Product Requirements & Specification Document

Project Name

SportifyHub - Sports Event Management Backend

Description

Develop a Spring Boot REST API for managing sports events, teams, and registrations. The system will support role-based access, event scheduling, and notifications. PostgreSQL will be used for data persistence, Docker for deployment, and Maven for project management.

---

1. Objectives

- Enable creation, management, and scheduling of sports events.
- Support team and participant registration workflows.
- Implement role-based access control (RBAC).
- Provide event notifications.
- Ensure scalable, maintainable, and secure backend architecture.

---

2. Stakeholders

Role	Responsibility
Product Owner	Requirements, prioritization
Developers	Implementation, testing
QA Engineers	Testing, validation
End Users	Event organizers, team managers, participants

---

3. Functional Requirements

3.1 User & Role Management

Feature	Description
User Registration	Users can register and authenticate
Roles	Admin, Organizer, Team Manager, Participant
RBAC	Access control based on user roles

3.2 Event Management

Feature	Description

Create/Edit/Delete	Organizers can manage events
Event Details	Name, description, date, location, status
Scheduling	Prevent event time conflicts

3.3 Team & Registration Management

Feature	Description
Team CRUD	Team managers can manage teams
Team Registration	Teams can register for events
Participant Management	Add/remove participants to/from teams

3.4 Notifications

Feature	Description
Event Notifications	Email/Push notifications for updates
Registration Alerts	Notify users on registration status

---

4. Non-Functional Requirements

Requirement	Specification
Performance	< 500ms API response time (95th percentile)
Security	JWT authentication, RBAC, input validation
Scalability	Support 10,000+ users/events
Maintainability	Modular code, RESTful standards
Deployment	Dockerized, environment configs via Docker
Documentation	OpenAPI/Swagger for API docs

---

5. Technical Specifications

5.1 Architecture

- **Backend:** Java 17+, Spring Boot (REST API)
- **Database:** PostgreSQL
- **Build/Dependency:** Maven
- **Containerization:** Docker

5.2 Key Entities

Entity	Attributes
User	id, name, email, password, role
Event	id, name, description, date, location, status

Team	id, name, manager_id, members
Registration	id, event_id, team_id, status, timestamp
Notification	id, user_id, type, message, sent_at

5.3 API Endpoints (Sample)

```
POST   /api/auth/register         // User registration
POST   /api/auth/login            // User login
GET    /api/events                // List events
POST   /api/events                // Create event (Organizer/Admin)
PUT    /api/events/{id}           // Update event
DELETE /api/events/{id}           // Delete event
POST   /api/teams                 // Create team
POST   /api/events/{id}/register  // Register team for event
GET    /api/notifications         // List user notifications
```

5.4 Security

- JWT-based authentication
- Role-based authorization at endpoint level
- Password hashing (BCrypt)

5.5 Deployment

- Dockerfile for backend service
- Docker Compose for local development (PostgreSQL + API)
- Environment variables for configuration

---

6. Acceptance Criteria

ID	Criteria
AC1	Users can register, login, and access endpoints per role
AC2	Organizers can create, edit, and delete events
AC3	Teams can register for events and manage participants
AC4	Notifications are sent on event updates and registrations
AC5	API is documented and deployable via Docker

---

7. Out of Scope

- Frontend/UI development
- Payment processing
- Advanced analytics

---

8. Milestones

Milestone	Description	Target Date
Project Setup	Repo, Docker, DB, CI/CD	Week 1
User & Role Management	Auth, RBAC	Week 2
Event & Team Management	CRUD, registration	Week 3
Notifications	Email/Push integration	Week 4
Testing & Documentation	Unit tests, Swagger	Week 5
Deployment	Docker Compose, staging	Week 6

---

9. Risks & Mitigations

Risk	Mitigation
Data consistency	Use DB transactions
Unauthorized access	Strict RBAC, JWT validation
Notification delivery	Retry logic, logging

---

10. References

- Spring Boot Documentation
- PostgreSQL Documentation
- Docker Documentation
- Maven Documentation
