High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **SportifyHub - Sports Event Management Backend**. The system is a Spring Boot API designed to manage sports events, teams, and registrations, with role-based access, event scheduling, and notification features. The backend uses PostgreSQL for data persistence, Docker for deployment, and Maven for project management.

---

1. System Architecture Overview

Architecture Description:
SportifyHub follows a layered architecture with RESTful API endpoints, service and business logic layers, a data access layer, and integration with external services (e.g., notifications). The system is containerized for deployment.

Module/Component	Role/Responsibility
API Layer	Exposes REST endpoints for clients
Authentication Module	Manages user authentication and role-based access
Event Management	Handles CRUD for sports events and scheduling
Team Management	Manages teams and their members
Registration Module	Manages event registrations
Notification Service	Sends notifications (e.g., email, push)
Persistence Layer	Interacts with PostgreSQL database
Docker Deployment	Containerizes the application for deployment

---

2. Component Interactions

Interaction Sequence
1. Client sends API request (e.g., create event, register team)
2. API Layer authenticates request via Authentication Module
3. Request routed to appropriate Service (Event, Team, Registration)
4. Service performs business logic, interacts with Persistence Layer
5. If applicable, Notification Service is triggered (e.g., on successful registration)
6. Response returned to client

---

3. Data Flow Overview

Data Flow Description
User data and credentials flow through Authentication Module for access control
Event, team, and registration data flow between API Layer, Service Layer, and Persistence Layer
Notification data flows from Service Layer to Notification Service, then to external channels
All persistent data stored and retrieved from PostgreSQL

---

4. Technology Stack

Layer/Function	Technology/Framework
Backend Framework	Java, Spring Boot
Database	PostgreSQL
Build/Dependency Mgmt	Maven
Containerization	Docker
Authentication	Spring Security (JWT/roles)
Notifications	Email/Push (pluggable)

---

5. Scalability & Reliability

- **Scalability:**
  - Stateless API enables horizontal scaling via Docker containers.
  - Database can be scaled vertically or via managed PostgreSQL services.
- **Reliability & Security:**
  - Role-based access control ensures secure operations.
  - Input validation and error handling at API and service layers.
  - Dockerized deployment supports consistent environments and easy rollbacks.

End of Document