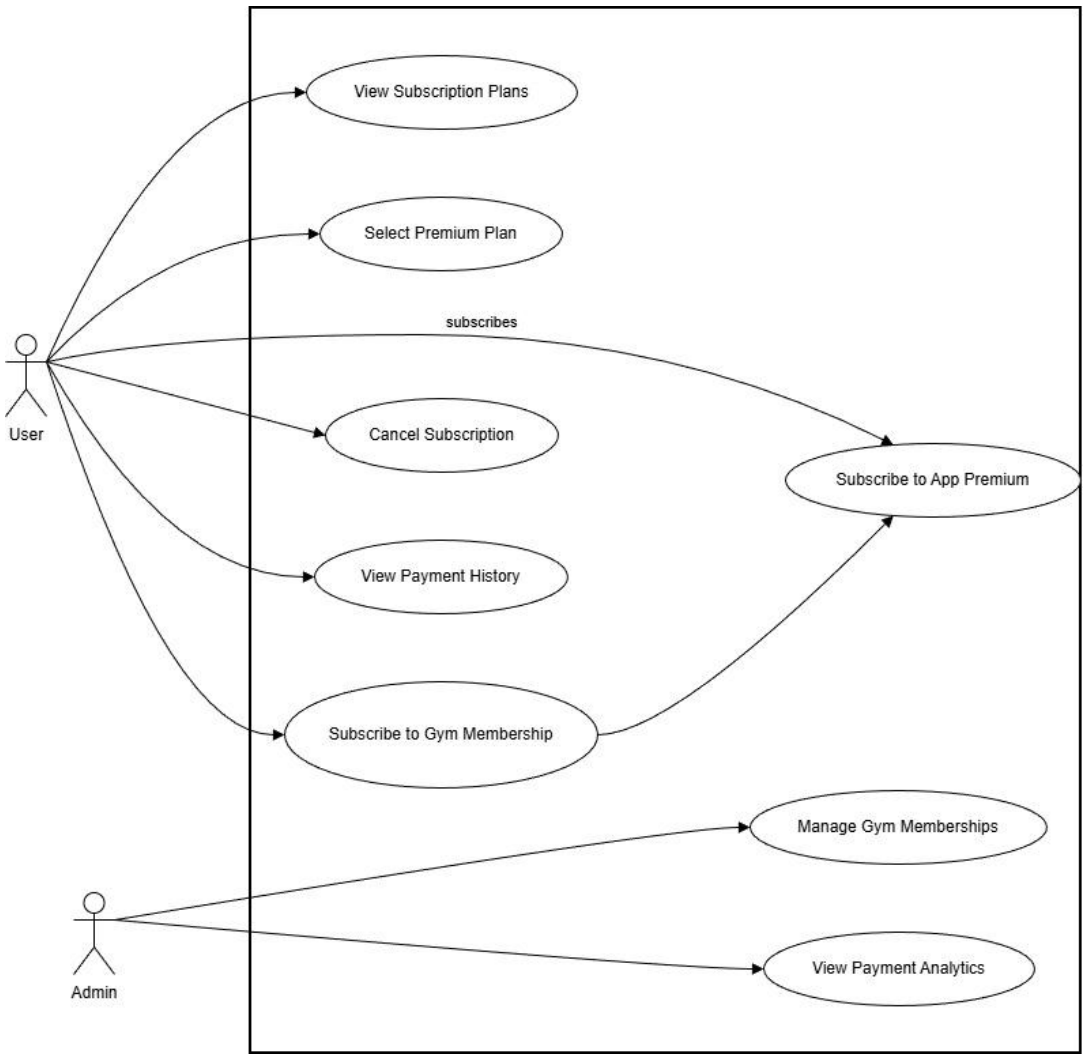
Attribute Name	Data Type	Description
id	UUID	Payment transaction id.
subscription_id	UUID	References SUBSCRIPTION(id).
gateway_transaction_id	String	Transaction id from payment gateway.
amount	Float	Amount charged.
currency	String	Currency of transaction.
status	String	pending, success or failed.
failure_reason	Text	Explanation if transaction failed.
processed_at	Datetime	Time when transaction was processed.
created_at	Datetime	Creation time.

Entity: PAYMENT_WEBHOOK

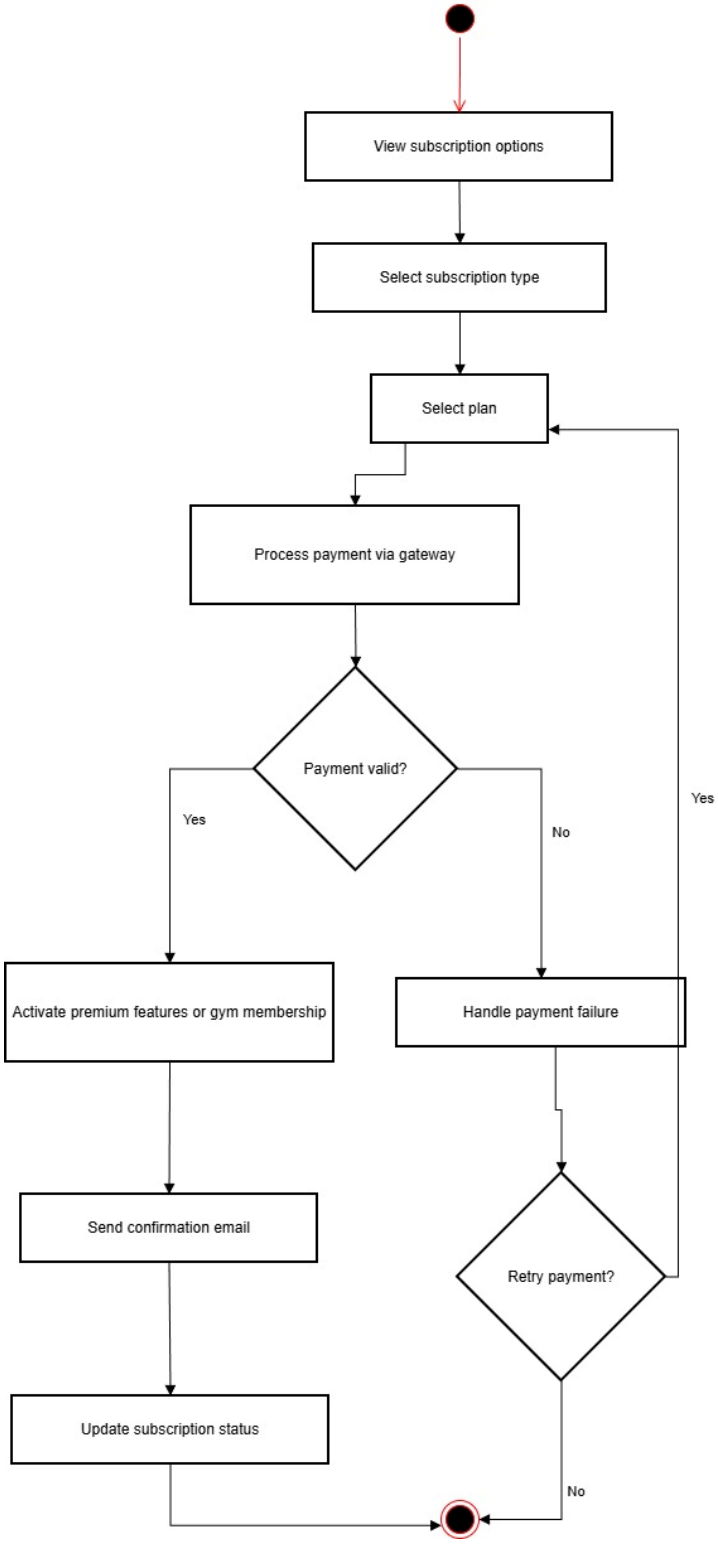
Attribute Name	Data Type	Description
id	UUID	Webhook event id.
gateway	String	Source payment gateway.

Attribute Name	Data Type	Description
event_type	String	Webhook event type.
payload	JSON	Raw event payload content.
processed	Boolean	Indicates whether event was handled.
received_at	Datetime	Time when webhook was received.

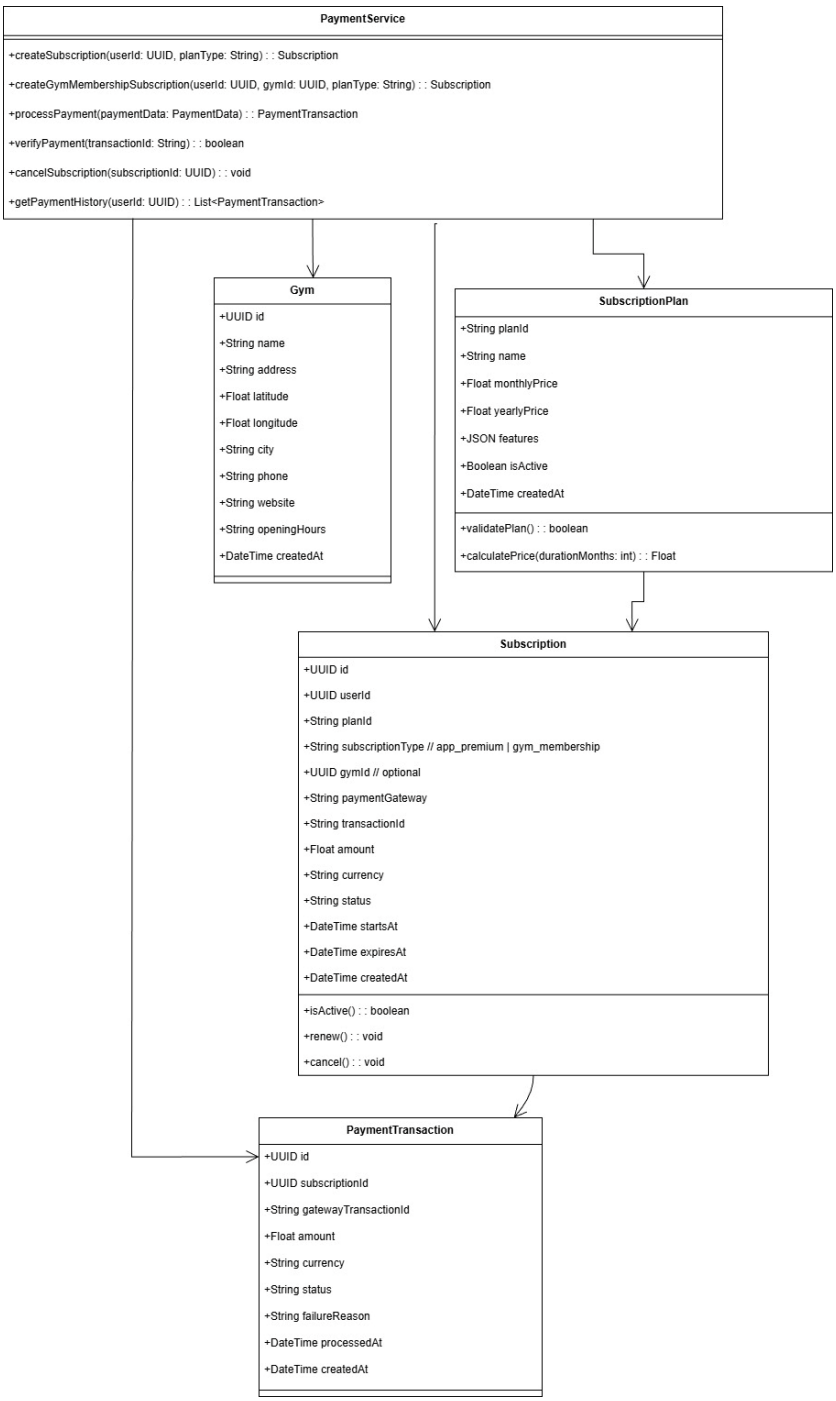
11.3. Diagrams:



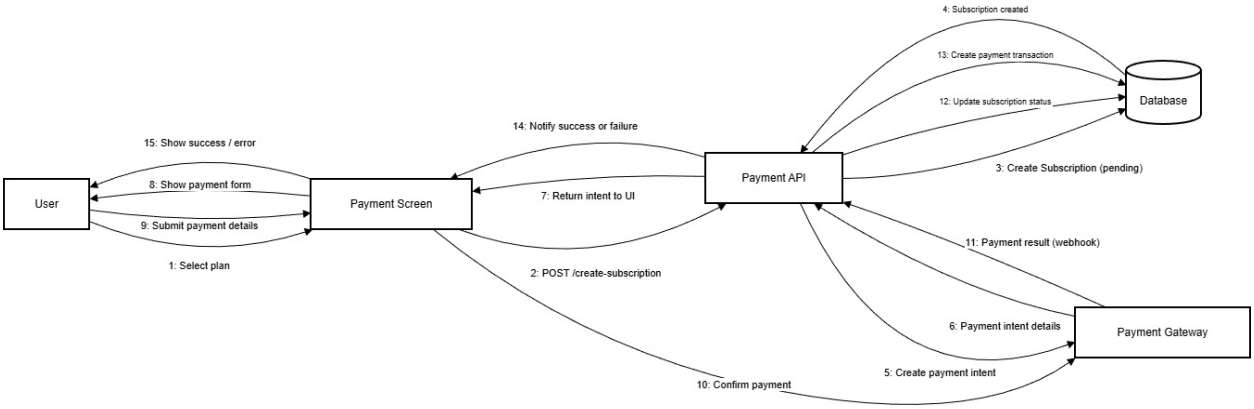
Payment Integration 1: Use Case Diagram



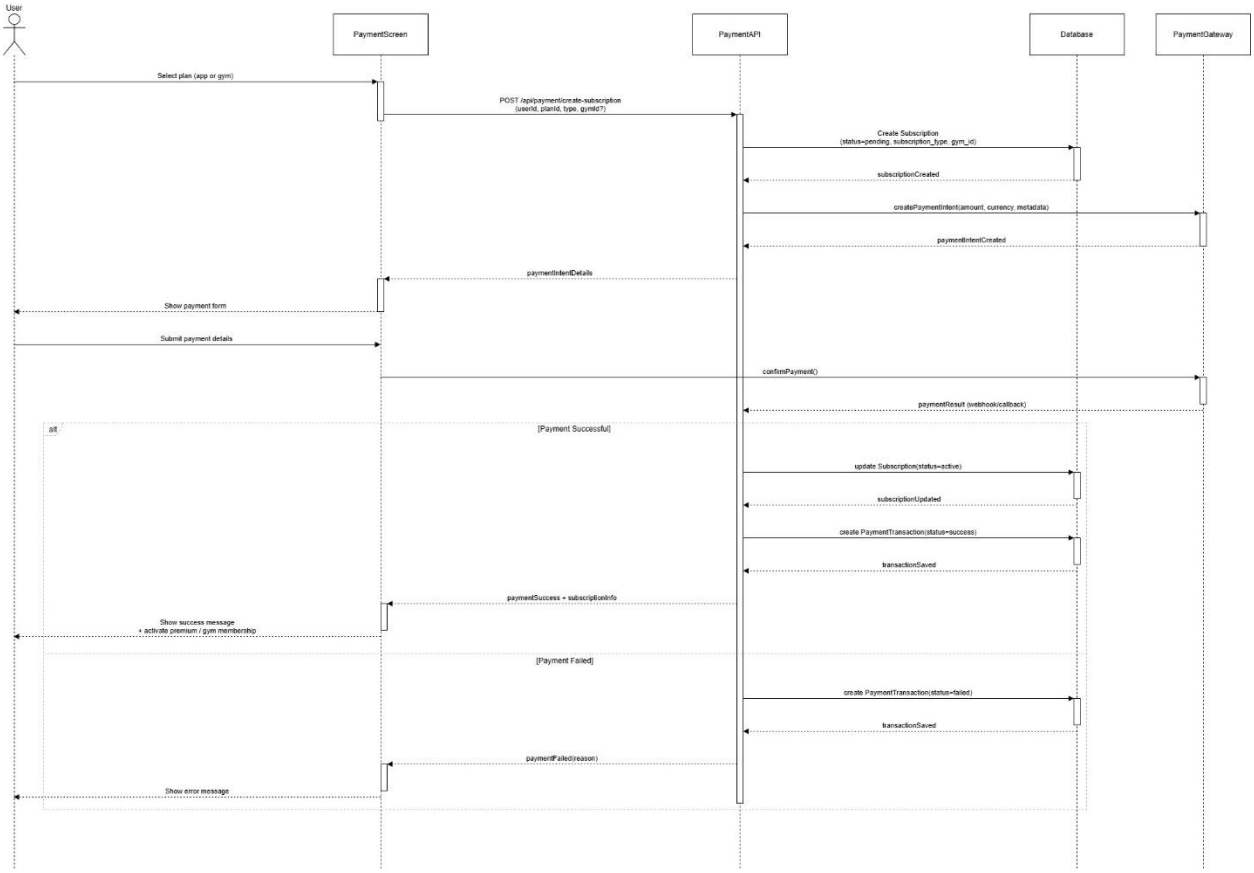
Payment Integration 2: Activity Diagram



Payment Integration 3: Class Diagram



Payment Integration 4: Collaboration Diagram



Payment Integration 5: Sequence Diagram

12. Gym Discovery and Membership

This subsystem helps users find gyms, check their details and join partner gyms from within the app. It also stores active and past memberships so users can see where they are currently enrolled.

12.1. SRS:

ID	User Story / Requirement	Subsystem	Acceptance Criteria
GYM-F-1.0	As a user, I want to find gyms near me so that I can pick a convenient one.	Gym Discovery & Membership	With location allowed, list shows nearby gyms sorted by distance. SRS_NutriLift.docx
GYM-F-2.0	As a user, I want to search gyms by city or name so that I can explore options.	Gym Discovery & Membership	Search bar filters gyms by typed city or name. SRS_NutriLift.docx
GYM-F-3.0	As a user, I want to view detailed information about a gym so that I can decide if it fits my needs.	Gym Discovery & Membership	Detail screen shows address, hours, contact, website and memberships. SRS_NutriLift.docx
GYM-F-4.0	As a user, I want to enroll in a partner gym from the app so that my membership is digital and linked to NutriLift.	Gym Discovery & Membership	After payment, a membership record is created and shown as active. SRS_NutriLift.docx
GYM-F-5.0	As a user, I want to see my active and past gym memberships so that I can manage them.	Gym Discovery & Membership	Membership list shows all memberships with status (active, cancelled, expired). SRS_NutriLift.docx

12.2. Data Dictionary:

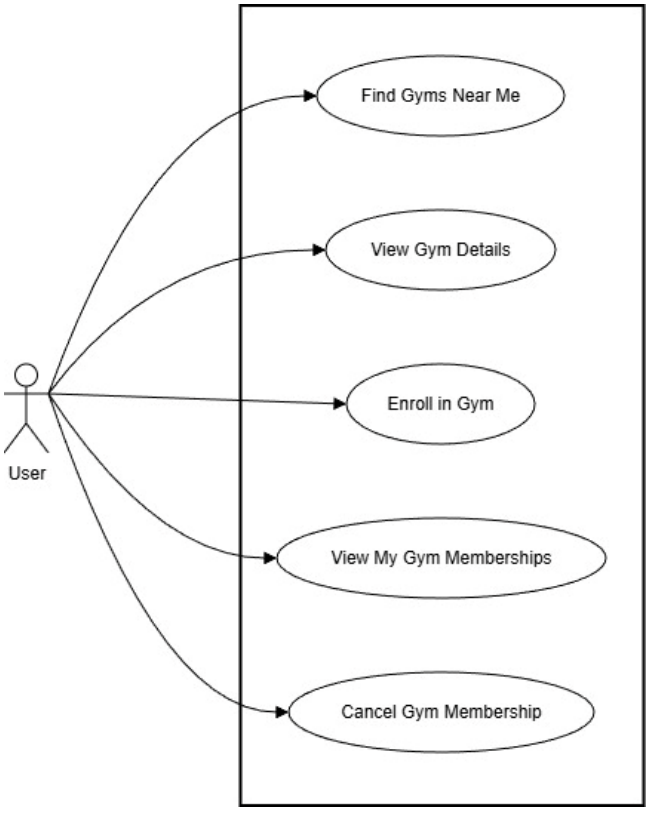
Attribute Name	Data Type	Description
id	UUID	Gym id.
name	String	Gym name.
address	String	Street address of the gym.
latitude	Float	GPS latitude.
longitude	Float	GPS longitude.
city	String	City of the gym.
phone	String	Contact phone number.
website	String	Website URL of the gym.
opening_hours	String	Opening hours information.
created_at	Datetime	Creation time.

Entity: GYM_MEMBERSHIP

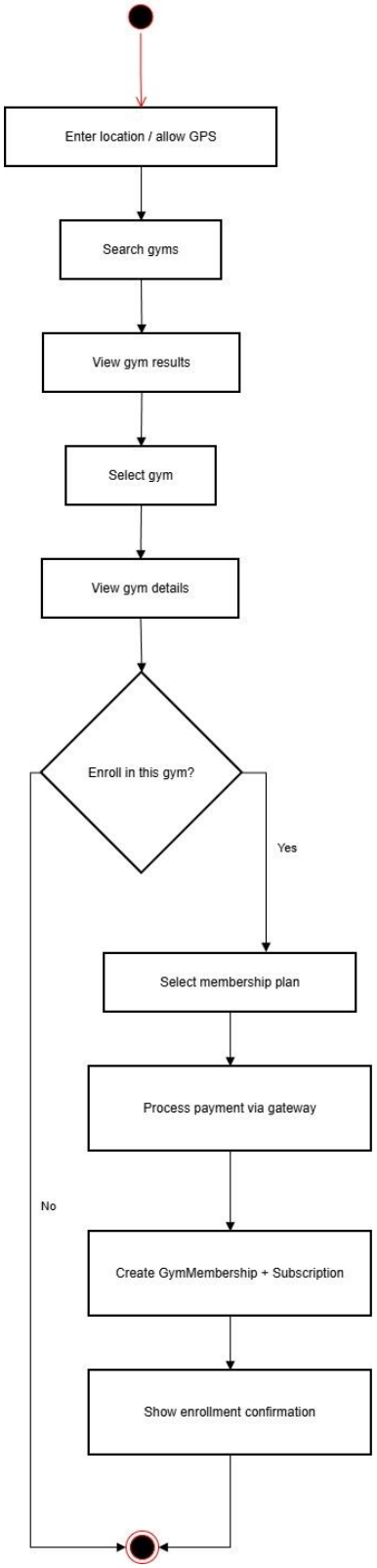
Attribute Name	Data Type	Description
id	UUID	Gym membership id.
user_id	UUID	References USER(id).

Attribute Name	Data Type	Description
gym_id	UUID	References GYM(id).
membership_type	String	Type such as monthly, yearly, day_pass.
start_date	Date	Membership start date.
end_date	Date	Membership end date.
status	String	active, cancelled or expired.
created_at	Datetime	Creation time.

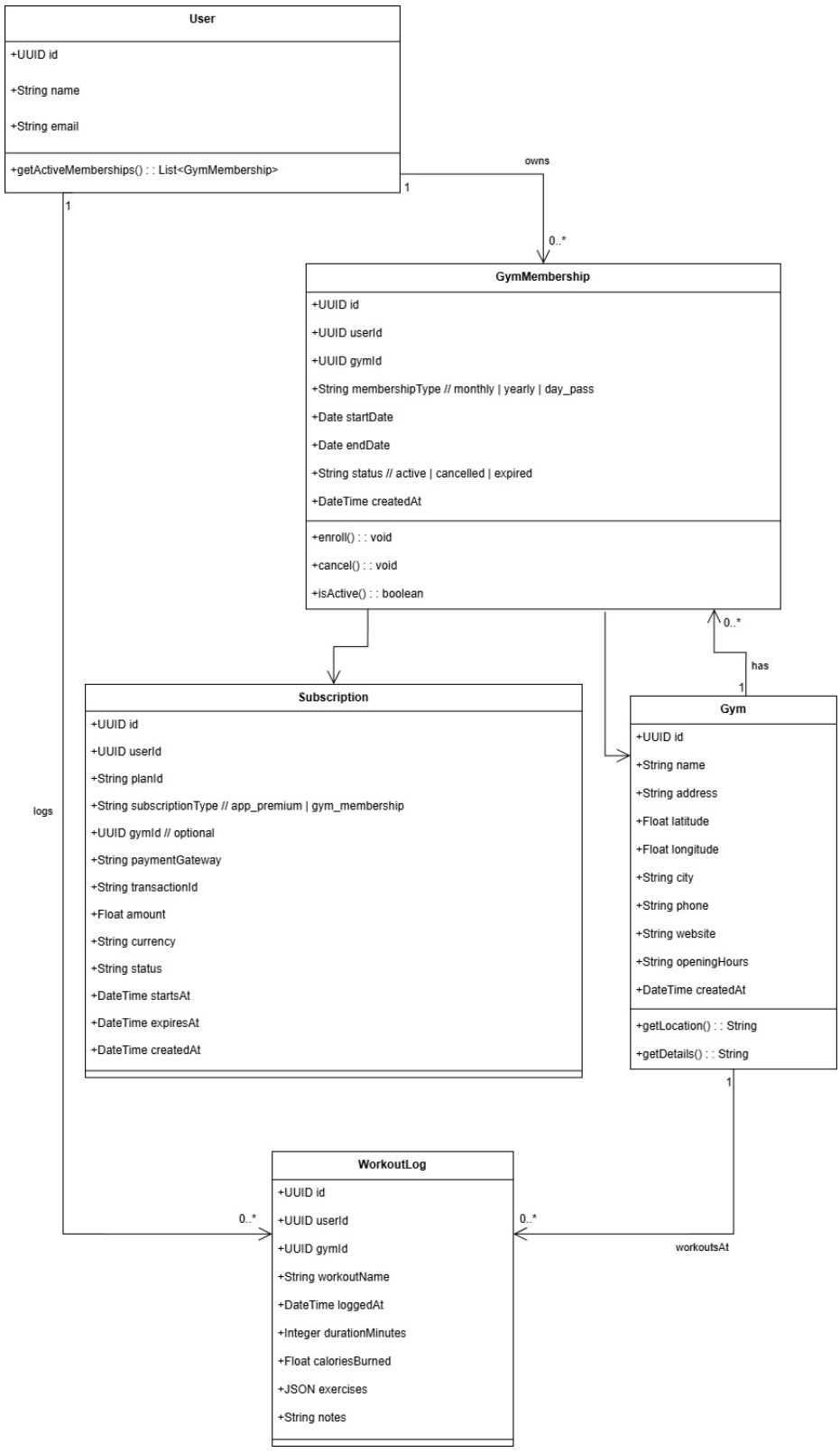
12.3. Diagrams:



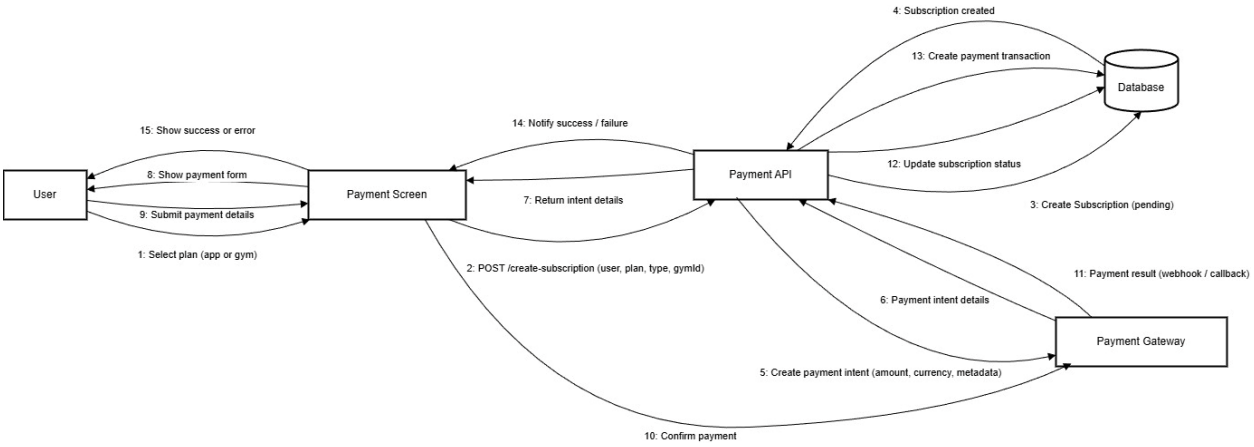
Gym Discovery 1: Use Case Diagram



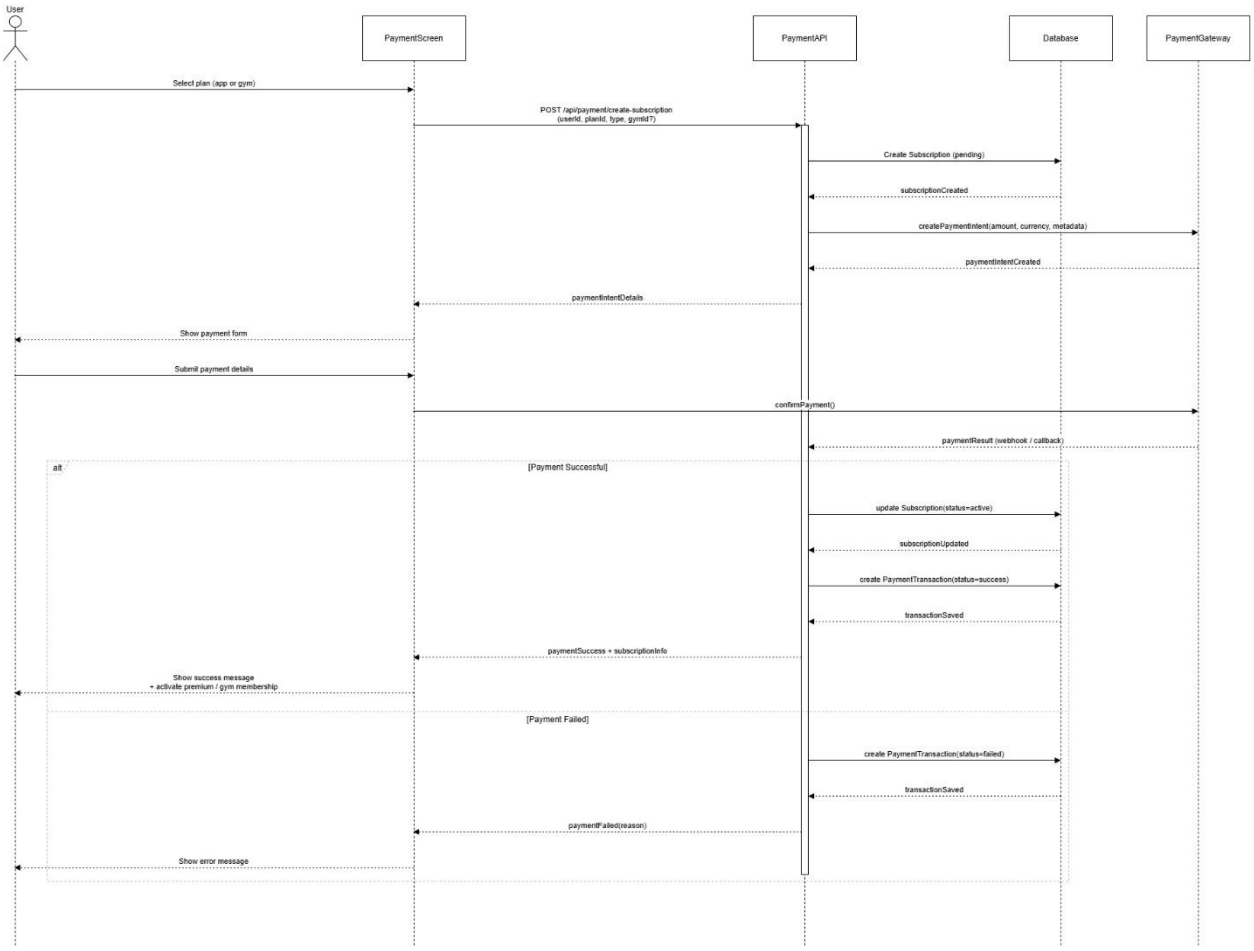
Gym Discovery 2: Activity Diagram



Gym Discovery 3: Class Diagram

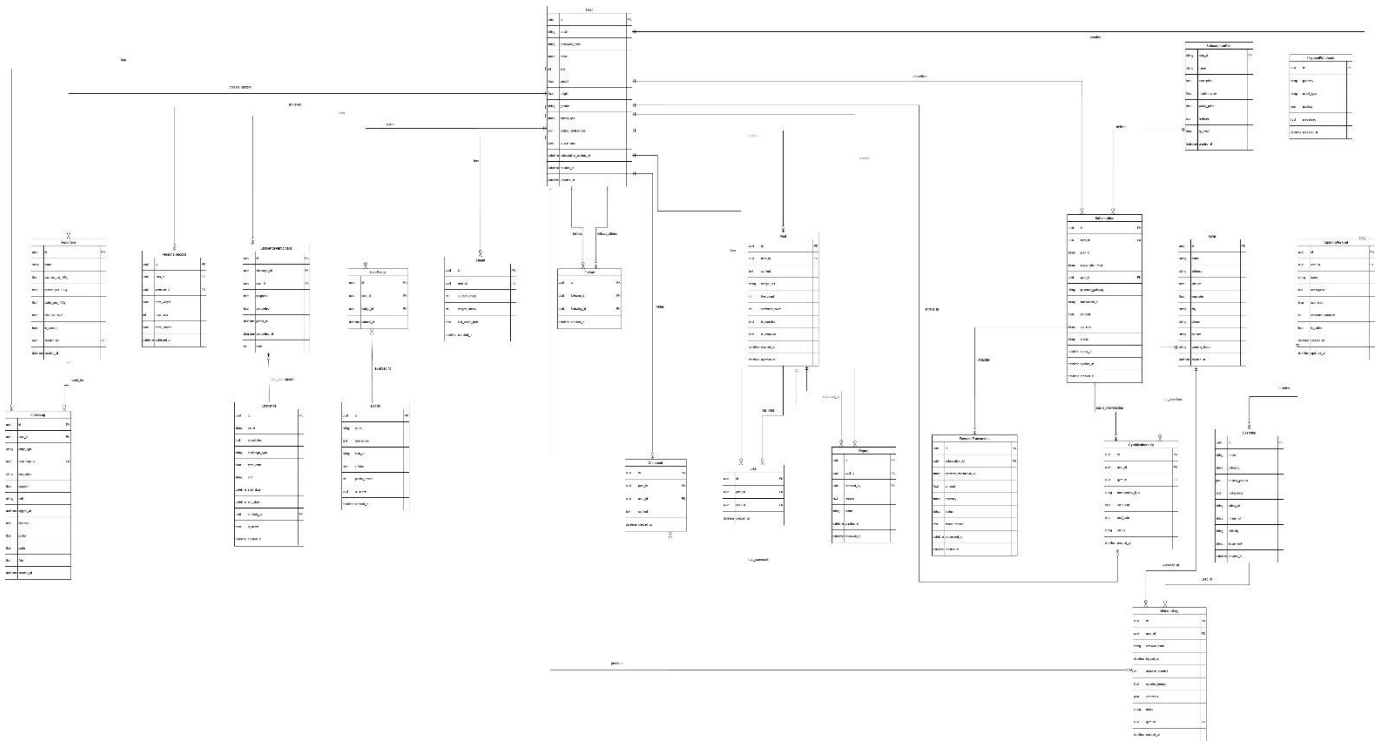


Gym Discovery 4: Collaboration Diagram



Gym Discovery 5: Sequence Diagram

13. ERD Diagram of whole system:



ERD Diagram 1

14. Test Plan:

This section outlines how the main features of NutriLift will be tested once the implementation is in place. The focus is on manual functional testing of core user journeys connected to a test backend. The table below lists representative test cases for each subsystem that will be used as a starting point during the development and integration phase. Detailed steps for each case are documented in the testing spreadsheet and can be expanded during implementation.

TC ID	Subsystem	Test Description	Preconditions	Main Steps	Expected Result
TC-UM-01	User Management	Create a new account using email and password.	App installed; backend running.	Open register screen, enter valid details, submit.	New user record is created and user is taken to the main dashboard.
TC-UM-02	User Management	Prevent a second account with the same email.	One account already exists for that email.	Try to register again with same email.	Registration is rejected and an appropriate error message is shown.
TC-UM-03	User Management	Log in and update profile data.	Existing user account.	Log in, open profile, change weight and goal, save.	Login is successful and new profile values are stored and displayed.

TC ID	Subsystem	Test Description	Preconditions	Main Steps	Expected Result
TC-NT-01	Nutrition Tracking	Log a meal using a food from the database.	Logged-in user; at least one food item available.	Add a meal with selected food and quantity.	Intake entry is stored and daily calorie and macro totals increase correctly.
TC-NT-02	Nutrition Tracking	Add a custom food and use it in a meal log.	Logged-in user.	Create custom food, then log a meal with it.	Custom food appears in search and behaves like other items when logged.
TC-NT-03	Nutrition Tracking	Record daily water intake.	Logged-in user.	Add one or more water entries.	Hydration total increases and progress towards the daily target is updated.
TC-WT-01	Workout Tracking	Log a workout containing multiple exercises.	Logged-in user; exercises present.	Add two exercises with sets and reps, save workout.	Workout is stored in history with all details.

TC ID	Subsystem	Test Description	Preconditions	Main Steps	Expected Result
TC-WT-02	Workout Tracking	Use a custom workout template.	Logged-in user.	Create a template and start a workout from it.	Template can be reused and generates a correct workout entry.
TC-RC-01	Rep Count	Count repetitions for a supported exercise.	Camera permission granted; good lighting.	Start a rep-count session, perform around ten clear reps.	Counter tracks repetitions with small error and saves the session.
TC-CG-01	Challenges & Gamification	Join a challenge and see progress update.	Active challenge exists.	Join challenge, then log relevant workouts/meals.	User appears as participant and progress bar increases.
TC-CM-01	Community	Create a post and interact with it.	Logged-in user; network available.	Publish a text+image post, like it and add a comment.	Post is visible in feed; like count and comments behave as expected.

TC ID	Subsystem	Test Description	Preconditions	Main Steps	Expected Result
TC-PR-01	Reports & Analytics	View a weekly summary and wellness score.	User has data for several days.	Open reports for the last week.	Charts and wellness score reflect the underlying logs.
TC-PY-01	Payment Integration	Complete a premium purchase in sandbox mode.	Logged-in user; gateway sandbox configured.	Select a plan, follow payment flow, return to app.	Subscription is marked active and premium features unlock.
TC-GYM-01	Gym Discovery & Membership	Find nearby gyms and enroll in one.	Location permission granted; gyms with coordinates exist.	Use “find gyms near me”, open a gym, choose membership, pay in sandbox.	List shows nearby gyms; after payment a membership record is created and visible in the membership screen.