**Final Report - SmartSphere Event Management System**



**Team Members:**

Ibtisam Shahzad 22i-1201

Ibrahim Asim 22i-1330

Haider Zia 22i-1196

[**1. Project Introduction**](#_u5rfndu4zdcn) **2**

[**2. Functional & Non-Functional Requirements**](#_trd8h72svlt7) **2**

[**3. User Stories (15 Total, 5 by Each Member)**](#_f7c1ql4z9aex) **3**

[**4. Product Backlog**](#_mb3mspp1t6ar) **4**

[**5. Sprint 1 and Sprint 2 Backlog**](#_kw03oksqo172) **5**

[**6. Project Plan**](#_ucc2o813h588) **6**

[**7. Architecture Diagram**](#_ngdhyi66eqxu) **7**

[**8. Design (Sprint 3 Items)**](#_bce164a9dg0e) **9**

[Quick Summary:](#_upza3vafc70k) 9

[**9. Actual Implementation Screenshots**](#_ohxqok75lwjq) **10**

[**10. Product Burndown Chart**](#_qpbus3d4okrp) **10**

[**11. Trello Board Screenshots**](#_76ctwsvukvz7) **11**

[**12. Test Cases - Black Box**](#_lu7bk2getlfj) **12**

[**13. Test Cases - White Box**](#_v4hgxibc38xv) **12**

[**14. Work Division Between Group Members**](#_dfxrfwmho09g) **13**

[**15. Lessons Learned**](#_njcwprerszyt) **13**

## **1. Project Introduction**

SmartSphere Event Management System is designed to simplify NASCON event registration, management, scheduling, and participant communication. It caters to multiple user roles including admins, organizers, and participants, enhancing event workflows from creation to feedback collection.

## **2. Functional & Non-Functional Requirements**

**Functional Requirements:**

* User Authentication (Signup/Login)
* Event Creation/Modification/Deletion
* Enrollment and Unenrollment in Events
* Feedback Submission and Notification Handling
* Admin Reporting and Announcements

**Non-Functional Requirements:**

* 99.9% System Uptime
* Responsive Web Design
* Secure Authentication (BCrypt Encryption)
* Scalable Architecture
* Data Privacy and Integrity

## **3. User Stories (15 Total, 5 by Each Member)**

**Ibtisam Shahzad:**

* US-01: User Registration
* US-02: User Login
* US-03: Create Event
* US-04: View Notifications
* US-05: Submit Feedback

**Haider Zia:**

* US-06: Modify Event
* US-07: Delete Event
* US-08: Cancel Enrollment
* US-09: View Feedback Reports
* US-10: Add Announcements

**Ibrahim Asim:**

* US-11: Enroll in Event
* US-12: Update Profile
* US-13: Filter Events
* US-14: Dashboard Access
* US-15: Generate Admin Reports

## **4. Product Backlog**

* The product backlog is a prioritized list of system features and user stories that need to be implemented in the Event Management System. Each item is categorized based on priority: **High**, **Medium**, or **Low**. High-priority items are essential for core functionality, while medium and low-priority items enhance the system but are not immediately critical.

| **ID** | **User Story** | **Priority** |
| --- | --- | --- |
| **US-01** | As a user, I want to register an account so that I can access the system. | **High** |
| **US-02** | As a user, I want to log in so that I can access my dashboard. | **High** |
| **US-03** | As an organizer, I want to create an event so that participants can register for it. | **High** |
| **US-04** | As an organizer, I want to modify an event so that I can update its details. | **Medium** |
| **US-05** | As an organizer, I want to remove an event so that it is no longer available. | **Medium** |
| **US-06** | As a participant, I want to enroll in an event so that I can attend it. | **High** |
| **US-07** | As a participant, I want to cancel my enrollment in an event so that I can withdraw from attending. | **Medium** |
| **US-08** | As a user, I want to send feedback so that I can share my experience about an event. | **High** |
| **US-09** | As a user, I want to send feedback so that I can share my experience about an event. | **Medium** |
| **US-10** | As a user, I want to view notifications so that I stay informed about event updates. | **High** |
| **US-11** | As a user, I want to update my profile so that my information is accurate. | **Medium** |
| **US-12** | As a user, I want to filter events so that I can find relevant events based on my preferences. | **Low** |
| **US-13** | As an organizer, I want to add announcements so that participants receive important event updates. | **Medium** |
| **US-14** | As a user, I want to view feedback reports so that I can see event reviews and ratings. | **Medium** |
| **US-15** | As a participant, I want to cancel my event enrollment so that I can opt out if needed. | **Low** |

## **5. Sprint 1 and Sprint 2 Backlog**

**Sprint 1 Tasks:**

* Register Account (US-01)
* Login (US-02)
* Create Event (US-03)
* Enroll in Event (US-11)
* Submit Feedback (US-05)

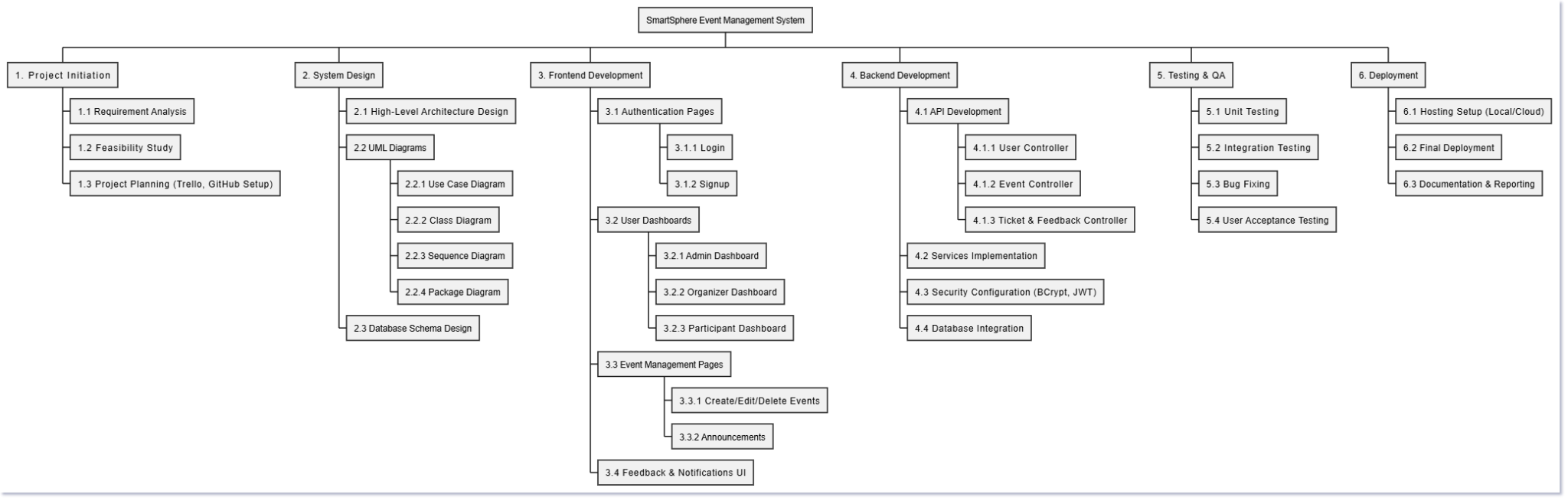
**Sprint 2 Tasks:**

* Modify Event (US-06)
* Delete Event (US-07)
* Cancel Enrollment (US-08)
* View Notifications (US-04)
* Add Announcements (US-10)

## **6. Project Plan**

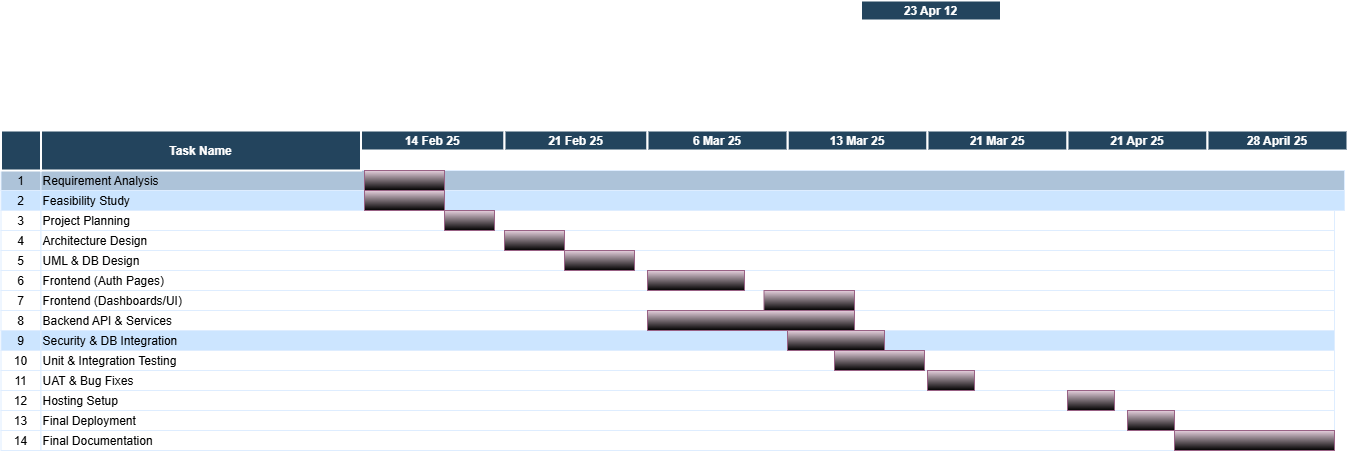
**Work Breakdown Structure (WBS)**

* Project Initiation
* System Design
* Frontend Development
* Backend Development
* Testing & QA
* Deployment



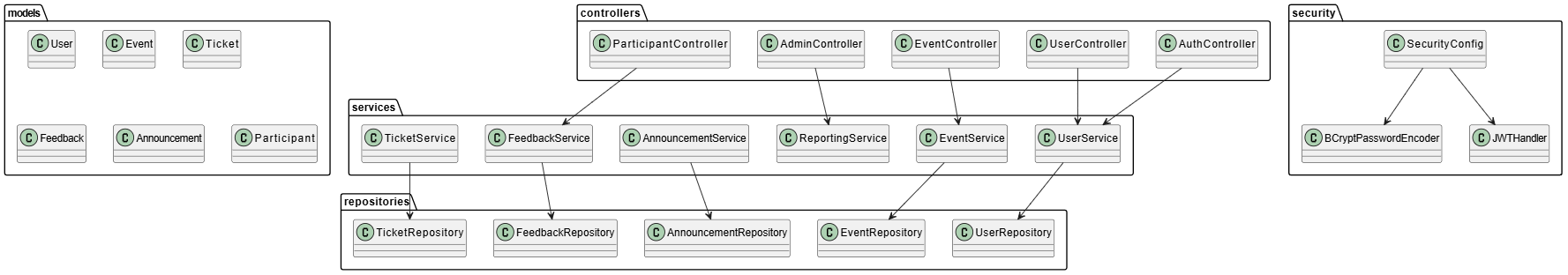
**Gantt Chart**

* 14 Feb - 28 Feb: D1 Completion
* 6 Mar - 23 Mar: D2 Completion
* 21 Apr - 28 Apr: D3 Finalization

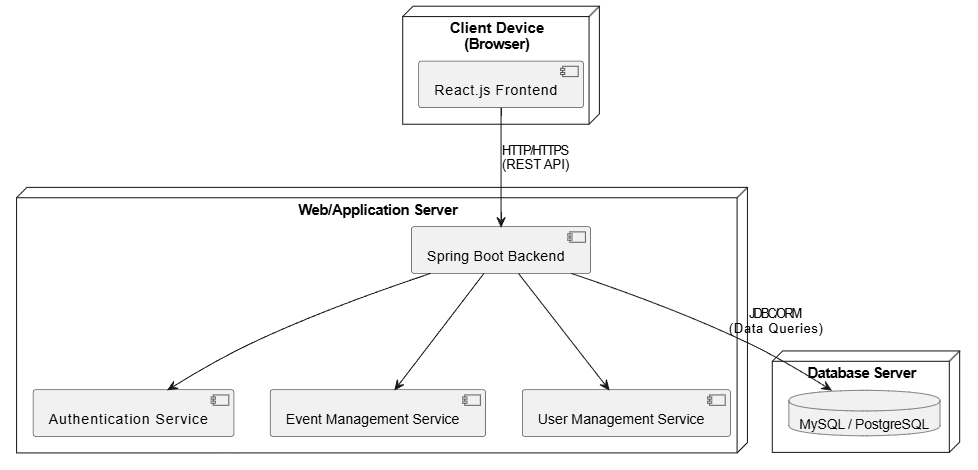


## **7. Architecture Diagram**

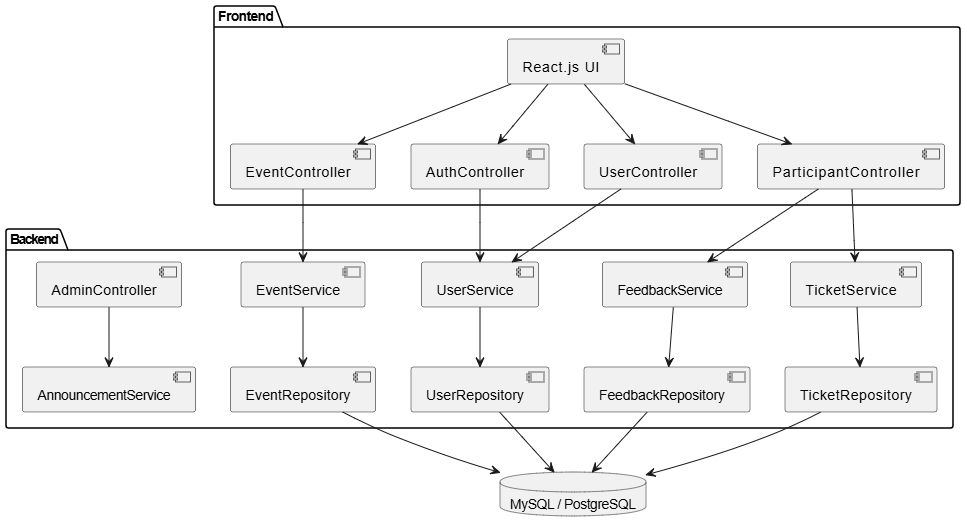
* **Package Diagram:** Authentication, Event Management, User Management, Notification/Feedback, Reporting, Security



* **Deployment Diagram:** Client → Web Server → Application Server → Database Server



* **Component Diagram:** Frontend, Controllers, Services, Repositories, Database



## **8. Design (Sprint 3 Items)**

In Sprint 3, we focused on completing the remaining functionalities that were not addressed in Sprint 1 and Sprint 2.

Key items completed include:

* **Modification and Deletion of Events** through dedicated pages and corresponding backend services.
* **Event Enrollment Cancellation** by participants.
* **Announcement Posting** by admins and organizers to communicate event updates.
* **Notification Management** where participants can view system-generated announcements.
* **Feedback Collection** and display, allowing participants to submit feedback and admins to view reports.

The full frontend was integrated with backend REST APIs ensuring seamless user flows for all three roles: Admin, Organizer, and Participant.

Security aspects, such as password encryption and role-based authorization using Spring Security and JWT tokens, were also finalized in Sprint 3.

Additionally, database models and repositories were reviewed and finalized to support all data operations in the system.

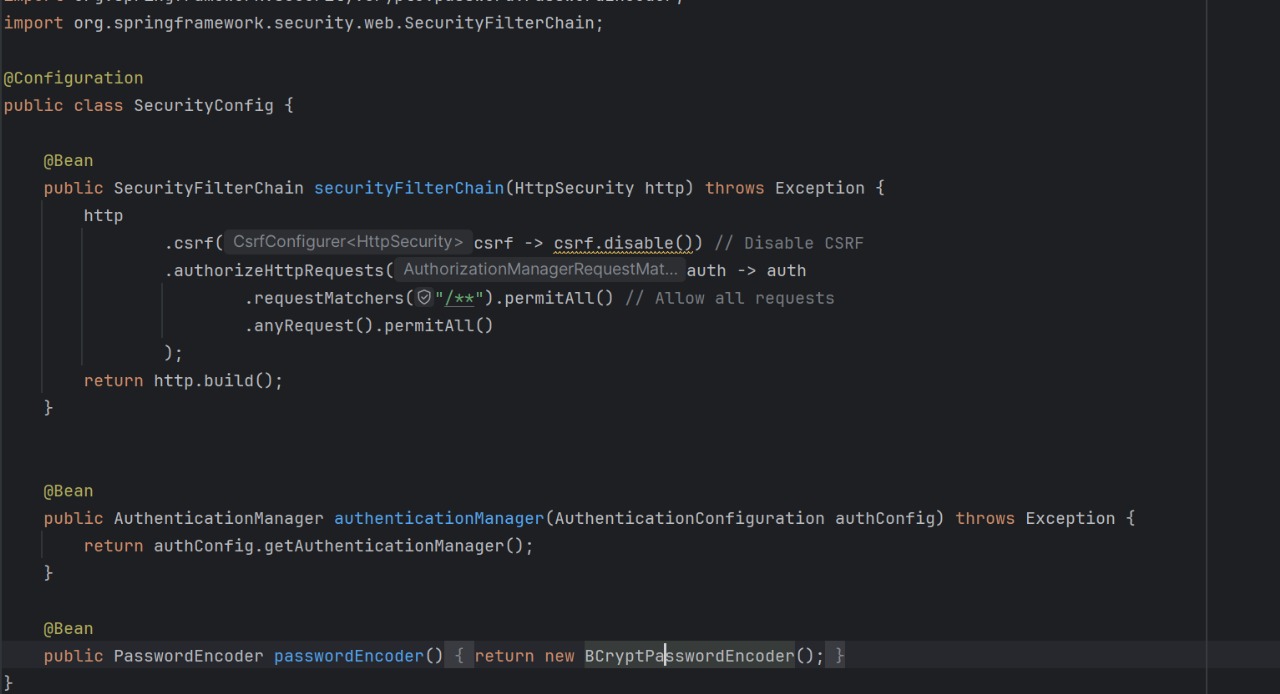
Admins were provided with detailed event feedback reporting functionality to enhance event management efficiency.

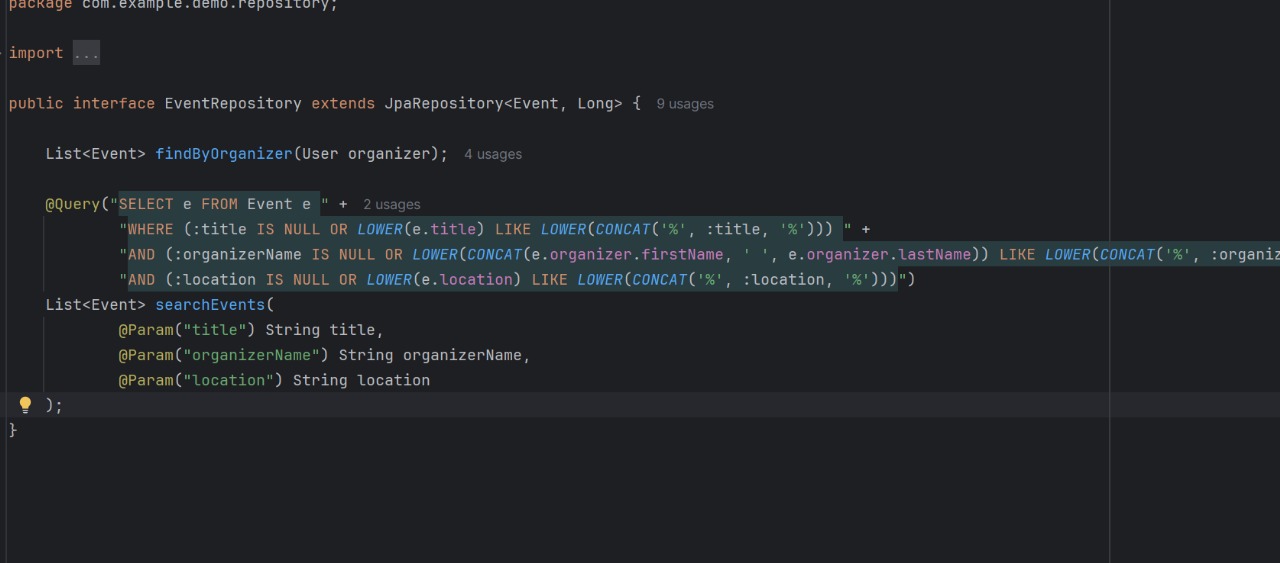
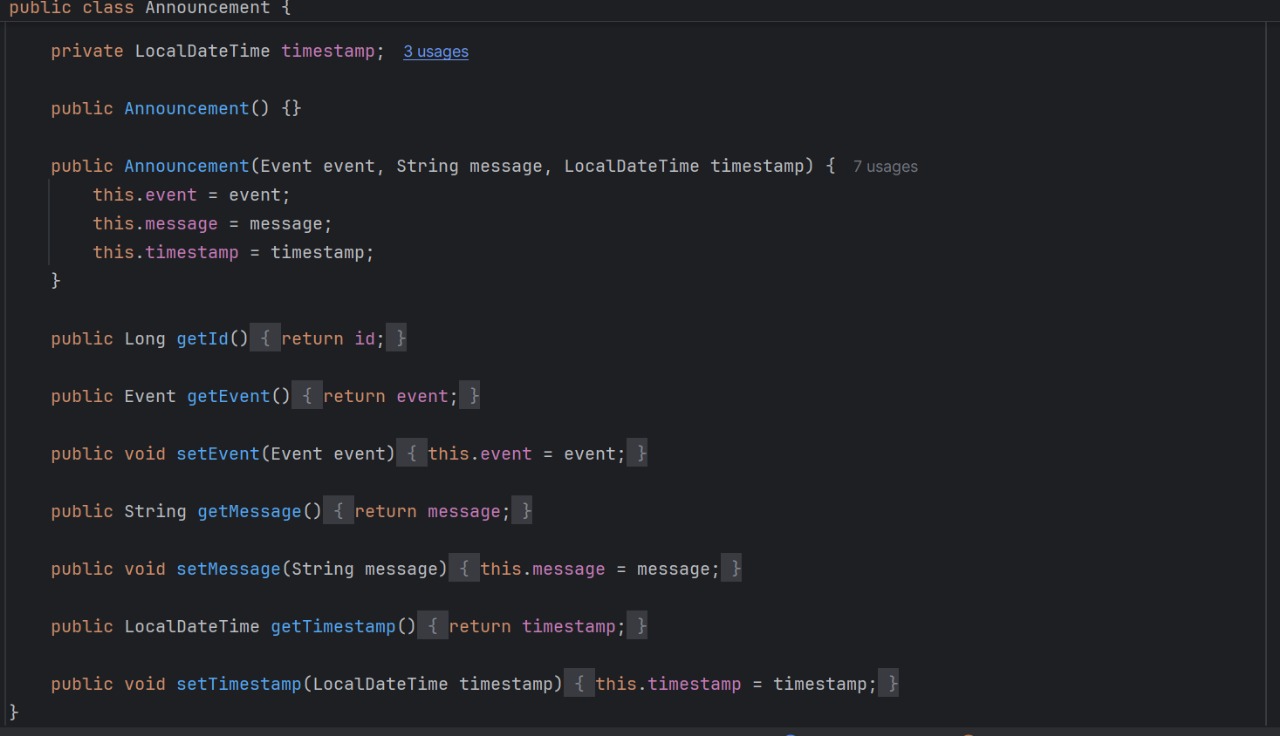
### Quick Summary:

* **User stories completed**: Modify, Delete Event, Feedback, Announcements, Cancel Enrollment.
* **Frontend finalized**: All UI pages connected to APIs.
* **Backend finalized**: REST APIs completed.
* **Security finalized**: Spring Security + JWT setup.
* **Database models finalized**: Entities complete and connected.
* **Reporting**: Admins can view reports.

## **9. Actual Implementation Screenshots**

**Backend:**

****

****

**Login:**



**SignUp:**

****

**Admin-Dashboard:**

****

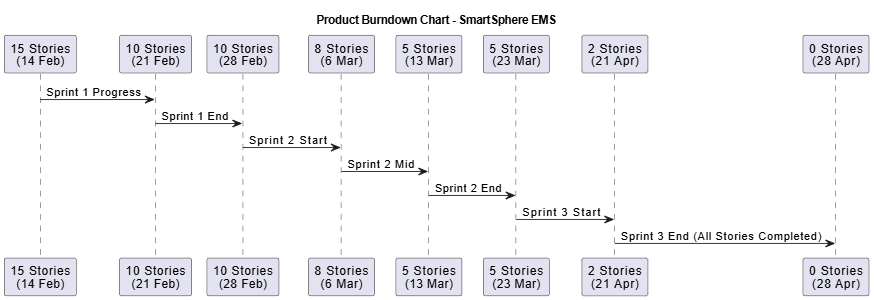
**Participant-Dashboard:**

****

**Organizer-Dashboard:**

****

## **10. Product Burndown Chart**

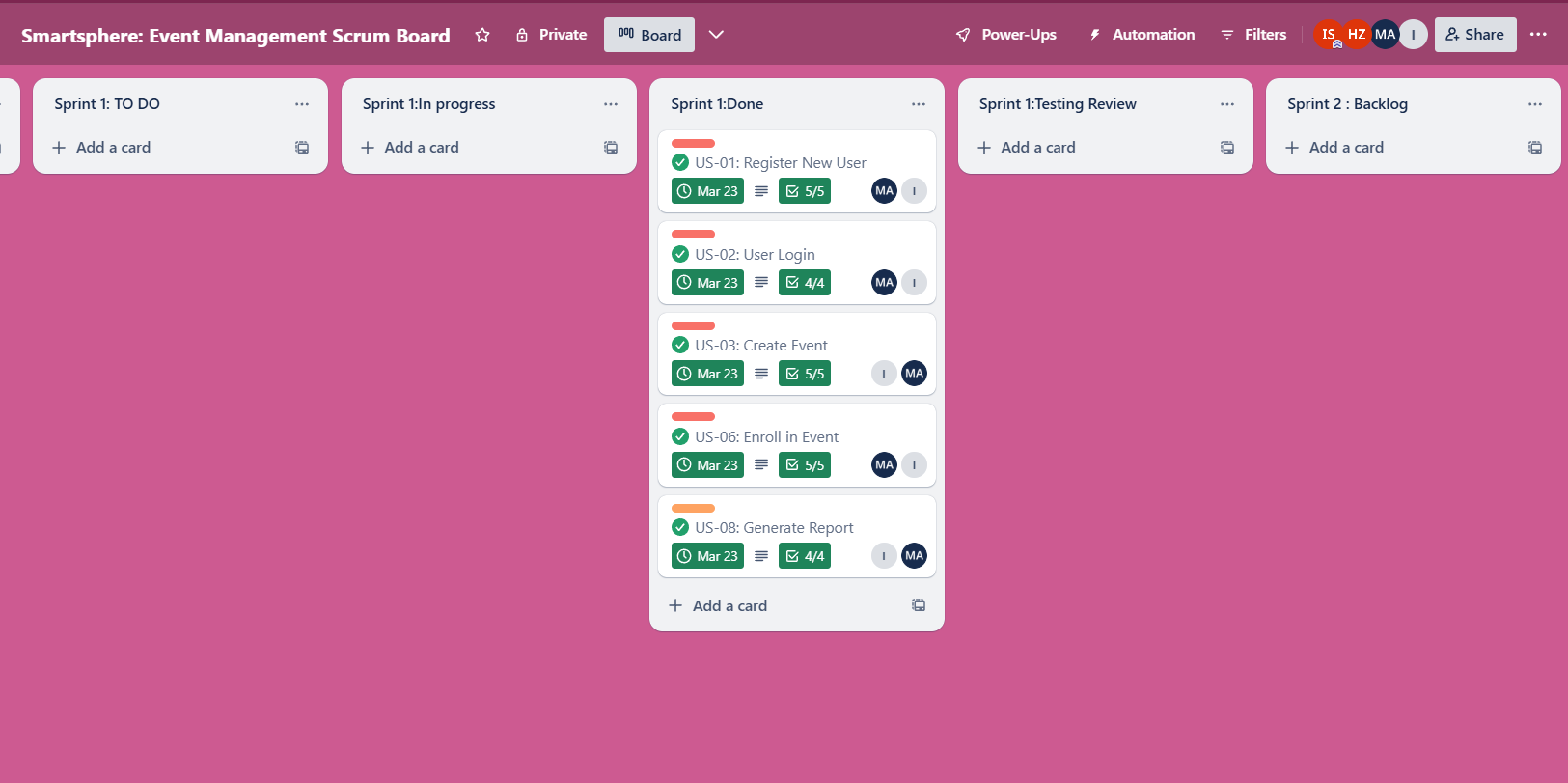


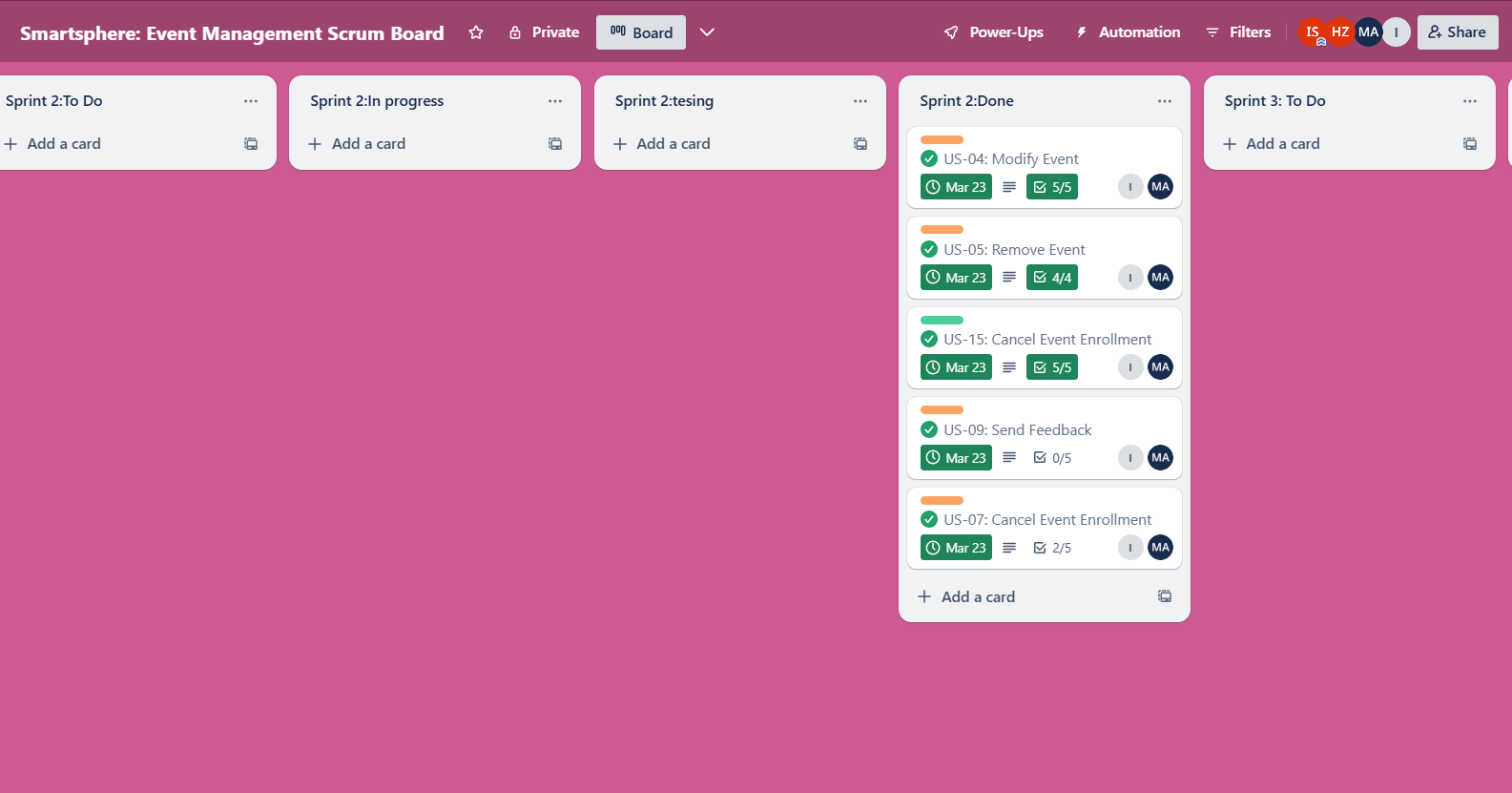
## 

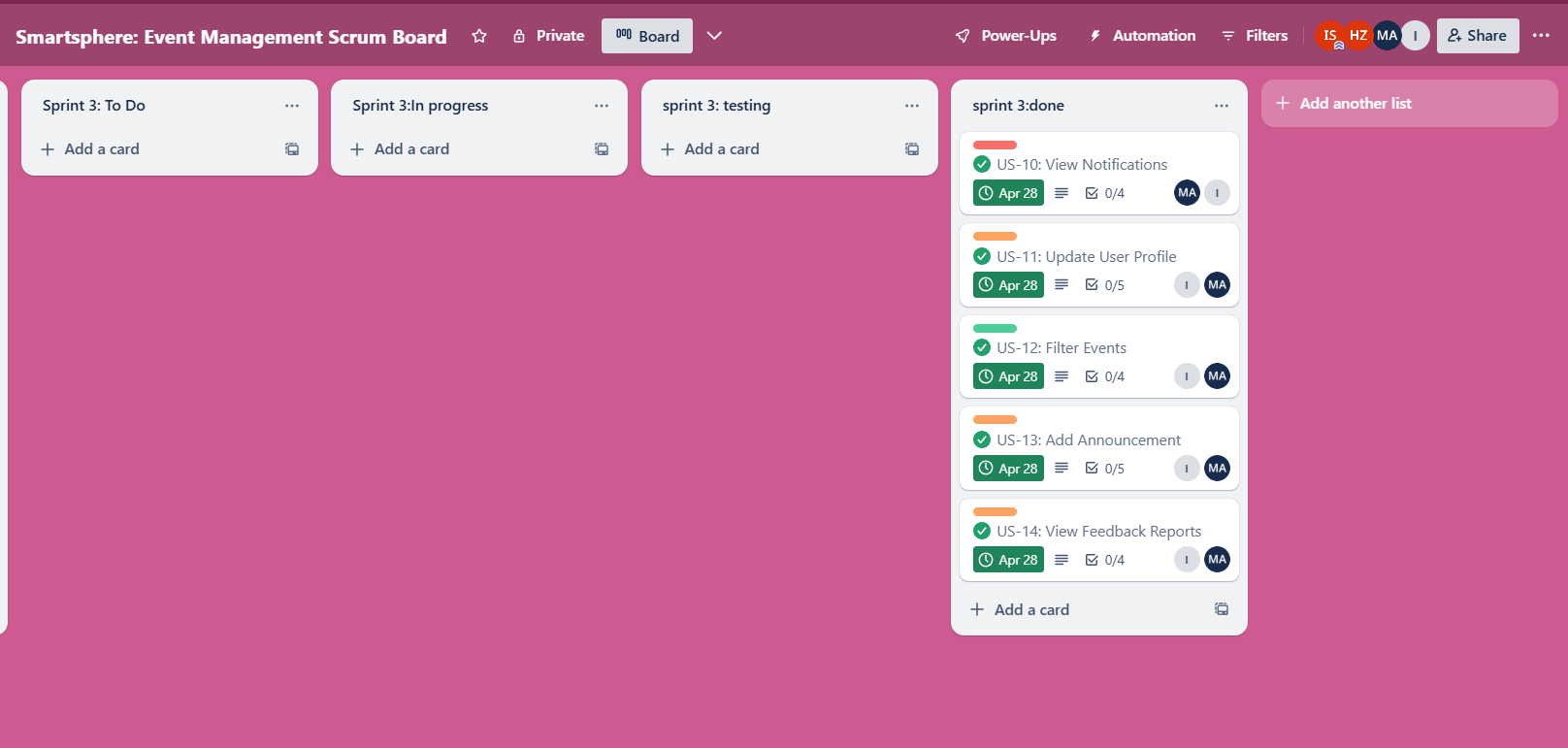
## 

## 

## **11. Trello Board Screenshots**







## 

## 

## 

## 

## 

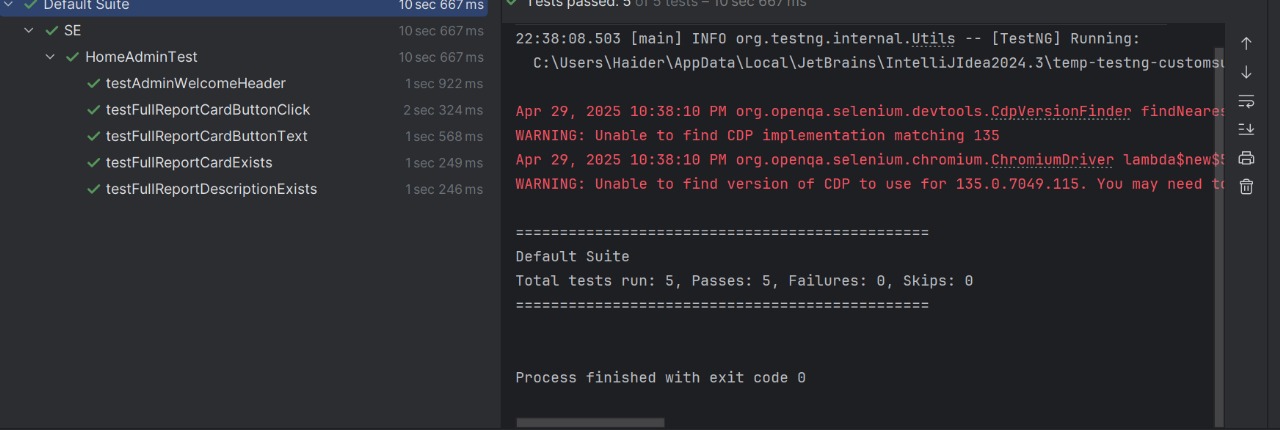
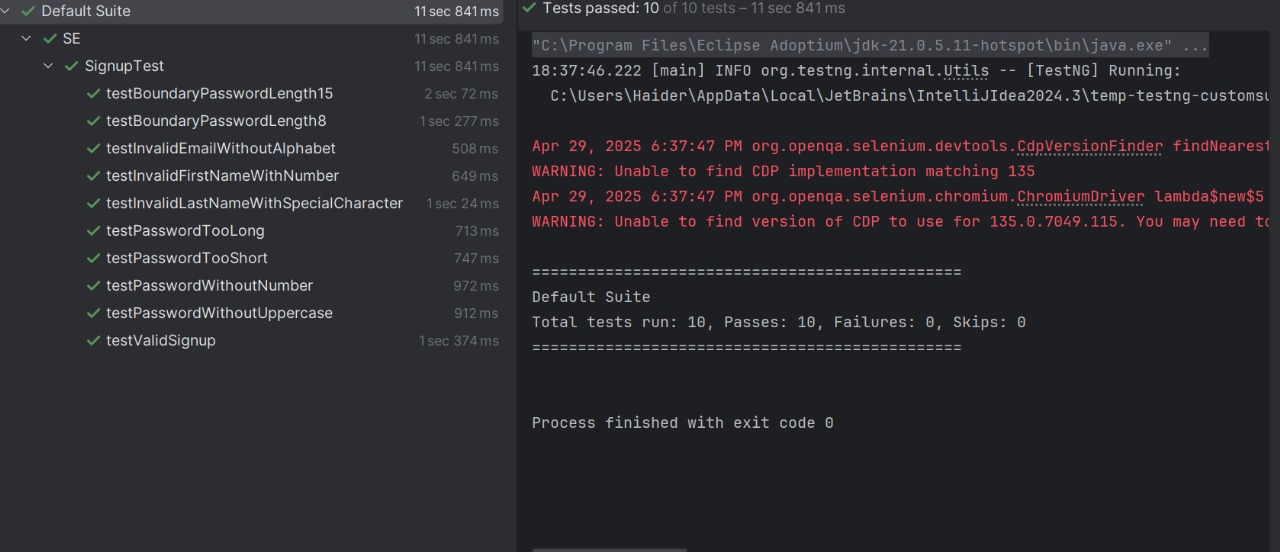
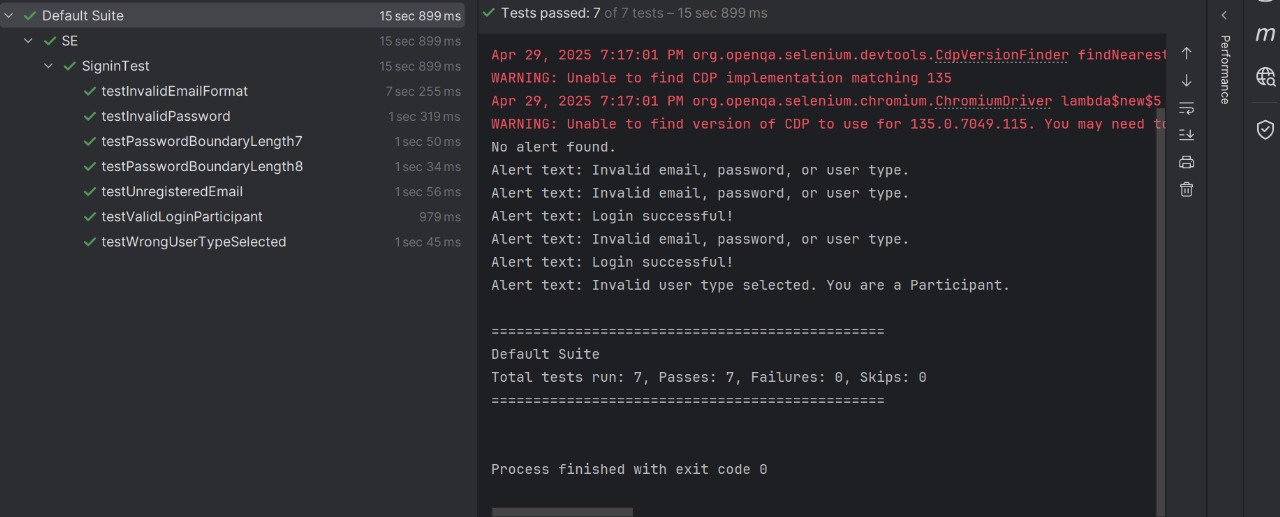
## 

## 

## 

## 

## **12. Test Cases - Black Box**



## 

## 

## **13. Test Cases - White Box**

**What Is Covered Well**:

**Controller Layer (controller):**

Achieved 100% coverage. All REST endpoints were tested through unit tests, ensuring every path is executed.

**Model Layer (models/):**

Achieved 100% coverage. Data classes like Event, User, etc., were fully instantiated and used within service/controller tests.

**Repository Layer (repository/):**

Interface-based with Spring Data JPA – considered covered when interacting through services.

**Security Layer (security/):**

100% coverage on custom classes (e.g., filters, configs).

**Main Application Class (DemoApplication):**

Fully covered during application context loading.

**Core Logic (EventService & AnnouncementService):**

Main methods like createEvent, getAllEvents, and searchEvents were invoked and validated.

**What Is Not Covered and Why:**

**Some Service Methods:**

Coverage in service/ layer is around 33%–40% in some files due to incomplete testing of:

**Exception branches (e.g., invalid organizer cases).**

Optional-based logic where isEmpty() or isPresent() was not triggered for all paths.

update and delete operations were partially tested or skipped.

**Branch Coverage Gaps:**

if/else and exception-throwing branches need more dedicated negative test cases to boost coverage.

**Third-party and Auto-generated Code:**

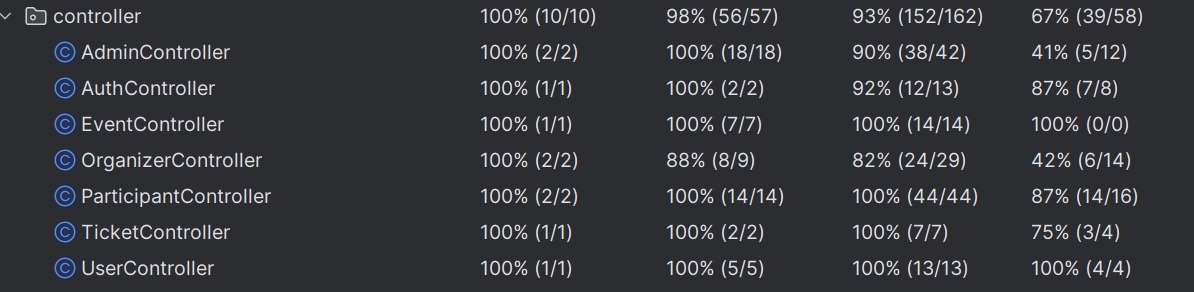
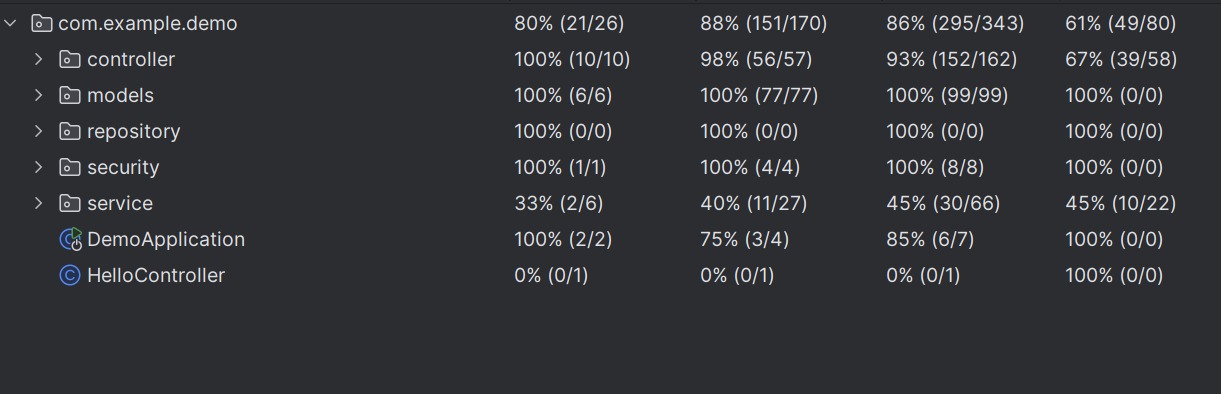
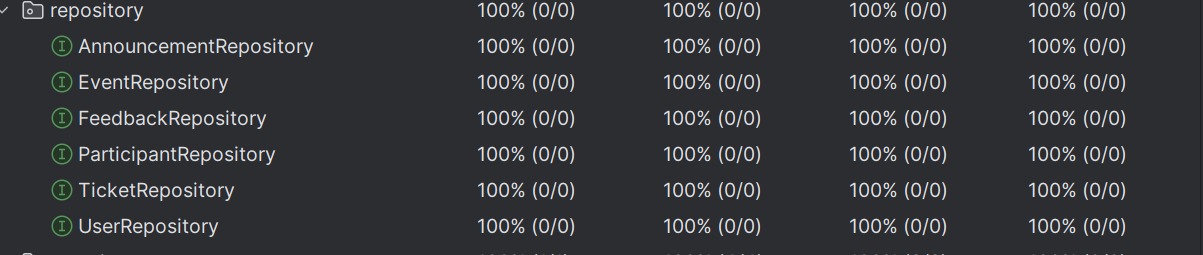
External frameworks (Spring Boot, Hibernate, JWT, etc.) are not included in coverage metrics.

These are assumed to work as intended and don’t need direct testing.

**Unreachable or Rare Execution Paths:**

Paths that rely on specific user states, request combinations, or database errors weren’t feasible to replicate in unit tests alone.

## 



## **14. Work Division Between Group Members**

* **Ibtisam Shahzad:** Requirement Analysis, Scrum Management, Documentation
* **Haider Zia:**, Testing, System Architecture , Frontend UI
* **Ibrahim Asim:** Backend API Development, Integration, Database Management

## **15. Lessons Learned**

* Gained real-world experience of Agile methodology.
* Improved collaboration through Trello and GitHub.
* Understood challenges of full-stack development.
* Learned project management, time estimation, and risk handling.