

International School

**Group Project**

**CS 403 - Software Engineering**

**Software Requirement Specification Document**

**Version 1.5**

**Date:** 2025-10-02

**GMS - Gym Management System**

|  |  |  |
| --- | --- | --- |
| Thanh, Pham Huu Tien |  | - 28210248934 |
| Hung, Nguyen Do Khanh |  | - 28210254466 |
| Thanh, Le Huu |  | - 26211228082 |
| Van, Nguyen Tat |  | - 28210254721 |
| Tuyen, Pham Thi Thanh |  | - 28200254837 |

**Approved by**

**MSc Huy, Nguyen Dang Quang**

**Document Review Panel Representative:**

Name Signature Date

**Group Project - Mentor:**

Name Signature Date

**Da Nang 11/2025**

**PROJECT INFORMATION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Acronym** | GMS | | | |
| **Project Title** | GYM MANAGEMENT SYSTEM | | | |
| **Start Date** | 12 Sep 2025 | **End Date** | 15 Dec 2025 | |
| **Lead Institution** | International School, Duy Tan University | | | |
| **Role** | **Name** | **Email** | | **Phone** |
| **Project Mentor** | MSc Huy, Nguyen Dang Quang | huyndq@gmail.com | | 0935525354 |
| **Scrum & contact details** | Thanh, Pham Huu Tien | tienthanh711204@gmail.com | | 0328396428 |
| **Team Member** | Tuyen, Pham Thi Thanh | tp3224973@gmail.com | | 0865065202 |
| Van, Nguyen Tat | tatvannq@gmail.com | | 0843017452 |
| Hung, Nguyen Do Khanh | hungtraitk145@gmail.com | | 0833282777 |
| Thanh, Le Huu | huuthanhle75@gmail.com | | 0795571527 |

**SIGNATURE**

***Document Approvals:*** *The following signatures are required for approval of this document.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Mentor** | Huy, Nguyen Dang Quang | **Signature:** |  |
| **Date:** |  |
| **Date:** |  |
| **Team member(s)** | Tuyen, Pham Thi Thanh | **Signature:** |  |
| **Date:** |  |
| Van, Nguyen Tat | **Signature:** |  |
| **Date:** |  |
| Thanh, Le Huu | **Signature:** |  |
| **Date:** |  |
| Hung, Nguyen Do Khanh | **Signature:** |  |
| **Date:** |  |

REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Pham Thi Thanh Tuyen | 2025-09-18 | Created initial SRS draft: project scope, system overview, actors. | 1.0 |
| Le Huu Thanh | 2025-09-20 | Added Functional Requirements (FR) and Non-Functional Requirements (NFR). | 1.1 |
| All teams | 2025-09-25 | Wrote detailed Use Case descriptions and refined actor interactions. | 1.2 |
| All teams | 2025-09-28 | Added Use Case Diagram, Activity Diagrams for Registration and Booking. | 1.3 |
| Pham Huu Tien Thanh | 2025-10-01 | Integrated AI module requirements (Workout Plan, Nutrition Plan, Pose Recognition, Injury Risk Analysis). | 1.4 |
| All teams | 2025-11-02 | Final formatting and polishing for official submission. | 1.5 |

TABLE OF CONTENS

[**1. Introduction** 10](#_Toc215833668)

[1.1 Purpose of the Document 10](#_Toc215833669)

[1.2 Scope of the Document 10](#_Toc215833670)

[1.3 Definitions, Acronyms, Abbreviations 11](#_Toc215833671)

[1.4 References 11](#_Toc215833672)

[1.5 Overview 11](#_Toc215833673)

[**2.** **Project Overview** 11](#_Toc215833674)

[2.1 Purpose of the System 11](#_Toc215833675)

[2.2 Scope of the System 12](#_Toc215833676)

[**3.** **Overall Description** 12](#_Toc215833677)

[3.1 Product Perspective 12](#_Toc215833678)

[3.2 Product Functions 13](#_Toc215833679)

[3.3 User Characteristics 14](#_Toc215833680)

[3. 4 Constraints 14](#_Toc215833681)

[3.5 Assumptions and Dependencies 15](#_Toc215833682)

[**4.** **Specific Requirements** 16](#_Toc215833683)

[4.1 Functional Requirements (FR) 16](#_Toc215833684)

[4.1.1 User Authentication & Profile Management 16](#_Toc215833685)

[4.1.2 Membership Management 16](#_Toc215833686)

[4.1.3 Training Session Booking 16](#_Toc215833687)

[4.1.4 Progress Tracking and AI Coaching 17](#_Toc215833688)

[4.1.5 Pesonal Trainer (PT) Functionalities 17](#_Toc215833689)

[4.1.6 Manager Fuctionalities 17](#_Toc215833690)

[4.1.7 Support Staff Fuctionalities 18](#_Toc215833691)

[4.1.8 Payment Processing 18](#_Toc215833692)

[4.1.9 Notifications System 18](#_Toc215833693)

[4.2 Non-functional Requirements (NFR) 18](#_Toc215833694)

[4.2.1 Performance 18](#_Toc215833695)

[4.2.2 Security 19](#_Toc215833696)

[4.2.3 Availability 19](#_Toc215833697)

[4.2.4 Usability 19](#_Toc215833698)

[4.2.5 Scalability 20](#_Toc215833699)

[4.2.6 Maintainability & Extensibility Requirements 20](#_Toc215833700)

[4.2.7 Compatibility Requirements 20](#_Toc215833701)

[4.3 External Interface Requirements 20](#_Toc215833702)

[**5.** **System models** 22](#_Toc215833703)

[5.1 Use Case Diagram Overview 22](#_Toc215833704)

[5.2 Use Case Specification 23](#_Toc215833705)

[5.2.1 Use case UC-01 Register Account 23](#_Toc215833706)

[5.2.2 Use Case UC-02 Login/ Logout 26](#_Toc215833707)

[5.2.3 Use Case UC-03 Update Profile 29](#_Toc215833708)

[5.2.4 Use Case UC-04 View Notifications 32](#_Toc215833709)

[5.2.5 Use Case UC-05 View Membership Packages 35](#_Toc215833710)

[5.2.6 Use Case UC-06 Purchase/ Renew Membership 38](#_Toc215833711)

[5.2.7 Use Case UC-07 View Membership Status 42](#_Toc215833712)

[5.2.8 Use Case UC-08 View Payment History 45](#_Toc215833713)

[5.2.9 Use Case UC-09 Download Invoice 48](#_Toc215833714)

[5.2.10 View Trainer Availability 51](#_Toc215833715)

[5.2.11 Use case UC-11 Book Training Session 55](#_Toc215833716)

[5.2.12 Use Case UC-12 Cancel Training Session 58](#_Toc215833717)

[5.2.13 Use Case UC-13 Reschedule Training Session 61](#_Toc215833718)

[5.2.14 Use Case UC-14 Check-in via QR code 65](#_Toc215833719)

[5.2.15 Use Case UC-15 Validate QR Code 69](#_Toc215833720)

[5.2.16 Use Case UC-16 View Progress Charts 72](#_Toc215833721)

[5.2.17 Use Case UC-17 AI Workout Plan 76](#_Toc215833722)

[5.2.18 Use Case UC-18 AI Nutrition Plan 79](#_Toc215833723)

[5.2.19 Use Case UC-19 Pose Recognition 83](#_Toc215833724)

[5.2.20 Use Case UC-20 Injury Risk Analysis 88](#_Toc215833725)

[5.2.21 Use Case UC-21 Submit Support Ticket 92](#_Toc215833726)

[5.2.22 Use Case UC-22 Track ticket status 95](#_Toc215833727)

[5.2.23 Use Case UC-23 Live chat support 99](#_Toc215833728)

[5.2.24 Use Case UC-24 Update Availability 102](#_Toc215833729)

[5.2.25 Use Case UC-25 View Work Schedule 106](#_Toc215833730)

[5.2.26 Use Case UC-26 Accept/ Decline Booking Requests 109](#_Toc215833731)

[5.2.27 Use Case UC-27 Conduct Training Session 112](#_Toc215833732)

[5.2.28 Use Case UC-28 Record Training Notes 116](#_Toc215833733)

[5.2.29 Use Case UC-29 View Member Progress 120](#_Toc215833734)

[5.2.30 Use Case UC-30 Send Session Feedback 123](#_Toc215833735)

[5.2.31 Use Case UC-31 Manage Branches 127](#_Toc215833736)

[5.2.32 Use Case UC-32 Manage Membership Packages 131](#_Toc215833737)

[5.2.33 Use Case UC-33 Process Refund Requests 135](#_Toc215833738)

[5.2.34 Use Case UC-34 View Financial Reports 139](#_Toc215833739)

[5.2.35 Use Case UC-35 Trainer Performance Reports 143](#_Toc215833740)

[5.2.36 Use Case Member Activity Reports 146](#_Toc215833741)

[5.2.37 Use Case UC-37 Manage Staff Accounts 150](#_Toc215833742)

[5.2.38 Use Case UC-38 Respond to Tickets 153](#_Toc215833743)

[5.2.39 Use Case UC-39 Escalate Ticket 156](#_Toc215833744)

[5.2.40 Use Case UC-40 Provide Live Chat Support 159](#_Toc215833745)

[5.2.41 Use Case UC-41 Manage Knowledge Base Articles 162](#_Toc215833746)

[5.2.42 Use Case UC-42 View Support Analytics Dashboard 166](#_Toc215833747)

[5.2.43 Use Case UC-43 Process Online Payment 170](#_Toc215833748)

[5.2.44 Use Case UC-44 Confirm Offline Payment 174](#_Toc215833749)

[5.3 Activity Diagram 179](#_Toc215833750)

[5.4 Sequence Diagram 193](#_Toc215833751)

[5.5 Class Diagram 207](#_Toc215833752)

[**6.** **Other Requirements** 208](#_Toc215833753)

[6.1 Business Rule 208](#_Toc215833754)

[6.2 System Constraints 208](#_Toc215833755)

[**7.** **Prototype Design** 209](#_Toc215833756)

[7.1 Login and Authentication Interfaces 209](#_Toc215833757)

[7.2 Member-Side Prototypes 209](#_Toc215833758)

[7.2.1 Member Dashboard 209](#_Toc215833759)

[7.2.2 Membership Package Browser 210](#_Toc215833760)

[7.2.3 Payment and Invoice Screen 210](#_Toc215833761)

[7.2.4 Trainer Selection and Availability Screen 210](#_Toc215833762)

[7.2.5 Member Session Management UI 211](#_Toc215833763)

[7.2.6 Progress Tracking Dashboard 211](#_Toc215833764)

[7.2.7 AI Workout Plan Interface 211](#_Toc215833765)

[7.2.8 AI Nutrition Plan Interface 212](#_Toc215833766)

[7.2.9 Pose Recognition Interface 212](#_Toc215833767)

[7.2.10 Injury Risk Analysis Interface 212](#_Toc215833768)

[7.3 Personal Trainer Interface 212](#_Toc215833769)

[7.3.1 Trainer Dashboard 212](#_Toc215833770)

[7.3.2 Trainer Availability Management 212](#_Toc215833771)

[7.3.3 Session Processing Screen 213](#_Toc215833772)

[7.3.4 Member Progress Management 213](#_Toc215833773)

[7.4 Manager Interface 213](#_Toc215833774)

[7.4.1 Manager Dashboard 213](#_Toc215833775)

[7.4.2 Package Management UI 213](#_Toc215833776)

[7.4.3 Branch Management UI 214](#_Toc215833777)

[7.4.4 Report Generation UI 214](#_Toc215833778)

[7.5 Support Staff Interface 214](#_Toc215833779)

[7.5.1 Support Ticket Console 214](#_Toc215833780)

[7.5.2 Live Chat Console 214](#_Toc215833781)

[7.5.3 Knowledge Base Editor 215](#_Toc215833782)

[7.6 Global UI Elements 215](#_Toc215833783)

[7.6.1 Navigation Bar 215](#_Toc215833784)

[7.6.2 Notification Center 215](#_Toc215833785)

[7.6.3 User Profile Settings 215](#_Toc215833786)

[7.7 User Experience (UX) Principles 215](#_Toc215833787)

[**Appendix A: Glossary** 216](#_Toc215833788)

# Introduction

## Purpose of the Document

This Software Requirements Specification (SRS) describes the complete set of requirements for the Gym Management System (GMS).

It provides a precise, structured description of the system’s:

* Functional Requirements
* Non-functional Requirements
* External and internal interfaces
* System models (UML)
* User interactions and workflows
* Constraints and dependencies

The document intended for:

* Developers
* Test engineers
* Project managers
* System architects
* Customers and stakeholders

## Scope of the Document

This document covers the following elements:

* Functional Modules (Membership, Booking, Payment, AI, Support,…)
* User roles and behavior
* Interface anf performances requirements
* Data requirements
* System constraints
* UML modeling

Excluded:

* Implementation details
* Deployment configurations
* Internal project management documents

## Definitions, Acronyms, Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| GMS | Gym Management System |
| PT | Personal Trainer |
| FR | Functional Requirement |
| NFR | Non-functional Requirement |
| AI | Artificial Intelligence |
| UI | User Interface |
| DBMS | Database Management System |
| API | Application Programming Interface |

## References

1. Sommerville, I. *Software Engineering*, 10th Edition, Pearson Education, 2015, [*https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf*](https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf)

2. Atlassian, Scrum Framework Documentation, [*https://www.atlassian.com/agile/scrum*](https://www.atlassian.com/agile/scrum)

3. React Documentation, [*https://react.dev/*](https://react.dev/)

4. Google Gemini AI API Documentation, 2025,[*https://ai.google.dev/gemini-api/docs#python*](https://ai.google.dev/gemini-api/docs#python)

## 1.5 Overview

* Chapter 2 describes the project overview and system purpose.
* Chapter 3 covers the overall system description.
* Chapter 4 defines detailed functional and non-functional reuirements.
* Chapter 5 provides system models (UML).
* Chapter 6 includes business rules and constraints.
* Appendix A lists glossary terms.

# Project Overview

## 2.1 Purpose of the System

The Gym Management System (GMS) aims to:

* Digitize gym operations
* Enable online membership registration & payment
* Allow members to book PT sessions
* Provide AI-driven recommendations and pose correction
* Support trainers in managing schedules
* Assist managers with monitoring and reporting
* Provide effective customer support through a dedicated Support Staff module

## 2.2 Scope of the System

The GMS will provide:

* A web-based platform accessible to members, trainers, and managers
* User authentication & authorization
* Membership package management
* Payment & billing
* PT booking & scheduling
* Progress tracking & analytics
* AI workout, nutrition & pose analysis
* Support Staff ticketing & live chat
* Multi-branch gym management
* Administrative dashboards & reports

Out-of-scope:

* Hardware device integraton
* In-person check-in hardware terminal
* Mobile native app (optional future expansion)

# Overall Description

## 3.1 Product Perspective

The GMS is an independent software product with:

* Web-based architecture
* SQL Server database
* Integration with:
  + Payment gateway (PayOS)
  + Google Gemini API (API Services)
  + Email/ SMS notification gateway

It follows a 3-tiers architecture:

1. Presentation Layer (Next.js, Tailwind)
2. Backend Layer (PHP)
3. Data Layer (SQL Server)

## 3.2 Product Functions

The Gym Management System (GMS) provides a complete set of features for four main user roles: Member, Personal Trainer, Manager, and Support Staff.  
The system supports membership operations, training session booking, AI-based fitness services, customer support, branch management, and staff management.

Member Functions

* Register, log in, and manage personal profiles
* View membership packages
* Purchase or renew memberships (online/offline)
* View membership status
* Book, cancel, or reschedule training sessions
* Check in via QR code
* Track personal fitness progress
* Receive AI workout and nutrition plans
* Receive AI pose correction feedback
* Submit support tickets
* View ticket status
* Access live chat support
* View payment history
* Configure notification preferences

Personal Trainer Functions

* Log in and manage their personal profile
* Define and update availability schedules
* View assigned work schedules
* Accept or decline training session requests
* Conduct training sessions
* Record workout notes and session results
* View fitness progress of assigned members

Manager Functions

* Manage gym branches
* Manage membership packages
* Process refund requests
* Monitor session and trainer schedules
* Generate reports (financial, member activity, trainer performance)
* Handle escalated support tickets
* View audit logs
* Manage staff accounts (create/update/deactivate PT or Support Staff)

Support Staff Functions

* Receive and respond to support tickets
* Assign or escalate tickets to Manager
* Provide real-time live chat support
* Manage knowledge base articles (create/edit/publish/archive)
* View support performance analytics

## User Characteristics

|  |  |  |
| --- | --- | --- |
| **User** | **Experence Level** | **Description** |
| **Member** | Basic | General gym customers who use booking, membership, and AI features. |
| **PT** | Medium | Gym trainers who manage availability, conduct sessions, and track member progress. |
| **Manager** | High | Oversees gym operations, manages staff, handles finances, supports escalations, and has highest system privileges. |
| **Support Staff** | Medium | Customer support personnel responsible for ticket handling, live chat, and knowledge base maintenance. |

## 3. 4 Constraints

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Note** |
| 1 | The system must satisfy all functional and non-functional requirements defined in the User Stories, SRS, and Product Backlog. | Schedule |
| 2 | The project must be completed within 13.5 weeks (from 12 Sep 2025 to 15 Dec 2025) according to the approved project timeline. | Schedule |
| 3 | The system must operate as a web-based application, accessible through modern browsers (Chrome, Firefox, Edge, Safari). | Scope |
| 4 | The development must be carried out by a team of 5 members following the Scrum model. | Resources |
| 5 | The project must work within a financial estimation limit of $2430, calculated based on working hours and team resources. | Budget |
| 6 | The system must maintain high performance with a page load time not exceeding 3 seconds under normal load conditions. | Quality |
| 7 | The system must integrate with third-party services (Payment Gateway, AI API) within their technical constraints and service-level limitations. | External Dependency |
| 8 | The system must ensure data security and apply role-based access control for Member, Trainer, Manager, Support Staff roles. | Security |
| 9 | The application must use SQL Server as the primary database management system according to the approved architecture. | Technical |
| 10 | The system must comply with privacy policies when handling personal data such as health metrics, payment information, and user identities. | Compliance |

## 3.5 Assumptions and Dependencies

Assumptions:

* All users (Members, Trainers, Managers, Support Staff) have valid email addresses or phone numbers to receive notifications.
* Users have stable internet access when using the web application.
* Members provide accurate personal and health information when using AI-based workout and nutrition services.
* Trainers maintain accurate availability schedules to ensure proper session booking.
* Managers actively monitor system activities and approve updates such as membership packages and refunds.
* Support Staff are available within working hours to handle support tickets and live chat interactions.
* AI services (e.g., Google Gemini API) will return accurate and timely insights for workout, nutrition, and pose recognition modules.

Dependencies:

* The system depends on the availability and uptime of the external Payment Gateway (e.g., PayOS) for processing online transactions.
* The system depends on the Google Gemini API for AI-driven features, including workout recommendations and pose analysis.
* The system requires SQL Server to operate reliably as the central database.
* The application depends on third-party email/SMS providers for sending notifications and verification codes.
* The system relies on browser compatibility across commonly used web browsers (Chrome, Firefox, Edge, Safari).
* The system depends on the hosting server’s performance and uptime to ensure accessibility for all user roles.
* The accuracy of AI recommendations depends on the completeness and quality of user historical data stored in the system.

# Specific Requirements

## 4.1 Functional Requirements (FR)

### 4.1.1 User Authentication & Profile Management

* FR-01 The system shall allow users to register using email or phone number.
* FR-02 The system shall authenticate users using secure login credentials.
* FR-03 The system shall allow users to update personal profile information.
* FR-04 The system shall allow users to view system notifications.

### 4.1.2 Membership Management

* FR-05 The system shall display available membership packages including price, duration, and benefits.
* FR-06 The system shall allow members to purchase or renew gym membership packages online or offline.
* FR-07 The system shall allow members to view their membership status (active, expired).
* FR-08 The system shall allow members to view their payment transaction history.
* FR-09 The system shall allow members to download invoices for completed payments.

### 4.1.3 Training Session Booking

* FR-10 The system shall allow members to view Personal Trainer availability.
* FR-11 The system shall allow members to book training sessions only during available time slots.
* FR-12 The system shall allow members to cancel booked training sessions.
* FR-13 The system shall allow members to reschedule training sessions before the session start time.
* FR-14 The system shall allow members to check in to a training session using QR code.
* FR-15 The system shall validate QR code check-in requests with time-based and authenticity checks.

### 4.1.4 Progress Tracking and AI Coaching

* FR-16 The system shall allow members to view training progress and performance charts.
* FR-17 The system shall generate personalized AI-based workout plans.
* FR-18 The system shall generate personalized AI-based nutrition plans.
* FR-19 The system shall provide real-time pose recognition feedback using camera input.
* FR-20 The system shall analyze training data to provide injury-risk predictions.

### 4.1.5 Pesonal Trainer (PT) Functionalities

* FR-21 The system shall allow PTs to set and update their availability schedule.
* FR-22 The system shall allow PTs to view their training schedule.
* FR-23 The system shall allow PTs to accept or decline training session requests.
* FR-24 The system shall allow PTs to conduct training sessions.
* FR-25 The system shall allow PTs to record training session notes.
* FR-26 The system shall allow PTs to view assigned members’ progress data.
* FR-27 The system shall allow PTs to send feedback to members after training sessions.

### 4.1.6 Manager Fuctionalities

* FR-28 The system shall allow managers to manage gym branches (create, update, deactivate).
* FR-29 The system shall allow managers to manage membership packages.
* FR-30 The system shall allow managers to process refund requests.
* FR-31 The system shall allow managers to view financial reports.
* FR-32 The system shall allow managers to view trainer performance reports.
* FR-33 The system shall allow managers to view member activity and attendance reports.
* FR-34 The system shall allow managers to manage staff accounts (Personal Trainers and Support Staff).

### 4.1.7 Support Staff Fuctionalities

* FR-35 The system shall allow Support Staff to receive and respond to support tickets submitted by members.
* FR-36 The system shall allow Support Staff to escalate unresolved tickets to managers.
* FR-37 The system shall provide Support Staff with a real-time live chat interface.
* FR-38 The system shall allow Support Staff to create, update, and manage Knowledge Base articles.
* FR-39 The system shall provide Support Staff with support analytics reports (ticket volume, response rate, SLA performance).

### 4.1.8 Payment Processing

* FR-40 The system shall process online payments using an integrated payment gateway.
* FR-41 The system shall allow authorized staff to confirm offline payments.
* FR-42 The system shall store all payment and refund transactions with timestamps.
* FR-43 The system shall automatically generate invoices for successful payments.

### 4.1.9 Notifications System

* FR-44 The system shall send notifications via email, SMS, or in-app messages.
* FR-45 The system shall allow users to configure their notification preferences.
* FR-46 The system shall send automated reminders for upcoming sessions, expiring memberships, and incomplete payments.

## 4.2 Non-functional Requirements (NFR)

### 4.2.1 Performance

* NFR-01 The system shall support at least 200 online payment transactions per hour without degradation in performance.
* NFR-02 The system shall maintain a page load time below 3 seconds under normal usage conditions.
* NFR-03 The booking availability response time shall not exceed 1 second.
* NFR-04 The system shall support up to 500 concurrent users performing standard operations (login, booking, viewing schedules, etc.).
* NFR-05 The system shall process AI-based workout and nutrition recommendations within 5 seconds.

### 4.2.2 Security

* NFR-06 All personal and financial data stored in the database shall be encrypted using AES-256 or an equivalent industry-standard encryption method.
* NFR-07 All communication between clients and servers shall use HTTPS with TLS 1.2 or above.
* NFR-08 User accounts shall be automatically locked after 5 consecutive failed login attempts.
* NFR-09 The system shall store passwords using salted hashing algorithms (BCrypt, Argon2, or equivalent).
* NFR-10 The system shall maintain audit logs for critical operations, including:
  + Staff management updates
  + Refund processing
  + Payment confirmation
  + Branch management
  + Membership package edits
* NFR-11 The system shall restrict access to administrative functions based on role-based access control (RBAC).

### 4.2.3 Availability

* NFR-12 System uptime shall be at least 99.5% annually, excluding scheduled maintenance.
* NFR-13 The system shall automatically recover after unexpected crashes without data loss, using transactional integrity controls.
* NFR-14 Daily automated backups must be created and stored for at least 30 days.
* NFR-15 The system shall handle partial payment failures and ensure no inconsistent financial states.

### 4.2.4 Usability

* NFR-16 The system shall support onboarding new gym branches without requiring architectural changes.
* NFR-17 The system shall scale horizontally to serve increasing traffic loads.
* NFR-18 The system database shall support indexing and partitioning for large-scale datasets (members, trainers, sessions, transactions).

### 4.2.5 Scalability

* NFR-19 The system shall provide a user-friendly interface accessible to non-technical users.
* NFR-20 The UI shall be responsive and compatible with desktop, tablet, and mobile devices.
* NFR-21 The system shall provide visual feedback for all user actions (loading, error, confirmation).
* NFR-22 The system shall comply with accessibility standards (color contrast, readable typography, keyboard navigation).

### 4.2.6 Maintainability & Extensibility Requirements

* NFR-23 The system shall be modular to support future upgrades, including AI enhancements or new features.
* NFR-24 The system shall use clean, maintainable architecture enabling updates with minimal downtime.
* NFR-25 Developer documentation, API documentation, and deployment instructions shall be maintained and updated.
* NFR-26 The system shall support logging and monitoring tools (e.g., ELK stack, Prometheus, or similar).

### 4.2.7 Compatibility Requirements

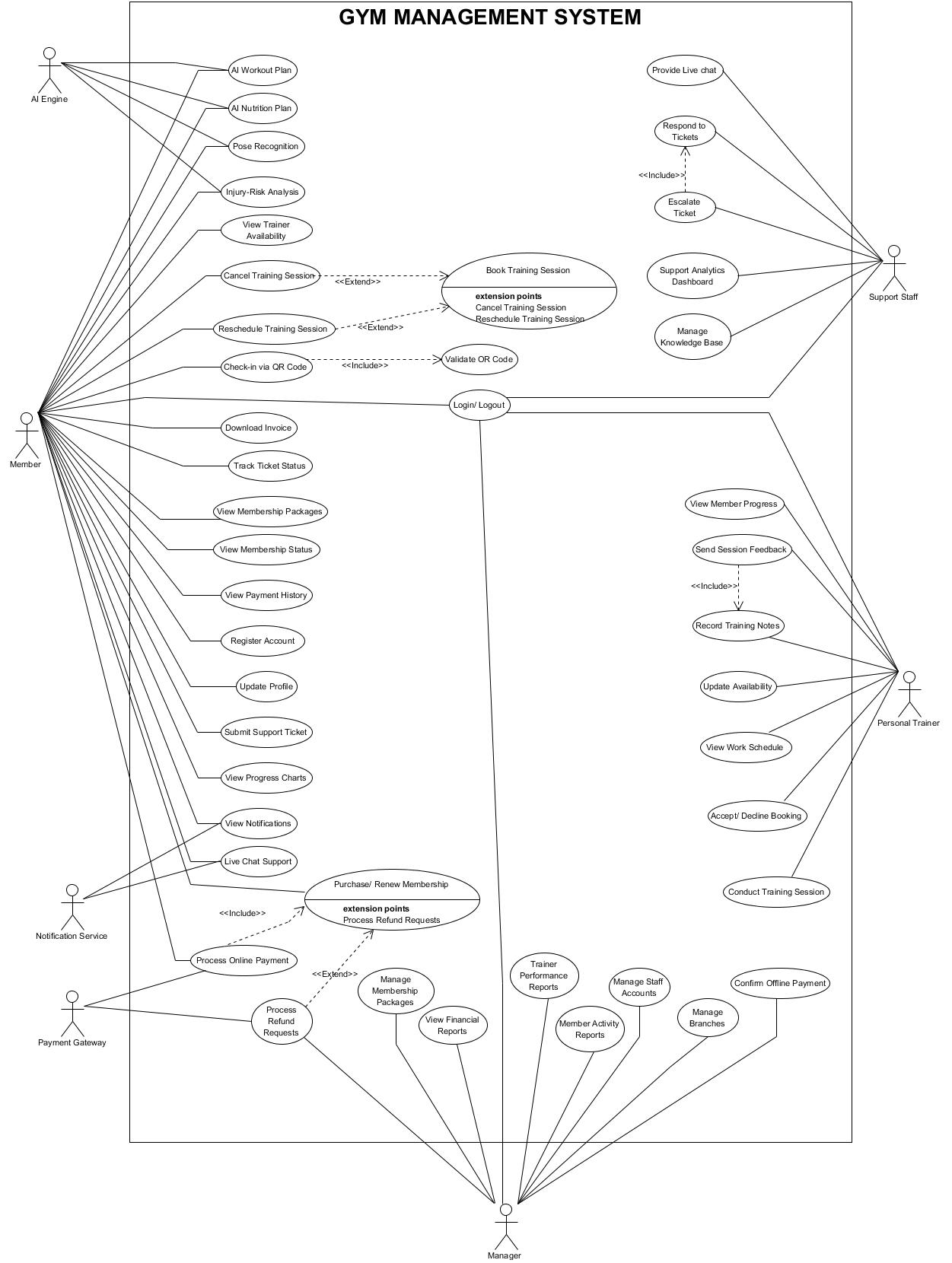
* NFR-27 The system shall run on modern web browsers (Chrome, Firefox, Safari, Edge).
* NFR-28 The application shall be deployable on both Windows Server and Linux Server environments.
* NFR-29 The system shall support deployment on web servers such as: IIS, Apache Tomat, Nginx (reverse proxy)

## 4.3 External Interface Requirements

* User Interface:
  + Web responsive (mobile/desktop/tablet)
  + Clean, modern UI
* AI API Interface: Gemini API for
  + Workout pla generation
  + Nutrition plan creation
  + Pose recognition
* Payment Gateway: PayOS API
* Database: SQL Server 2019+
* Email/ SMS Gateway: SMTP/ SMS Provide

# System models

## 5.1 Use Case Diagram Overview



## 5.2 Use Case Specification

### 5.2.1 Use case UC-01 Register Account

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-01 | | |
| **Use Case Name** | Register Account | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes the process in which a new user creates an account in the Gym Management System (GMS). The user submits registration information, and the system validates and stores the account in the Authentication System and User Database. | | |
| **Goal** | To successfully register a new member account that is valid, unique, and stored securely in the system. | | |
| **Trigger** | The user opens the “Sign Up” page from the login screen or system homepage. | | |
| **Pre – conditions** | * The Gym Management System is online and accessible. * The user does not already have an existing registered account. | | |
| **Post – conditions** | Success Post-Conditions   * A new user account is created and stored in the Authentication System. * User profile data is saved in the Membership subsystem. * A confirmation email/SMS (if applicable) is sent to the user. * User account state is set to “Pending Verification”.   Failure Post-Condidtions   * No account is created. * Error message is shown. * System logs the failed registration attempt. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User navigates to the “Register” page. | System displays the registration form containing fields: Name, Email, Phone, Password. |
| **2** | User enters valid registration information. | System performs client-side validation (format check for email, phone, password). |
| **3** | User clicks “Register”. | System sends registration data to the Authentication System for account creation. |
| **4** |  | Authentication System hashes the password and creates a secure user record. |
| **5** |  | Membership System creates a new profile with status Pending Verification. |
| **6** | System displays success confirmation. | System redirects user to login page and sends verification message (email/SMS). |
| **Alternative Flow** | **Alternative Flow: Invalid or Duplicated Information** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User enters invalid or duplicate email/phone. |  |
| **2** | User clicks “Register”. | Authentication System detects invalid format or existing account. |
| **3** |  | System displays message: *"Email or phone already in use."* or *"Password must be at least 8 characters including uppercase, lowercase, and number."* |
| **4** | User corrects the input. | System reloads form with previously entered data retained. |
| **5** | User clicks “Register” again. | System re-submits and re-validate (go to Step 3 of Main Flow). |
| **Exception Flow** | **Exception Flow: Authentication System Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User enters data and clicks “Register”. |  |
| **2** |  | System fails to connect to Authentication System. |
| **3** |  | System displays: *"Registration failed. Please try again later."* |
| **4** | User retries later or contacts support. | System logs error for administrative review. |
| **Priority** | High | | |
| **Business Rule** | * Password must be at least 8 characters with uppercase, lowercase, and a number. * Email and phone must be unique. * Account status is "Pending Email Verification" until confirmed. | | |

### 5.2.2 Use Case UC-02 Login/ Logout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID** | UC-02 | | | |
| **Use Case Name** | Login / Logout | | | |
| **Created by** | Thanh Pham | | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member, Personal Trainer, Manager, Support Staff | | | |
| **Brief Description** | This use case describes how users securely authenticate into the Gym Management System using email/phone and password. It also defines how users log out of the system. The Authentication System validates credentials, manages sessions, and updates login status. | | | |
| **Goal** | To authenticate a user and establish a secure session for system access, or to properly terminate an active user session during logout. | | | |
| **Trigger** | The user selects “Login” from the home screen or attempts to access a protected feature. | | | |
| **Pre – conditions** | * The user has a valid, registered account in the system. * The Authentication System is operational. * For logout: The user must already be logged in. | | | |
| **Post – conditions** | Success Post-Conditions   * A secure authenticated session is created for the user. * User role (Member/PT/Manager/Support Staff) is identified and stored in session context. * For logout: session is terminated safely.   Failure Post-Condidtions   * User remains unauthenticated. * No changes are made to the system. * Error message is shown. * Failed login attempt is logged. | | | |
| **Main Flow** | **Main Flow 1: Login** | | | |
| **Step** | **Actor Action** | | **System Respond** |
| **1** | User navigates to the Login page. | | System displays input fields: Email/Phone, Password. |
| **2** | User enters login credentials. | | System performs client-side field validation. |
| **3** | User clicks “Login”. | | System sends credentials to Authentication System. |
| **4** |  | | Authentication System verifies credentials. |
| **5** |  | | Authentication System identifies user role (Member/PT/Manager/Support Staff). |
| **6** |  | | Authentication successful → System creates user session. |
| **7** | System displays dashboard matching the user’s role. | | User is now authenticated. |
| **Main Flow 2: Logout** | | | |
| **Step** | **Actor Action** | | **System Respond** |
| **1** | User clicks “Logout”. | | System clears current session data. |
| **2** |  | | User is redirected to Login page. |
| **Alternative Flow** | **Alternative Flow 1: Incorrect Credentials** | | | |
| **Step** | **Actor Action** | | **System Response** |
| **1** | Users submits invalid credentials. | | Authentication System detects mismatch. |
| **2** |  | | System displays: *“Incorrect email/phone or password.”* |
| **3** | User retries login. | | Flow returns to Main Flow Step 2. |
| **Alternative Flow 2: Account Locked** | | | |
| **Step** | **Actor Action** | | **System Respond** |
| **1** | User enters incorrect password multiple times. | | Authentication System locks the account after 5 failed attempts. |
| **2** |  | | System displays: *“Your account is locked. Please reset your password.”* |
| **3** | User clicks *“Forgot Password”* (if available). | | Redirects to password reset flow (out of scope). |
| **Exception Flow** | **Exception Flow 1: Authentication System Down** | | | |
| **Step** | **Actor Action** | | **System Response** |
| **1** | User attempts to log in. | |  |
| **2** |  | | System cannot communicate with Authentication System. |
| **3** |  | | System shows: *“Login service unavailable. Please try again later.”* |
| **4** | User retries later. | | Failure is logged. |
| **Priority** | High | | | |
| **Business Rule** | * Passwords must be stored using salted hashing (e.g., Argon2 or BCrypt). * Account is locked after 5 failed login attempts. * Session expires automatically after 30 minutes of inactivity. | | | |

### 5.2.3 Use Case UC-03 Update Profile

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-03 | | |
| **Use Case Name** | Update Profile | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member, Personal Trainer, Manager, Support Staff | | |
| **Brief Description** | This use case describes how a user updates their personal information (name, email, phone number, profile picture, etc.) within the Gym Management System. The system validates the new data and updates it in the user profile database. | | |
| **Goal** | To allow users to modify and maintain accurate personal information within their account. | | |
| **Trigger** | The user selects “Edit Profile” from the Profile or Account Settings page. | | |
| **Pre – conditions** | * User must be successfully authenticated (logged in). * User profile data must exist in the system. | | |
| **Post – conditions** | Success Post-Conditions   * Updated profile information is stored in the database. * System reflects the updated information immediately. * User receives a confirmation message.   Failure Post-Condidtions   * No profile data is changed. * System shows an appropriate error message. * System logs the failed update attempt. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User navigates to the Profile or Account Settings page. | System retrieves and displays the current profile information. |
| **2** | User edits one or more profile fields. | System enables “Save Changes” button. |
| **3** | User clicks “Save Changes”. | System validates updated fields (format, unique constraints). |
| **4** |  | System updates profile information in the database. |
| **5** |  | System displays success message: *“Profile updated successfully.”* |
| **6** | User views updated profile details. | System refreshes profile page with new data. |
| **Alternative Flow** | **Alternative Flow 1: Invalid Input** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User enters invalid format (e.g., wrong phone/email). |  |
| **2** | User attempts to save. | System performs validation and detects invalid format. |
| **3** |  | System displays error message: *“Invalid email format.”* or *“Phone number must contain digits only.”* |
| **4** | User corrects the data and submits again. | Returns to Main Flow Step 3. |
| **Alternative Flow 2: Email/Phone Already Exists** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | User enters a new email/phone already used by another account. |  |
| **2** | User clicks “Save Changes”. | System checks for data uniqueness. |
| **3** |  | System displays: *“Email/Phone already exists.”* |
| **4** | User revises the input. | Flow returns to Main Flow Step 3. |
| **Exception Flow** | **Exception Flow: Profile Update Service Unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User submits updated information. |  |
| **2** |  | System fails to connect to Profile Management database. |
| **3** |  | System displays: *“Unable to update profile. Please try again later.”* |
| **4** | User retries later. | Failure is logged. |
| **Priority** | Medium | | |
| **Business Rule** | * Email and phone number must be unique across all user accounts. * Email must follow standard email format. * Phone number must contain digits only. * Profile photo (if uploaded) must not exceed configured file size. | | |

### 5.2.4 Use Case UC-04 View Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-04 | | |
| **Use Case Name** | View Notifications | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member, Personal Trainer, Manager, Support Staff | | |
| **Brief Description** | This use case describes how users view system notifications such as payment confirmations, booking updates, training reminders, AI feedback alerts, ticket replies, or system announcements. Notifications may be generated by the GMS, by the Notification Service, or automatically by system events. | | |
| **Goal** | To allow users to conveniently access all system-generated and event-generated notifications in a centralized interface. | | |
| **Trigger** | User selects “Notifications” or taps the notification bell icon. | | |
| **Pre – conditions** | * User is authenticated. * Notification records exist for the user (optional: empty list still valid). * Notification Service is operational for externally pushed messages. | | |
| **Post – conditions** | Success Post-Conditions   * Notifications are retrieved and displayed in the UI. * User may mark notifications as read. * Notification read status is updated in the database.   Failure Post-Condidtions   * No notification data is updated. * Error message displayed if retrieval fails. * System logs failure event. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User opens the Notifications pannel/ page. | System retrieves all unread and read notifications for the user. |
| **2** |  | System sorts notifications by timestamp (newest first). |
| **3** |  | System displays notifications with icon, title, short message, timestamp, and read/ unread state. |
| **4** | User selects a notification. | System opens full notification details (if applicable). |
| **5** | User marks notification as “Read”. | System updates read status in the notification database. |
| **6** |  |  |
| **Alternative Flow** | **Alternative Flow 1: No Notifications Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User opens notifications. | System retrieves an empty list. |
| **2** |  | System displays: *“No notifications yet”.* |
| **Alternative Flow 2: User Filters Notifications** | | |
| **Step** | **Actor Action** | **System Respond** |
|  | User selects a filter (e.g., “Unread only”, “System alerts”, “Payment updates”). |  |
|  |  | System filters and displays matching notifications. |
| **Exception Flow** | **Exception Flow 1: Notification Service Unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User opens the notification list. |  |
| **2** |  | System fails to retrieve lastest pushed notifications. |
| **3** |  | System displays: *“Unable to load notifications. Please try again later.”* |
| **4** | User retries. | Failure logged. |
| **Exception Flow 2: Database Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Users attempts to load notifications. |  |
| **2** |  | Database error occurs during query. |
| **3** |  | System displays: *“Error loading notifications.”* |
| **4** |  | System logs the exception. |
| **Priority** | Medium | | |
| **Business Rule** | * Notifications must include timestamp and type (system, payment, booking, support). * Unread notifications must be visually emphasized. * Notification content must not contain sensitive information (e.g., passwords). | | |

### 5.2.5 Use Case UC-05 View Membership Packages

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-05 | | |
| **Use Case Name** | View Membership Packages | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member views the list of available membership packages offered by the Gym Management System (monthly, quarterly, yearly plans, prices, benefits). The system retrieves package data configured by the Manager and displays it in a user-friendly interface. | | |
| **Goal** | To allow members to browse and understand available gym membership options so they can select a package for purchase or renewal. | | |
| **Trigger** | The user selects “Membership Packages” from the home screen or the membership page. | | |
| **Pre – conditions** | * User is authenticated in the system. * At least one membership package exists in the database. | | |
| **Post – conditions** | Success Post-Conditions   * System displays all available membership package details. * User may proceed to purchase a package (UC-06).   Failure Post-Condidtions   * No package information is displayed. * Error is logged by the system. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User navigates to the “Membership Packages” page. | System retrieves list of active membership packages from database. |
| **2** |  | System displays packages including: name, duration, price, description, benefits. |
| **3** | User selects a package for more details (optional). | System displays full package details. |
| **4** | User decides whether to purchase/renew. | If user clicks “Purchase”, system transitions to UC-06. |
| **Alternative Flow** | **Alternative Flow 1: No Packages Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User opens Membership Packages page. | System retrieves an empty dataset. |
| **2** |  | System displays: *“No membership packages are currently available.”* |
| **Alternative Flow 2: Package Filtering (Optional Feature)** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | User applies filters (e.g., duration, price range). | System filters the membership packages. |
| **2** |  | System displays results based on applied filters. |
| **Exception Flow** | **Exception Flow 1: Database Connection Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User attempts to load packages. |  |
| **2** |  | System cannot retrieve package data. |
| **3** |  | System displays: *“Unable to load packages.Please try again later.”* |
| **4** |  | System logs the database error. |
| **Exception Flow 2: Package Data Corrupted / Invalid** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System attempts to load packages data. |  |
| **2** |  | System detects missing or invalid fields. |
| **3** |  | System hides corrupted packages and logs internal data issue. |
| **Priority** | High | | |
| **Business Rule** | * Only active membership packages are displayed to the user. * Package information must include duration, price, and benefits. * Manager is the only actor allowed to create/update membership packages (UC-32). | | |

### 5.2.6 Use Case UC-06 Purchase/ Renew Membership

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-06 | | |
| **Use Case Name** | Purchase / Renew Membership | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member purchases a new membership plan or renews an existing one. The user selects a package, makes a payment (online or offline), and the system updates membership status after successful payment. Online payments are performed through a Payment Gateway. | | |
| **Goal** | To allow Members to purchase or renew gym memberships in a secure and efficient manner. | | |
| **Trigger** | User clicks “Purchase” or “Renew Membership” on the Membership Packages page (UC-05). | | |
| **Pre – conditions** | * User must be logged in. * Membership packages must exist and be active. * Payment Gateway is operational (for online payments). * User has selected a specific membership package. | | |
| **Post – conditions** | Success Post-Conditions   * Membership status is updated (Active or Renewed). * Payment transaction is recorded. * Invoice is generated. * A confirmation notification is sent to the Member.   Failure Post-Condidtions   * Membership remains unchanged. * Payment is not recorded. * Failed transaction may be logged or reversed. * User receives an error message. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User selects a membership package from UC-05 | System displays package details and price summary. |
| **2** | User selects payment method (Online/ Offline). | System loads payment options accordingly. |
| **3** | User confirms purchase/ renewal | System prepares transaction details. |
| **4** | User proceeds with selected payment method. | For online: UC-43 Process Online Payment (<<include>>). For offline: waiting for staff confirmation. |
| **5** |  | Upon payment success, system updates membership status and validity dates. |
| **6** |  | System generates invoice and stores it (UC-09). |
| **7** |  | Notification Service sends confirmation message. |
| **8** | System shows purchase summary. | User views updated membership status. |
| **Alternative Flow** | **Alternative Flow 1: User Cancels During Payment** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User cancels payment on gateway. | System receives cancellation event. |
| **2** |  | System displays: *“Payment canceled.”* |
|  |  | No changes to membership. |
| **Alternative Flow 2: Offline Payment Flow** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | User selects “Offline Payment”. | System marks transaction as Pending. |
| **2** | Manager or Staff confirms payment (UC-44). | System updates membership status. |
| **3** |  | System sends “Payment confirmed” notification. |
| **Alternative Flow 3: Payment Gateway Timeout** | | |
| **Step** | **Actor Actions** | **Sytem Respond** |
| **1** | User submits payment. | Payment Gateway does not respond in time. |
| **2** |  | System dispalys: *“Transaction timeout. Please try again.’* |
| **Exception Flow** | **Exception Flow 1: Payment Failed** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User attempts online payment. |  |
| **2** |  | Payment Gateway returns failure (insufficient funds, declined card). |
| **3** |  | System displays: *“Payment failed.”* |
| **4** |  | No membership changes occur. |
| **5** |  | System logs failed transaction. |
| **Exception Flow 2: Refund Required** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System detects internal error after successful payment. |  |
| **2** |  | System triggers UC-33 Process Refund Requests (<<extend>>). |
| **3** | Manager approves refund. | Payment reversed through Payment Gateway. |
| **Exception Flow 3: Database Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | User attempts to purchase or renew. |  |
| **2** |  | System fails to save transaction or membership update. |
| **3** |  | System displays: *“Unable to complete purchase.”* |
| **4** |  | Refund may be triggered if necessary. |
| **Priority** | High | | |
| **Business Rule** | * Only active membership packages may be purchased or renewed. * Payment must be confirmed before activating membership. * Online payment processing must comply with PCI DSS rules. * Offline payments must be manually verified by Manager or Staff. * Membership renewal extends from current end date, not from purchase date. | | |

### 5.2.7 Use Case UC-07 View Membership Status

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-07 | | |
| **Use Case Name** | View Membership Status | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member views their current membership status, validity dates, upcoming expiration, and any associated package details. It allows the Member to track whether their membership is Active, Expired, or Nearing Expiration. | | |
| **Goal** | To allow Members to check their current membership status and renewal information. | | |
| **Trigger** | The Member selects “Membership Status” from the dashboard or membership menu. | | |
| **Pre – conditions** | * The user must be authenticated. * The user must have at least one membership record (active or expired). * Membership data must be available in the system. | | |
| **Post – conditions** | Success Post-Conditions   * Membership status, validity dates, and package details are displayed to the user. * System may display recommendations for renewal if applicable.   Failure Post-Condidtions   * No data is shown. * System logs a data retrieval error if applicable. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens the “Membership Status” page. | System retrieves user’s membership records. |
| **2** |  | System identifies the current membership (active or most recent expired). |
| **3** |  | System displays:   * Membership package * Start date * End date * Status (Active/ Expired) * Days remaining (if active) |
| **4** | Member reviews the information. | System may show a “Renew Now” button if membership is expired or expiring soon. |
| **Alternative Flow** | **Alternative Flow 1: User Has No Active or Previous Membership** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User opens membership status page. | System finds no membership records. |
| **2** |  | System displays: *“You do not have an active membership. Please purchase one.”* |
| **3** | User selects “View Packages”. | System transitions to UC-05. |
| **Alternative Flow 2: Membership Expiring Soon** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | User views status with less than configured threshold days (e.g., <7 days). | System highlights expiration warning. |
| **2** |  | System displays: *“Your membership is expiring soon”.* |
| **Exception Flow** | **Exception Flow 1: Membership Data Retrieval Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User open Membership Status page. |  |
| **2** |  | System fails to retrieve membership data from database. |
| **3** |  | System displays: *“Unable to load membership status.”* |
| **4** |  | System logs database exception. |
| **Priority** | High | | |
| **Business Rule** | * Membership status can only be: Active or Expired. * Active membership validity is determined by start date + purchased duration. * Membership renewal date depends on UC-06 rules (renewal extends from current expiration date). * Notification reminders for expiration must comply with system notification rules (handled outside UC-07). | | |

### 5.2.8 Use Case UC-08 View Payment History

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-08 | | |
| **Use Case Name** | View Payment History | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member views a list of their payment transactions, including details such as payment date, method, amount, status, and invoice availability. It enables the user to track previous membership purchases or renewals. | | |
| **Goal** | To provide Members with a clear, accurate, and complete overview of all their past payments for transparency and financial record-keeping. | | |
| **Trigger** | The user selects “Payment History” from the profile or membership section. | | |
| **Pre – conditions** | * User is authenticated. * At least one payment record exists (empty list still allowable). * Payment records are stored in the GMS transaction database. | | |
| **Post – conditions** | Success Post-Conditions   * Payment history is displayed. * User can open details of each transaction. * User can download associated invoices (link to UC-09)   Failure Post-Condidtions   * No data is changed. * Error message shown if system failure occurs. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “Payment History”. | System retrieves all past payment records for the member. |
| **2** |  | System displays list of payments including:   * Transaction ID * Date * Amount * Payment method (online/ offline) * Status (Success/ Failed/ Pending). * Invoice download option. |
| **3** | Member selects a specific payment. | System displays: full transaction details. |
| **4** | Member chooses “Download Invoice”. | System transitions to UC-09 Download Invoice |
| **Alternative Flow** | **Alternative Flow 1: No Payment Records Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens payment history. | System retrieves an empty dataset. |
| **2** |  | System displays: *“No payment records found”* |
| **3** | User selects “Purchase Membership”. | System transactions to UC-06 |
| **Alternative Flow 2: User Filters Payment History** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | User selects filters (data range, payment method, status). | System applies selected filters. |
| **2** |  | System displays filtered payment list. |
| **Exception Flow** | **Exception Flow 1: Database Read Error** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to load payment history. |  |
| **2** |  | Database query fails. |
| **3** |  | System displays: *“Unable to load payment history. Please try again later”.* |
| **4** |  | System logs error event. |
| **Exception Flow 2: Payment Gateway Data Mismatch** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System attempts to verify reference data from Payment Gateway. |  |
| **2** |  | Payment Gateway fails to respond or returns mismatched data. |
| **3** |  | System displays: *“Some payment information could not be verified.”* |
|  | **4** |  | System marks transaction details with a warning icon. |
| **Priority** | Medium | | |
| **Business Rule** | * All completed payments must have an associated transaction record. * Only successful transactions may have invoices available. * Payment records must be immutable (cannot be modified by users). * Failed or canceled transactions must also be shown for clarity. | | |

### 5.2.9 Use Case UC-09 Download Invoice

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-09 | | |
| **Use Case Name** | Download Invoice | | |
| **Created by** | Thanh Pham | **Last updated by** | Thanh Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member downloads an invoice for a completed payment. The invoice is generated automatically after a successful membership purchase or renewal (UC-06) and can be accessed from the Payment History (UC-08). | | |
| **Goal** | To allow Members to download a formal invoice for their completed transactions for personal, financial, or reimbursement purposes. | | |
| **Trigger** | The Member selects “Download Invoice” from the Payment History screen or the transaction details page. | | |
| **Pre – conditions** | * User must be authenticated. * A successful payment trasaction exists. * An invoice has been generated and stored (PDF or digital format). * Transaction status must be Success. | | |
| **Post – conditions** | Success Post-Conditions   * Invoice file is downloaded to the user’s device. * System logs a “download invoice” event for security and auditing. * No changes are made to payment data.   Failure Post-Condidtions   * Invoice is not downloaded. * System may show an error message. * System logs failure event if applicable. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | User open a payment record in UC-08 | System displays payment details. |
| **2** | User clicks “Download Invoice”. | System locates associated invoice Billing Module. |
| **3** |  | System generates a secure download link |
| **4** |  | System starts the invoice download (PDF or supported format). |
| **5** | User opens or saves the file. | Process completes. |
| **Alternative Flow** | **Alternative Flow 1: Invoice Already Downloaded** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User clicks “Download Invoice”. | System shows a note: “This invoice was previously downloaded on <date>.” |
| **2** | User continues. | System starts download. |
| **Alternative Flow 2: Invoice Preview** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | User selects “Preview Invoice” | System renders invoice in browser or modal window. |
| **2** | User clicks “Download”. | Flow continues from Main Flow step 3. |
| **Exception Flow** | **Exception Flow 1: Invoice not found** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | User clicks “Download Invoice” |  |
| **2** |  | System cannot locate the invoice file. |
| **3** |  | System shows: *“Invoice not available.”* |
| **4** |  | System logs missing invoice error. |
| **Exception Flow 2: Payment not completed** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | User attempts to download invoice. | System checks transaction status. |
| **2** |  | System determines payment is not successful (Failed/Pending/Canceled). |
| **3** |  | System displays: *“Invoice is only available for completed payments.”* |
| **Exception Flow 3: File download failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | User triggers download. |  |
| **2** |  | System encounters file transmission or server error. |
| **3** |  | System displays: *“Unable to download invoice. Please try again.”* |
| **4** |  | Event logged. |
| **Priority** | Medium | | |
| **Business Rule** | * Invoices must be generated only after successful transactions. * Invoice format must be unmodifiable (read-only PDF). * Invoice format must be unmodifiable (read-only PDF). * Invoice file must be securely stored and accessible only by the invoice owner and authorized personnel. | | |

### 5.2.10 View Trainer Availability

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-10 | | |
| **Use Case Name** | View Trainer Availability | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member views available training time slots for Personal Trainers (PTs). The system retrieves each PT’s availability schedule, including training hours, booked sessions, and free slots. Members use this information to initiate UC-11 (Book Training Session). | | |
| **Goal** | To allow Members to see accurate and real-time availability of Personal Trainers so they can book training sessions without conflicts. | | |
| **Trigger** | Member selects “Trainer Availability” or navigates from the booking section. | | |
| **Pre – conditions** | * Member is authenticated. * At least one PT exists in the system. * PTs have defined their availability using UC-24 (Update Availability). * Scheduling data is synchronized and up to date. | | |
| **Post – conditions** | Success Post-Conditions   * Member sees list of PTs along with their available time slots. * Member may initiate UC-11 to book a session.   Failure Post-Condidtions   * Display error message if availability data cannot be retrieved. * No booking actions can proceed. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member selects “View Trainer Availability”. | System retrieves PT availability schedules. |
| **2** |  | System checks PT availability, existing bookings, and blocked time slots. |
| **3** |  | System displays PT list with:   * Name and photo * Specialty * Available time slots |
| **4** | Member selects a PT. | Sytem shows detailed availability calendar fow what PT (daily/ weekly view). |
| **5** | Member chooses a desired time slot. | System prepares data for booking and transitions to UC-11 Book Training Session. |
| **Alternative Flow** | **Alternative Flow 1: No availability found** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member selects a PT. | System finds no avilable time slots. |
| **2** |  | System displays: *“This trainer has no available time slots.”* |
| **3** | Member selects another PT. | System repeats main flow. |
| **Alternative Flow 2: Filtering PTs** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member applies filters (specialty, gender, time range, rating). | System filters PT list. |
| **2** |  | System displays filtered results. |
| **Alternative Flow 3: Date Navigation** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member navigates to next week or specific date. | System updates availability calendar. |
| **Exception Flow** | **Exception Flow 1: Scheduling Service Unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to view availability. |  |
| **2** |  | System cannot retrieve scheduling data. |
| **3** |  | System displays: *“Unable to load trainer availability.”* |
| **4** |  | System logs service failure. |
| **Exception Flow 2: Data Inconsistency (e.g., double-booked slots)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System detects conflicts due to inconsistent data. |  |
| **2** |  | Conflict slots are hidden or marked invalid. |
| **3** |  | System notifies administrator. |
| **Priority** | High | | |
| **Business Rule** | * Only time slots not previously booked should be shown as available. * PT availability comes exclusively from UC-24 Update Availability. * System must update availability in real time to prevent double booking. * Member cannot view or book PTs from other branches unless multi-branch permission is granted (future rule). | | |

### 5.2.11 Use case UC-11 Book Training Session

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-11 | | |
| **Use Case Name** | Book Training Session | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member books a training session with a Personal Trainer. After viewing PT availability (UC-10), the Member selects a time slot, the system validates the booking (checking conflicts, membership status), and sends the booking request to the PT for confirmation. | | |
| **Goal** | To allow Members to request a training session with a Personal Trainer at a specific time slot. | | |
| **Trigger** | The Member selects an available time slot shown in UC-10 (View Trainer Availability). | | |
| **Pre – conditions** | * Member is authenticated. * Member must have an active membership (UC-07). * Trainer availability must exist (UC-24). * Selected time slot is not already booked. * Scheduling service is operational. | | |
| **Post – conditions** | Success Post-Conditions   * A booking request is created and stored. * Trainer is notified of the request. * Member sees booking in “Pending” status.   Failure Post-Condidtions   * No booking is created. * Member receives error message. * System logs the failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member selects an available time slot from UC-10. | System loads booking details (PT, date, time). |
| **2** | Member confirms booking request. | System validates membership status and time slot availability. |
| **3** |  | System creates a booking entry with Pending status. |
| **4** |  | System sends notification to the PT. |
| **5** |  | System displays booking confirmation message. |
| **6** | Member sees booking request status as Pending Approval. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Tries to Book Without Active Membership.** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member selects a time slot. | System checks membership status. |
| **2** |  | System displays: *“You must have an active membership to book a session.”* |
| **3** |  | System suggests renewing membership (UC-06). |
| **Alternative Flow 2: Member Selects an Already-Booked Time Slot** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member selects time slot. | System detects conflicting booking. |
| **2** |  | System displays: *“This time slot is no longer available.”* |
| **3** |  | System prompts user to choose another slot. |
| **Exception Flow** | **Exception Flow 1: Scheduling System Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to submit booking. | Scheduling service cannot save the booking. |
| **2** |  | System displays: *“Unable to create booking. Please try later*.” |
| **3** |  | Error logged. |
| **Exception Flow 2: Trainer Profile or Availability Missing** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member selects a time slot. | System attempts to load trainer data. |
| **2** |  | System cannot find availability or trainer info. |
| **3** |  | System shows: *“Trainer unavailable.”* |
| **Priority** | High | | |
| **Business Rule** | * Member must have active membership to book a session. * Each time slot can be booked by only one Member. * PT must confirm the booking via UC-26 unless auto-accept is enabled. * Booking requests must include timestamp, PT ID, Member ID, and session type (if applicable). | | |

### 5.2.12 Use Case UC-12 Cancel Training Session

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-12 | | |
| **Use Case Name** | Cancel Training Session | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member cancels a previously booked or confirmed training session. The system validates cancellation rules (time limits, membership status, session type) and updates the booking status accordingly. The cancellation may notify the PT and free the time slot for others to book. | | |
| **Goal** | To allow Members to cancel their training sessions in a timely manner, updating the schedule and notifying the PT. | | |
| **Trigger** | The Member selects an existing session from their training schedule and clicks “Cancel Session.” | | |
| **Pre – conditions** | * Member is authenticated. * Member has at least one booked or confirmed training session. * Session is eligible for cancellation (based on business rules). * Scheduling System is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Training session status is updated to Canceled. * PT is notified of cancellation. * The time slot becomes available for rebooking.   Failure Post-Condidtions   * No changes are made to the session. * System displays an error message. * Failure is logged. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens their training schedule. | System displays list of scheduled sessions. |
| **2** | Member selects one session to cancel. | System loads session details (date, time, PT). |
| **3** | Member clicks “Cancel Session.” | System asks for confirmation. |
| **4** | Member confirms cancellation. | System validates cancellation rules (time restrictions, status). |
| **5** |  | System updates session status to Canceled. |
| **6** |  | System frees the time slot in the Scheduling System. |
| **7** |  | System notifies the PT about the cancellation. |
| **8** |  | System displays confirmation message. |
| **9** | Member sees updated schedule. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Cancels a Session Too Late** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to cancel. | System checks cancellation window. |
| **2** |  | System displays: *“This session cannot be canceled within 30 minutes of start time.”* |
| **3** | Member acknowledges message. | System returns to schedule. |
| **Alternative Flow 2: Session Already Canceled / Completed** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member select a session. | System detects status = Canceled or Completd. |
| **2** |  | System displays: *“This session cannot be canceled.”* |
| **Alternative Flow 3: Member decides not to cancel** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member clicks “Cancel Session”. | System shows confirmation popup. |
| **2** | Member selects “No” | System closes popup and returns to schedule. |
| **Exception Flow** | **Exception Flow 1: Scheduling service failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member confirm cancellation. |  |
| **2** |  | Scheduling system fails to update. |
| **3** |  | System shows: *“Unable to cancel session. Please try again later.”* |
| **4** |  | System logs technical error. |
| **Exception Flow 2: Database write error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member confirm cancellation. | System attempts to update booking. |
| **2** |  | Database error occurs. |
| **3** |  | System shows: *“Cancellation failed due to system error.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Sessions cannot be canceled after the allowed time window (configurable). * Only the Member who booked the session may cancel it. * Session cancellation must free up the associated time slot. * PT must be notified immediately upon cancellation. | | |

### 5.2.13 Use Case UC-13 Reschedule Training Session

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-13 | | |
| **Use Case Name** | Reschedule Training Session | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member reschedules an existing training session by selecting a new available time slot. The system validates the new time slot, updates the booking, and notifies the assigned PT about the change. | | |
| **Goal** | To allow Members to modify the date and time of their training sessions while ensuring scheduling accuracy and avoiding conflicts. | | |
| **Trigger** | The Member selects an existing session in their schedule and clicks “Reschedule Session.” | | |
| **Pre – conditions** | * Member is authenticated. * Member has at least one upcoming training session. * Session must be eligible for rescheduling (not expired, not within restricted time window). * PT availability exists for comparison (via UC-10). * Scheduling system is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Session is updated to new date and time. * PT is notified of the rescheduled session. * Old time slot is freed, and new one is reserved.   Failure Post-Condidtions   * No changes are made to the session. * Member receives an error message. * System may log the failed operation. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens their training schedule. | System displays list of upcoming sessions. |
| **2** | Member selects a session to reschedule. | System loads session details. |
| **3** | Member clicks “Reschedule Session.” | System displays PT’s availability (UC-10) |
| **4** | Member selects a new time slot. | System checks time slot availability and conflicts. |
| **5** | Member confirms new time. | System validates eligibility and updates booking. |
| **6** |  | System updates booking status and timestamps. |
| **7** |  | System notifies PT about the reschedule session. |
| **8** |  | System show confirmation message. |
| **9** | Member sees updated schedule. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Selects an Unavailable Time Slot** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member selects new slot. | System detects time conflict (PT busy or booked). |
| **2** |  | System displays: *“ Selected time slot is unavailabe.”* |
| **3** |  | System prompts user to choose another slot. |
| **Alternative Flow 2: Member Tries to Reschedule Too Late (Within Restricted Time Window)** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member tries to reschedule close to start time. | System checks time rules. |
| **2** |  | System displays: *“This session cannot be rescheduled at this time.”* |
| **Alternative Flow 3: PT has disable rescheduling** | | |
| **Step** | **Actor Action** | **System Respond** |
|  | Member selects new slot. | System detects PT does not allow rescheduling. |
|  |  | System displays: *“Rescheduling not allowed for this trainer.”* |
| **Alternative Flow 4: Member decides not to continue** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member cancels rescheduling or close dialog. | System returns to main schedule without changes. |
| **Exception Flow** | **Exception Flow 1: Scheduling service failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member confirms new time. |  |
| **2** |  | Scheduling system cannot update booking. |
| **3** |  | System displays: *“Unable to reschedule. Please try agan later.”* |
| **4** |  | System logs error. |
| **Exception Flow 2: Database update error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member confirms reschedule. |  |
| **2** |  | Database cannot save updated session data. |
| **3** |  | System displays: *“Rescheduling failed due to system error.”* |
| **4** |  | Error logged. |
| **Priority** | Medium | | |
| **Business Rule** | * Member may only reschedule upcoming sessions. * Sessions cannot be rescheduled within the defined cutoff time (e.g., 30 minutes before start). * New time slot must be verified against PT's availability. * If rescheduled, PT must be notified immediately. * Rescheduling should not cause overlapping sessions for the Member or PT. | | |

### 5.2.14 Use Case UC-14 Check-in via QR code

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-14 | | |
| **Use Case Name** | Check-in via QR code | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member checks in for a scheduled training session by scanning a QR code at the gym or within the mobile/web app. The system validates the QR code, verifies the session, and records attendance. | | |
| **Goal** | To allow Members to check in seamlessly to their scheduled training sessions, confirming attendance and triggering any necessary downstream actions (PT notification, attendance tracking). | | |
| **Trigger** | User scans the QR code displayed at the gym entrance or inside the app. | | |
| **Pre – conditions** | * Member is authenticated. * Member has a scheduled training session at the current time. * QR code is valid and unexpired. * QR Code Validation Module is operational (UC-15). * Scheduling System has accurate session data. | | |
| **Post – conditions** | Success Post-Conditions   * Check-in is recorded in the system. * Session status updated to “In Progress” or “Checked-in”. * PT receives a notification. * Attendance data stored for reporting.   Failure Post-Condidtions   * No check-in recorded. * System may log failed attempt. * Member receives appropriate error message. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member scans QR code. | Sytem read encoded session and member information. |
| **2** |  | System triggers UC-15 Validate QR Code |
| **3** | System verifies that Member has a scheduled session now. | System checks booking record. |
| **4** |  | System records attendance and updates session status. |
| **5** |  | System notifies PT of Member check-in. |
| **6** |  | System displays confirmation message. |
| **7** | Member sees success screen with session details. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Arrives Early (Before Allowed Check-in Window)** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member scans QR code early. | System validates time difference. |
| **2** |  | System displays: *“You cannot check in yet.”* |
| **Alternative Flow 2: Member Arrives Late** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member checks in after session start time. | System checks lateness policy. |
| **2** |  | System may still allow check-in or may show warning, depending on gym rules. |
| **Exception Flow** | **Exception Flow 1: Invalid or expired code** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member scans code. | System forwards to UC-15. |
| **2** |  | UC-15 returns invalid result. |
| **3** |  | System displays: *“Invalid or expired QR code”.* |
| **Exception Flow 2: No scheduled session found** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member scans code. | System checks schedule. |
| **2** |  | No session matches the scan time. |
| **3** |  | System displays: *“You have no scheduled session at this time.”* |
| **Exception Flow 3: Scheduling service failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System attempts to update attendance. | Scheduling service fails. |
| **2** |  | System displays: *“Unable to check-in. Please try again later.”* |
| **3** |  | Failure logged. |
| **Priority** | High | | |
| **Business Rule** | * Member must have a valid training session at the current time to check in. * QR code must be uniquely generated and securely encoded. * QR code validation (UC-15) must occur before check-in is recorded. * A Member may check into only one session at a time. * All check-in events must be time-stamped for reporting accuracy. | | |

### 5.2.15 Use Case UC-15 Validate QR Code

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-15 | | |
| **Use Case Name** | Validate QR Code | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member, System (QR Validation Module) | | |
| **Brief Description** | This use case describes how the system validates a QR code scanned by a Member during check-in. The system verifies that the QR code is valid, belongs to the correct Member, matches an upcoming training session, and has not expired. UC-15 is always triggered by UC-14 as a mandatory include step. | | |
| **Goal** | To ensure that only valid, secure, and authorized QR codes can be used for check-in. | | |
| **Trigger** | UC-14 (Check-in via QR Code) requests validation after the Member scans the QR code. | | |
| **Pre – conditions** | * QR code must be readable (camera/app scan successful). * QR code contains encoded session data (session ID, member ID, timestamp). * System clock is synchronized for time-sensitive validation. * Scheduling system is operational. | | |
| **Post – conditions** | Success Post-Conditions   * System confirms that QR code is valid. * UC-14 continues and proceeds with check-in logic.   Failure Post-Condidtions   * System returns validation failure. * UC-14 displays appropriate error message. * No check-in is recorded. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member scans QR code (From UC-14) | System extracts encoded data from QR image. |
| **2** |  | System verifies QR code structure and signature (if secure QR is used). |
| **3** |  | System checks if QR code is expired based on timestamp. |
| **4** |  | System checks if session ID corresponds to an active upcoming session. |
| **5** |  | Sytem verifies that the QR code belong to the same Member who is logged in. |
| **6** |  | System returns result: Valid |
| **7** | UC-14 continues with check-in steps. |  |
| **Alternative Flow** | **Alternative Flow 1: QR Code Does Not Match Logged-In Member** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | System read QR data. | ID mismatch detected. |
| **2** |  | System returns: *”QR code not associated with this account.”* |
| **3** |  | UC-14 stops check-in. |
| **Alternative Flow 2: QR code refer to wrong session** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | System checks session ID. | No matching upcoming session found. |
| **2** |  | System returns: *“Invalid session.”* |
| **Exception Flow** | **Exception Flow 1: QR code expired** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | System verifies timestamp. | QR timestamp older/ newer than allowed window. |
| **2** |  | System returns: *“QR code expired.”* |
| **3** |  | UC-14 terminates check-in. |
| **Exception Flow 2: Session already completed or canceled** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System loads session status. | Status = Completed/ Canceled. |
| **2** |  | System returns: *“Session not eligible for check-in.”* |
| **Exception Flow 3: Corrupted or tampered QR code** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System cannot decode QR content. | Signature mismatch or corrupted data. |
| **2** |  | System returns: *“Invalid QR code.”* |
| **3** |  | Event logged in security audit. |
| **Exception Flow 4: Scheduling System Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System attempts to fetch session data. | Scheduling system unreachable. |
| **2** |  | System returns: *“Unable to validate at this time.”* |
| **Priority** | High | | |
| **Business Rule** | * QR code must be uniquely generated per session. * QR codes must expire after a configured time window. * QR code must be bound to the Member and session ID. * Check-in cannot occur without successful validation of UC-15. * Tampered QR codes must be rejected and logged for security audit. | | |

### 5.2.16 Use Case UC-16 View Progress Charts

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-16 | | |
| **Use Case Name** | View Progress Charts | | |
| **Created by** | Thanh Le | **Last updated by** | Thanh Le |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how Members view their training progress through visual charts, including body measurements, performance metrics, historical trends, and improvements over time. Visualization helps Members understand their fitness progress and identify areas needing improvement. | | |
| **Goal** | To allow Members to view their training progress in graphical format (charts, graphs, trends). | | |
| **Trigger** | The Member selects “Progress” or “View Progress Charts” from the dashboard or profile menu. | | |
| **Pre – conditions** | * Member must be authenticated. * At least one metric, training note, or recorded session must exist. * Progress tracking data must be stored and accessible. * Charts and visualization components must function properly. | | |
| **Post – conditions** | Success Post-Conditions   * Visual charts are displayed to the user. * Summary insights may be shown (optional). * Member can navigate between metrics, time ranges, and categories.   Failure Post-Condidtions   * No charts displayed. * System logs errors if data retrieval or rendering fails. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “Progress Charts”. | System retrieves progress data (body metrics, session logs, PT notes). |
| **2** |  | System processes and aggregates data for visualization. |
| **3** |  | System renders charts such as:   * Weight trend * Body fat% * Workout frequency * Performance metric (reps, sets, loads) * Personal best history |
| **4** | Member selects different chart views or dat ranges. | System updates charts in real time. |
| **5** | Member reviews insights. | System may display summary analysis (*“Your squat strength improved 12%”*). |
| **Alternative Flow** | **Alternative Flow 1: No progress data available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens progress page. | System finds no recorded metrics or training history. |
| **2** |  | System displays: *“No progress data available yet.”* |
| **3** |  | System suggests: *“Start loggin your workouts or ask your PT to add session notes.”* |
| **Alternative Flow 2: Member filters progress data** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member applies filters (e.g., last 7 days, monthly, custom range). | System retrieves filtered data. |
| **2** |  | System updates the charts accordingly. |
| **Alternative Flow 3:Metric category filtering** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member selects a metric category (e.g., Bodyweight, Strength, Endurance). | System retrieves category-specific data. |
| **2** |  | System displays relevant charts. |
| **Exception Flow** | **Exception Flow 1: Data retrieval failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens progress charts. |  |
| **2** |  | System fails to load data due to daabase or service issue. |
| **3** |  | *System displays: “Unable to load progress data.”* |
| **4** |  | System logs exception for debugging. |
| **Exception Flow 2: Chart rendering error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System trues to render chart. | Rendering engine fails. |
| **2** |  | System displays: *“Error generating chart.”* |
| **3** |  | System logs technical error. |
| **Priority** | Medium | | |
| **Business Rule** | * Progress charts must be based only on valid recorded data. * All metrics must be time-stamped and associated with the correct Member. * Charts should provide accurate and up-to-date information at all times. * PT-entered training notes may influence some performance charts (if configured). * Users cannot edit data directly through this UC (read-only). | | |

### 5.2.17 Use Case UC-17 AI Workout Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-17 | | |
| **Use Case Name** | AI Workout Plan | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member receives a customized AI-generated workout plan based on their fitness goals, body metrics, training history, and personal preferences. The system sends relevant data to the external AI Engine, receives recommendations, and displays the final workout plan in a structured format. | | |
| **Goal** | To provide Members with a personalized workout plan optimized by AI to improve their training efficiency and progress. | | |
| **Trigger** | The Member opens “AI Workout Plan” from the Progress or Training Assistant menu. | | |
| **Pre – conditions** | * Member is authenticated. * Member has provided personal fitness goals and baseline metrics (optional but recommended). * AI Engine service is available. * Historical training data exists (from PT notes or member logs), or system can operate with minimum dataset. | | |
| **Post – conditions** | Success Post-Conditions   * AI-generated workout plan is displayed. * Member may save or follow the plan. * System stores the generated plan for future reference.   Failure Post-Condidtions   * No plan generated or displayed. * System logs the failure or external AI error. * Member may be asked to retry later. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens the AI Workout Plan page. | System fetches the member’s metrics, history, and preferences. |
| **2** | Member enters optional goal settings (e.g., “fat loss”, “muscle gain”). | System collects input. |
| **3** | Member clicks “Generate Workout Plan” | System complies data package for AI. |
| **4** |  | System sends request to AI Engine with:   * Body metrics * Workout history * PT notes * Fitness goals * Limitations/ injuries |
| **5** |  | AI Engine analyzes data and generates tailored workout plan. |
| **6** |  | System receives structured workout plan (exercises, sets, reps, rest, weekly schedule). |
| **7** |  | System displays the plan to the member. |
| **8** | Member reviews personalized workout program. |  |
| **Alternative Flow** | **Alternative Flow 1: Member has no training history** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens AI plan. | System sees no training history. |
| **2** |  | System displays: *“Provide initial fitness goals for a starter program.”* |
| **3** | Member submit goals. | System continues to Main Flow step 3. |
| **Exception Flow** | **Exception Flow 1: AI Engine unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to generate plan. |  |
| **2** |  | System receives timeout of failure from AI Engine. |
| **3** |  | *System displays: “Unable to generate AI plan at this moment. Please try again later.”* |
| **Exception Flow 2: AI Engine returns invalid data** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System receives response. | Plan format invalid or incomplete. |
| **2** |  | System logs error. |
| **3** |  | System displays: *“An error occurred while generating your plan.”* |
| **Exception Flow 3: Missing member data** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member attempts to generate plan. | System detects missing key data (e.g., no weight, no goal). |
| **2** |  | System displays: *“Please update your profile or goals before generating a plan.”* |
| **Priority** | High | | |
| **Business Rule** | * AI plan must be based only on verified Member data. * AI engine must receive anonymized data if required by privacy regulations. * Workout plan must include at least warm-up, main exercises, and cooldown. * Stored AI plans cannot be modified manually by Member. | | |

### 5.2.18 Use Case UC-18 AI Nutrition Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-18 | | |
| **Use Case Name** | AI Nutrition Plan | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member receives a personalized nutrition plan generated by the AI Engine. The system collects Member’s personal attributes (weight, height, goals), activity levels, and dietary preferences (if provided), sends them to the AI Engine, and displays a structured nutrition plan. | | |
| **Goal** | To automatically generate a nutrition plan tailored to the Member’s health profile and fitness goals. | | |
| **Trigger** | The Member clicks “AI Nutrition Plan” in the AI Assistant or Progress menu. | | |
| **Pre – conditions** | * Member is authenticated. * Basic profile data exists (age, weight, height, gender). * Nutritional preferences or dietary restrictions (optional) are available. * AI Engine is online and accessible. * Member has selected training goals (optional but recommended). | | |
| **Post – conditions** | Success Post-Conditions   * AI nutrition plan is displayed to the Member. * Plan is saved for future reference. * Member may choose to regenerate or adjust plan inputs.   Failure Post-Condidtions   * No nutrition plan generated. * System logs error or AI service failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “AI Nutrition Plan” page. | System retrieves Member’s body metrics, goal settings, dietary preferences. |
| **2** | Member updates optional preferences (e.g., vegetarian, allergy). | System stores or validates new inputs. |
| **3** | Member clicks “Generate Nutrition Plan”. | System complies data package for the AI Engine. |
| **4** |  | System sends request to AI Engine containing:   * Body metrics * Fitness goals * Activity level * Dietary preferences * Trainng schedule |
| **5** | AI Engine processes request. | AI Engine generates personalized nutrition plan. |
| **6** |  | System receives structured plan (meals, calories, macros, daily/ weekly schedule). |
| **7** |  | System displays nutrition plan to the member. |
| **8** | Member reviews AI-generated recommendations. |  |
| **Alternative Flow** | **Alternative Flow 1: Member has no body metric data** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to generate nutrition plan. | System detects missing BMI-related data. |
| **2** |  | System displays: *“Please update your body metrics to generate a nutrition plan.”* |
| **3** | Member updates metrics. | Flow returns to Main Flow step 1. |
| **Alternative Flow 2: Member Adjusts Preferences After Seeing Initial Plan** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member modifies preferences (e.g., no seafood). | System updates local preference set. |
| **2** | Member clicks “Regenerate Plan”. | Flow returns to Main Flow Step 3. |
| **Alternative Flow 3: Suggestion Mode** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member opens tips section. | System retrieves AI suggestions. |
| **Exception Flow** | **Exception Flow 1: AI Engine Timeout / Unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member clicks “Generate Nutrition Plan”. |  |
| **2** |  | System receives timeout or error. |
| **3** |  | System displays: *“AI service unavailable. Please try again later.”* |
| **Exception Flow 2: AI Returns Invalid or Incomplete Plan** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System receives malformed response. |  |
| **2** |  | System logs AI failure. |
| **3** |  | System displays: *“Unable to generate nutrition plan.”* |
| **Exception Flow 3: Missing Required User Information** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member requests plan. | System detects missing mandatory info (e.g., weight). |
| **2** |  | System displays: *“Please update your profile.”* |
| **Priority** | High | | |
| **Business Rule** | * Nutrition plan must include daily calorie target and macronutrient breakdown. * AI-generated plan must respect Member’s dietary restrictions. * Recommendations cannot contradict verified medical restrictions (if known). * Nutrition plan cannot be manually edited. * Regeneration of plans may be throttled to reduce AI processing load. | | |

### 5.2.19 Use Case UC-19 Pose Recognition

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-19 | | |
| **Use Case Name** | Pose Recognition | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how the system analyzes a Member’s training movements through their device camera using AI pose recognition technology. The system detects and evaluates exercise form, identifies incorrect posture, and provides real-time feedback to help prevent injuries and improve technique. | | |
| **Goal** | To analyze Members’ exercise posture in real time and provide corrective feedback. | | |
| **Trigger** | Member selects “Pose Recognition” from the AI Training Assistant or Workout Tools menu. | | |
| **Pre – conditions** | * Member is authenticated. * Camera permissions are granted. * AI Engine for pose detection is operational. * Internet connection available (if cloud AI is used). * Member has started or selected an exercise for analysis. | | |
| **Post – conditions** | Success Post-Conditions   * Pose evaluation results are displayed or spoken in real-time. * Session data may be logged for progress tracking. * System may store detected form issues for recommendations.   Failure Post-Condidtions   * No pose analysis performed. * Member sees error message. * System logs failure reasons (device/camera/AI issues). | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens Pose Recognition feature. | System checks device camera permission. |
| **2** | Member starts workout pose analysis. | System activates camera and prepares AI model. |
| **3** | Member performs exercise movements. | System captures frames and sends them to AI Engine. |
| **4** |  | AI Engine identifies body keypoints and posture alignment. |
| **5** |  | System evaluates form based on exercise-specific rules. |
| **6** |  | System displays real-time feedback:   * “Back to rounded” * “Knees not aligned” * “Good form!” |
| **7** |  | System saves optional pose history. |
| **8** | Member end session. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Declines Camera Permission** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens the feature. | System requests camera permission. |
| **2** | Member denies permission. | System displays: *“Camera access required to use Pose Recognition.”* |
| **Alternative Flow 2: Poor Lighting or Camera Quality** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member starts session. | System detects poor video quality. |
| **2** |  | System displays: *“Low visibility detected. Improve lighting for accurate analysis.”* |
| **Alternative Flow 3: No person detected** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member starts analysis. | AI Engine receives frames. |
| **2** |  | AI detects no human in frames. |
| **3** |  | System displays: *“Unable to detect user. Please adjust camera position.”* |
| **Alternative Flow 4: Unsupported exercise** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member selects advanced or unsupported movement. | AI Engine attempts analysis. |
| **2** |  | System shows *“Exercise not supported for pose detection yet.”* |
| **Exception Flow** | **Exception Flow 1: AI Engine Timeout/ Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | System sends frame to AI. |  |
| **2** | AI Engine fails to respond. |  |
| **3** |  | System displays: *“Pose recognition temporarily unavailable.”* |
| **Exception Flow 2: Camera Malfunction or Not Found** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member attempts to start analysis. |  |
| **2** | System detects no functional camera. |  |
| **3** |  | System shows: *“Camera error. Please reconnect your device.”* |
| **Exception Flow 3: Network Failure (for cloud AI)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member starts session. | System sends frames to AI. |
| **2** | |  | | --- | |  |  |  | | --- | | Internet interrupted. | |  |
| **3** |  | System shows: *“Connection lost. Pose analysis paused.”* |
| **Priority** | High | | |
| **Business Rule** | Pose recognition must work within configured accuracy thresholds.  Camera frames must be processed securely according to privacy rules.  Feedback must be exercise-specific and based on aligned anatomical models.  Pose analysis should not be stored unless Member opts in.  System must not store raw video unless explicitly permitted. | | |

### 5.2.20 Use Case UC-20 Injury Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-20 | | |
| **Use Case Name** | Injury Risk Analysis | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case allows the system to analyze a Member’s injury risk using AI models. The analysis may include posture data, recent workout loads, frequency, fatigue level, past injuries, and trends from progress charts. The output is a risk score and recommendations to prevent injury. | | |
| **Goal** | To help Members prevent injuries by receiving an AI-generated risk assessment and safety recommendations. | | |
| **Trigger** | Member selects “Injury Risk Analysis” from the AI Assistant or Progress menu. | | |
| **Pre – conditions** | * Member is authenticated. * AI Engine is available. * At least minimal training data exists (session logs, metrics, or pose data). * If integrated with UC-19, pose data may enhance accuracy | | |
| **Post – conditions** | Success Post-Conditions   * An injury risk report is displayed. * Recommendations for safe training are provided. * Risk score stored for history tracking (optional).   Failure Post-Condidtions   * No analysis performed. * Error is logged. * Member informed to retry later. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens Injury Risk Analysis screen. | System retrieves recent training data, metrics, and optional pose data. |
| **2** | Member inputs optional additional info (e.g., fatigue, soreness level). | System updates parameters. |
| **3** | Member clicks “Analyze Risk”. | System prepares dataset for AI Engine. |
| **4** |  | System sends data to AI Engine, including:   * Workout history * Load progression  Frequency & intensity * Body metrics * Pose recognition patterns (if available) * Past injuries (if stored) |
| **5** | AI Engine processes data. | AI generates risk score and recommendations. |
| **6** |  | System receives structured risk assessment. |
| **7** |  | System displays:   * Injury risk score (e.g., Low / Medium / High) * Risk factors * Recommended corrective actions |
| **8** | Member reviews report. |  |
| **Alternative Flow** | **Alternative Flow 1: Member Has Minimal Data** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens analysis screen. | System detects insufficient historical data. |
| **2** |  | System displays: *“Limited data available. Analysis may be less accurate.”* |
| **3** | Member continues. | System continues to Main Flow Step 3. |
| **Alternative Flow 2: Member Manually Inputs Recent Activity** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member enters missing data manually (e.g., external workout info). | System stores supplemental data temporarily. |
| **Alternative Flow 3: PT-Assisted Mode** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT views Member risk analysis (UC-29). | PT submits recommendations. |
| **Exception Flow** | **Exception Flow 1: AI Engine Timeout or Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member confirms analysis request. |  |
| **2** |  | AI Engine does not respond in time. |
| **3** |  | System displays: *“Unable to perform injury analysis right now.”* |
| **Exception Flow 2: Data Corrupted or Inconsistent** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System compiles data. | System detects missing or conflicting values. |
| **2** |  | System displays: *“Some data could not be analyzed.”* |
| **Exception Flow 3: Network Failure (Cloud AI)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System sends data to AI Engine. | Network interruption occurs. |
| **2** |  | System reports temporary connectivity issue. |
| **Priority** | Medium | | |
| **Business Rule** | * AI risk scores must be based solely on validated data sources. * Pose data (if used) must follow privacy rules and opt-in requirements. * Risk levels must follow defined thresholds (Low/Medium/High). * Injury risk results cannot override medical advice. * Generated recommendations must prioritize safety. | | |

### 5.2.21 Use Case UC-21 Submit Support Ticket

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-21 | | |
| **Use Case Name** | Submit Support Ticket | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member submits a support ticket to request assistance regarding issues such as payment problems, booking issues, technical errors, or general inquiries. The system stores the ticket, assigns it an ID, and sends it to the Support Staff for processing. | | |
| **Goal** | To allow Members to send support requests so the Support Staff can handle and respond to them. | | |
| **Trigger** | Member selects “Submit Ticket” or clicks “Support” from the menu. | | |
| **Pre – conditions** | * Member is authenticated. * Ticket system is operational. * Member has at least one issue or inquiry to report. | | |
| **Post – conditions** | Success Post-Conditions   * A support ticket is created and stored in the system. * Ticket receives a unique ID and timestamp. * Support Staff is notified of the new ticket. * Member can view ticket status through UC-22.   Failure Post-Condidtions   * Ticket is not created. * Member receives error notification. * Error is logged. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens the Support Ticket forms | System displays required fields: Subject, Category, Description, Attachments. |
| **2** | Member fills in details and optionally attaches screenshots/images. | System validates field formats. |
| **3** | Member clicks “Submit Ticket”. | System validates mandatory fields (subject, description). |
| **4** |  | System creates new ticket with status “Open”. |
| **5** |  | System assign ticket DI, timestamp, and category. |
| **6** |  | System notifies Support Staff of the new ticket. |
| **7** |  | System displays confirmation message: *“Your ticket has been submitted.”* |
| **8** | Member sees new ticket in UC-22. |  |
| **Alternative Flow** | **Alternative Flow 1: Missing Required Information** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member attempts to submit without entering all required fields. | System detects missing values. |
| **2** |  | System displays: *“Please provide subject and description.”* |
|  | Member fills missing info. | Flow returns to Main Flow Step 3. |
| **Alternative Flow 2: Member Cancels Submission** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member opens ticket form. |  |
| **2** | Member clicks “Cancel”. | System discards all entered data. |
| **Alternative Flow 3: Auto-Categorization** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member enters description. | System uses AI/NLP to guess ticket category. |
| **2** | |  | | --- | | System suggests category. |  |  | | --- | |  | | Member may accept or override suggestion. |
| **Exception Flow** | **Exception Flow 1: Ticket Creation Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member clicks “Submit Ticket”. | System attempts to save ticket. |
| **2** |  | Database or ticket management service fails. |
| **3** |  | System displays*: “Unable to submit ticket. Please try again later.”* |
| **Exception Flow 2: Attachment Upload Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member adds attachments. | System attempts to upload files. |
| **2** |  | Upload fails due to size/format errors. |
| **3** |  | System displays: *“Attachment upload failed.”* |
| **4** | Member may retry upload or submit without attachments. |  |
| **Priority** | High | | |
| **Business Rule** | * All tickets must have a unique Ticket ID. * Tickets must be classified by category for routing. * Only authenticated Members may submit tickets. * Tickets cannot be edited once submitted (only Support Staff may update status). * Attachments must follow allowed file type and size limits. | | |

### 5.2.22 Use Case UC-22 Track ticket status

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-22 | | |
| **Use Case Name** | Track ticket status | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member checks the status of support tickets they previously submitted. The system retrieves all tickets associated with the Member and shows their statuses (Open, In Progress, Waiting for Member, Resolved, Closed, or Escalated). Member can view individual ticket details, staff responses, and timestamps. | | |
| **Goal** | To allow Members to track the progress and resolution of their support requests | | |
| **Trigger** | Member navigates to “My Tickets”, “Support History”, or a similar option. | | |
| **Pre – conditions** | * Member is authenticated. * At least one ticket exists (optional; if none exist, an empty list is shown). * Ticket Management Module is operational. | | |
| **Post – conditions** | Success Post-Conditions   * System displays tickets belonging to the Member. * Member can open a ticket to view full details. * If ticket requires Member response, system highlights it.   Failure Post-Condidtions   * No ticket information is shown. * Error is logged. * Member receives an error notification. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “My Tickets”. | System retrieves all tickets created by the Member. |
| **2** |  | System displays each ticket with summary:   * Ticket ID * Title * Category * Created date * Current status (Open/ In progress/ Resolved/ etc.) |
| **3** | Member selects a ticket to view details. | System displays full ticket content:   * Description * Attachments * Staff responses * Status history * Time stamps |
| **4** | Member reviews the ticket. | System highlights any required Member action (e.g., “Please provide additional details.”) |
| **5** | Member optionally closes screen. | System returns to dashboard or prior menu. |
| **Alternative Flow** | **Alternative Flow 1: Member has no ticket** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “My Tickets” | System finds zero tickets. |
| **2** |  | System displays: *“You have not submitted any tickets yet”* |
| **3** |  | System shows shortcut to Submit Ticket (UC-21) |
| **Alternative Flow 2: Tickets requires member response** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member opens “My Tickets”. | System identifies a ticket waiting on user input. |
| **2** |  | System displays tag “Waiting for Your Response”. |
| **3** | Member opens ticket to view Staff question. | System displays chat-style message thread. |
| **Alternative Flow 3: Filter / Sort Tickets** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member selects filter (e.g., Open Only, Resolved). | System updates list. |
| **2** | Member sorts tickets (e.g., by date, status). | System displays rearranged results. |
| **Exception Flow** | **Exception Flow 1: Ticket Module Unavailable** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens “My Tickets”. | System fails to retrieve ticket data. |
| **2** |  | System displays: *“Unable to retrieve tickets at this moment.”* |
| **3** |  | System logs failure. |
| **Exception Flow 2: Database Error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | System requests ticket list. | Query fails. |
| **2** |  | System displays: *“An error occurred while loading your tickets.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Members may only view tickets they created. * Each ticket must include a valid status. * System must highlight tickets requiring Member input. * Closed or resolved tickets must remain visible in history. * Ticket data must be immutable except by authorized Support Staff. | | |

### 5.2.23 Use Case UC-23 Live chat support

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-23 | | |
| **Use Case Name** | Live Chat Support | | |
| **Created by** | Hung Nguyen | **Last updated by** | Hung Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes how a Member interacts with the Support Staff through real-time chat to receive assistance. The system establishes a live chat session, routes the Member to an available Support Staff, enables two-way messaging, and maintains the session history. | | |
| **Goal** | o provide Members with instant, real-time support through a chat interface. | | |
| **Trigger** | Member clicks “Live Chat” or selects “Chat with Support” from the Support menu. | | |
| **Pre – conditions** | * Member is authenticated. * Chat Service Module is operational. * At least one Support Staff is online (or system places user in queue). * Member has a need for support or inquiry. | | |
| **Post – conditions** | Success Post-Conditions   * A real-time chat session is established. * Conversation is logged. * Support Staff may close or resolve the chat. * Member may review chat history later.   Failure Post-Condidtions   * No chat session is created. * Member receives an error message. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member selects “Live Chat Support”. | System initializes chat session. |
| **2** |  | System attempts to connect to an available Support Staff representative. |
| **3** | Support Staff accepts the chat. | System displays chat interface to Member. |
| **4** | Member sends a message describing the issue. | System relays message instantly to Support Staff. |
| **5** | Support Staff replies. | System displays staff response to Member. |
| **6** | Member and Staff continue real-time conversation. | System maintains active chat session. |
| **7** | Support Staff ends or resolves the chat. | System displays chat summary and status (“Resolved” or “Closed”) |
| **8** | Member exits chat. | System saves session to chat history. |
| **Alternative Flow** | **Alternative Flow 1: No Support Staff Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member opens Live chat | System checks staff availability. |
| **2** |  | System shows: *“All agents are currently busy. Please wait…”* |
| **3** | System places Member in queue. | Member waits for assignment. |
| **Alternative Flow 2: Member Leaves Chat Mid-Session** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member closes chat window. | System marks session as “Member left”. |
| **2** | Support Staff may add final notes. | System updates chat log. |
| **Alternative Flow 3: Offline Mode (Convert to Ticket)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member attempts to start chat. | System unavailable. |
| **2** |  | System suggests: *“Would you like to submit a ticket instead?”* |
| **3** | Member accepts. | System redirects to UC-21 Submit Support Ticket |
| **Exception Flow** | **Exception Flow 1: Chat Service Connection Lost** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member or Staff sends a message. | Connection drops. |
| **2** |  | System displays: *“Connection lost. Reconnecting…”* |
| **3** |  | If fails, system ends chat and logs issue. |
| **Exception Flow 2: Notification Service failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff replies. | Notification service down. |
| **2** |  | System still shows reply in chat but cannot send push notification. |
| **Exception Flow 3: Unauthorized access attempt** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Guest tries open chat | System detects no authentication. |
| **2** |  | System displays: *“Please log in to use Live Chat Support.”* |
| **Priority** | High | | |
| **Business Rule** | * Each chat session must be logged and timestamped. * Only Members may initiate live chat sessions. * Chat history must remain accessible to the Member. * Chat sessions may be closed only by authorized Staff. * Support Staff must acknowledge Member messages promptly. | | |

### 5.2.24 Use Case UC-24 Update Availability

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-24 | | |
| **Use Case Name** | Update Availability | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer updates their availability schedule (days, times, repeating weekly slots). This availability will be visible to Members for booking training sessions under UC-10 View Trainer Availability and used by the system in UC-11 Book Training Session. | | |
| **Goal** | To allow PTs to manage their working hours and availability so Members can book training sessions accurately. | | |
| **Trigger** | PT selects “Availability” or “Update Availability” in their dashboard or schedule management menu. | | |
| **Pre – conditions** | * PT is authenticated. * Scheduling System is operational. * PT has at least one branch assignment (optional depending on system design). * Existing availability data (if any) is stored and retrievable. | | |
| **Post – conditions** | Success Post-Conditions   * Updated availability is saved to the system. * Members can view updated availability in UC-10. * System may notify Members with pending requests (optional feature).   Failure Post-Condidtions   * No changes made. * Error logged. * PT may retry the action. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens “Update Availability” page. | System retrieves existing availability slots (if any). |
| **2** | PT selects days and times they are available (single or recurring weekly). | System validates input format. |
| **3** | PT optionally marks breaks, lunch hours, or unavailable periods. | System updates preview schedule. |
| **4** | PT clicks “Save Availability”. | System checks for conflicts (existing bookings, closed hours, etc.). |
| **5** |  | System saves the updated availability. |
| **6** |  | System confirms: “Availability updated successfully.” |
| **7** | PT sees updated schedule. |  |
| **Alternative Flow** | **Alternative Flow 1: PT remove all availability** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT clears all availability slots. | System warns: *“Members will not be able to book sessions.”* |
| **2** | PT confirms removal. | System saves empty availability. |
| **Alternative Flow 2: PT Sets Recurring Weekly Schedule** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT selects recurring option (e.g., repeat every Monday 9–11 AM). | System duplicates schedule weekly. |
| **Alternative Flow 3: PT adjusts availability for one specific day** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT selects specific date. | System shows override mode. |
| **2** | PT modifies schedule for that date. | System saves date-specific exception. |
| **Exception Flow** | **Exception Flow 1: Scheduling Conflict Detected** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT attempts to save availability. | System detects conflict with existing booked sessions. |
| **2** |  | System displays: *“Cannot modify availability due to existing bookings.”* |
| **3** | PT adjusts schedule. | Flow returns to Main Flow Step 2. |
| **Exception Flow 2: Invalid Time Range** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT sets end time earlier than start time. | System detects invalid entry. |
| **2** |  | System displays: *“Invalide time range.”* |
| **Exception Flow 3: System error saving availability** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT saves schedule. | System fails to update database. |
| **2** |  | System displays: *“Unable to save availability. Please try again later.”* |
| **3** |  | System logs failure. |
| **Priority** | High | | |
| **Business Rule** | * PT availability must not overlap with existing booked sessions. * PT availability must follow gym working hours. * Recurring availability must follow defined recurrence rules. * PTs may override recurring availability for specific dates. * Members may only book within current PT availability. | | |

### 5.2.25 Use Case UC-25 View Work Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-25 | | |
| **Use Case Name** | View Work Schedule | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer views their work schedule, including upcoming sessions, pending booking requests, canceled sessions, and availability blocks created via UC-24. PTs use this schedule to manage their day-to-day training sessions. | | |
| **Goal** | To allow PTs to see all scheduled training sessions and manage their time effectively. | | |
| **Trigger** | PT selects “Work Schedule”, “Calendar”, or “My Sessions” in the PT dashboard. | | |
| **Pre – conditions** | * PT is authenticated. * Scheduling System is available. * PT has availability or sessions stored in the system. | | |
| **Post – conditions** | Success Post-Conditions   * PT sees complete schedule for a selected time range (daily/weekly/monthly). * Session statuses are clearly displayed. * PT may proceed to other actions (Accept/Decline, Conduct Session).   Failure Post-Condidtions   * No schedule is displayed. * System logs retrieval error. * PT receives notification of schedule loading failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens “Work Schedule” page. | System loads schedule data (upcoming sessions, pending requests, availability blocks). |
| **2** | PT selects time view (daily/weekly/monthly). | System updates the calendar accordingly. |
| **3** | PT clicks on a session to see details. | System displays:   * Member name * Date and time * Status (Pending/ Confirmed/ Canceled). * Notes (If any) |
| **4** | PT reviews schedule. | PT may proceed to UC-26 (Accept/Decline) or UC-27 (Conduct Session). |
| **5** | PT optionally filters schedule. | System updates view based on filter criteria. |
| **Alternative Flow** | **Alternative Flow 1: PT has no schedule session** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT opens schedule. | System detects no sessions. |
| **2** |  | System displays: *“No scheduled sessions.”* |
| **3** |  | System optionally suggests updating availability (UC-24). |
| **Alternative Flow 2: PT filters sessions by status or member** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT applies filters (e.g., Pending only). | System refreshes the schedule list. |
| **2** | PT selects a Member name. | System filters and displays sessions with that Member. |
| **Exception Flow** | **Exception Flow 1: Scheduling System Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT tries to load schedule. | Scheduling service unavailable. |
| **2** |  | System displays: *“Unable to load schedule. Please try again later.”* |
| **3** |  | System logs error. |
| **Exception Flow 2: Session data corrupted/ missing** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT views schedule. | System detects incomplete data. |
| **2** |  | System displays: *“Some session data could not be loaded.”* |
| **Priority** | High | | |
| **Business Rule** | * PT may only view sessions assigned to them. * Pending booking requests must be clearly flagged. * Canceled sessions must remain visible with proper indication. * Schedule must always show real-time updated booking statuses. * PT availability (UC-24) must integrate directly into schedule view. | | |

### 5.2.26 Use Case UC-26 Accept/ Decline Booking Requests

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-26 | | |
| **Use Case Name** | Accept/ Decline Booking Requests | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer reviews, accepts, or declines booking requests submitted by Members. The system updates booking status, notifies the Member, and ensures session scheduling is consistent. | | |
| **Goal** | To allow PTs to manage pending booking requests and confirm their training schedule. | | |
| **Trigger** | PT selects “Booking Requests” or clicks on a session in the Work Schedule (UC-25) with Pending status. | | |
| **Pre – conditions** | * PT is authenticated. * At least one pending booking request exists. * Scheduling System is operational. * Member must have active membership (verified before initial booking). | | |
| **Post – conditions** | Success Post-Conditions (Accept)   * Booking status updated to Confirmed. * Time slot locked. * Member receives confirmation notification.   Success Post-Conditions (Decline)   * Booking status updated to Declined. * Time slot remains free for others to book. * Member receives decline notification.   Failure Post-Condidtions   * No changes are made. * Member receives no update. * System logs failure. | | |
|  | **Main Flow 1: Accepting a Booking Request** | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens “Booking Requests”. | System retrieves list of pending requests. |
| **2** | PT selects a request to review. | System displays details: Member, date/time, session type. |
| **3** | PT clicks “Accept”. | System checks time slot availability (no double booking). |
| **4** |  | System updates booking status to Confirmed. |
| **5** |  | System sends confirmation notification to Member. |
| **6** | PT sees session marked as Confirmed in schedule. | End of flow. |
| **Main Flow 2: Declining a Booking Request** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT reviews request. | System shows booking details. |
| **2** | PT clicks “Decline”. | System prompts optional *“Reason for Decline.”* |
| **3** | PT confirms decline. | System marks session as Declined. |
| **4** |  | System notifies Member of decline and encourages rebooking. |
| **Alternative Flow** | **Alternative Flow 1: Member Cancels Before PT Responds** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT opens pending request. | System checks ststus. |
| **2** |  | System shows: *“Member canceled this request.”* |
| **3** |  | Request removed. |
| **Exception Flow** | **Exception Flow 1: Scheduling Conflict Detected at the Moment of Acceptance** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT clicks Accept. | System re-checks availability. |
| **2** | Another Member booked the same slot moments earlier. | System displays: *“Time slot no longer available.”* |
| **3** |  | System marks request as Failed / Conflict. |
| **Exception Flow 2: System or database failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT tries to accept or decline. | Database error occurs. |
| **2** |  | System displays: *“Unable to update booking. Please try again later.”* |
| **3** |  | System logs failure. |
| **Priority** | High | | |
| **Business Rule** | * Only PTs assigned to a session may accept or decline that request. * Accepted sessions must block the time slot for others. * Declined sessions must not block time slots. * Notifications must be sent after accept/decline actions. * PT cannot accept a booking if it overlaps with another confirmed session. | | |

### 5.2.27 Use Case UC-27 Conduct Training Session

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-27 | | |
| **Use Case Name** | Conduct Training Session | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer conducts a scheduled training session with a Member. It includes verifying session details, marking attendance (if needed), delivering the training session, and optionally preparing notes (UC-28 Record Training Notes). | | |
| **Goal** | To allow PTs to perform and manage a training session that has been scheduled and confirmed. | | |
| **Trigger** | At the scheduled session time, PT selects “Start Session” from the Work Schedule (UC-25). | | |
| **Pre – conditions** | * PT is authenticated. * Session must be in Confirmed status. * Member has checked in via UC-14 (optional; depends on workflow). * Scheduling System is functioning. | | |
| **Post – conditions** | Success Post-Conditions   * Session is marked as Completed. * PT may proceed to UC-28 to add notes. * Member sees the completed session in their training history.   Failure Post-Condidtions   * Session may remain in Scheduled or In Progress state. * Failure logged. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens Work Schedule and selects a confirmed session. | System shows sessions details. |
| **2** | PT clicks “Start Session” | System changes session status to In progress. |
| **3** | PT trains the Member according to the planned activities. | System keeps session active. |
| **4** | PT may track performance metrics during the session. | Metric temporarily stored. |
| **5** | PT clicks “End session”. | Session time ends. |
| **6** |  | System updates session status to Completed. |
| **7** |  | System prompts PT to add notes. |
| **8** | PT proceeds to UC-28 (Record Training Notes) or skips. |  |
| **Alternative Flow** | **Alternative Flow 1: Member has not checked-in** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT attempts to start session. | System detects no check-in. |
| **2** |  | System displays: *“Member has not checked in.”* |
| **3** | PT may manually override. | Session begins or is canceled. |
| **Alternative Flow 2: Member Cancels Last Minute** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT opens schedule. | System shows session status as Canceled. |
| **2** | PT takes no action. | Session ends without training. |
| **Alternative Flow 3: PT Marks No-Show** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Member does not appear. | PT selects “Mark Now-Show” |
| **2** |  | System sets sessions status to Now-Show. |
| **3** | Member notified automatically. |  |
| **Alternative Flow 4: PT Reschedules Session Mid-Training** | | |
| **Step** | **Acttor Action** | **System Respond** |
| **1** | PT ends session early. | System asks: *“Reschedule remaining time?”* |
| **2** | PT selects new time. | System goes to UC-13 |
| **Exception Flow** | **Exception Flow 1: Scheduling System Failure During Session** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT tries to start/complete session. | System cannot update session status. |
| **2** |  | System displays: *“Unable to update session state.”* |
| **3** |  | System logs error; PT may retry later. |
| **Exception Flow 2: Member Switches Trainer Unexpectedly** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Session starts. | System detects member reassigned. |
| **2** |  | System displays alert to PT. |
| **Exception Flow 3: Network/Device Failure (If session requires app input)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT tries to enter metrics. | System loses connection. |
| **2** |  | System stores data locally or shows error. |
| **Priority** | High | | |
| **Business Rule** | * Only the assigned PT may conduct the session. * Session cannot be started before scheduled time (unless early start is allowed). * Session cannot be marked “Completed” without starting it. * Training results must be recorded in UC-28 if required by gym policy. * System must prevent two overlapping “In Progress” sessions for the same PT. | | |

### 5.2.28 Use Case UC-28 Record Training Notes

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-28 | | |
| **Use Case Name** | Record Training Notes | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer records session notes after conducting a training session. Notes may include exercises performed, performance metrics, attendance quality, PT feedback, and suggestions for improvement. The information supports progress tracking (UC-16), AI workout plan customization (UC-17), and session feedback (UC-30). | | |
| **Goal** | To allow PTs to document important training insights and performance metrics for the Member. | | |
| **Trigger** | After completing a training session (UC-27), the PT chooses “Record Notes”. | | |
| **Pre – conditions** | * PT is authenticated. * Session must be in Completed status. * Member must exist and be assigned to the session. * Progress Tracking Module is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Notes are saved and linked to the training session. * Member can view progress through UC-16. * Notes may be used for future AI-based recommendations. * Session feedback (UC-30) may be triggered.   Failure Post-Condidtions   * Notes are not saved. * System logs error. * PT may retry. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens the completed training session. | System loads session details. |
| **2** | PT clicks “Record Notes”. | System displays note entry form. |
| **3** | PT enters training details, such as:   * Exercise performed * Sets/ reps/ weights * Performance metrics * Member strengths/ weaknesses * Suggestion for improvement | System validates input format. |
| **4** | PT optionally attaches media (photos, videos). | System uploads files if allowed. |
| **5** | PT clicks “Save Notes”. | System saves notes and links them to the session. |
| **6** |  | System displays confirmation. |
| **7** | PT may proceed to UC-30 (Send Session Feedback). |  |
| **Alternative Flow** | **Alternative Flow 1: PT saves notes later** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT closes session without saving notes. | System asks: *“Save notes later?”* |
| **2** | PT confirms | System keeps session incomplete until notes are added |
| **Alternative Flow 2: Quick note template selection** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT selects “Use Template”. | System loads predefined training note templates. |
| **2** | PT modifies template content. | System proceeds to Step 3. |
| **Exception Flow** | **Exception Flow 1: System fails to save notes** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT clicks Save. | Database error occurs. |
| **2** |  | System displays: *“Unable to save notes. Please try again later.”* |
| **3** |  | System logs error. |
| **Exception Flow 2: Invalid Attachments** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT attaches media. | File violates size/type rules. |
| **2** |  | System displays: *“Attachment not allowed.”* |
| **3** | PT removes file. | Flow returns to Step 3. |
| **Exception Flow 3: Session Not Completed** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT attempts to open notes. | System checks session state. |
| **2** |  | System displays: *“You can record notes only after completing the session.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Only the assigned PT may record notes for a session. * Notes must be linked to a specific completed session. * Notes must remain immutable after submission (unless PT edits are allowed by policy). * Any recorded metrics must use standardized units. * Notes may contribute to Member progress insights (UC-16 & AI modules). | | |

### 5.2.29 Use Case UC-29 View Member Progress

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-29 | | |
| **Use Case Name** | View Member Progress | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer views a Member’s training progress, including historical training notes, performance metrics, AI-generated analysis, and visual progress charts. PTs use this information to tailor future training sessions and provide better coaching. | | |
| **Goal** | To allow PTs to monitor Member performance and progress in order to provide personalized training guidance. | | |
| **Trigger** | PT selects “Member Progress” from the PT dashboard or opens a specific Member profile. | | |
| **Pre – conditions** | * PT is authenticated. * PT has permission to view the Member’s data (assigned Member or gym policy). * Progress data exists (training notes, metrics, charts). * Progress Tracking Module is operational. | | |
| **Post – conditions** | Success Post-Conditions   * PT can view comprehensive progress history. * PT may update future training plans. * PT may use insights to improve session planning.   Failure Post-Condidtions   * No data displayed. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT opens “Member Progress.” | System loads list of Members assigned to PT. |
| **2** | PT selects a Member. | System retrieves Member profile and progress data. |
| **3** |  | System displays:   * Body metrics trends * Performance charts * Training history * Session notes (UC-28) * Attendance (UC-14) * AI insights |
| **4** | PT selects a specific date or session. | System displays detailed breakdown for that session. |
| **5** | PT reviews progress and adjusts plans if needed. | PT may proceed to UC-27 or AI modules (UC-17/18). |
| **Alternative Flow** | **Alternative Flow 1: Member has no progress data** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT selects Member. | System detects no logs or metrics. |
| **2** |  | System displays: *“No progress data available.”* |
| **3** |  | System suggests: *“Start recording notes in UC-28.”* |
| **Alternative Flow 2: PT Filters Progress by Category** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT selects filter (e.g., Strength, Endurance, Weight). | System updates displayed charts. |
| **Alternative Flow 3: PT views multi-session comparison** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT selects two dates or sessions. | System compares performance and shows difference. |
| **Exception Flow** | **Exception Flow 1: Data retrieval failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT selects member. | Systen fails to load data. |
| **2** |  | System displays: *“Unable to load progress data.”* |
| **3** |  | System logs failure. |
| **Exception Flow 2: Unauthorized Access** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT attempts to open Member progress. | System verifies permissions. |
| **2** |  | System displays: *“You are not authorized to view this Member’s progress.”* |
| **Exception Flow 3: Incomplete or corrupted data** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT attempts to open Member progress. | System retrieves data. |
| **2** |  | Some entries missing or invalid. |
| **3** |  | System displays warning: *“Some progress data may be incomplete.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Data visualization must be accurate and up to date. * Historical progress may not be edited by PT. * AI-generated insights must be based on verified training data. * All progress data must be time-stamped and linked to sessions. * PTs may only access progress data for Members assigned to them. | | |

### 5.2.30 Use Case UC-30 Send Session Feedback

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-30 | | |
| **Use Case Name** | Send Session Feedback | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Personal Trainer (PT) | | |
| **Brief Description** | This use case describes how a Personal Trainer sends structured feedback to a Member after completing a training session. The feedback may include performance evaluation, technique correction, recommended improvements, or next steps in the training plan.  Feedback helps Members understand how they performed and stay motivated in their training journey. | | |
| **Goal** | To provide Members with actionable, personalized feedback after each training session. | | |
| **Trigger** | After recording training notes in UC-28, PT clicks “Send Feedback to Member.” | | |
| **Pre – conditions** | * PT is authenticated. * Session must be completed (UC-27). * Training notes must exist (from UC-28). * Member must have a valid account and notification channel. | | |
| **Post – conditions** | Success Post-Conditions   * Feedback is saved and linked to the session. * Member receives notification. * Feedback appears in the Member’s training history and progress dashboard.   Failure Post-Condidtions   * Feedback not sent or saved. * Error logged. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | PT finishes recording notes (UC-28). | System shows option: “Send Feedback to Member.” |
| **2** | PT opens feedback form. | System loads template or blank form. |
| **3** | PT writes feedback such as:   * Performance quality * Form corrections * Strengths and weaknesses * Next steps and goals * Homework exercise | System validates input. |
| **4** | PT clicks “Send Feedback.” | System saves feedback and links it to session record. |
| **5** |  | System sends notification to Member (“New training feedback available”). |
| **6** |  | System confirms success message. |
| **7** | PT returns to dashboard or next session. |  |
| **Alternative Flow** | **Alternative Flow 1: Uses Feedback Template** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT selects “Use Template” | System laods predefined templates:   * Strength Training Session Feedback * Cardio Session Feedback * Beginner’s Form Feedback |
| **2** | PT edits template text. | System proceeds to step 3. |
| **Alternative Flow 2: PT Saves Feedback as Draft** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | PT decides not to send immediately. | System allows “Save as Draft”. |
| **2** | PT saves draft. | System stores incomplete feedback. |
| **3** | PT returns later to complete and send. | System resumes to Step 2. |
| **Alternative Flow 3: Member has diable notifications** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT sends feedback. | System detects Member disabled notifications. |
| **2** |  | System stores feedback normally, but no push notification sent. |
| **3** |  | Member sees feedback only by checking app manually. |
| **Exception Flow** | **Exception Flow 1: Feedback save failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | PT clicks “Send Feedback.” | Database error occurs. |
| **2** |  | System displays: *“Unable to send feedback. Please try again later.”* |
| **3** |  | Feedback not saved. Error logged. |
| **Exception Flow 2: Member account error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Feedback is ready to send. | System cannot locate Member. |
| **2** |  | System displays: *“Member unavailable. Cannot send feedback.”* |
| **Exception Flow 3: Network Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | PT submits feedback. | System loses connection temporarily. |
| **2** |  | System retries or shows connectivity error. |
| **Priority** | Medium | | |
| **Business Rule** | * Feedback must be linked to a specific completed training session. * Only the assigned PT may send feedback. * Feedback must remain permanently visible to the Member. * Feedback must follow system templates if mandatory (gym policy). * Notification must be sent immediately unless disabled by Member. | | |

### 5.2.31 Use Case UC-31 Manage Branches

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-31 | | |
| **Use Case Name** | Manage Branches | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager manages gym branches in the system. The Manager can create new branches, update existing branch information, or deactivate branches that are no longer operational.  Branch information typically includes branch name, address, contact info, operating hours, and assigned staff. | | |
| **Goal** | To allow the Manager to manage gym branch information in the system. | | |
| **Trigger** | Manager selects “Manage Branches” from the management dashboard. | | |
| **Pre – conditions** | * Manager is authenticated. * Branch Management Module is operational. * Manager has proper authorization to modify branch data. | | |
| **Post – conditions** | Success Post-Conditions   * Branch is created, updated, or deactivated. * Staff, PTs, and Members may see updated branch information. * Booking and scheduling logic respects branch availability.   Failure Post-Condidtions   * No changes are made. * System logs failed attempt. * Manager receives appropriate error message. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Manage Branches page. | System retrieves list of all branches. |
| **2** | Manager selects:   * Add branch * Edit branch * Deactive branch | System displays corresponding form. |
| **3** | Manager enters or updates branch details:   * Name * Address * Operating hours * Contact info * Amenities | System validates input format. |
| **4** | Manager clicks Save. | System performs validation (duplicate name, invalid hours, etc.). |
| **5** |  | System saves changes to branch database |
| **6** |  | System displays confirmation: *“Branch information updated successfully.”* |
| **7** | Manager returns to branch list. |  |
| **Alternative Flow** | **Alternative Flow 1: Manager Deactivates a Branch** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects a branch. |  |
| **2** | Manager clicks “Deactivate Branch.” | System warns: *“This will prevent new bookings.”* |
| **3** | Manager confirms. | Branch status becomes “Inactive.” |
| **Alternative Flow 2: Assigning Staff to Branch** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager selects branch. | System displays staff list. |
| **2** | Manager assigns PTs or Support Staff. | System saves assignment. |
| **Alternative Flow 3: Viewing Branch Insights** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks “View Reports.” | System retrieves branch metrics:   * Active members * PT utilization * Revenue |
| **2** | Manager reviews analytics. |  |
| **Exception Flow** | **Exception Flow 1: Duplicate Branch Name** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager enters duplicate name. | System detects conflict. |
| **2** |  | System displays: *“Branch name already exists.”* |
| **3** | Manager corrects name. | Returns to Step 3. |
| **Exception Flow 2: Invalid Operating Hours** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager enters invalid hours (e.g., closing < opening). | System detects error. |
| **2** |  | System displays: *“Invalid operating hours.”* |
| **Exception Flow 3: Database error** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager saves branch. | Database failure. |
| **2** |  | System shows: *“Unable to save branch information.”* |
| **3** |  | System logs error. |
| **Priority** | Medium | | |
| **Business Rule** | * Branch name must be unique. * Only Managers may modify branch data. * Deactivated branches cannot accept new bookings. * Branch must have valid operating hours. * Member and PT assignments must correspond to active branches only. | | |

### 5.2.32 Use Case UC-32 Manage Membership Packages

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-32 | | |
| **Use Case Name** | Manage Membership Packages | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager manages the gym’s membership packages. Actions include creating new packages, editing package details, updating pricing, enabling or disabling packages, and adjusting package duration or benefits. | | |
| **Goal** | To allow the Manager to maintain accurate, up-to-date membership package offerings for Members. | | |
| **Trigger** | Manager selects “Manage Membership Packages” from the management dashboard. | | |
| **Pre – conditions** | * Manager is authenticated. * Membership Management Module is operational. * Manager has permission to modify membership packages. | | |
| **Post – conditions** | Success Post-Conditions   * Membership package changes are saved. * Updated packages become visible to Members (UC-05 View Membership Packages). * Updated prices are applied to future purchases (UC-06 Purchase Membership).   Failure Post-Condidtions   * No changes saved. * System logs error. * Manager notified of failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens “Manage Membership Packages.” | System retrieves current list of membership packages. |
| **2** | Manager chooses an action:   * Add new package * Edit existing package * Disable/ Enable package | System loads appropriate form. |
| **3** | Manager chooses an action:   * Package name * Duration (monthly, quarterly/ yearly) * Price * Benefits included * Description | System validates data. |
| **4** | Manager clicks Save. | System performs verification (duplicate name, invalid data). |
| **5** |  | System saves the updated package information. |
| **6** |  | System displays: *“Membership package updated successfully.”* |
| **7** | Manager returns to package list. |  |
| **Alternative Flow** | **Alternative Flow 1: Manager Disables a Package** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects an existing package. | System shows detailed package info. |
| **2** | Manager clicks “Disable Package”. | System warns: *“Disabled packages cannot be purchased.”* |
| **3** | Manager confirms. | System changes package status to “Inactive.” |
| **Alternative Flow 2: Duplicate Existing Package as Template** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager selects “Duplicate Package.” | System copies all fields except name. |
| **2** | Manager updates new name. | Flow continues to Main Flow Step 3. |
| **Exception Flow** | **Exception Flow 1: Duplicate Package Name** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager enters a package name already in use. | System detects conflict. |
| **2** |  | System displays: *“Membership package name already exists.”* |
| **3** | Manager corrects name. | Flow returns to Step 3. |
| **Exception Flow 2: Invalid price** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager enters invalid price (negative, zero). | System validates pricing. |
| **2** |  | System displays: *“Invalid price. Price must be greater than zero.”* |
| **Exception Flow 3: Database or Service Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks Save. | System cannot write to database. |
| **2** |  | System displays: *“Failed to update membership package.”* |
| **3** |  | System logs error. |
| **Priority** | High | | |
| **Business Rule** | * Package names must be unique. * Disabled packages must not be visible in UC-06 purchase workflow. * Price updates apply only to new purchases, not active memberships. * Packages must define clear duration in days/months/years. * All package changes must be timestamped for audit trails. | | |

### 5.2.33 Use Case UC-33 Process Refund Requests

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-33 | | |
| **Use Case Name** | Process Refund Requests | | |
| **Created by** | Van Nguyen | **Last updated by** | Van Nguyen |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager reviews and processes refund requests submitted by Members. A refund request may arise from mistaken payments, duplicated charges, service issues, or system errors. Manager decides to approve or deny the refund. If approved, the system initiates refund processing through the Payment Gateway. | | |
| **Goal** | To ensure proper review and handling of Member refund requests in compliance with gym policies and payment rules. | | |
| **Trigger** | Manager opens “Refund Requests” from the Finance or Payment Management section. | | |
| **Pre – conditions** | * Manager is authenticated. * Refund requests exist in the system (generated via UC-06 Purchase/Renew Membership or from Member contacting support). * Payment Gateway must be operational for online refunds. * Original transaction must exist. | | |
| **Post – conditions** | Success Post-Conditions (Approved)   * Refund is triggered through Payment Gateway. * Member is notified of approval. * Refund status changes to Approved / Refunded. * Financial logs updated.   Success Post-Conditions (Rejected)   * Refund request marked Rejected. * Member receives notification with reason (optional).   Failure Post-Condidtions   * Refund request remains Pending. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Refund Requests list. | System loads all pending refund requests. |
| **2** | Manager selects a specific refund request. | System displays:   * Member details * Transaction ID * Amount * Package purchased * Date/ time of transaction * Member’s reason for refund. |
| **3** | Manager reviews request and supporting info. |  |
| **4** | Manager chooses Approve or Reject. | System prepares required validation. |
| **Main Flow 1: Approve refund** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks “Approve Refund” | System validates original transaction. |
| **2** |  | System sends refund request to Payment Gateway. |
| **3** | Payment Gateway processes refund. | Returns success status. |
| **4** |  | System updates refund status → Refunded. |
| **5** |  | System notifies Member of successful refund. |
| **Main Flow 2: Reject Refund** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks “Reject Refund.” | System prompts Manager for optional rejection reason. |
| **2** | Manager enters reason and confirms. | System sets refund status → Rejected. |
| **3** |  | System notifies Member: *“Your refund request was rejected.”* |
| **Alternative Flow** | **Alternative Flow 1: Partial Refund** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager chooses “Partial Refund” | System displays partial refund amount input. |
| **2** | Manager enters amount. | System validates amount. |
| **3** |  | System sends partial refund request to Payment Gateway |
| **4** |  | Flow continues to Main Flow |
| **Exception Flow** | **Exception Flow 1: Payment Gateway Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** |  | System sends refund request. |
| **2** | Payment Gateway returns error. | System displays: *“Refund failed. Please try again later.”* |
| **3** |  | Refund remains Pending. |
| **Exception Flow 2: Transaction Not Found** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager reviews request | System cannot find the original payment record |
| **2** |  | System displays: *“Original transaction not found.”* |
| **Exception Flow 3: Database Error during Save** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager approves or rejects refund. | System fails to update record. |
| **2** | System displays error and logs failure. |  |
| **Priority** | High | | |
| **Business Rule** | * Only Managers may approve or reject refund requests. * Refund amount cannot exceed the original transaction amount. * Approved refunds must be processed only via Payment Gateway. * Refund requests must be stored with audit logs. * Rejected refunds must include a reason. | | |

### 5.2.34 Use Case UC-34 View Financial Reports

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-34 | | |
| **Use Case Name** | View Financial Reports | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager views financial reports generated by the system. Reports include revenue summaries, membership sales, payment transaction logs, refund statistics, offline and online payment segmentation, and branch-level financial breakdowns.  These reports help Managers track business performance and make informed decisions. | | |
| **Goal** | To provide Managers with accurate and up-to-date financial insights for operational and strategic decision-making. | | |
| **Trigger** | To provide Managers with accurate and up-to-date financial insights for operational and strategic decision-making. | | |
| **Pre – conditions** | * Manager is authenticated. * Reporting Engine is operational. * Financial data exists in the system. * Manager has permission to access financial data. | | |
| **Post – conditions** | Success Post-Conditions   * Manager sees the financial metrics and charts. * Manager may export the report. * Manager may filter by date range, branch, and package.   Failure Post-Condidtions   * No data shown. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Financial Reports. | System retrieves default report (e.g., current month). |
| **2** | Manager selects report type:   * Revenue reports * Membership sales report * Payment transactions * Refund statistics * Branch Revenue comparison | System loads requested dataset. |
| **3** | Manager optionally applies filters:   * Date range * Branch * Payment method * Membership type | System recalculates data. |
| **4** | Manager views charts/ tables:   * Line charts for revenue trends * Bar charts for branch comparison * Pie charts for payment types. | System displays formatted visuals. |
| **5** | Manager may click “View Details” for specific entries (e.g., transaction list). | System loads itemized breakdown. |
| **6** | Manager may export report (PDF/ CSV). | System generates export file. |
| **7** | Manager logs out or returns to dashboard. | End of use case. |
| **Alternative Flow** | **Alternative Flow 1: Manager exports reports** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager clicks Export. | System prompts: PDF or CSV. |
| **2** | Manager selects format. | System generates file. |
| **3** | Manager downloads the report. | Flow returns to Step 7 |
| **Alternative Flow 2: Drill-Down Analysis** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager clicks specific data point. | System retrieves deeper details. |
| **2** | Manager views detailed sub-report. | Flow continues to Step 5. |
| **Exception Flow** | **Exception Flow 1: Insufficient Permissions** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager tries to open financial report. | System checks permissions. |
| **2** |  | System displays: *“Access denied.”* |
| **Exception Flow 2: No Financial Data Available** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects a date range with no data. | System detects empty dataset. |
| **2** |  | System displays: *“No data available for the selected period.”* |
| **Exception Flow 3: Reporting Engine Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects report type | System fails to generate report |
| **2** |  | System shows: *“Unable to generate report.”* |
| **3** |  | System logs error. |
| **Priority** | High | | |
| **Business Rule** | * Only Managers may view or export financial reports. * Financial data must be aggregated in real time or near real time. * Refunds (UC-33) must be reflected accurately in financial reports * All exported reports must include timestamp and Manager ID. * Branch-based reporting must reflect active branch assignments. | | |

### 5.2.35 Use Case UC-35 Trainer Performance Reports

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-35 | | |
| **Use Case Name** | Trainer Performance Reports | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager views performance reports for Personal Trainers. Reports include metrics such as:   * number of sessions conducted * average Member satisfaction rating * Member progress improvement indicators * punctuality and attendance * cancellation/no-show rates * feedback quality   These insights help Managers evaluate PT performance, optimize scheduling, and identify areas for professional development. | | |
| **Goal** | To provide the Manager with comprehensive PT performance analytics for evaluation and operational decision-making. | | |
| **Trigger** | Manager selects “Trainer Performance Reports” from the Reporting dashboard. | | |
| **Pre – conditions** | * Manager is authenticated. * Reporting Engine and analytics modules are operational. * Session and progress data exist in the system. | | |
| **Post – conditions** | Success Post-Conditions   * Manager views PT performance statistics. * Manager may export data for HR or evaluation purposes. * Insights may be used to adjust staffing or training plans.   Failure Post-Condidtions   * No report is shown. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Trainer Performance Reports. | System loads key PT metrics. |
| **2** | Manager filters reports by date range, PT name, or branch. | System updates displayed metrics. |
| **3** | Manager views detailed PT metrics:   * Total session conducted * Acceptance/ decline rate (UC-26) * On-time start percentage * Member progress improvement (from UC-29) * Member feedback (from UC-30) * No-show or cacellation frequency. | System displays charts and tables. |
| **4** | Manager selects a PT to see full report. | System shows PT-specific performance dashboard. |
| **5** | Manager may export the report. | Export file generated (PDF/ CSV). |
| **6** | Manager logs out or continues to another task. | End UC. |
| **Alternative Flow** | **Alternative Flow 1: Manager Views Trend Analysis Over Time** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects “Trend Mode.” | System shows performance trends over months/years. |
| **Exception Flow** | **Exception Flow 1: No Performance Data Found** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects a PT with no activity. | System detects empty dataset. |
| **2** |  | System displays: *“No performance data available for this trainer.”* |
| **Exception Flow 2: Reporting Engine Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects report type. | System fails to load or calculate metrics. |
| **2** |  | System shows: *“Unable to generate report.”* |
| **Exception Flow 3: Insufficient Permissions** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Unauthorized user tries to access report. | System checks permissions. |
| **2** |  | System displays: *“Access Denied: Manager Role Required.” System displays: “Access Denied: Manager Role Required.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Only Managers may access PT performance reports. * Metrics must be calculated using complete, accurate session data. * Feedback from UC-30 must be linked to the correct PT and session. * PT comparison must follow standardized KPIs. * Reports must include timestamp and data source transparency. | | |

### 5.2.36 Use Case Member Activity Reports

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-36 | | |
| **Use Case Name** | Member Activity Reports | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager views detailed activity reports for Members. Reports include:   * Check-in frequency * Training session attendance (PT & non-PT sessions) * Membership usage metrics * Body metrics trends (weight, BMI, fat %, etc.) * Progress over time * AI analysis (optional) * Engagement indicators (e.g., missed sessions, irregular activity)   These insights help Managers understand member engagement, retention risks, and performance. | | |
| **Goal** | To allow Managers to evaluate Member activity and engagement for operational improvements and customer success strategies. | | |
| **Trigger** | Manager selects “Member Activity Reports” from the reporting dashboard. | | |
| **Pre – conditions** | * Manager is authenticated. * Reporting & Data Tracking Modules are operational. * Member data exists (check-in logs, session data, progress metrics, etc.). | | |
| **Post – conditions** | Success Post-Conditions   * Manager views member activity metrics and charts. * Insights may be exported or used for decision-making. * Data may be used for identifying low activity or churn risk.   Failure Post-Condidtions   * Reports not shown. * System logs error. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Member Activity Reports. | System loads default activity summary. |
| **2** | Manager filters by:   * Member * Branch * Date range * Membership type | System refreshes data results. |
| **3** | Manager views activity dashboard, including:   * Total check-ins * Attendance rate * Missed/ Canceled sessions * PT session participation * Body metrics trends (from UC-16) * AI alerts (injury-risk, fatigue level-optional) | System shows visual charts and tables. |
| **4** | Manager selects a Member for detailed view. | System loads complete timeline of member activity. |
| **5** | Manager optionally downloads report. | System generates PDF/CSV export. |
| **6** | Manager returns to dashboard or logs out. | End of use case. |
| **Alternative Flow** | **Alternative Flow 1: Manager Views Summary Insights Only** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects “Summary Mode.” | System displays overall activity health of members. |
| **2** | Manager views KPIs such as:   * Average check-ins/week * Inactive members * Members at risk of churn |  |
| **Alternative Flow 2: Compare Activity Between Member Groups** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager selects “Compare Groups.” | System prompts selection of two or more member segments. |
| **2** | Manager chooses groups (e.g., Beginners vs Advanced). | System shows comparison charts. |
| **Exception Flow** | **Exception Flow 1: No Activity Data Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects a member with no tracked activity. | System detects missing data. |
| **2** |  | System displays: *“No activity data available for this member.”* |
| **Exception Flow 2: Reporting System Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects report. | Reporting Engine fails. |
| **2** |  | System displays error and logs issue. |
| **Priority** | Medium | | |
| **Business Rule** | * Only Managers may view full member activity reports. * Reports must reflect accurate real-time or near real-time data. * Member progress metrics must sync with UC-29 and UC-16. * Sensitive health data must be masked unless needed for reporting. * Exported reports must be timestamped for auditing purposes. | | |

### 5.2.37 Use Case UC-37 Manage Staff Accounts

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-37 | | |
| **Use Case Name** | Manage Staff Accounts | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how the Manager manages staff accounts in the system. Actions include:   * Creating new PT or Support Staff accounts * Updating account information * Assigning staff to branches * Changing roles (PT ↔ Support Staff) * Deactivating accounts (when staff leave the gym)   Since the system has no Admin, Manager performs all staff management tasks. | | |
| **Goal** | To allow the Manager to maintain accurate, secure staff account records. | | |
| **Trigger** | Manager selects “Manage Staff Accounts” from the management dashboard. | | |
| **Pre – conditions** | * Manager is authenticated. * Authentication & User Management modules are operational. * If editing or deactivating: staff account must exist. | | |
| **Post – conditions** | Success Post-Conditions   * Staff accounts are created, updated, or deactivated. * Login access is enabled/disabled based on status. * Updated information is visible to Members (PT schedules), Support Staff system, and Manager tools.   Failure Post-Condidtions   * No update is made. * System logs error. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Manage Staff Accounts. | System displays list of all PTs and Support Staff. |
| **2** | Manager selects an action:   * Create new staff account * Edit staff account * Deactive account | System loads corresponding form. |
| **3** | Manager enters staff details:   * Name * Email * Phone * Role (PT/ Support Staff) * Assigned branch * Hourly rate or contact info. | System validates input. |
| **4** | Manager clicks Save. | System updates or creates staff profile in database. |
| **5** |  | System displays confirmation: *“Staff account updated successfully.”* |
| **6** | Staff appears in list. |  |
| **Alternative Flow** | **Alternative Flow 1: Create New Staff Account** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager selects “Add New Staff.” | System opens blank form. |
| **2** | Manager fills all required fields. | System validates data. |
| **3** | Manager saves. | System sends activation email to staff member. |
| **Alternative Flow 2: Assign Staff to Branch** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager selects staff account. | Staff profile opens. |
| **2** | Manager assigns or changes branch. | System updates staff-branch relationship. |
| **Alternative Flow 3: Deactivate Staff Account** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects staff account. | Profile loads. |
| **2** | Manager clicks “Deactivate Account.” | System warns: *“ Staff will lose login access. Pending sessions must be reassigned.”* |
| **3** | Manager confirms. | System sets account status to Inactive. |
| **Exception Flow** | **Exception Flow 1: Email Already Exists** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager enters email for new staff. | System detects duplicate. |
| **2** |  | System displays: *“Email already in use.”* |
| **Exception Flow 2: Branch Not Found** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager assigns staff to nonexistent branch. | System error. |
| **2** |  | System displays: *“Invalid branch selected.”* |
| **Exception Flow 3: Database Save Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks Save. | Database connection fails. |
| **2** |  | System alerts: *“Unable to save changes.”* |
| **Priority** | High | | |
| **Business Rule** | * Only Managers may create or modify staff accounts. * Every staff must belong to exactly one branch. * Deactivated accounts must not allow login or booking involvement. * Email must be unique across all users. | | |

### 5.2.38 Use Case UC-38 Respond to Tickets

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-38 | | |
| **Use Case Name** | Respond to Tickets | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Support Staff | | |
| **Brief Description** | This use case describes how Support Staff review support tickets submitted by Members, respond to those tickets, update ticket statuses, and assist Members with their issues. If Support Staff cannot resolve the issue, they may escalate the ticket to a Manager (UC-39). | | |
| **Goal** | To allow Support Staff to efficiently manage and resolve Member support tickets. | | |
| **Trigger** | Support Staff selects “Ticket Management” or “Open Tickets” from Support dashboard. | | |
| **Pre – conditions** | * Support Staff is authenticated. * At least one ticket exists. * Ticket Management Module is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Ticket is responded to and status updated. * Member is notified of progress or resolution.   Failure Post-Condidtions   * Ticket remains unresolved. * System logs error. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff opens Ticket List. | System displays all open tickets with filters (priority, date, type, status). |
| **2** | Support Staff selects a ticket. | System loads full ticket details:   * Member info * Issue description * Attachments * Ticket category * Priority level |
| **3** | Support Staff writes a response to Member. | System validates required fields. |
| **4** | Support Staff clicks “Send Response.” | System updates ticket status to In Progress or Resolved. |
| **5** |  | System sends notification to Member. |
| **6** | Support Staff closes the ticket or continues follow-up as needed. | Ticket status updated accordingly. |
| **Alternative Flow** | **Alternative Flow 1: Request Additional Information from Member** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff selects “Request More Info.” | System sends request to Member. |
| **2** | Member submits more information. | Ticket updates and notifies Support Staff. |
| **3** | Support Staff continues handling. | Returns to Main Flow Step 3. |
| **Alternative Flow 2: Reassign Ticket to Another Support Staff** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Support Staff selects “Assign to Another Agent.” | System shows list of Staff. |
| **2** | Support Staff chooses target agent. | Ticket ownership transferred. |
| **Exception Flow** | **Exception Flow 1: Ticket Not Found** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff selects a ticket. | Ticket ID not found (deleted or completed). |
| **2** |  | System displays: *“Ticket no longer available.”* |
| **Exception Flow 2: Database Error While Updating Ticket** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff sends response. | System fails to save |
| **2** |  | System displays: *“Unable to update ticket. Please try again later.”* |
| **Exception Flow 3: Notification Service Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff responds to ticket. | Notification service unavailable. |
| **2** |  | System logs failure and displays warning: *“Response saved but Member notification failed.”* |
| **Priority** | High | | |
| **Business Rule** | * Only Support Staff and Managers may respond to support tickets. * Ticket responses must be timestamped. * Ticket status must always reflect current handling stage * Member must be notified of any status or response update. | | |

### 5.2.39 Use Case UC-39 Escalate Ticket

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-39 | | |
| **Use Case Name** | Escalate Ticket | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Support Staff | | |
| **Brief Description** | This use case describes how Support Staff escalates an unresolved or complex ticket to a Manager. Escalation is used when the issue is beyond the Support Staff’s authority or requires managerial intervention (e.g., billing disputes, refund issues, policy exceptions). | | |
| **Goal** | To ensure difficult or high-priority tickets are escalated for faster and more authoritative resolution. | | |
| **Trigger** | Triggered during UC-38 Respond to Tickets, when Support Staff selects “Escalate Ticket.” | | |
| **Pre – conditions** | * Support Staff is authenticated. * Ticket exists and is currently Open or In Progress. * Ticket has not already been escalated. * Manager user account exists and is active. | | |
| **Post – conditions** | Success Post-Conditions   * Ticket status becomes Escalated. * Manager receives notification and gains ownership of ticket. * Member receives update that ticket is being escalated.   Failure Post-Condidtions   * Ticket remains un-escalated. * System logs failure. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff clicks “Escalate Ticket.” | System opens escalation form. |
| **2** | Support Staff enters escalation reason (mandatory). | System validates input. |
| **3** | Support Staff confirms escalation. | System updates ticket status to Escalated. |
| **4** |  | System assigns ticket to Manager. |
| **5** |  | System notifies Manager: *“A ticket has been escalated to you.”* |
| **6** |  | System notifies Member: *“Your support ticket has been escalated for further review.”* |
| **7** | Support Staff returns to ticket list. | Ticket no longer appears under Support Staff queue. |
| **Alternative Flow** | **Alternative Flow 1: Add Supporting Files Before Escalation** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff uploads screenshots/logs. | System validates attachments. |
| **2** | Staff completes escalation. | Flow continues to Main Step 3. |
| **Exception Flow** | **Exception Flow 1: Manager Not Available** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff confirms escalation. | System detects no active Manager online |
| **2** |  | System displays: *“No Manager available. Escalation pending.”* |
| **3** |  | Ticket marked as Pending Escalation. |
| **Exception Flow 2: Ticket Already Escalated** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff attempts to escalate. | System detects escalated status. |
| **2** |  | System displays: *“Ticket has already been escalated.”* |
| **Exception Flow 3: Database Error During Escalation** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Staff confirms escalation. | System fails to save escalation status. |
| **2** |  | System displays error msg and logs issue. |
| **Priority** | Medium | | |
| **Business Rule** | * Only Support Staff may escalate tickets. * Escalation requires a mandatory reason. * Escalated tickets must be immediately assigned to a Manager. * Escalation actions must be timestamped for auditing. * Member must be notified of escalation status. | | |

### 5.2.40 Use Case UC-40 Provide Live Chat Support

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-40 | | |
| **Use Case Name** | Provide Live Chat Support | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Support Staff | | |
| **Brief Description** | This use case describes how Support Staff provides real-time chat support to Members. The Support Staff receives incoming chat sessions from Members, joins the conversation, replies to messages, manages multiple chat sessions if needed, and may escalate issues to Manager via UC-39. | | |
| **Goal** | To enable Support Staff to assist Members instantly through a live chat interface. | | |
| **Trigger** | Member initiates a chat session in UC-23 Live Chat Support, and the system assigns the chat to an available Support Staff. | | |
| **Pre – conditions** | * Support Staff is authenticated. * Support Staff is marked “Online” or “Available for Chat.” * There is at least one Member waiting for support. * Live Chat Module is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Chat session is completed. * Member receives the necessary support. * Chat history is saved.   Failure Post-Condidtions   * Chat session is not handled properly. * System logs failure. * Member may be disconnected or transferred. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff receives notification of a new chat request. | System assigns chat to available Support Staff. |
| **2** | Support Staff accepts the chat. | System opens live chat interface. |
| **3** | Support Staff reads Member’s initial message. | System displays full chat history (if any). |
| **4** | Support Staff types and sends a reply. | System delivers message to Member in real time. |
| **5** | Support Staff continues conversation to resolve inquiry. | System updates chat timeline. |
| **6** | Support Staff marks chat as Resolved or Requires Follow-up. | Chat session ends or is stored for continuation. |
| **7** | System saves full chat transcript. | End of session. |
| **Alternative Flow** | **Alternative Flow 1: Support Staff Handles Multiple Chats Simultaneously** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff switches between active sessions. | System highlights unread messages. |
| **2** | Staff responds to each Member accordingly. | System preserves chat states. |
| **Alternative Flow 2: Escalate Chat to Manager** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Support Staff selects “Escalate to Manager.” | System requests escalation reason. |
| **2** | Staff submits reason. | Chat is transferred to Manager for handling. |
| **Exception Flow** | **Exception Flow 1: Member Disconnects Unexpectedly** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Staff writes a reply. | System shows: “Member offline.” |
| **2** |  | Chat remains open until timeout or closure. |
| **Exception Flow 2: Chat Server Connection Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Staff tries sending messages | System cannot deliver messages |
| **2** |  | System displays: “Connection lost. Reconnecting…” |
| **Exception Flow 3: Notification Service Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Chat assigned to Support Staff. | Notification fails. |
| **2** |  | System still assigns chat but shows warning. |
| **Priority** | High | | |
| **Business Rule** | * Only available Support Staff may receive live chat requests. * Chat history must be saved for audit and QA. * Staff must respond within acceptable SLA (e.g., < 30 seconds). * Escalation must follow standard procedures (UC-39). * Staff must maintain professionalism in communication. | | |

### 5.2.41 Use Case UC-41 Manage Knowledge Base Articles

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-41 | | |
| **Use Case Name** | Manage Knowledge Base Articles | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Support Staff | | |
| **Brief Description** | This use case describes how Support Staff create, update, publish, and archive Knowledge Base (KB) articles. KB articles help Members find self-service solutions for common questions, reducing support workload and improving response efficiency. | | |
| **Goal** | To allow Support Staff to maintain up-to-date, accurate, helpful Knowledge Base articles for Members. | | |
| **Trigger** | Support Staff selects “Knowledge Base Management” from Support dashboard. | | |
| **Pre – conditions** | * Support Staff is authenticated. * Knowledge Base module is operational. * If editing or archiving: article must exist. | | |
| **Post – conditions** | Success Post-Conditions   * Articles are created, updated, or archived. * Published articles become visible to Members. * Archived articles no longer appear in search but remain stored.   Failure Post-Condidtions   * No changes saved. * System logs error. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff opens Knowledge Base Management. | System displays list of all KB articles (Active + Archived). |
| **2** | Staff selects:   * Create new article * Edit article * Publish article * Archive article | System opens appropriate form/UI. |
| **3** | Staff fills in article fields:   * Title * Category * Content/ Instructions * Tags/ Keyworks * Related FAQs | System validates required fields. |
| **4** | Staff clicks Save or Publish. | System saves the changes. |
| **5** |  | System displays confirmation: “Article updated successfully.” |
| **6** |  | Article becomes available (if published). |
| **Alternative Flow** | **Alternative Flow 1: Create Draft Article** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Staff selects “Save as Draft.” | System saves draft status. |
| **2** | Staff returns later to edit and publish. | Continues from Main Flow Step 2. |  |
| **Alternative Flow 2: Archive Article** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Staff clicks “Archive.” | System confirms action. |
| **2** | Staff confirms. | System marks article as Archived. |
| **3** | Archived article becomes unavailable to Members. |  |
| **Alternative Flow 3: View Member Feedback on Articles** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Staff selects an article. | System shows Member feedback: Upvotes/downvotes “Helpful” ratings Comments |
| **2** | Staff updates article based on feedback. | Continues to Main Flow Step 3. |
| **Exception Flow** | **Exception Flow 1: Duplicate Article Title** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Staff enters title that already exists | System detects duplicate. |
| **2** |  | System displays: *“Article title already exists.”* |
| **Exception Flow 2: Missing Required Fields** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Staff tries to publish with empty fields. | System checks fields. |
| **2** |  | System displays: *“Please complete all required fields.”* |
| **Exception Flow 3: Database Save Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Staff saves/publishes article. | Database error. |
| **2** |  | System shows: *“Unable to save article. Please try again later.”* |
| **Priority** | High | | |
| **Business Rule** | * Only Support Staff and Managers may modify KB articles. * KB articles must be categorized for search efficiency. * Archived articles must remain stored for audit purposes. * Each article must have a unique title. * Published articles must be accessible to Members 24/7. | | |

### 5.2.42 Use Case UC-42 View Support Analytics Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-42 | | |
| **Use Case Name** | View Support Analytics Dashboard | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Support Staff | | |
| **Brief Description** | This use case describes how Support Staff (and optionally Managers) view analytics related to customer support performance. The dashboard includes metrics such as:   * Number of tickets received * Respond time * Resolution time * SLA compliance * Escalation rate * Ticket categories distribution * Live chat session statistics * Knowledge Base article helpfulness   Support analytics help staff understand workload, performance,and support quality trends. | | |
| **Goal** | To provide actionable insights into support operations, enabling better performance, workload balancing, and service quality. | | |
| **Trigger** | Support Staff selects “Support Analytics Dashboard” from Support menu. | | |
| **Pre – conditions** | * Support Staff or Manager is authenticated. * Enough data exists (tickets, chat logs, resolution statistics). * Reporting Engine is operational. | | |
| **Post – conditions** | Success Post-Conditions   * Dashboard is displayed with accurate real-time metrics. * Support Staff may identify workload bottlenecks. * Manager may export or use data for performance evaluation.   Failure Post-Condidtions   * Dashboard not loaded. * System logs error. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff opens Support Analytics Dashboard. | System retrieves support-related data. |
| **2** | Support Staff selects a date range (e.g., Today / Last 7 days / Last month). | System recalculates metrics. |
| **3** | Support Staff views dashboard widgets, such as:   * Total tickets created * Ticket resolved * Average first response time * Average resolution time * SLA compliance rate * Escalate rate (UC-39) * Ticket volume by category * Chat session stats (from UC-40) * KB article helpfulness (UC-41) | System displays charts and tables. |
| **4** | Support Staff selects a metric to drill down. | System shows detailed report for that metric. |
| **5** | Support Staff may export results. | System generates PDF/CSV file. |
| **6** | Support Staff returns to dashboard or exits. | End of UC. |
| **Alternative Flow** | **Alternative Flow 1: Filter by Support Staff Member** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff selects “Filter by Agent.” | System shows list of Support Staff. |
| **2** | Support Staff selects a name. | System refreshes analytics for selected agent. |
| **Alternative Flow 2: View Weekly or Monthly Trends** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Support Staff selects “Trend Mode.” | System shows multi-period comparison graphs. |
| **Alternative Flow 3: Identify High-Risk Tickets** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff selects “Critical Tickets.” | System filters tickets with SLA warnings or escalations. |
| **Exception Flow** | **Exception Flow 1: Not Enough Data to Generate Report** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Support Staff selects dashboard. | System detects insufficient ticket/chat data. |
| **2** |  | System displays: *“No data available for selected period.”* |
| **Exception Flow 2: Reporting Engine Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Support Staff requests metrics. | System cannot compute results. |
| **2** |  | System displays: *“Unable to load analytics.”* |
| **Priority** | Medium | | |
| **Business Rule** | * Only Support Staff and Managers may access support analytics. * SLA metrics must be calculated using real response timestamps. * Escalations must be included in analytics (from UC-39). * Chat statistics must include handling time, satisfaction * Reports must maintain data integrity and accurate time zones. | | |

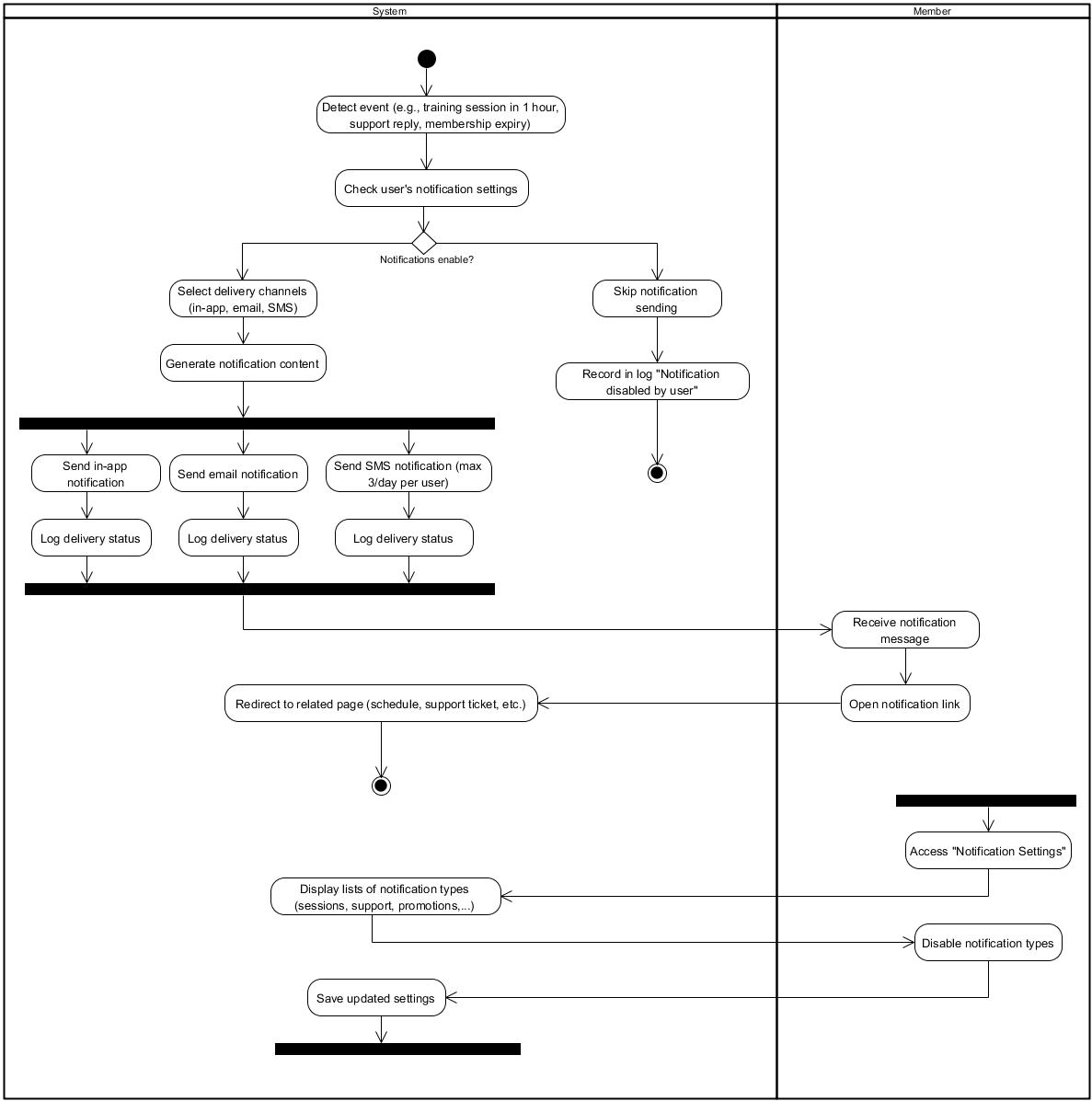
### 5.2.43 Use Case UC-43 Process Online Payment

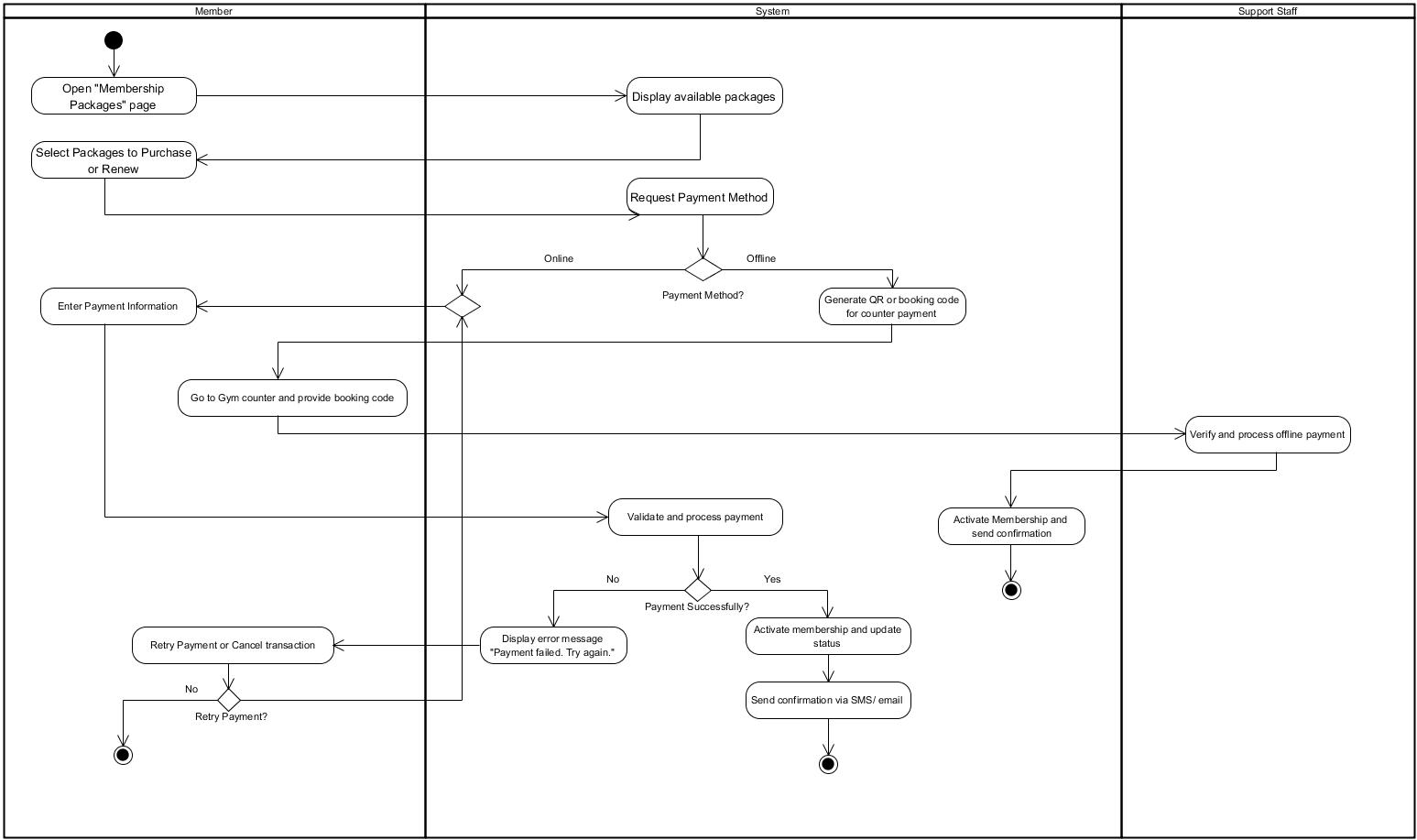
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-43 | | |
| **Use Case Name** | Process Online Payment | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Member | | |
| **Brief Description** | This use case describes the full flow of processing an online payment for a membership purchase or renewal (UC-06). It includes redirecting the Member to an external Payment Gateway, validating payment results, updating their membership status, generating receipts, and saving transaction logs. | | |
| **Goal** | To successfully process a secure online payment and activate the Member’s membership package. | | |
| **Trigger** | Member selects “Pay Online” during UC-06 Purchase/Renew Membership. | | |
| **Pre – conditions** | * Member is authenticated. * Membership package exists and is valid. * Payment Gateway service is available. * Member has a valid payment method (card, e-wallet, etc.). | | |
| **Post – conditions** | Success Post-Conditions   * Payment is completed and verified. * Membership becomes Active or Renewed. * Invoice is generated and saved. * Member receives confirmation notification.   Failure Post-Condidtions   * Membership remains unchanged. * Payment record marked as Failed. * Member receives failure notification. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Member confirms purchase and selects Online Payment. | System creates a pending transaction. |
| **2** |  | System redirects Member to Payment Gateway with transaction details. |
| **3** | Member enters payment information on Payment Gateway. | Payment Gateway validates info. |
| **4** | Payment Gateway processes transaction. | Returns success status to system. |
| **5** | System receives callback from Payment Gateway. | System verifies signature and transaction token. |
| **6** |  | System marks transaction as Successful |
| **7** |  | System updates Membership Status (Activate or Renew). |
| **8** |  | System generates invoice/receipt and logs transaction. |
| **9** |  | System notifies Member: *“Payment Successful.”* |
| **10** | Member is redirected to confirmation screen. | End UC. |
| **Alternative Flow** | **Alternative Flow 1: Payment Gateway Returns Failure** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Payment Gateway rejects card or fails charge. | System marks transaction as Failed. |
| **2** |  | System notifies Member: “Payment Failed. Please try again.” |
| **3** |  | Member returns to UC-06 to retry payment. |
| **Alternative Flow 2: Member Cancels Payment Midway** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Member closes Payment Gateway page. | Payment Gateway sends cancel status. |
| **2** |  | System marks transaction as Cancelled. |
| **3** |  | System notifies Member. |
| **Alternative Flow 3: Payment Gateway Timeout** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Payment Gateway times out. | System receives timeout callback. |
| **2** |  | System marks transaction as Timeout. |
| **3** | Member prompted to try again. |  |
| **Alternative Flow 4: Insufficient Funds / Authentication Failure** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Gateway declines payment. | System logs failure reason. |
| **2** |  | System notifies Member. |
| **Exception Flow** | **Exception Flow 1: System Cannot Reach Payment Gateway** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Member selects Online Payment. | System attempts to connect to gateway. |
| **2** |  | Connection fails → System displays *“Payment service unavailable.”* |
| **Exception Flow 2: Callback Signature Invalid (Security Issue)** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Gateway sends callback. | System detects invalid signature. |
| **2** |  | System rejects callback and logs potential fraud. |
| **Exception Flow 3: Database Error When Updating Membership** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Payment successful. | System tries to activate membership. |
| **2** |  | Database error occurs. |
| **3** |  | System marks transaction as Error and notifies support/admin. |
| **Priority** | High | | |
| **Business Rule** | * Payment must be authenticated by a secure Payment Gateway. * Membership activation happens only after confirmed successful payment. * All financial transactions must be logged for auditing. * Only valid membership packages may be purchased. * Refund requests must follow UC-33 rules. | | |

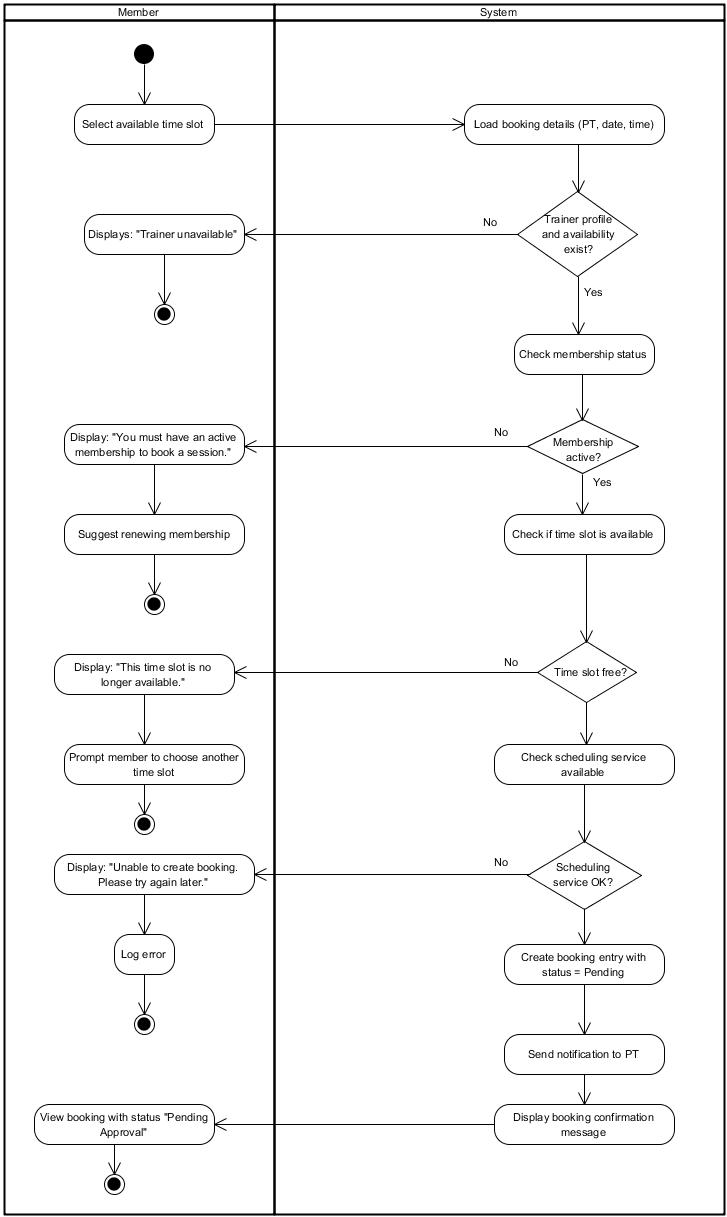
### 5.2.44 Use Case UC-44 Confirm Offline Payment

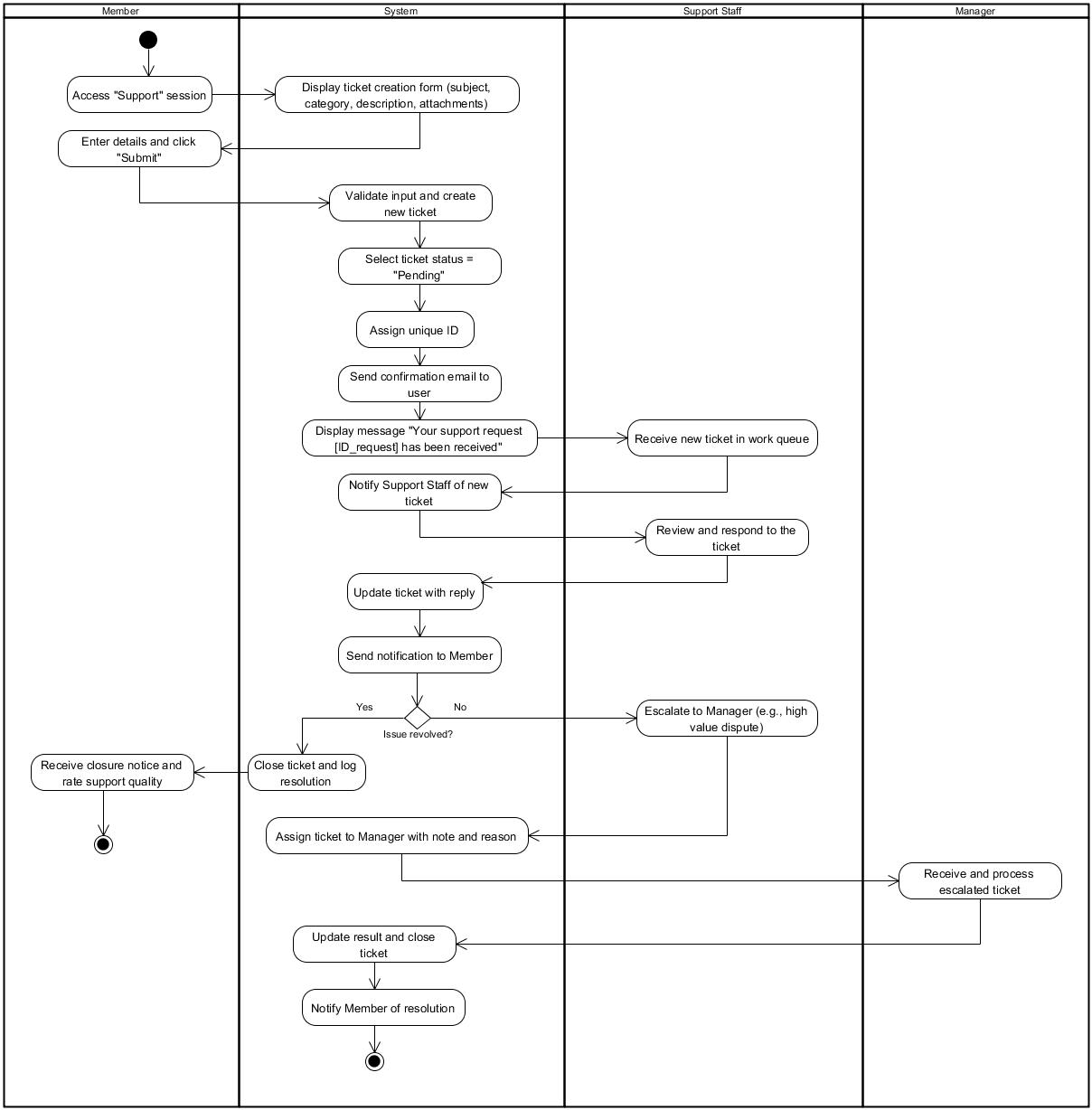
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-44 | | |
| **Use Case Name** | Confirm Offline Payment | | |
| **Created by** | Tuyen Pham | **Last updated by** | Tuyen Pham |
| **Date Created** | Sept 13, 2025 | **Date last updated** | Sept 13, 2025 |
| **Actors** | Manager | | |
| **Brief Description** | This use case describes how a Manager confirms an offline payment made by a Member. Offline payments include:   * Cash at the Gym counter * Bank transfer * POS machine/ QR code payment * Manual adjustments (e.g., staff-internal arrangements)   Once confirmed, the Member’s membership becomes active or extended, and a receipt is generated. | | |
| **Goal** | To validate and record offline payments and activate or renew a Member’s membership package. | | |
| **Trigger** | Member completes an offline payment and staff submits a pending payment record OR Manager manually creates a new offline payment entry. | | |
| **Pre – conditions** | * Manager is authenticated. * Member has selected a membership package. * Offline payment proof exists (cash receipt, transaction ID, transfer screenshot, etc.). * Membership package details are valid. | | |
| **Post – conditions** | Success Post-Conditions   * Offline payment is marked Confirmed. * Membership becomes Active or Renewed. * Transaction recorded in system logs. * Member receives notification and invoice.   Failure Post-Condidtions   * Payment remains Pending. * Membership remains inactive/unrenewed. * Error is logged. | | |
| **Main Flow** | **Step** | **Actor Action** | **System Response** |
| **1** | Manager opens Offline Payment Requests. | System displays list of pending offline payments. |
| **2** | Manager selects a specific payment entry. | System shows:   * Member info * Package purchased * Payment method (cash/ transfer/ POS) * Amount * Uploaded payment proof |
| **3** | Manager reviews information and payment evidence. |  |
| **4** | Manager clicks Confirm Payment. | System validates payment entry. |
| **5** |  | System marks payment as Confirmed. |
| **6** |  | System updates Member’s membership status. |
| **7** |  | System generates invoice/receipt. |
| **8** |  | System sends notification to Member: *“Your offline payment has been confirmed.”* |
| **9** | Manager returns to dashboard. | End of UC. |
| **Alternative Flow** | **Alternative Flow 1: Manager Rejects Offline Payment** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager clicks Reject Payment. | System asks for rejection reason. |
| **2** | Manager enters reason | System marks payment as Rejected. |
| **3** | Member receives notification: *“Your payment could not be verified.”* |  |
| **Alternative Flow 2: Payment Proof Requires Clarification** | | |
| **Step** | **Actor Acion** | **System Respond** |
| **1** | Manager selects Request More Info. | System sends message to Member. |
| **2** | Member uploads additional proof. | System updates payment entry. |
| **3** | Manager continues from Main Flow Step 3. |  |
| **Alternative Flow 3: Manager Creates Manual Offline Payment Entry** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager clicks “Create New Offline Payment.” | System opens a blank form. |
| **2** | Manager fills payment details manually. | System validates and saves. |
| **3** |  | System proceeds to Main Flow Step 5 onwards. |
| **Exception Flow** | **Exception Flow 1: Amount Mismatch** | | |
| **Step** | **Actor Action** | **System Response** |
| **1** | Manager tries to confirm payment. | System detects mismatched amount. |
| **2** |  | System warns: “Payment amount does not match package price.” |
| **Exception Flow 2: Missing Payment Proof** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager selects payment. | No proof provided. |
| **2** |  | System prompts: *“Payment evidence required before confirmation.”* |
| **Exception Flow 3: Database Error Saving Transaction** | | |
| **Step** | **Actor Action** | **System Respond** |
| **1** | Manager confirms payment. | Database error occurs. |
| **2** |  | System shows error and logs failure. |
| **Priority** | High | | |
| **Business Rule** | * Only Managers may confirm offline payments. * Offline payments must include valid evidence before confirmation. * Confirmed payments must immediately activate the membership. * All offline transactions must be logged for auditing. * Rejected payments must include a reason. | | |

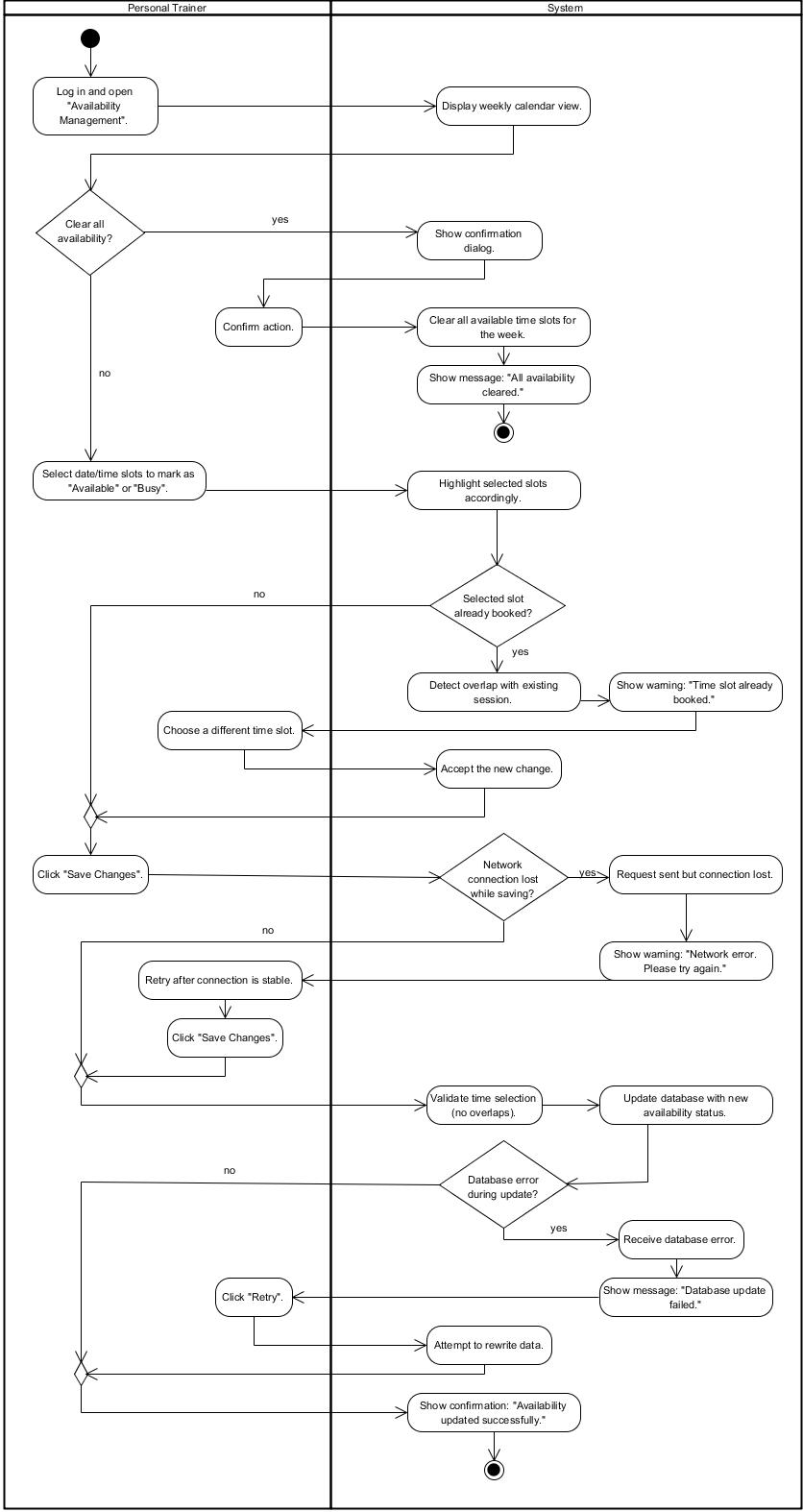
## 5.3 Activity Diagram

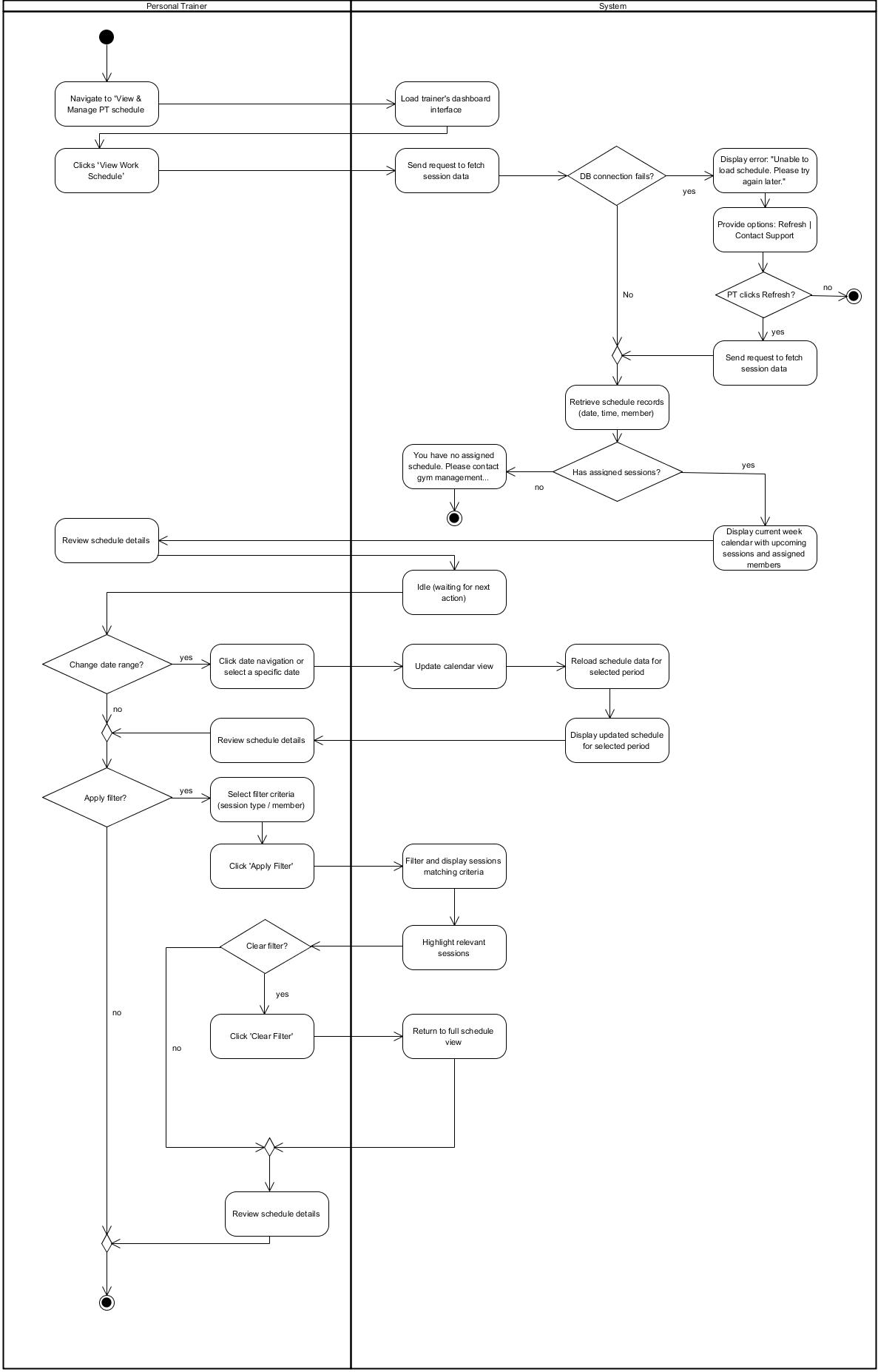
Receive System Notificaitions 

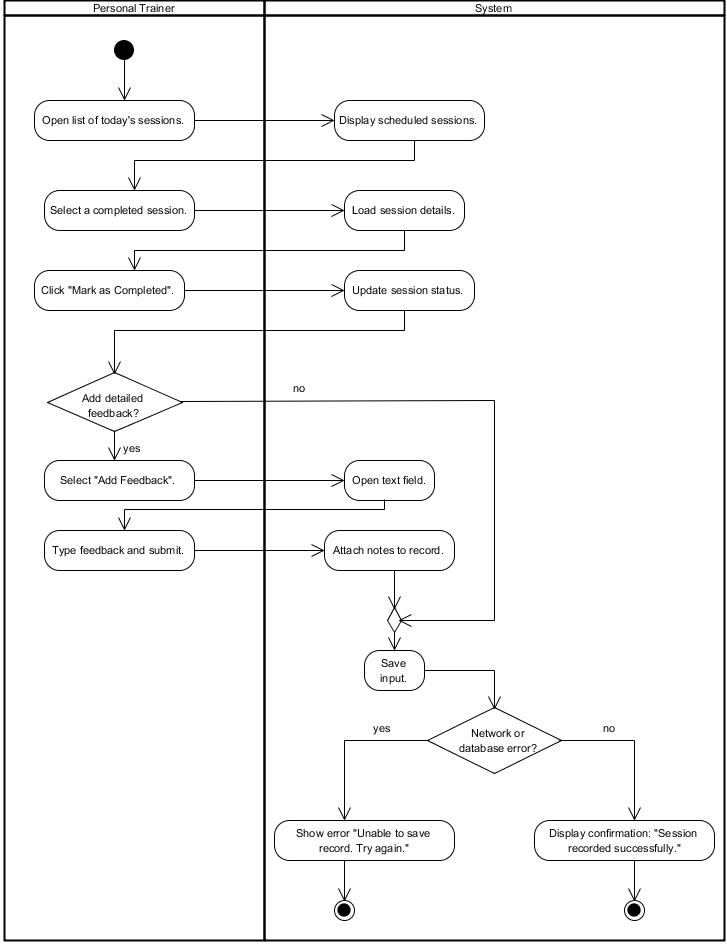
Purchase or Renew Membership 

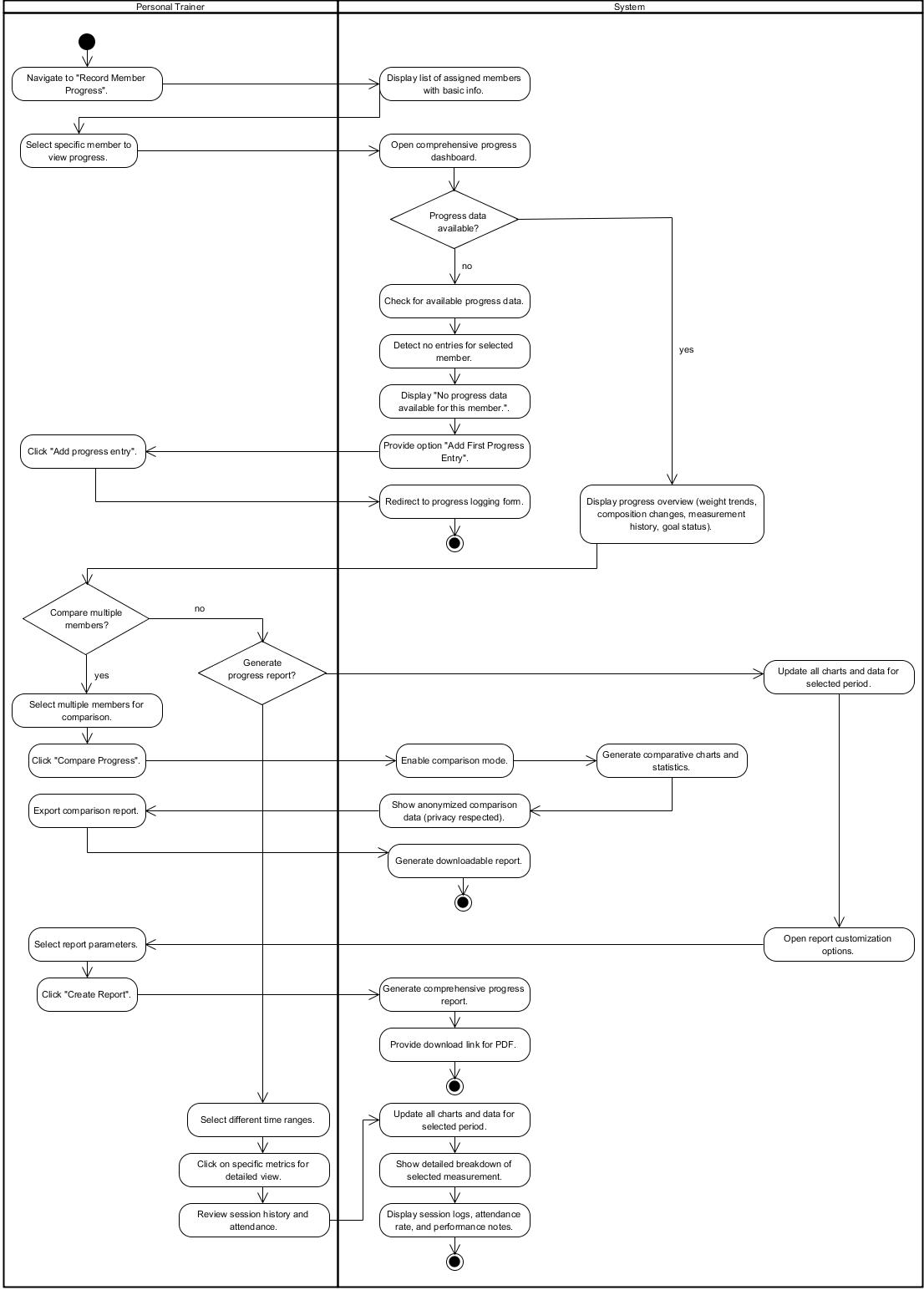
Book Training Sesion 

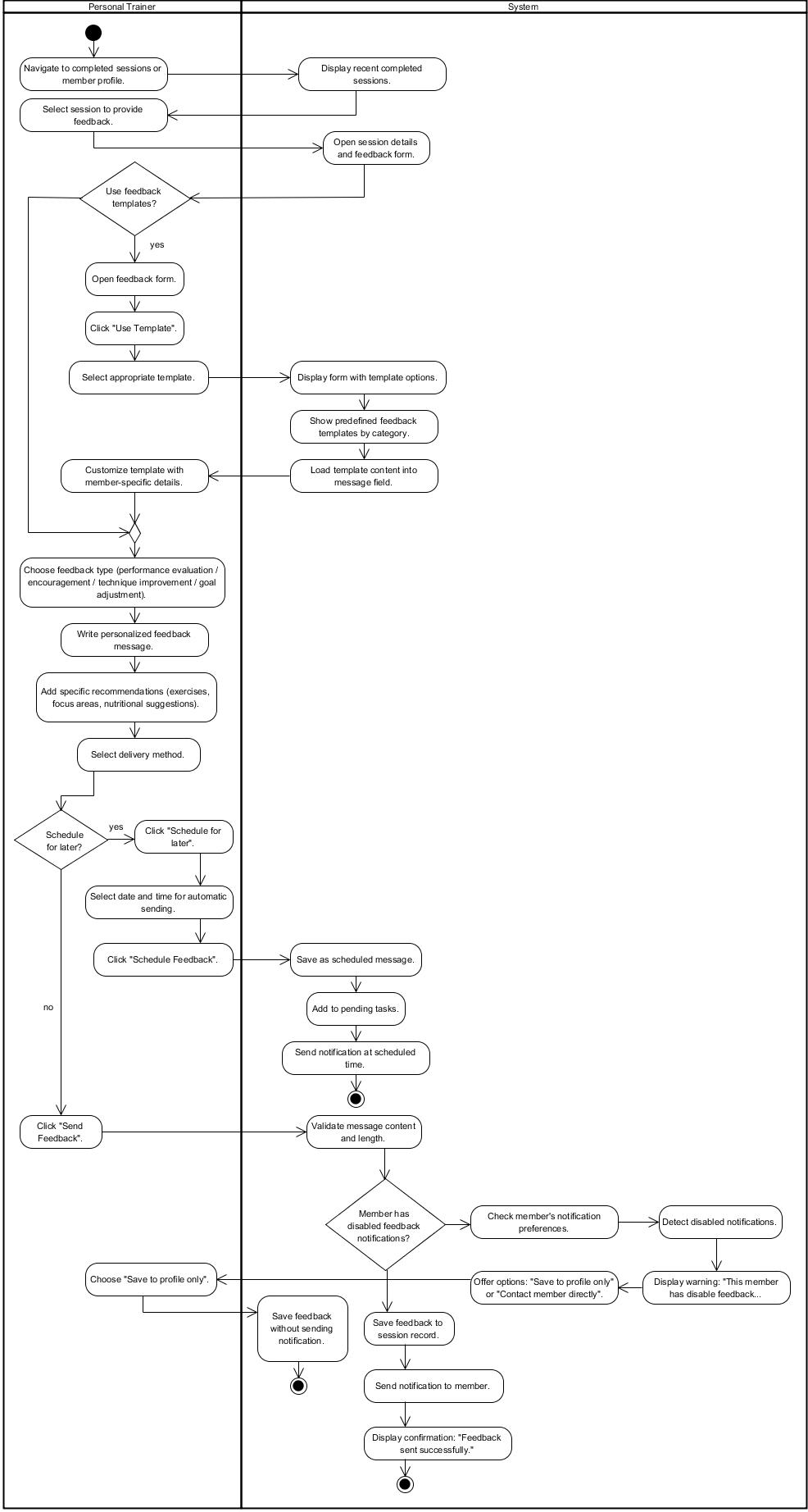
Submit Support Request

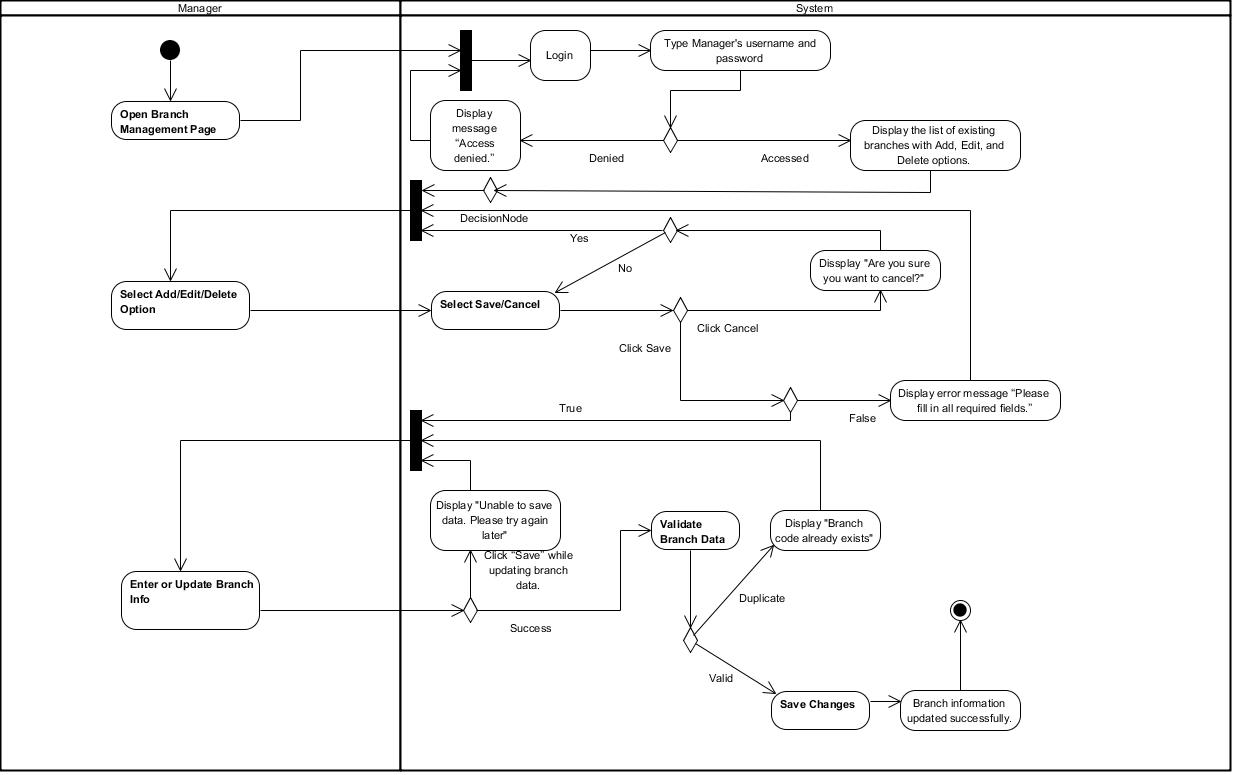
Update available

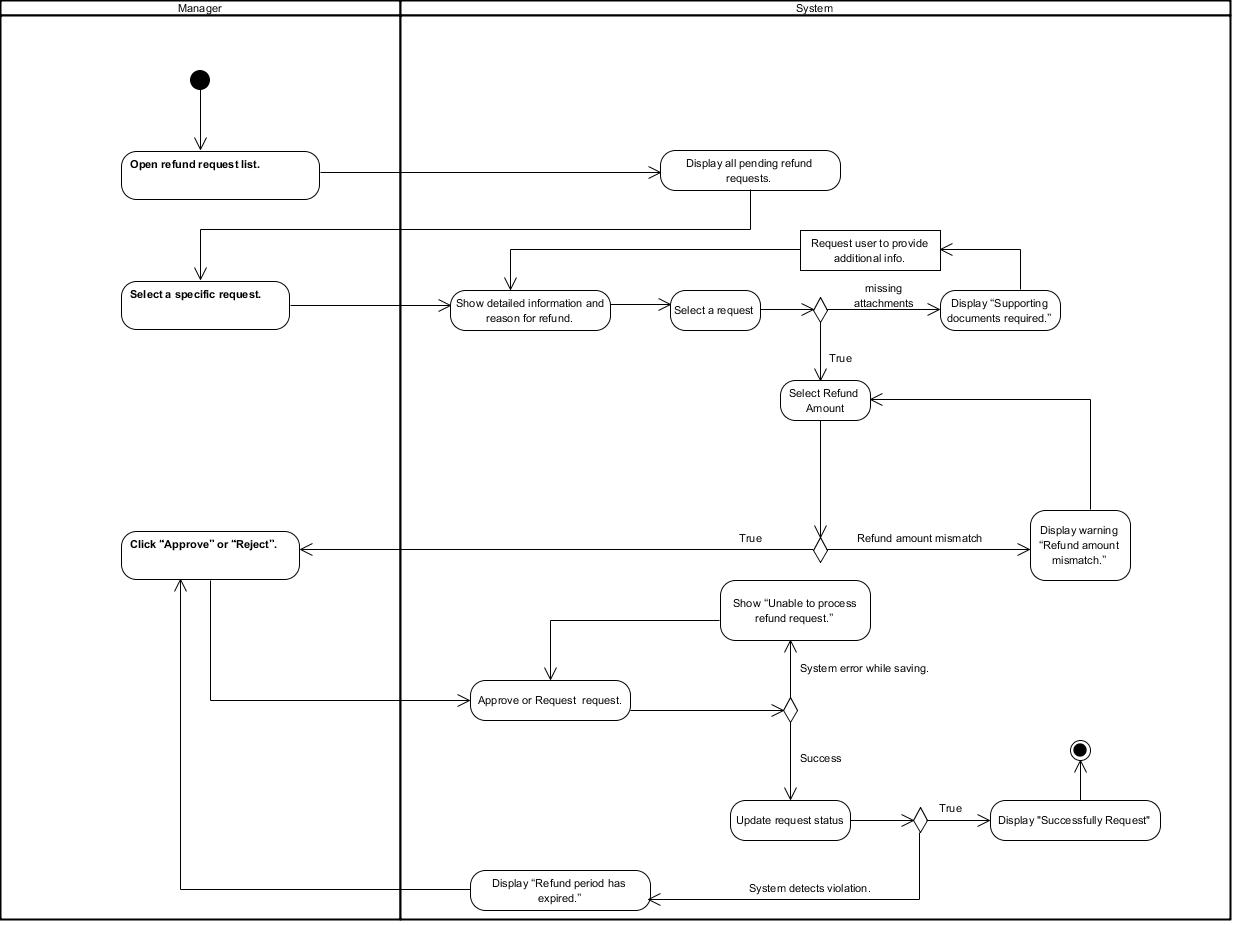
View Work Schedule

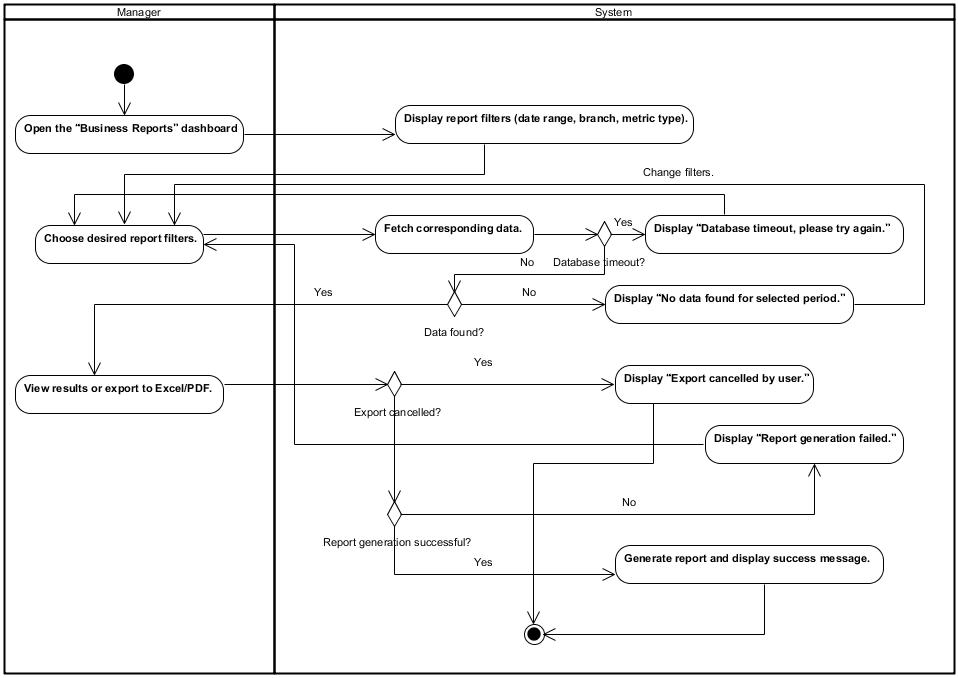
Record Training Notes

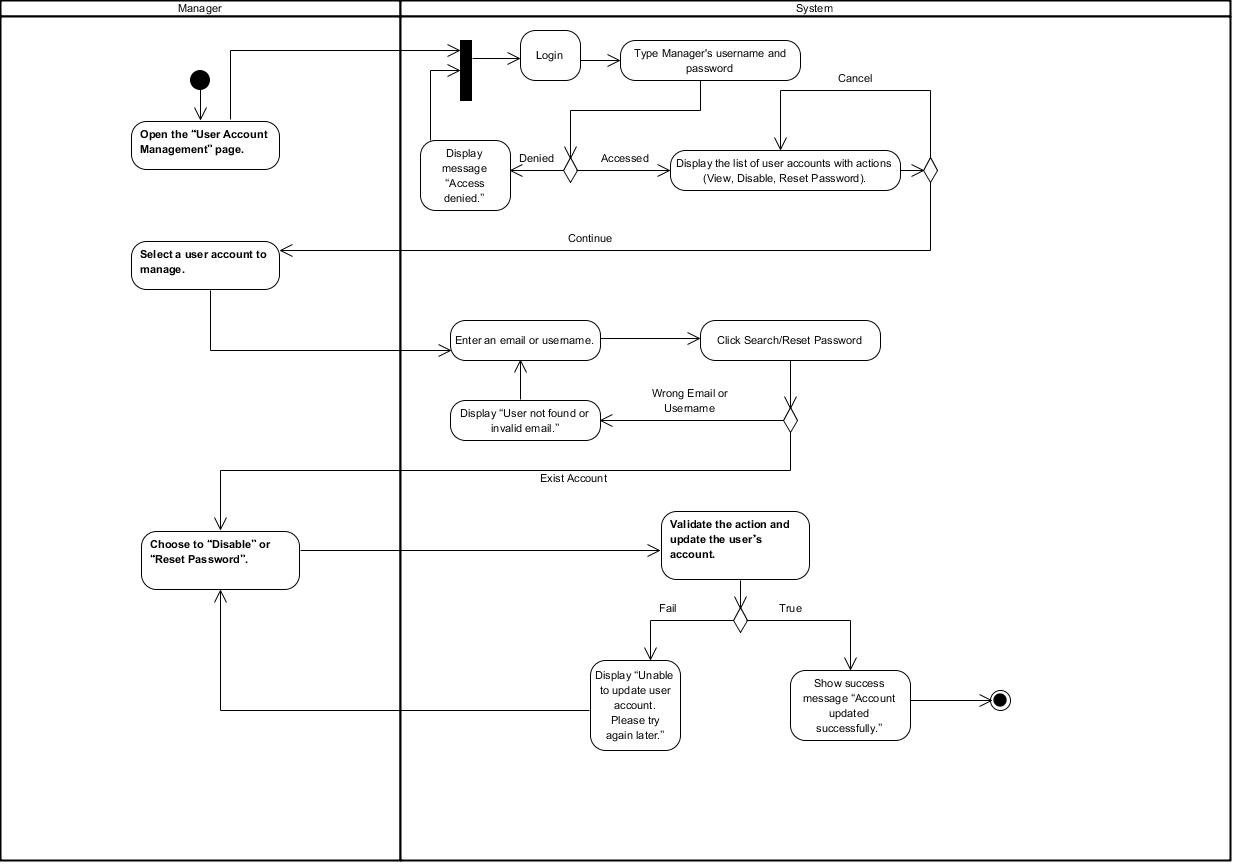
View Assigned Member Progress

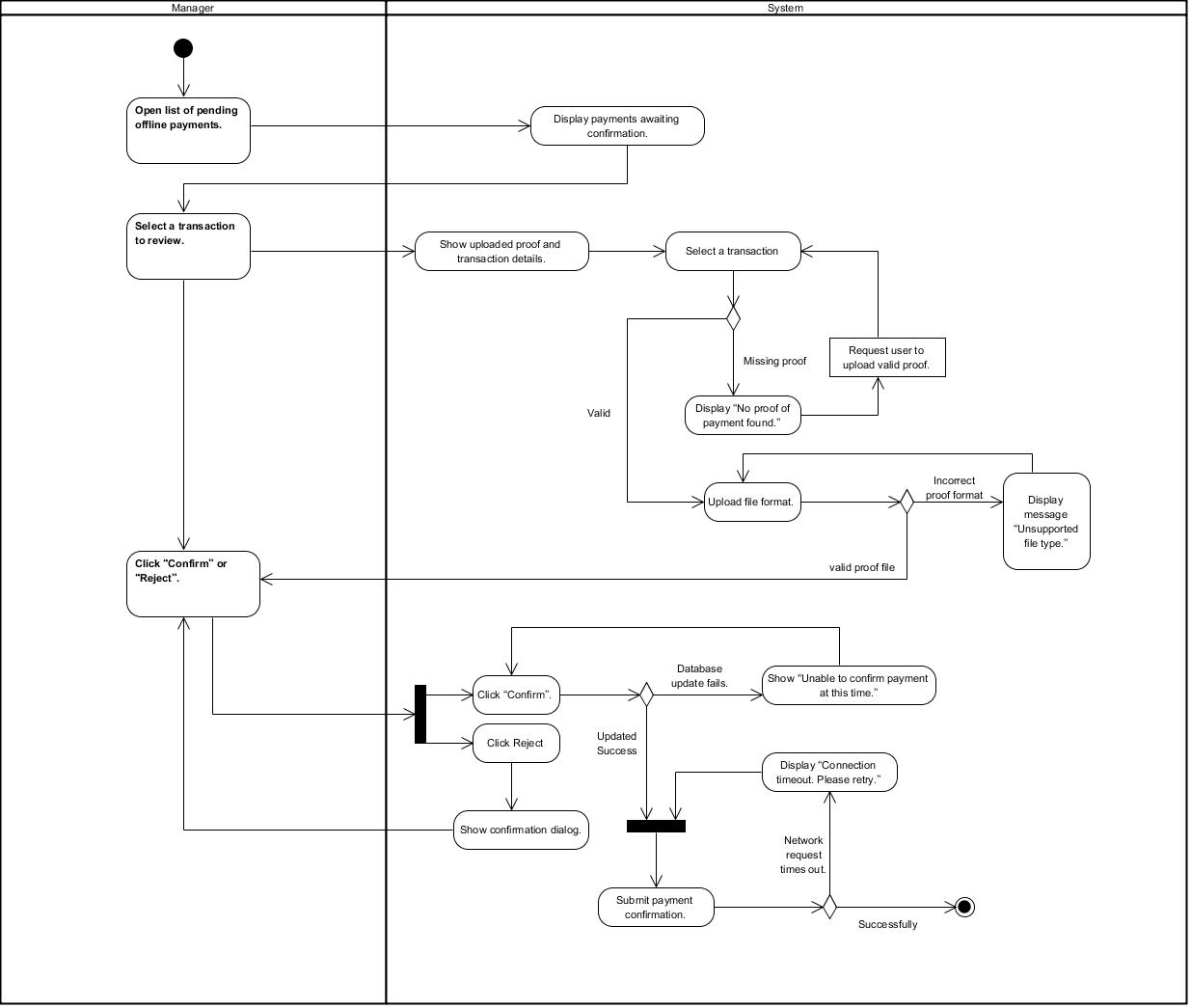
Send Session Feedback

Manage Gym Branches 

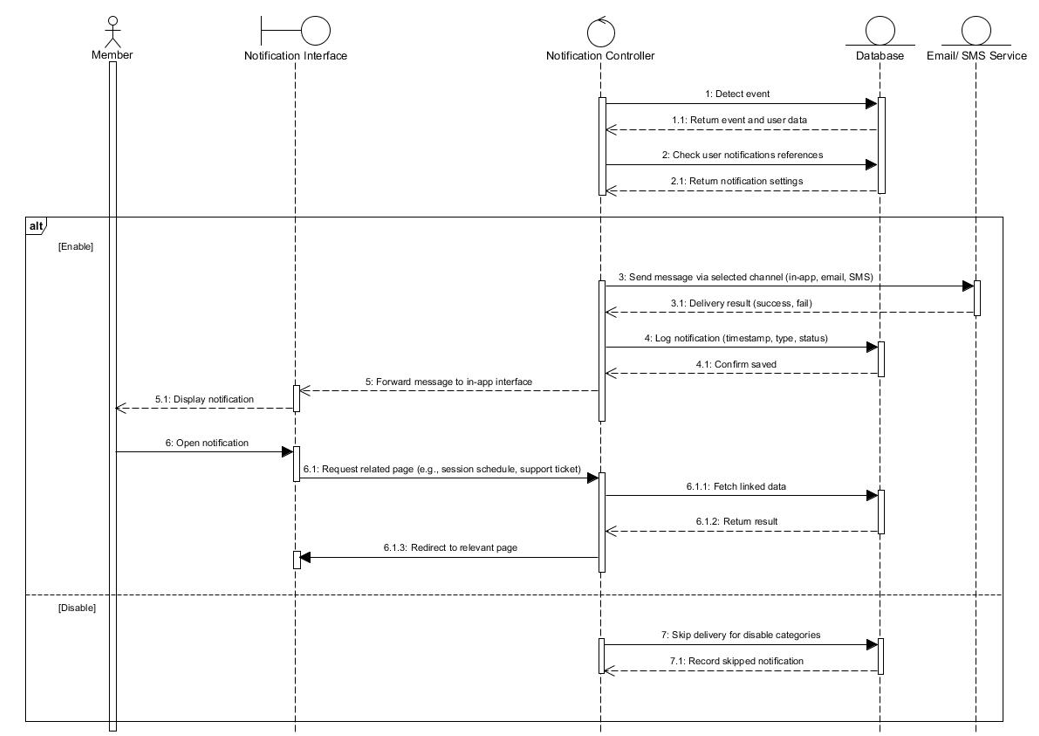
Process Refund Request

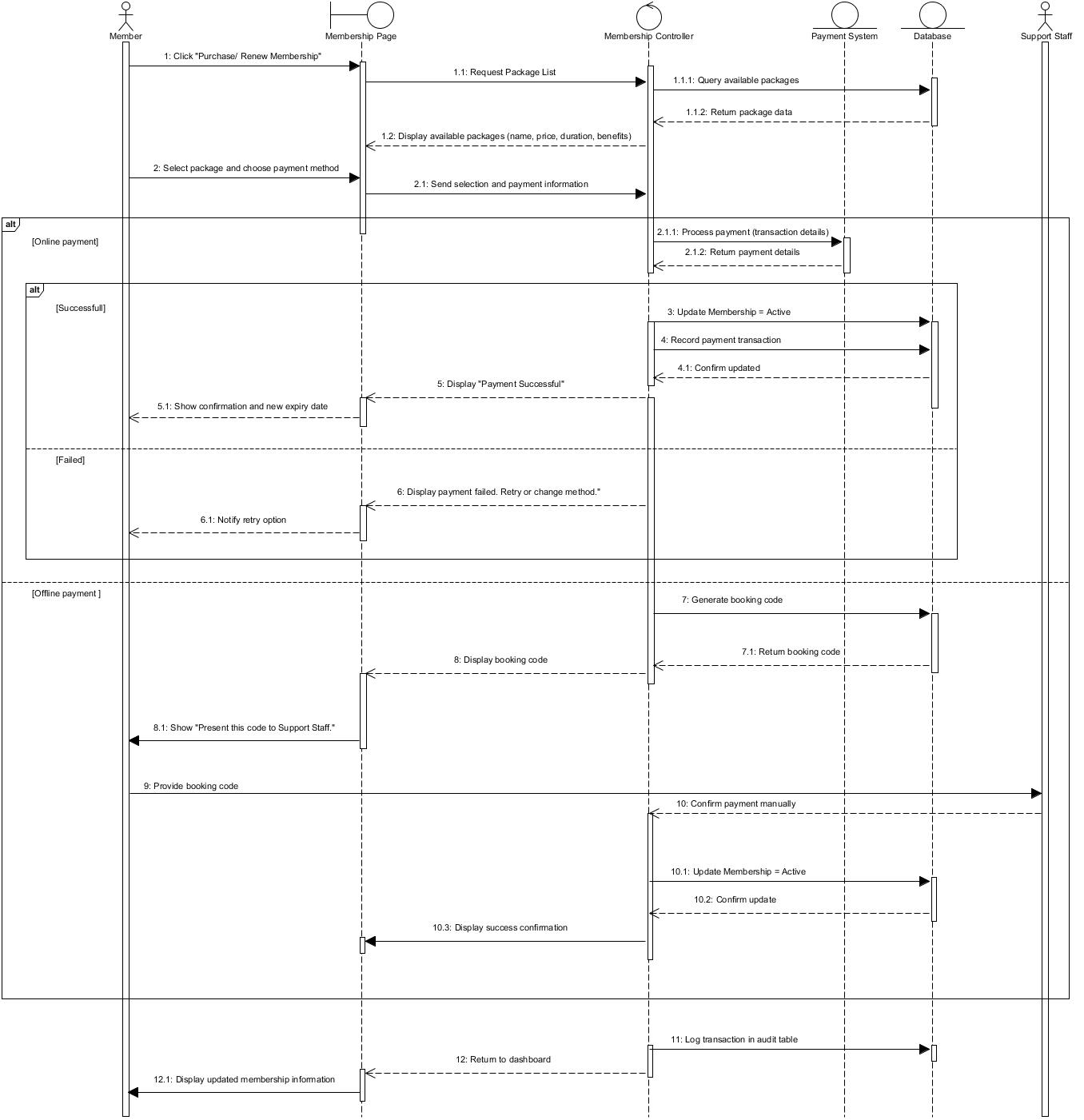
View Financial Reports

Manage Staff Accounts

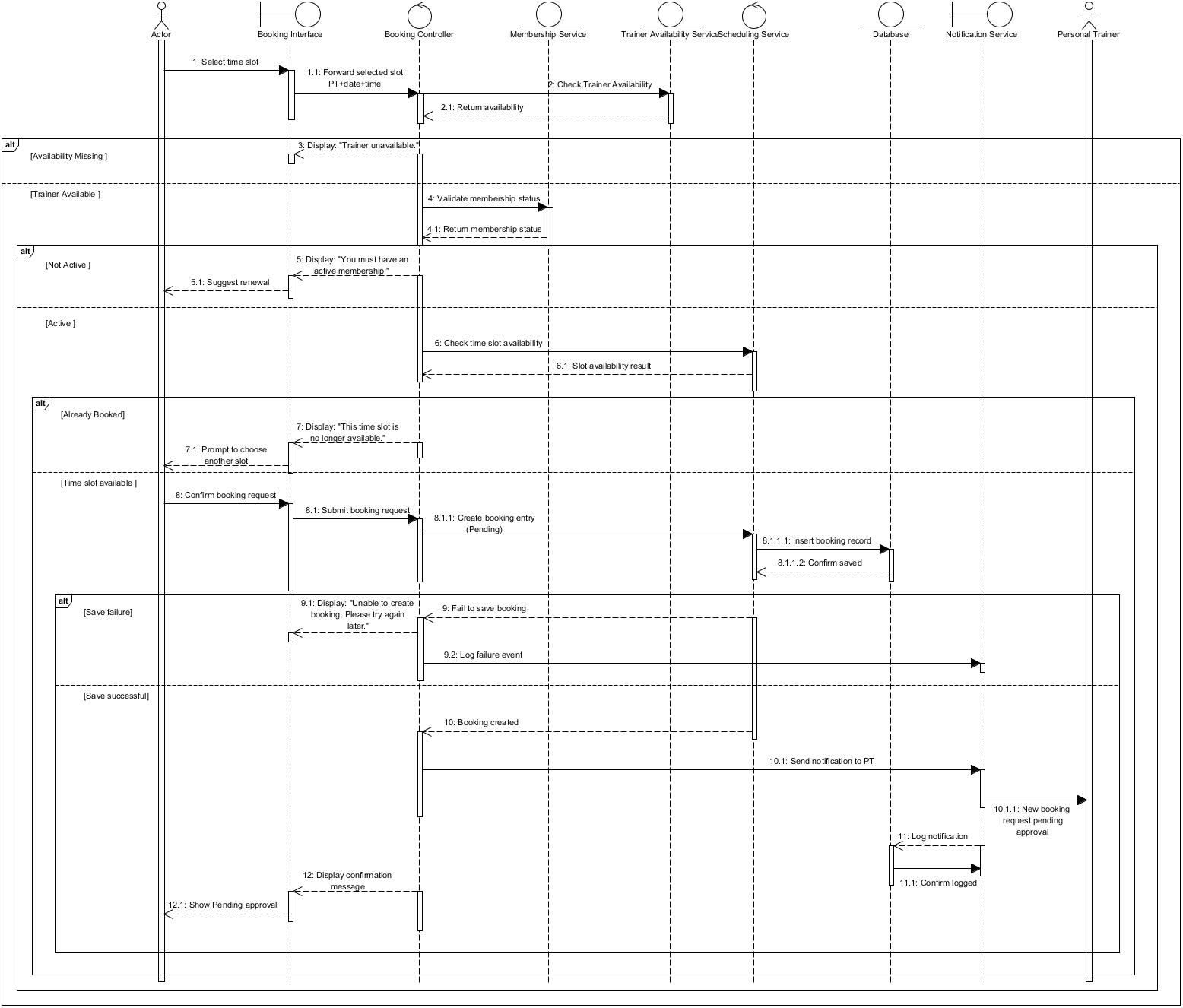
Confirm Offline Payment 

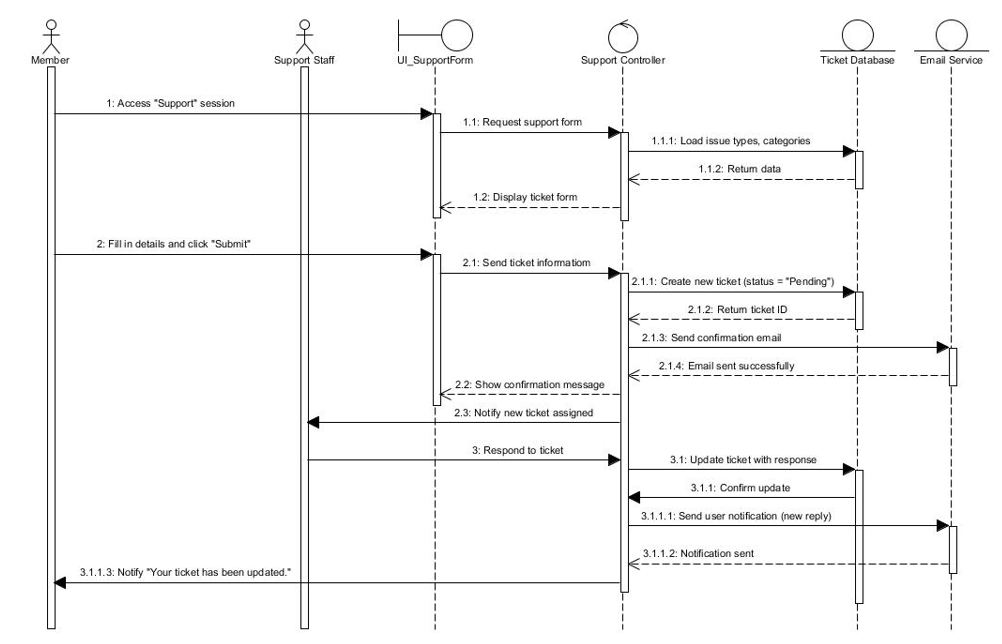
## 5.4 Sequence Diagram

 View Notifications

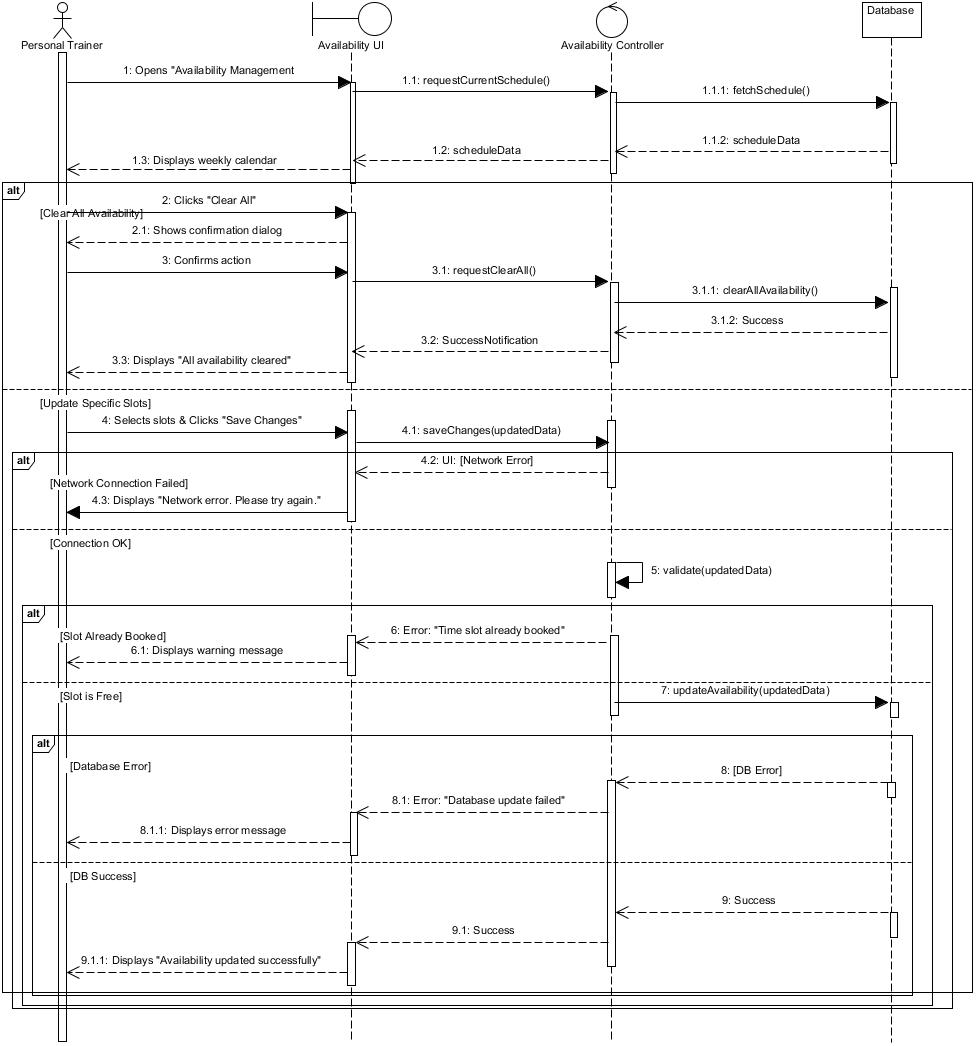


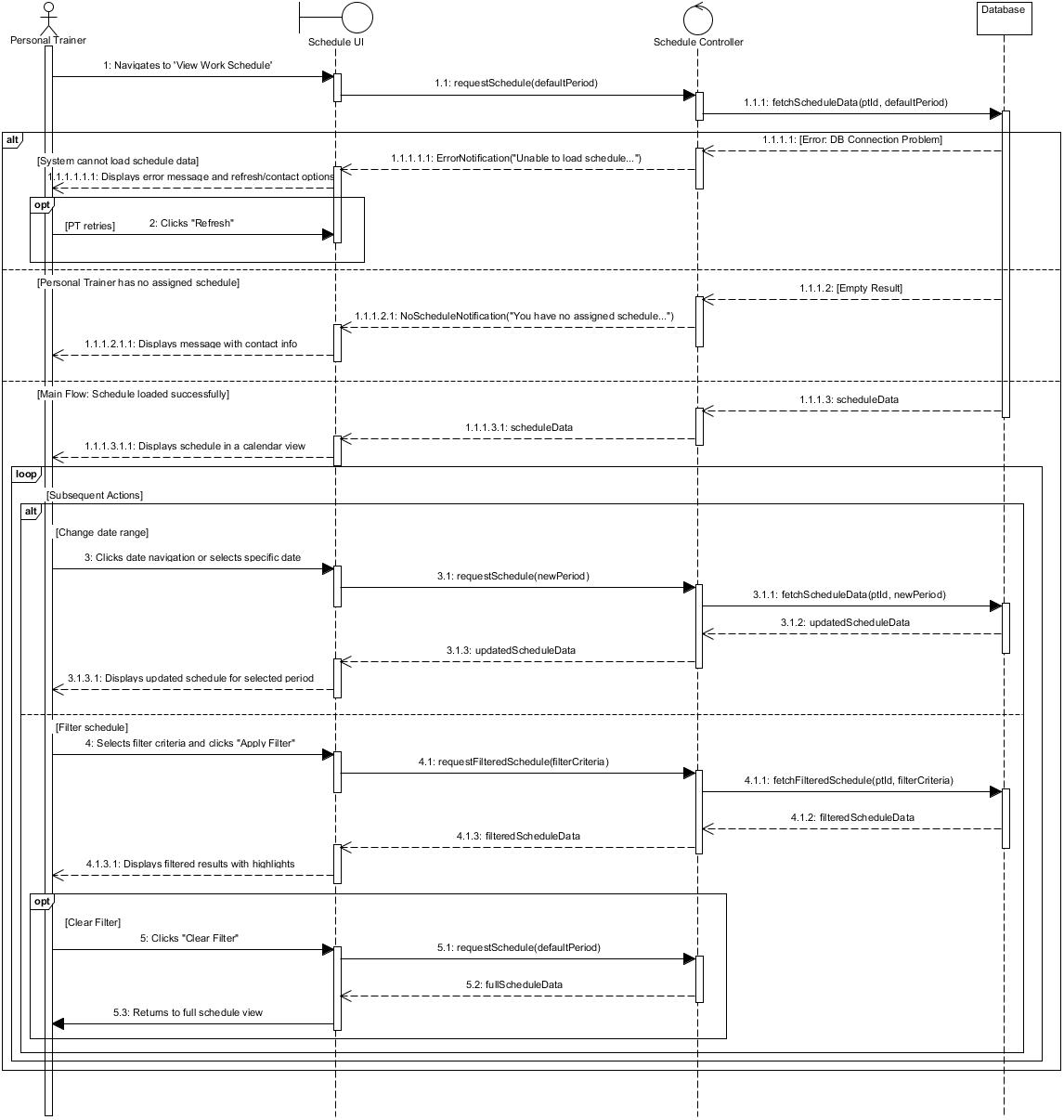
Purchase or Renew Membership

 Book Training Session

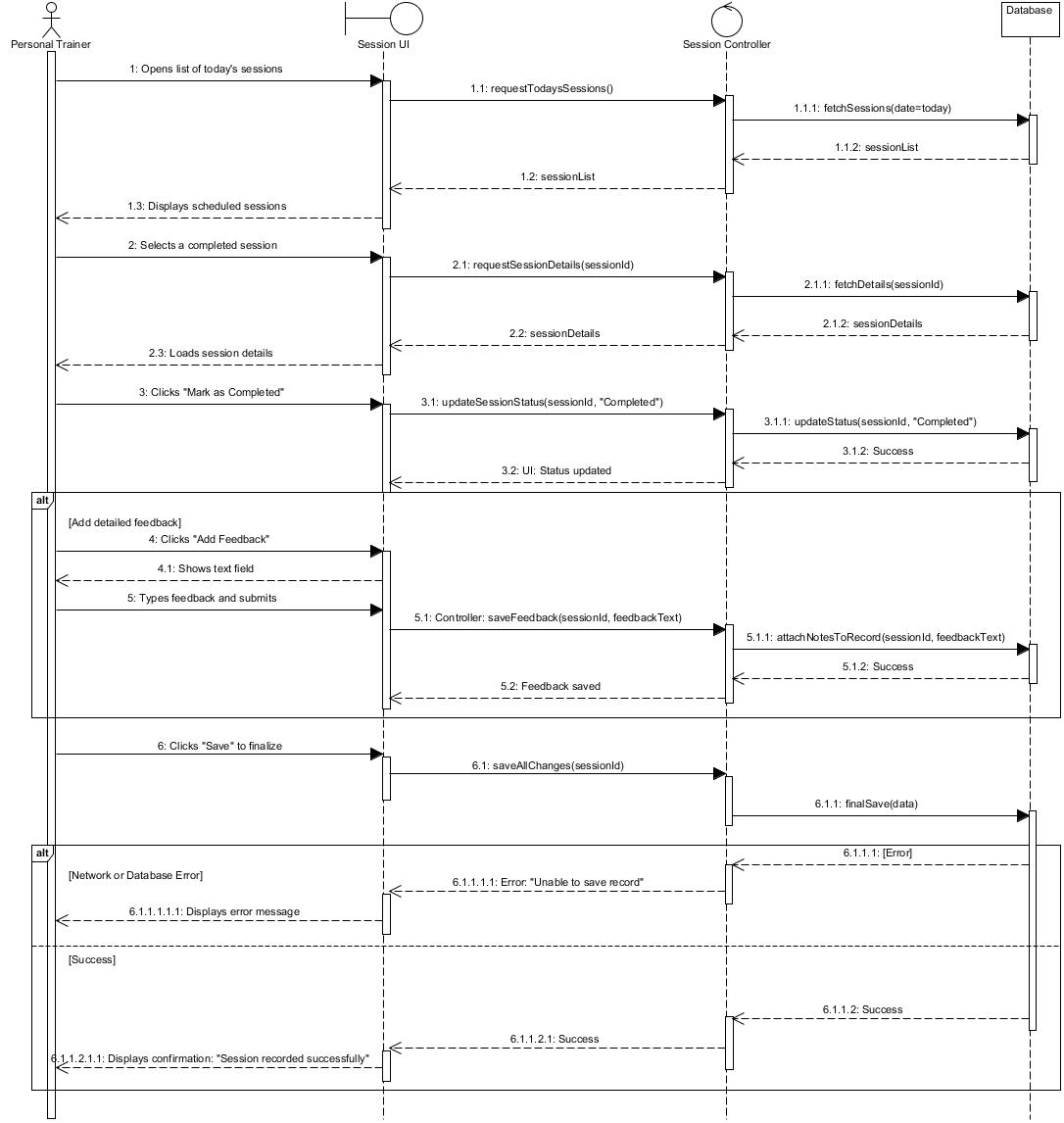


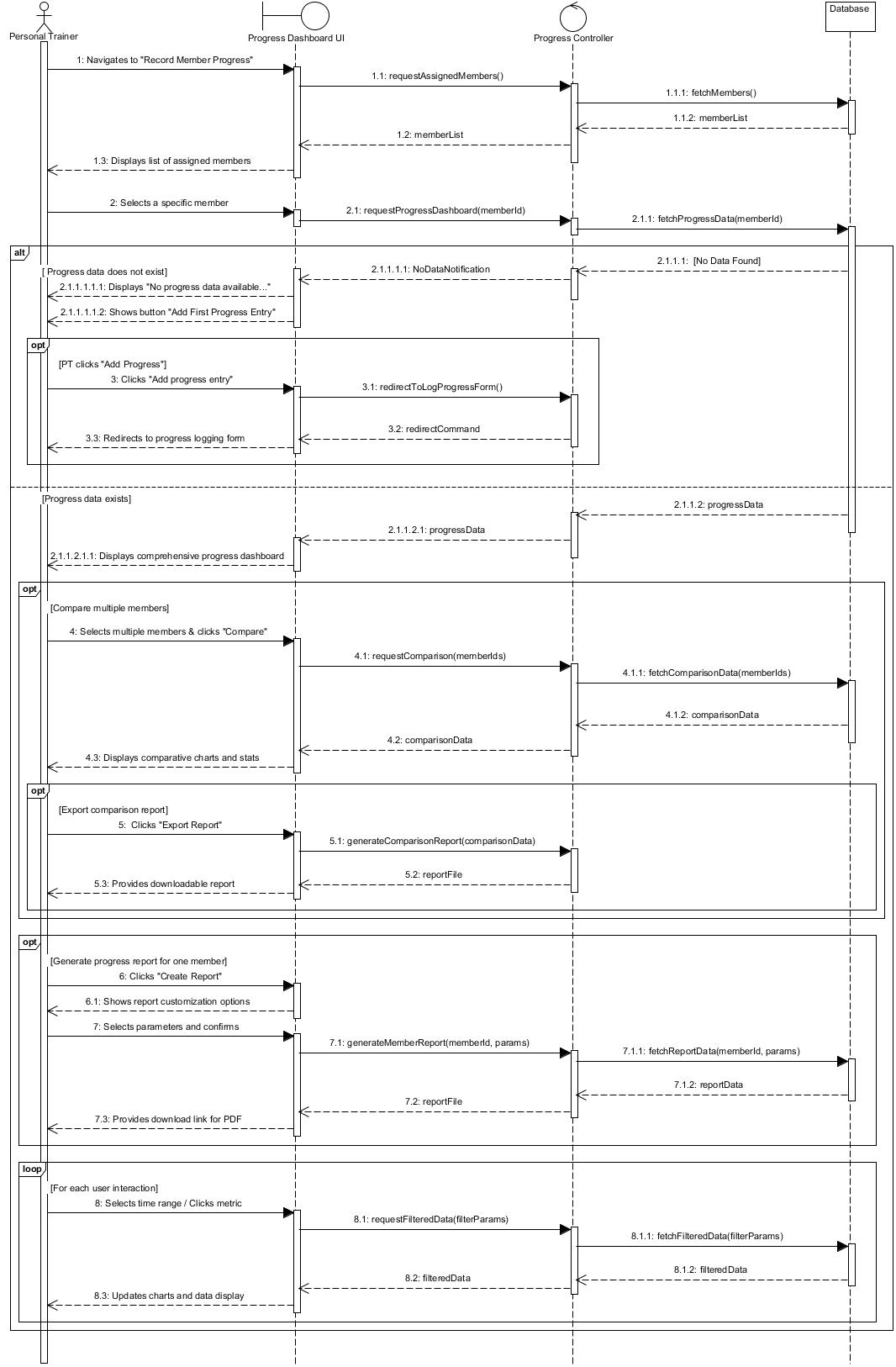
Submit Support Ticket

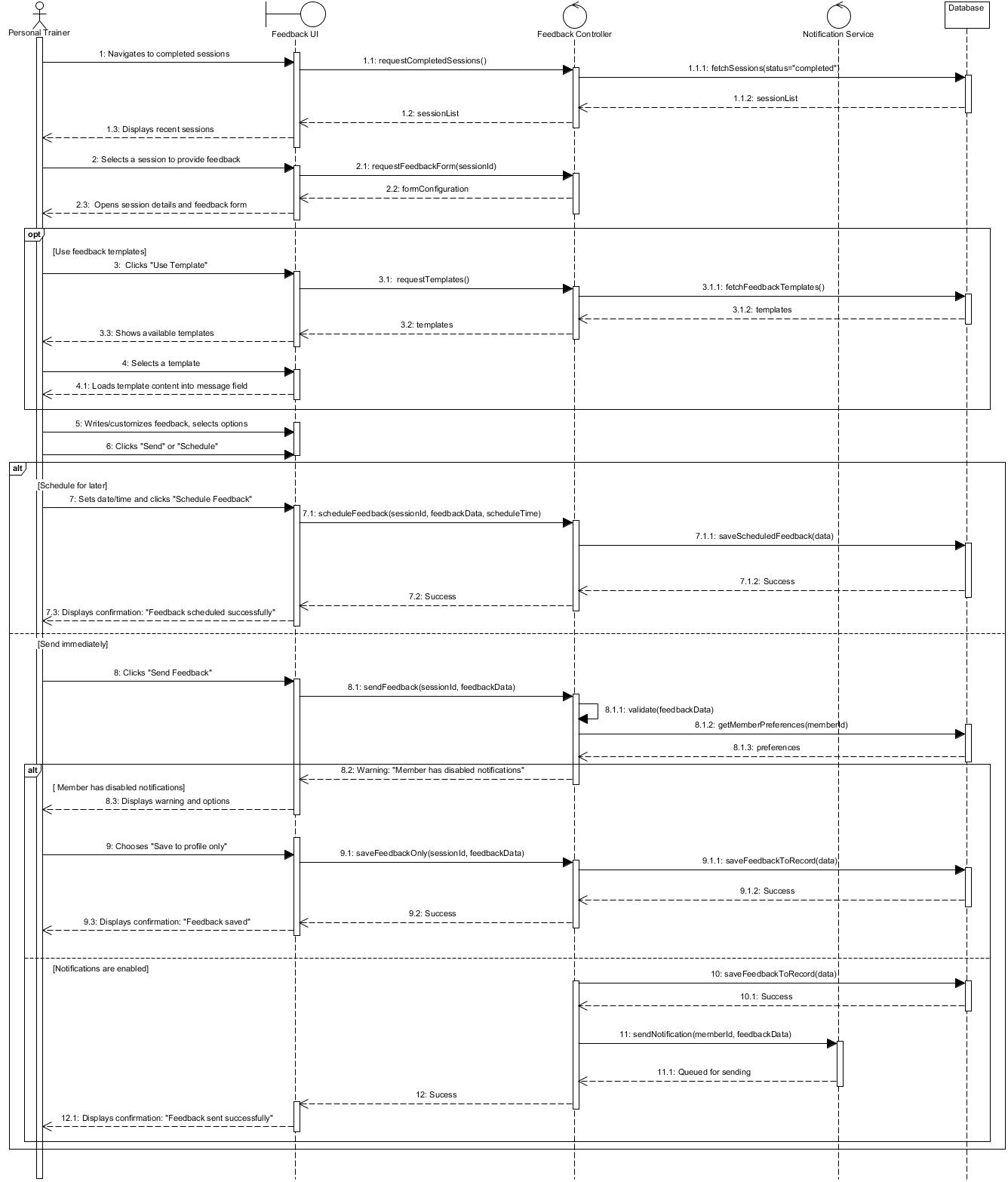
 Update available



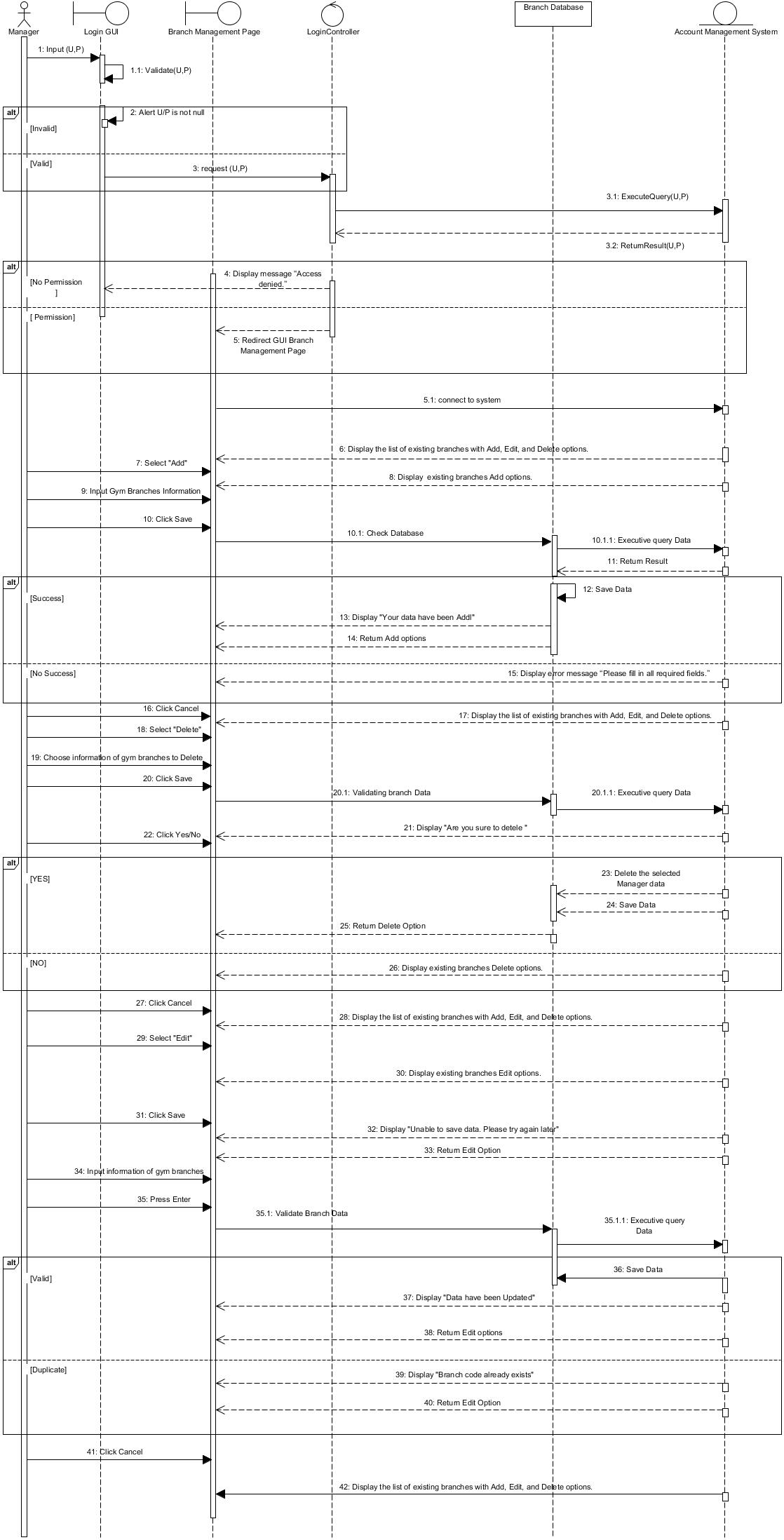
View Work Schedule

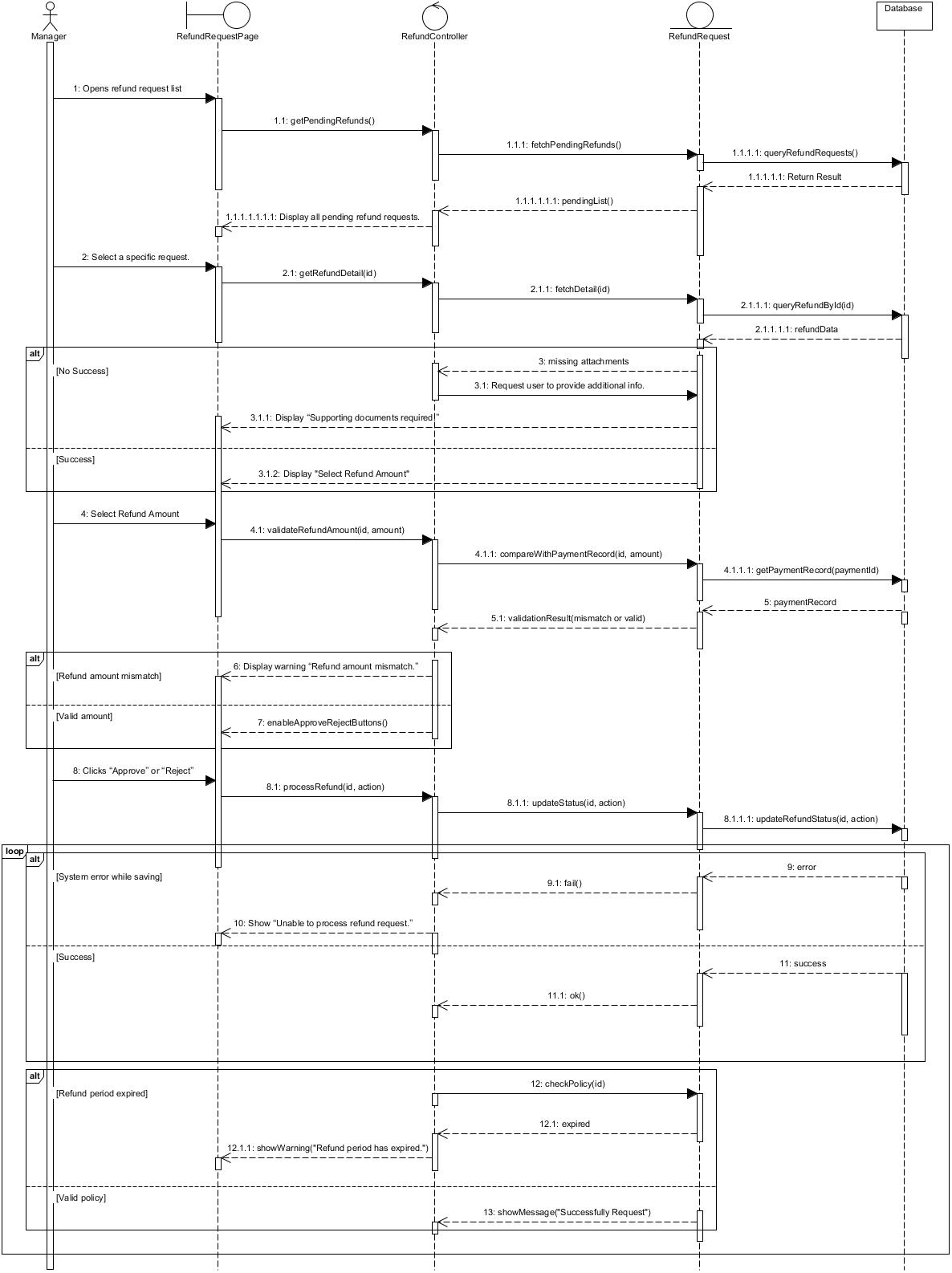
 Record Training Notes

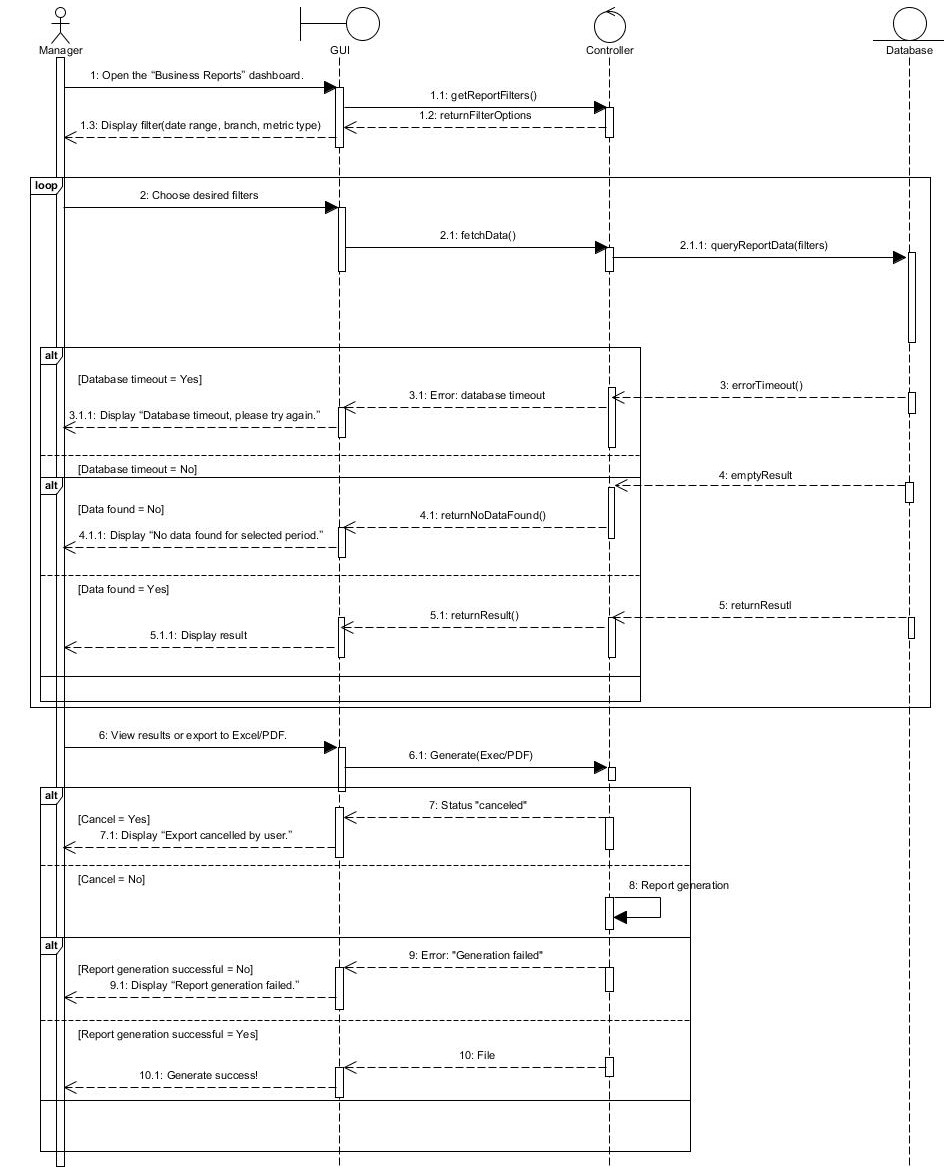
View Assigned Member Progress

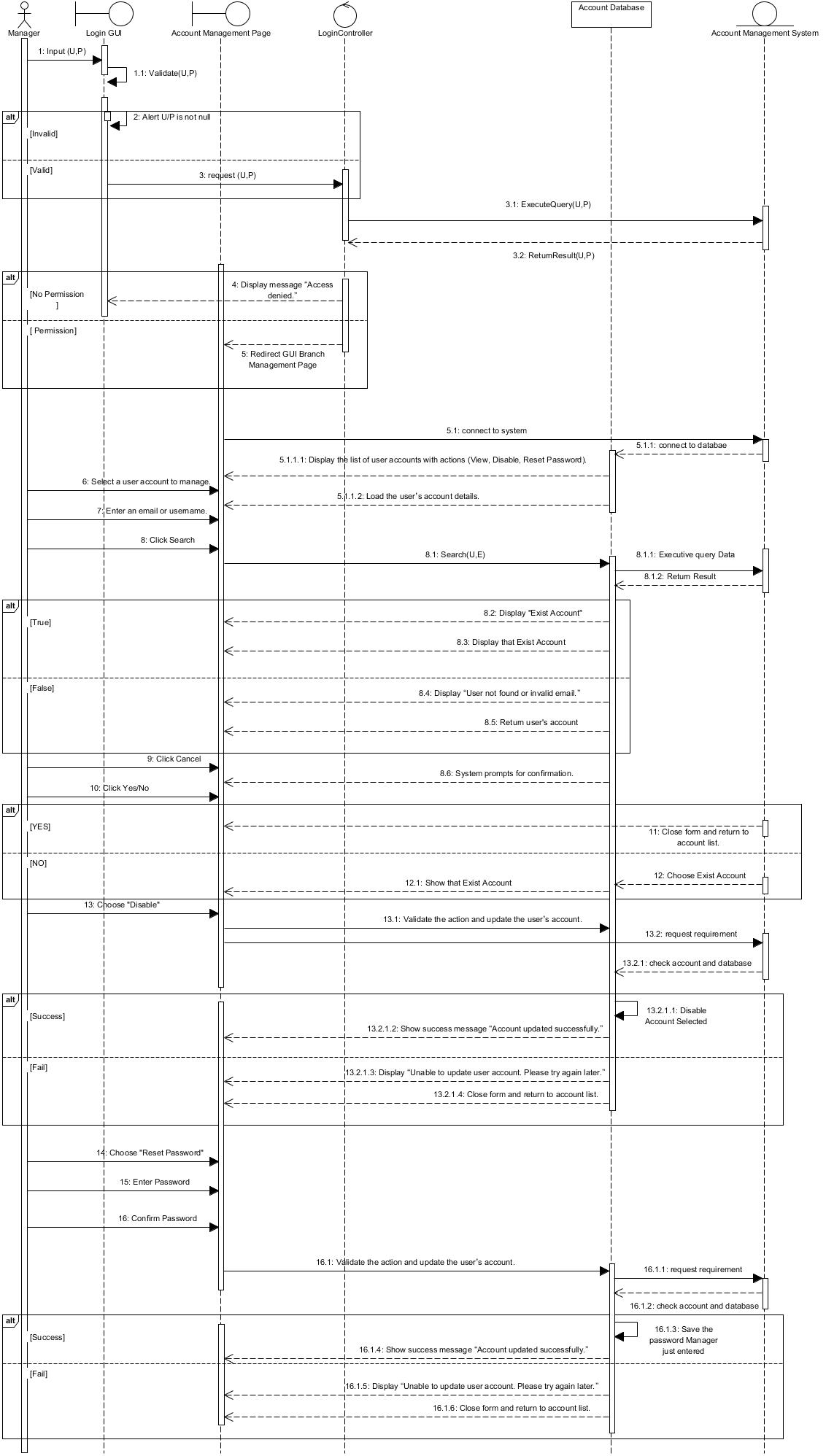


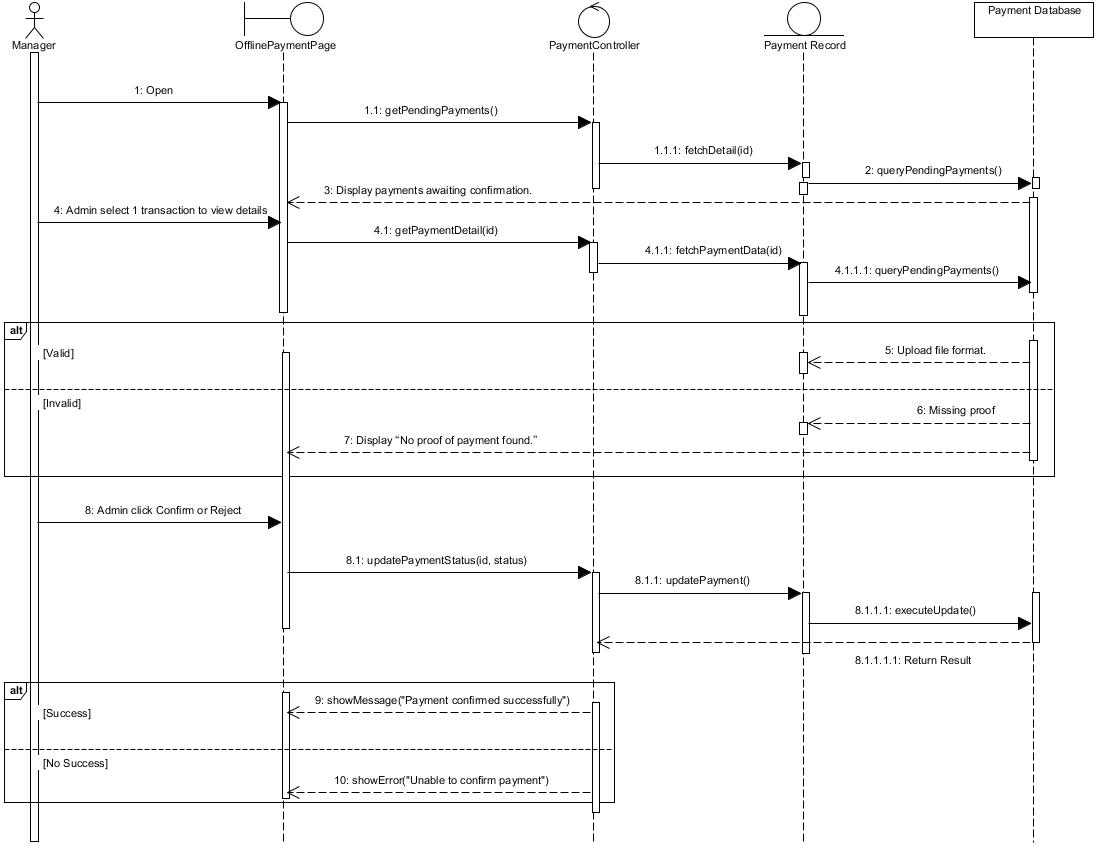
Send Session Feedback

 Manage Gym Branch

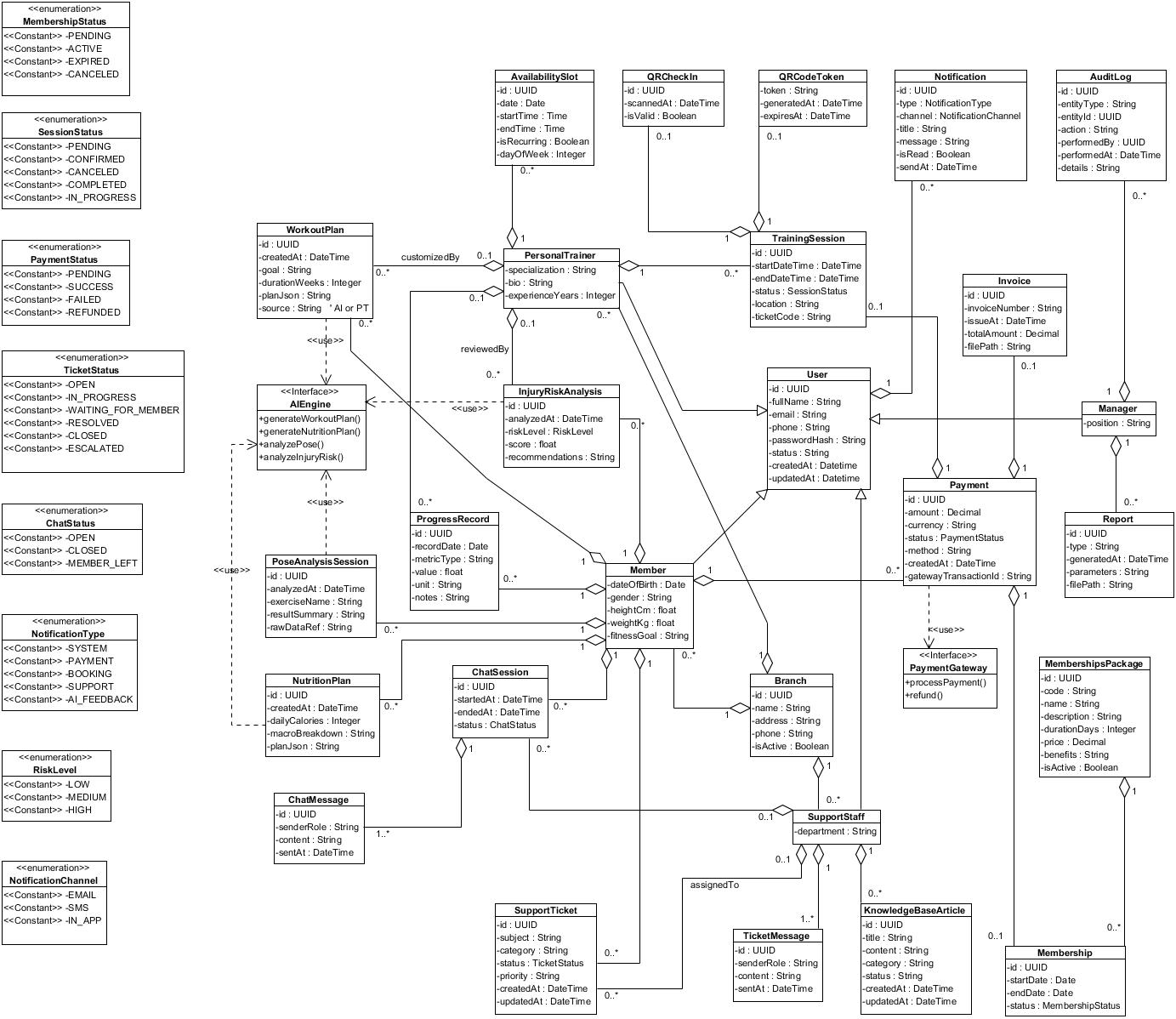
Process Refund Requests

View Financial Reports

Manage Staff Accounts

Confirm Offline Payment 

## 5.5 Class Diagram



# Other Requirements

## 6.1 Business Rule

|  |  |
| --- | --- |
| **Business Rule ID** | **Business Rule Name** |
| BR.01 | A member must have at least one active membership to book a training session. |
| BR.02 | A member can only book time slots that are marked as available by the personal trainer. |
| BR.03 | Booking requests must be approved by the personal trainer unless auto-approval is enabled. |
| BR.04 | Members cannot cancel a training session within 30 minutes before the session start time. |
| BR.05 | Rescheduling is only allowed for sessions with status Pending or Confirmed. |
| BR.06 | QR check-in is valid only within the allowed time window and for one-time use. |
| BR.07 | Membership becomes ACTIVE only after a successful online payment transaction. |
| BR.08 | Invoices can only be generated for successful payments and cannot be edited afterward. |
| BR.09 | AI workout and nutrition plans require complete personal information from the member. |
| BR.10 | Pose recognition stores only analysis results unless the member allows video recording. |
| BR.11 | Support tickets must start in status *OPEN* and follow the defined resolution workflow. |
| BR.12 | A member may have only one active live chat session at a time. |
| BR.13 | Only managers can modify membership packages, branch information, or system settings. |
| BR.14 | All critical actions (booking, payment, membership updates) must be logged for auditing. |

## 6.2 System Constraints

* The system is web-based and must run on a modern web server environment.
* Must support deployment on both Windows and Linux servers.
* All client–server communication must use HTTPS (TLS 1.2 or higher).
* The system must integrate with approved third-party services such as PayOS (payment) and Google Gemini API (AI features).
* The database must be a relational SQL system (MySQL or SQL Server).
* The frontend must run on modern web browsers (Chrome, Firefox, Edge, Safari).
* Mobile responsiveness is required for all user-facing modules.
* The system must support multi-branch scalability for future expansion.

# Prototype Design

## 7.1 Login and Authentication Interfaces

Purpose: Allow users (Member, PT, Manager, Support Staff) to authenticate into the system.

Main UI components:

* Email / Phone Number input field
* Password input field
* “Forgot Password?” link
* “Login” button
* “Register” link (visible only for Members)

Key behaviors:

* Invalid login attempts show an error message.
* After 3 failed login attempts, the system may temporarily lock the account (based on Business Rule).
* Role-based redirection after successful login:
  + Member → Member Dashboard
  + PT → Trainer Dashboard
  + Manager → Admin Dashboard
  + Support Staff → Support Console

## 7.2 Member-Side Prototypes

### 7.2.1 Member Dashboard

Purpose: Provide an overview of membership status, upcoming sessions, AI recommendations, and shortcuts to main functionalities.

UI Elements:

* Membership Status Card (Active, Expired, Pending)
* Quick Action Buttons:
  + View Packages
  + Book Training Session
  + AI Workout Plan
  + AI Nutrition Plan
  + Pose Recognition
* Upcoming Training Sessions (list or calendar preview)
* Progress Overview (small charts for weight, strength, or attendance)
* Notification Panel showing unread system alerts

### 7.2.2 Membership Package Browser

Purpose: Allow members to browse, compare, and purchase membership plans.

UI Elements:

* Package List (Monthly, Quarterly, Yearly)
* Price, duration, and benefit description
* “View Details” modal for each package
* “Purchase / Renew” button
* Payment summary popup after selection

Behaviors:

* If a member already has an active membership, the UI highlights the renewal option.
* After purchase → redirect to invoice page.

### 7.2.3 Payment and Invoice Screen

Purpose: Handle payment confirmation and invoice viewing.

UI Elements:

* Payment summary (package name, duration, total cost)
* Payment Method selector
* “Confirm Payment” button
* Success page showing:
  + Invoice number
  + Download/ Print button

### 7.2.4 Trainer Selection and Availability Screen

Purpose: Display PT profiles and available time slots.

UI Elements:

* Trainer list with photo, specialization, rating
* Filters: specialization, gender, availability
* Weekly calendar layout
* Available slot (green), booked slot (gray), unavailable (red)

Booking Workflow:

* User selects a PT
* Calendar updates to show available slots
* User clicks a slot → “Confirm Booking” modal
* System checks membership status before confirming

### 7.2.5 Member Session Management UI

Purpose: Manage bookings (reschedule, cancel, view details)

UI Elements:

* List of upcoming sessions
* Status badges: Pending, Confirmed, Completed, Canceled
* Buttons:
  + “Reschedule”
  + “Cancel Session”
  + “View Detail”

Rules: Cancellation/rescheduling disabled when approaching session start time (Business Rule).

### 7.2.6 Progress Tracking Dashboard

Purpose: Display workout progress and body metrics.

UI Elements:

* Line chart: Weight change over time
* Bar chart: Workout frequency per week
* Strength progression (1RM charts)
* Session history table with PT notes
* Filter by: 7 days / 30 days / Custom range

### 7.2.7 AI Workout Plan Interface

Purpose: Provide personalized training plans generated by the AI engine.

UI Elements:

* Input form: fitness goal, experience level, available equipment
* “Generate Plan” button
* Weekly plan layout (Day 1–Day 7)
* Exercise cards: name, sets/reps, instructions
* “Regenerate Plan” or “Download Plan” options

### 7.2.8 AI Nutrition Plan Interface

Purpose: Display daily nutritional recommendations.

UI Elements:

* Daily calorie target
* Macro pie chart (carbs/protein/fats)
* Suggested meals for breakfast, lunch, dinner
* Option to modify dietary preferences

### 7.2.9 Pose Recognition Interface

Purpose: Provide real-time posture analysis via camera.

UI Components:

* Live camera preview
* Skeleton overlay for detected joints
* Real-time feedback panel (warnings, tips)
* “Start Session” and “End Session” controls
* Summary report after completion

### 7.2.10 Injury Risk Analysis Interface

UI Elements:

* Risk Score Gauge (Low / Medium / High)
* Contributing Factors List
* Recommended preventive adjustments
* Workout history correlation chart

## 7.3 Personal Trainer Interface

### 7.3.1 Trainer Dashboard

Purpose: Centralized view of PT schedules and assigned members.

UI Elements:

* Today's Sessions
* Weekly Calendar
* Member Progress Quick View
* Announcements / System Alerts

### 7.3.2 Trainer Availability Management

Purpose: Allow PT to set or update availability.

UI Elements:

* Weekly timetable editor
* Add/Edit time slot modal
* Toggle recurring availability
* “Block Time” function for personal events

### 7.3.3 Session Processing Screen

Purpose: PT views booked sessions and takes actions.

UI Elements:

* Session list with details
* Buttons:
  + Confirm Request
  + Decline Request
  + Mark as Completed
* Notes section to record session outcomes

### 7.3.4 Member Progress Management

Purpose: PT records progress after each training session.

UI Elements:

* Input fields for metrics (reps, sets, weights)
* Free-text notes
* File upload for optional media
* Save & update buttons

## 7.4 Manager Interface

### 7.4.1 Manager Dashboard

Purpose: Provide operational oversight and analytics.

Key Panels:

* Revenue Overview (Daily / Monthly / Yearly)
* Member Growth Chart
* PT Performance Summary
* System Usage Metrics
* Quick Actions:
  + Manage Packages
  + Manage Branches
  + Manage Trainers
  + View Reports

### 7.4.2 Package Management UI

Purpose: Allow managers to add/edit/remove membership packages.

UI Elements:

* Package list table
* Add/Edit Package modal
* Benefits editor
* Status toggle (active/inactive)

### 7.4.3 Branch Management UI

Elements:

* List of gym branches
* Add new branch button
* Edit branch details
* Status control (active/inactive)

### 7.4.4 Report Generation UI

Purpose: Generate system reports.

Report Types:

* Financial Report
* Trainer Performance Report
* Member Activity Report
* Support System Report

UI Features:

* Filter panel (date range, trainer, branch)
* “Generate” and “Export PDF” buttons

## 7.5 Support Staff Interface

### 7.5.1 Support Ticket Console

Purpose: Manage all tickets submitted by members.

UI Elements:

* Ticket list table
* Filters: priority, category, status
* Ticket detail pane
* Comment thread for communication
* Status actions: In Progress, Escalate, Resolve, Close

### 7.5.2 Live Chat Console

Purpose: Provide real-time support.

UI Elements:

* List of active chats
* Chat window
* Predefined response shortcuts
* Member profile preview
* Transfer chat to another staff option

### 7.5.3 Knowledge Base Editor

Purpose: Staff manage help articles.

UI Elements:

* Article list
* Create/Edit Article modal
* Category selector
* Publish / Archive buttons

## 7.6 Global UI Elements

### 7.6.1 Navigation Bar

* Home
* Dashboard
* Notifications
* Profile Settings
* Logout

### 7.6.2 Notification Center

* Dropdown list of unread notifications
* Click to open full notification page
* Mark all as read

### 7.6.3 User Profile Settings

* Personal info update
* Change password
* Notification Preferences
* Privacy & AI data consent options

## 7.7 User Experience (UX) Principles

* Consistency: Uniform color scheme, icon set, and layout.
* Responsiveness: All pages optimized for desktop, tablet, and mobile.
* Accessibility:
  + Clear font sizes
  + ARIA labels for screen readers
  + High contrast mode
* Error Prevention: Warnings for irreversible actions (canceling sessions, deleting records).
* Guided Flow: Tooltips, placeholders, and onboarding prompts for new members.

# Appendix A: Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Dashboard | The main overview screen displaying key system metrics depending on the user role (Member, Trainer, Manager, Support Staff). |
| Membership | A gym access package purchased by a Member, containing duration, pricing, and benefits. |
| Booking | The process of scheduling a training session between a Member and a Personal Trainer. |
| Training Session | A one-on-one appointment between a Member and a Personal Trainer. |
| PT (Personal Trainer) | A certified trainer responsible for conducting training sessions and tracking member progress. |
| Session Availability | Time slots defined by the Personal Trainer indicating when they are available for training. |
| QR Check-in | A QR-based mechanism allowing Members to confirm attendance for a scheduled session. |
| AI Workout Plan | An automatically generated exercise plan using AI models based on user goals and past training data. |
| AI Nutrition Plan | A personalized nutrition recommendation produced by the AI module. |
| Pose Recognition | An AI feature that analyzes user movements through camera input and provides corrective feedback. |
| Support Ticket | A request for help submitted by Members, handled by Support Staff. |
| Escalation | The process of forwarding a ticket to a Manager when the Support Staff cannot resolve it. |
| Knowledge Base (KB) | A collection of help articles maintained by Support Staff to assist Members. |
| Branch | A physical gym location managed by the system. |
| Invoice | A digital receipt generated after a successful membership purchase or renewal. |
| Refund | The process of returning payment to a Member for a failed or disputed transaction. |
| Audit Log | A system-generated log documenting sensitive or administrative actions for security and compliance. |
| Live Chat | A real-time communication channel between Members and Support Staff. |
| Support Analytics | Reports showing ticket load, response times, and support team performance. |
| Manager | A high-level system user responsible for branch operations, staff management, reporting, and escalated cases. |
| Support Staff | System users responsible for resolving support tickets, live chat, and managing the Knowledge Base. |