Deliverable 3

Team Name: Mint

Project Name: UJ Sport Athlete Management & Booking System

Team Number: 26

Mentor Name: Mr. Daniel Ogwak

Team Members

220046244 – Pashin Soma

221060349 – Tshepo Mohale

221000696 – Morena Ramaili

221105301 – Paidamoyo Mapfuwa

Contents

Updated Problem Statement, Proposed Solution & Use Cases 5
Problem Statement 5
Proposed Solution: 6
Use Cases 8
Profile Creation & Management Subsystem 8
Performance Management, Reporting, and Analytics Subsystem 9
Injury Management Subsystem 10
Attendance Management Subsystem 11
Team Management Subsystem 12
Event/Calendar Scheduling Subsystem 13
Announcements Subsystem 14
Individual Training Session Subsystem 15
Gym Slot Booking Subsystem 16
Transportation Booking Subsystem 17
Interaction Sequence Diagrams 18
Profile Creation and Management Subsystem 18
Use Case: Log into System 18
Use Case: View Profile 19
Use Case: Update Profile 20
Use Case: Create Profile 21
Use Case: Update Password 22
Use Case: View Users 23
Performance Management 24
Use Case: Add Athlete Performance Entry 24
Use Case: View Athlete Performance Entries 25
Use Case: Edit Athlete Performance Entries 26
Use Case: View Performance Graph 27
Use Case: View Average Performance Graph 28
Team Management Subsystem 29
Use Case: Create Team 29
Use Case: Edit Team 30
Use Case: View Team 31
Use Case: Add Athlete to Team 32
Use Case: Remove Athlete from Team 33

View Team Athletes 34
Individual Training Session Subsystem 35
Use Case: Complete Session Completion Form 35
Use Case: Create Session Completion Form 36
Use Case: View Session Completion Form 37
Use Case: Edit Session Completion Form 38
Use Case: Delete Session Completion Form 39
Use Case: View Athlete Session Completion Graph 40
Use Case: View Average Session Completion Graph 41
Attendance Management Subsystem 42
Use Case: Submit Attendance Code 42
Use Case: Show Attendance Code 43
Use Case: View Athlete Attendance Graph 44
Use Case: View Average Attendance Graph 45
Event Scheduling Subsystem 46
Use Case: Add Event 46
Use Case: View Event 47
Use Case: Edit Event 48
Use Case: Delete Event 49
Use Case: View Calendar 50
Announcement Subsystem 51
Use Case: Add Announcement 51
Use Case: Delete Announcement 52
Use Case: View Announcement 53
Use Case: Edit Announcement 54
Injury Management Subsystem 55
Use Case: Upload Athlete Progress Report 55
Use Case: Download Athlete Progress Report 56
Gym Slot Booking Subsystem 57
Use Case: View Gym Slot Availability 57
Use Case: Book Gym Slot 58
Use Case: Cancel Gym Slot Booking 59
Transportation Booking Subsystem 60
Use Case: Submit Transportation Request 60
Use Case: View Transportation Request 61

Use Case: Delete Transportation Request 62
Use Case: Cancel Booking .....**Error! Bookmark not defined.**
Use Case: Submit Transportation Confirmation 63
Class Diagram..... 64

Updated Problem Statement, Proposed Solution & Use Cases

Problem Statement

The current athlete management system at UJ Sport is riddled with inefficiencies and limitations, impeding the organization's ability to effectively manage athlete-related activities and operational logistics. These challenges have manifested in these key areas:

1. Lack of Integration: The existing systems and manual methods employed by UJ Sport operate in silos, lacking integration and coherence. Data fragmentation and communication gaps hinder collaboration and decision-making processes. Coaches, athletes, and managers struggle to access and share critical information seamlessly, leading to disjointed workflows and missed opportunities for optimization.

2. Inefficient Communication: Informal communication channels, such as word of mouth and messaging platforms like WhatsApp, have become the primary means of disseminating information regarding training schedules, updates, and event notifications. However, these channels are prone to inconsistencies, resulting in missed updates, scheduling conflicts, and confusion among stakeholders. As a result, athletes may miss important sessions, coaches may overlook performance updates, and managers may struggle to coordinate logistics effectively.

3. Manual Booking Processes: Booking procedures for gym slots and sports buses are characterized by tedious manual processes, consuming valuable time and resources. Coaches and administrators must navigate through multiple channels and engage in time-consuming phone calls to secure bookings. This inefficiency not only impacts operational efficiency but also contributes to athlete dissatisfaction and frustration. Moreover, the lack of real-time availability updates and capacity management exacerbates scheduling conflicts and compromises the athlete experience.

4. Data Inaccuracy and Redundancy: The prevalence of manual data entry and disparate systems contributes to data inaccuracy and redundancy. Inconsistent data entry practices and the absence of centralized data management result in duplicate records, outdated information, and inaccuracies in athlete profiles, performance metrics, and scheduling details.

5. Limited Access to Critical Information: The lack of centralized access to critical information hampers decision-making and performance analysis. Coaches, athletes, and managers struggle to access comprehensive athlete profiles, performance trends, and scheduling updates, hindering their ability to make informed decisions and optimize training strategies.

6. Inefficient Injury Management: The absence of a dedicated injury management module complicates the process of tracking and managing athlete injuries. Coaches, athletes, and medical professionals lack a centralized platform to record injury

history, track rehabilitation progress, and communicate treatment plans, leading to delays in recovery and increased risk of re-injury.

7. Limited Coaching Support: Coaches lack comprehensive tools to provide personalized feedback, training plans, and performance evaluations to athletes. The absence of interactive coaching tools hampers the coaching process, hindering skill enhancement and technique refinement for athletes.

These challenges collectively undermine UJ Sport's ability to deliver optimal athlete performance, facilitate effective communication, and streamline operational workflows. The disjointed nature of existing systems and procedures results in suboptimal performance, missed opportunities for improvement, and overall dissatisfaction among athletes, coaches, and managers. Urgent intervention is required to address these issues and implement a comprehensive solution that integrates systems, enhances communication channels, and systemises booking processes to drive efficiency and improve the athlete experience.

Proposed Solution:

To address these challenges, we propose the implementation of the UJ Sport Athlete and Booking Management System. This system will provide a centralized platform for managing athlete-related activities and operational logistics, offering the following key features:

Comprehensive Athlete Profiles: Detailed profiles containing essential athlete information, demographics, medical history, and performance metrics.

Dynamic Scheduling and Calendar: Streamlined scheduling capabilities for managing training sessions, competitions, and personal appointments, integrated with external calendars and notification systems.

Performance metrics: Dedicated athlete performance tracking enabling monitoring of training progress and performance metrics.

Virtual Injury Management: Seamless communication between athletes, coaches, and medical professionals for tracking injuries.

Interactive Coaching and Feedback: Tools for personalized feedback, training plans, and performance evaluations.

Enhanced Team Management: Features for roster organization, communication, and collaboration, including group training schedules and coordination tools.

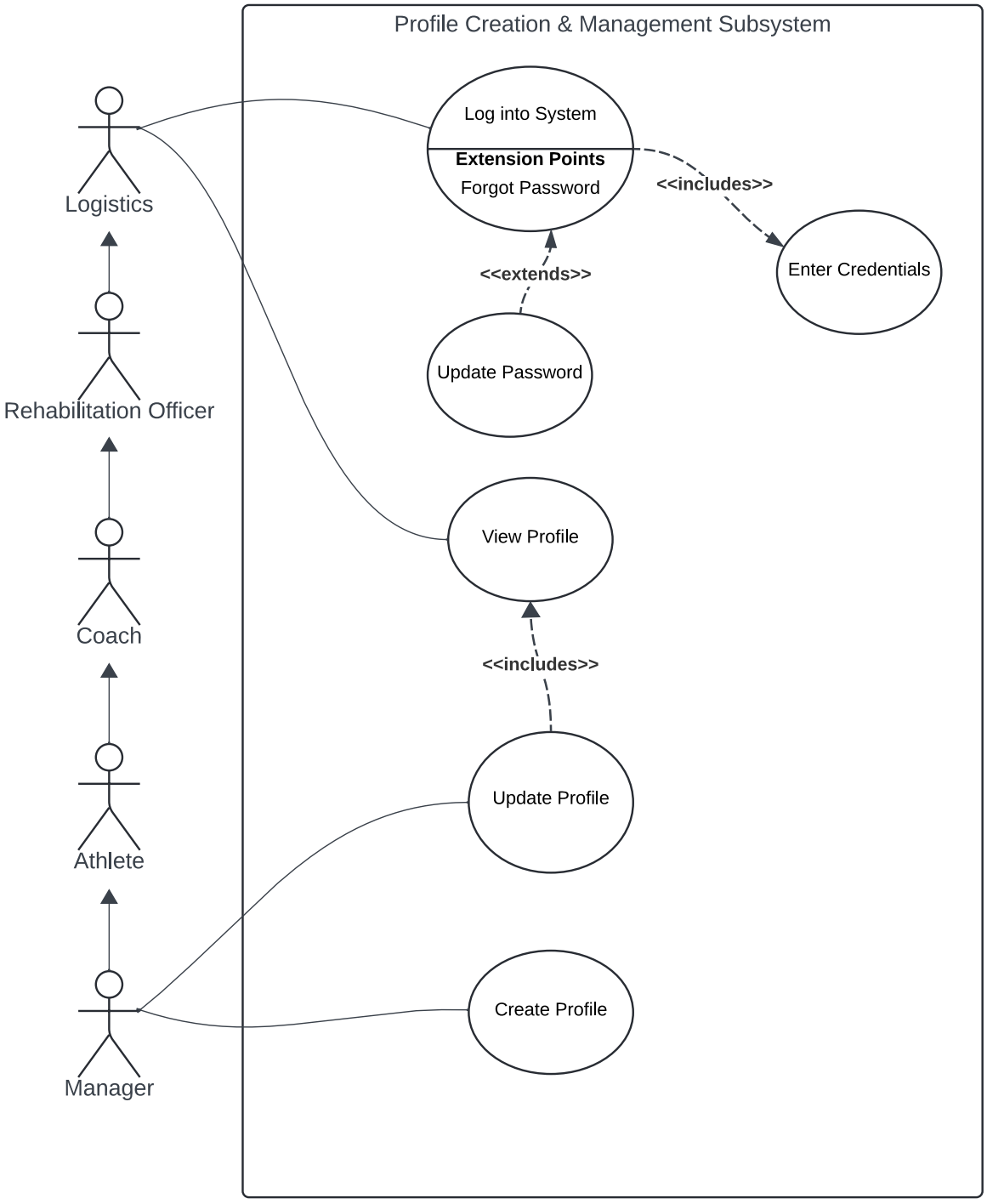
Robust Reporting and Analytics: Comprehensive reporting and analytics capabilities providing valuable insights into athlete performance, team dynamics, and strategic planning.

Additionally, the system will include efficient booking systems for gym slots and transportation, offering real-time availability updates and capacity management to optimize facility usage and streamline transportation logistics.

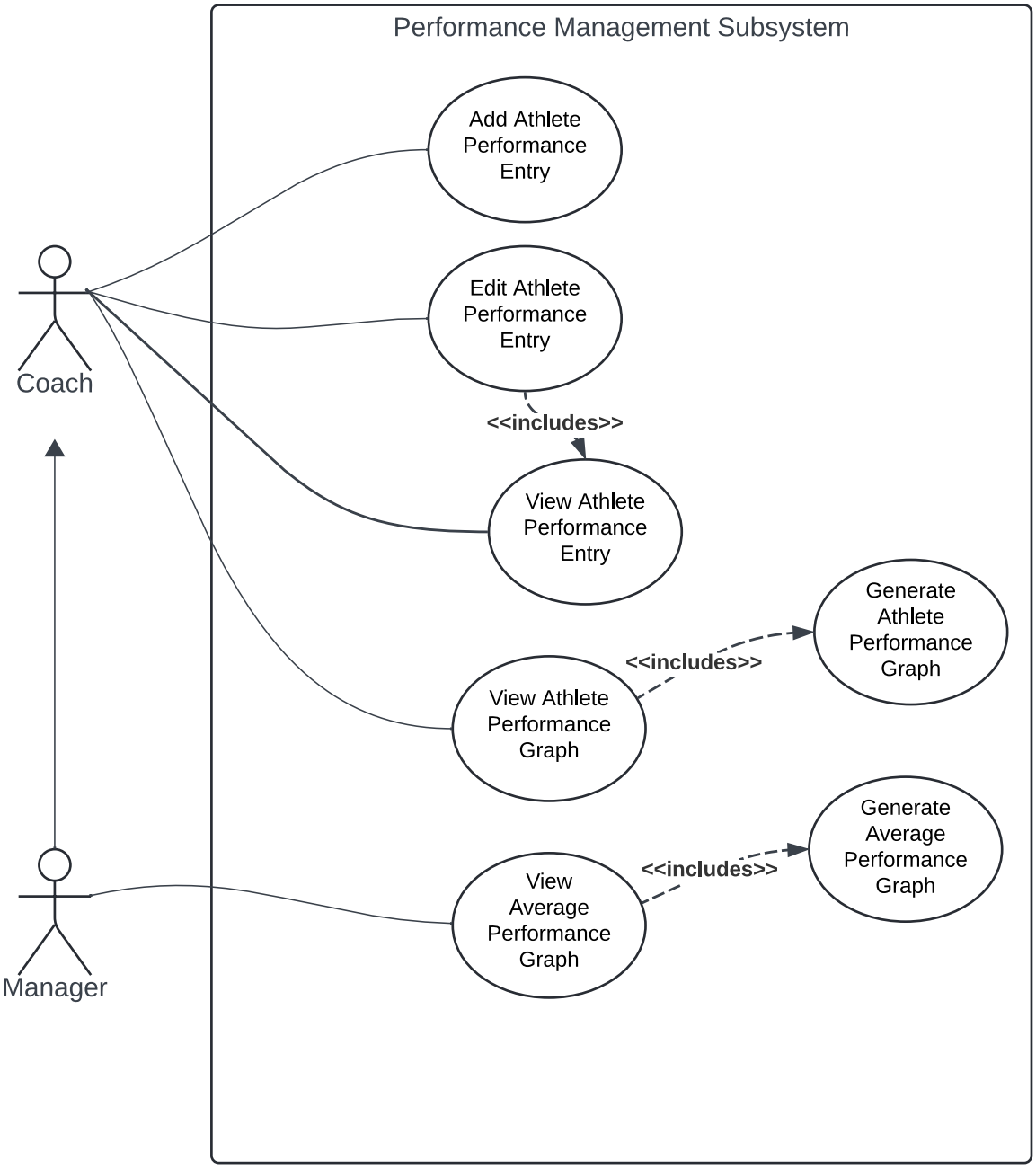
By implementing this solution, UJ Sport will significantly enhance its operations, athlete performance, and overall efficiency, empowering athletes and coaches to achieve new levels of success.

Use Cases

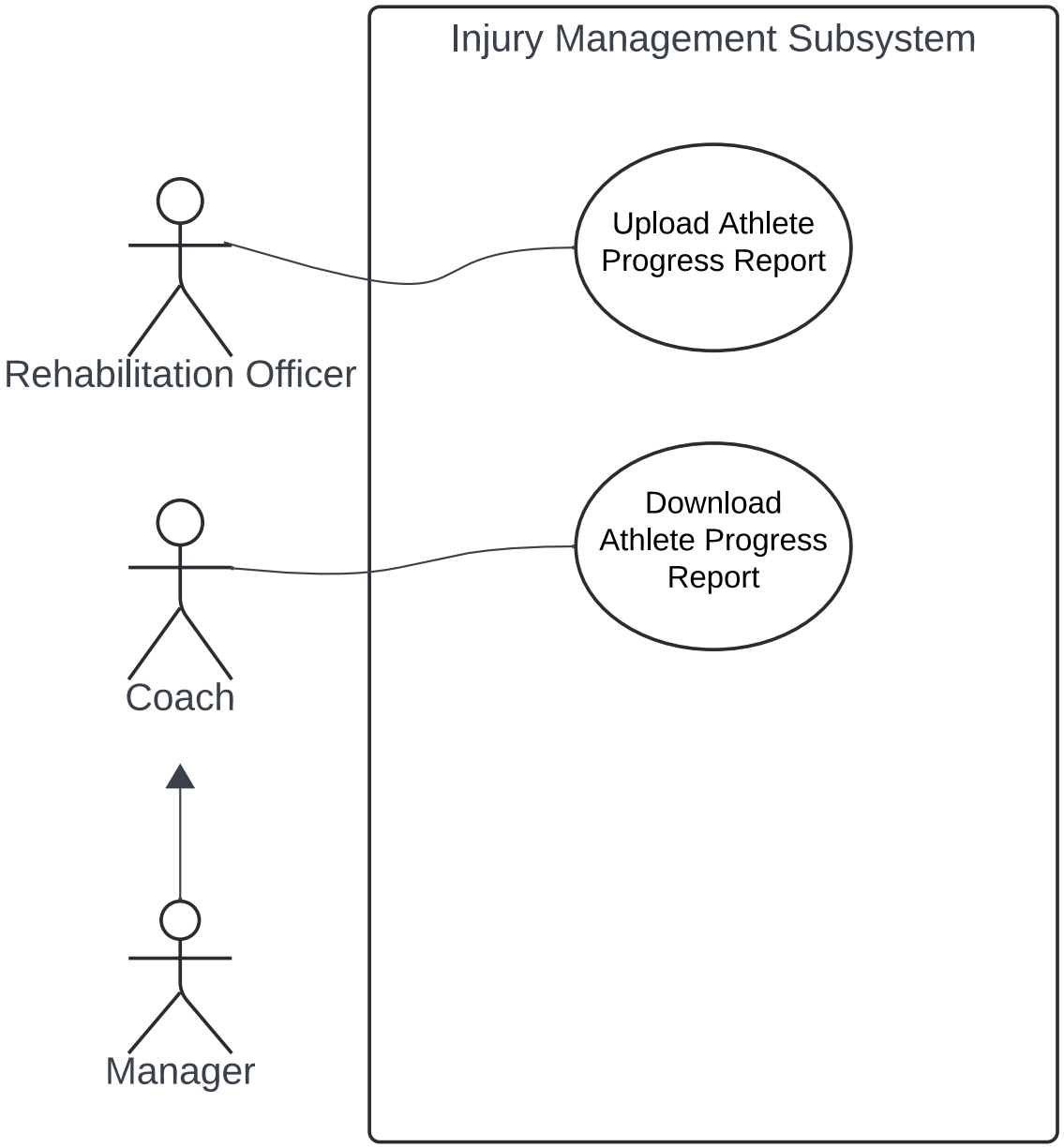
Profile Creation & Management Subsystem



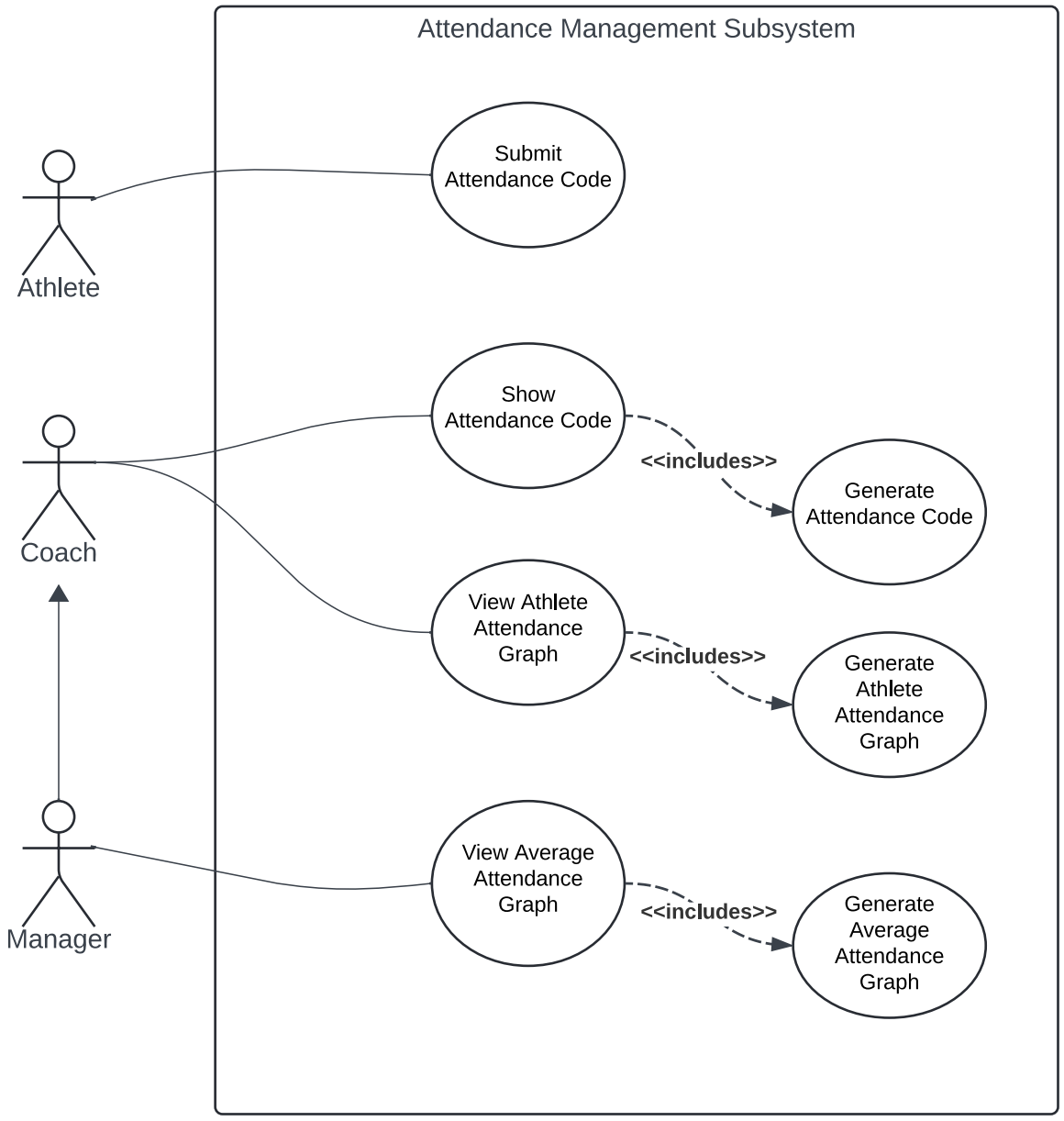
Performance Management, Reporting, and Analytics Subsystem



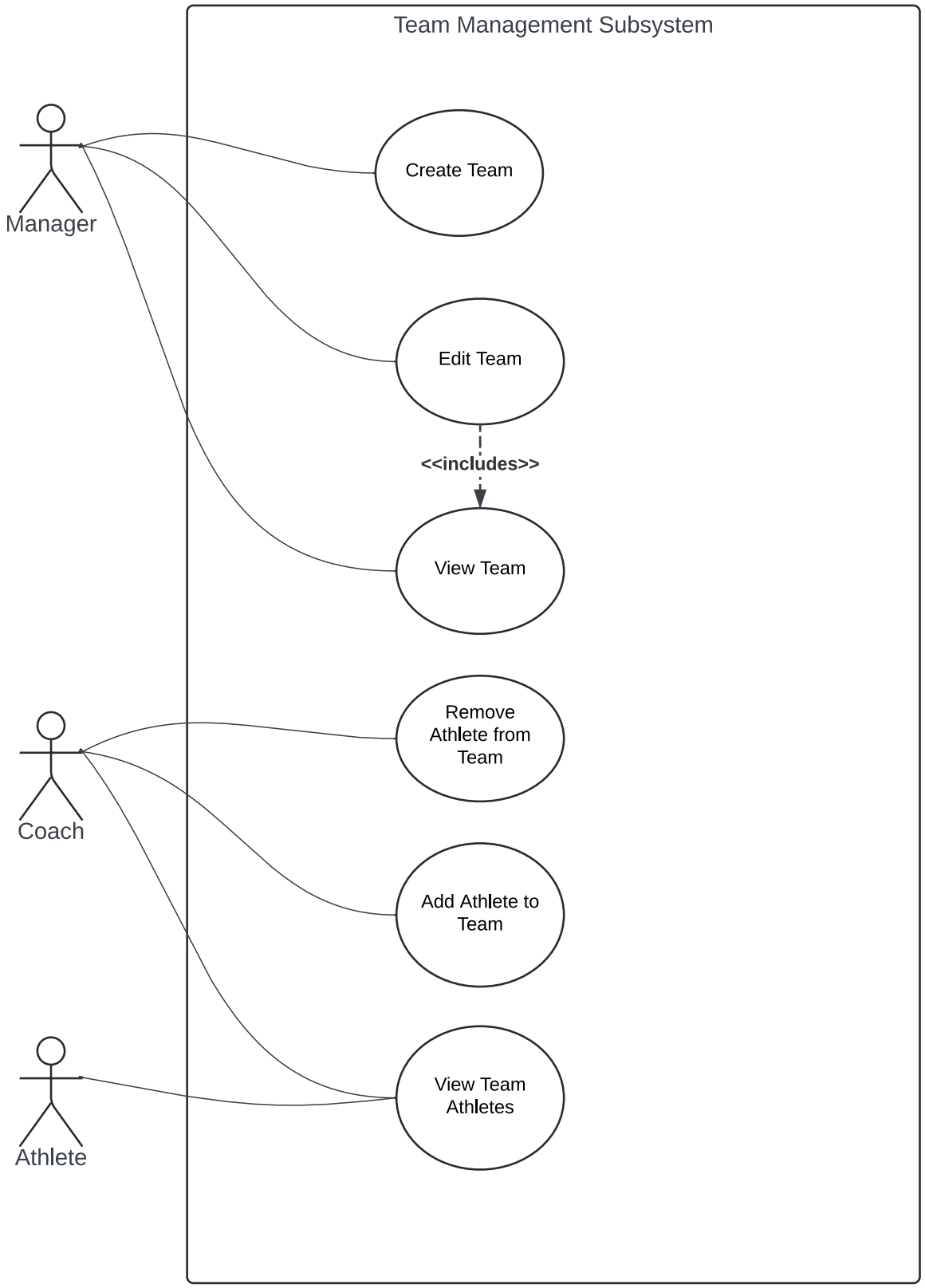
Injury Management Subsystem



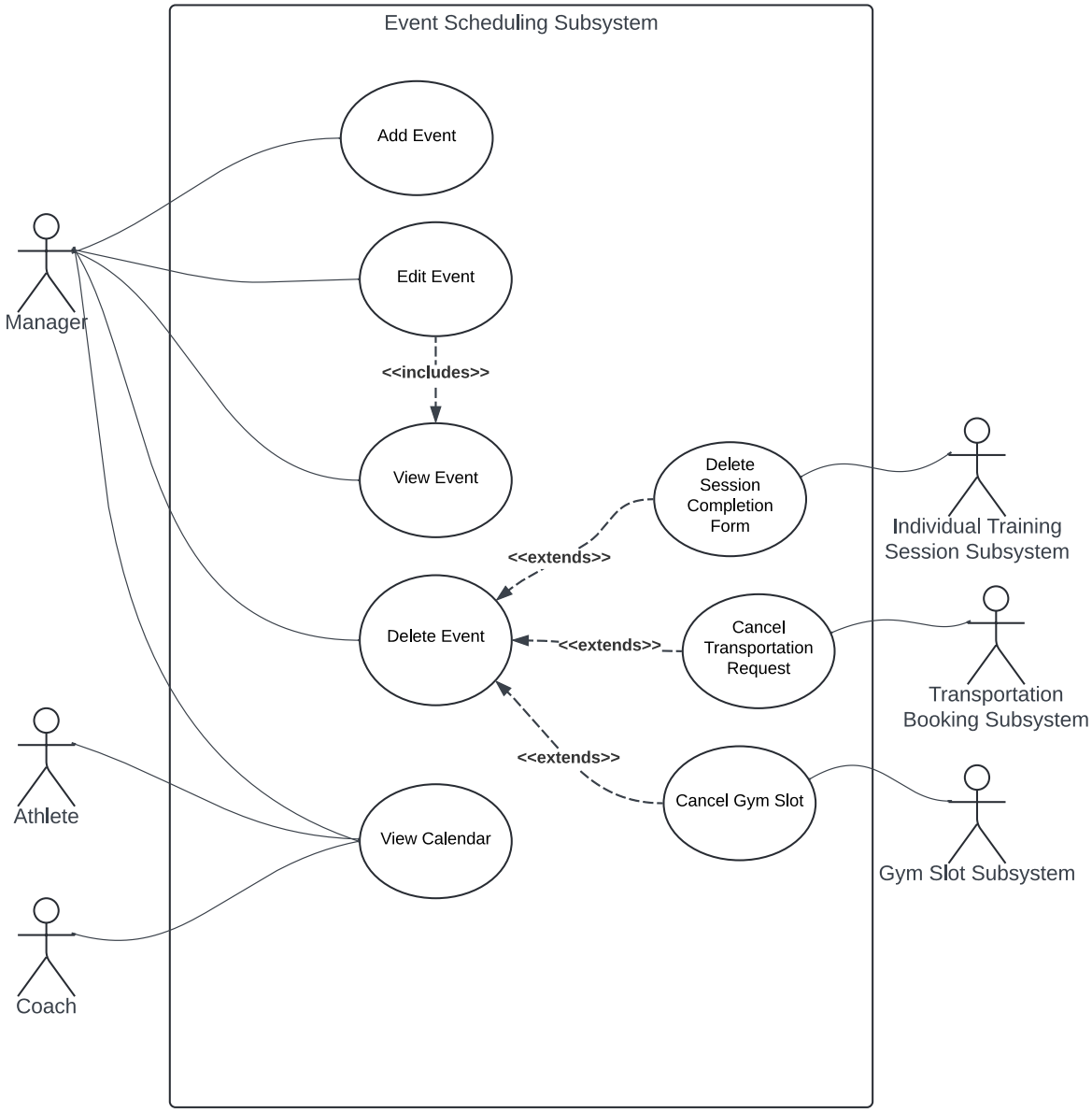
Attendance Management Subsystem



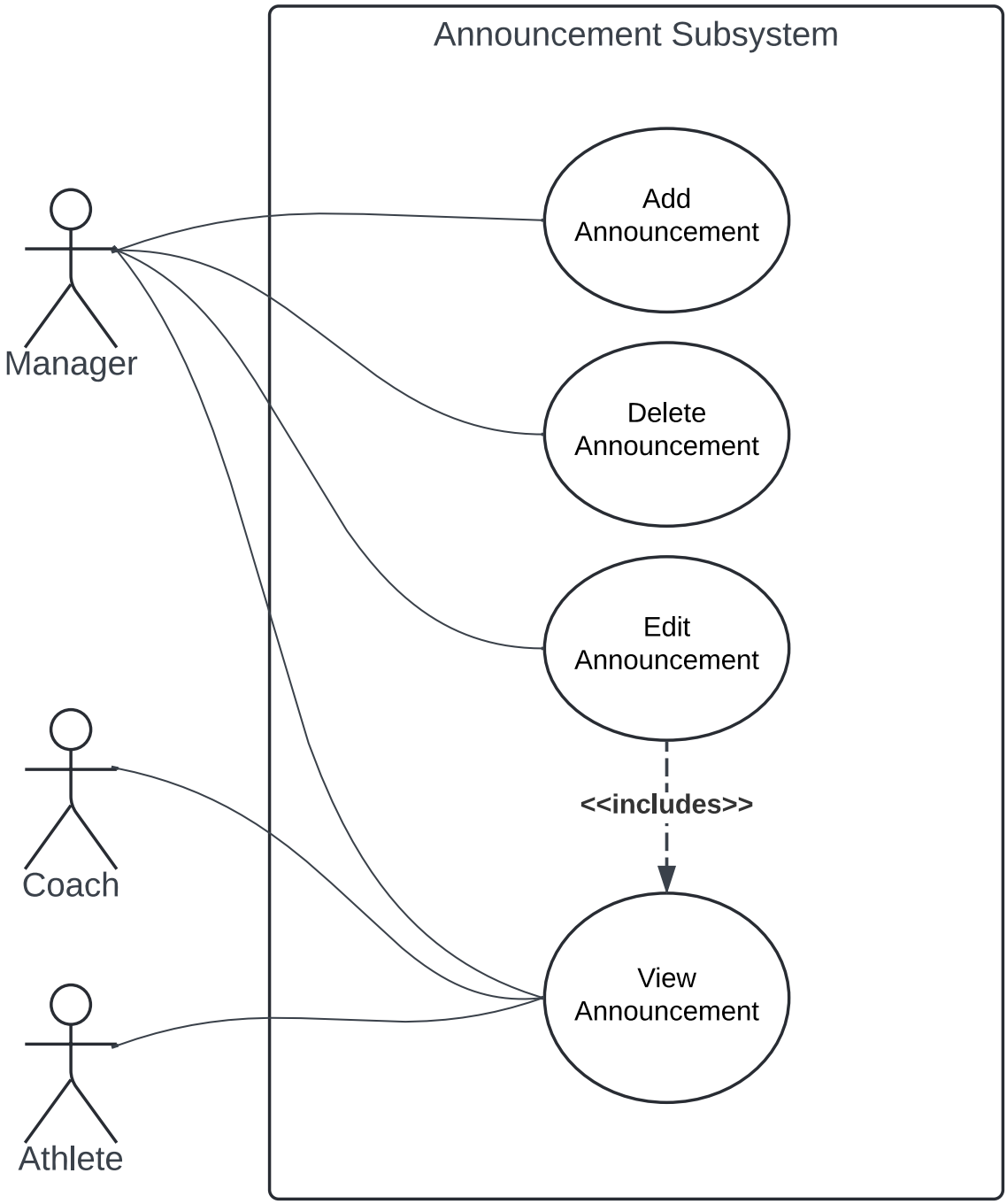
Team Management Subsystem



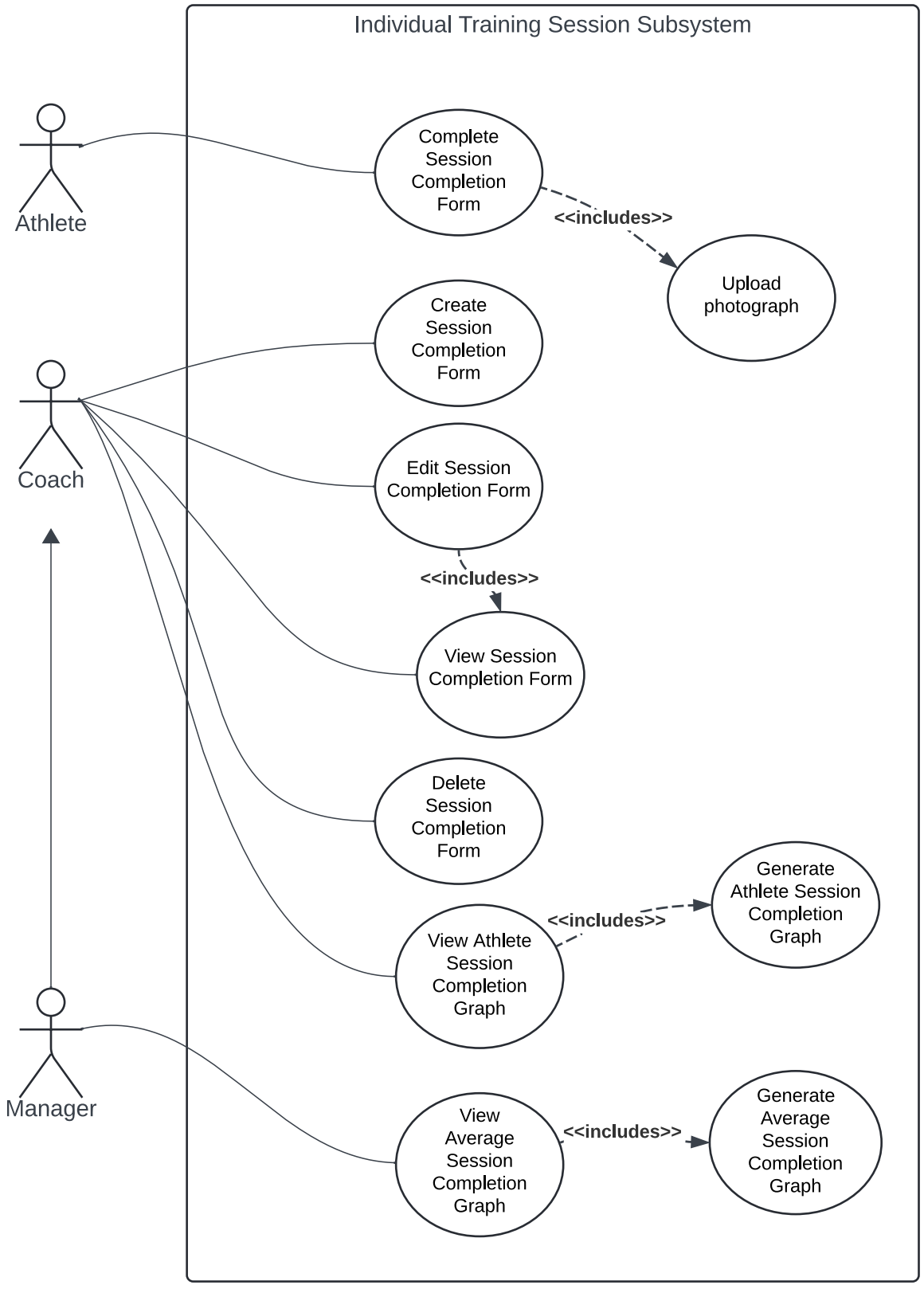
Event/Calendar Scheduling Subsystem



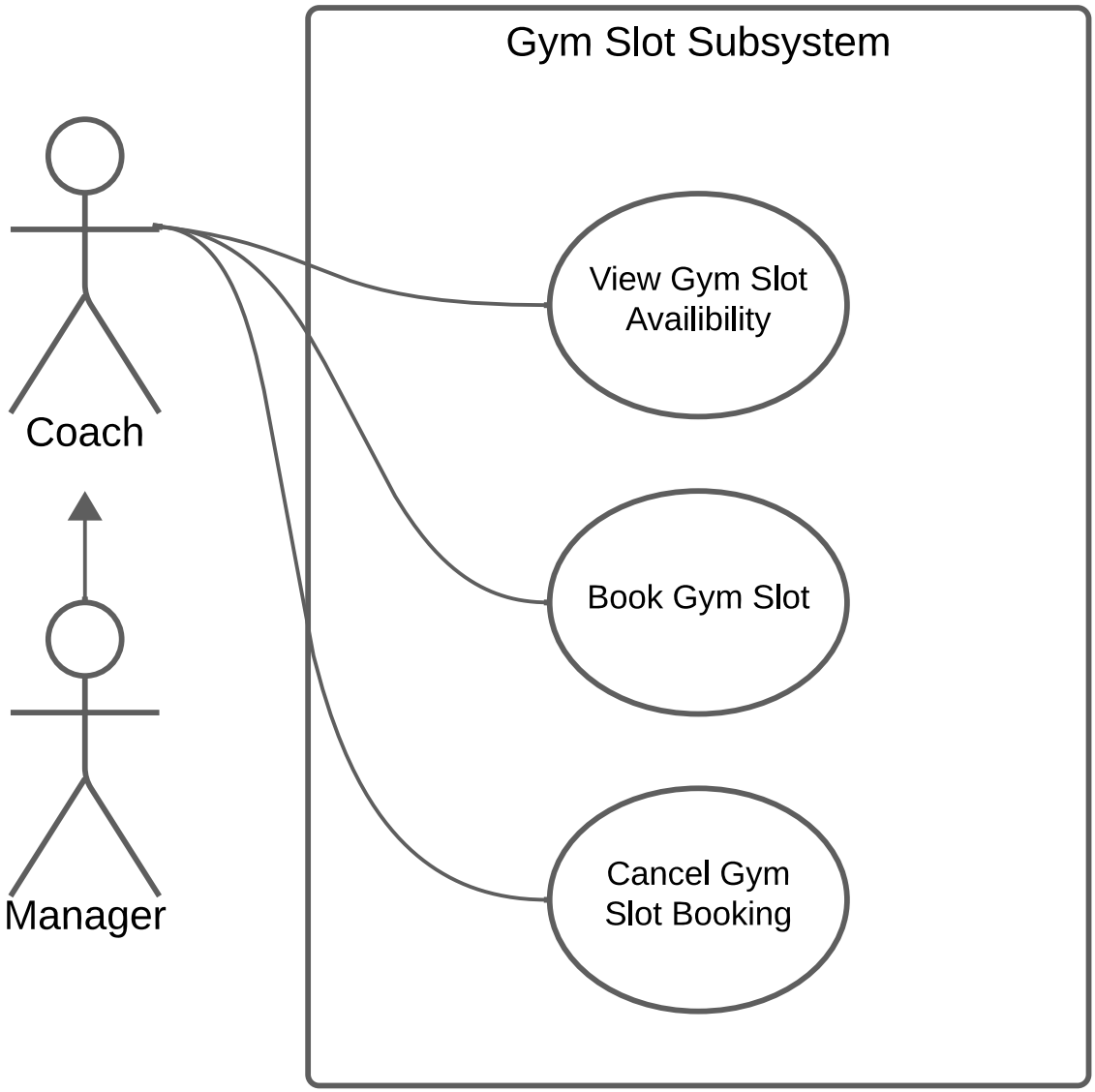
Announcements Subsystem



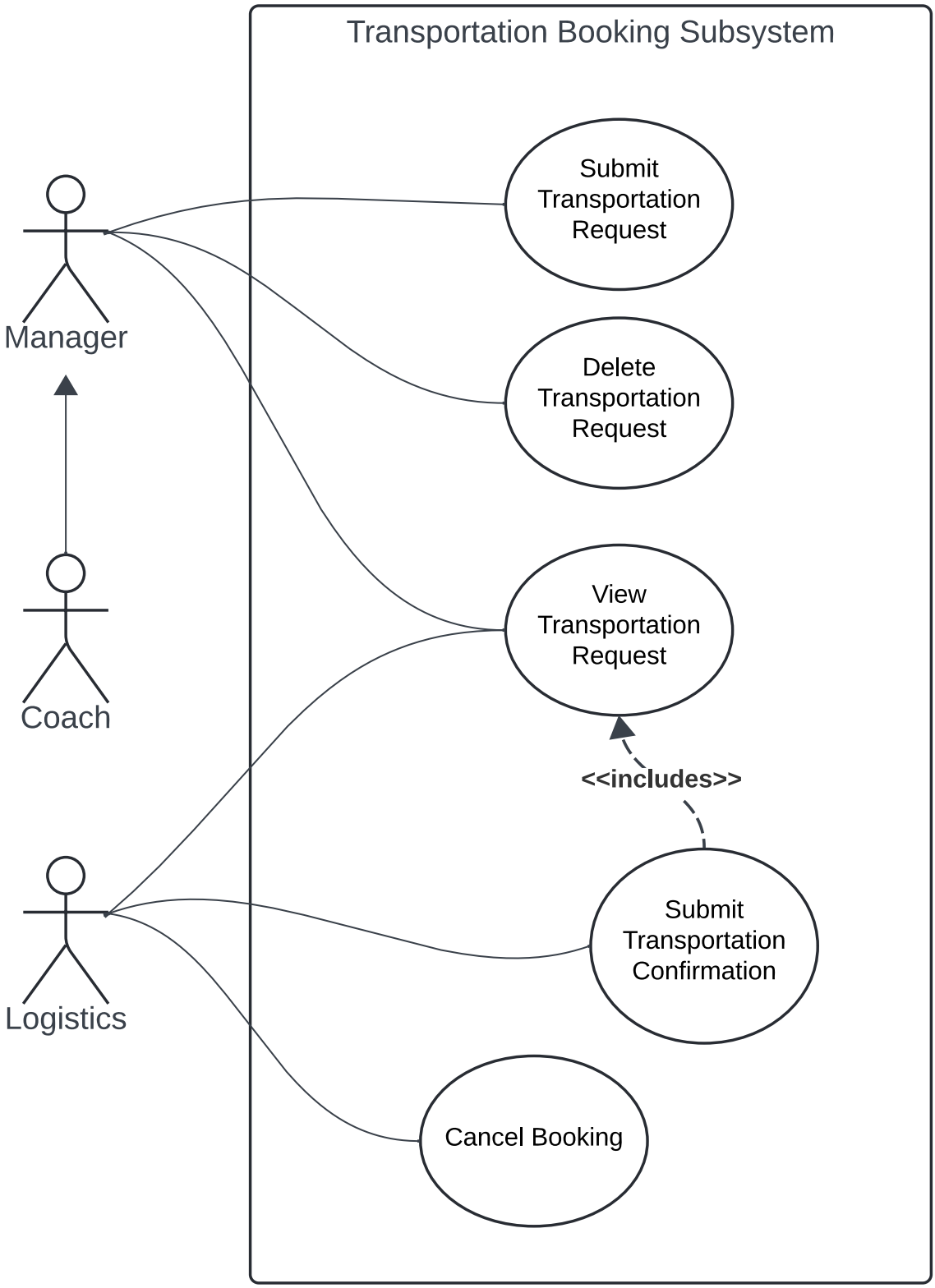
Individual Training Session Subsystem



Gym Slot Booking Subsystem



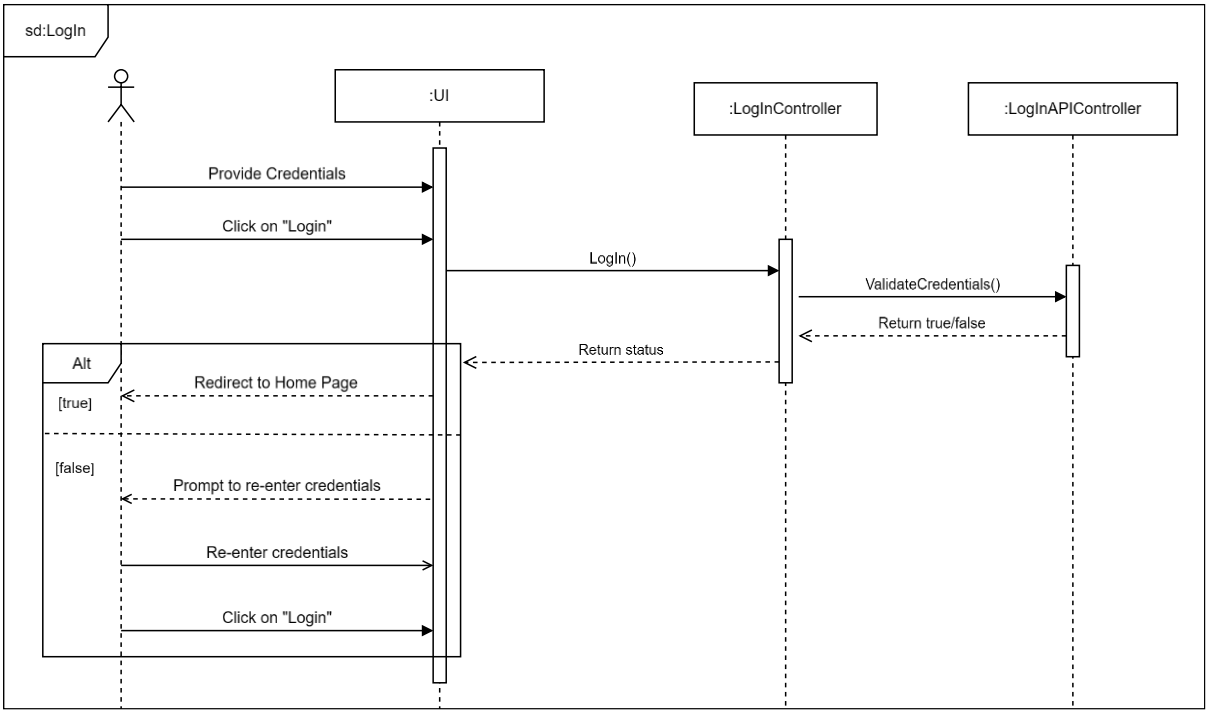
Transportation Booking Subsystem



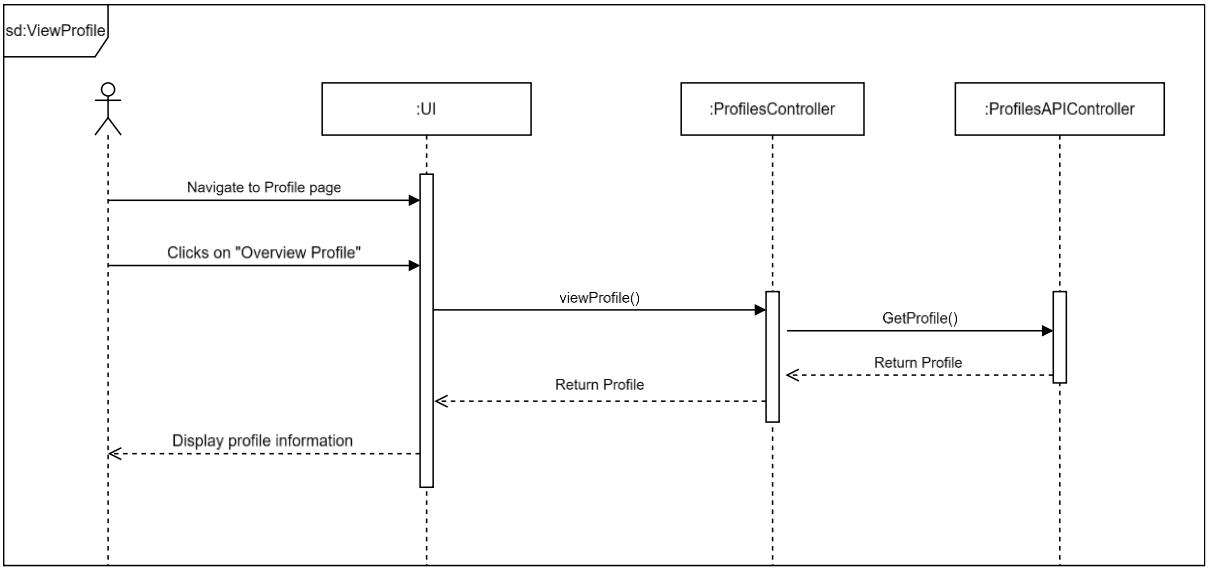
Interaction Sequence Diagrams

Profile Creation and Management Subsystem

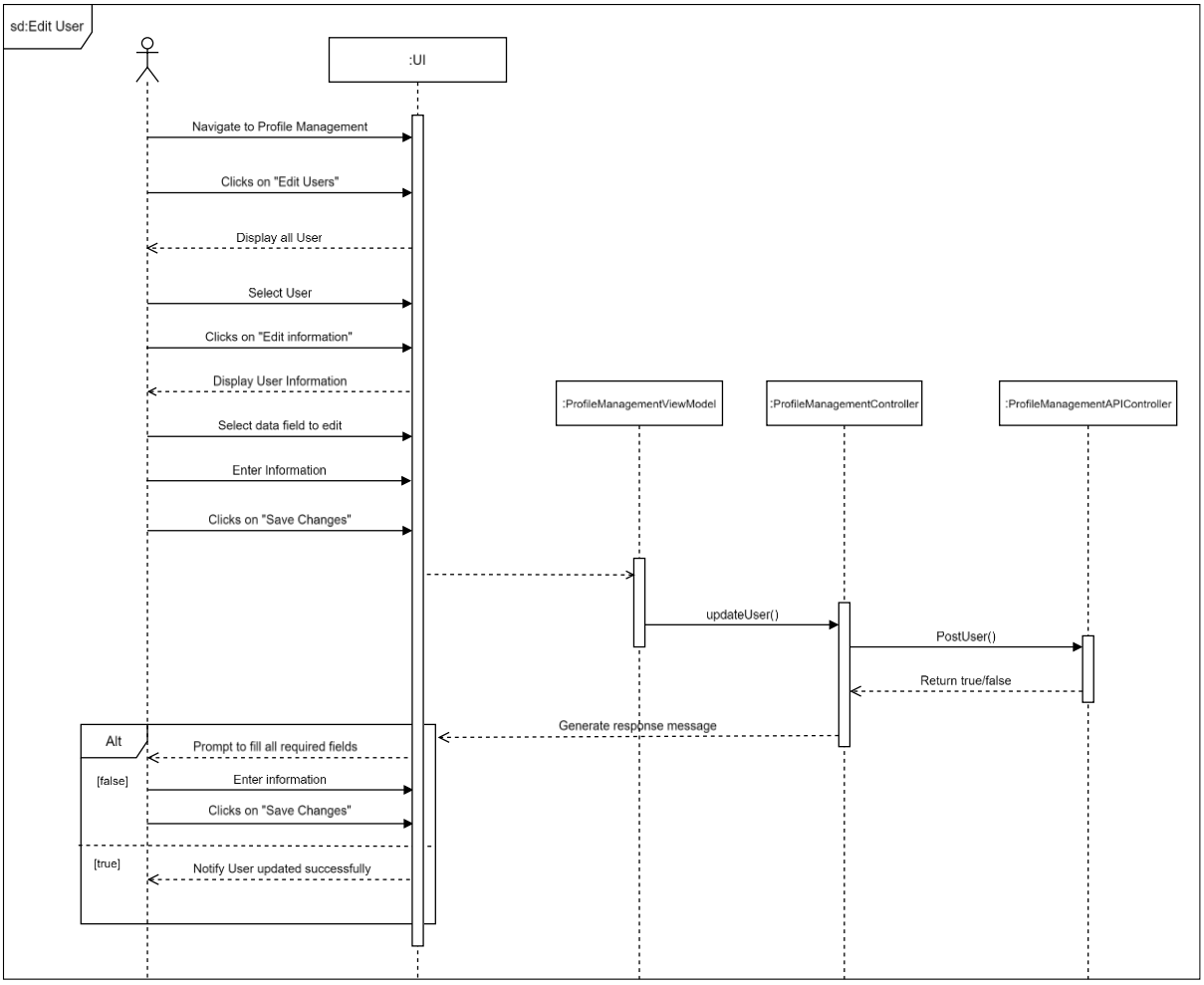
Use Case: Log into System



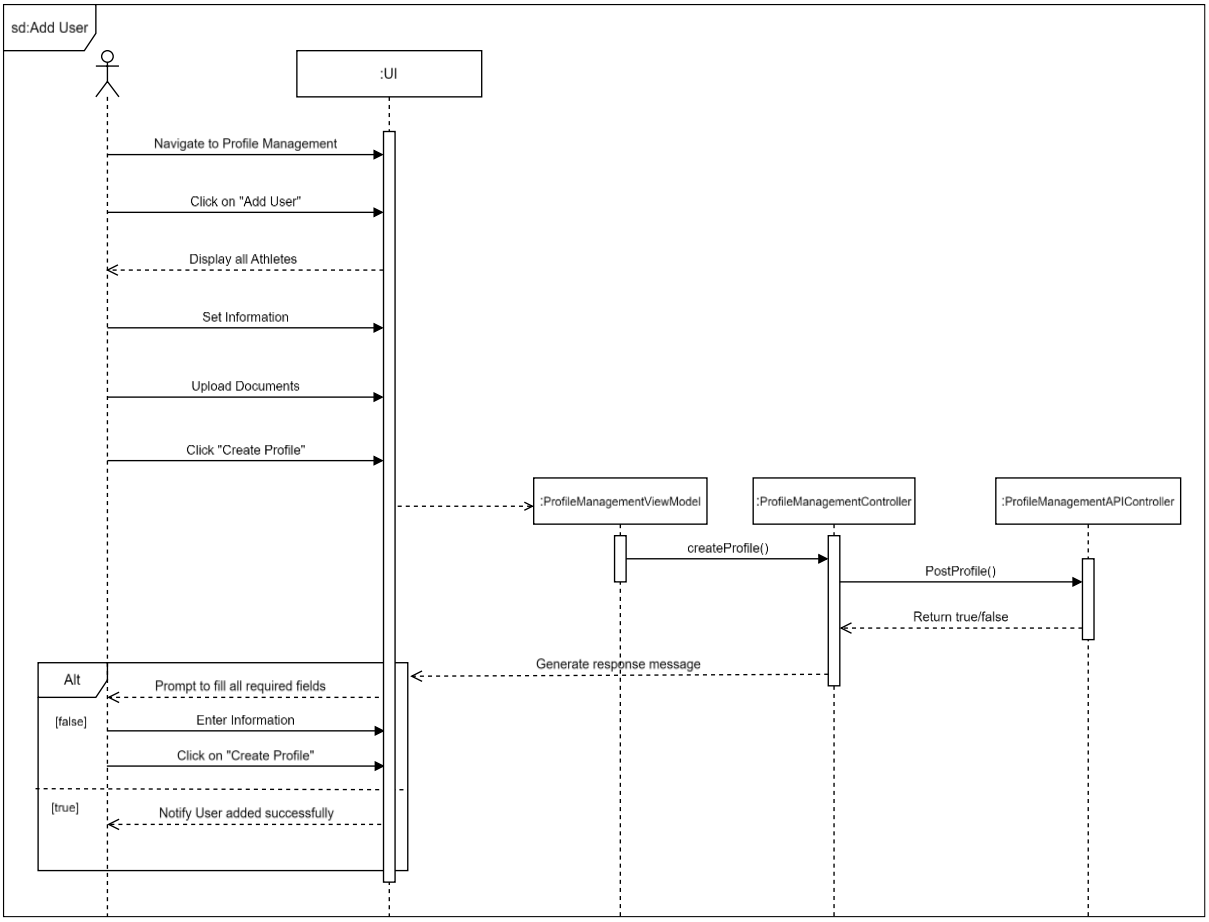
Use Case: View Profile



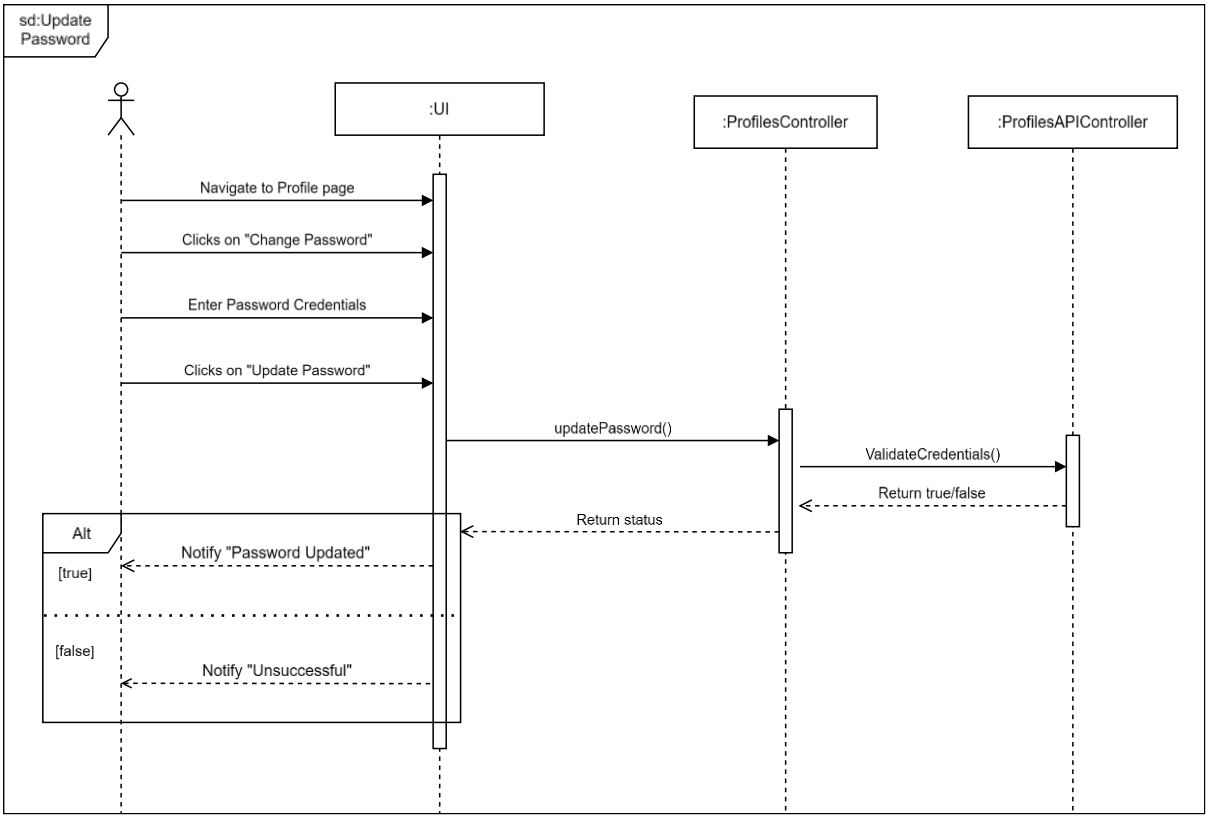
Use Case: Update Profile



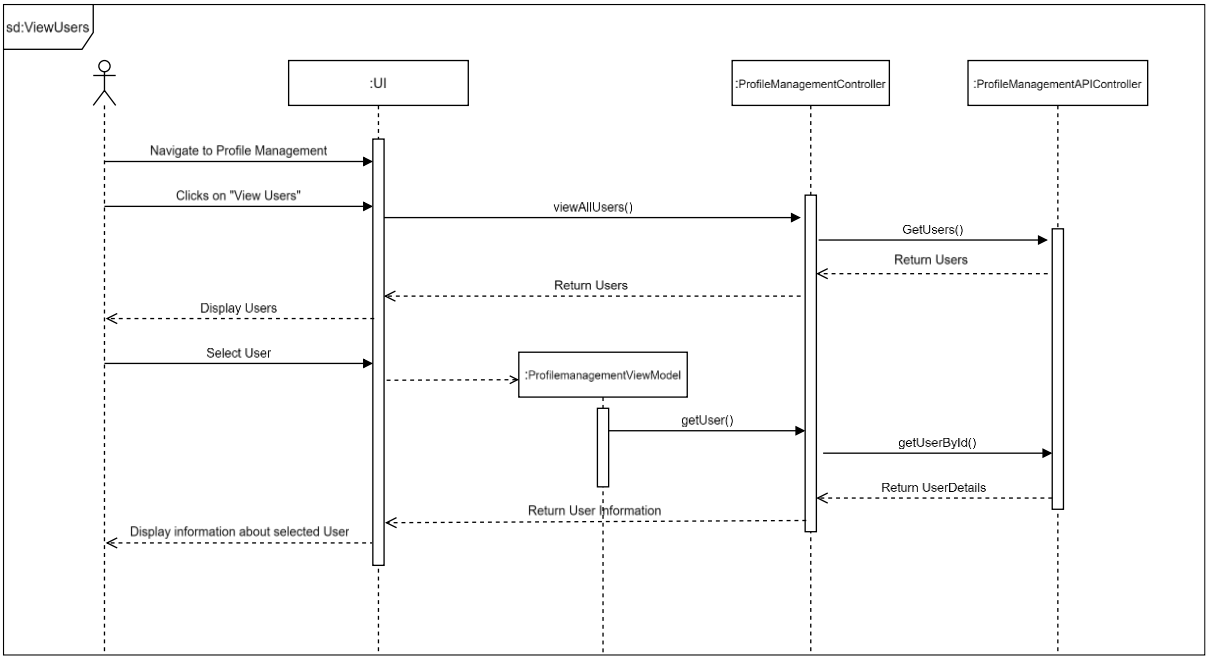
Use Case: Create Profile



Use Case: Update Password

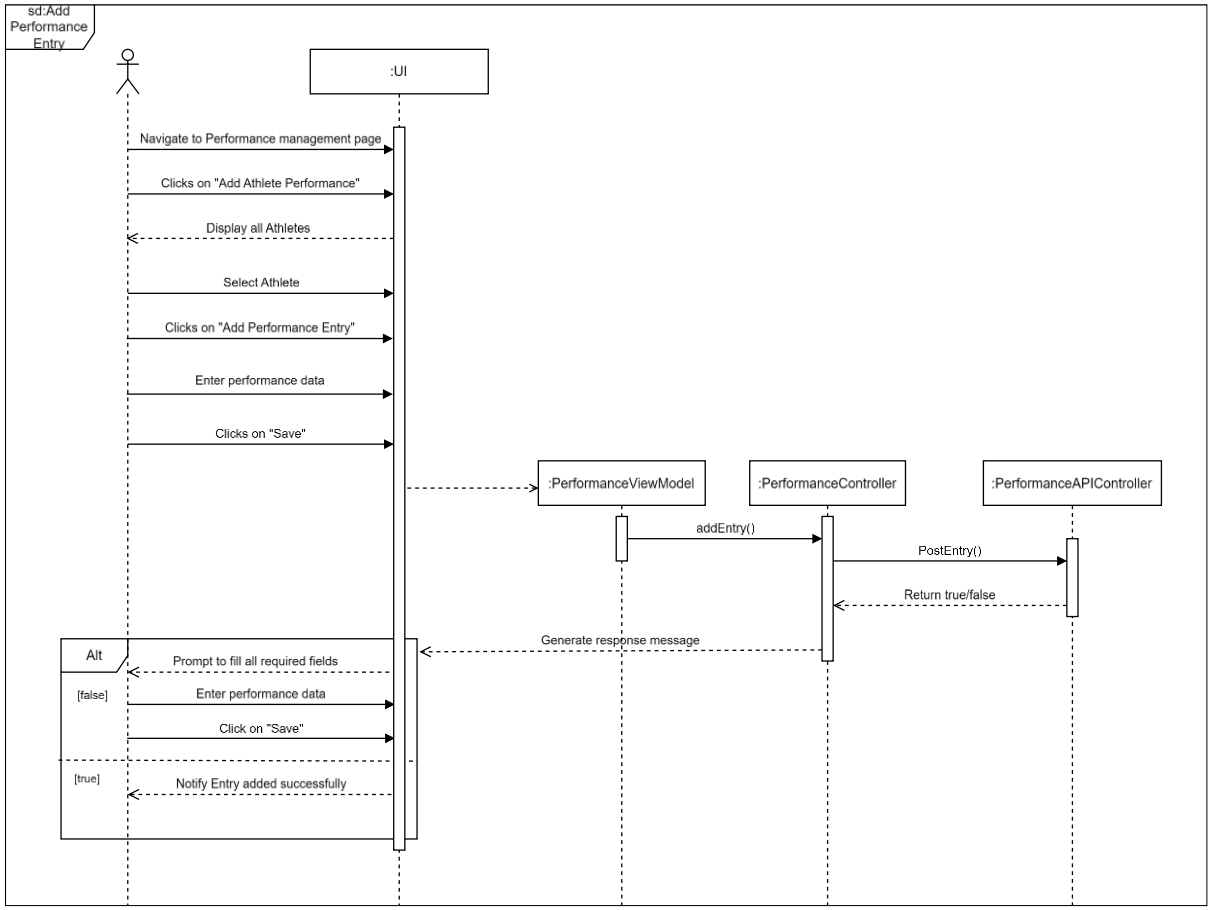


Use Case: View Users

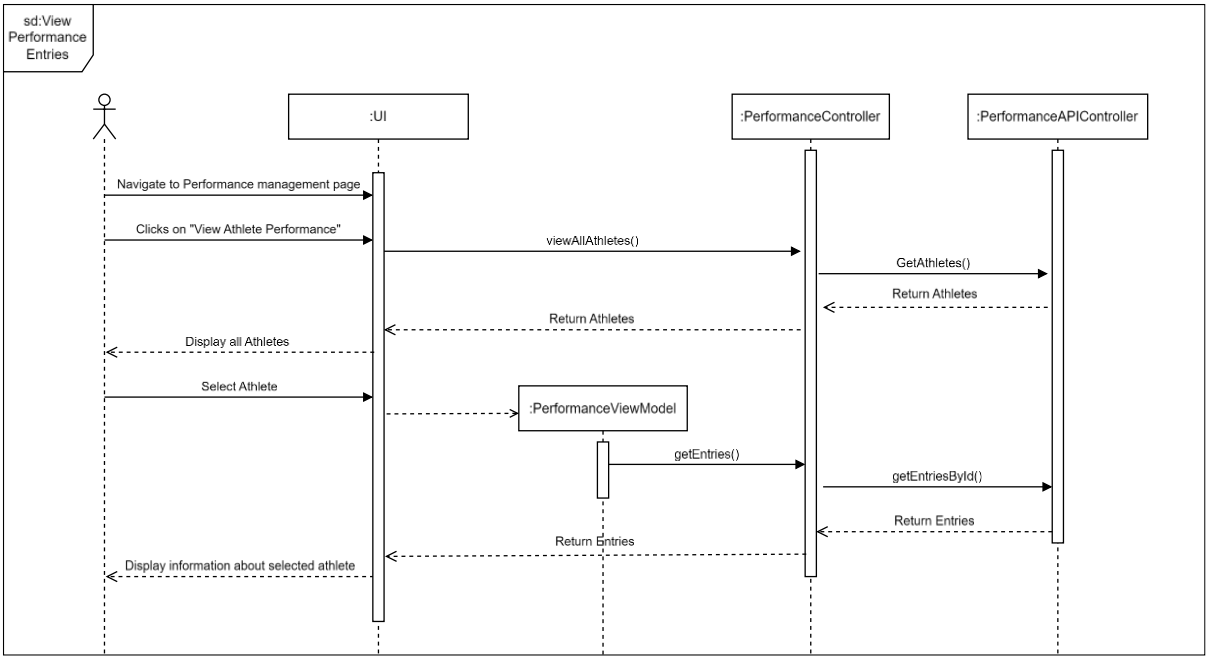


Performance Management

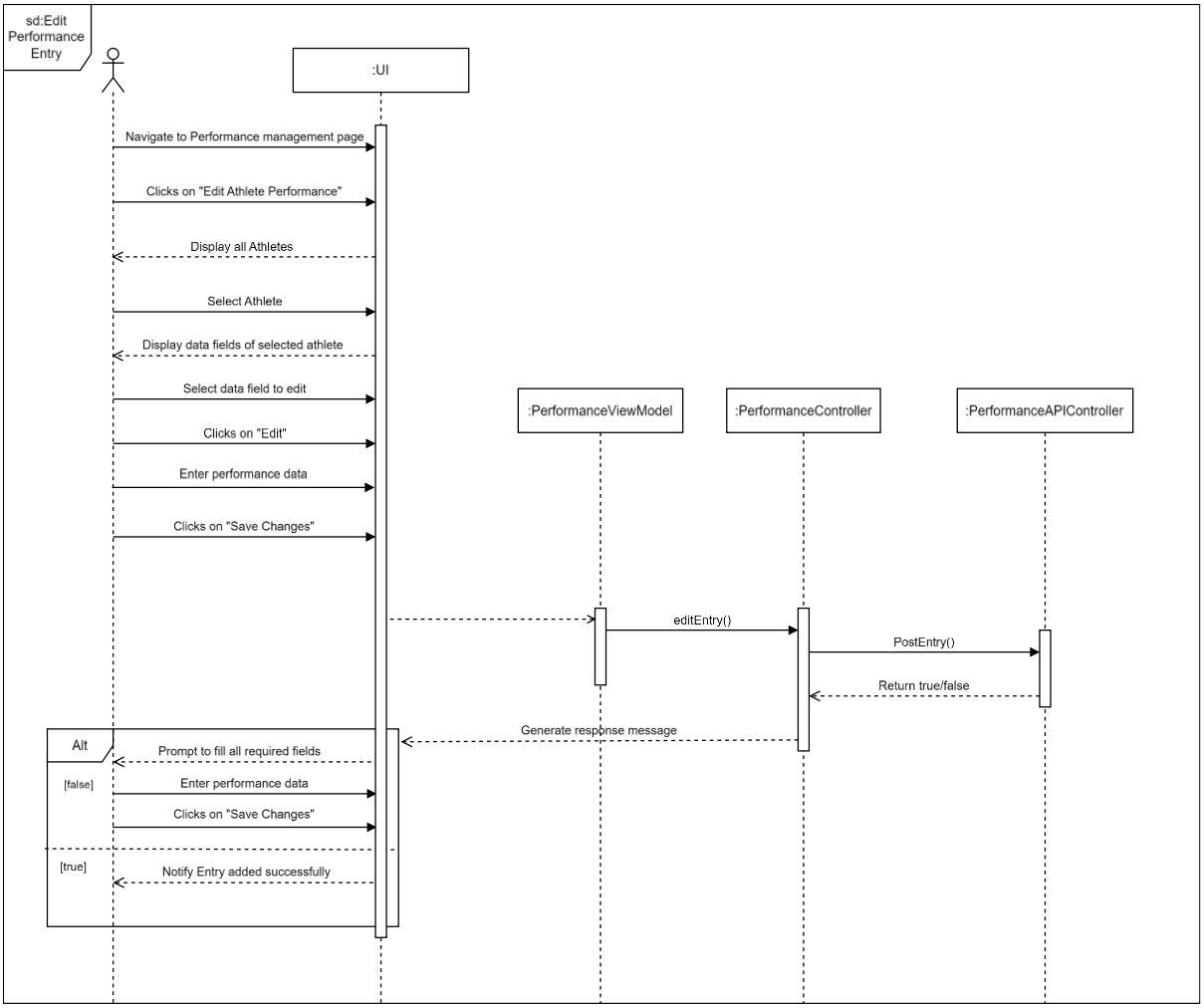
Use Case: Add Athlete Performance Entry



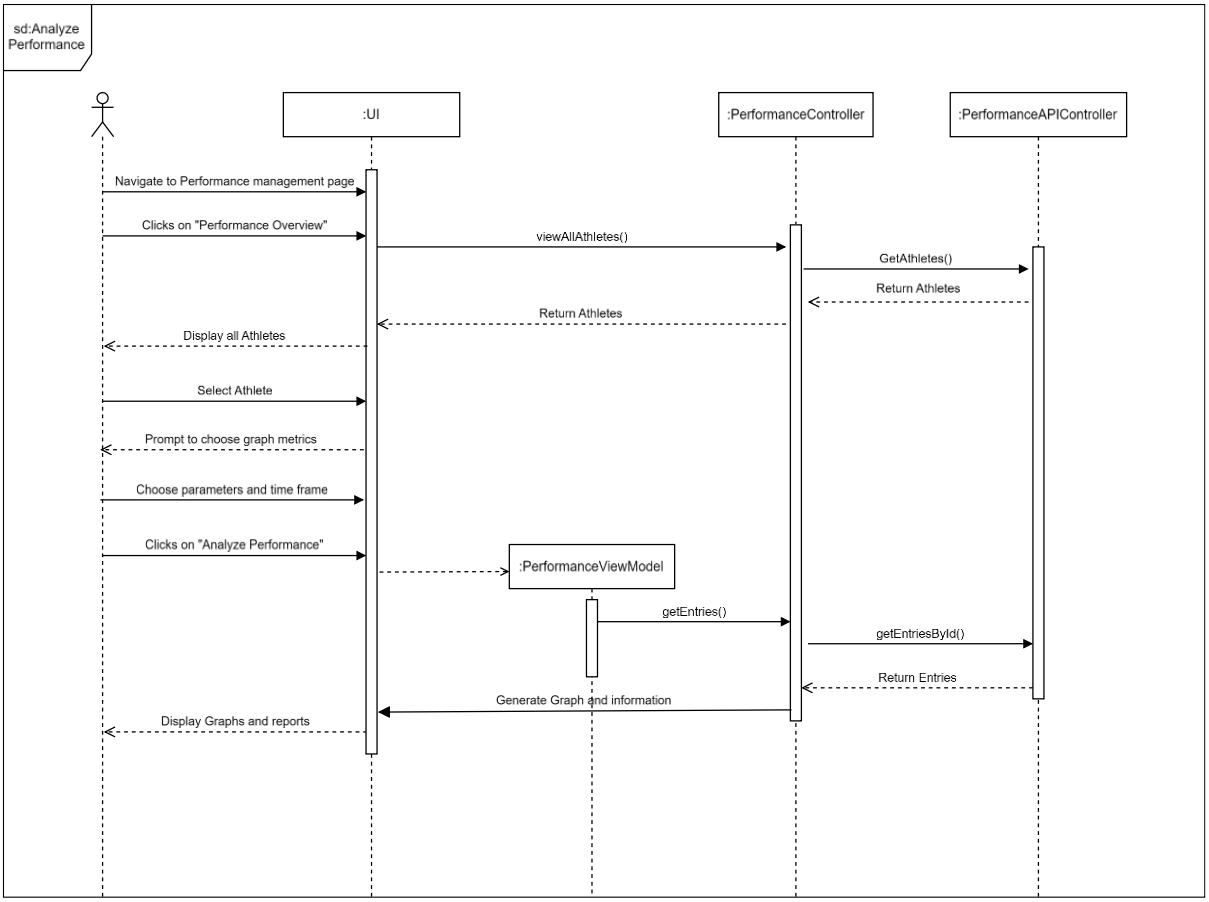
Use Case: View Athlete Performance Entries



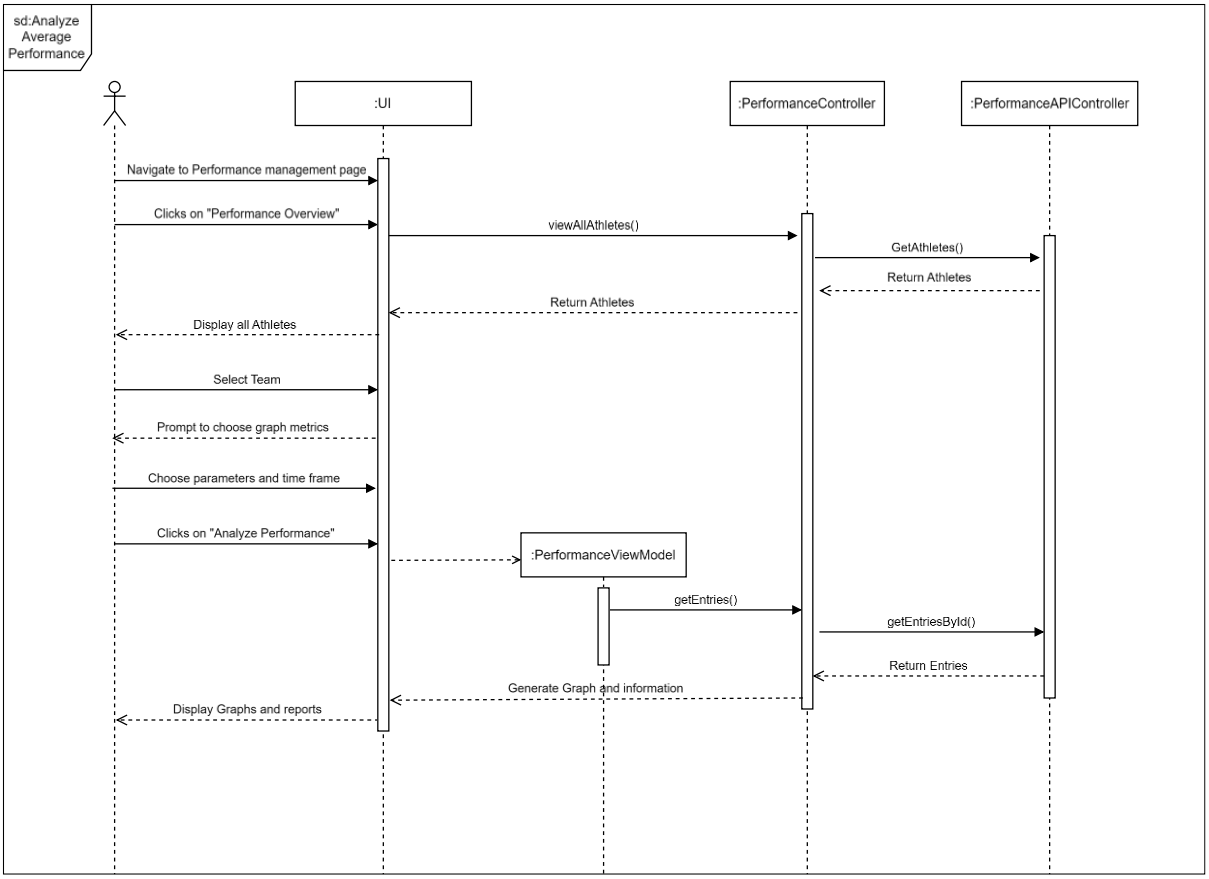
Use Case: Edit Athlete Performance Entries



Use Case: View Performance Graph

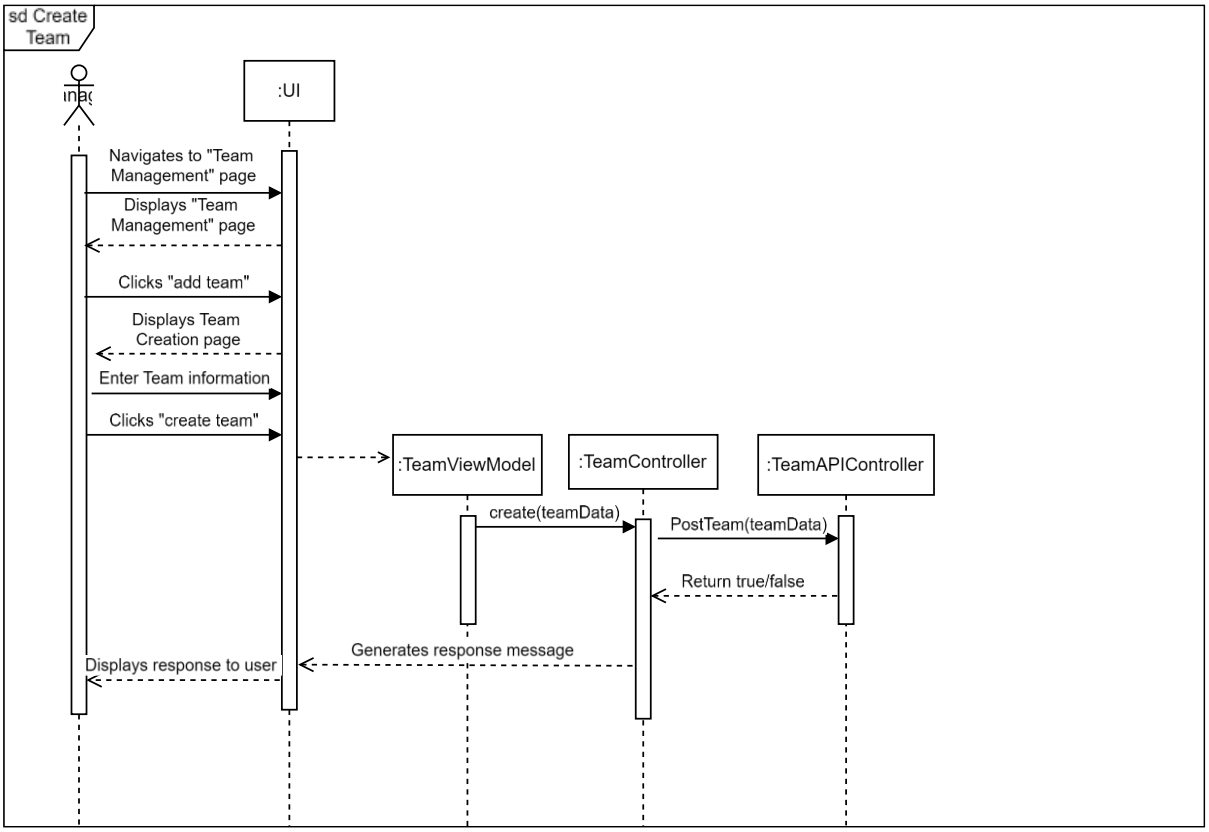


Use Case: View Average Performance Graph

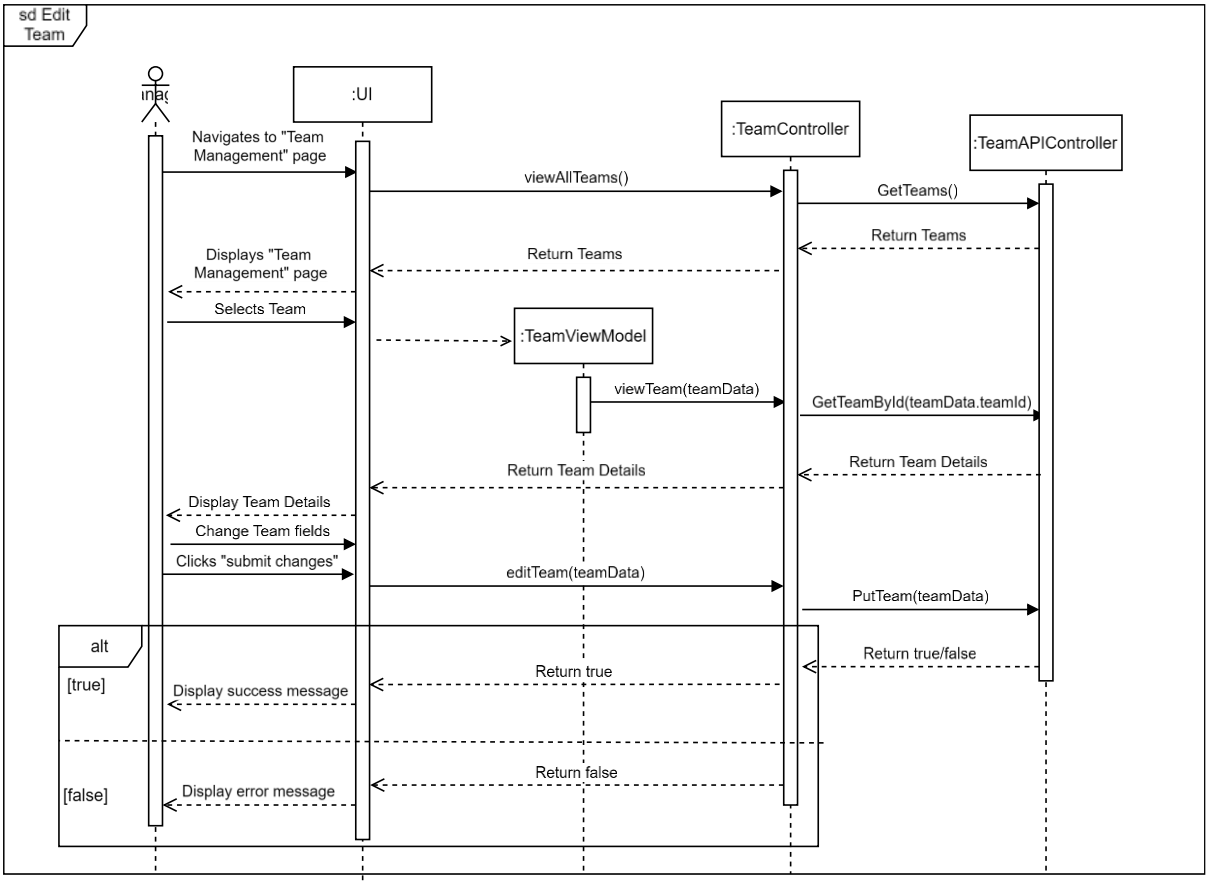


Team Management Subsystem

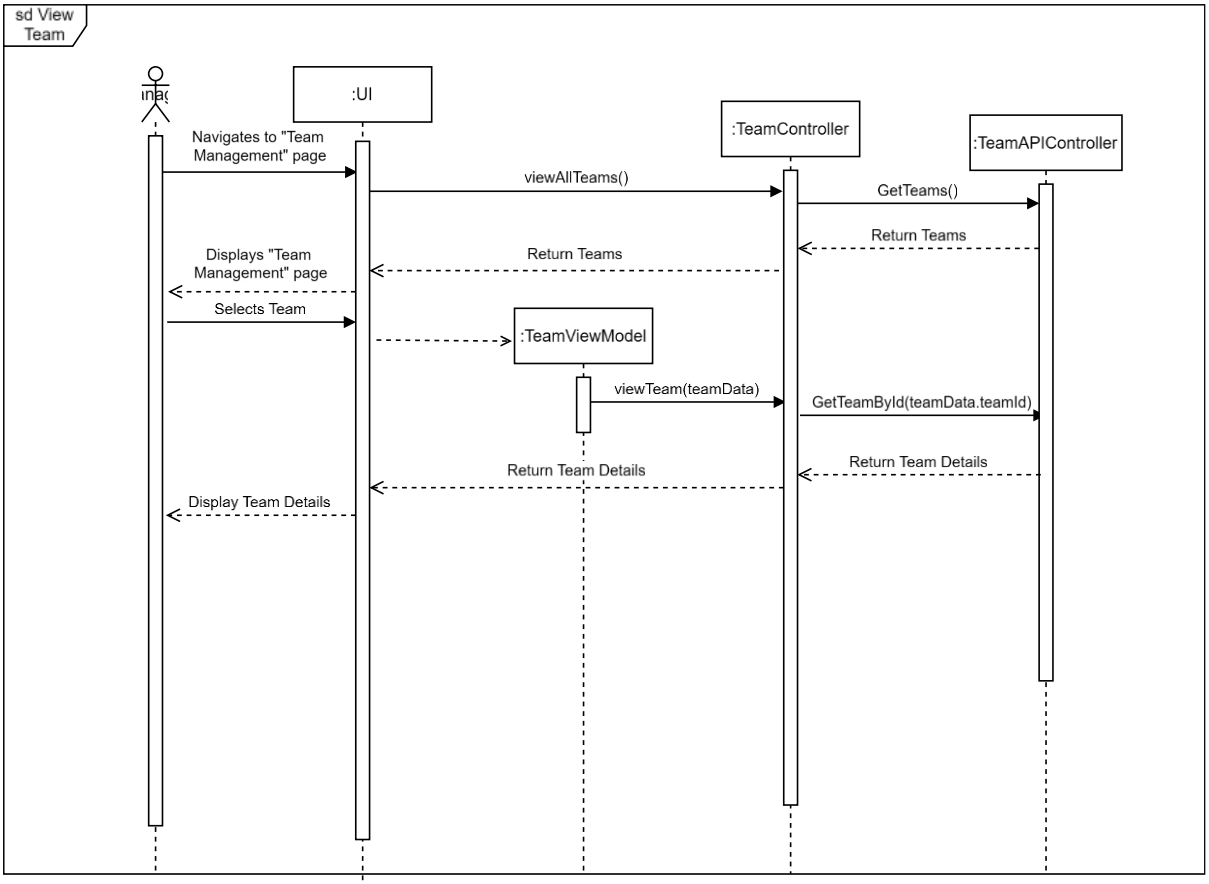
Use Case: Create Team



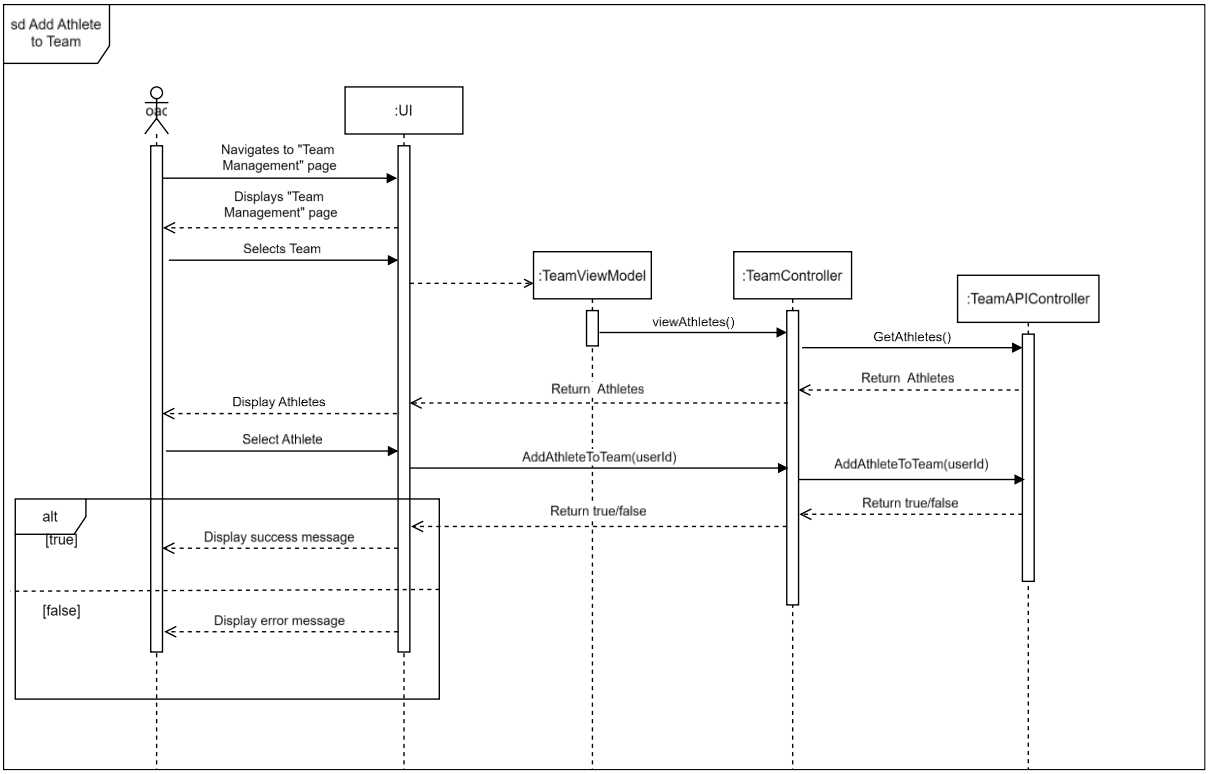
Use Case: Edit Team



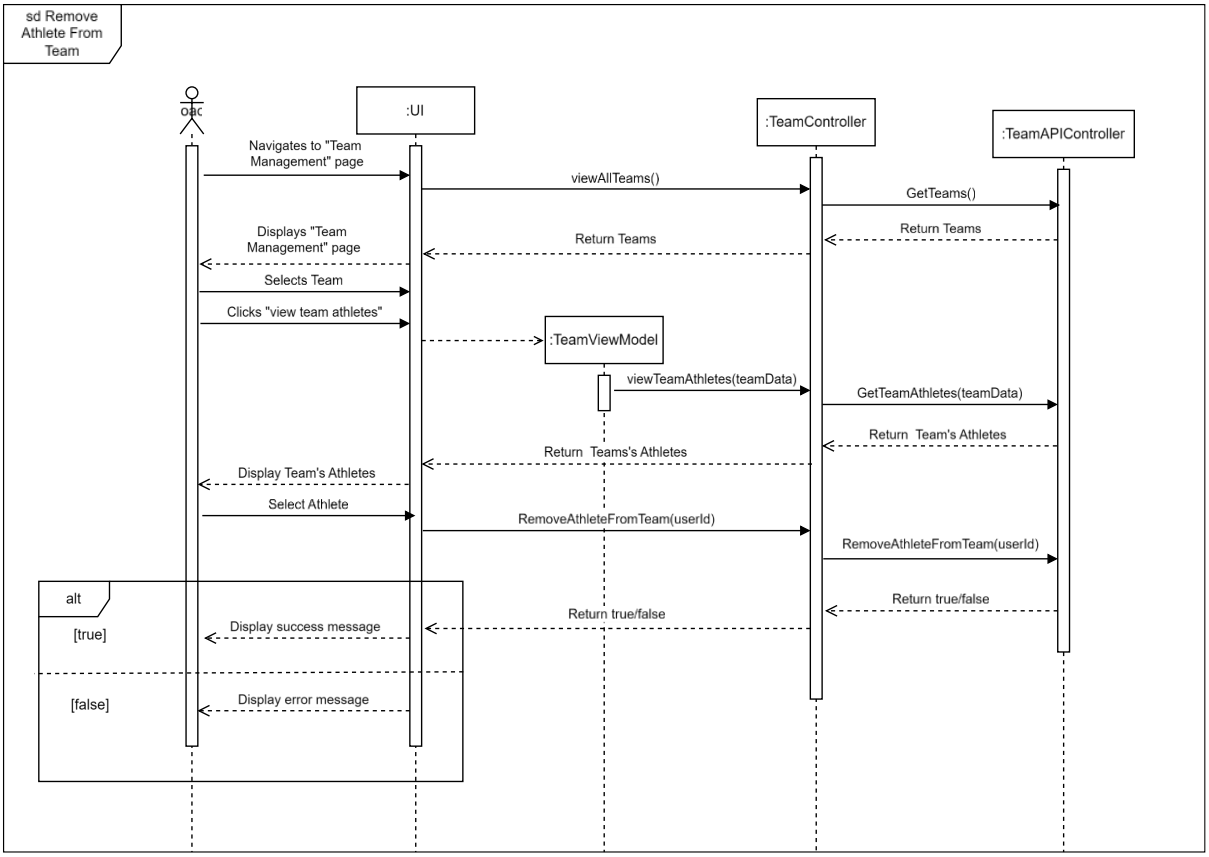
Use Case: View Team



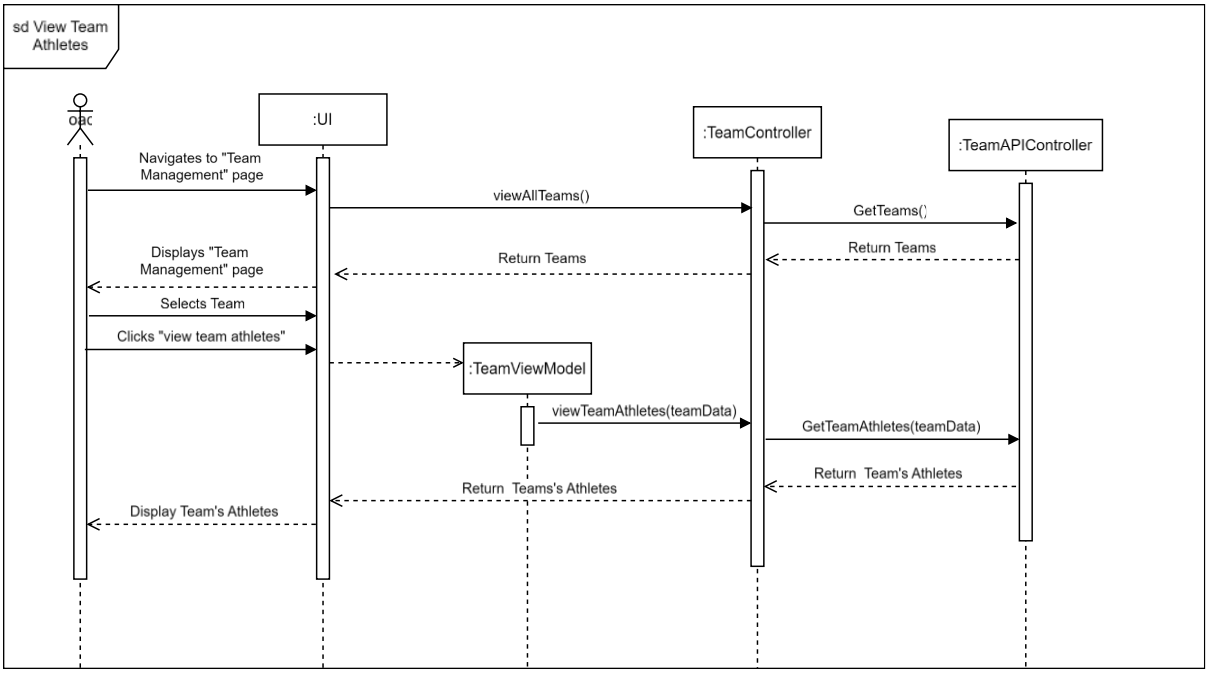
Use Case: Add Athlete to Team



Use Case: Remove Athlete from Team

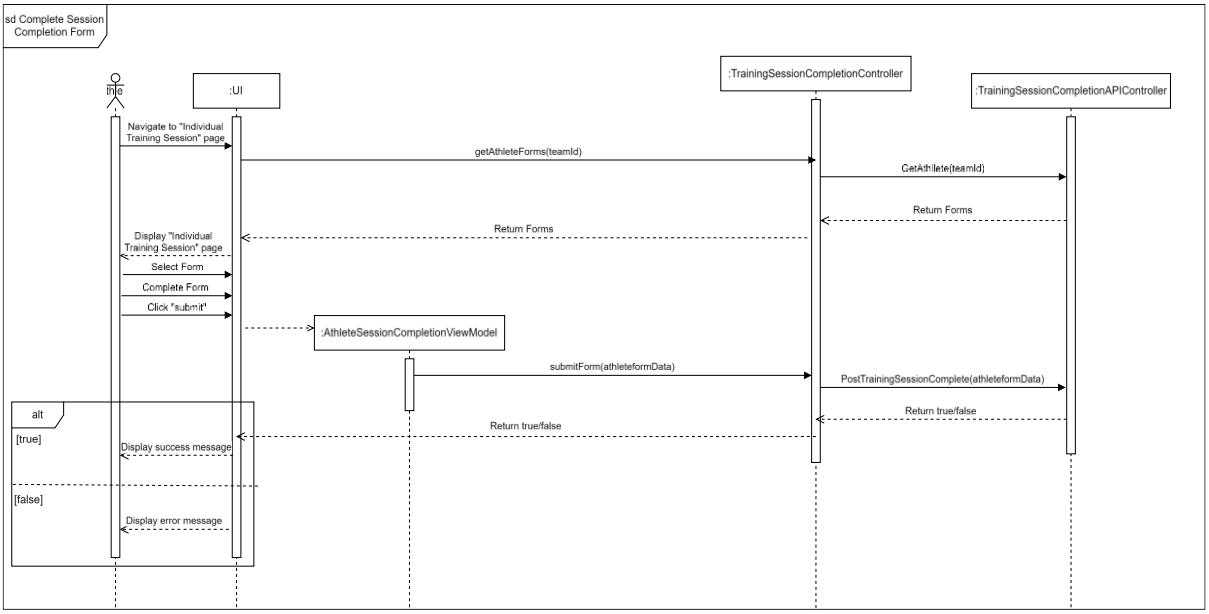


View Team Athletes

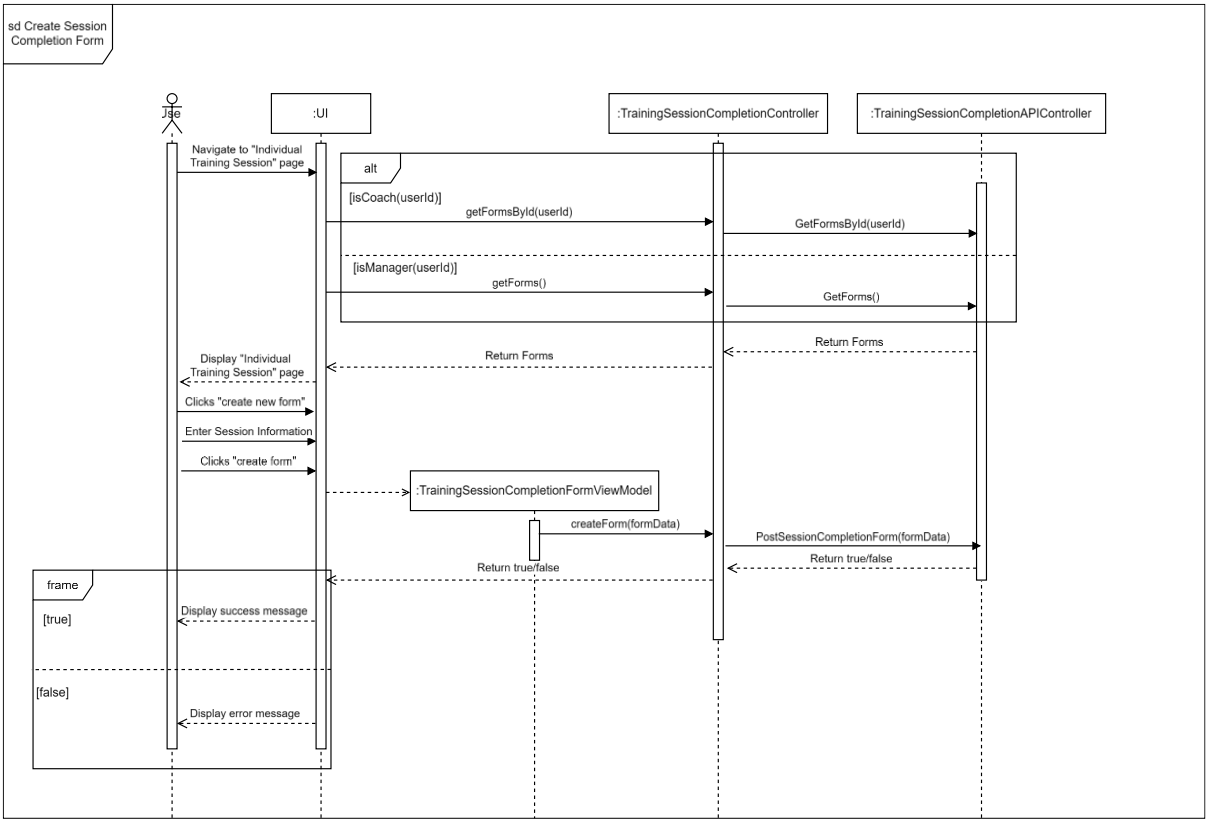


Individual Training Session Subsystem

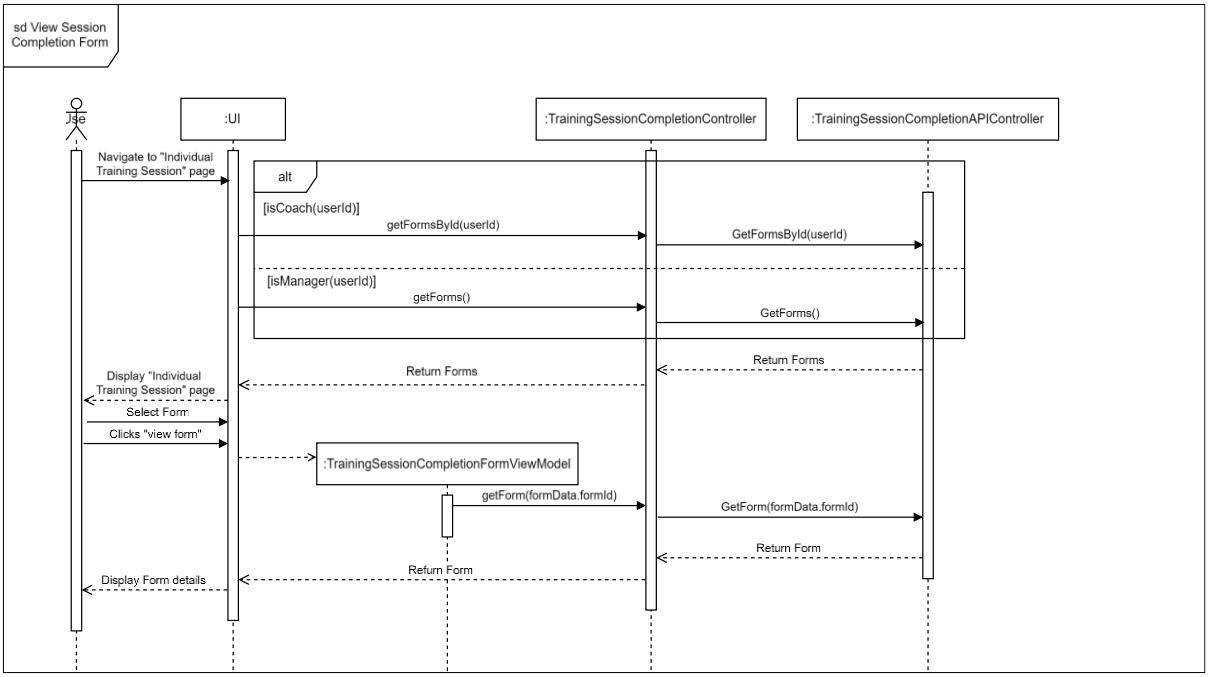
Use Case: Complete Session Completion Form



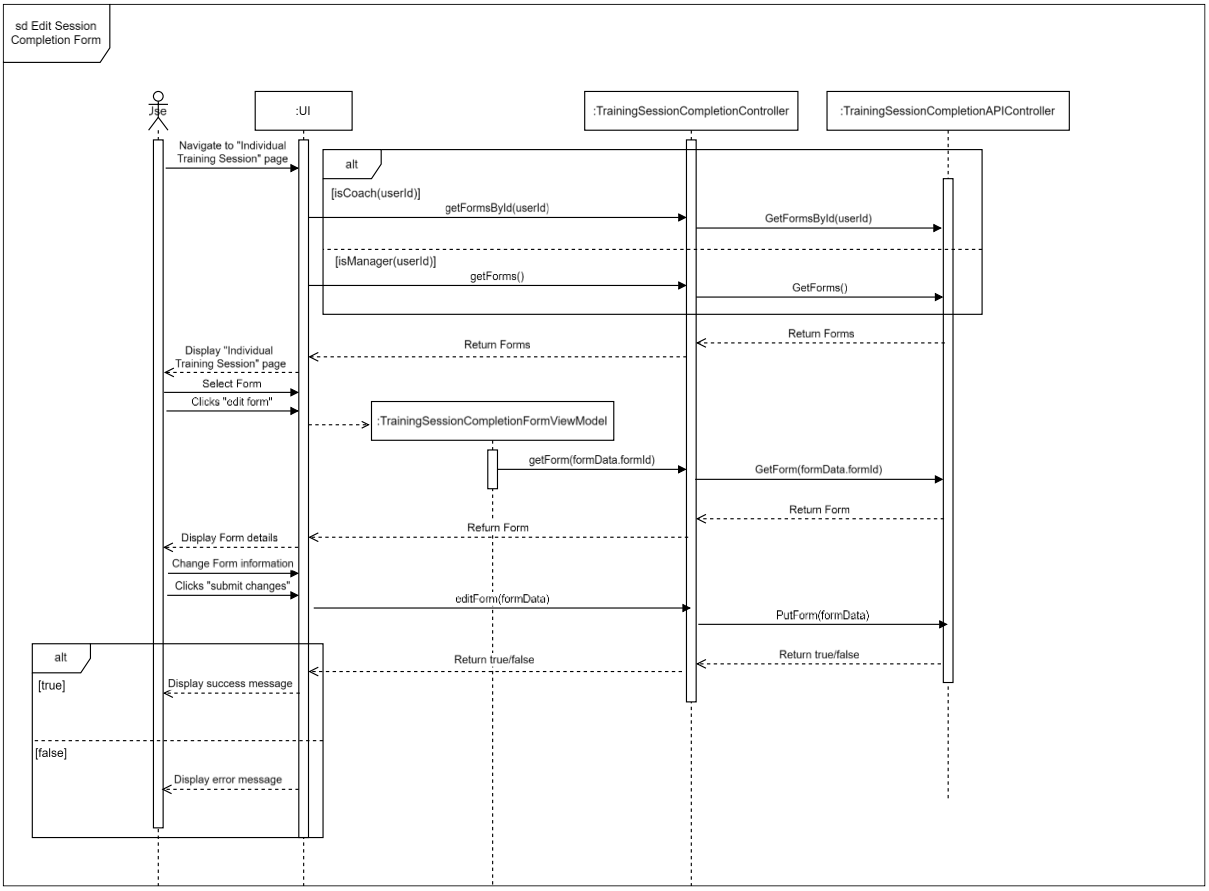
Use Case: Create Session Completion Form



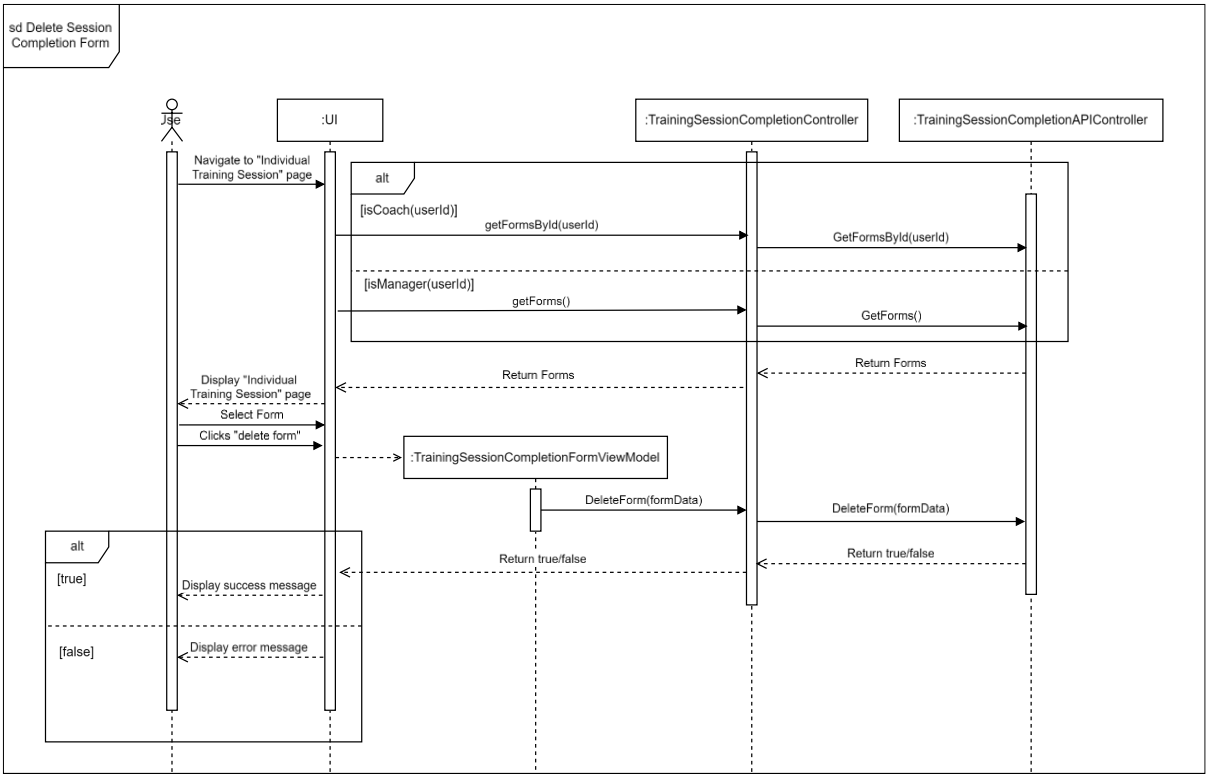
Use Case: View Session Completion Form



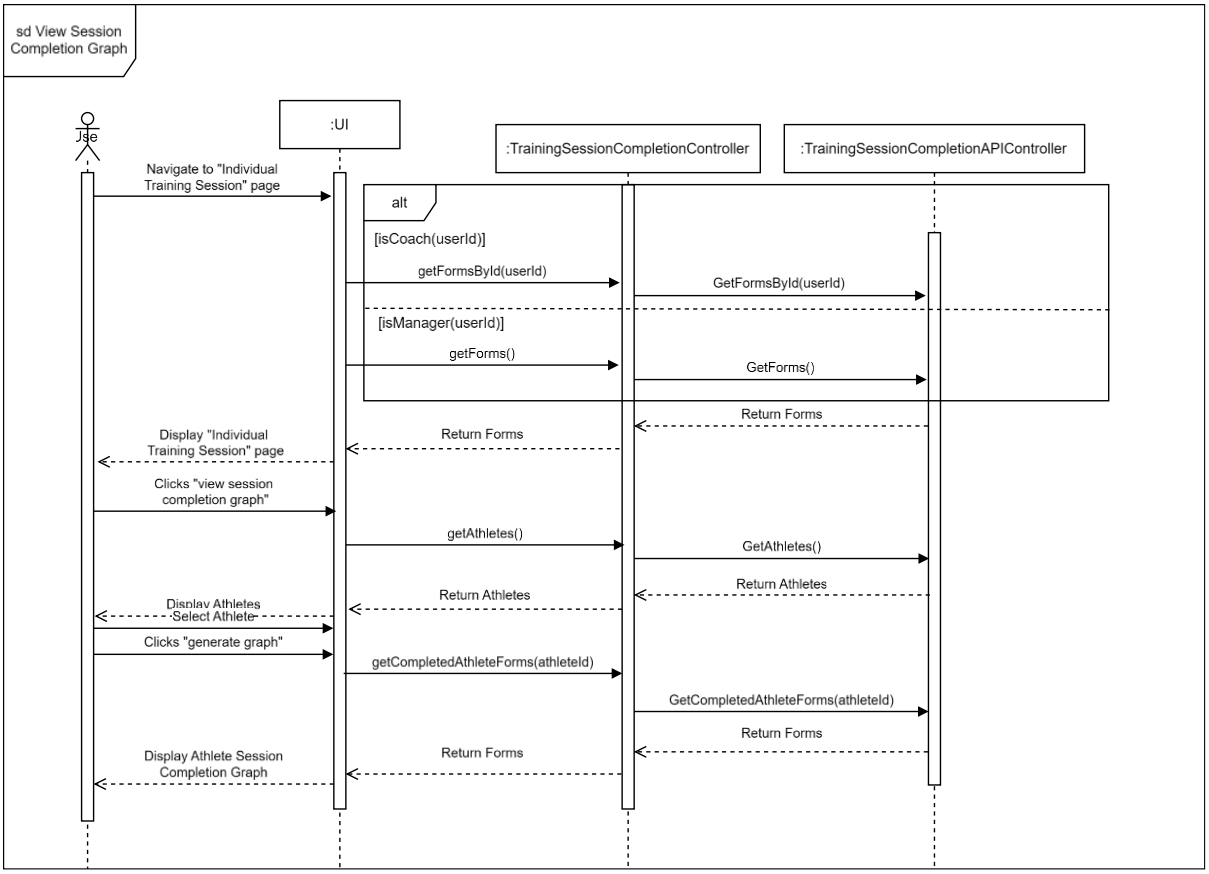
Use Case: Edit Session Completion Form



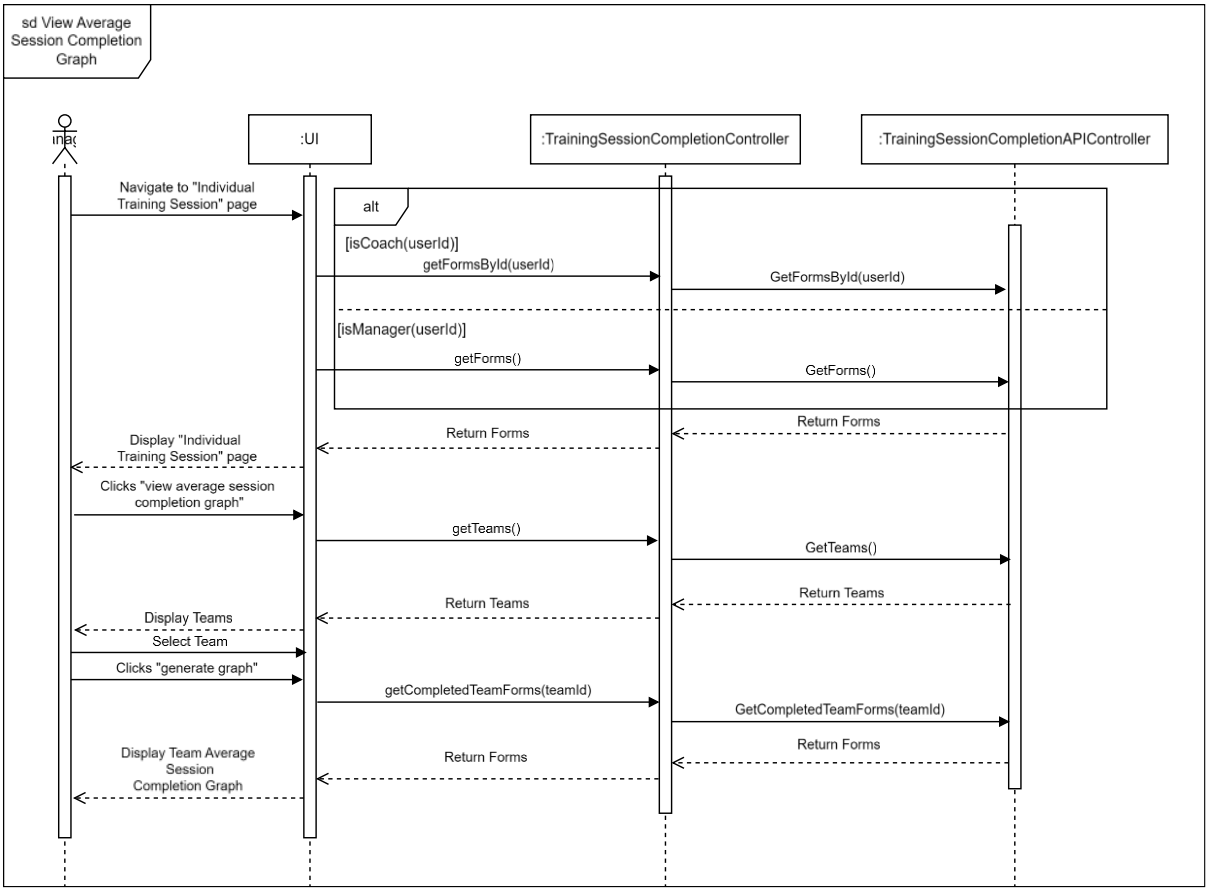
Use Case: Delete Session Completion Form



Use Case: View Athlete Session Completion Graph

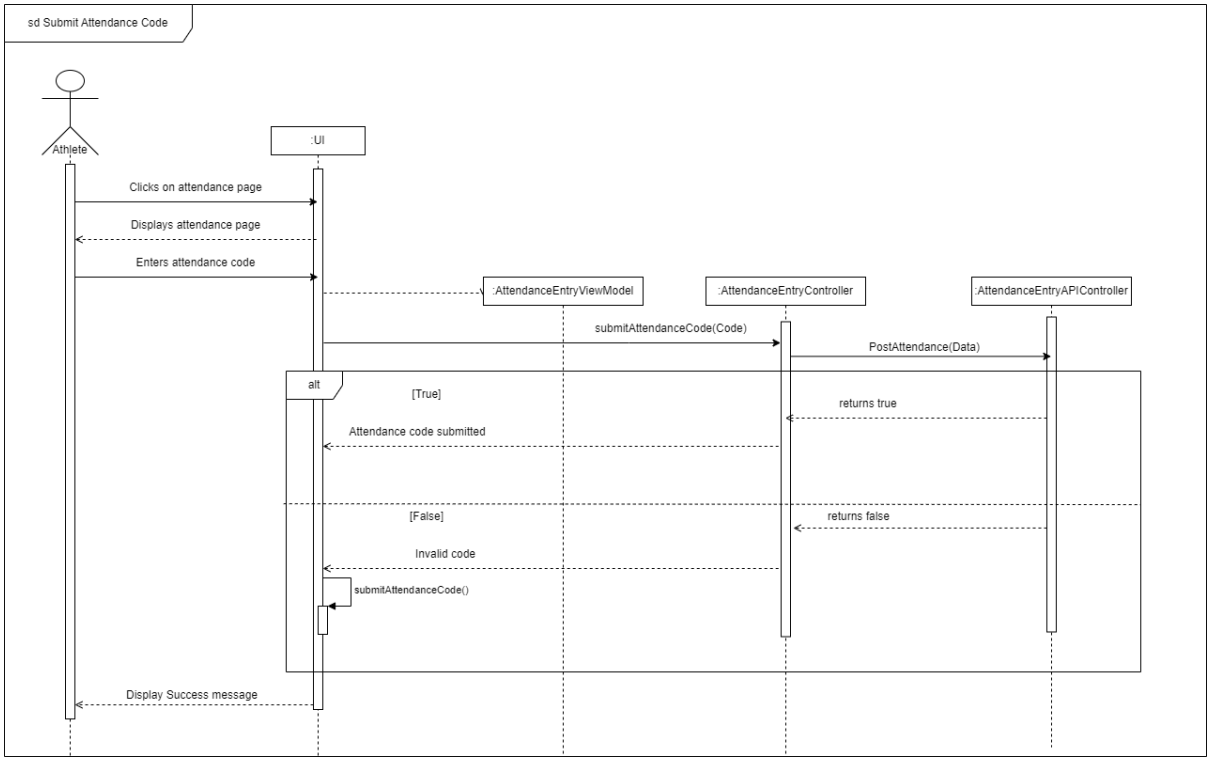


Use Case: View Average Session Completion Graph

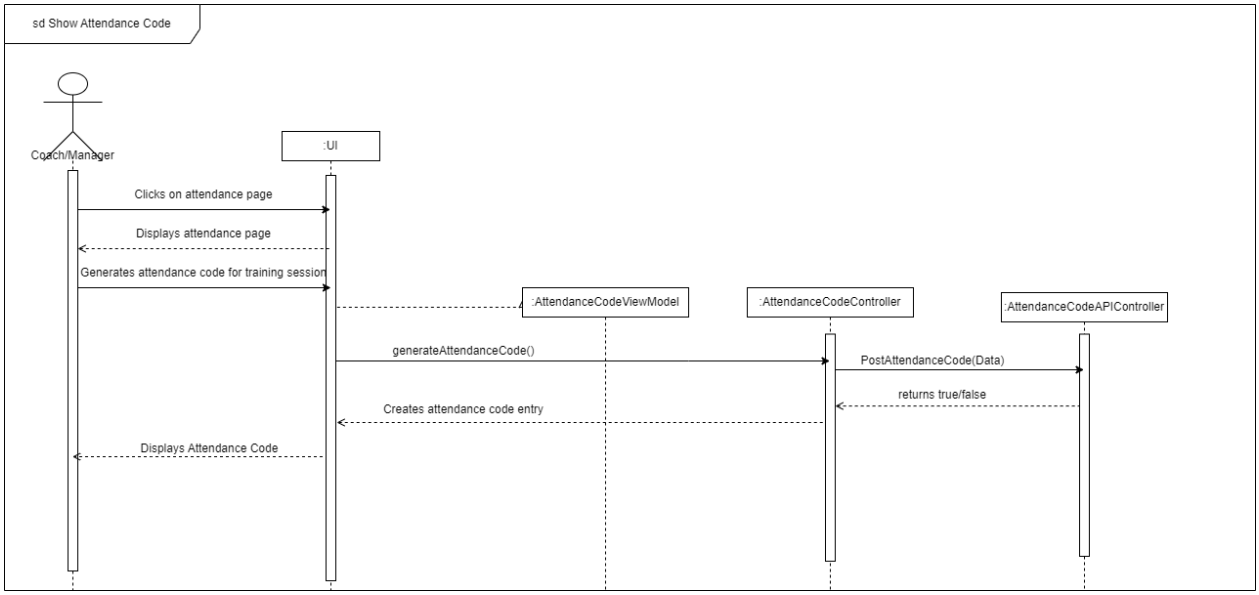


Attendance Management Subsystem

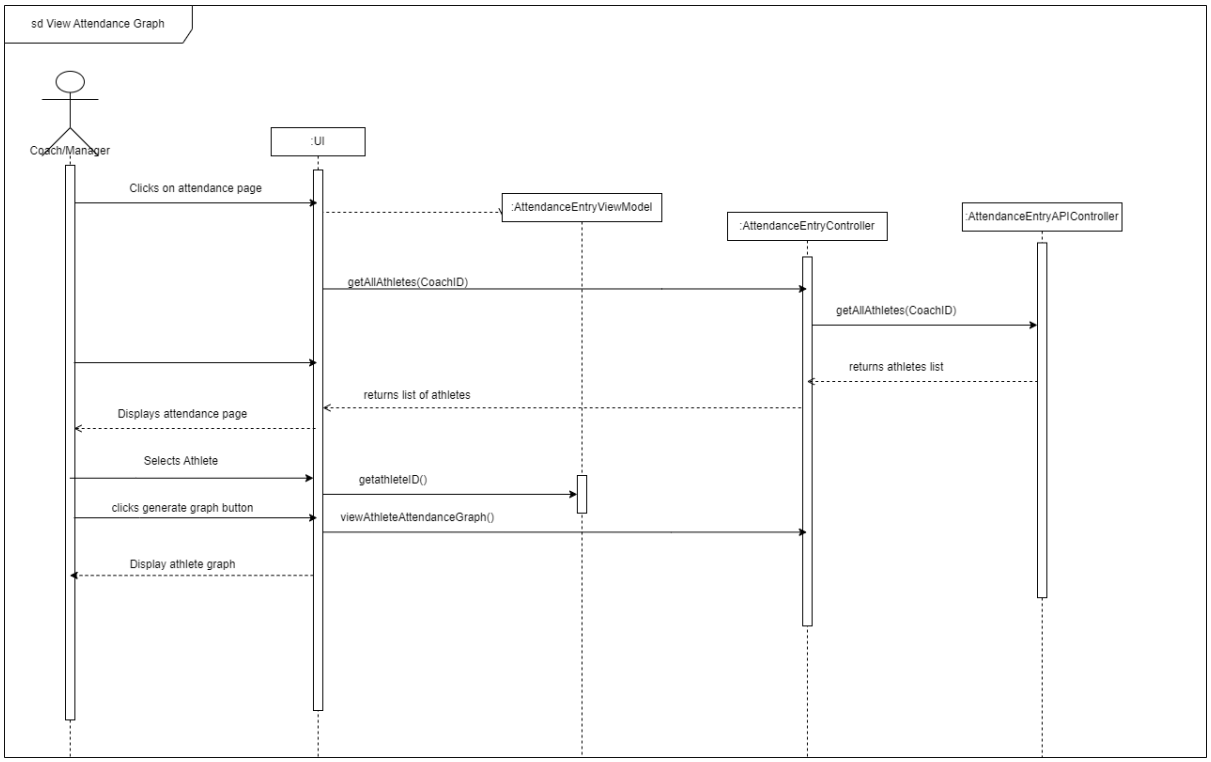
Use Case: Submit Attendance Code



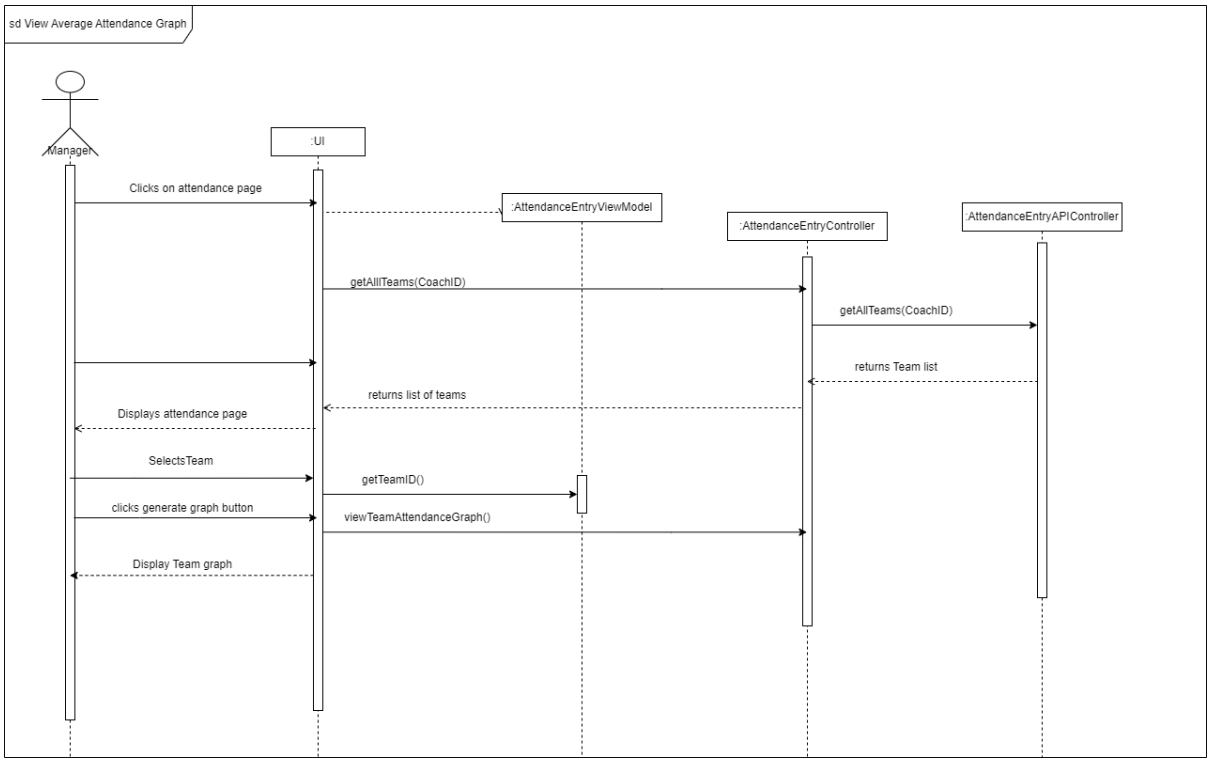
Use Case: Show Attendance Code



Use Case: View Athlete Attendance Graph

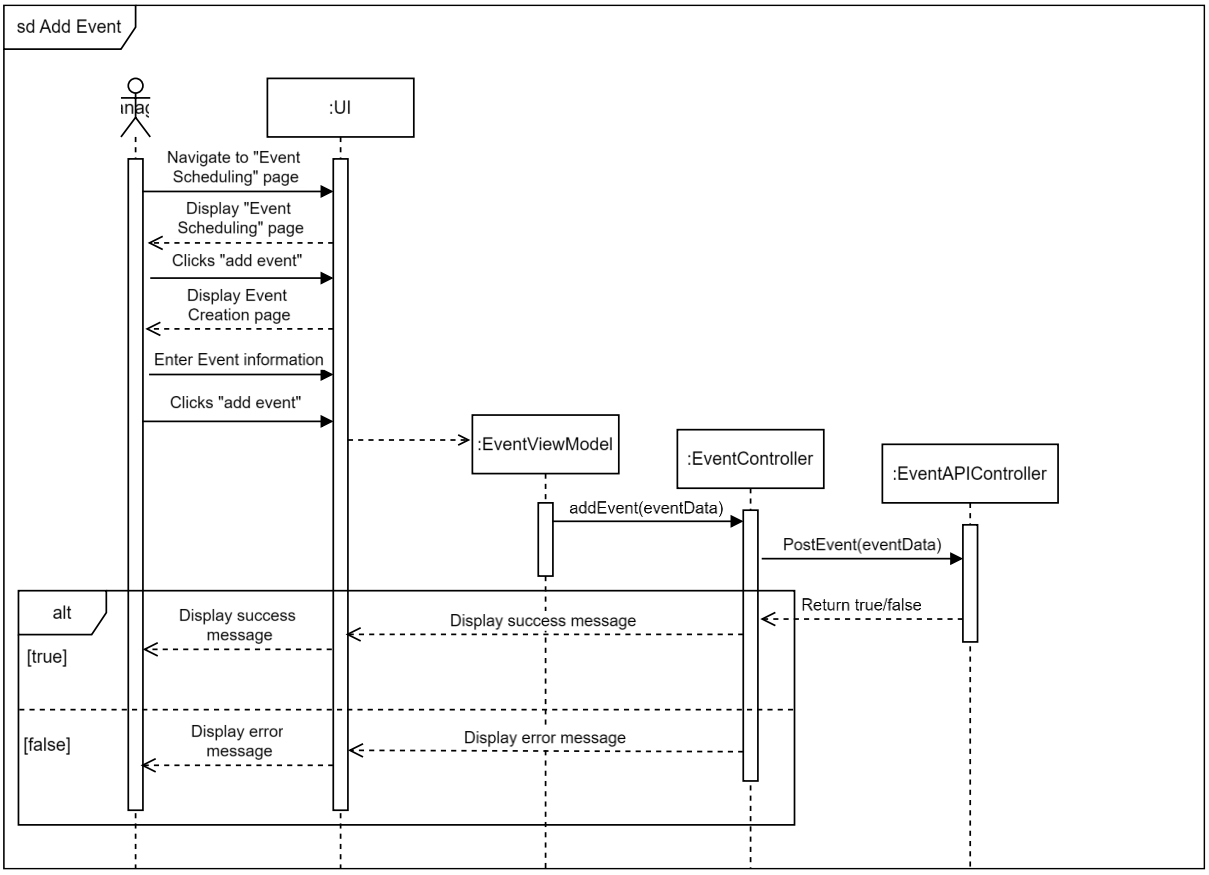


Use Case: View Average Attendance Graph

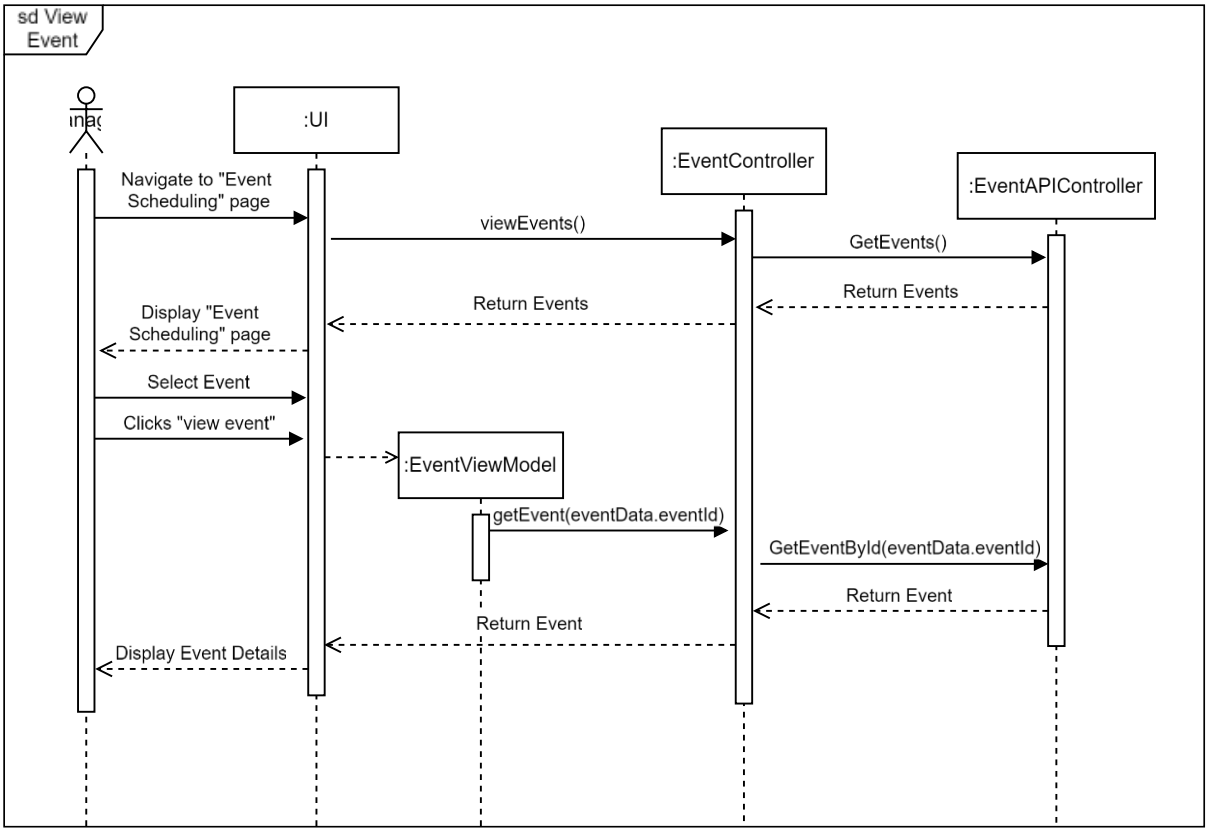


Event Scheduling Subsystem

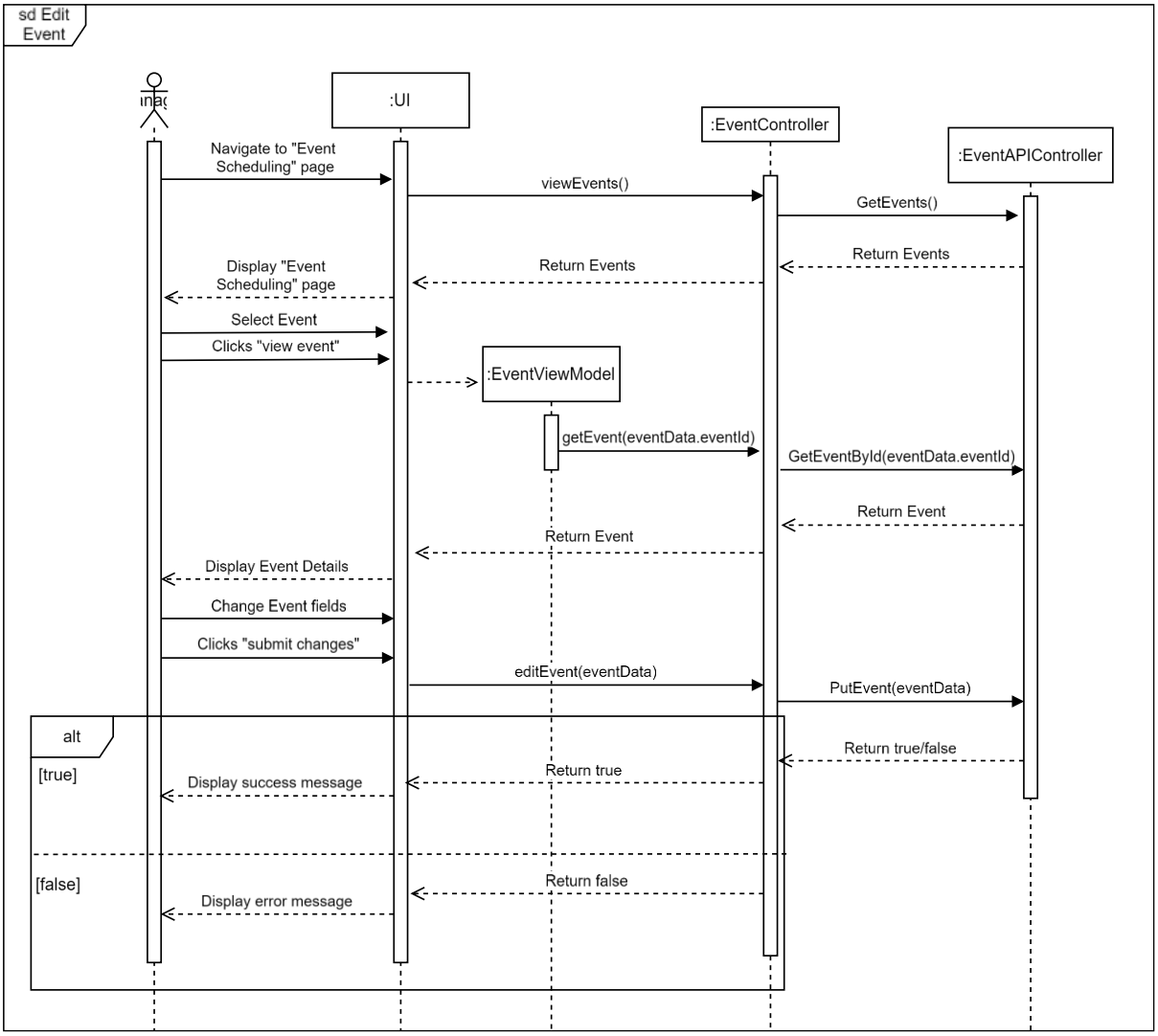
Use Case: Add Event



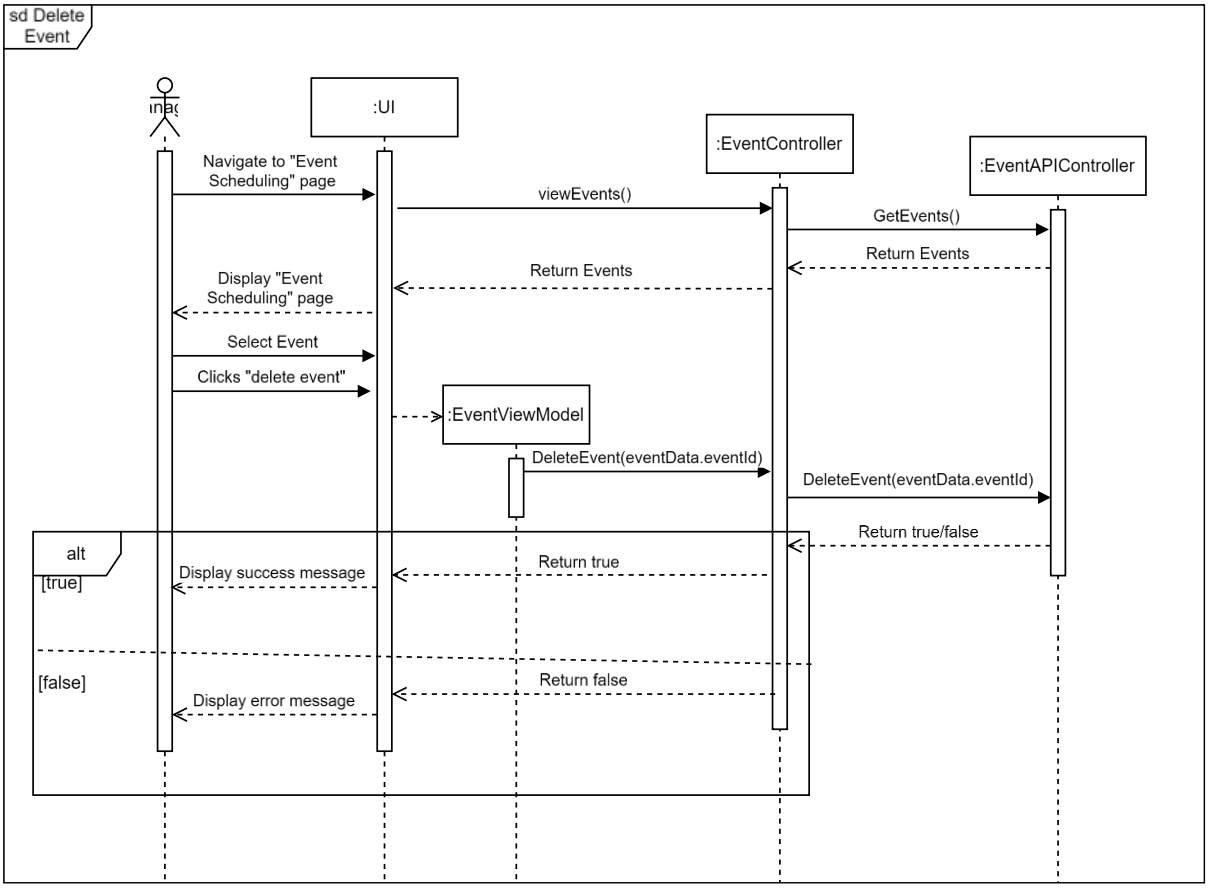
Use Case: View Event



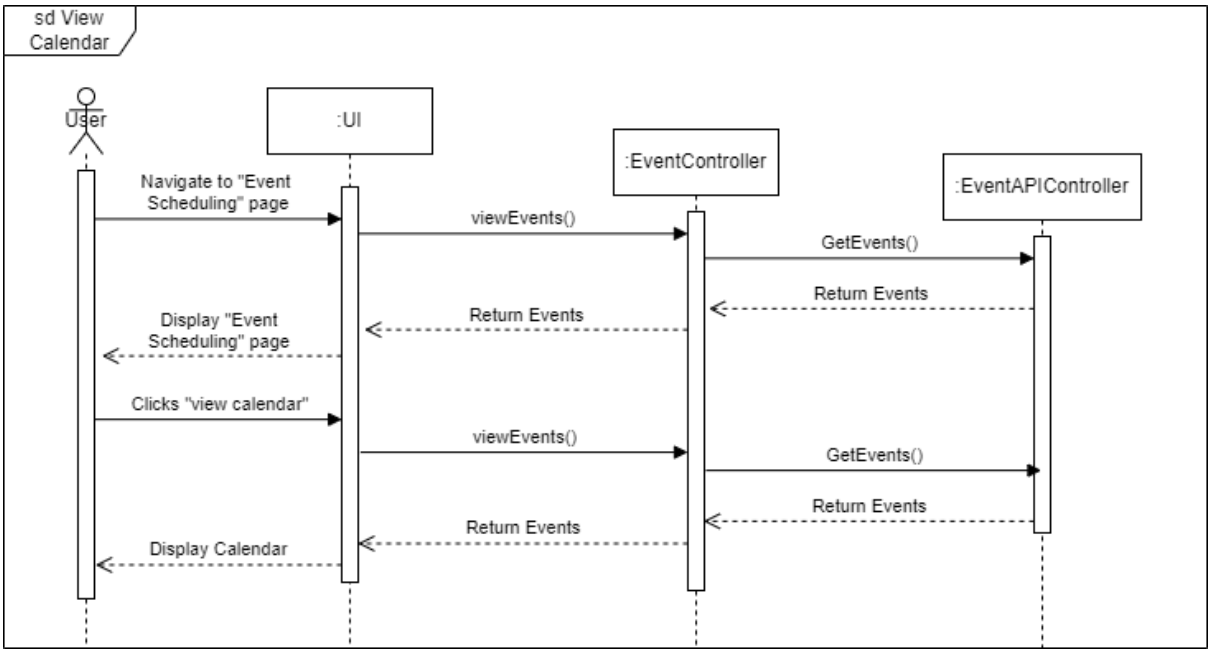
Use Case: Edit Event



Use Case: Delete Event

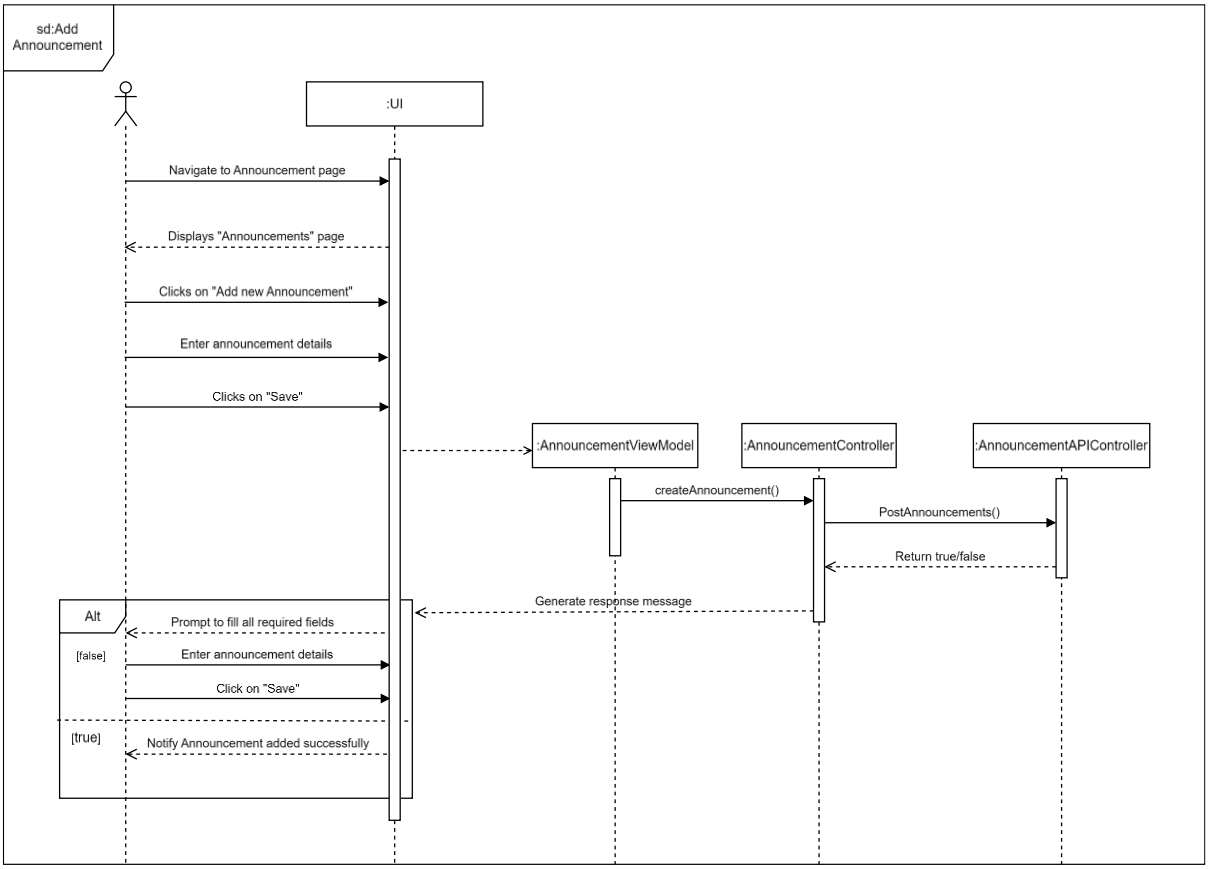


Use Case: View Calendar

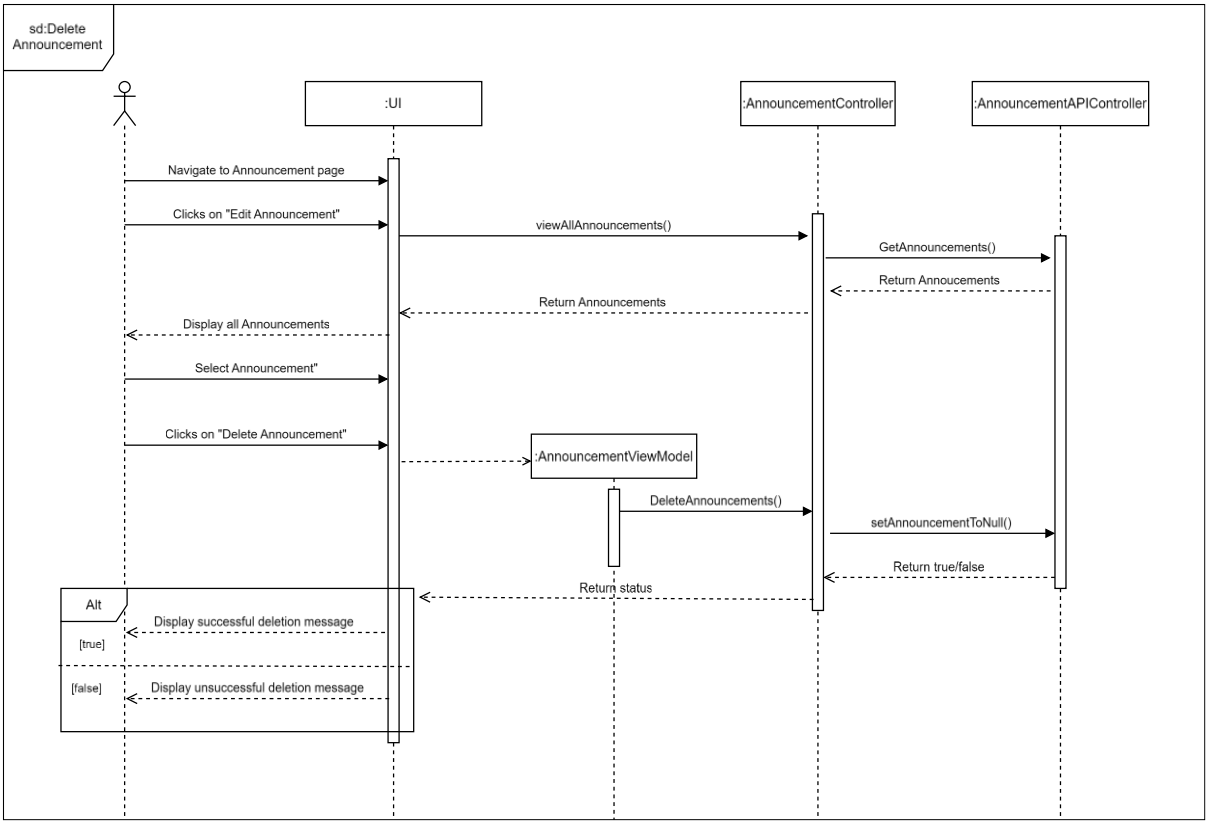


Announcement Subsystem

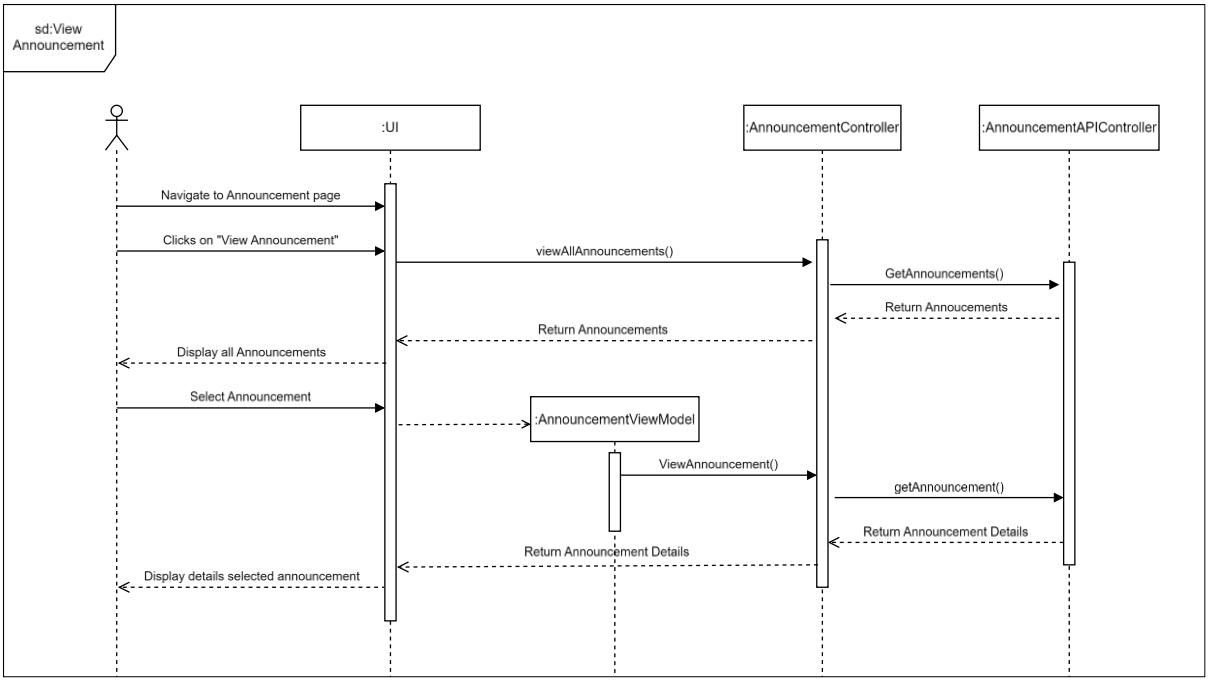
Use Case: Add Announcement



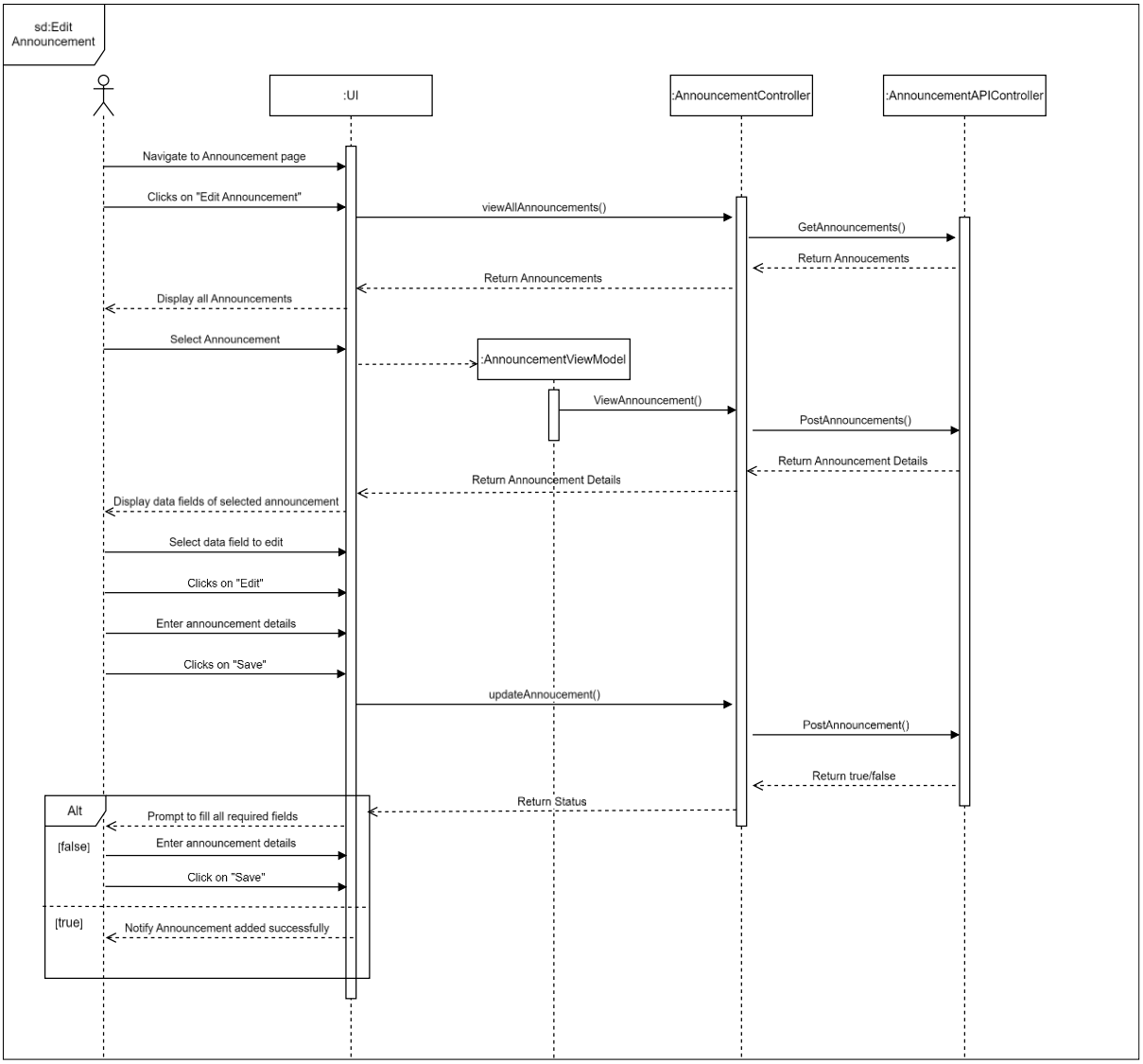
Use Case: Delete Announcement



Use Case: View Announcement

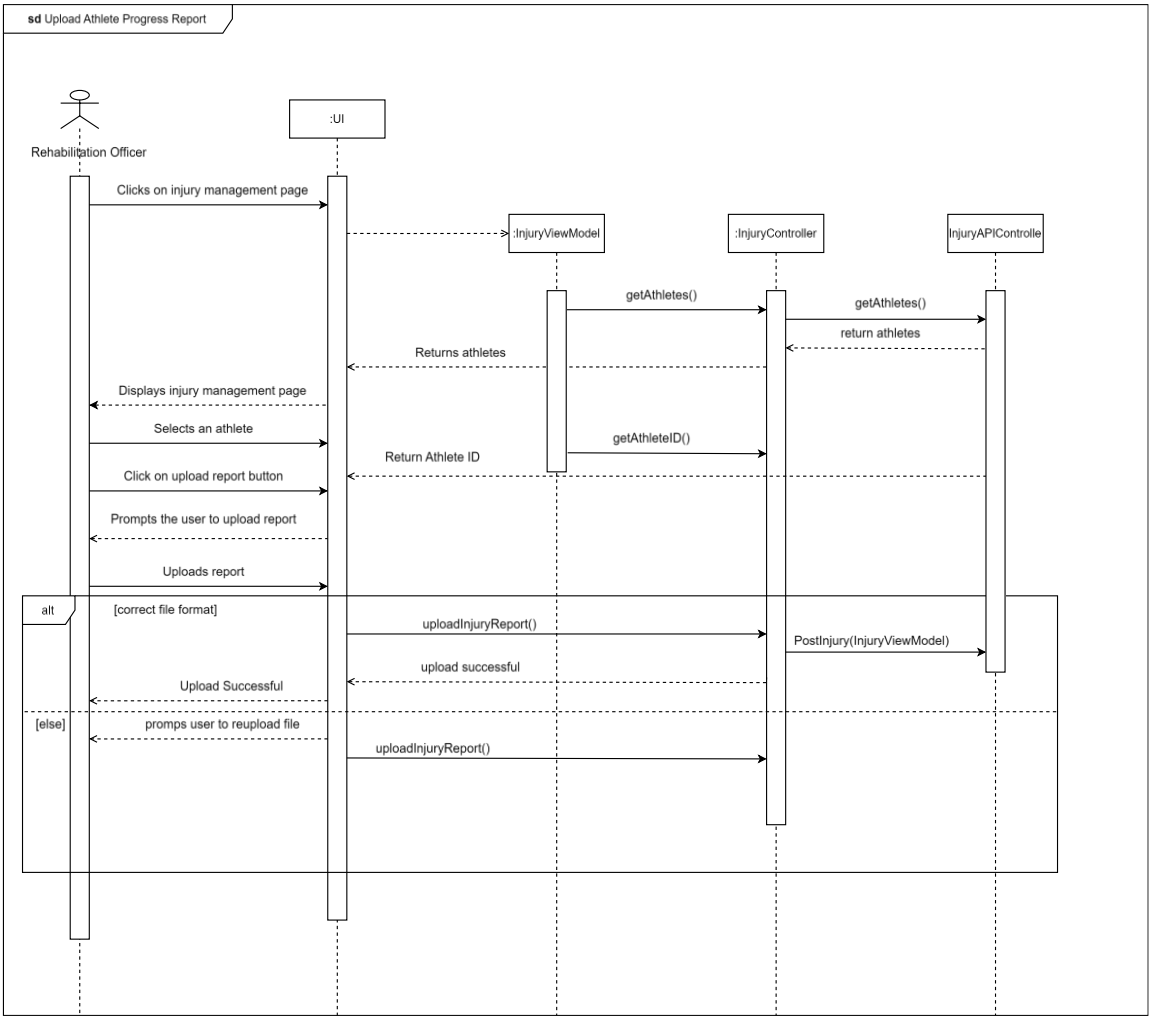


Use Case: Edit Announcement

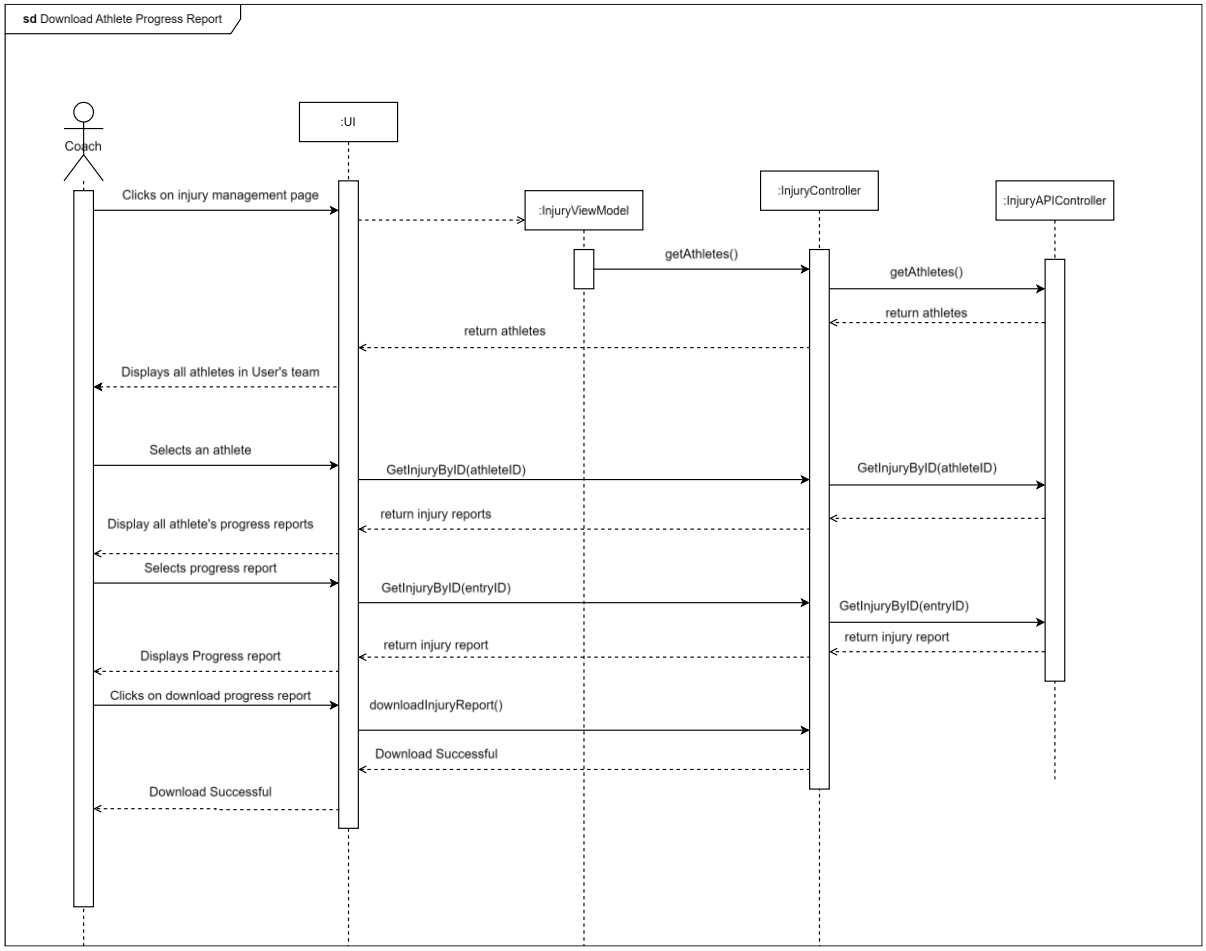


Injury Management Subsystem

Use Case: Upload Athlete Progress Report

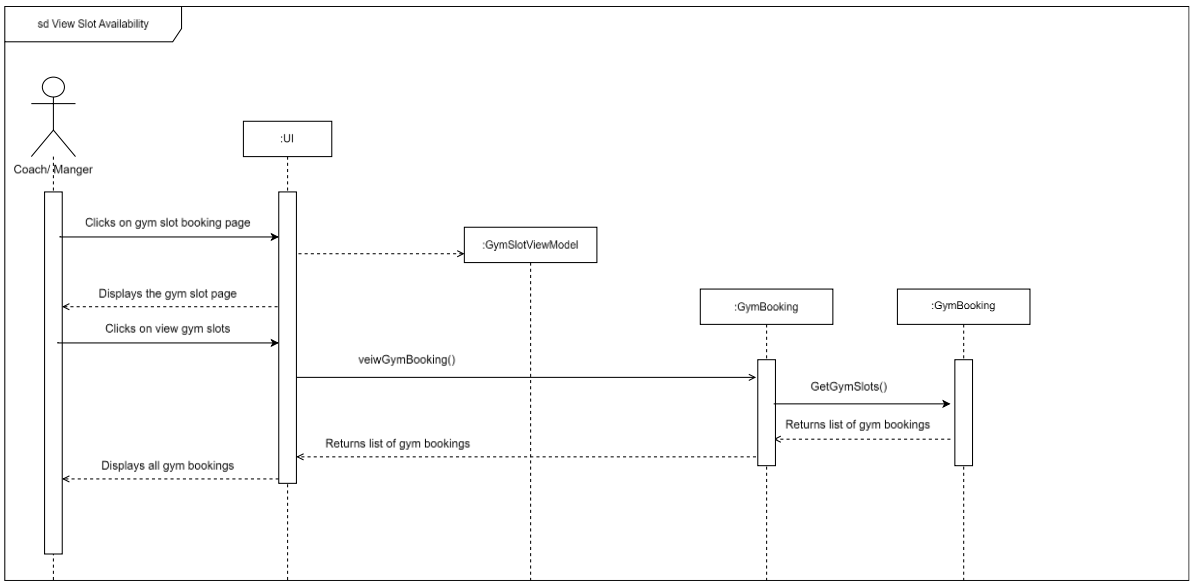


Use Case: Download Athlete Progress Report

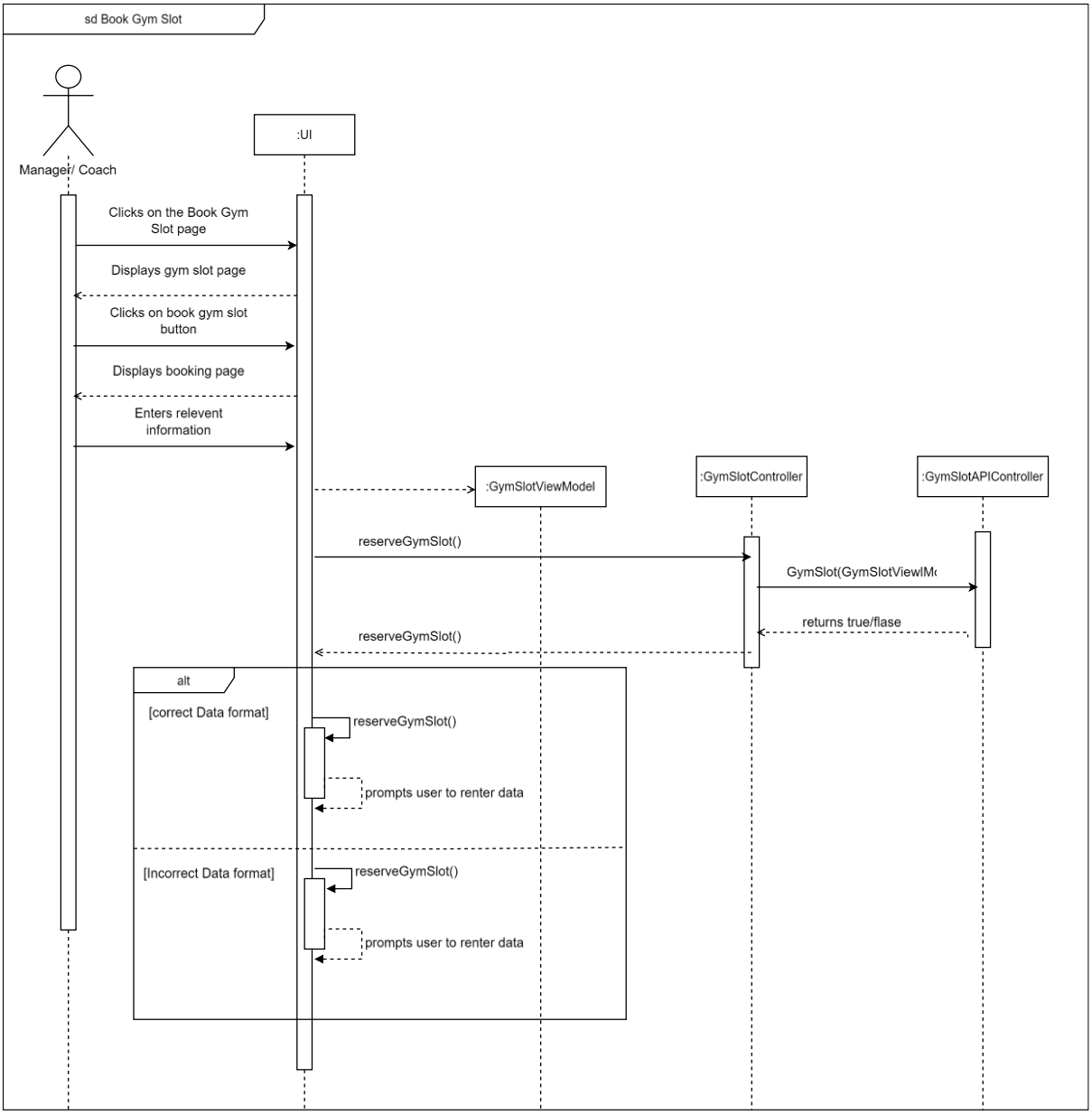


Gym Slot Booking Subsystem

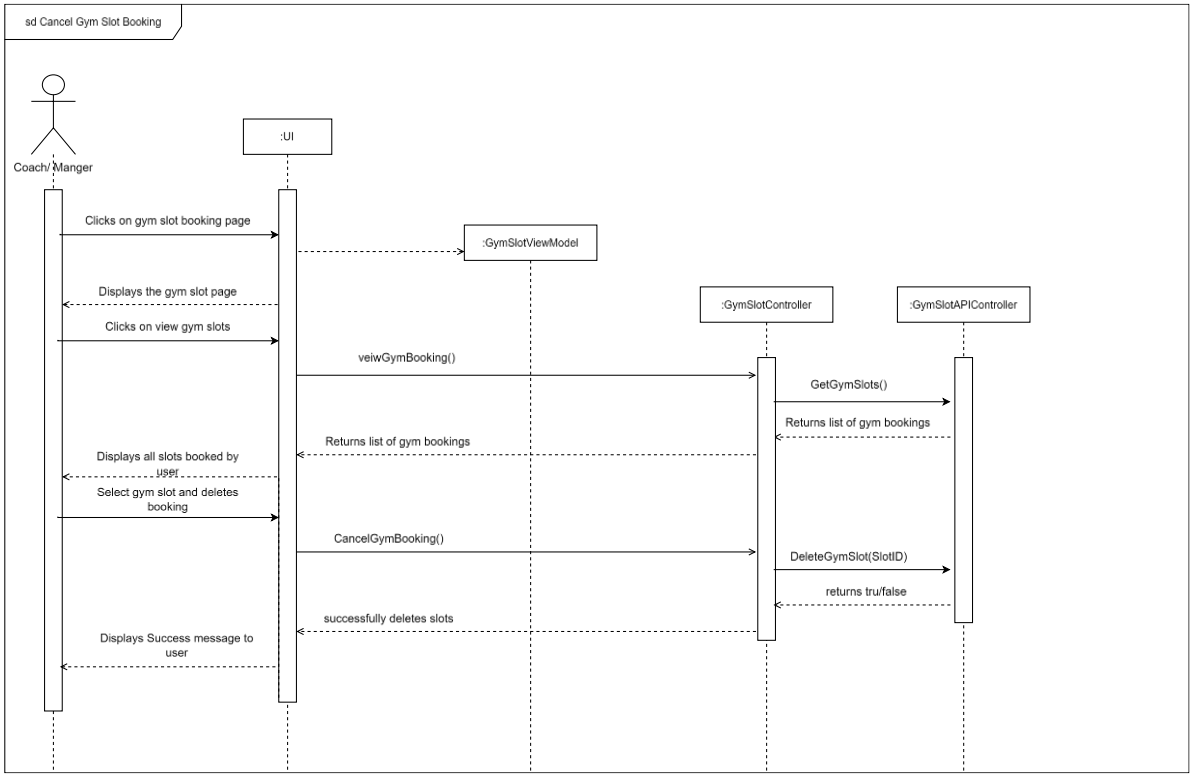
Use Case: View Gym Slot Availability



Use Case: Book Gym Slot

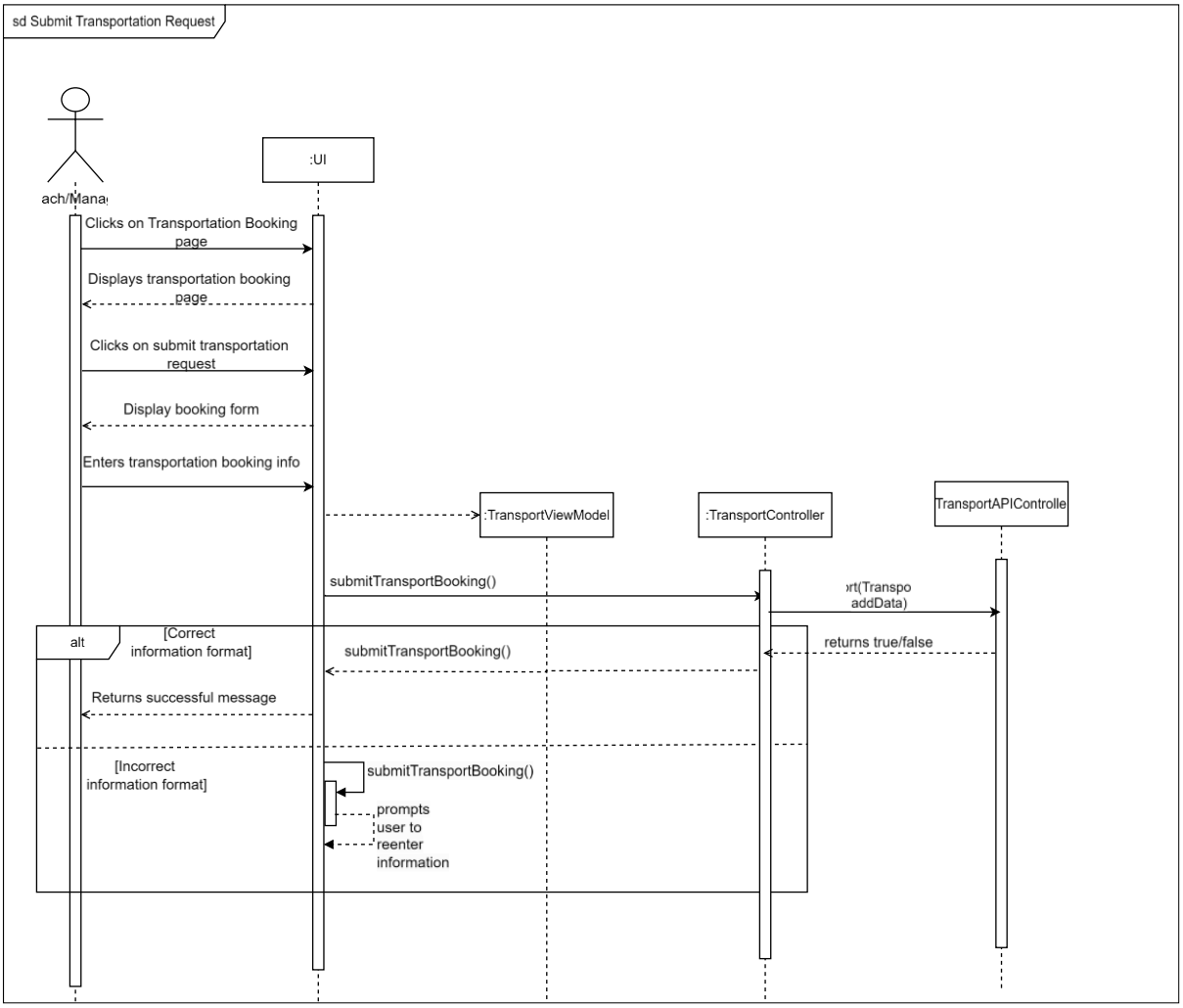


Use Case: Cancel Gym Slot Booking

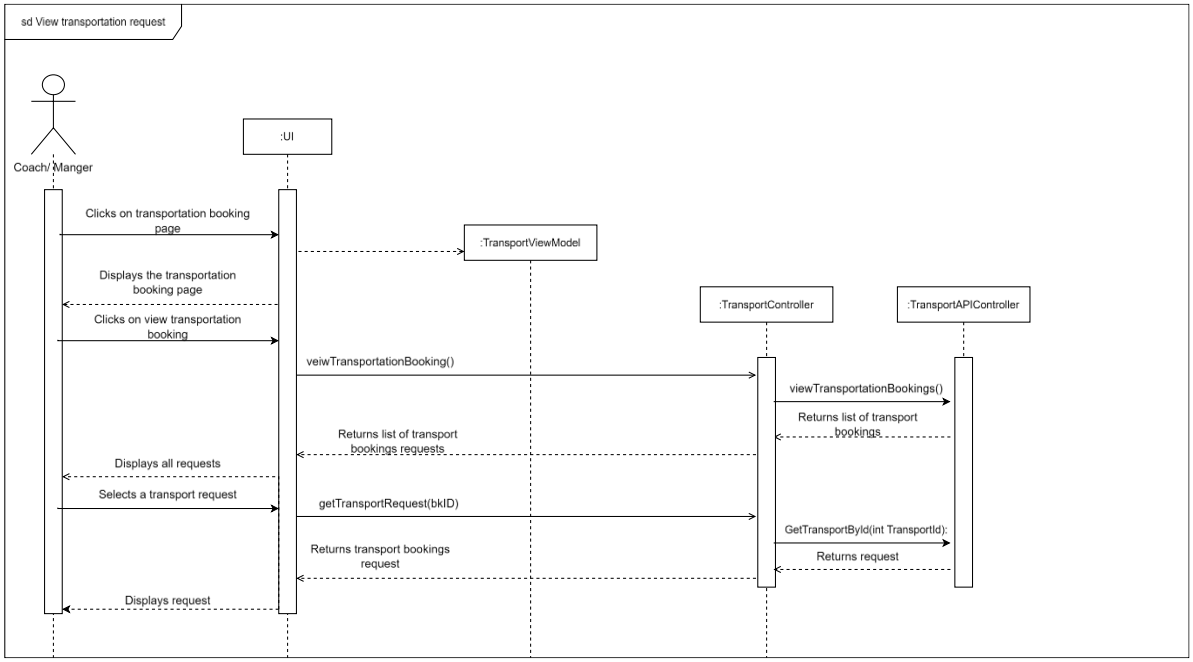


Transportation Booking Subsystem

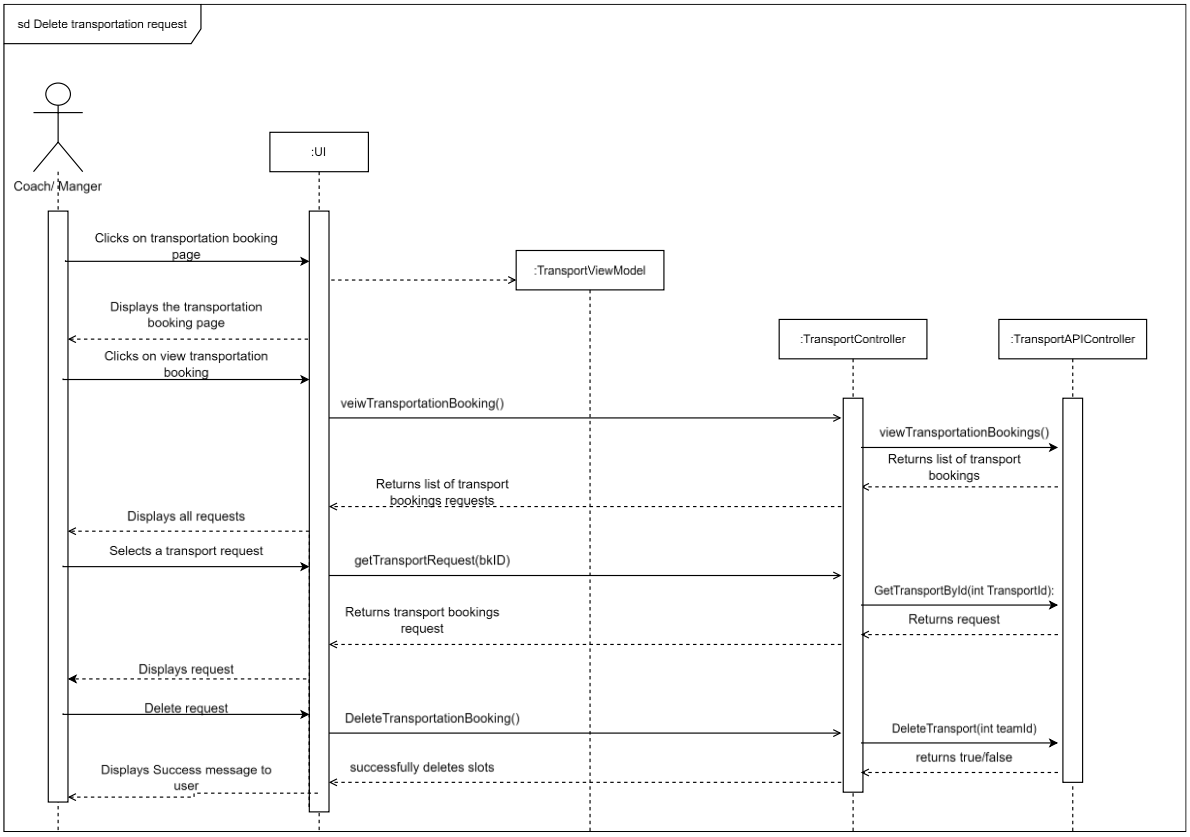
Use Case: Submit Transportation Request



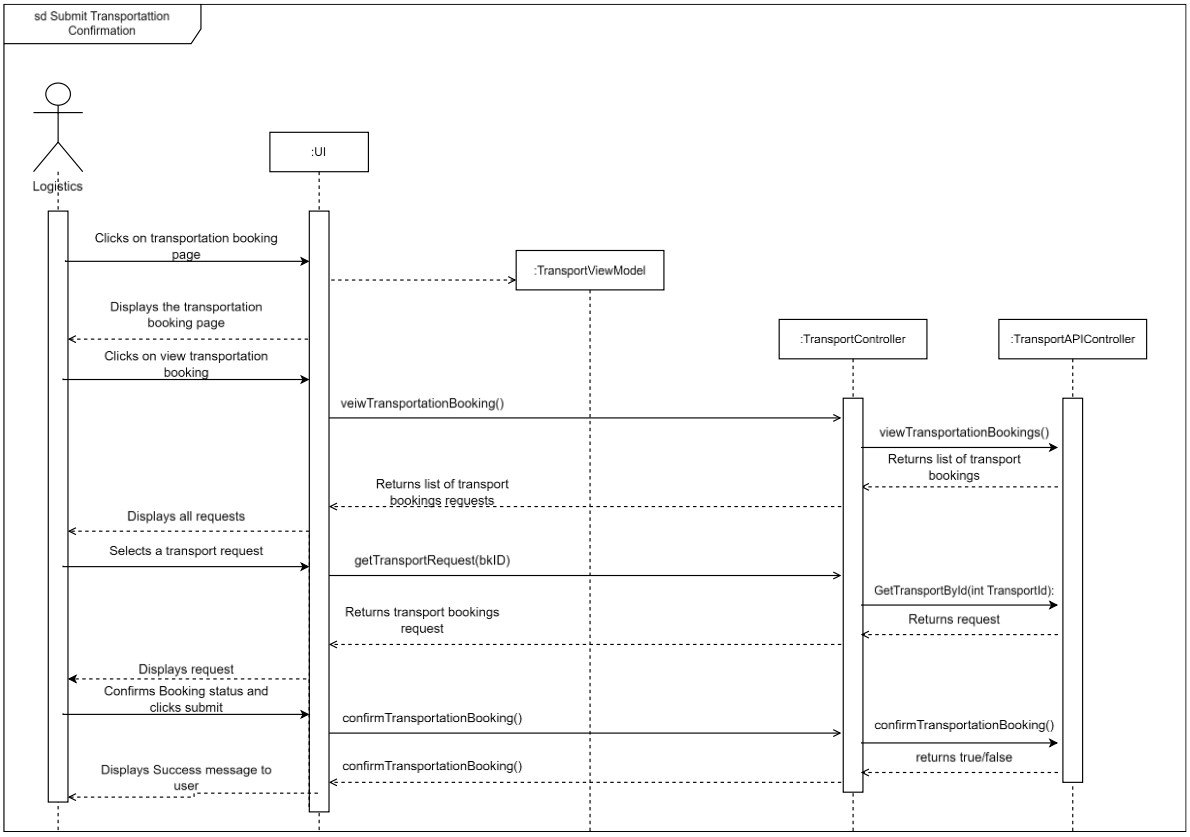
Use Case: View Transportation Request



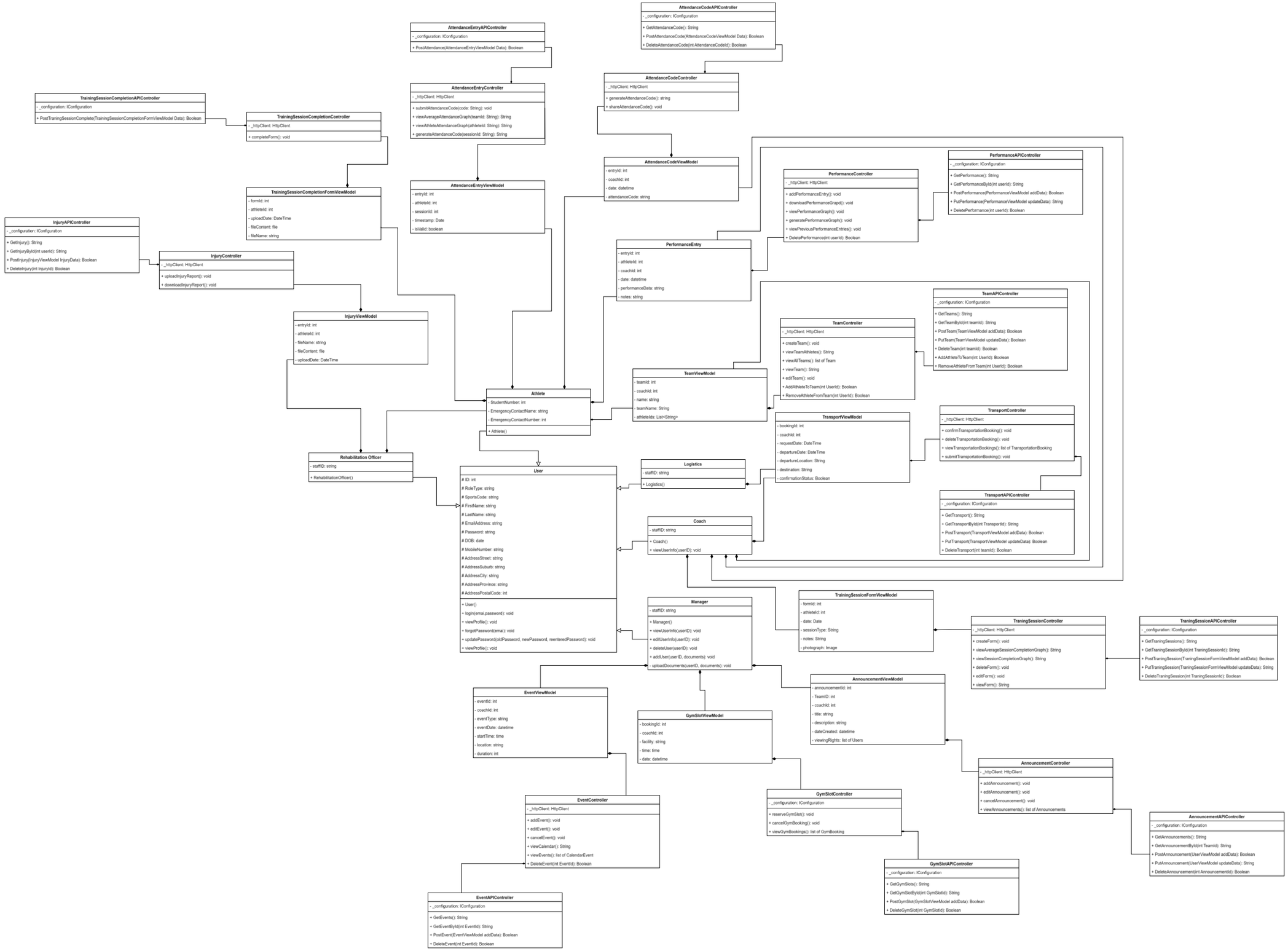
Use Case: Delete Transportation Request



Use Case: Submit Transportation Confirmation



Class Diagram


AttendanceCodeAPIController
- _configuration: IConfiguration
+ GetAttendanceCode(): String
+ PostAttendanceCode(AttendanceCodeViewModel Data): Boolean
+ DeleteAttendanceCode(int AttendanceCodeId): Boolean
AttendanceEntryAPIController
- _configuration: IConfiguration
+ PostAttendance(AttendanceEntryViewModel Data): Boolean
AttendanceCodeController
- _httpClient: HttpClient
+ generateAttendanceCode(): string
+ shareAttendanceCode(): void
AttendanceEntryController
- _httpClient: HttpClient
+ submitAttendanceCode(code: String): void
+ viewAverageAttendanceGraph(teamId: String): String
+ viewAthleteAttendanceGraph(athleteId: String): String
+ generateAttendanceCode(sessionId: String): String
TrainingSessionCompletionAPIController
- _configuration: IConfiguration
+ PostTraningSessionComplete(TrainingSessionCompletionFormViewModel Data): Boolean
TrainingSessionCompletionController
- _httpClient: HttpClient
+ completeForm(): void
AttendanceCodeViewModel
- entryId: int
- coachId: int
- date: datetime
- attendanceCode: string
AttendanceEntryViewModel
- entryId: int
- athleteId: int
- sessionId: int
- timestamp: Date
- isValid: boolean
TrainingSessionCompletionFormViewModel
- formId: int
- athleteId: int
- uploadDate: DateTime
- fileContent: file
- fileName: string
PerformanceController
- _httpClient: HttpClient
+ addPerformanceEntry(): void
+ downloadPerformanceGrapd(): void
+ viewPerformanceGraph(): void
+ generatePerformanceGraph(): void
+ viewPreviousPerformanceEntries(): void
+ DeletePerformance(int userId): Boolean
PerformanceAPIController
- _configuration: IConfiguration
+ GetPerformance(): String
+ GetPerformanceById(int userId): String
+ PostPerformance(PerformanceViewModel addData): Boolean
+ PutPerformance(PerformanceViewModel updateData): String
+ DeletePermpance(int userId): Boolean
InjuryAPIController
- _configuration: IConfiguration
+ GetInjury(): String
+ GetInjuryById(int userId): String
+ PostInjury(InjuryViewModel InjuryData): Boolean
+ DeleteInjury(int InjuryId): Boolean
InjuryController
- _httpClient: HttpClient
+ uploadInjuryReport(): void
+ downloadInjuryReport(): void
PerformanceEntry
- entryId: int
- athleteId: int
- coachId: int
- date: datetime
- performanceData: string
- notes: string
InjuryViewModel
- entryId: int
- athleteId: int
- fileName: string
- fileContent: file
- uploadDate: DateTime
TeamAPIController
- _configuration: IConfiguration
+ GetTeams(): String
+ GetTeamById(int teamId): String
+ PostTeam(TeamViewModel addData): Boolean
+ PutTeam(TeamViewModel updateData): Boolean
+ DeleteTeam(int teamId): Boolean
+ AddAthleteToTeam(int UserId): Boolean
+ RemoveAthleteFromTeam(int UserId): Boolean
TeamController
- _httpClient: HttpClient
+ createTeam(): void
+ viewTeamAthletes(): String
+ viewAllTeams(): list of Team
+ viewTeam(): String
+ editTeam(): void
+ AddAthleteToTeam(int UserId): Boolean
+ RemoveAthleteFromTeam(int UserId): Boolean
Athlete
- StudentNumber: int
- EmergencyContactName: string
- EmergencyContactNumber: int
+ Athlete()
TeamViewModel
- teamId: int
- coachId: int
- name: string
- teamName: String
- athleteIds: List<String>
TransportViewModel
- bookingId: int
- coachId: int
- requestDate: DateTime
- departureDate: DateTime
- departureLocation: String
- destination: String
- confirmationStatus: Boolean
TransportController
- _httpClient: HttpClient
+ confirmTransportationBooking(): void
+ deleteTransportationBooking(): void
+ viewTransportationBookings(): list of TransportationBooking
+ submitTransportationBooking(): void
TransportAPIController
- _configuration: IConfiguration
+ GetTransport(): String
+ GetTransportById(int TransportId): String
+ PostTransport(TransportViewModel addData): Boolean
+ PutTransport(TransportViewModel updateData): Boolean
+ DeleteTransport(int teamId): Boolean
Rehabilitation Officer
- staffID: string
+ RehabilitationOfficer()
User
ID: int
RoleType: string
SportsCode: string
FirstName: string
LastName: string
EmailAddress: string
Password: string
DOB: date
MobileNumber: string
AddressStreet: string
AddressSuburb: string
AddressCity: string
AddressProvince: string
AddressPostalCode: int
+ User()
+ login(email,password): void
+ viewProfile(): void
+ forgotPassword(email): void
+ updatePassword(oldPassword, newPassword, reenteredPassword): void
+ viewProfile(): void
Logistics
- staffID: string
+ Logistics()
Coach
- staffID: string
+ Coach()
+ viewUserInfo(userID): void
Manager
- staffID: string
+ Manager()
+ viewUserInfo(userID): void
+ editUserInfo(userID): void
+ deleteUser(userID): void
+ addUser(userID, documents): void
+ uploadDocuments(userID, documents): void
TrainingSessionFormViewModel
- formId: int
- athleteId: int
- date: Date
- sessionType: String
- notes: String
- photograph: Image
TraningSessionController
- _httpClient: HttpClient
+ createForm(): void
+ viewAverageSessionCompletionGraph(): String
+ viewSessionCompletionGraph(): String
+ deleteForm(): void
+ editForm(): void
+ viewForm(): String
TraningSessionAPIController
- _configuration: IConfiguration
+ GetTraningSessions(): String
+ GetTraningSessionById(int TraningSessionId): String
+ PostTraningSession(TraningSessionFormViewModel addData): Boolean
+ PutTraningSession(TraningSessionFormViewModel updateData): String
+ DeleteTraningSession(int TraningSessionId): Boolean
EventViewModel
- eventId: int
- coachId: int
- eventType: string
- eventDate: datetime
- startTime: time
- location: string
- duration: int
GymSlotViewModel
- bookingId: int
- coachId: int
- facility: string
- time: time
- date: datetime
AnnouncementViewModel
- announcementId: int
- TeamID: int
- coachId: int
- title: string
- description: string
- dateCreated: datetime
- viewingRights: list of Users
AnnouncementController
- _httpClient: HttpClient
+ addAnnouncement(): void
+ editAnnouncement(): void
+ cancelAnnouncement(): void
+ viewAnnouncements(): list of Announcements
AnnouncementAPIController
- _configuration: IConfiguration
+ GetAnnouncements(): String
+ GetAnnouncementById(int TeamId): String
+ PostAnnouncement(UserViewModel addData): Boolean
+ PutAnnouncement(UserViewModel updateData): String
+ DeleteAnnouncement(int AnnouncementId): Boolean
EventController
- _httpClient: HttpClient
+ addEvent(): void
+ editEvent(): void
+ cancelEvent(): void
+ viewCalendar(): String
+ viewEvents(): list of CalendarEvent
+ DeleteEvent(int EventId): Boolean
GymSlotController
- _configuration: IConfiguration
+ reserveGymSlot(): void
+ cancelGymBooking(): void
+ viewGymBookings(): list of GymBooking
GymSlotAPIController
- _configuration: IConfiguration
+ GetGymSlots(): String
+ GetGymSlotById(int GymSlotId): String
+ PostGymSlot(GymSlotViewModel addData): Boolean
+ DeleteGymSlot(int GymSlotId): Boolean
EventAPIController
- _configuration: IConfiguration
+ GetEvents(): String
+ GetEventById(int EventId): String
+ PostEvent(EventViewModel addData): Boolean
+ DeleteEvent(int EventId): Boolean