UNIT: (Capstone CPRO306)

Timely - Online Sports Events Management System

FINAL SRS REPORT

Team Members		
No.	**Name**	**SID**
1	**Nitesh Sapkota**	**K230702**
2	**Sandesh Rimal**	**K231587**
3	**Aayush KC**	**K231801**
4	**Rohan Rai**	**K231128**
5	**Ujjwal Lamsal**	**K232084**

Table of Contents

1. Project Description 3
1.1. Business Case 3
1.2. Purpose and Objectives 3
1.3. Detailed Scope 4
1.4. Proposed System Design Specifications 4
1.5. Methodology 5
2. System Analysis and Requirements 7
2.1. Software and Hardware Requirements 7
2.2 Functional and Non-Functional Requirements 8
2.3 User Requirements 9
2.3.1 Actors 9
2.3.2 Use Case List (by Actors) 9
Actor/Role: 15
Organizer 15
3. Database Design 36
3.1 Entity–Relationship Design (ERD) 36
3.2 Data Flow Diagram (DFD) 37
3.3 Security Implementation 39
3.3.1 Overview (CIA and privacy-by-design) 39
3.3.2 Authentication & Authorization 39
3.3.3 Security of Transport & Session 40
3.3.4 Data Protection-in-Rest 40
4. User Interface Design 41
4.1 User Interface Storyboard 41
4.2 Input Data Forms 44
4.3 Output Report Forms 45
5. Test Plan & System Implementation 45
5.1 System Implementation Plan 45
5.2. Test Plan 47
References 49

1. Project Description

1.1. Business Case

Sports event management today is highly fragmented and human in order with the dependence on spreadsheets, email, and phone communications. This non-collective procedure would be the cause of scheduling collisions, lost or repeated certifications, and delayed correspondence between the organizers of the event, the attendants and the other stakeholders. These inefficiencies lead to a bad experience by users, excessive administrative work and lost revenue due to booking without profits and overbooking. The Sports Events committee has identified these bottlenecks and is keen to deploy an online sports events management system (OESM) which will concentrate all actions that concerns the organization and participation in sports events. Such a system will enhance the efficiency of operations, lessen errors and offer a smooth and digital experience to organizers and participants. Back on the same line is the fact that the application of OESM falls in line with overall objectives of digital transformation, operational excellence and improved stakeholder engagement

1.2. Purpose and Objectives

The main aim of the given project is the development and creation of a scalable, secure, and simple web application that supports the whole cycle of managing sports events. This entails planning, registration, scheduling, and communication of the events. The system is expected to benefit the event organizers because they may have convenient ways of managing the event logistics and factual information about participants and the participants enjoy convenience in accessing the event information and enrolling in it.

The major goals are:

- To establish an all-inclusive platform that will facilitate a number of sports disciplines with convenience in scheduling.
- To afford safe user registration and authentication services, guarding delicate user personal and payment information.
- Provide instant messaging and notification features to decrease no-shows and provide timely information as a way to spread event information
- To provide an opportunity to present a full range of the reporting and analytics capabilities that will assist the organizers in monitoring the participation trends and the performance.

- Ensuring the responsiveness and availability of the system in other devices through a responsive design with the ability to accommodate the desktops, tablets, and mobile phones.

1.3. Detailed Scope

The OSEMS scope consists of the core modules, which are aimed to address the needs of various sets of users.

- **User Registration and authentication**: Enable simple authentication of the user and the registration of the users where the roles include: Organizer or a participant. Passwords shall be hashed and securely stored to be confidential.
- **Event Management**: They can make in depth event listings, event descriptions on a particular event, and the type of event . Events may be edited or canceled and participants be notified, provided that the organizer has the option of doing so.
- **Participation registration and management**: Event searchable by participants: The participants may use filters to search the events depending on the type of the sport, place, date, and the level of the skills. They will be able to sign up to events, have the ability to see their signed up events as well as having automated reminders.
- **Appointment and Reminder**: The System will be calendar compatible and will also give automated reminders via email or sms messages on confirmation, reminders and change of an event.
- **Reporting and Analytics**: Create configurable reports detailing the demographic of participants, registrant trends, event attendance and feedback to continually improve
- **Security and access control**: Role based access and control guarantees accessibility to user features based on such needs to a particular role. The Sensitive data will be encrypted, and communication secures employing the https protocols.
- **Payment Processing(optional)**: Dynamic and safe connection with the payment gateways to receive money regarding the conducted events, in addition to handling reuters in the events of such need.

1.4. Proposed System Design Specifications

Presentation Layer: A server run by a mobile-friendly web interface built on React.js, which qualifies both organizers and participants in a way that is user-friendly and efficient in nature. Role-based dashboards will be offered within interfaces, which will allow users an overview of approaching events, pending works, and major activities (e.g., registrations, purchase of tickets, approve).

Application level: The backend will be operated by the Django REST Framework, including dealing with the main business functions that include event processes, user authorization, and registration, payment, notifications, and reports. The layer provides modularity, scalability, and safe API-based communication with the frontend.

Data layer: User profile, event data, registration, payment, and delivery of results would be stored and managed with the help of an effective relational database management system (PostgreSQL). The specific schema will be illustrated as an Entity-Relationship Diagram (ERDs) to achieve the data consistency, normalization and scaling.

Security measures: The sensitive data including the payment information and personal information will be encrypted with AES standards of encryption whenever data is resting and TLS/SSL when data is in transit. The Role-Based Access Control (RBAC) will limit access to sensitive procedures based on user roles (sender may approve participants or manage financial reports, only admins can do it). In critical operations, audit logs will be kept.

Notifications System: There will be an inbuilt notification system that will be used to send real time updates through email and SMS to include confirmations of the event; venue, time changes; reminders and announcements. This will keep the participants and organizers in line with the information at all times.

3rd party Integrations: The system will also have an integration with payment gateways (Stripe, PayPal) to handle secure money transactions over the internet and may as well use calendar applications (Google Calendar, Outlook) in order to enable users to synchronize their event calendars with ease.

1.5. Methodology

Agile development methodology will be used in the project in order to enable the project to have an iterative design, feedback and a flexibility to meet any changing requirements. The major stages are:

- **Requirements Analysis**: integrate functions and non functions preparation by conducting stakeholder interviews, surveys, workshops and capturing their requirements. Write up user stories and acceptances criteria.
- **Design**: Develop UI wireframes and prototypes. Draw diagrams such as architecture, and ER diagrams. Review to be vetted by stakeholders.
- **Implementation**: build the system in sprints addressing modules such as user management, event scheduling, and notification services as the core functionality. There will be regular code reviews and integration tests.

- **Testing**: A thorough testing will be conducted with unit testing , integration testing, security testing(including penetration test), and user acceptance testing(UAT) by involving real users.
- **Deployment**: Put the system to use in a secure cloud hosting environment, which is scalable and reliable. Make data migration according to necessity.
- **Maintenance and Support**: After deployment< offer and ongoing maintenance and support such as bug fixing, performance optimization, and feature improvement depending on the feedback of the users. Introduce a help desk to reply to questions of users.

2. System Analysis and Requirements

2.1. Software and Hardware Requirements

Software Tools

- Frontend: React.js and responsive design to make it compatible with other devices.
- Backend: API services and authentication via using Django REST Framework.
- Database: PostgreSQL to store and manage data in relations.
- Software (Development): Windows 10 or 11 / macOS Ventura or newer.
- Operating System (Server): Linux (Ubuntu 22.04 LTS recommended) to be deployed.

Development Tools:

- IDE Visual Studio Code / PyCharm
- Version control on GitHub
- Postman/ReqBin to test API
- UI wireframes and prototypes Figma
- Hosting: Hosted over the cloud using AWS or Render, and SSL enabled.
- Third-Party Integrations: Stripe/ PayPal (payment gateways), Twilio / SendGrid (sms / email notifications), Google calendar / Outlook (event sync).

Hardware Requirements

Development Machines:

- Computer processor: Intel i5 (minimum) Intel i7 (recommended)
- RAM: 8GB(min.), 16GB (recommended)
- Memory storage: 256GB SSD (the minimum)

Server Requirements:

- CPU: Quad core
- RAM: 16GB
- Storage: 200GB SSD
- Bandwidth: 100 Mbps at least and supportive of scalability

2.2 Functional and Non-Functional Requirements

Functional Requirements (FR):

- Allocate a user registration and / or login using email activation.
- Offer role based access control (Athletes, Coaches, Organizers, Spectators, Admins).
- Give creators and managers an opportunity to create and administer events (name, date, location, and eligibility).
- Allows athletes / teams to enroll and post the necessary documentation.
- Take online payments with receipts.
- Permit scheduling and creation of fixtures (round-robin/ knockout).
- Post in real time matches, leader boards, and statistics.
- Allow viewers to buy QR coded digital tickets.
- Make confirmations, reminders and announcements through email/SMS notifications.
- Provide an administration interface to enable users, events, venues management and reports.

Non-Functional Requirements

- Performance: 200+ concurrent users 2s response time.
- Security: Enforce bcrypt password hashing, AES-256 data encryption, HTTPS and RBAC.
- Availability: Maintain 99.5 percent access by utilising recovery and failure recovery systems.
- Reliability: Ensure there is data integrity in registration, payments and results.
- Usability: Present agile user interfaces responding at any device.
- Compatibility: Functionality on the current Internet browsers (Chrome, Firefox, Safari, Edge).
- Scalability: Facilitate the expansion of features in modules, and database replication.
- Maintainability: Adhere to best codes (PEP-8, React best practices), in-line documentation and CI/CD pipelines.
- Compliance: Comply with the Australian Privacy Principles (APPs) and GDPR when it applies.

2.3 User Requirements

2.3.1 Actors

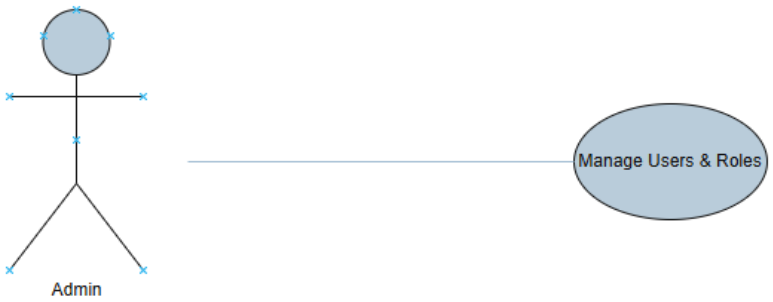
- Admin – manages users, roles, events, venues, payments, reports.
- Organizer – creates/edits events, scheduling/fixtures, approves entries.
- Athlete/Team – registers for events, uploads docs, views schedules/results.
- Coach/Manager – manages team roster/entries, views schedules/results.
- Spectator – browses events, buys tickets, views results/news.
- Payment Gateway (external system) – processes payments, returns status.
- Notification Service (external system) – sends email/SMS.

2.3.2 Use Case List (by Actors)

Admin

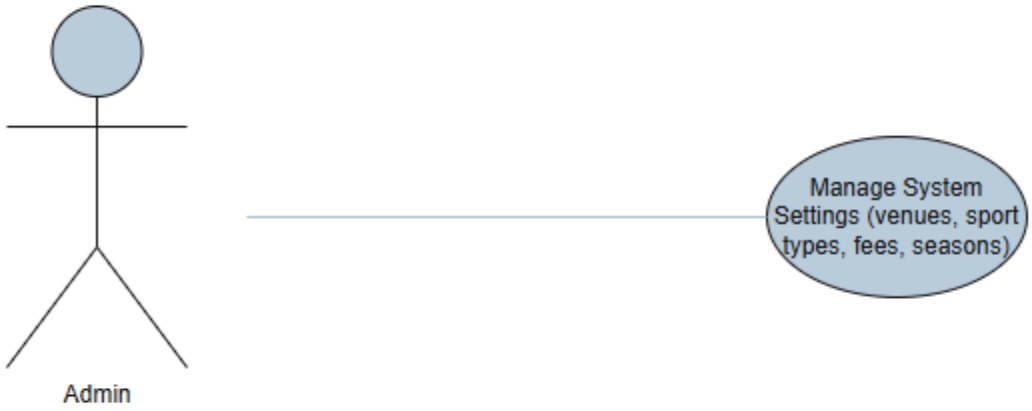
- Manage Users & Roles

Manage Users & Roles Use Case	
Name:	Manage Users & Roles
Actor/Role:	Admin
Description:	Demonstrate how to handle and manage different users and their roles
Successful completion:	Search/create users. View profile and current roles. Assign/modify roles (Athlete/Coach/Organizer/Admin). Activate/deactivate account; trigger password reset if needed. Save changes. The system writes audit logs, invalidates affected sessions, and (optionally) emails users.
Alternative:	Email already exists → show conflict, offer merge/cancel.
Precondition:	Admin authenticated with “Admin” role.
Postcondition:	User record updated, role mappings persisted, audit entry stored.
Assumptions:	User wants his password reset.



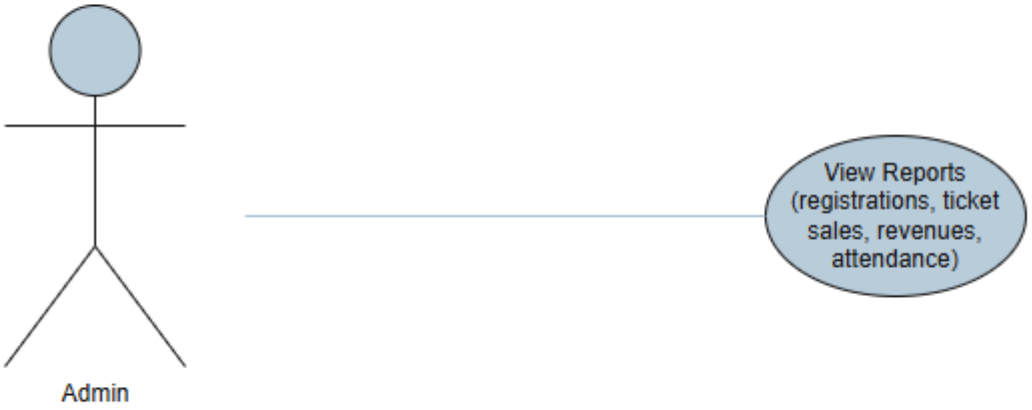
- Manage System Settings

Manage System Settings Use Case	
Name:	Manage System Settings
Actor/Role:	Admin
Description:	Maintain reference data, venues, fees, and platform options
Successful completion:	1. Open Settings. 2. Update sports/divisions/seasons and eligibility rules. 3. Add/edit venues (capacity, facilities, availability). 4. Configure fees, refund windows, ticket rules. 5. Update content pages (About/FAQs) and feature toggles. 6. Save, system validates and applies changes.
Alternative:	Venue/date conflict detected → system highlights clash and suggests alternatives.
Precondition:	Admin should be logged in.
Postcondition:	New settings version stored; changes active; audit entry created.
Assumptions:	Event fee need to be changed.



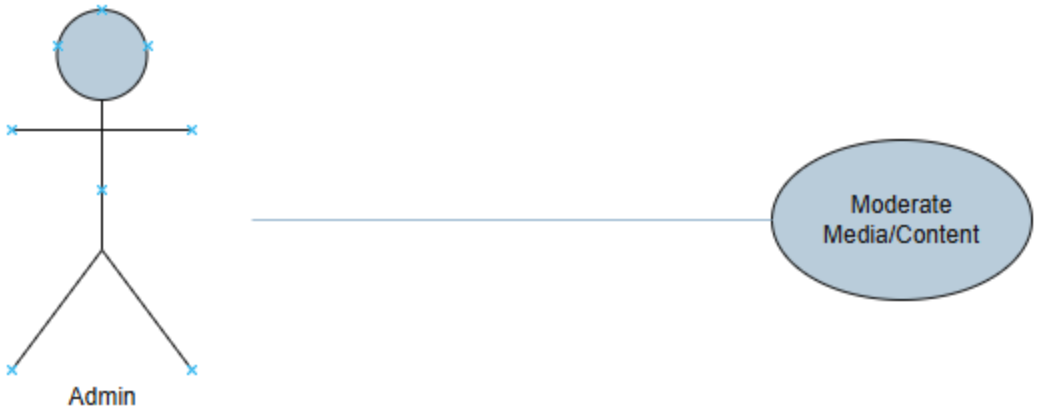
- View Reports

View Reports Use Case	
Name:	View Reports
Actor/Role:	Admin
Description:	Generate/export reports on registrations, payments, attendance, and results
Successful completion:	1. Open Reports. 2. Select report type and filters (dates, sport, event, venue). 3. Generate preview. 4. Export CSV/PDF or schedule recurring email delivery.
Alternative:	Large data range → run asynchronously; notify on completion.
Precondition:	Admin should be logged in and data must be available for selected period.
Postcondition:	Report generated/exported, action logged, schedules (if any) saved.
Assumptions:	Organizer asked for event report.



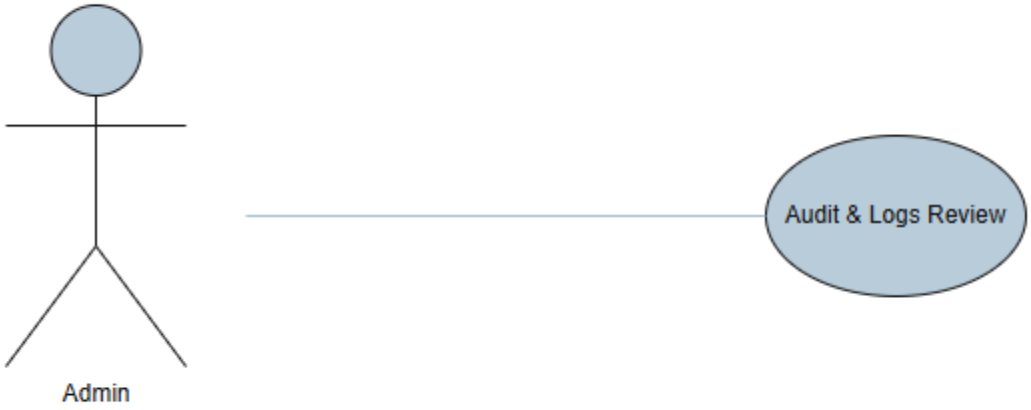
- Moderate Media/Content

Moderate Media/Content Use Case	
Name:	Moderate Media/Content
Actor/Role:	Admin
Description:	Approve/reject media and news items linked to events.
Successful completion:	1. Open Moderation Queue. 2. Review item; approve or reject with reason. 3. System publishes/removes content; notifies submitter.
Alternative:	Rights complaint → auto-takedown, flag for legal review.
Precondition:	Pending items exist, Admin authenticated.
Postcondition:	Content status updated; audit entry stored, notifications sent.
Assumptions:	Organizer published spam event.



- Audit & Logs Review

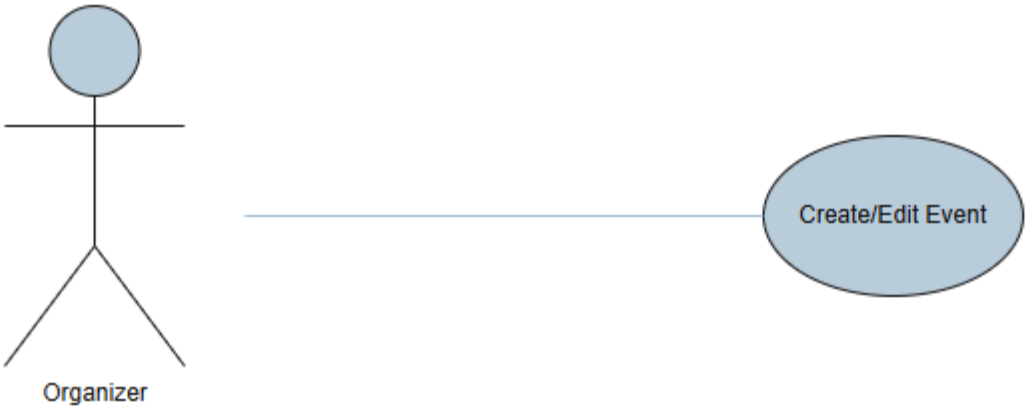
Audit & Logs Review Use Case	
Name:	Audit & Logs Review
Actor/Role:	Admin
Description:	Query immutable audit logs for sensitive operations.
Successful completion:	1. Open Audit & Logs. 2. Filter by actor/action/time/resource. 3. View details (who/when/what, before, after, IP/device). 4. Export evidence (CSV/PDF).
Alternative:	Sensitive category → require step-up 2FA before view.
Precondition:	Admin authenticated, elevated permission if required.
Postcondition:	No data changed, access recorded.
Assumptions:	Log retention/integrity controls in place.



Organizer

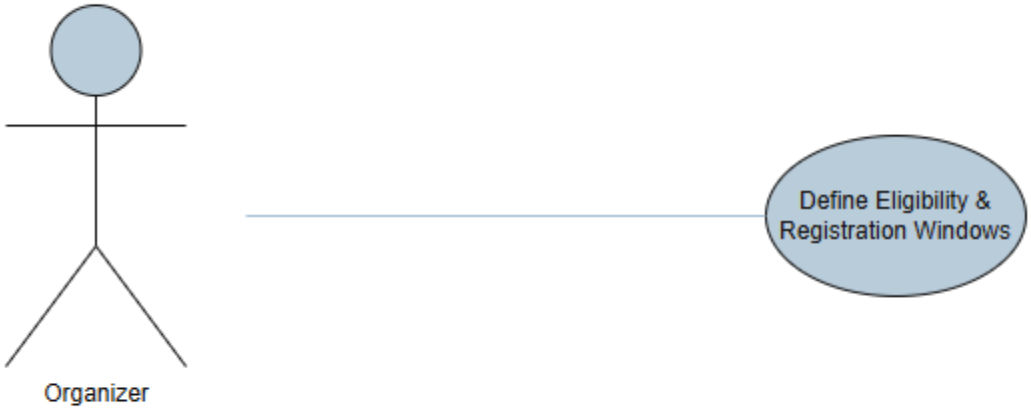
- Create/Edit Event

Create/Edit Event Use Case	
Name:	Create/Edit Event
Actor/Role:	Organizer
Description:	Create and maintain event details.
Successful completion:	1. Open Events → New/Edit. 2. Enter name, sport, dates, venue, capacity. 3. Save draft; resolve validations. 4. Publish when ready.
Alternative:	Venue/date clash → suggest alternate slot/venue.
Precondition:	Organizer authenticated, venue calendar available.
Postcondition:	Event saved (Draft/Published), audit logged.
Assumptions:	Organizer has permission for target venue/sport.



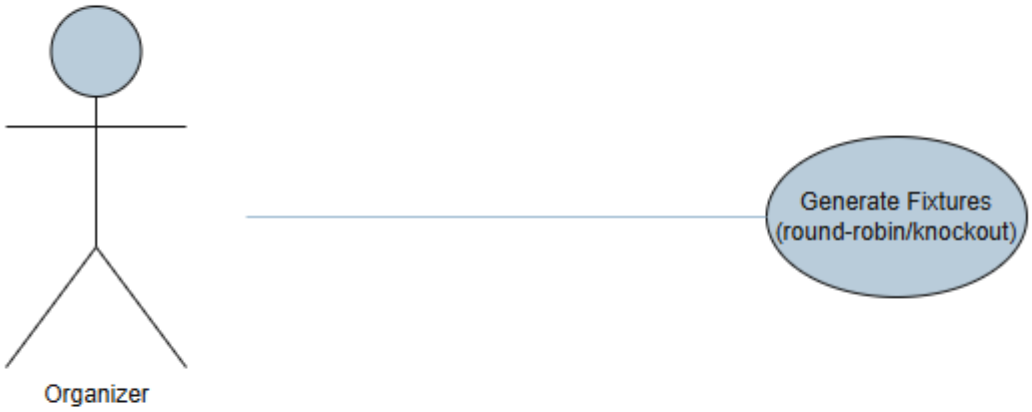
- Define Eligibility & Registration Windows

Define Eligibility & Registration Windows Use Case	
Name:	Define Eligibility & Registration Windows
Actor/Role:	**Organizer**
Description:	Set eligibility criteria, fees, required documents, open/close dates.
Successful completion:	1. Open Event → Registration Settings. 2. Configure eligibility rules and required documents. 3. Set fees and registration open/close times. 4. Save; validations pass.
Alternative:	Invalid window → show error, correct and save.
Precondition:	Event exists (Draft/Published).
Postcondition:	Registration policy stored and enforced.
Assumptions:	System validates rule conflicts.



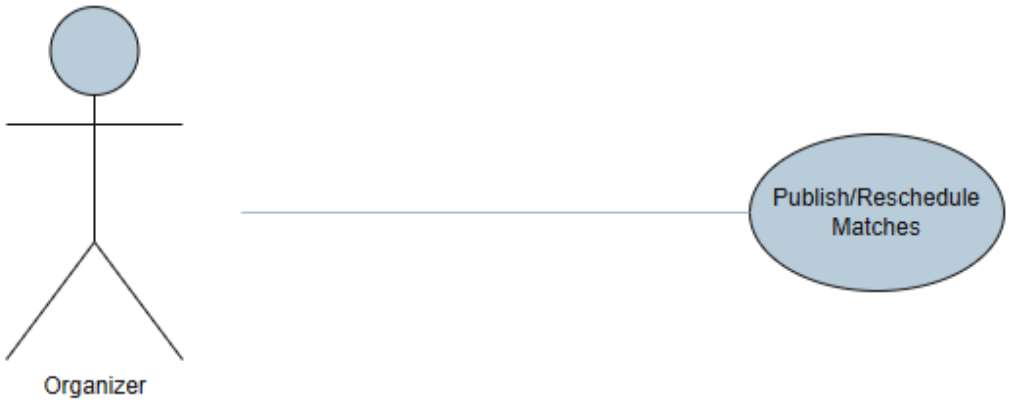
- Generate Fixtures

Generate Fixtures Use Case	
Name:	Generate Fixtures
Actor/Role:	Organizer
Description:	Generate conflict-free schedules and brackets.
Successful completion:	1. Open Event → Fixtures. 2. Choose format (RR/KO) and constraints (venues/slots). 3. Generate proposal; review. 4. Accept and publish.
Alternative:	Conflicts detected → system proposes adjustments; manual edits allowed.
Precondition:	Confirmed participants exceeds minimum required.
Postcondition:	Fixtures saved and visible to stakeholders; notifications queued.
Assumptions:	Algorithm supports constraints, organizer can override.



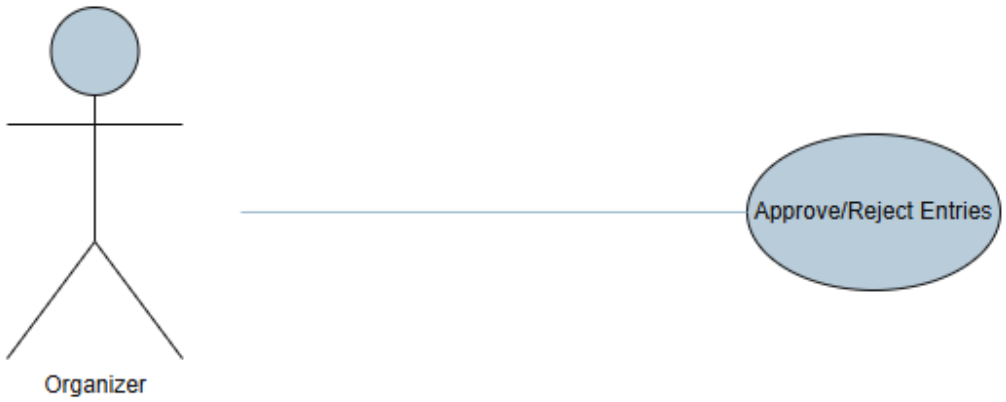
- Publish/Reschedule Matches

Publish/Reschedule Matches Use Case	
Name:	Publish/Reschedule Matches
Actor/Role:	Organizer
Description:	Publish or adjust match timings/venues.
Successful completion:	1. Open Fixtures. 2. Select match; edit date/time/venue. 3. Save; system checks conflicts. 4. Publish, notify impacted users.
Alternative:	Venue unavailable → suggest alternatives, require confirm.
Precondition:	Fixtures exist, organizer authenticated.
Postcondition:	Updated schedule stored, notifications sent; audit logged.
Assumptions:	Reschedule window/policy defined.



- Approve/Reject Entries

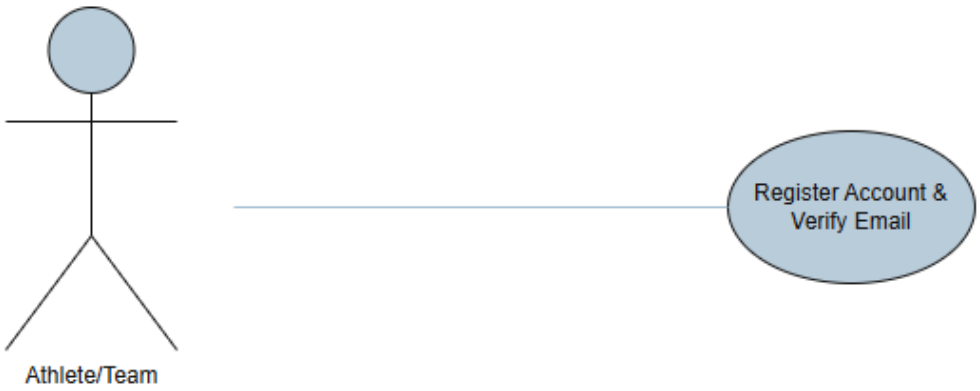
Approve/Reject Entries Use Case	
Name:	Approve/Reject Entries
Actor/Role:	Organizer
Description:	Moderate participant registrations for an event.
Successful completion:	1. Open Event → Registrations. 2. Review application (profile, docs, payment). 3. Approve or reject with reason. 4. System updates status and notifies applicant.
Alternative:	Missing/invalid document → request re-upload, keep Pending.
Precondition:	Registrations submitted, policy defined.
Postcondition:	Entry status updated (Approved/Rejected/Pending fix).
Assumptions:	Organizer can view required docs securely.



Athlete/Team

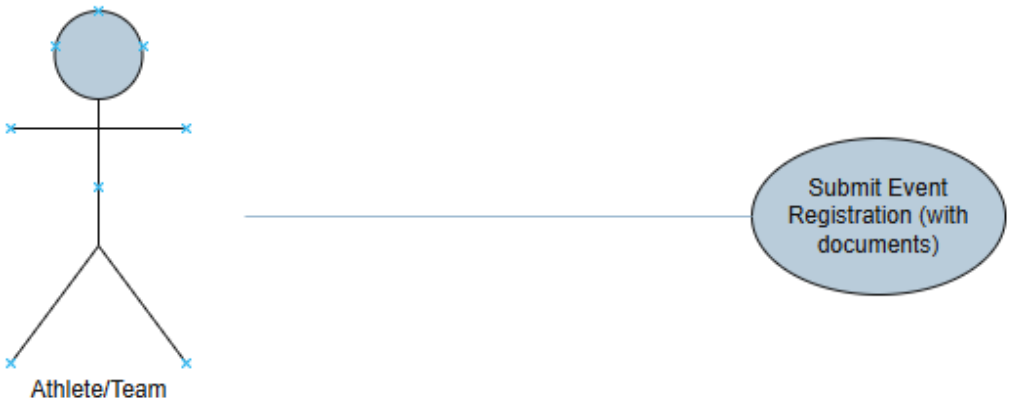
- Register Account & Verify Email

Register Account & Verify Email Use Case	
Name:	Register Account & Verify Email
Actor/Role:	Athlete or Team
Description:	Create a new account and verify ownership of email.
Successful completion:	1. Open Register 2. Enter profile details and email 3. Submit form 4. System sends verification link 5. User clicks link and account becomes active
Alternative:	Link expired or invalid, request a new verification link
Precondition:	User is not logged in and does not already have an account
Postcondition:	Active user account exists and is ready for login
Assumptions:	Email delivery service is available



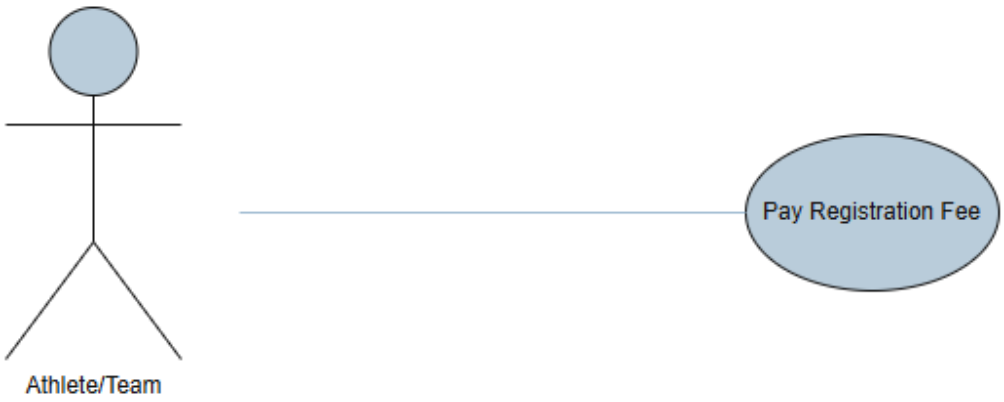
- Submit Event Registration

Submit Event Registration Use Case	
Name:	Submit Event Registration
Actor/Role:	Athlete or Team
Description:	Apply to participate in an event by providing details and documents
Successful completion:	1. Open Event page and click Register 2. Fill participant or team details and choose division 3. Upload required documents such as ID and medical clearance 4. Review fees and confirm 5. System creates a pending registration and redirects to payment
Alternative:	Missing or invalid document, show inline error and request reupload
Precondition:	Event is open for registration and user is authenticated
Postcondition:	Registration record created with status Pending Payment or Waitlisted
Assumptions:	Eligibility rules are configured and enforced



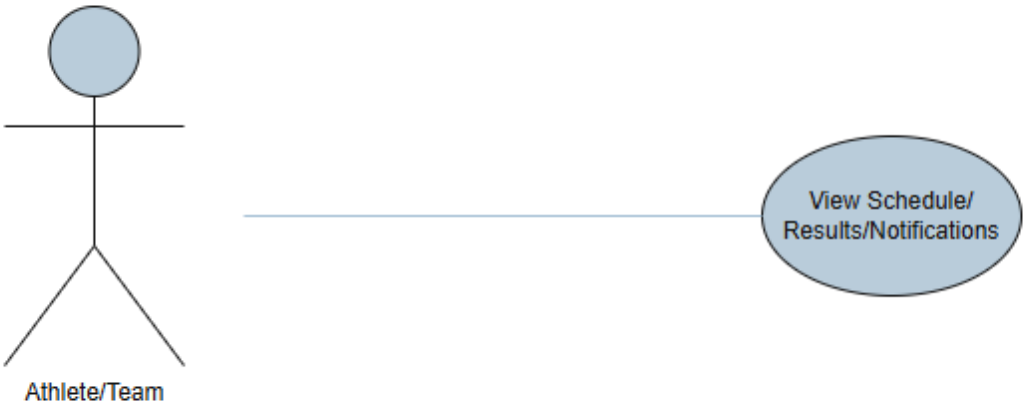
- Pay Registration Fee

Pay Registration Fee Use Case	
Name:	Pay Registration Fee
Actor/Role:	Athlete or Team
Description:	Complete online payment for the registration
Successful completion:	1. Click Proceed to payment 2. System sends payment request to gateway with amount and reference 3. User completes payment on gateway page 4. Gateway returns success status with receipt 5. System marks registration Confirmed and sends receipt by email
Alternative:	Payment fails or is abandoned, status becomes Payment Failed and user can retry
Precondition:	Pending registration exists for the user
Postcondition:	Payment stored and registration confirmed and notification sent
Assumptions:	Payment gateway credentials and webhooks are configured



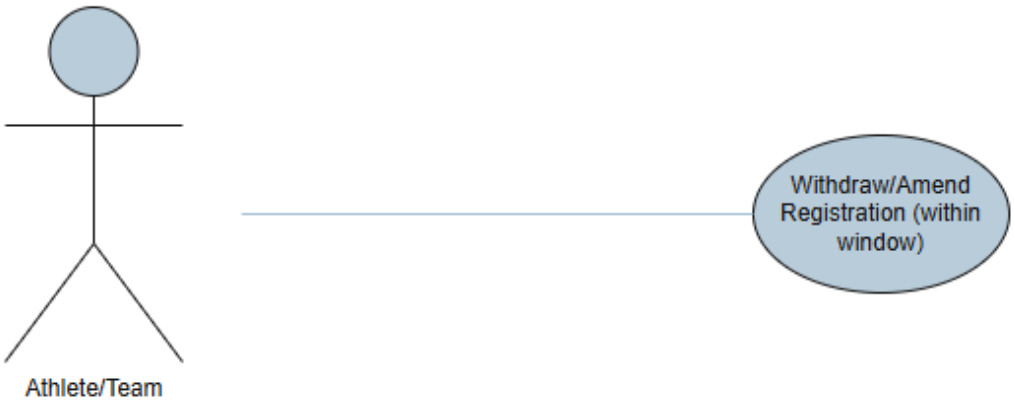
- View Schedule, Results, Notifications

View Schedule, Results, Notifications Use Case	
Name:	View Schedule, Results, Notifications
Actor/Role:	Athlete or Team
Description:	View personal fixtures, results, and system messages
Successful completion:	1. Open Dashboard 2. View upcoming matches and venues 3. View results and leaderboards after matches 4. Read notifications for changes and confirmations
Alternative:	Event rescheduled, system highlights change and sends alert
Precondition:	User is authenticated and has entries or results available
Postcondition:	No data change, only views recorded for analytics
Assumptions:	Fixtures and results have been published



- Withdraw or Amend Registration

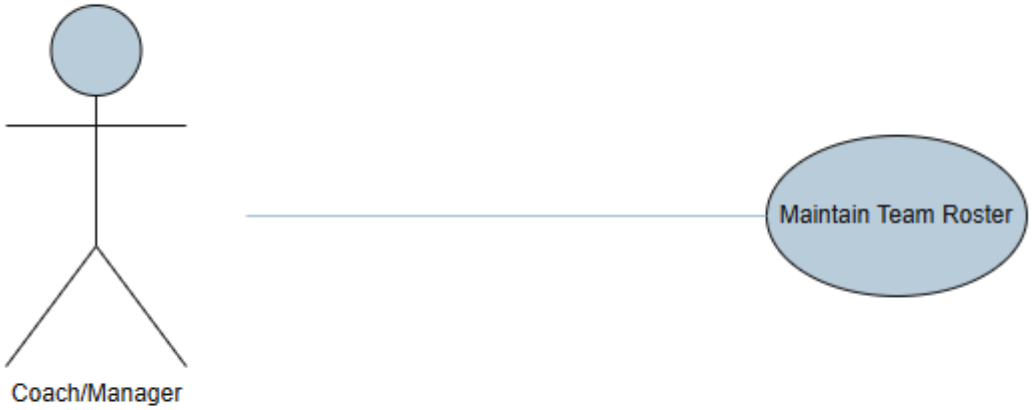
Withdraw or Amend Registration Use Case	
Name:	Withdraw or Amend Registration
Actor/Role:	Athlete or Team
Description:	Update or withdraw a submitted registration within allowed window
Successful completion:	1. Open My Registrations 2. Select registration and choose Amend or Withdraw 3. Edit fields or confirm withdrawal 4. System validates policy and updates status 5. Notifications sent to organizer and user
Alternative:	Window closed by policy, show reason and provide contact path
Precondition:	Registration exists and policy allows change
Postcondition:	Registration updated or withdrawn with audit entry
Assumptions:	Refund rules are applied where relevant



Coach/Manager

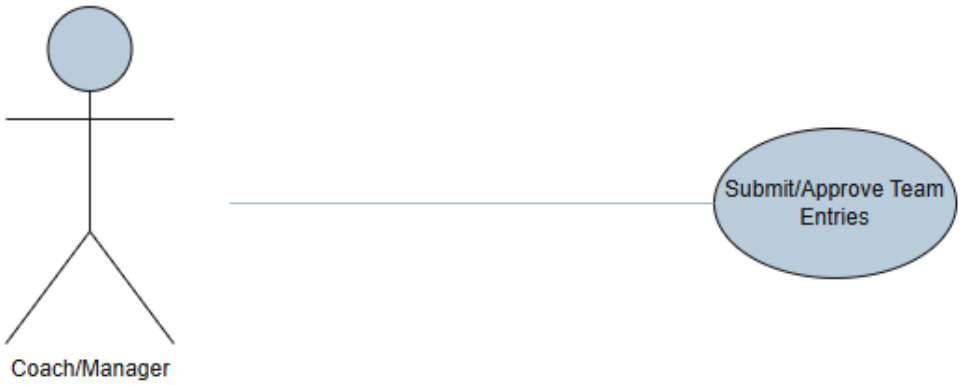
- Maintain Team Roster

Maintain Team Roster Use Case	
Name:	Maintain Team Roster
Actor/Role:	Coach or Manager
Description:	Add or remove players and assign roles for a team
Successful completion:	1. Open Team Roster 2. Add players or update details 3. Save roster 4. System validates eligibility and updates team record
Alternative:	Player not eligible, show reason and block save
Precondition:	Coach or Manager role assigned and team exists
Postcondition:	Roster stored and ready for entries
Assumptions:	Eligibility rules are up to date



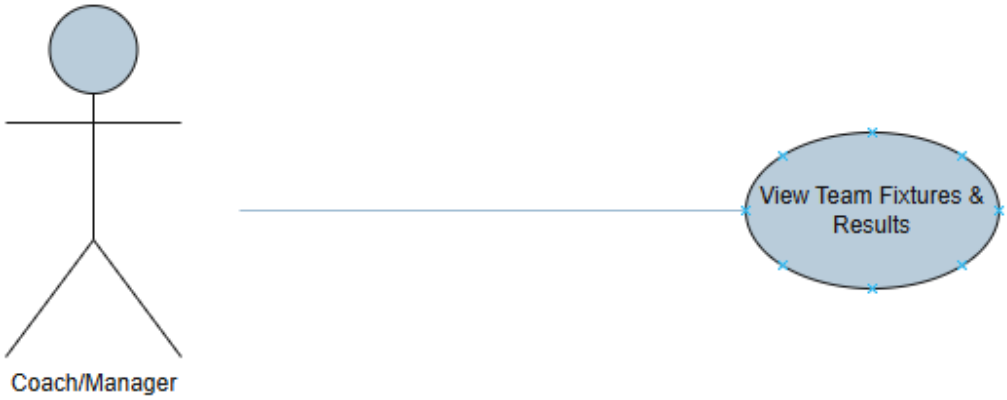
- Submit or Approve Team Entries

Submit or Approve Team Entries Use Case	
Name:	Submit or Approve Team Entries
Actor/Role:	Coach or Manager
Description:	Enter team into an event and confirm entry details
Successful completion:	1. Open Event → Team Entry 2. Select division and provide required details 3. Review fees and confirm 4. System creates pending entry and redirects to payment if required
Alternative:	Roster incomplete, prompt to complete roster first
Precondition:	Team roster exists and event is open for entries
Postcondition:	Entry created with appropriate status and notifications queued
Assumptions:	Coach has permission to act for the team



- View Team Fixtures and Results

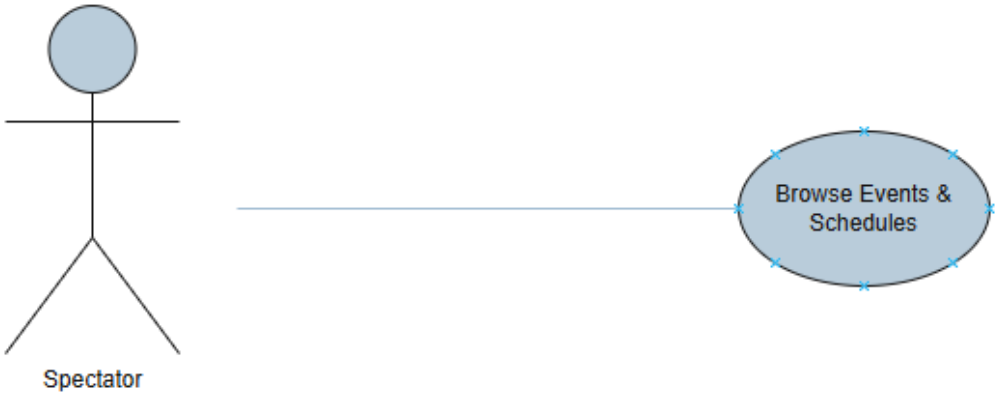
View Team Fixtures and Results Use Case	
Name:	View Team Fixtures and Results
Actor/Role:	Coach or Manager
Description:	Track scheduled matches and outcomes for the team
Successful completion:	1. Open Team Dashboard 2. View fixtures with dates and venues 3. View results and standings
Alternative:	Fixture changes detected, system highlights updates
Precondition:	Published fixtures or results exist
Postcondition:	No data changed, only viewing activity logged
Assumptions:	Organizer has published schedules and results



Spectator

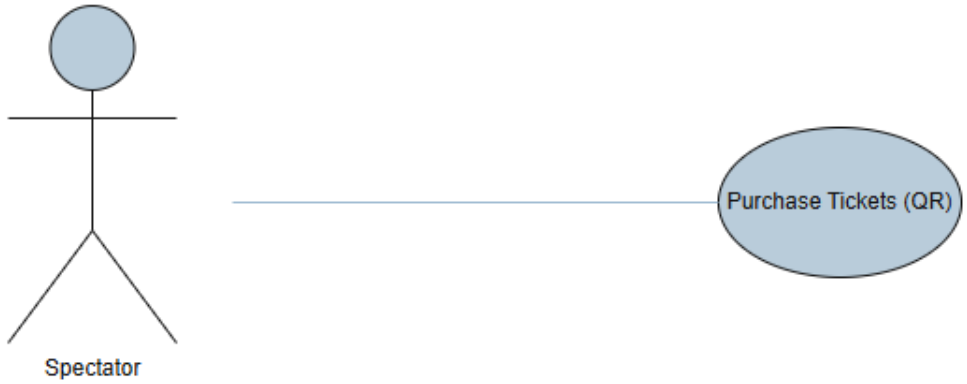
- Browse Events and Schedules

Browse Events and Schedules Use Case	
Name:	Browse Events and Schedules
Actor/Role:	Spectator
Description:	Discover events and view public schedules
Successful completion:	1. Open Events 2. Filter by sport or date or location 3. View event details and schedules
Alternative:	No events match filters, show suggestions or clear filters
Precondition:	Public portal available
Postcondition:	No state change, only browsing analytics captured
Assumptions:	Events are published for public viewing



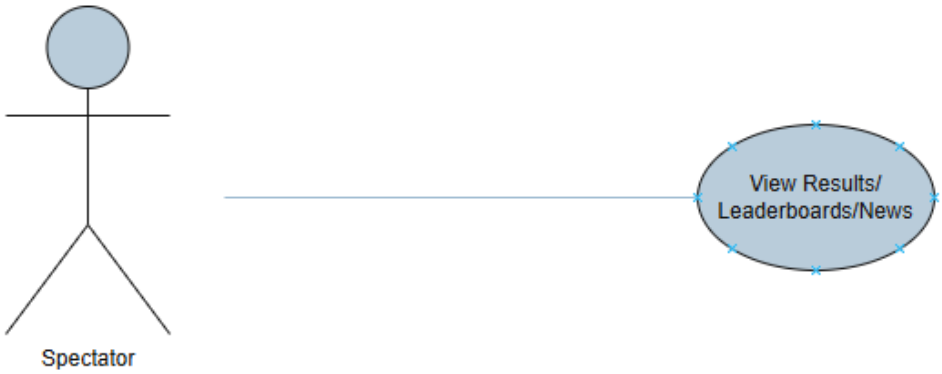
- Purchase Tickets

Purchase Tickets Use Case	
Name:	Purchase Tickets
Actor/Role:	Spectator
Description:	Buy digital tickets for an event and receive QR codes
Successful completion:	1. Select event and ticket type and quantity 2. Proceed to payment 3. Complete payment on gateway page 4. System issues QR code tickets and sends email or SMS
Alternative:	Sold out, offer waitlist option
Precondition:	Ticketing enabled with available inventory
Postcondition:	Ticket records created and revenue logged and delivery confirmed
Assumptions:	Payment gateway and email or SMS services are active



- View Results, Leaderboards, News

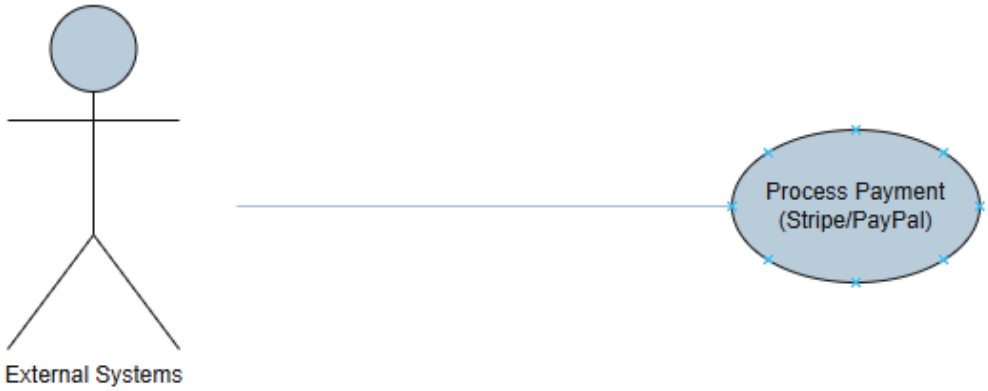
View Results, Leaderboards, News Use Case	
Name:	View Results, Leaderboards, News
Actor/Role:	Spectator
Description:	View public outcomes and updates
Successful completion:	1. Open Results 2. Search or filter by event or sport 3. Read leaderboards and news posts
Alternative:	Results not yet published, show Coming Soon
Precondition:	Public results or news exist
Postcondition:	No data changed, only views logged
Assumptions:	Organizer publishes results and news in a timely manner



Payment Gateway

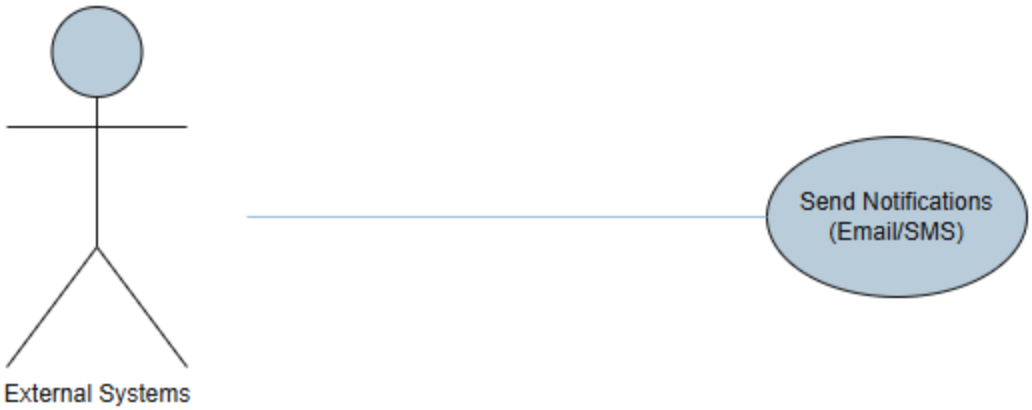
- Process Payment

Process Payment Use Case	
Name:	Process Payment
Actor/Role:	Payment Gateway
Description:	Authorize and capture payments for registrations and tickets
Successful completion:	1. System creates payment request with amount and reference 2. Gateway authorizes and captures funds 3. Gateway sends success response and webhook callback
Alternative:	Authorization fails, send failure status for retry or alternate method
Precondition:	Valid payment intent from system
Postcondition:	Transaction state synchronized between gateway and system
Assumptions:	Webhooks are secure and reliable



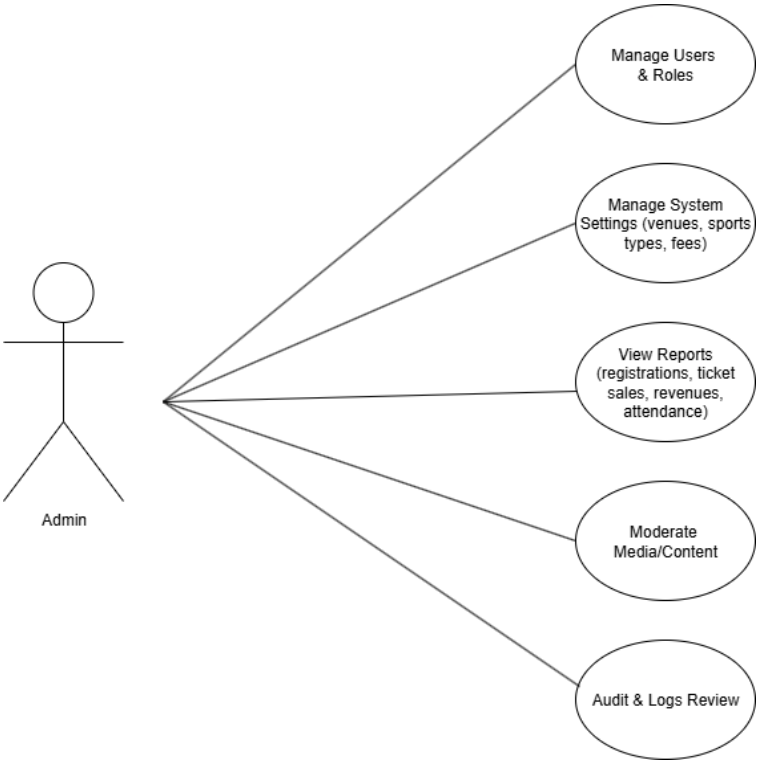
- Send Notifications

Send Notifications Use Case	
Name:	Send Notifications
Actor/Role:	Notification Service
Description:	Deliver email and SMS messages for confirmations and updates
Successful completion:	1. System queues a notification with recipient and template and payload 2. Service sends message 3. Service returns delivery status for logging
Alternative:	Delivery failure, system retries and escalates if threshold reached
Precondition:	Recipient contact details exist and template is available
Postcondition:	Notification status logged and visible in user history where applicable
Assumptions:	Rate limits and compliance rules are respected

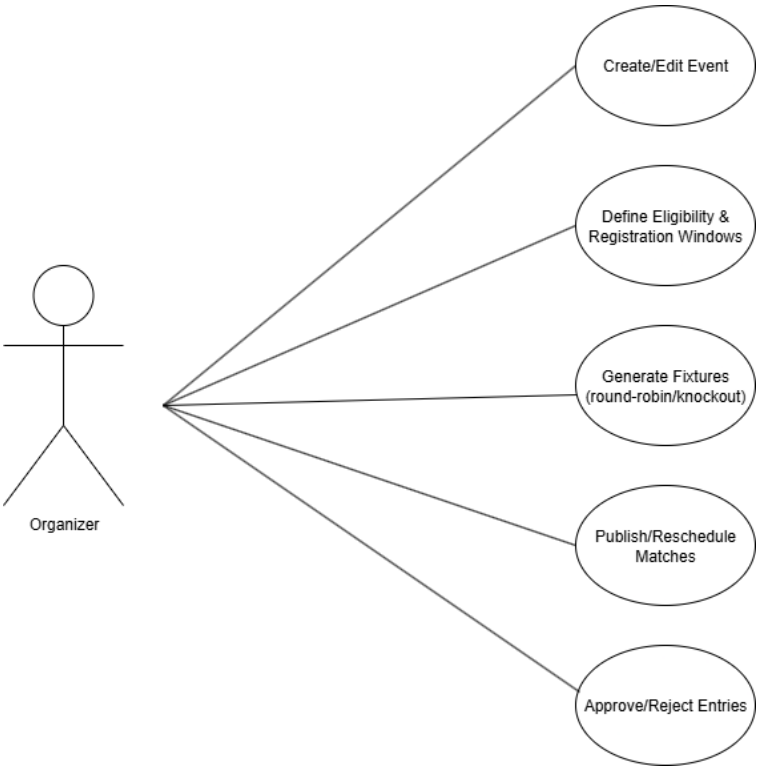

Send Notifications
(Email/SMS)
External Systems

Use Cases of different Roles

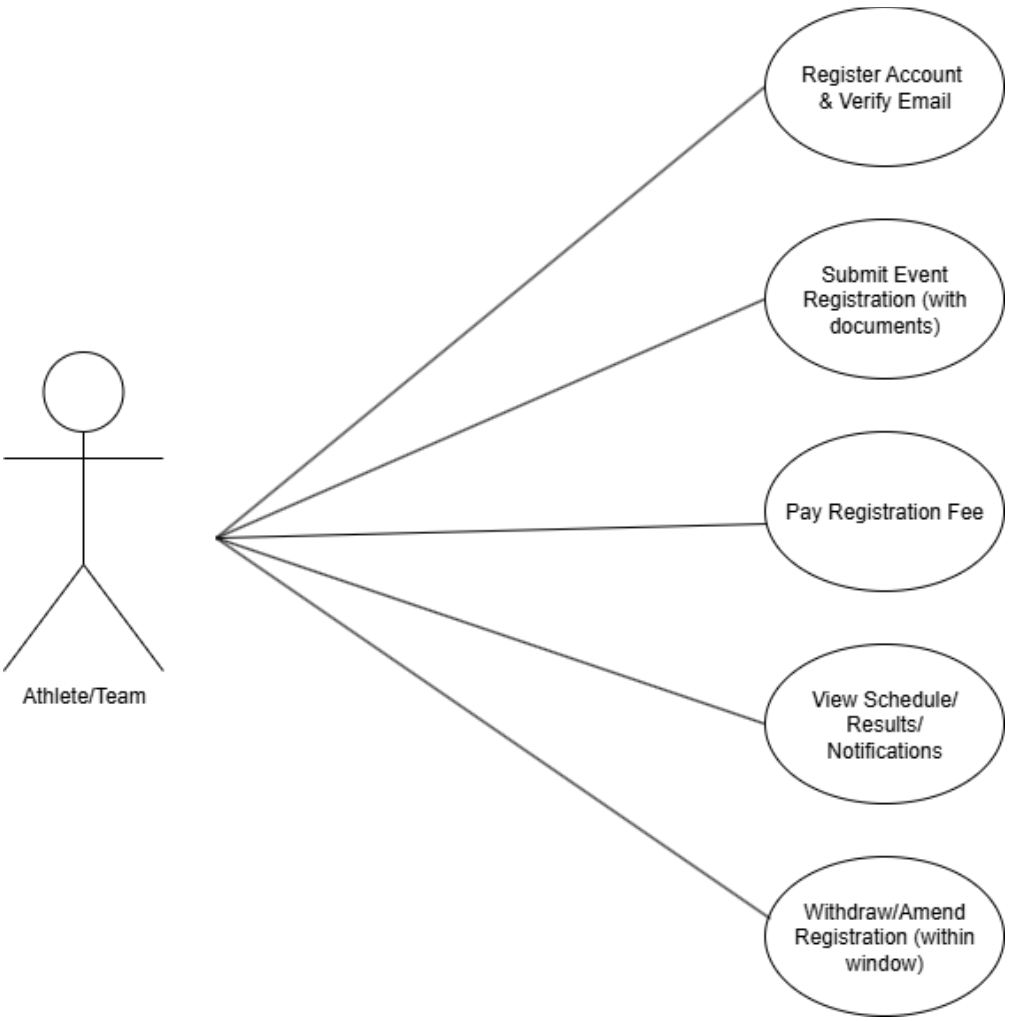
Admin



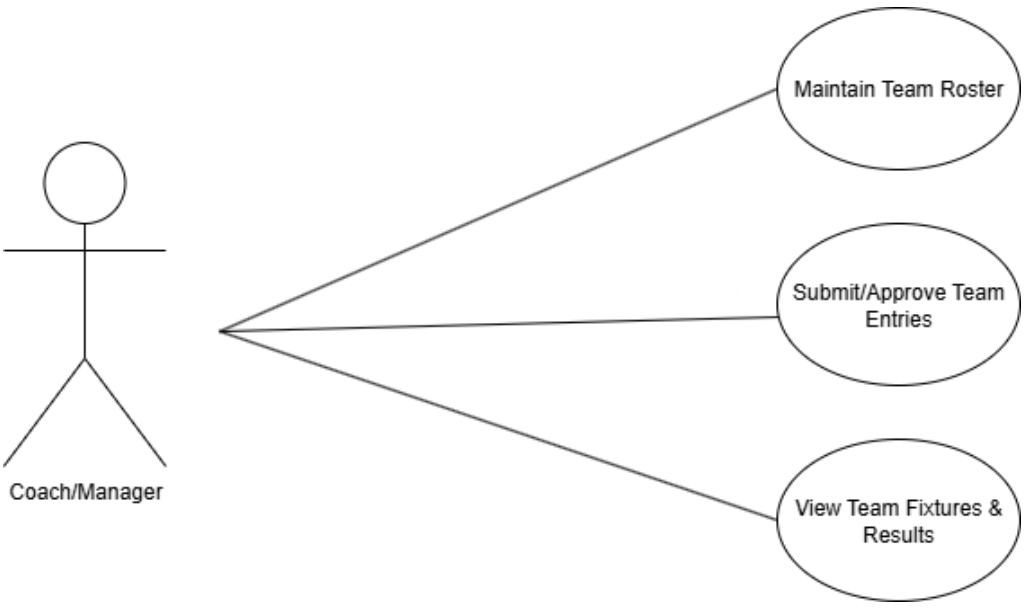
Organizer



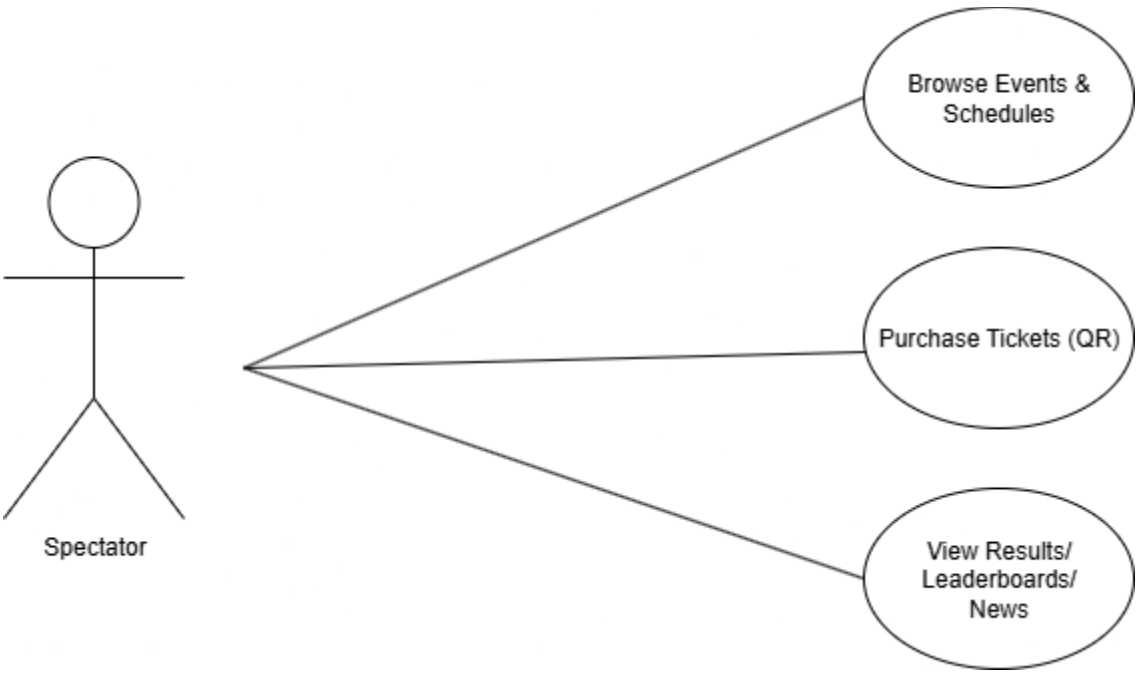
Athlete/Team



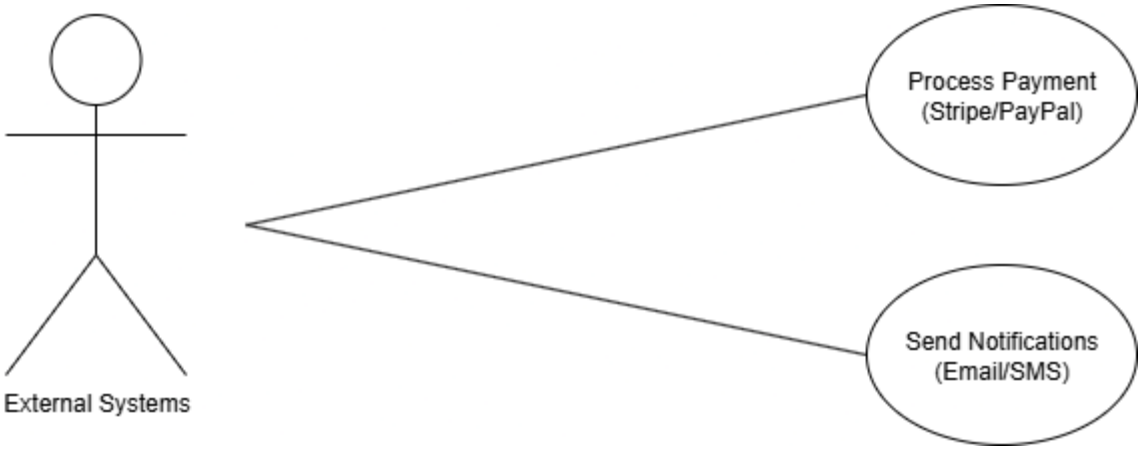
Coach/Manager



Spectator



External Systems



3. Database Design

3.1 Entity–Relationship Design (ERD)

Core entities and keys

- **users** (user_id PK) → email, password_hash, full_name, phone, is_active, created_at
- **roles** (role_id PK) → name
- **user_roles** (user_id FK → users, role_id FK → roles, PK (user_id, role_id))
- **sports** (sport_id PK) → name
- **venues** (venue_id PK) → name, address, capacity
- **events** (event_id PK) → sport_id FK, name, description, start_date, end_date, venue_id FK, capacity, reg_open_at, reg_close_at, fee_cents, status, created_by FK → users
- **event_divisions** (division_id PK) → event_id FK, name, rules_json, min_age, max_age, gender, weight_class
- **teams** (team_id PK) → name, coach_user_id FK → users
- **team_members** (team_id FK, user_id FK, PK (team_id, user_id)) → role_on_team
- **registrations** (registration_id PK) → event_id FK, division_id FK, participant_type {user, team}, participant_id, submitted_by FK, status {pending, waitlisted, confirmed, rejected, withdrawn}, submitted_at, approved_by FK, approved_at
- **payments** (payment_id PK) → registration_id OR ticket_order_id, provider, provider_ref, amount_cents, currency, status {succeeded, failed, pending, refunded}, paid_at, refund_ref
- **fixtures** (fixture_id PK) → event_id FK, division_id FK, round_no, scheduled_at, venue_id FK, status {scheduled, completed, postponed, cancelled}
- **participants** (participant_id PK) → polymorphic link to user or team for fixture entries
- **match_entries** (fixture_id FK, side {home, away}, participant_id FK, PK (fixture_id, side))
- **results** (result_id PK) → fixture_id FK, home_score, away_score, outcome, recorded_by FK, recorded_at
- **ticket_types** (ticket_type_id PK) → event_id FK, name, price_cents, inventory, sales_start, sales_end
- **ticket_orders** (order_id PK) → user_id FK, event_id FK, total_cents, status, created_at, payment_id FK
- **tickets** (ticket_id PK) → order_id FK, ticket_type_id FK, qr_code, status
- **documents** (document_id PK) → owner_user_id FK, registration_id FK NULL, doc_type, storage_url, checksum, encrypted_at
- **notifications** (notification_id PK) → user_id FK, channel {email, sms}, template, payload_json, status, sent_at
- **audit_logs** (audit_id PK) → actor_user_id FK, action, entity_type, entity_id, old_values JSONB, new_values JSONB, ip, created_at

ER Diagram

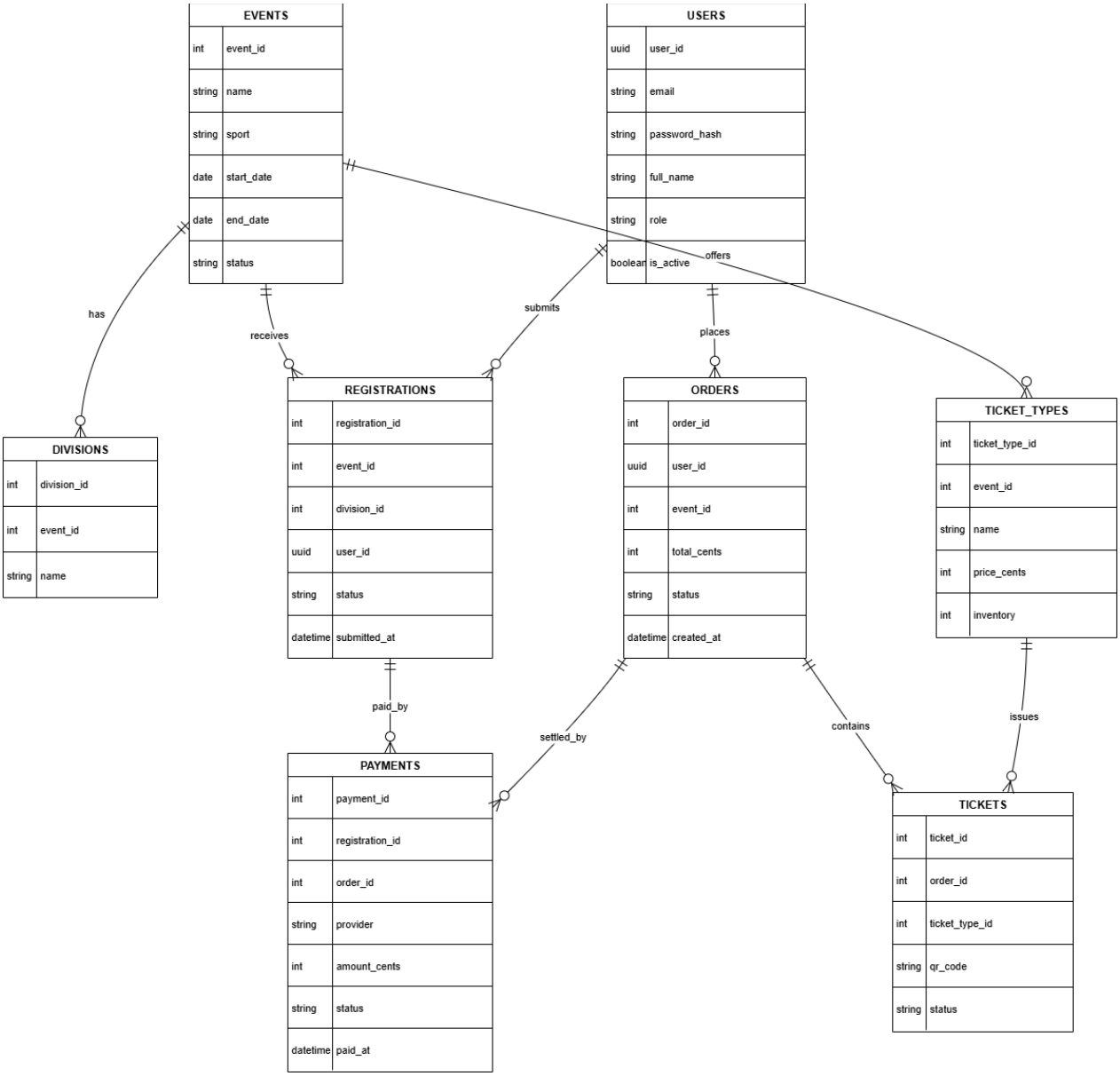


Fig. ER Diagram

3.2 Data Flow Diagram (DFD)

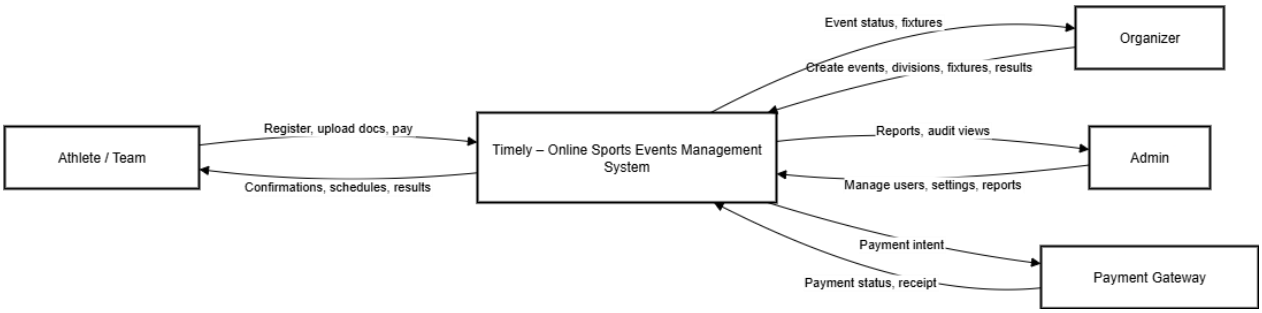


Fig. Data Flow Diagram Level 0

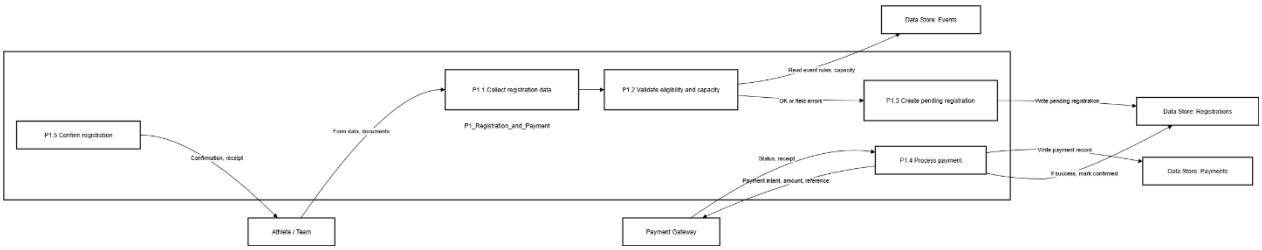


Fig. Data Flow Diagram Level 1 – Registration and Payment

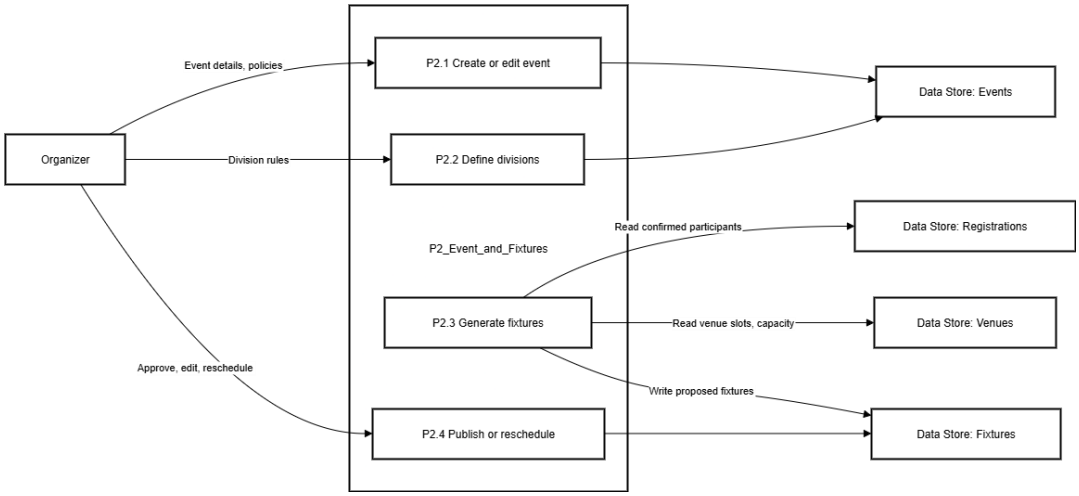


Fig. Data Flow Diagram Level 1 – Event and Fixtures Management

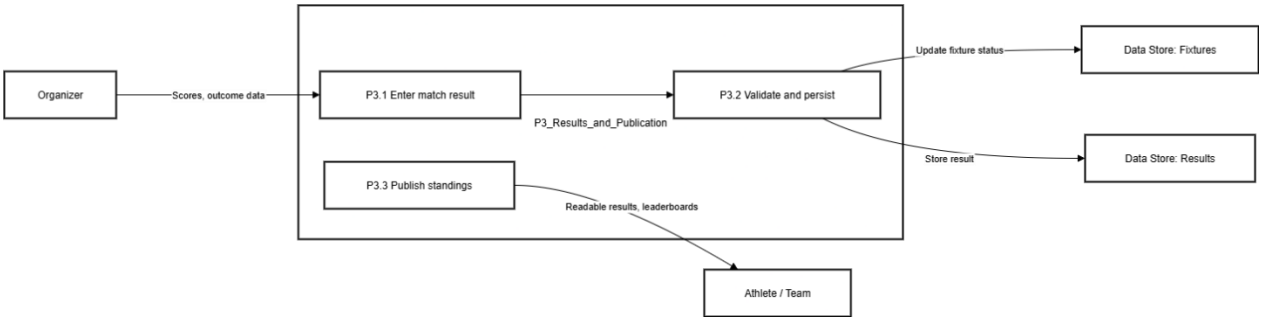


Fig. Data Flow Diagram Level 1 – Results Recording and Publication

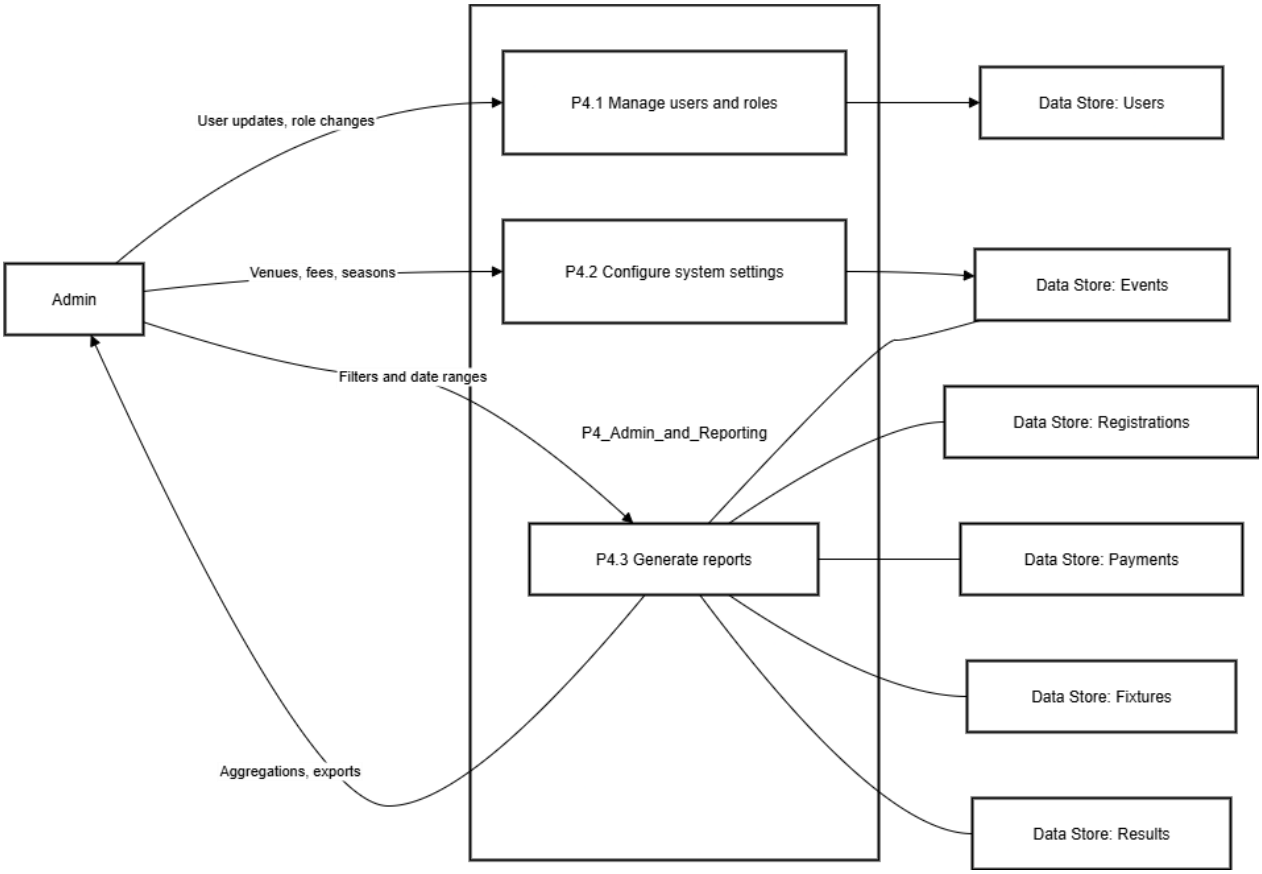


Fig. Data Flow Diagram Level 1 – Admin Operations and Reporting

3.3 Security Implementation

3.3.1 Overview (CIA and privacy-by-design)

- Confidentiality: encrypt sensitive data at rest, TLS in transit, strict RBAC, least-privilege DB access.
- Integrity: server-side validation, signed webhooks, optimistic checks on updates, audit trails.
- Availability: managed Postgres with backups, health checks, rate-limits to prevent abuse.
- Privacy by design: data minimization, purpose limitation, opt-in communications, clear retention and deletion policies.

3.3.2 Authentication & Authorization

- Auth model: Django Auth + DRF.
- Password storage: bcrypt with per-user salts via Django's password hashers. Optional app "pepper" stored outside source control.
- Sessions: web app uses secure cookie sessions with CSRF protection; short idle timeout and absolute session expiry.

- API tokens: for future external integration, issue short-lived JWT access tokens with refresh rotation.
- RBAC: map roles to permissions:

Role	Core Permission
Admin	Manage users and roles, configure events and venues, view reports and refunds
Organizer	Create/edit events and divisions, generate and publish fixtures, record results
Athlete/Team	Register for events, view own schedules and results
Spectator	Browse public info

- Step-up security: 2FA (TOTP) required for Admin, recommended for Organizer.

3.3.3 Security of Transport & Session

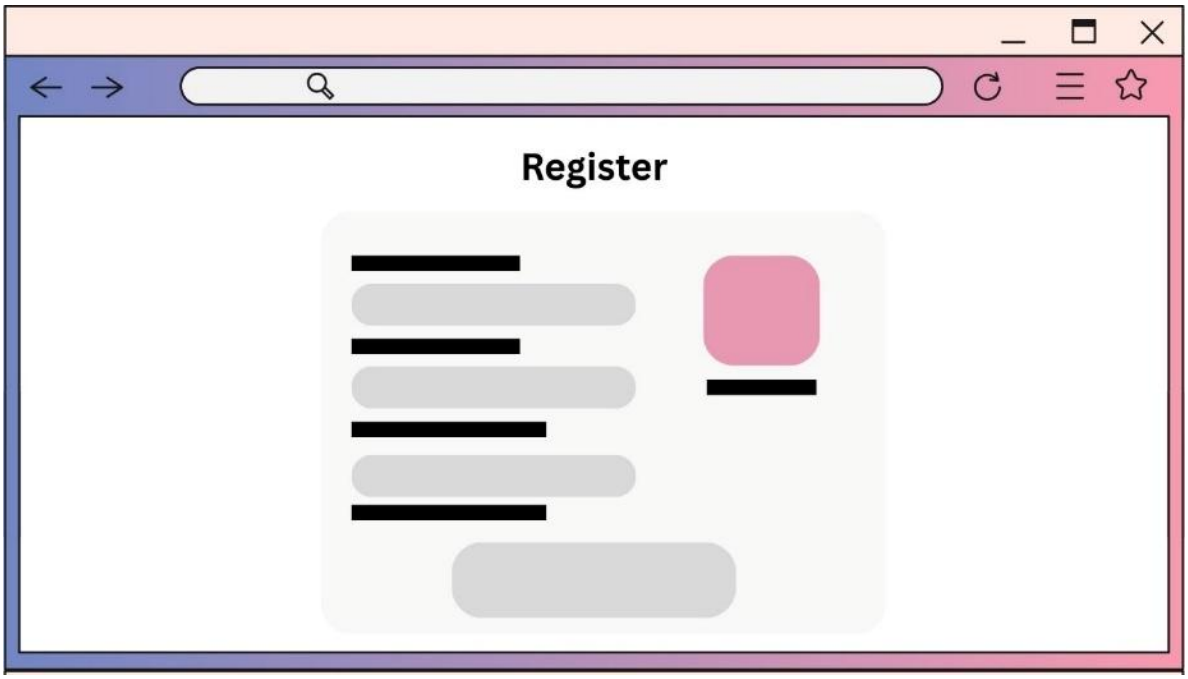
- TLS: Only HTTPS, TLS 1.2+, HSTS.
- Cookies: Secure, HttpOnly (Where app), SameSite=Lax, separate names of session and CSRF.
- CSRF: Django CSRF token on any requests that change a state in the web client.
- CORS: limited to the production scopes, no asterisk berries.

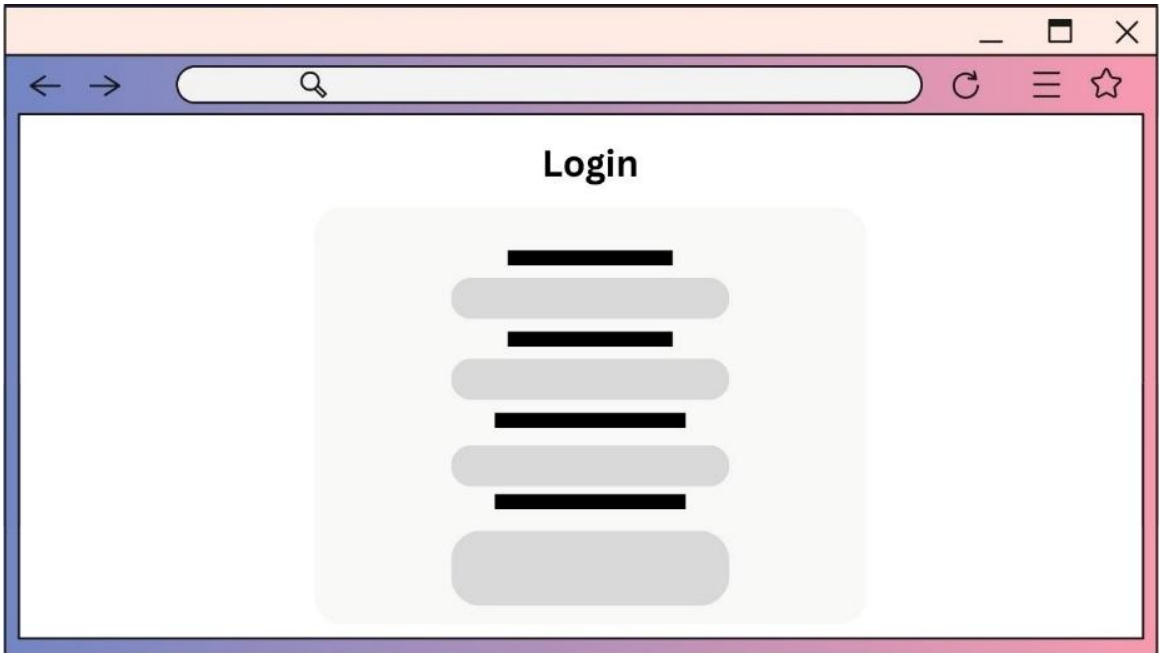
3.3.4 Data Protection-in-Rest

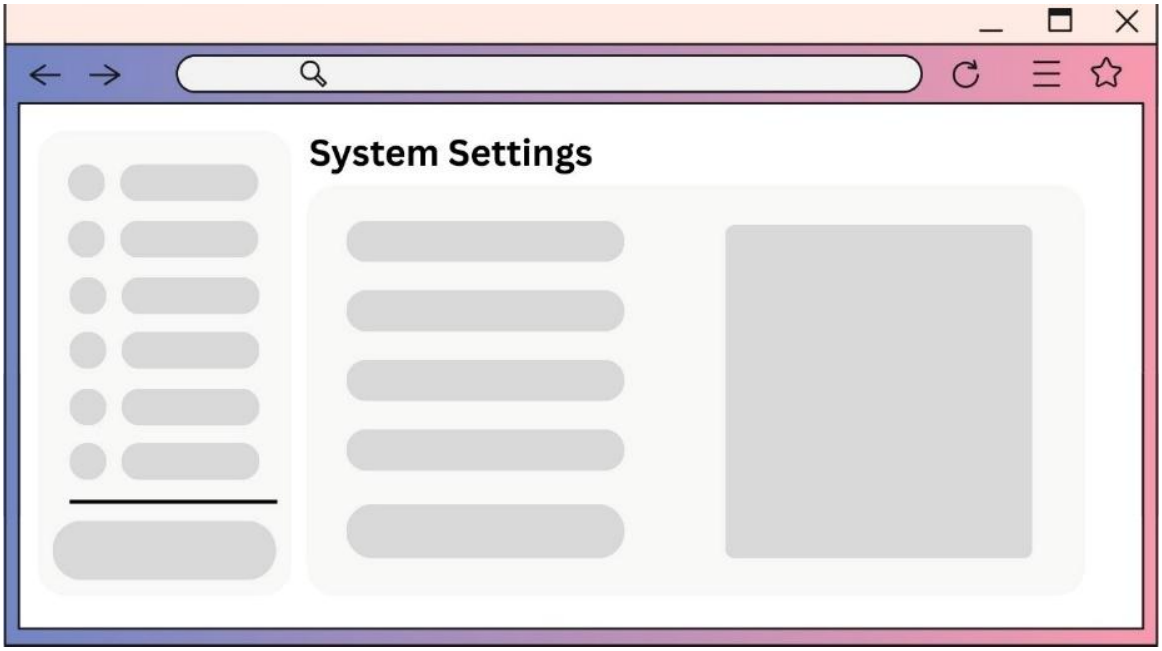
- Encryption of fields (AES-256 app level):
    - users.phone and optional sensitive profile fields,
    - any uploaded document metadata if enabled in future (stored off-DB).
- Encrypted storage:
    - Database: stored Postgres using storage level encryption, encrypted automated backups,
    - Files: in the case of introducing document uploads, it can be stored in an object storage with server-side encryption and managed in bucket with private access.
- Key management: rotation periodically on a known schedule, scoped to each environment, keys never checked into source control or placed in version control.

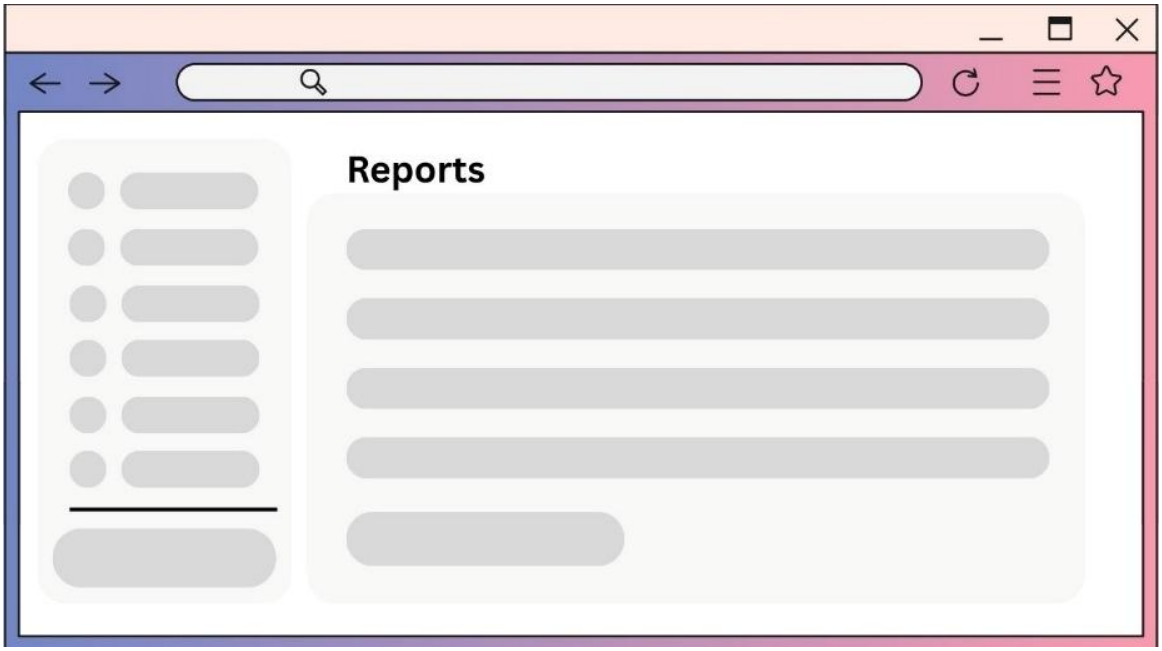
4. User Interface Design

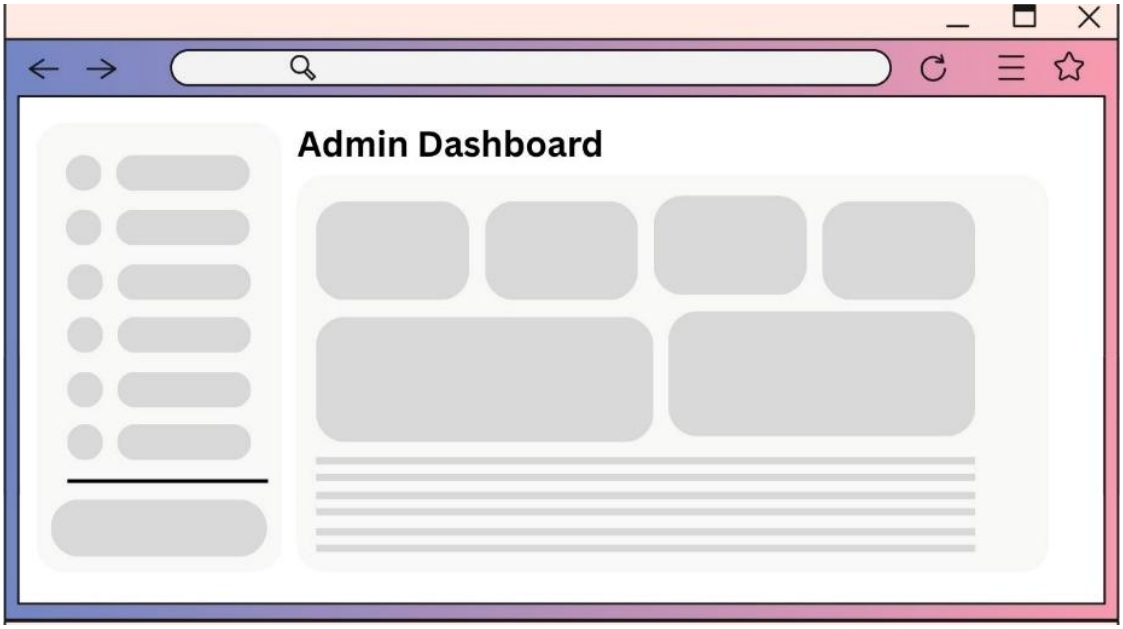
4.1 User Interface Storyboard

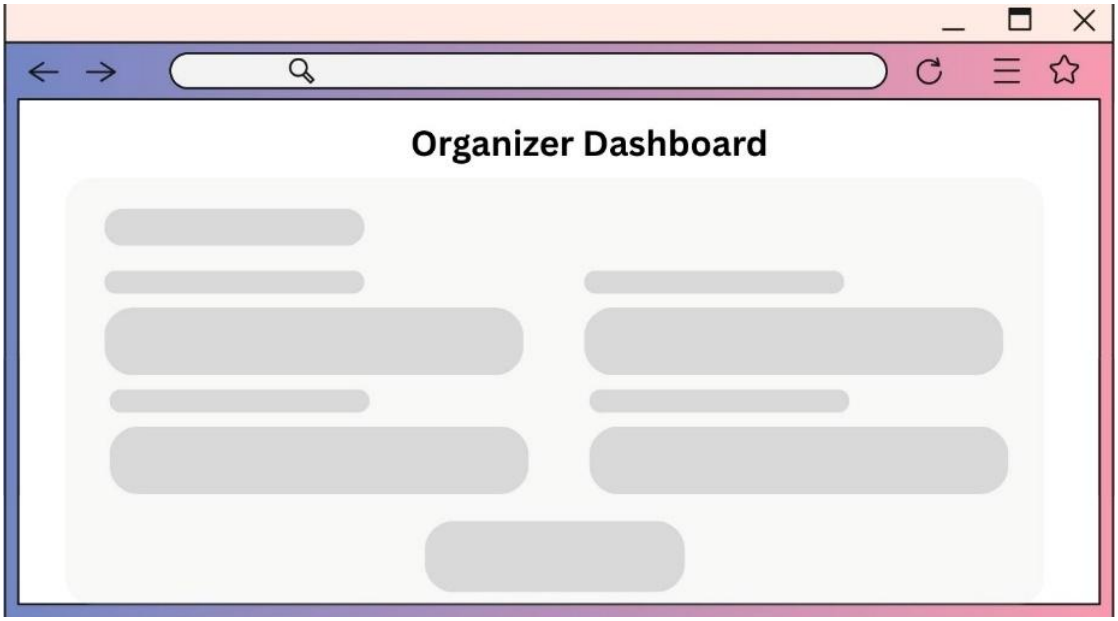


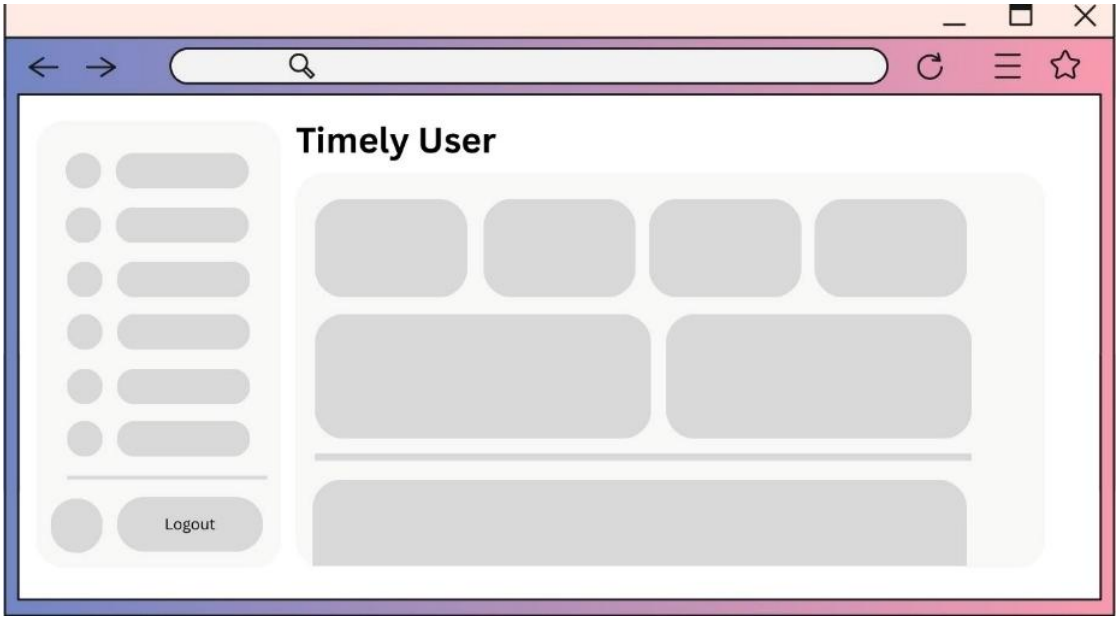



System Settings


Reports


Admin Dashboard


Organizer Dashboard


Timely User
Logout

4.2 Input Data Forms

Key forms, fields, and validations (keep others analogous).

A) User Registration

- Fields: full_name (text, required), email (email, required, unique), password (password, min 8, 1 number), phone (optional, pattern), role (Athlete, Organizer by invite, default Athlete)
- Validations: client + server, duplicate email check, strong password meter
- Errors: inline under field, toast for global errors

B) Login

- Fields: email, password
- Controls: “Remember me”, “Forgot password”
- Security: CSRF token on POST, lockout after repeated failures

C) Create/Edit Event (Organizer)

- Fields: name, sport, description, start_date, end_date, venue, capacity, reg_open_at, reg_close_at, fee_cents
- Rules: end_date ≥ start_date, reg_close_at ≥ reg_open_at, no venue clashes
- Actions: Save draft, Publish, Unpublish

D) Define Divisions (Organizer)

- Fields: division_name, min_age, max_age, gender, rules (rich text or JSON tags)
- Rules: min_age ≤ max_age, at least one division before fixtures

E) Registration Form (Athlete/Team)

- Fields: event, division, team_name optional, emergency_contact, document uploads (ID, medical)
- Rules: within reg window, capacity check, required documents present
- After submit: create “Pending” registration, redirect to payment

F) Payment

- Fields: read-only summary (event, division, fee), payer email, accept T&Cs
- Flow: redirect to hosted gateway, return “success” or “failed”
- Output: receipt email, status update to “Confirmed” on success

G) Fixtures Editor (Organizer)

- Fields: format (round-robin or knockout), venue slots, round count, start times
- Actions: Generate proposal, manual edit, publish

- Guards: conflict detection, warn before overwrite

H) Results Entry (Organizer)

- Fields: fixture selector, home_score, away_score, outcome
- Rules: only scheduled fixtures, scores non-negative integers
- After save: fixture → completed, results visible on public pages

4.3 Output Report Forms

1) Registration Summary (per event)

- Form: table view + export CSV/PDF
- Columns: Reg ID, Athlete/team, Division, Status, Submitted at, Payment status, amount
- Filters: status, date range, division

2) Payments Report

- Columns: Payment ID, Registration ID, Payer, Amount, Status, Provider Ref, Paid At
- Notes: no card information is stored, reference only to a provider
- Exports CSV/PDF, totals footer

3) List of fixtures

- Columns: Fixture ID, Division, Round, Home, Away, Date/Time, Venue, Status
- Actions: Reschedule, Enter Result

4) Results/ Leader board

- Perspectives: event wise, division wise
- Columns: Team/Athlete, Played, Won, Lost, Points, Rank
- PDF to post: export

5) Outputs through emails

- Confirmation of registration, Payment receipt, changes in schedule
- Template: subject, greeting, summary card, CTA link, footer with support

5. Test Plan & System Implementation

5.1 System Implementation Plan

1. **Initial Setup:**

Team Building: Get the project team together composed of project managers, web developers, database design administrators, quality assurance testers, deployment specialists and tech support.

Environment Setup: it is the stage of setting the environment of development, testing and production. Make sure that you get secure socket level(SSL) certificates, backup services and disaster recovery plans.

Data Migration: The old data of the old sports event, players and teams migrated over into the new database of the system.

2. Development and Immigration:

Development: Basic Functions such as character creation, team creation, score updating and access control as per the roles should be created. AES encrypts users' secrets.

Integration: Merge payment gateways, email/SMS message and scheduling tasks onto a calendar events. Make sure every module is able to integrate well.

3. Testing:

First Testing: For each module, e.g, login, registration, event scheduling and score management, do the unit testing.

Quality Assurance: Conduct end-to-end functional testing, usability testing and security check to make sure that system is of a quality.

4. Deployment:

Staging: Put the system into the staging server and do the verification.

Training: Organize training and user manuals to the administrators and event managers.

Production Launch: Release the system to the production-landscap and monitoring of this system in real time.

5. Post-Deployment:

Support: Go there when there is the need to fix issues, through the provision of technical support.

Feedback Gathering: Request feedback of the users in order to facilitate the development of the system.

Documentation: Maintain appropriate user and technical documentation so that they can be used by the future users.

6. Project Timeline:

Week1 & Week2: Setup and Team Set-up

Week3 to 6: Development & integration

Week 7-8: the inner examination

Week 9: Staging/Training

Week 10: Go-Live

Global: Remarks and fix git internacional: remarks & fix git.

5.2. Test Plan

1. **Testing Goals:**

- Specification tests the system.
- Ensure that there is AES encryption and data protection operations.
- Ensure that the system can achieve the standards of the performance, usability and security.

2. **Testing Stages:**

- **Unit Testing:** Get a unit of isolation such as a player registration, the creation of events and the addition of scores.
- **Integration Testing:** It is performed so as to make sure that the different modules can function correctly(e.g., the notification system and the event schedule).
- **System Testing:** Verify end-to-end workflows such as end to end event management and reporting.
- **Security Testing**: Testing access controls, encryption and defense to threats.
- **Performance Testing**: How Tact responds and the maximum load in periods that Tact is highly utilized.
- **User Acceptance Testing (UAT):** it will also test whether the system has been designed to be friendly to the needs of the user as real users (coaches, players, admins) will be used in this process.

3. **Testing Scheduling**

   In week 7 the unit test was conducted

   Week8 Integration testing

4. **Testing Resources**

   Testing Environment: apply to a staging server which is exactly like production.

   Tools: Work with automated testing programs, with load-testing programs and security programs to scan.

Test Data: The sample data should be about the players, teams, etc. and match outcomes have to be realistic.

5. **Reporting**

- Put into records all those arising problems and rate their priorities.
- Fix, re-test documents and verify.
- Before deployment, attain a list of final test reporting reports.

References

Django Software Foundation. (n.d.). Authentication (Django documentation). https://docs.djangoproject.com/en/stable/topics/auth/

European Parliament and Council of the European Union. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation). Official Journal of the European Union, L 119. https://eur-lex.europa.eu/eli/reg/2016/679/oj

International Organization for Standardization & International Electrotechnical Commission. (2011). ISO/IEC 25010:2011 Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models. https://www.iso.org/standard/35733.html

Meta Open Source. (n.d.). React documentation. https://react.dev/

Nielsen Norman Group. (2020). 10 usability heuristics for user interface design. https://www.nngroup.com/articles/ten-usability-heuristics/

Office of the Australian Information Commissioner. (n.d.). Australian Privacy Principles. https://www.oaic.gov.au/privacy/australian-privacy-principles

OWASP Foundation. (2021). OWASP Top 10:2021—The ten most critical web application security risks. https://owasp.org/Top10/

PostgreSQL Global Development Group. (n.d.). PostgreSQL documentation. https://www.postgresql.org/docs/

Pressman, R. S., & Maxim, B. R. (2019). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The definitive guide to Scrum. Scrum.org; ScrumAlliance.org. https://scrumguides.org/

World Wide Web Consortium. (2023). Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/