Software Requirements Specification

SportHub

Multi-Sport Scores & Favorites Tracker

CS3338 – Software Engineering Tools
Final Project

Team 4

Miguel – SDD Lead, Jira Setup
Oscar – SRS Lead, Backend Structure
Florencio – User Manual & Frontend UI
Jesus – Design Spec & TestRail

Semester: Fall 2025

Instructor: _____

Contents

Version History

Version	Date	Author	Description
1.0	2025-11-24	Oscar Herrera	Initial SRS for Snapshot 1 (Start)

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to describe the functional and non-functional requirements for **SportHub**, a web-based sports application developed as the final project for CS3338 – Software Engineering Tools.

SportHub allows users to view upcoming and recent games across multiple sports (NBA, NFL, soccer, MLB), filter schedules, and mark teams as favorites to quickly see upcoming games for those teams. This document serves as a contract between the development team and the instructor and will guide design, implementation, testing, and documentation activities.

1.2 Scope

SportHub is a multi-sport scores and favorites tracker deployed as a web application. At a high level, it will provide:

- Viewing of upcoming and recent games for multiple sports.
- Filtering of games by sport and date.
- Fan accounts with the ability to mark teams as favorites.
- A "My Favorites" view to display upcoming games for favorite teams.
- An administrative interface for managing teams and games (planned for later snapshots).

For **Snapshot 1 – Start**, the scope is primarily:

- Establishing the basic architecture (frontend, backend, database, Docker).
- Implementing a basic schedule view using dummy or seed data.
- Defining initial API endpoints and data models.
- Producing first drafts of all project documents (SRS, SDD, User Manual, Design Spec, Snapshot Objectives).
- Integrating project tooling: GitHub, Jira, TestRail scaffolding, Docker, and LaTeX.

1.3 Definitions, Acronyms, and Abbreviations

API Application Programming Interface.

CRUD Create, Read, Update, Delete operations.

DB Database.

Frontend The client-side portion of the application that runs in the user's browser (React).

Backend The server-side portion that exposes APIs and communicates with the database (Node.js/Express).

Snapshot One of the four major delivery checkpoints (Start, Checkpoint 1, Checkpoint 2, Final).

Favorites Teams selected by a user that will appear in a dedicated "My Favorites" view.

UI User Interface.

1.4 References

- CS3338 Final Project description and snapshot requirements (course document).
- Project GitHub repository: https://github.com/uchiha1121/cs3338-final-sportapp
- SportHub project README in the repository (overview of goals, team roles, and snapshot breakdown).

1.5 Overview

The remainder of this SRS is organized as follows:

- Section 2: Overall description of the system, including product perspective, user classes, and constraints.
- Section 3: External interface requirements, including user interfaces and software interfaces.
- Section 4: System features (functional requirements).
- Section 5: Non-functional requirements (performance, security, usability, maintainability, etc.).
- Section 6: Legal and ethical considerations, data and privacy requirements, and other constraints.
- Section 7: Appendices, including glossary and future work.

2 Overall Description

2.1 Product Perspective

SportHub is a new, standalone web application. It follows a typical three-tier architecture:

- **Frontend:** React-based single-page application providing the user interface.
- **Backend:** Node.js/Express server exposing RESTful endpoints.
- **Database:** MongoDB for storing users, teams, and game schedules.

Services are orchestrated with Docker Compose to simplify local deployment and eventual hosting. External sports data APIs may be integrated in future snapshots, but Snapshot 1 may rely on dummy or seeded data.

2.2 Product Functions

At a high level, SportHub will provide the following functionality:

- Display upcoming and recent games for supported sports.
- Filter games by sport and date.
- Permit users to register, log in, and manage their account (basic scaffolding in early snapshots).
- Allow users to mark teams as favorites.
- Provide a "My Favorites" view listing upcoming games involving those teams.
- Provide administrative functionality to manage teams and game records (planned for later snapshots).

2.3 User Classes and Characteristics

Fans (End Users): Users interested in viewing sports schedules and scores. They may create accounts and mark favorite teams. Technical expertise is not assumed.

Admin Users: Trusted users who can manage teams and games in the system. May be members of the development team or instructor.

Developers & Testers: Internal users who maintain the system, write tests, configure CI/CD, and manage Jira/TestRail artifacts.

2.4 Operating Environment

- **Client:** Modern web browser (Chrome, Firefox, Edge, Safari) with JavaScript enabled.
- **Server:** Node.js runtime running in a Linux-based environment (e.g., via Docker container).
- **Database:** MongoDB (containerized).
- **Deployment:** Docker Compose, with potential hosting on a cloud VM (e.g., AWS EC2) or institutional server.
- **Documentation tools:** LaTeX for all project documents compiled into PDF.

2.5 Design and Implementation Constraints

- The project must use GitHub for version control and store all documentation in the repository.
- Core documents (SRS, SDD, User Manual, Design Spec, Snapshot Objectives) must be authored in `.tex` files.
- The system must be containerized using Docker and `docker-compose.yml`.
- React should be used on the frontend; Node.js/Express on the backend; MongoDB for persistence.
- Jira and TestRail must be incorporated into the workflow as specified by the course.

2.6 Assumptions and Dependencies

- Users have a stable internet connection and an up-to-date browser.
- For Snapshot 1, game and team data may be manually seeded; later snapshots may depend on third-party sports APIs.
- All team members have access to GitHub, Jira, TestRail, Docker, and LaTeX tooling.
- The course infrastructure and grading expectations remain consistent throughout the semester.

3 External Interface Requirements

3.1 User Interfaces

3.1.1 Home Page

- Provides a high-level introduction to SportHub.
- Displays a summary of recent or upcoming games.
- Offers navigation to other pages: Teams, Games, My Favorites (when logged in), and Admin (for authorized users).

3.1.2 Teams Page

- Lists teams across supported sports.
- Allows users to view basic team information (name, sport, and optionally logo).
- For logged-in users, offers controls to add/remove the team from their favorites.

3.1.3 Games Page

- Displays a list of games with information such as sport, date/time, and participating teams.
- Allows filtering by sport (NBA, NFL, soccer, MLB) and date.
- May show scores for completed games or "TBD" for upcoming games.

3.1.4 My Favorites Page

- Accessible to logged-in users.
- Shows upcoming games for the user's favorite teams.
- Encourages users with no favorites to add teams from the Teams page.

3.1.5 Admin Dashboard (Future Snapshot)

- Restricted to admin users.
- Supports adding, editing, and deleting teams and games.
- Provides basic data validation feedback on input.

3.2 Software Interfaces

3.2.1 Backend REST API

Representative endpoints (names and details may evolve):

- **Teams**
  - `GET /api/teams`
    Returns all teams, optionally filtered by sport.
  - `POST /api/teams` (admin)
    Creates a new team.
- **Games**
  - `GET /api/games?sport={sport}&date={date}`
    Returns games filtered by sport and/or date.
  - `POST /api/games` (admin)
    Creates a new game entry.
- **Users & Favorites**
  - `POST /api/users/register`
    Registers a new user.
  - `POST /api/users/login`
    Authenticates a user and returns a session token or cookie.
  - `GET /api/users/me/favorites`
    Returns the list of favorite teams for the logged-in user.
  - `POST /api/users/me/favorites`
    Adds a team to the user's favorites.
  - `DELETE /api/users/me/favorites/:teamId`
    Removes a team from favorites.

3.2.2 Database Interface

The backend will interact with MongoDB using Mongoose or a similar ODM. Primary collections include:

- `users`: user credentials, profile data, and favorite teams.
- `teams`: team name, sport, and optionally logo/metadata.
- `games`: participating teams, sport, date/time, and scores.

3.3 Communications Interfaces

- Client-to-server communication via HTTP/HTTPS.
- Internal service communication (frontend, backend, database) via the Docker Compose network.
- Future integration with external sports data APIs via HTTPS.

4 System Features

4.1 Feature 1: View Upcoming and Recent Games

Description

Users can view lists of upcoming and recent games for supported sports.

Functional Requirements

- F1.1 – The system shall provide an endpoint to retrieve games (upcoming and recent).
- F1.2 – The UI shall display a list of games with sport, participating teams, and date/time.
- F1.3 – If no games are available for the selected filters, the system shall display a "No games available" message.

4.2 Feature 2: Filter Games by Sport and Date

Description

Users can filter games based on sport and date.

Functional Requirements

- F2.1 – The UI shall provide a control for selecting a sport (NBA, NFL, soccer, MLB).
- F2.2 – The UI shall provide a date picker or equivalent control to select a date.
- F2.3 – The backend shall filter results by the selected sport and date.
- F2.4 – Invalid date parameters shall result in an error response; the frontend shall show a user-friendly error message.

4.3 Feature 3: User Accounts and Authentication (Basic)

Description

The system will support basic user accounts for fans.

Functional Requirements

- F3.1 – The system shall allow a user to register with a unique identifier (e.g., email or username) and password.
- F3.2 – The system shall allow a user to log in using valid credentials.
- F3.3 – The system shall prevent access to user-specific features (e.g., My Favorites) when the user is not authenticated.

4.4 Feature 4: Manage Favorite Teams

Description

Authenticated users can mark certain teams as favorites.

Functional Requirements

- F4.1 – The system shall allow an authenticated user to add a team to their favorites.
- F4.2 – The system shall allow an authenticated user to remove a team from their favorites.
- F4.3 – The system shall prevent duplicate favorites for the same team and user.

4.5 Feature 5: My Favorites View

Description

Authenticated users can view a list of upcoming games for their favorite teams.

Functional Requirements

- F5.1 – The system shall allow an authenticated user to request games only for their favorite teams.
- F5.2 – The UI shall display an informative message when the user has no favorite teams.

4.6 Feature 6: Admin Management of Teams and Games (Planned)

Description

Admin users can manage teams and games.

Functional Requirements

- F6.1 – The system shall restrict admin operations to users with admin privileges.
- F6.2 – The system shall allow admin users to create, update, and delete team records.
- F6.3 – The system shall allow admin users to create, update, and delete game records.

5 Non-Functional Requirements

5.1 Performance Requirements

- N1 – Under typical load, API responses for basic game and team queries should complete within 500 ms on the backend.
- N2 – Pages should render in under two seconds on a modern browser and broadband connection.

5.2 Security Requirements

- N3 – Passwords shall never be stored in plain text; they must be hashed using a secure algorithm (e.g., bcrypt) in production-ready snapshots.
- N4 – Authentication tokens or cookies shall be protected against common web vulnerabilities (e.g., session hijacking).
- N5 – Admin-only features must not be accessible to non-admin users.

5.3 Usability Requirements

- N6 – Navigation shall be consistent across pages, using a common navigation bar.
- N7 – The application shall be usable on common desktop resolutions; responsive design for tablets is recommended.
- N8 – Error messages shall be clear and human-readable.

5.4 Maintainability Requirements

- N9 – Code should be modular, with separate folders for routes, controllers, and models in the backend, and for components and pages in the frontend.
- N10 – The repository shall include documentation for setup and contributions in the README and User Manual.
- N11 – Developers should follow consistent coding styles enforced with linters and formatters where possible.

5.5 Portability Requirements

- N12 – The system shall be deployable via Docker Compose on any host that supports Docker (Linux, macOS, Windows with Docker Desktop).
- N13 – Environment-specific configuration (e.g., DB URI) shall be provided via environment variables rather than hard-coded.

5.6 Reliability and Availability

- N14 – The backend shall handle transient database connection failures gracefully (e.g., retries on startup).
- N15 – Critical errors on the server shall be logged for later inspection.

6 Legal and Ethical Considerations

6.1 User Data and Privacy

- L1 – The system shall store only the minimum user data required for functionality (e.g., login and favorites).
- L2 – User passwords and any sensitive data must be handled securely, following best practices for web security.
- L3 – Users shall be informed (e.g., in a privacy or terms section) how their data is stored and used.

6.2 Sports Data Usage

- L4 – If real sports data APIs are used, their terms of service must be followed, including any usage limits and attribution requirements.
- L5 – The project shall avoid scraping sports data from websites in violation of their terms of use.

6.3 Ethical Considerations

- L6 – The system shall not encourage abusive or harmful behavior toward teams, players, or other users.
- L7 – Any logging or analytics must respect user privacy and avoid unnecessary collection of personally identifiable information.

7 Appendices

7.1 Glossary

End User / Fan A user who uses SportHub to view sports schedules and mark favorites.

Admin A privileged user who can modify teams and games.

Snapshot A required checkpoint deliverable for the course, including code, documents, Jira/TestRail artifacts, and reflections.

Docker Compose A tool for defining and running multi-container Docker applications.

7.2 Future Work

Potential enhancements beyond Snapshot 1 include:

- Integration with live sports data APIs for real-time scores and schedules.
- Push notifications or email alerts for upcoming games and final scores.
- More advanced filtering (by team, venue, league, or playoff status).
- Mobile-friendly or native mobile app versions.
- Rich statistics (player stats, standings, advanced analytics).