Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core backend architecture for **SportifyHub - Sports Event Management Backend**. The system provides RESTful APIs for managing sports events, teams, and registrations, with role-based access, event scheduling, and notifications. Technologies: Java, Spring Boot, PostgreSQL, Docker, Maven.

---

1. System Components

Component	Description / Responsibility
API Layer	REST controllers for events, teams, registrations
Service Layer	Business logic, validation, orchestration
Repository Layer	Data access using Spring Data JPA
Security Module	Role-based access control (RBAC)
Notification Module	Event-triggered notifications (email/placeholder)
Persistence	PostgreSQL database
Deployment	Dockerized application

---

2. Class/Interface Overview

Class/Interface	Responsibility / Key Methods
`EventController`	API endpoints for event CRUD
`TeamController`	API endpoints for team CRUD
`RegistrationController`	API endpoints for registration actions
`EventService`	`createEvent()`, `updateEvent()`, `scheduleEvent()`
`TeamService`	`createTeam()`, `addMember()`
`RegistrationService`	`registerUser()`, `cancelRegistration()`
`UserService`	User management, role assignment
`NotificationService`	`sendNotification()`
`EventRepository`	JPA repository for Event entity
`TeamRepository`	JPA repository for Team entity
`RegistrationRepository`	JPA repository for Registration entity
`UserRepository`	JPA repository for User entity

`SecurityConfig`	Spring Security configuration

Relationships:

- Controllers use Services
- Services use Repositories
- SecurityConfig applies RBAC to endpoints

---

3. Data Structure Overview

Entity	Key Attributes
`User`	id, username, password, email, role (ADMIN, ORGANIZER, USER)
`Team`	id, name, members (List), createdBy
`Event`	id, name, description, startTime, endTime, location, teams
`Registration`	id, user, event, status (REGISTERED, CANCELLED), timestamp

---

4. Algorithms / Logic

Event Registration Flow (Pseudocode):

```
if (user.hasRole(USER) && event.isOpenForRegistration()) {
    if (!registrationRepository.existsByUserAndEvent(user, event)) {
        registrationRepository.save(new Registration(user, event, REGISTERED));
        notificationService.sendNotification(user, "Registration successful");
    } else {
        throw new AlreadyRegisteredException();
    }
} else {
    throw new RegistrationNotAllowedException();
}
```

Role-Based Access (Summary):

- Endpoints are secured via Spring Security.
- Only ADMIN/ORGANIZER can create/update events/teams.
- USERS can register/cancel for events.

---

5. Error Handling

Scenario	Handling Approach
Invalid input/data	Return 400 Bad Request with error message
Unauthorized access	Return 401/403 with error message
Entity not found	Return 404 Not Found

Duplicate registration	Return 409 Conflict
Database errors	Return 500 Internal Server Error
Notification failure	Log error, return 202 Accepted (non-blocking)

End of Document