Smart Education Events System

SOEN 343 Section WW:
Software Architecture and Design

Phase 2:
System Architecture, Use Cases, and Sequence Diagram

Lab Section WL -X

Presented to:
Krupali Dobariya

Italian Stallions

Contributors:
(Team lead) Massimo Caruso - 40263285
Jammie Assenov - 40174965
Alessandro Tiseo - 40262416
Abdullah Taha - 40261146
Gabriel Rodriguez - 40208888
Vlad Tita - 40209853

Submission date: March 6, 2025

Table of contents

1. System Architecture

Within the phase one constraints, our design for SEES adopts a layered architecture that clearly separates responsibilities while leveraging the technologies already defined. We chose this approach because our Phase 1 deliverables—specifically the context diagram and domain model—directly map to the three layers: Presentation, Application, and Data. In our proposed solution, SEES is designed to handle comprehensive event management—from creating detailed event agendas and managing venue bookings to organizing in-person, online, and hybrid events. A role-based access system ensures that users have appropriate permissions, enabling organizers to plan and reserve venues and handle event-related payments while allowing attendees to sign up and receive event details.

By choosing a layered architecture, we can neatly map these functionalities to our system components:

- **Presentation Layer:**
  Built with React.js, this layer delivers a modular, interactive user interface for organizers, attendees, and administrative users. It handles user interactions and communicates with the backend via Socket.IO for real-time updates.
- **Application (Business Logic) Layer:**
  Implemented in Node.js, this layer encapsulates core functionalities such as event scheduling, user authentication, secure payment processing, and real-time engagement features. By centralizing the business logic here, we ensure that the system remains maintainable and that changes in functionality can be managed without impacting the user interface.
- **Data Layer:**
  Following our hybrid approach, we use PostgreSQL for relational data like user accounts, event details, and payment records, and NoSQL for flexible, real-time components such as chat interactions and networking features.

Justification & Trade-offs:

This layered approach aligns well with our phase one requirements by promoting separation of concerns, ease of maintenance, and scalability within a familiar technology stack. It avoids the additional complexity and overhead that a microservices architecture would introduce—complexities that are unnecessary for our current scope. While microservices might offer granular scalability and isolated deployments, the layered model fits our project's current needs by keeping development straightforward and within the defined constraints. This architecture also mirrors our context diagram and domain model, ensuring consistency across all design phases.

Alternative Comparison:

While we have chosen a **layered architecture** for SEES, a **microservices** approach is a viable alternative, particularly for larger-scale or more complex systems.

Trade-Offs of Microservices

1. **Increased Complexity**
    - Designing and maintaining multiple services requires more sophisticated DevOps practices.
    - Monitoring and logging become more challenging, as each service must be tracked individually.
2. **Communication Overhead**
    - Services often communicate via REST APIs, gRPC, or message brokers, adding latency and complexity to data flow.
    - Handling distributed transactions (e.g., a payment that triggers an event registration) can be more difficult than in a single codebase.
3. **Higher Operational Costs**
    - More services typically mean more servers, containers, or cloud resources to manage.
    - This can lead to increased costs for hosting, monitoring, and maintenance.

Strengths of Microservices

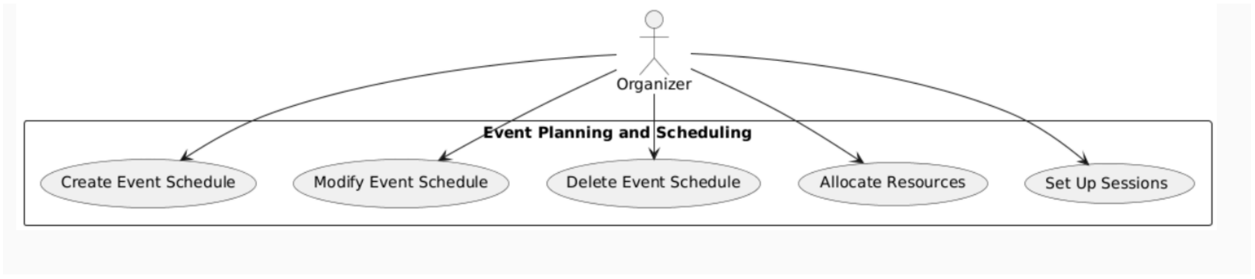
1. **Independent Deployability**
    - Each service (e.g., User Service, Event Service, Payment Service) can be developed, deployed, and scaled independently.
    - This isolation reduces the impact of failures; if one service goes down, others remain functional.
2. **Granular Scalability**
    - Services experiencing heavy load (like a real-time chat module) can be scaled independently of the rest of the system.
    - This is more resource-efficient than scaling an entire monolith or layered application.

A **layered architecture** supports the SEES platform's Phase One goals by providing a clear structure that aligns directly with the domain model. It balances simplicity, maintainability, and scalability, ensuring each domain entity is well-represented and easily managed. While microservices might be a future direction, this layered approach is the most practical solution for now, given the current scope and constraints.

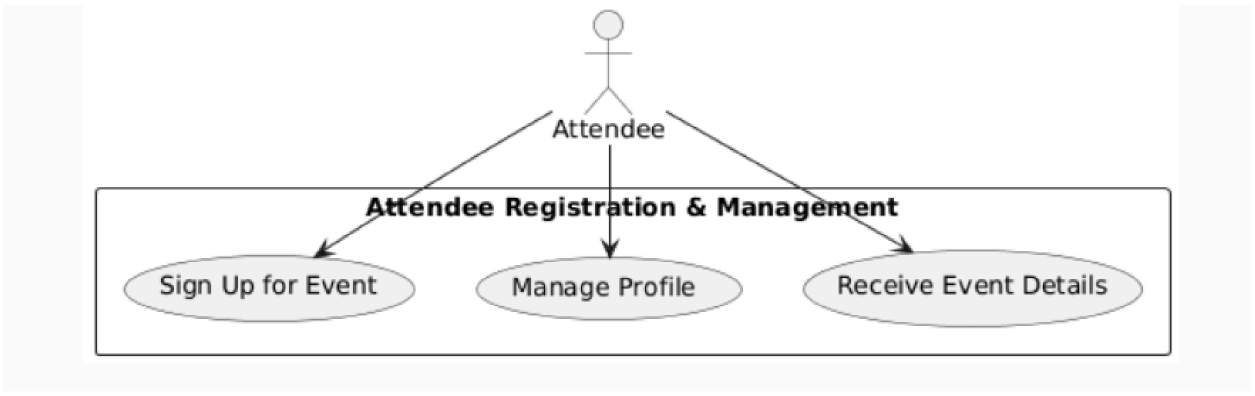
2. Use Cases

2.1 Use case Diagrams

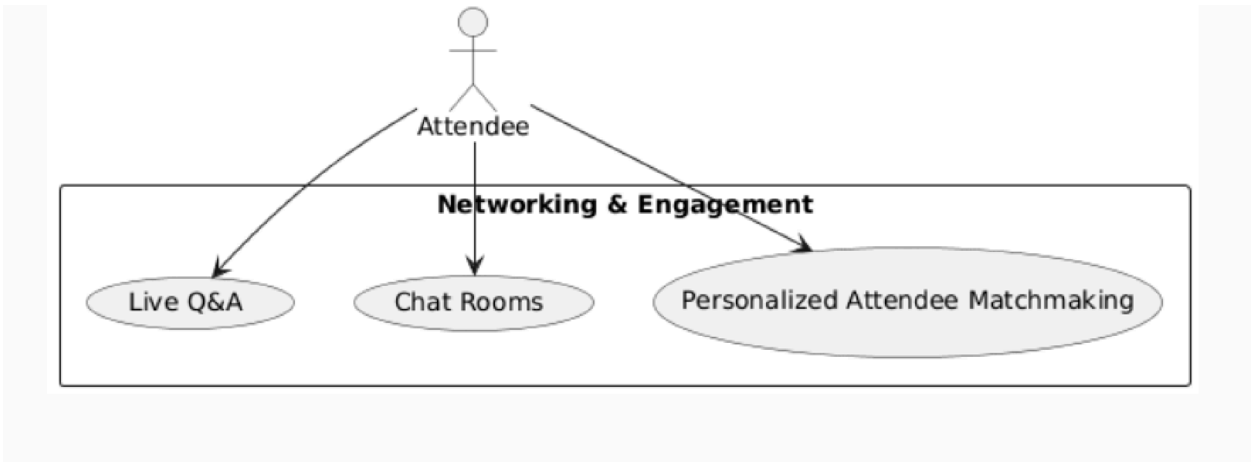
USE CASE 1



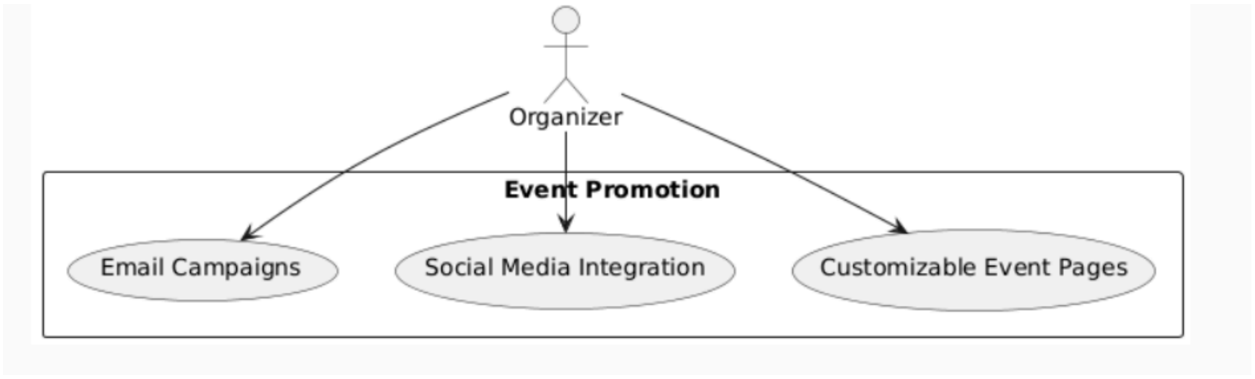
USE CASE 2



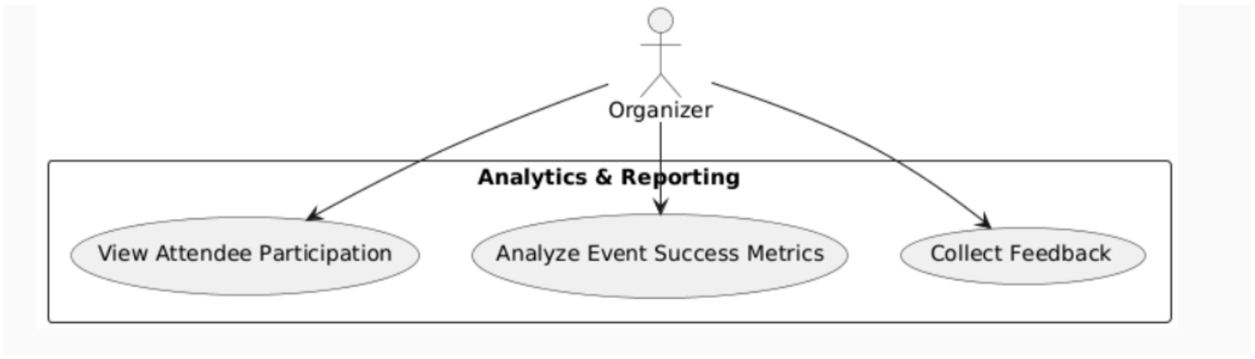
USE CASE 3



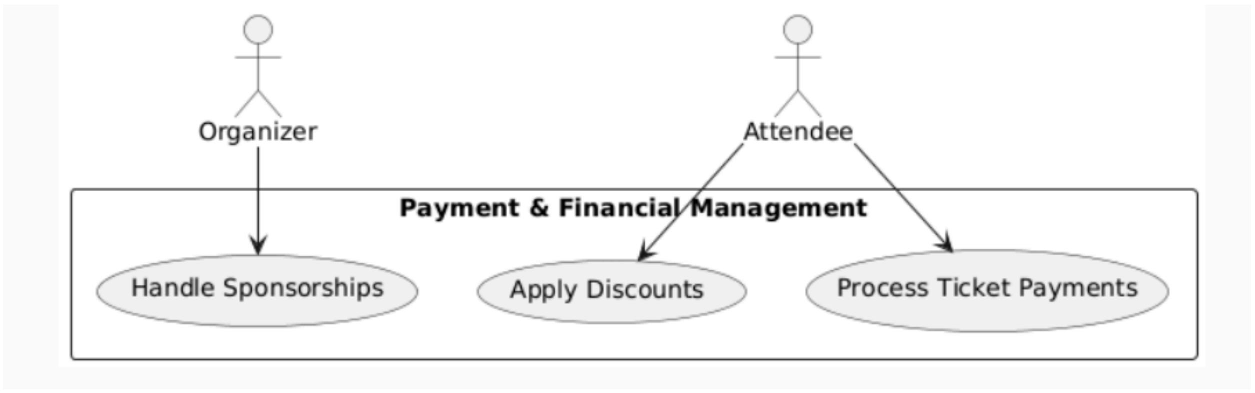
USE CASE 4



USE CASE 5



USE CASE 6



2.2 Use Case Scenarios

ID:	UC-1
Title:	Create and Manage Event Schedule
Description:	This use case describes how an Organizer creates a new event, sets up sessions, and allocates necessary resources to schedule an event.
Primary Actor:	Organizer
Preconditions:	● The Organizer is authenticated and logged into the system. ● Venue and resource information are available in the system.
Postconditions:	● A new event, along with its sessions and allocated resources, is successfully stored in the system. ● The event is ready for further modifications and can be viewed by potential attendees if published
Inputs:	● Event details ● Session details ● Resource details
Outputs:	● Confirmation message indicating successful event creation ● An updated event schedule displayed on the Organizer's dashboard
Main Success Scenario:	1. The Organizer selects the "Create Event" option from the dashboard. 2. The system presents a form for entering event details. 3. The Organizer fills in the event information and selects a venue from the resource list. 4. The Organizer adds one or more sessions with specific time slots. 5. The Organizer assigns required resources to each session. 6. The system validates all inputs and saves the event, sessions, and resource allocations. 7. A confirmation message is displayed, and the new event appears on the Organizer's dashboard.

ID:	UC-2
Title:	Register for Event
Description:	This use case details how an Attendee registers for an event.
Primary Actor:	Attendee
Preconditions:	● The Attendee has an active account or is able to create one. ● The selected event is currently open for registration.
Postconditions:	● The Attendee is successfully registered for the selected event. ● The Attendee’s profile is updated to include the event registration, which is then visible in their dashboard.
Inputs:	● Attendee credentials (username, password) ● Selected event identifier ● Payment information (if applicable)
Outputs:	● A registration confirmation message ● Updated profile information reflecting the event registration
Main Success Scenario:	1. The Attendee logs into the system or creates a new account and then logs in. 2. The system displays a list of available upcoming events. 3. The Attendee selects an event to view detailed information. 4. The Attendee clicks the "Register" button for the chosen event. 5. The system processes the registration, including payment if required. 6. A confirmation message is displayed, and the event registration is recorded in the Attendee’s profile. 7. The Attendee can view or update their profile information via their dashboard.

ID:	UC-3
Title:	Participating in discussion forums and chat rooms
Description:	This use case details how an Attendee uses our networking features
Primary Actor:	Attendee
Preconditions:	● The Attendee has an active account or is able to create one. ● The Attendee selects an open forum or live chat about an event.
Postconditions:	● The Attendee is actively participating with others with the networking features.. ● The Attendee's profile is updated to include the chat rooms and/or discussion forum, which is then visible in their dashboard.
Inputs:	● Attendee credentials (username, password) ● Selected event or discussion topic identifier
Outputs:	● A confirmation message for joining the networking activity. ● Updated profile information reflecting the activity registration
Main Success Scenario:	1. The Attendee logs into the system or creates a new account and then logs in. 2. The system displays a list of available networking activities. 3. The Attendee selects an activity to view. 4. The Attendee clicks the "Join" button for the chosen activity. 5. The system processes the registration 6. A confirmation message is displayed, and the networking activity is recorded in the Attendee's profile. 7. The Attendee can view or update their profile information via their dashboard.

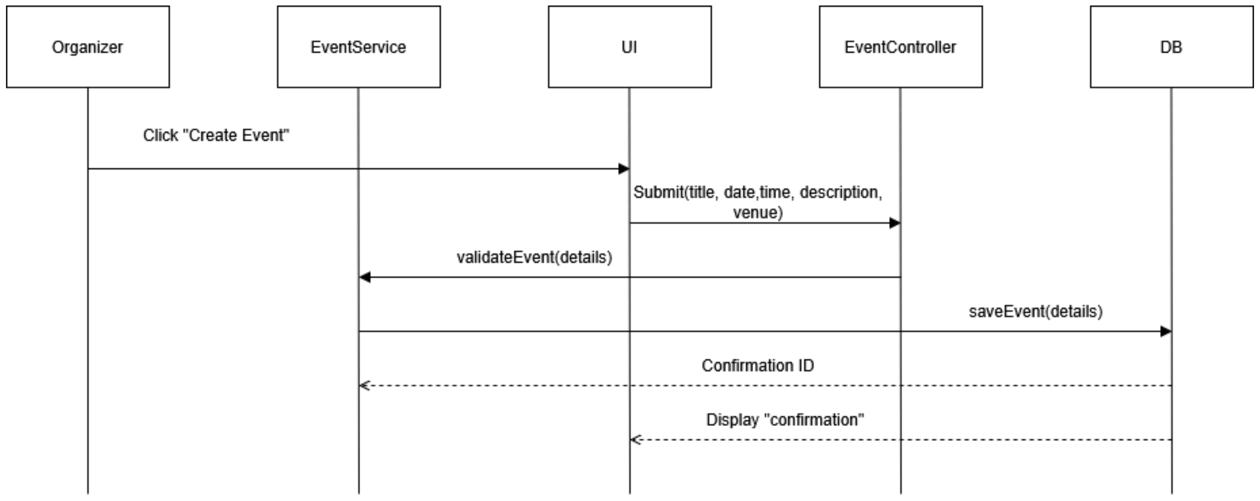
ID:	UC-4
Title:	Marketing & Promotion Tools for Event Planners
Description:	This use case describes how an Organizer promotes a new event,
Primary Actor:	Organizer
Preconditions:	● The Organizer is authenticated and logged into the system. ● The Organizer has at least one event created.
Postconditions:	● The event is successfully promoted and visible to Attendees ● The promotion is displayed in the Organizer’s dashboard.
Inputs:	● Selected Event ● Promotion settings (target audience, date)
Outputs:	● Confirmation message indicating successful promotion creation ● An updated event schedule displayed on the Organizer’s dashboard
Main Success Scenario:	1. The Organizer selects the "Promote Event" option from the dashboard. 2. The system presents promotion options and settings 3. The Organizer fills in the required details 4. The Organizer adds promotional content (images, descriptions) 5. The system validates all inputs and confirms the promotion 6. A confirmation message is displayed, and the new event promotion appears on the Organizer’s dashboard.

ID:	UC-5
Title:	Receiving Event Analytics and Event Reports
Description:	This use case describes how an Organizer tracks the metrics of an event.
Primary Actor:	Organizer
Preconditions:	● The Organizer is authenticated and logged into the system. ● The Organizer has an ongoing event or a past completed event.
Postconditions:	● The Organizer successfully receives the analytics of an event. ● The system displays the event's performance with charts and graphs.
Inputs:	● Selected event to be analyzed ● Timeframe for analytics and/or specifics (budget, attendance)
Outputs:	● A graphical reports with data tables of event metrics ● Feedback and recommendations based on collected data
Main Success Scenario:	1. The Organizer selects the "View Analytics" option from the dashboard. 2. The system presents event analytics options 3. The Organizer selects the event and the desired timeframe 4. The system retrieves and displays event metrics 5. The Organizer views the analytics report and receives auto generated feedback and recommendation on what to improve.

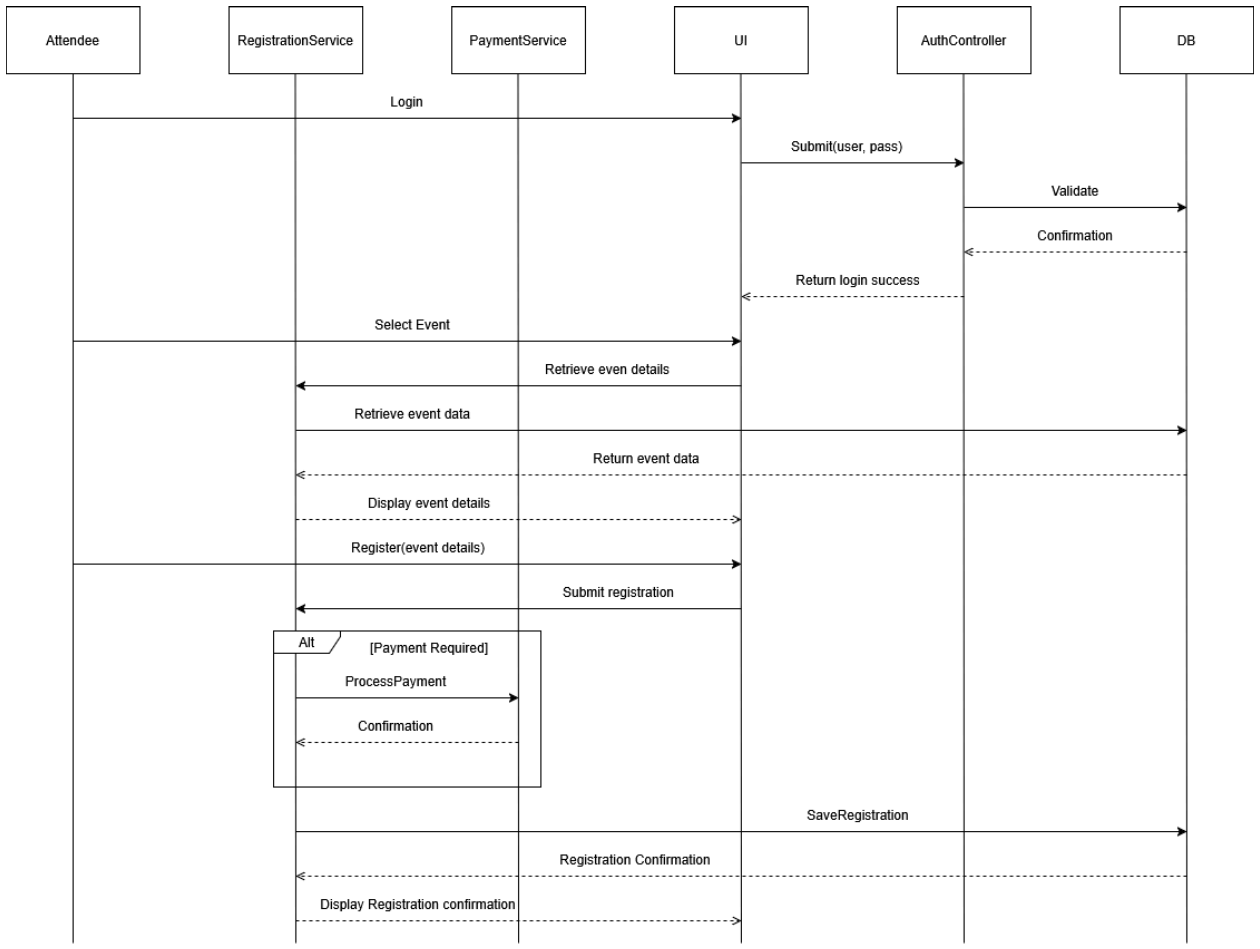
ID:	UC-6
Title:	Securing Payments and Budgeting
Description:	This use case details how an Attendee pays for an event and uses a budget management system
Primary Actor:	Attendee
Preconditions:	● The Attendee has an active account or is able to create one. ● The Attendee receives a prompt to pay for a selected event ● The Attendee has a valid payment method linked or is able to enter one.
Postconditions:	● The payment is successfully processed. ● The event ticket is issued to the Attendee's profile ● The Attendee's budget records are updated
Inputs:	● Attendee credentials (username, password) ● Selected event ● Payment method details
Outputs:	● A confirmation message for the successful payment. ● Updated profile information reflecting the event registration
Main Success Scenario:	1. The Attendee logs into the system or creates a new account and then logs in. 2. The Attendee selects an event and proceeds to payment 3. The Attendee chooses a payment method and enters payment details. 4. The system processes the payment method. 5. A confirmation message is displayed, and the event ticket is sent to the Attendee's profile. 6. The Attendee can view their financial analytics and their budget information via the budget management system.

3. Sequence Diagrams

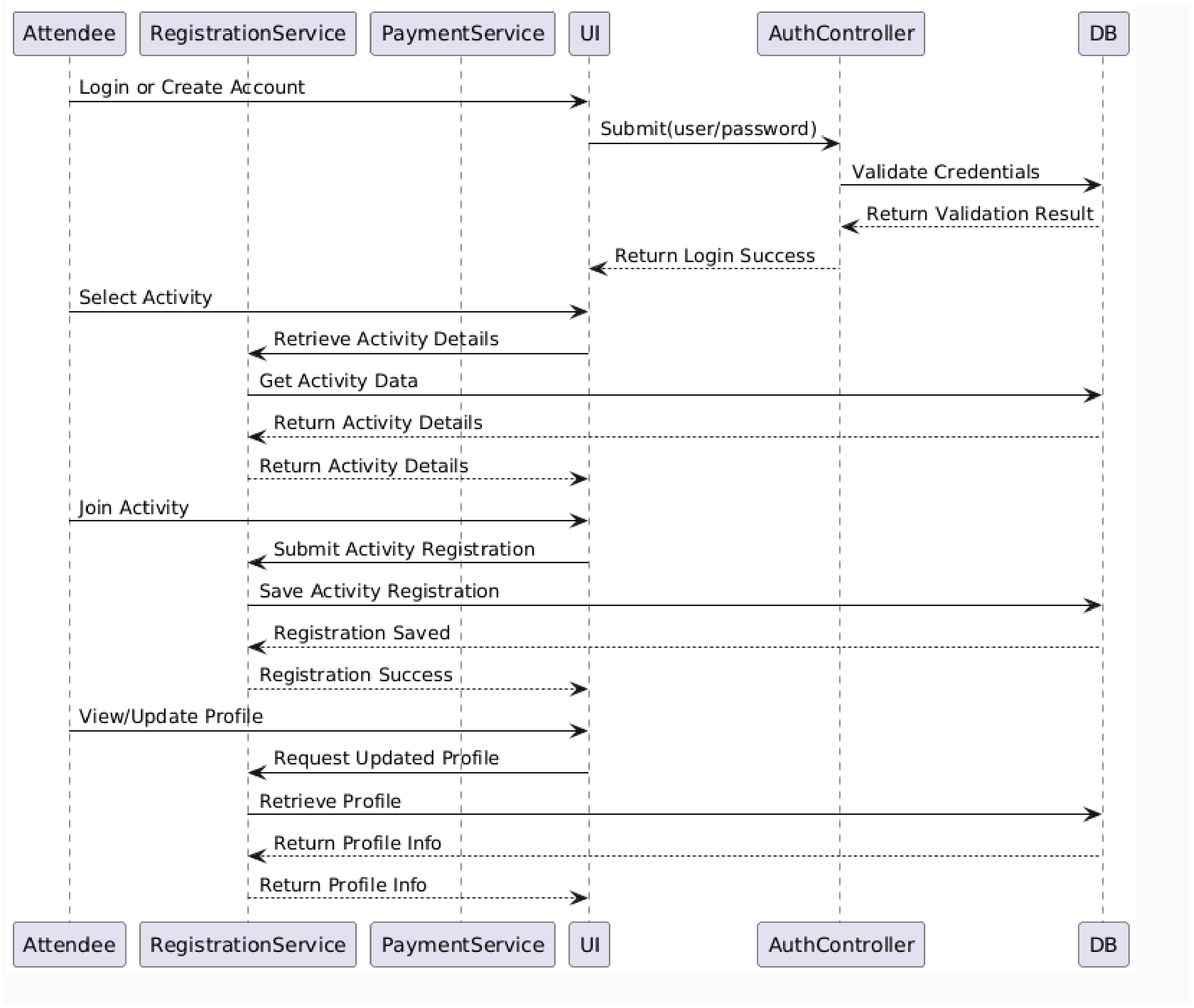
Sequence diagram for UC-4



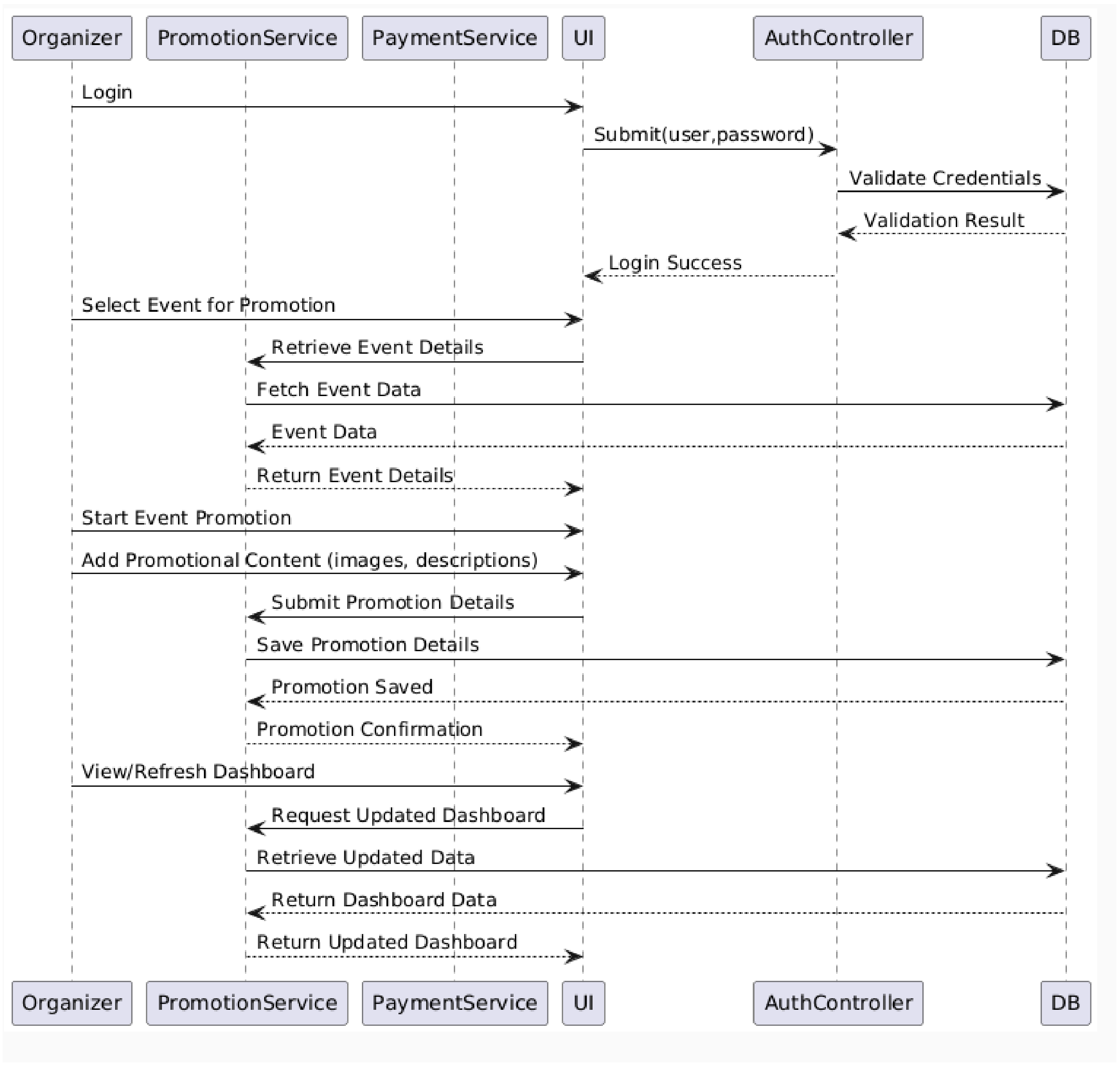
Sequence diagram for UC-2



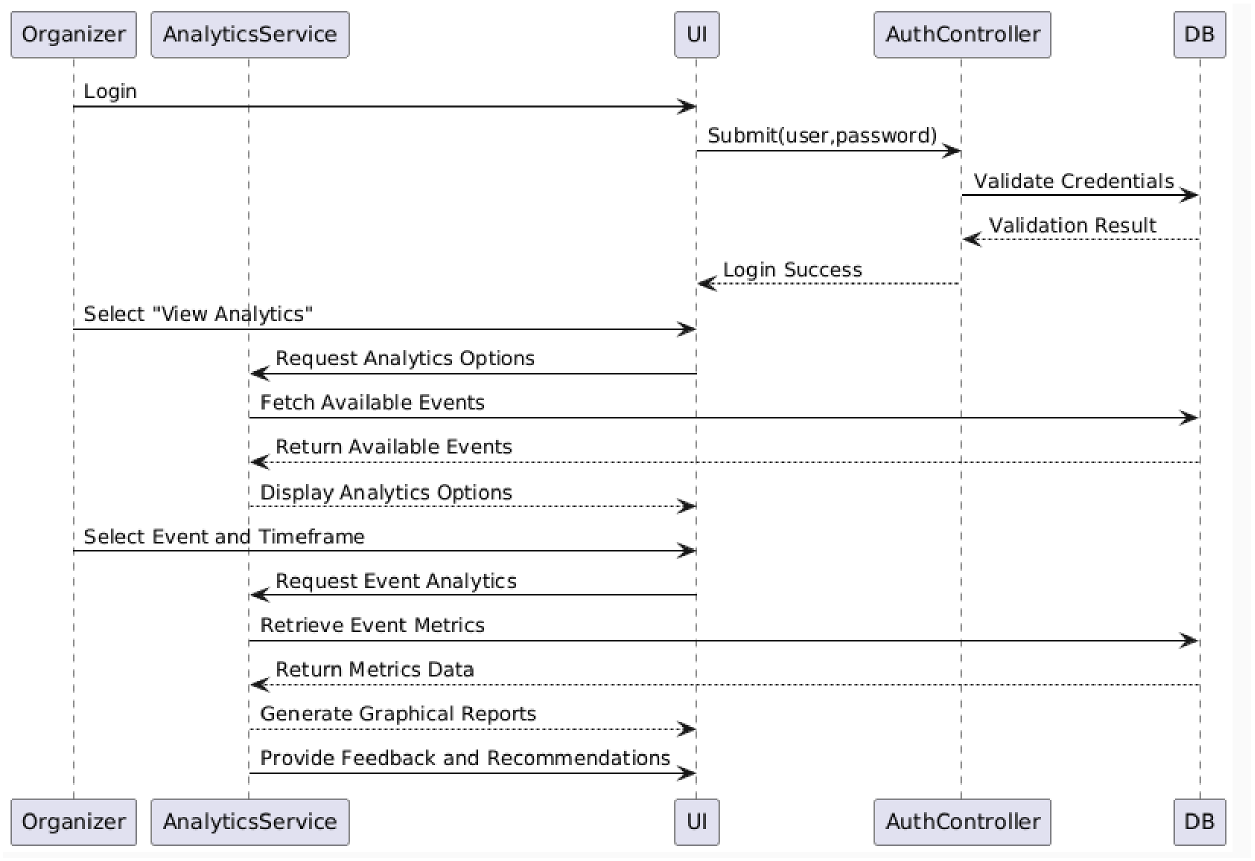
Sequence diagram for UC-3



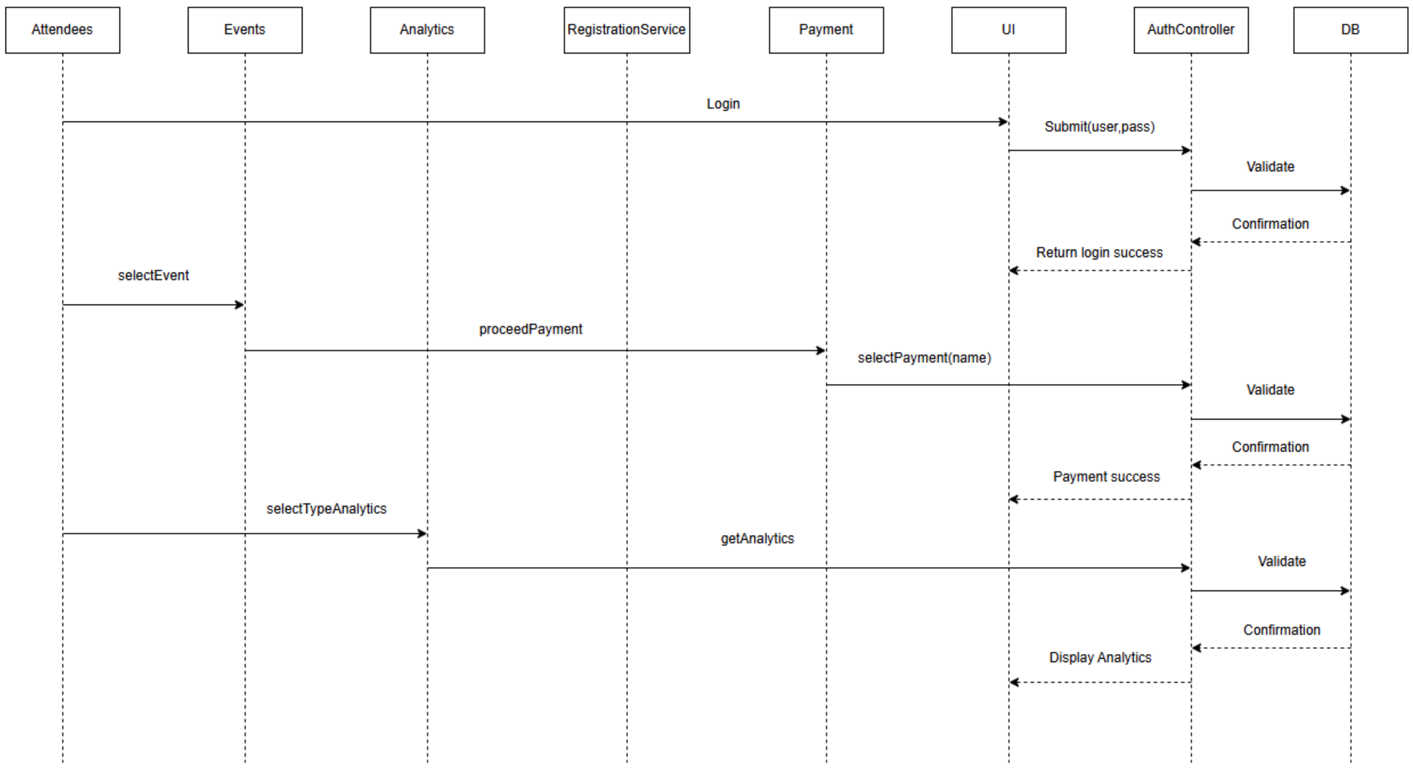
Sequence diagram for UC-4



Sequence diagram for UC-5



Sequence diagram for UC-6


Attendees
Events
Analytics
RegistrationService
Payment
UI
AuthController
DB
Login
Submit(user,pass)
Validate
Confirmation
Return login success
selectEvent
proceedPayment
selectPayment(name)
Validate
Confirmation
Payment success
selectTypeAnalytics
getAnalytics
Validate
Confirmation
Display Analytics