**Sprint 2 – SOEN 343**

**Smart Education Events System**

**Team members ( Stats and Furious):**

* Peter Samaha (40238955)
* Boudy Joe Samaha (40238965)
* Samer Hasna (40234608)
* Georges Ghazal (40231026)

**Document Purpose:**

The goal of this document is to present the second phase of the Smart Education Events System. This document contains the chosen architecture, the reasoning behind it, use cases and sequence diagrams.

Presented on: 2025-03-06

**Table of contents:**

|  |  |
| --- | --- |
| Content | Page |
| Introduction page | 1 |
| Table of Content | 2 |
| System Architecture | 3-4 |
| Comparison of architectures | 5 |
| Use case diagram | 6 |
| Use case scenarios | 7-8-9-10-11-12 |
| Sequence diagram | 13-14-15-16-17-18-19 |

**1)System Architecture**

**Chosen Architecture: Layered Architecture**

For the Smart Education Events System (SEES), a **layered architecture** is the most suitable choice due to its ability to structure complex functionalities into distinct, manageable layers. SEES is designed to handle multiple responsibilities, including event planning, attendee management, networking, analytics, and payment processing. A structured approach ensures that each of these features operates efficiently while remaining maintainable and scalable.

**The layered architecture organizes SEES into four primary layers, ensuring a clear separation of concerns:**

**1)Presentation Layer (Frontend/UI):**

This is the interface through which users interact with SEES, via a web page in this case.

Organizers can create and manage events, attendees can register, and stakeholders can monitor event success.

It communicates with the backend using RESTful APIs for data retrieval and submission.

**2)Business Logic Layer (Service Layer):**

This layer processes the core functionalities of SEES, ensuring that events are scheduled properly, payments are validated, and attendee participation is tracked.

For example, when a Graduate Students Association (GSA) event offers discounted tickets for university students, the business logic layer handles discount verification and eligibility checking.

Features like analytics for event performance are implemented here.

**3)Data Access Layer (Persistence Layer):**

This layer ensures secure and efficient interaction with the database.

It manages user profiles, event details, registrations, and financial transactions.

**4)Database Layer:**

This is the central repository for all system data.

It stores event metadata, user credentials, attendee participation history, and financial records and all other relevant data.

**Why Layered Architecture is the correct choice:**

**Scalability** – SEES must handle various event sizes, from small workshops to large conferences. The presentation and business logic layers can scale independently to support more users without affecting backend operations.

**Maintainability** – The platform must remain adaptable for new features (e.g., AI-driven event recommendations that we proposed in the first sprint). The modular structure makes enhancements easier.

**Security** – SEES manages sensitive data, including payment information and user credentials. A separate data access layer ensures secure database transactions, while role-based access control (RBAC) in the business logic layer restricts functionalities based on user roles.

**Comparison with an Alternative: Microservices Architecture**

A layered architecture provides a structured and modular approach to SEES, making it easier to develop, maintain, and scale in a controlled manner. By separating the system into distinct layers—presentation, business logic, data access, and database—it ensures a clear separation of concerns. This makes debugging and updates more manageable since changes in one layer do not necessarily affect others. Additionally, security and role-based access control can be implemented efficiently at different layers. However, a major trade-off is that scalability is limited at the layer level. As traffic increases, the entire application may need to scale, rather than just the most demanding components. This can lead to performance bottlenecks, particularly in high-traffic scenarios where certain functionalities, like event analytics or ticket purchasing, require more resources than others.

On the other hand, a microservices architecture would break SEES into independent services, each handling a specific responsibility—event management, payments, networking, analytics, and so on. This allows scalability, and only the most demanding components can be scaled independently, optimizing resource usage and performance. Additionally, fault isolation ensures that if one service, such as payments, fails, the rest of the system continues to function. However, microservices introduce significant complexity. They require API gateways, container orchestration, and inter-service communication. Managing data consistency across multiple services also becomes more challenging, requiring distributed databases or event-driven mechanisms. While microservices offer great flexibility and scalability, they demand a higher level of infrastructure and operational expertise, making them less practical for SEES’s current development stage.

Given the scope of SEES, a layered architecture is the best choice for now, providing a balance between modularity, maintainability, and security.

2.1) Use case Diagram:

A diagram of a person's organization

AI-generated content may be incorrect.

2.2)Use case Scenarios:

**Use Case 1: Event Planning & Scheduling ID**

**Title: Event Planning & Scheduling**

Description: Allows organizers to create, update, and manage event schedules, sessions, and allocate resources (rooms, speakers, etc.).

Primary Actor: Organizer

**Preconditions:**

1.The organizer is logged into the system.

2.The organizer has the necessary permissions to create or modify events.

**Postconditions:**

3.The event is created, updated, or canceled.

4.Sessions are scheduled with allocated resources.

**Inputs:**

•Event details (title, date, format, start/end times).

•Session details (topics, speakers, timeslots).

•Resource allocation (venues, equipment).

**Outputs:**

•A finalized or updated event schedule.

•Confirmation message for successful changes.

**Main Success Scenario:**

1.The organizer logs in and navigates to “Manage Event.”

2.The organizer selects “Create New Event” or “Edit Existing Event.”

3.The organizer enters/modifies event details.

4.The organizer adds/modifies sessions and assigns venues/resources.

5.The system verifies conflicts (venue availability, time overlaps).

6.The system updates the event and displays confirmation.

**Use Case 2: Attendee Registration & Management ID**

**Title: Attendee Registration & Management**

Description: Allows attendees to register for events, manage their profiles, and receive confirmation details.

Primary Actor: Attendee

**Preconditions:**

1.The event is published and open for registration.

2.The attendee can access the registration page.

**Postconditions:**

3.The registration is saved in the system.

4.A ticket is generated, and payment is processed (if required).

5.The attendee’s profile is created/updated.

**Inputs:**

•Attendee details (name, email, affiliation).

•Payment details (if applicable).

**Outputs:**

•Registration confirmation (email or ticket).

•Updated attendee profile.

**Main Success Scenario:**

1.The attendee accesses the event registration page.

2.The attendee fills out personal and payment information.

3.The system validates data and processes payment (if required).

4.The system generates a ticket and sends confirmation.

5.The attendee can view and update their profile.

**Use Case 3: Networking & Engagement ID**

**Title: Networking & Engagement**

Description: Enables real-time Q&A, chat rooms, and attendee matchmaking.

Primary Actor: Attendee

**Preconditions:**

1.The attendee is registered and logged in.

2.The networking features are enabled by the organizer.

**Postconditions:**

3.Attendees can interact in real-time networking features.

4.Matchmaking suggestions are generated.

**Inputs:**

•Chat messages, Q&A submissions, matchmaking preferences.

**Outputs:**

•Real-time chat, Q&A, and networking recommendations.

**Main Success Scenario:**

1.The attendee navigates to the networking area.

2.The attendee interacts in chat( or AI bot) or Q&A sessions.

3.The system processes input and updates interactions.

4.The system provides matchmaking recommendations.

**Use Case 4: Event Promotion ID**

**Title: Event Promotion**

Description: promotion use marketing tools to promote events.

Primary Actor: Promotion

**Preconditions:**

1.Event details exist in the system.

2.The promotion has promotional permissions.

**Postconditions:**

3.Marketing campaigns are scheduled/launched.

4.Event pages are updated with promotional content.

**Inputs:**

•Marketing content, campaign settings.

**Outputs:**

•Scheduled campaigns, social media posts, analytics.

**Main Success Scenario:**

1.The promotion accesses the promotion module.

2.The promotion sets up a campaign and audience.

3.The system schedules or launches the campaign.

4.The promotion tracks campaign performance.

**Use Case 5: Analytics & Reporting ID**

**Title: Analytics & Reporting**

Description: stakeholdder access real-time event insights.

Primary Actor: Stakeholder

**Preconditions:**

1.Event data collection is enabled.

Postconditions:

2.stakeholders can view and export reports.

**Inputs:**

•Event attendance, session engagement, feedback.

**Outputs:**

•Dashboards, analytics reports.

**Main Success Scenario:**

1.The stakeholder accesses the analytics dashboard.

2.The system aggregates event data.

3.The stakeholder views and filters reports by event or date.

4.The system updates analytics in real-time.

5.The organizer exports reports.

**Use Case 6: Payment & Financial Management ID**

**Title: Payment & Financial Management**

Description: Handles payment processing for tickets and sponsorships.

Primary Actor: Payment

**Preconditions:**

1.The event has ticket pricing or sponsorship options.

2.The user has valid payment details.

**Postconditions:**

3.Payment is processed, and records are updated.

**Inputs:**

•Payment details.

**Outputs:**

•Payment confirmation, updated financial records.

**Main Success Scenario:**

1.The user selects a ticket or sponsorship package.

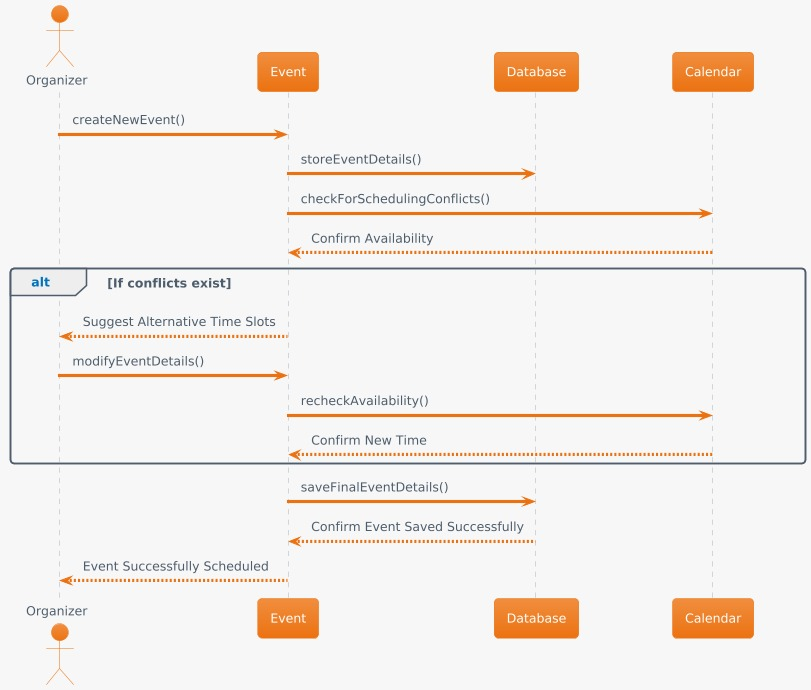
2.The user enters payment details.

3.The system processes the payment securely.

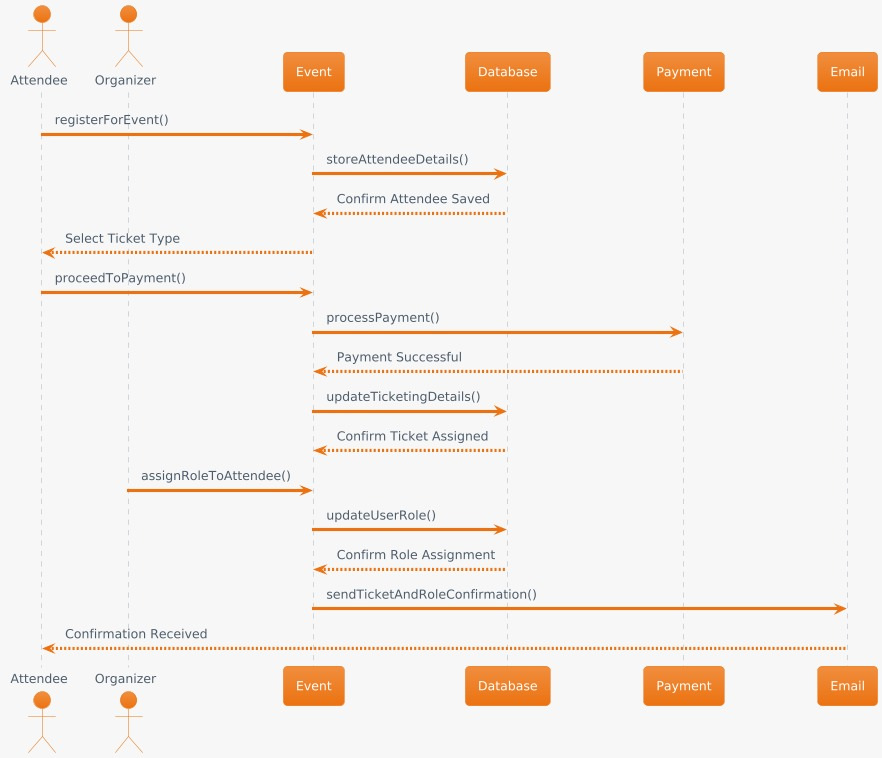
4.The system updates financial records.

3) Sequence diagrams:

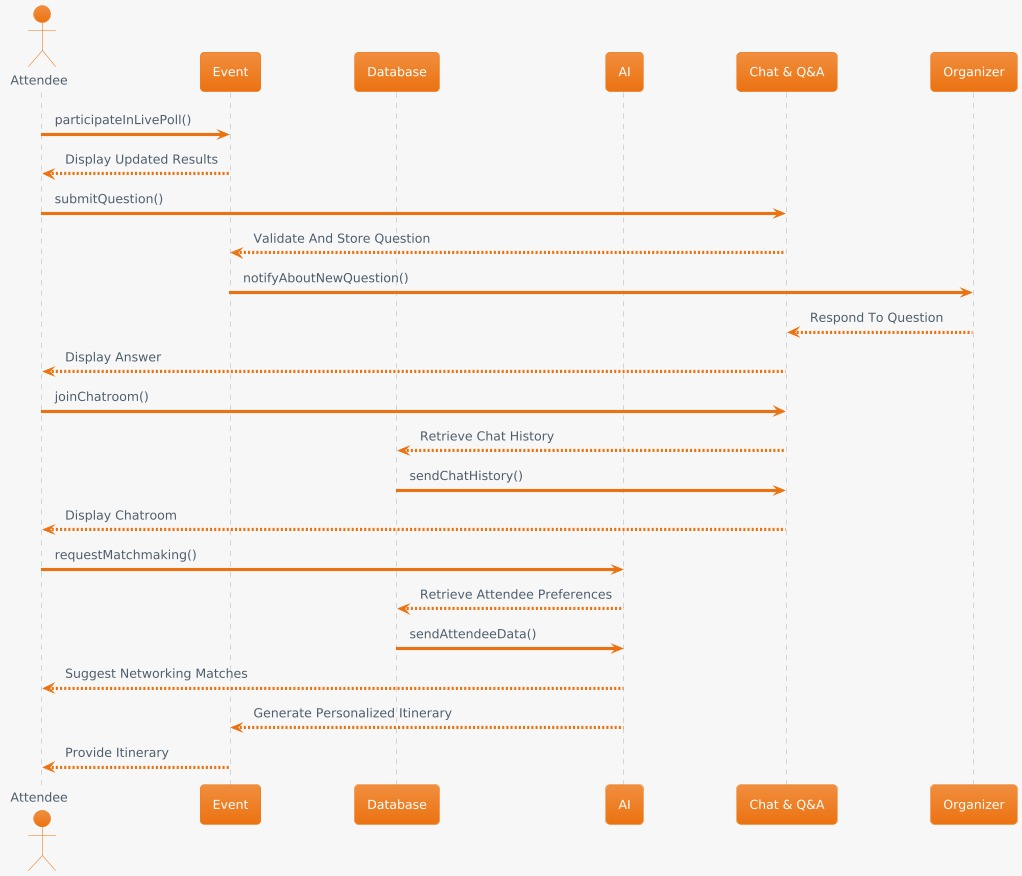
1. Event planning and scheduling

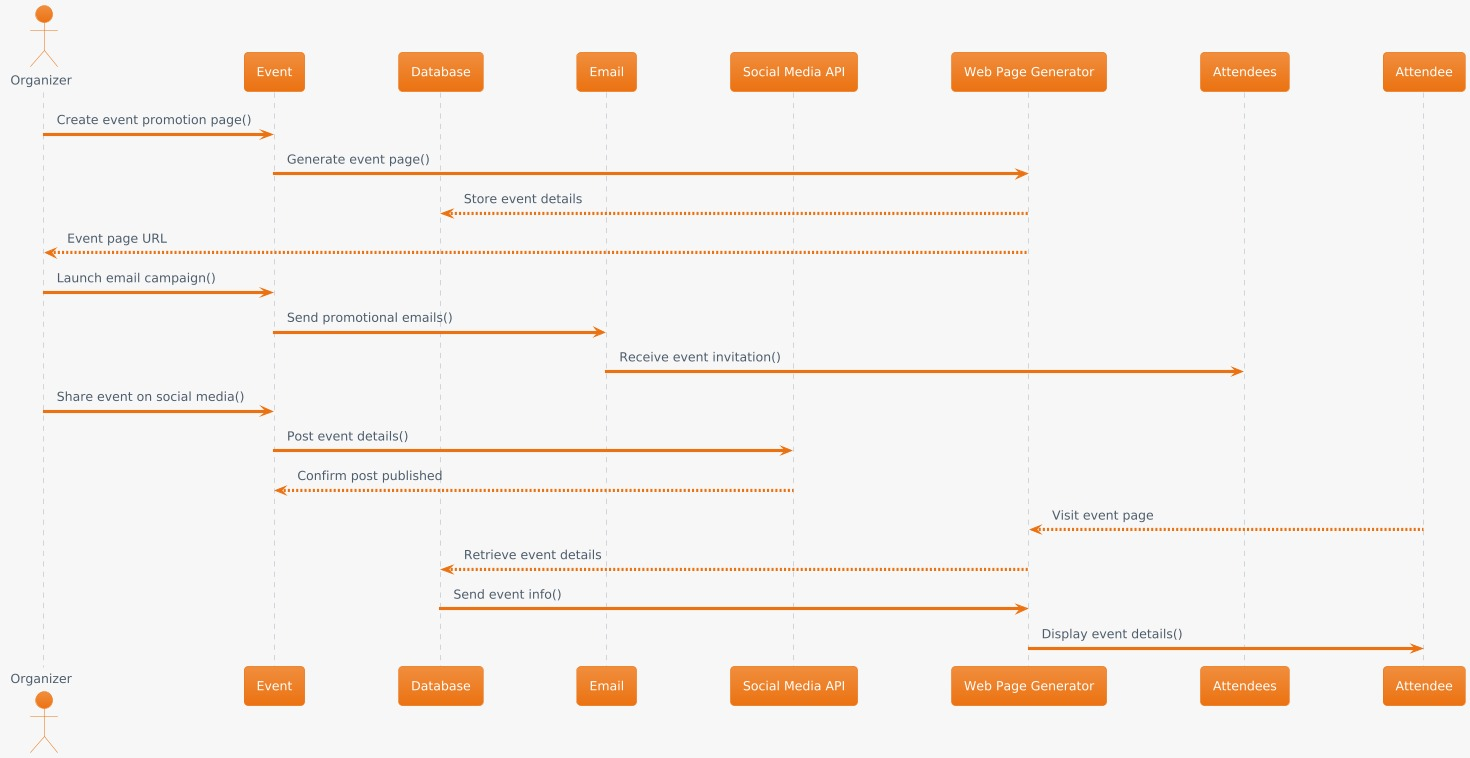


2. Stakeholders and attendees management

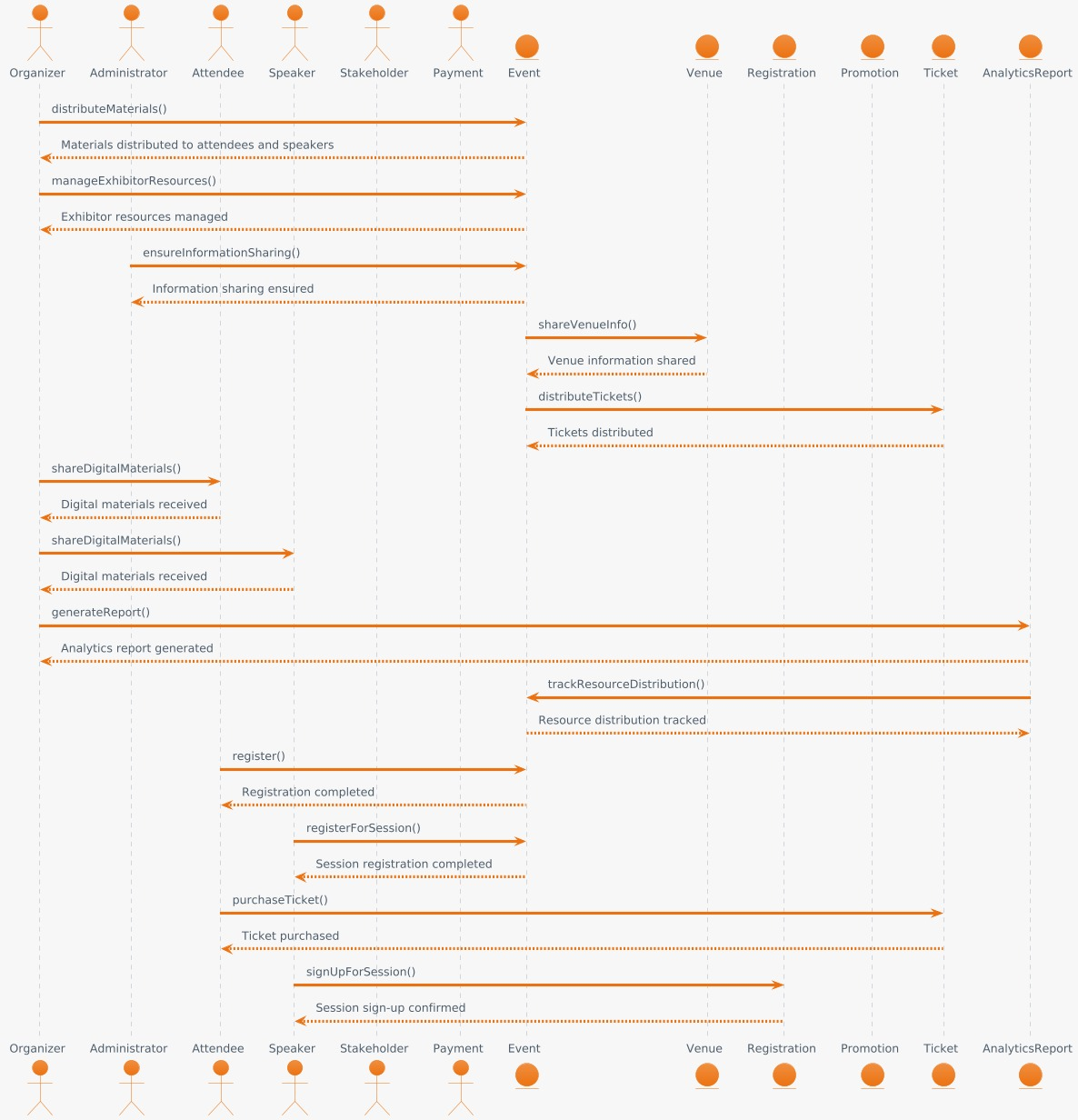


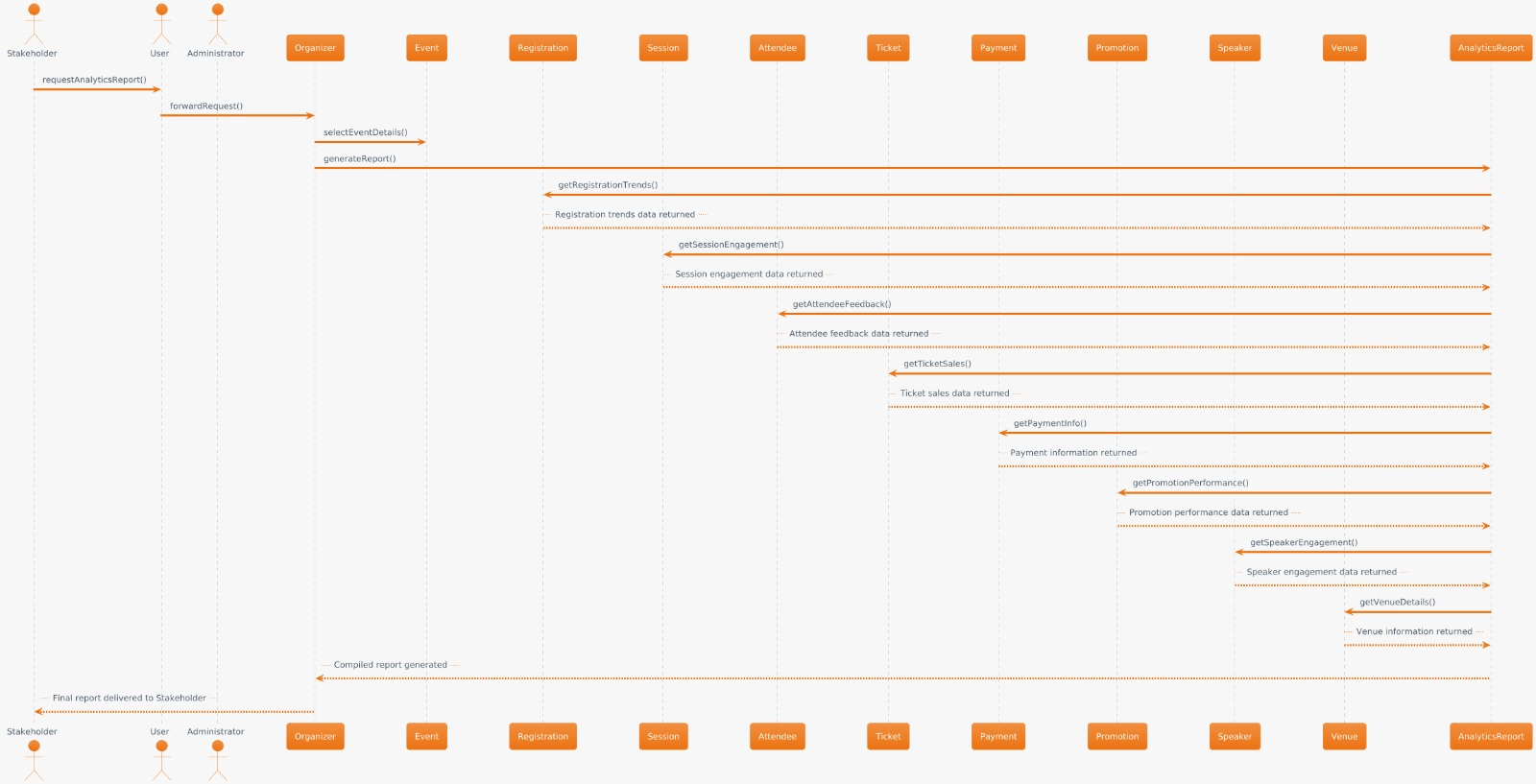
3. Networking and engagement



 4. Event promotion

5. Resource management



6. Analytics and reporting

7. Payment and financial management

