



Event Management System

Name: Kulcsar Noemi

Group: 30234

Table of Contents

Deliverable 1	3
Project Specification	3
Functional Requirements	3
Use Case Model	3
Use Cases Identification	3
UML Use Case Diagrams	Error! Bookmark not defined.
Supplementary Specification	4
Non-functional Requirements	4
Design Constraints	5
Glossary	5
Deliverable 2	5
Domain Model	5
Architectural Design	5
Conceptual Architecture	5
Package Design	5
Component and Deployment Diagram	5
Deliverable 3	5
Design Model	5
Dynamic Behavior	5
Class Diagram	5
Data Model	5
System Testing	6
Future Improvements	6
Conclusion	6
Bibliography	6

Deliverable 1

Project Specification

This project is a web-based event management system that allows users to create, edit, view, and delete events. The system provides functionalities for event organization, venue management, and ticket allocation. The platform enables users to manage event details, including date, location, and capacity, while ensuring data consistency across the system.

Functional Requirements

The key functional requirements of the system are:

1. **Event Creation** – Users can create new events by providing event details such as name, description, date, and venue.
2. **Event Editing** – Users can modify existing events, including event details and venue information.
3. **Event Deletion** – Users can remove events from the system.
4. **Event Listing** – The system displays all available events, including their details and associated venues.
5. **Venue Management** – Users can select venues from a predefined list when creating or editing events.
6. **Capacity Synchronization** – If a venue's capacity is modified, the change is reflected across all events using that venue.

Use Case Model 1

Use Cases Identification

Use Case: Create Event

Level: User interaction

Primary Actor: Event Organizer

Main success scenario:

1. User navigates to the "Create Event" page.
2. User fills in event details (name, description, date).
3. User selects a venue from the predefined list.
4. User submits the form.
5. The system stores the event in the database and confirms creation.

Extensions:

- 3a: If no venue is selected, the system prompts the user to choose one.
- 4a: If invalid data is entered, the system displays an error message.

Use Case: Edit Event

Level: User interaction

Primary Actor: Event Organizer

Main success scenario:

1. User selects an event to edit.
2. User modifies event details.
3. User changes the venue if needed.
4. User saves the changes.

5. The system updates the event and reflects changes across all instances of the venue.

Extensions:

- 3a: If venue capacity changes, all related events update accordingly.

Use Case: Delete Event

Level: User interaction

Primary Actor: Event Organizer

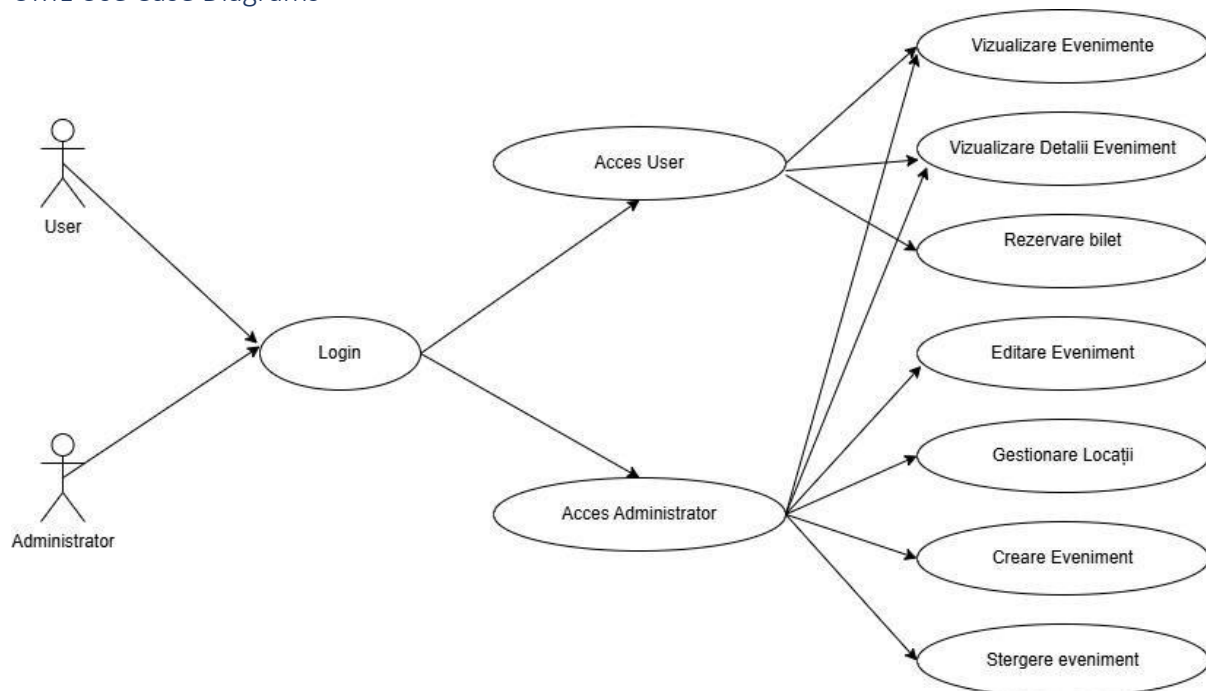
Main success scenario:

1. User selects an event to delete.
2. System prompts for confirmation.
3. User confirms deletion.
4. The system removes the event from the database.

Extensions:

- 2a: If the event has active tickets, the system prevents deletion.

UML Use Case Diagrams



Supplementary Specification

The system provides additional functionalities such as:

- Ensuring venue capacity remains synchronized across all events.
- Allowing easy selection of venues from a predefined list.
- Displaying event lists with relevant filtering options.

Non-functional Requirements

Scalability – The system must support hundreds of simultaneous users creating and managing events.

Usability – The UI must be intuitive, allowing users to create and edit events without requiring technical knowledge.

Data Consistency – Changes in venue information must be reflected across all related events.

Security – Users should only be able to edit or delete events they created, preventing unauthorized modifications.

Design Constraints

Technology Stack: The system is implemented using Spring Boot for the backend, Thymeleaf for the frontend, and a mock database for development.

Development Frameworks: Java 21 with Spring Boot 3.4 is used for backend logic.

Database Management: Currently, the project uses an in-memory mock database, with potential migration to a SQL-based system.

Architecture: Follows an MVC (Model-View-Controller) pattern for clear separation of concerns.

Glossary

Event: A scheduled gathering with a name, description, date, and venue.

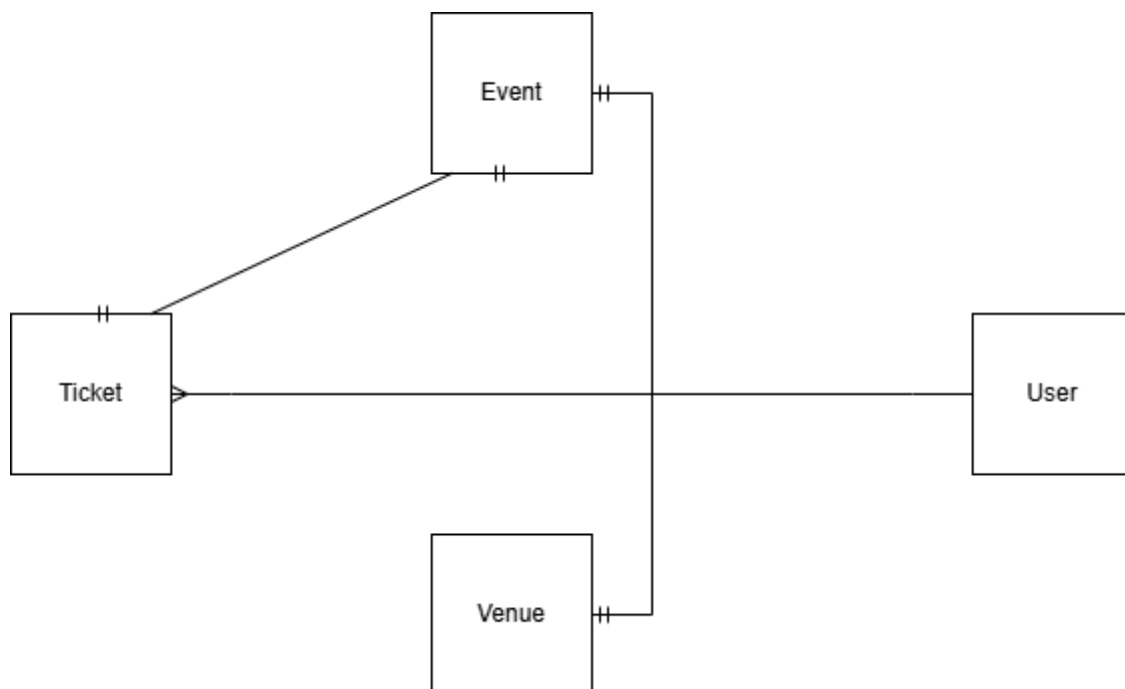
Venue: A location where an event is hosted, including details like name, address, and capacity.

Event Organizer: A user responsible for managing events.

Mock Database: A temporary database used for development and testing purposes.

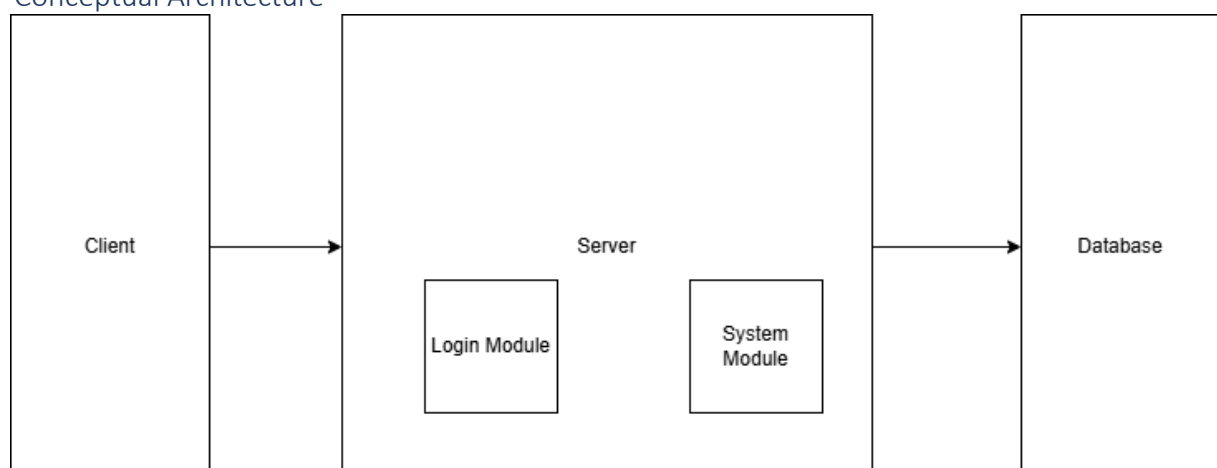
Deliverable 2

Domain Model

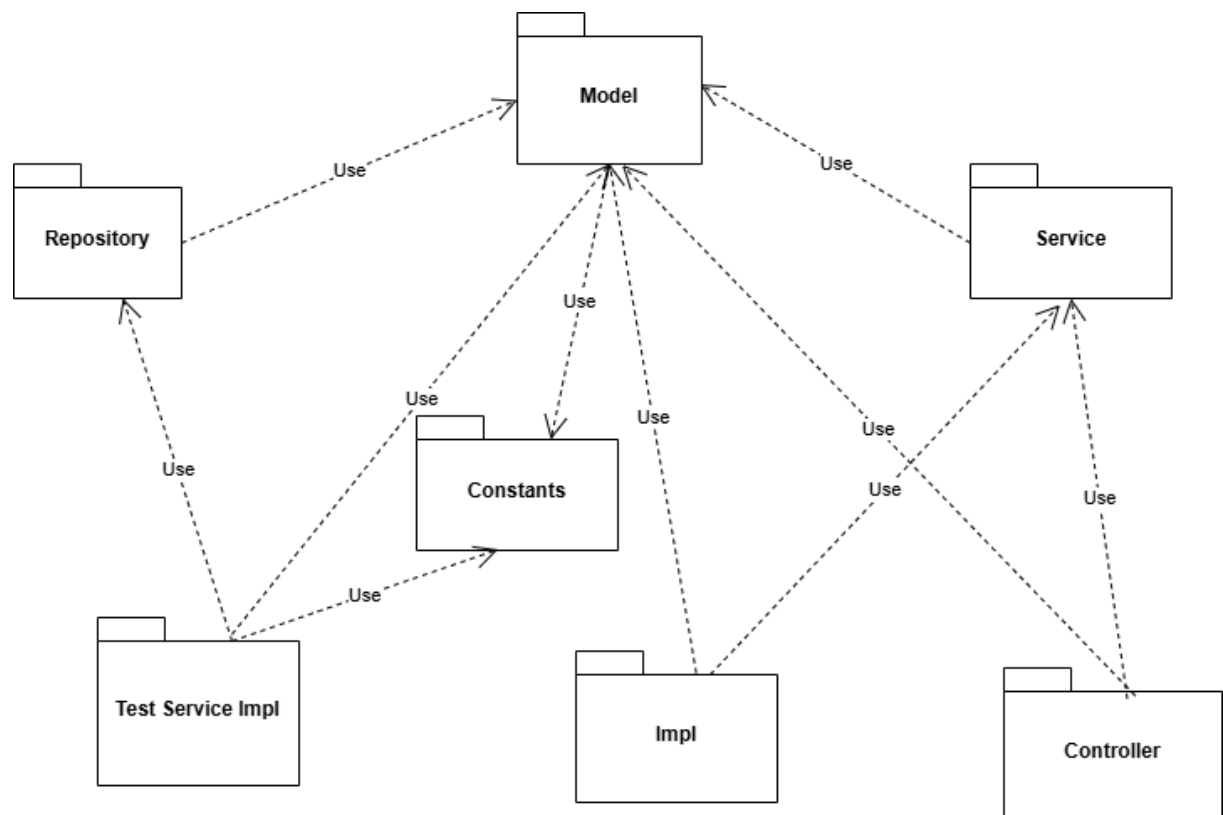


Architectural Design

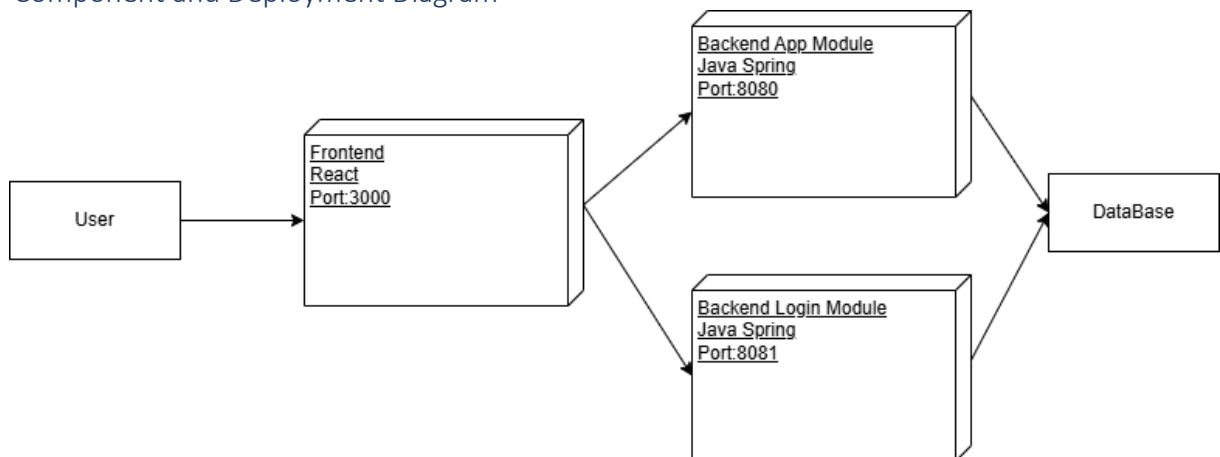
Conceptual Architecture



Package Design



Component and Deployment Diagram



Deliverable 3

Design Model

Dynamic Behavior

[Create the interaction diagrams (2 sequence) for 2 relevant scenarios]

Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

Data Model

[Create the data model for the system.]

System Testing

[Describe the testing methods and some test cases.]

Future Improvements

[Present some features that apply to the application scope.]

Conclusion

Bibliography