Team Project: *Group 01*

Team Members

Number	Name	Email(s)	CSGitLab Username
TM1	Qiyue Zhu	qiyue.zhu@ucdconnect.ie	@Victoria
TM2	Wenyi Liang	wenyi.liang@ucdconnect.ie	@Wenyi
TM3	Yunfei Xu	yunfei.xu@ucdconnect.ie	@Kira210
TM4	Zitong Wang	zitong.wang@ucdconnect.ie	@sukikatte
TM5	Siqi Du	siqi.du@ucdconnect.ie	@Yuki
TM6	Jiayi Cai	jiayi.cai@ucdconnect.ie	@22207285
TM7	Yani Yang	yani.yang@ucdconnect.ie	@hs
TM8	Boran Duan	boran.duan@ucdconnect.ie	@boran

Implementation

This part includes the implementation of the Event Management System. In order to execute the application you will need to have the following installed: 1. Java 17 2. Maven 3. Docker 4. Docker Compose

Running the Application

Assuming that you have correctly installed and set up the above, you can run the application by following these steps: 1. Use maven to create a jar file of the application by running `mvn package` in the root directory of the project.

```
mvn package
```

2. Run the docker compose command to build and run the application.

```
docker-compose up --build -d
```

This will build the application and run it on port 8080. You can then access the application by navigating to `localhost:8080` in your browser. This also starts the database, as such any changes you make should persist between runs.

Stopping the Application

When you are finished and want to stop the application, you can use the following command:

```
docker-compose down
```

If you want to stop the application and remove the database, you can use the following command:

```
docker-compose down -v
```

Event Management System

Administrator

View All Events

1. Log in to the system and access the **“01-main-admin”** dashboard.
2. Navigate to **“check-event-admin”** in the menu.
3. The system will display a list of all events. The event container includes event names.
4. Click on a specific event to view detailed information, including:
    - Event Name
    - Venue ID
    - Date and Time
    - Ticket Type, Total Number, and Sold Number

Delete an Event

1. In the **“check-event-admin”** section, locate the event you want to delete.
2. Click the **“Delete”** button next to the event.
3. A confirmation dialog will appear. Confirm the action, and the event will be permanently removed from the database.

Organizer

Add a New Event

1. Log in to the system and access the Organizer dashboard.
2. Navigate to **“20-Event-organisers”** and click **“Add New Event”**.
3. Fill in the required details:
    - Event Name
    - Description
    - Date
    - Start Time and End Time
    - Venue (select from existing venues)
4. Submit the form. The system will automatically check for venue and time conflicts. If no conflicts

are found, the event will be saved to the database.

View and Edit Your Events

1. In the **“Event Management”** section, view the list of events you have created.
2. Click on a specific event to view its details (e.g., Event Name, Date, Start Time, End Time, Ticket Type).
3. To make changes, click the **“Edit”** button, update the necessary fields, and save the changes. The system will update the database with the new information.

User

View Event List

1. Log in to the system and access the **“03-main-user”** dashboard.
2. Navigate to **“View Events”** in the menu.
3. The system will display a list of all upcoming events, including event names.

View Event Details

1. From the event list, click on the event you are interested in.
2. The system will open the event’s detail page, showing:
    - Event Description
    - Venue
    - Date and Time
    - Ticket Type and Number
    - Organizer Contact Details

Technical Implementation

Architecture

- **Backend**: Spring Boot 3.3.6 with Spring Security
- **Database**: MySQL with JPA/Hibernate ORM
- **Frontend**: Thymeleaf templates with HTML5/CSS3/JavaScript
- **Deployment**: Docker containerization

Key Features

- Multi-role authentication (Administrator, Organizer, User)
- Complete CRUD operations for events, venues, and tickets
- Role-based access control
- Responsive web interface
- Database persistence with MySQL

Project Structure

```
src/
├── main/
│   ├── java/
│   │   └── ucd/comp3013j/ems/
│   │       ├── model/          # Entity classes
│   │       ├── controllers/    # Spring MVC controllers
│   │       ├── services/       # Business logic services
│   │       └── websecurity/    # Security configuration
│   └── resources/
│       ├── templates/          # Thymeleaf HTML templates
│       ├── static/            # Static resources (CSS, JS,
images)
│       └── application.yaml    # Configuration files
```

Milestone 4 Implementation

Distribution of work on this milestone

Overall Distribution of Work

Team Member	TM1	TM2	TM3	TM4	TM5	TM6	TM7	TM8	Sean
Percentage	12%	12%	12%	12%	12%	12%	12%	12%	4%

Task Allocation

Item	Contributor
UI development	@sukikatte, @22207285, @boran
Use Case 1: Create Account	@hs
Use Case 2: List Account	@Yuki
Use Case 3: View Account	@Yuki
Use Case 4: Modify Account	@Yuki
Use Case 5: Register Account	@hs
Use Case 6: List Venues	
Use Case 7: Add Venue	
Use Case 8: View Venue	
Use Case 9: Modify Venue	@hs
Use Case 10: Delete Venue	@hs
Use Case 11: List Events	
Use Case 12: Add Event	
Use Case 13: View Event	
Use Case 14: Modify Event	
Use Case 15: Delete Event	@hs
Use Case 16: List My Event	
Use Case 17: List Tickets	
Use Case 18: Buy Tickets	@boran
Use Case 19: View Tickets	

Reflection Statements

Team Member	Contribution Reflection Statement
@hs	Implement Register Account, Modify Venue, Delete Venue, Delete Event.
TM2	Implemented user authentication and security features, including Spring Security configuration and role-based access control.
TM3	Developed the core entity classes and database relationships, including Account, Event, Venue, and Ticket models with JPA annotations.
TM4	Created comprehensive UI templates and frontend components, ensuring responsive design and user-friendly interfaces across all user roles.
TM5	Implement profile pages for three user types, an admin list accounts page, and a view & modify account page.
TM6	Developed event management functionality including CRUD operations for events and venue management features.
TM7	Implemented ticket purchasing system and user ticket management features, including purchase history and ticket details.
TM8	Developed venue management system and event creation functionality, including venue CRUD operations and event-organizer relationships.

System Implementation Images

This section showcases the key images used in the Event Management System implementation.

Background Images

Login Background

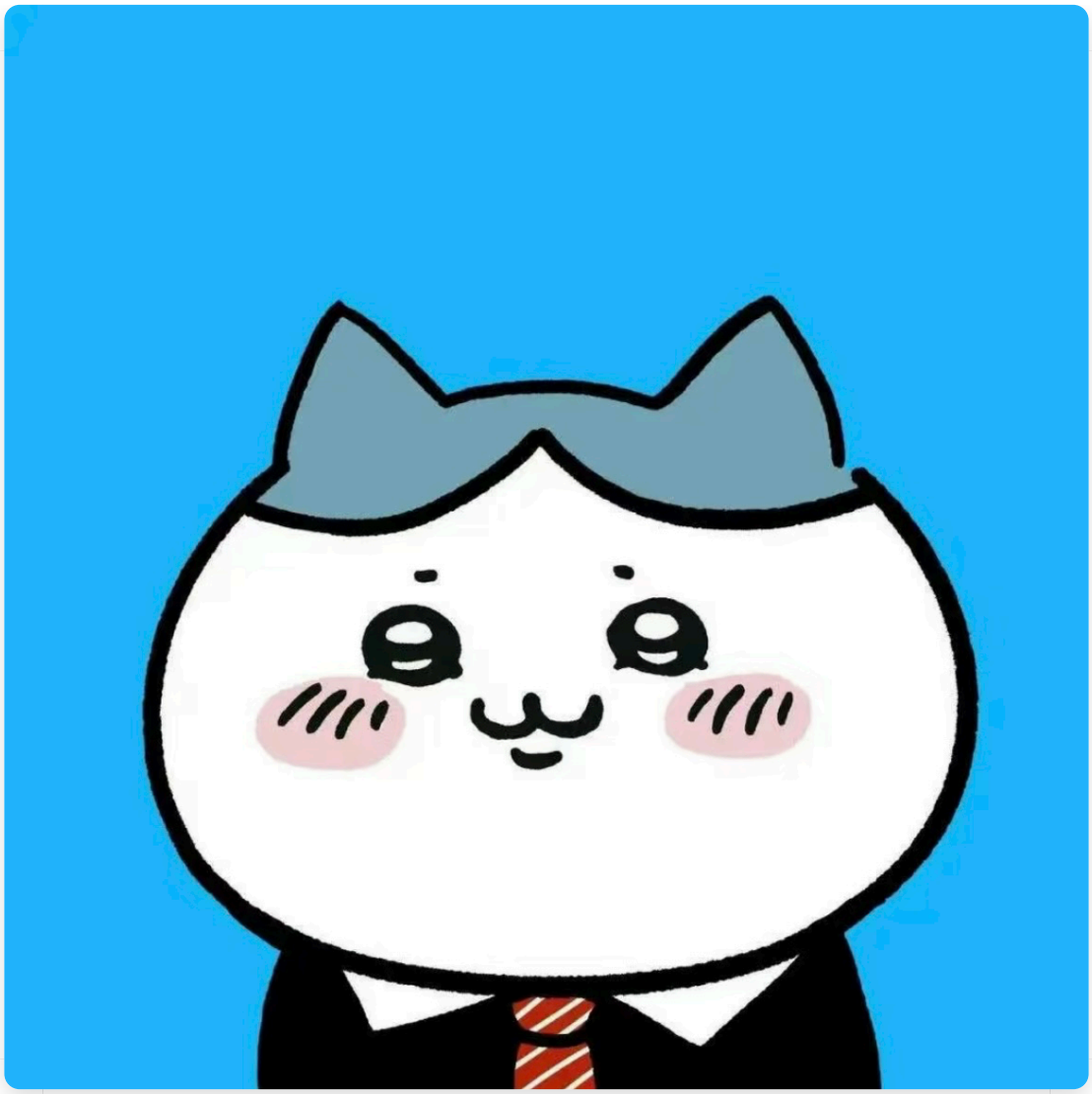
Login Background

Main Page Background

Main Page Background

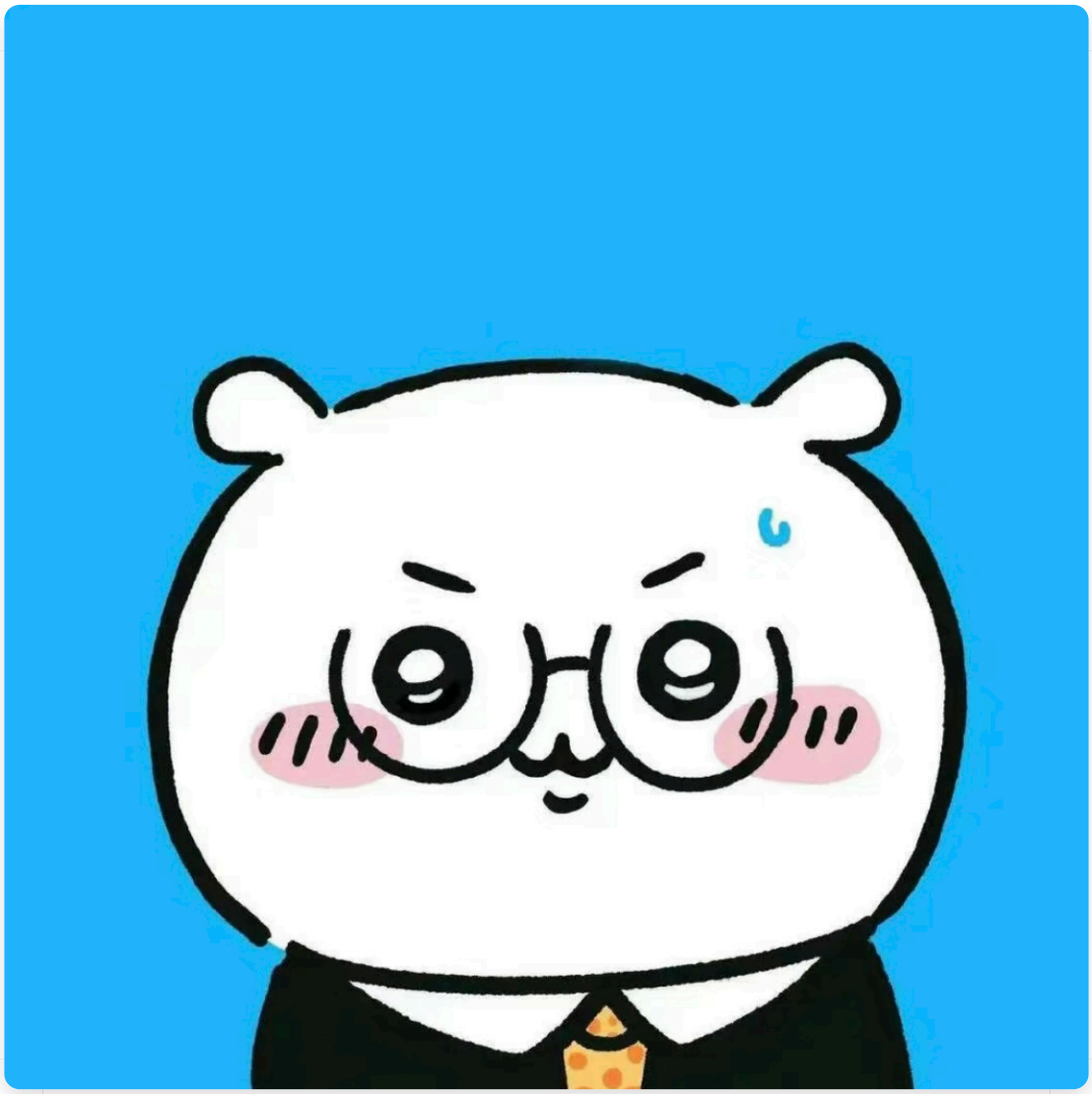
User Role Icons

Administrator Icon



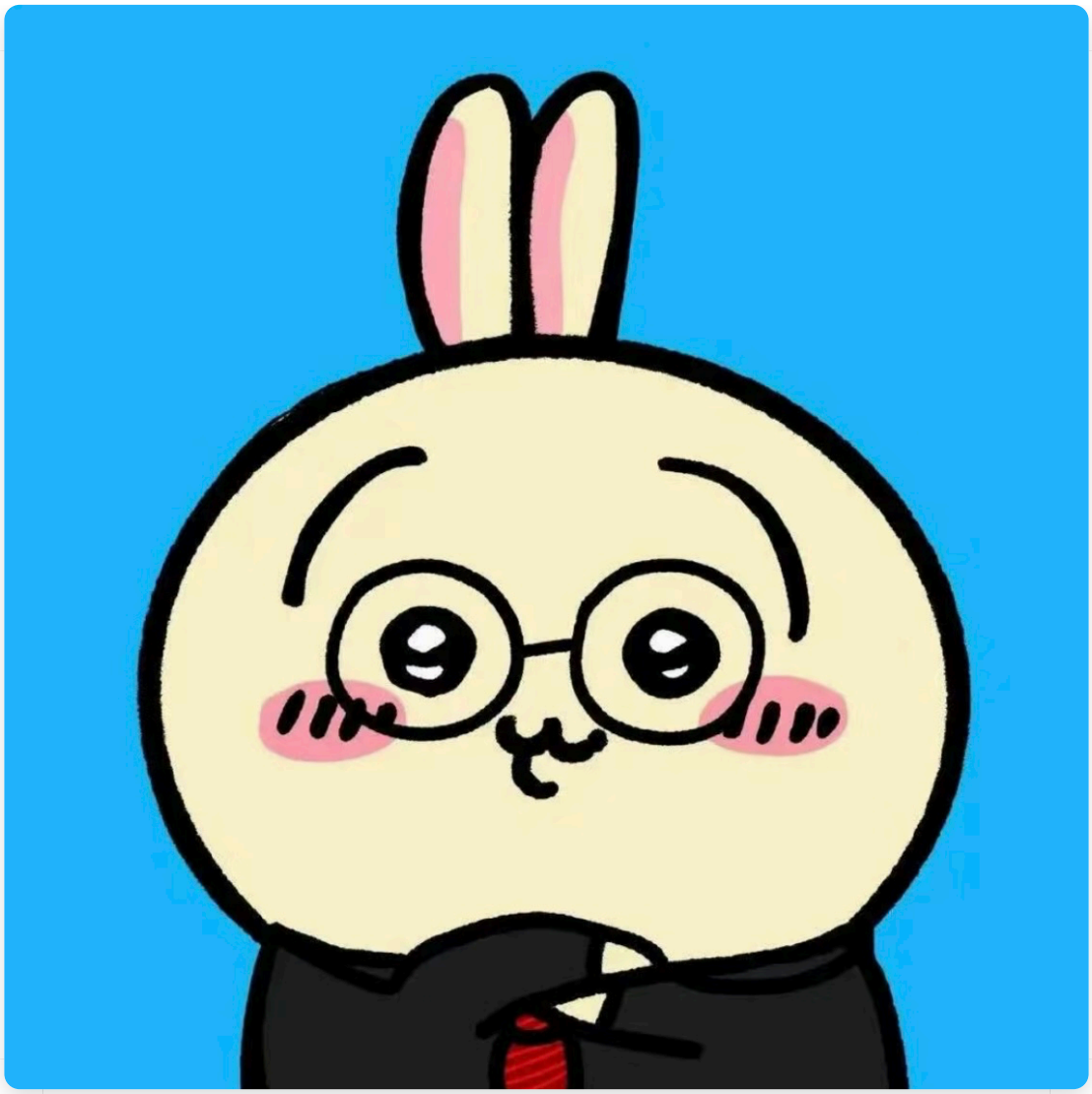
Administrator Icon

Organiser Icon



Organiser Icon

User Icon



User Icon

Event Management

Event Icon

Event Icon

Technical Implementation

The Event Management System is implemented using the following technologies:

- **Backend**: Spring Boot 3.3.6 with Spring Security
- **Frontend**: Thymeleaf templates with HTML5/CSS3/JavaScript
- **Database**: MySQL with H2 for local testing
- **Deployment**: Docker and Docker Compose
- **Build Tool**: Maven

System Architecture

The system follows a layered architecture:

1. **Presentation Layer**: Thymeleaf templates and static resources
2. **Controller Layer**: Spring MVC controllers handling HTTP requests
3. **Service Layer**: Business logic and transaction management
4. **Repository Layer**: Data access using Spring Data JPA
5. **Database Layer**: MySQL database with H2 for testing

Key Features Implemented

- **Multi-role Authentication**: Administrator, Organiser, and User roles
- **Event Management**: Create, read, update, delete events
- **Venue Management**: Manage venues for events
- **Ticket System**: Purchase and manage tickets
- **User Management**: Account creation and profile management
- **Responsive Design**: Mobile-friendly interface