**Squad Master - Design Document**

**1. Introduction**

Squad Master is a desktop application designed to help users build optimal football squads by filtering and selecting players based on various criteria. This document outlines the system’s architecture, key implementations, use cases, and design decisions.

**2. System Overview**

**Brief Project Description**

Squad Master provides an interactive and user-friendly experience for squad selection. The application is built using Java Swing for the UI and integrates a RESTful API to fetch player data. Users can filter players based on attributes such as position, rating, and physical characteristics, then construct a squad accordingly.

**System Architecture**

Squad Master follows a **layered architecture**:

* **Presentation Layer:** Java Swing-based GUI for user interaction.
* **Business Logic Layer:** Filtering, optimization, and player selection algorithms.
* **Data Access Layer:** API integration to fetch and process player data.
* **Persistence Layer:** Saving squad configurations and user preferences (if required).

**Technology Stack**

* **Programming Language:** Java (Maven-based project)
* **UI Framework:** Java Swing
* **API Integration:** REST API for fetching player data
* **Testing Frameworks:** JUnit for unit testing, Mockito for integration testing
* **Version Control:** GitHub for collaboration and documentation tracking

**3. Implementation Details**

**Codebase Structure**

The project will be structured as follows:

SquadMaster/

│── src/

│ ├── main/java/org/example

│ │ ├── model/ (Java Swing UI components)

│ │ ├── service/ (API integration and data processing)

│ │ ├── utils/ (Filtering & optimization algorithms)

│ │

│ ├── test/java/org/example

│ ├── unit/ (Unit tests)

│ ├── integration/ (Integration tests)

│── pom.xml (Maven dependencies)

│── README.md (Project documentation)

**Key Implementations**

* **Filtering System:** Allows users to filter players by position, rating, skills, and physical attributes.
* **Squad Selection Algorithm:** Ensures balance in team formation by optimizing player selection.
* **Data Processing:** API calls fetch player details, which are then processed and stored for further operations.
* **GUI Components:** Interactive Java Swing-based UI enabling drag-and-drop squad formation.

**Component Interfaces**

* **Player Model:** Represents individual player attributes.
* **Squad Model:** Stores the list of selected players and their assigned positions.
* **API Handler:** Fetches data from an external API.
* **UI Controller:** Manages interactions between GUI components and business logic.

**4. Use Case Support in Design**

**Use Case Selection**

The following four primary use cases define the system's core functionalities:

1. **Filtering and Selecting Players Based on Performance Metrics**
2. **Fixing Specific Positions and Generating an Optimal Squad**
3. **Adjusting Model Parameters to Generate Alternative Squads**
4. **Displaying Squad Analysis and Exporting the Final Squad**

**Requirement Mapping**

Each of these use cases maps directly to functional requirements outlined in the requirements document.

**Use Case Design**

* **Data Flow Diagrams:** Illustrate how user inputs filter player data and how squads are generated.
* **State Diagrams:** Define how squad configurations are updated dynamically.
* **Interaction Diagrams:** Showcase user interactions with the UI.

**5. Design Decisions**

**Technology Comparisons**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Java Swing** | **JavaFX** |
| UI Complexity | Medium | High |
| Performance | High | Medium |
| Styling Support | Limited | Advanced |
| Learning Curve | Easy | Moderate |

**Decision Justification:** Java Swing is chosen due to its ease of integration, lightweight UI, and better performance for a simple desktop application. . If there is a need for enhanced visual improvements and animations in the future, transitioning to JavaFX can be considered.

**Final Notes:** This document serves as a guideline for the Squad Master project, ensuring all technical and functional aspects are well-defined. Updates may be made based on further development and testing phases.