



Software Requirements Specification (SRS) – Nafes Super Platform

Version	Date	Changes	Updated By
1.0	2025/5/25	Create the document	Dhiaaddin
1.1	2025/5/26	Add the Admin Panel	Dhiaaddin
1.2	2025/6/5	- Game ID store in the Profile Level - Add (3.3 F.) Scheduled Mode	Dhiaaddin

Contents

1. Introduction.....	2
1.1 Purpose.....	2
1.2 Scope.....	2
1.3 Definitions, Acronyms, and Abbreviations.....	3
1.4 References.....	3
1.5 Overview.....	3
2. Overall Description.....	4
2.1 Product Perspective.....	4
2.2 Product Functions.....	4
2.3 User Classes and Characteristics.....	4
2.4 Operating Environment.....	5
2.5 Design and Implementation Constraints.....	5
3. System Features.....	5
3.1 Unified Profile.....	5
3.2 Tournament Management (Do this later - Something might change).....	8
3.3 League Management.....	10
3.4 Admin Panel (Dashboard).....	12
3.8 Smart Rule and Eligibility Engines.....	13
3.9 Localization and Accessibility.....	14
3.10 Error Handling and UX Guidance.....	15
4. External Interface Requirements.....	15
4.1 User Interfaces (UI).....	15

5. System Architecture.....	17
5.1 Frontend Architecture.....	17
5.2 Backend Architecture.....	18
5.3 Database Design Principles.....	18
5.4 Security & Permissions Framework.....	19
6. Non-Functional Requirements.....	20
6.1 Performance.....	20
6.2 Scalability.....	20
6.3 Availability.....	20
6.4 Security.....	20
6.5 Maintainability.....	20
6.6 Compliance.....	21
7. Use Case Scenarios.....	21
7.1 Tournament Registration Flow (Player).....	21
7.2 Admin Creating a Multi-Stage Tournament.....	21
7.3 Resolving Match Result Conflicts.....	22
7.4 Solo Player Joins Prime Queue.....	22

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) outlines the functional and non-functional requirements for the **Super Platform**, a comprehensive esports tournament and event management system developed by **Nafes**. The goal is to establish a centralized, scalable platform capable of managing all esports initiatives, such as the **Saudi Prime League (Prime)**, **Saudi eLeague (SEL)**, and the official **SEF blog**, with modular customization, powerful admin tools, and immersive user experiences.

This document serves as the foundation for system design, development, testing, and deployment. It is intended for internal use by stakeholders including project managers, developers, UI/UX designers, QA testers, and client representatives.

1.2 Scope

The Super Platform consolidates player registration, team management, match scheduling, result tracking, leaderboard generation, blog publishing, and administrative oversight. It provides:

- A Unified player profile and identity management system.
 - Custom tournament engine (single-elim, double-elim, round robin, and battle royale)
 - Custom league matchmaking engine (solo queue and team queue)
 - Role-based admin control with fine-grained permission management.
 - A culturally sensitive, multilingual interface (Arabic/English).
 - Smart matchmaking and ranking algorithms using ELO-based logic.
 - Modular architecture allowing project-specific customization.
 - Integrate a Full Content Management System functionality for blogs and public announcements (e.g., wordpress).
 - A comprehensive, customizable admin dashboard covering all needs across projects.
-

1.3 Definitions, Acronyms, and Abbreviations

- **Nafes**: Esports technology provider and project lead.
- **SEF**: Saudi Esports Federation. (Client)

- **Prime:** Saudi Prime League. (Project)
 - **SEL:** Saudi eLeague. (Project)
 - **SBMM:** Skill-Based Matchmaking.
 - **ELO:** Rating system used to measure relative player skill.
 - **Unified Profile:** Centralized player/team identity across all SEF initiatives.
 - **Competitions:** Refers collectively to both Tournaments and Leagues.
-

1.4 References

- UI Design System: [Figma Project] ([Link](#))
 - Brand Guidelines and Compliance Policies ([Link](#))
-

1.5 Overview

The rest of this document details the complete system architecture, user workflows, interfaces, and requirements of the Super Platform. It includes core modules such as tournament management, leagues, Unified profiles, the blog system, and dashboard functionalities—alongside non-functional aspects like scalability, accessibility, and localization.

2. Overall Description

2.1 Product Perspective

The Super Platform is a modular web-based system designed to centralize all esports event operations under the Saudi Esports Federation. It is built as a single platform that hosts multiple projects, such as PRIME, SEL, and others, each running as independent instances but sharing core engines (e.g., leagues, tournaments, profile management,).

The system supports both admin and player-facing modules and is built to be thematically immersive, game-inspired, and highly adaptable to each competition's unique needs. Unlike existing esports tools, Super Platform is project-first, rule-flexible, culturally aware, and built to scale with the growth of esports in the Kingdom and beyond.

2.2 Product Functions

Key functionalities include:

- **Unified Profile System:** Single user account with verified identity, match history, and tournament eligibility.
 - **Tournament Engine:** Create any format (elimination, round robin, custom stages).
 - **League Engine:** Long-form competition management including Prime solo queue structure.
 - **Admin Dashboard:** Role-based access to manage players, teams, matches, blog content, and support.
 - **Smart Matchmaking:** SBMM with ELO-driven skill calibration.
 - **Content Management:** Full blog system with tagging, scheduling, SEO tools, and view metrics.
 - **Error & Verification Handling:** Clear user guidance and validations based on context.
 - **Support Center:** Ticket system with admin tools and user-facing FAQs.
 - **Localization:** Arabic/English support with dynamic RTL layout adaptation.
-

2.3 User Classes and Characteristics

- **Player (default)** : Registers, joins teams, competes in leagues/tournaments.
 - **Admin:** Platform user with full access and configurable permissions.
 - **Super Admin:** Can configure permissions and assign Admins.
 - **Admin**
 - **Operator:** Limited admin role for managing specific competition.
-

2.4 Operating Environment

- **Web-Based:** Fully accessible via all major modern web browsers.
- **Responsive UI:** Optimized for all screen sizes using mobile-first responsive design.
- **Frontend:** Built using React.js with TailwindCSS and Framer Motion for UI interactivity.

- **Backend:** Node.js with a scalable architecture supporting RESTful APIs (GraphQL-ready).
- **Database:** Hybrid storage using PostgreSQL for relational logic and MongoDB for flexible, dynamic content.
- **Time Zone Standardization:** All data and time-related operations are aligned to Saudi Arabia Time (GMT+3).

2.5 Design and Implementation Constraints

- **LTR and RTL Support:** The platform must support both Left-to-Right (English) and Right-to-Left (Arabic) layouts, dynamically switching based on language.
 - **Audit Logging:** All sensitive actions (e.g., admin updates, bans, content edits) must be logged with the responsible user ID, timestamp, and contextual data.
 - **Server-Side Critical Flows:** Core interactions must be validated and executed server-side to ensure consistency and prevent manipulation.
 - **IP Address Tracking:** Each user login must record the IP address (for internal auditing, not shown to users).
-

3. System Features

3.1 Unified Profile

3.1.1 Description

The Unified Profile system is the foundation of user identity within the Super Platform. It is a centralized system used across all competitions, leagues, and support services hosted under the Saudi Esports Federation.

3.1.2 Functional Requirements

- Users must register with **Layer 1 (mandatory) fields** to create account:
 - IP Address (auto-captured) (Not visible for user)
 - Profile ID (auto-generated) (Our unique identifier) (Not visible for user)
 - Username (text) (unique)
 - First Name (Text)
 - Last Name (Text)

- Date of Birth (Date picker)
- Gender (Male/Female)
- Email (No OTP verification for now)
- Password (strong rules enforced)
 - Be **at least 8 characters** long.
 - Include at least **one uppercase letter** (A-Z).
 - Include at least **one lowercase letter** (a-z).
 - Include at least **one number** (0-9).
 - Include at least **one special character** (e.g., ! @ # \$ % ^ & *).
- Phone (default country: Saudi Arabia) (No OTP verification for now)
- Nationality (default: Saudi Arabia)
- Role (dropdown):
 - Player “Default”
 - Referee
 - Couch
 - Team Manager
 - Club admin
 - Analyst
 - Caster
 - Host
 - Media outlet
 - Video Editor
 - Graphic Designer
 - Streamer
 - Frontend creator
 - Game developer
 - Tournament Organizer
 - Club Owner

- **Layer 2 (optional)** fields include: (Get saved on the profile level)

- Profile Picture (Upload Photo)
- Favorite Game (dropdown)
- Game ID in the Favorite Game (Text) (Unique)
- Game Role / Position (dropdown) depend on the game
- Social media handles
 - Discord (Username)
 - X <https://x.com/>(Username)
 - Instagram <https://www.instagram.com/>(Username)
 - TikTok <https://www.tiktok.com/@>(Username)
 - Facebook <https://www.facebook.com/>(Username)
 - Youtube <https://www.youtube.com/@>(Username)
 - Twitch <https://www.twitch.tv/>(Username)
 - Kick <https://kick.com/>(Username)
- **Layer 3 (competition-specific)** fields include:
 - Custom text box fields that only get saved on the tournament level
 - Check box for the predefined optional fields (Layer 2 fields) to become mandatory to join competitions

3.1.3 Behavioral Requirements

- If a user attempts to register for a tournament without completing the required profile data:
 - popup should appear prompting them to fill in the missing fields directly within the popup to continue registration.
- Specific competitions can make Layer 2/3 fields **mandatory** based on configuration.

3.2 Tournament Management (Do this later - Something might change)

3.2.1 Description

The Tournament Management Engine is a core feature of the Super Platform. It enables admins to create and manage tournaments of any structure—single elimination, double elimination, round robin, Swiss round and battle royale, or fully custom stages.

3.2.2 Functional Requirements

A. Tournament Creation & Configuration

- Admins/Operators can define:
 - Tournament Title (Text)
 - Game (dropdown, like: Cod, EA FC..)
 - Platform (dropdown, like: PS5, PC..)
 - Organizer/Partner/Project (Dropdown)
 - Open registration / Manual registration (Pick one)
 - Tournament Type (Solo / Team)
 - If team is selected ask for:
 - Minimum number of players in a team
 - Maximum number of players in a team
 - Max Participants (Number)
 - Registration Start (Date Picker)
 - Registration End Dates (Date Picker)
 - Description (Text)
 - Rules (rich text or uploaded PDF)
 - Custom Registration Fields (e.g., Discord ID, Student ID)

B. Stage Builder

- Visual UI for building tournament flows.
- Supported Stage Types:
 - Single Elimination
 - Double Elimination
 - Round Robin
 - Swiss Format
 - Battle Royale

- Custom Manual Stage
- Features:
 - Manual or automatic seeding
 - Real-time bracket visualization

C. Team and Player Eligibility

- Smart eligibility engine:
 - Enforce age, gender, nationality, etc.
 - Validate Layer 2/3 profile fields as needed
 - Example: "Only Saudi Female aged 18-25 can register"

D. Roster Management (for team tournaments)

- Validate roster size before allowing entry (e.g., 5v5 = 5 verified players).
- Rosters must be complete before tournament registration.
- Roster is editable until a configured "lock deadline."
- After lock, changes are disabled.

F. Match Generation & Linking

- Matches are auto-generated per stage rules or can be created manually
- Match linking ensures winners auto-advance (or be placed in lower bracket)

G. Prize Pool Management

- Shows prize structure with tiers and placement info
- Optional file or link upload for prize terms

3.2.3 Behavioral Requirements

- Registration closes automatically at the defined end time.
- Custom error handling for issues like:
 - "Tournament Full"
 - "Your team must have 5 players"

3.2.4 Non-Functional Notes

- All actions must be logged in the admin audit trail.
 - Localization (Arabic/English) for tournament names, rules, and alerts.
 - Responsive UI required for registration and brackets on mobile.
-

3.3 League Management

3.3.1 Description

League Management in the Super Platform supports long-running, ranked esports competitions with persistent participation. Unlike one-off tournaments, leagues may run over weeks or months and include solo or team-based formats. A key feature includes **Solo queue**, where individual players queue and get matched into temporary teams by the system.

Each league operates independently within its project (e.g., Prime, SEL), but shares core engines like matchmaking, result tracking, and ELO calibration.

The Super Platform's matchmaking and ranking system powers competitive integrity, especially in **Solo queue** leagues. It intelligently forms balanced teams from individual players using ELO-based logic and monitors player skill over time to produce accurate, evolving rankings.

This system supports real-time queue handling, match generation, and rating adjustments, and is modular enough to serve different game titles and competition formats.

3.3.2 Functional Requirements

A. League Creation & Configuration

- Admin can configure:
 - League Title (Text)
 - Project (e.g, Prime, SEL..) automatically chosen
 - Game (dropdown, e.g.,: Cod, EA FC..)
 - Platform (dropdown, e.g.,: PS5, PC..)
 - Format: 1v1, solo queue or party queue.

- Solo queue = Player join without team and create a temporarily team
- Party queue = Player must have a team to join
- If solo queue or party queue is selected: Number of players in a team
- Max number of matches per player (Checkbox) “Optional format”
- Queue Settings:
 - Always-on or Scheduled (specific days/times) **(More details later)**
- Qualifying line (e.g., top3)
- Prizepool (Text) “Can be left empty”
- Rules (rich text or uploaded PDF)
- Custom Registration Fields (e.g., Discord ID, Student ID) layer2 - layer3

C. Solo Queue Mode

- Player joins queue without team
- System builds temporary teams from queue entries
- Team skill = average of players' ELO
- Match result adjusts each player's ELO equally
- Prevents repeat team compositions in consecutive matches

E. Notifications

- Notify players when:
 - Queue opens/closes (if applicable)
 - They are matched (match found)
 - Weekly reports or summaries are published (by email)

F. Always-on or Scheduled Mode

The system must support two types of queue availability modes:

1. **Always On** – The queue is open 24/7 with no time restrictions.
2. **Scheduled** – The queue is only open on specific days and at specific times.

For the **Scheduled** mode, the admin must be able to configure:

- **Selected Days:** Choose one or more days of the week (e.g., Sunday, Monday, or all days).
- **Start Time:** The time when the queue becomes available on the selected day(s).
- **Finish Time:** The time when the queue closes on the selected day(s).

Examples:

- Queue open on **Sunday and Monday** from **9:00 AM to 9:00 PM**.
- Queue open on **Sunday and Monday** for the **entire day**.
- Queue open only on **Sunday** from **9:00 AM to 11:00 AM**.
- Queue open on **all days** from **9:00 AM to 11:00 AM**.
- Queue can also be set to run **all day** on selected days.

The system must allow combining both day and time configurations together, and it must apply these settings consistently across all relevant league operations.

3.4 Admin Panel (Dashboard)

3.4.1 Description

The Admin Panel is the operational brain of the Super Platform. It enables full lifecycle management of players, teams, competitions, blog posts, platform content, permissions, and support. It's modular, role-based, and project-specific, allowing separate control for Prime, SEL, or any other initiative under the same backend engine.

3.4.2 Functional Requirements

A. Role-Based Access Control

- Roles: Admin, Operator
- Permissions configurable per project
- Example:
 - Prime admin can approve rosters

- SEL operator can manage match data but not platform-wide settings

B. Dashboard Tabs & Use Cases

1. Overview

2. User Control

- Users
- Teams
- Banned
- Admin Access

3. General Control

- Games
- Partners (Prime, Saudi eLeague)

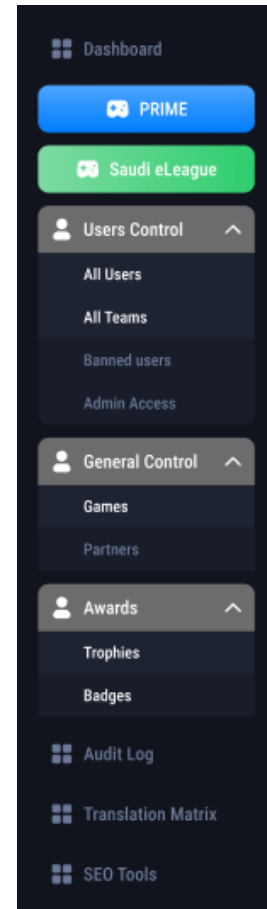
4. Awards

- Trophies
- Badges

5. Audit Log (optional)

6. Translation Matrix (optional)

7. SEO Tools (optional)



3.4.3 Behavioral Requirements

- Permissions must reflect immediately after role changes.
- All admin actions (e.g., match edits, user bans) are logged with timestamp and actor.
- Each project's dashboard is isolated unless explicitly merged (e.g., shared operators across Prime and SEL).

3.4.4 Non-Functional Notes

- Must support multilingual UI (admin can switch languages).
 - Audit logs viewable by master admins.
 - Dashboard must be responsive and optimized for widescreen desktop use.
 - Search, filters, and bulk actions are essential for scaling.
-

3.8 Smart Rule and Eligibility Engines

3.8.1 Description

These engines provide dynamic, customizable control over who can access what within the system. Rather than relying on hardcoded logic, they allow for flexible configuration of participant eligibility and admin permissions based on tournament type, user profile attributes, and organizational structure.

3.8.2 Functional Requirements

A. Smart Eligibility Engine (for Tournaments & Leagues)

- Admins can create rule sets for registration eligibility using a visual rule builder.
 - Rules can be based on:
 - Age (e.g., 18–25 only)
 - Gender
 - Nationality
 - Email domain
 - Multi-layer logic:
 - AND/OR condition trees
 - Nested conditions
 - Examples:
 - Player must be Saudi AND female
 - Player must under 25 years and have its email domain “edu.sa”
-

3.9 Localization and Accessibility

3.9.1 Description

The Super Platform is designed to be culturally aligned with its user base, primarily serving Arabic and English-speaking audiences. The interface dynamically adapts its layout, typography, and flow to match language and cultural expectations. It also ensures accessibility for users with different needs, from keyboard navigation to screen reader compatibility.

3.9.2 Functional Requirements

A. Multilingual Support

- Platform must support:
 - Arabic (RTL layout)
 - English (LTR layout)
- Language-specific UX adjustments:
 - Arabic headers in stylized calligraphy
 - Text alignment, flow, and form mirroring for RTL
- Automatic language detection on first visit, overrideable by user (optional)

All translations are managed via a centralized matrix or sheet that maps each interface term to its translations, editable directly through the admin dashboard for seamless updates.

 AMERICAN ENGLISH	 SPAIN	 ARGENTINA	 COLOMBIA
earrings	pendiente	aro	arete
women's underpants	bragas	bombachas	calzones
eye glasses	gafas	lentes	lentes
traffic jam	atasco	nudo	trancón
warehouse	almacén	depósito	bodega
license plate	matrícula	chapa	placa
zipper	cremallera	cierre	cierre/cremallera

The system is designed to support scalable multilingual integration, enabling administrators to easily add and configure new languages without development changes.



3.10 Error Handling and UX Guidance

3.10.1 Description

Clear, actionable error feedback is critical to the usability of a complex platform like the Super platform. This section defines standards for how the platform handles form validation, system errors, verification issues, and competition-specific edge cases.

3.10.2 Functional Requirements

A. Inline Form Validation

- All forms must:
 - field-specific errors *below or beside* the input
 - Use real-world language, e.g.:
 -  “Game ID is required for Valorant tournaments”
 -  “Field invalid”
- Form should not submit until all required inputs are corrected

B. Role-Specific Alerts

- Errors and warnings should vary by role
-

4. External Interface Requirements

4.1 User Interfaces (UI)

The Super Platform has two major UI domains:

A. Public & Player-Facing Interfaces

- Registration, Login, Profile Management
- Tournament and League Browsing
- Blog Articles and News
- Match Participation and Results
- Support and FAQ Access

B. Admin Dashboard Interfaces

- Modular tabs for managing players, tournaments, posts, settings, and support
- Project-specific dashboards (e.g., Prime, SEL)
- Role-specific visibility and controls

UX/UI Design Principles

- Game-inspired interface aesthetics

- Dark mode by default, responsive across devices
 - Language toggle with RTL/LTR switch
-

4.2 Hardware Interfaces

- Not applicable (web-only platform)
-

4.4 Communication Interfaces

- **Email (Transactional):** Verification, match updates, support tickets
 - **SMS (Optional):** Account verification, alerts
 - **Discord/Webhook Integration** (future scope): Match result sync, ticket notifications (later stage)
 - **Analytics:** Google Analytics, Google Tag manager.
-

5. System Architecture

5.1 Frontend Architecture

Framework & Libraries

- React.js with TailwindCSS for styling (open for recommendation)
- Framer Motion for smooth animations and transitions
- i18next for internationalization (Arabic/English support)
- Responsive design using mobile-first principles

Key Features

- Modular components (reusable across projects)
- Dynamic theming per project (e.g., Prime, SEL, Blog)
- Persistent sidebar + project-themed main area
- Optimized performance via code splitting and lazy loading
- Form validation with contextual error feedback

- Accessible (keyboard navigation, ARIA labels)
-

5.2 Backend Architecture

Core Technologies

- Node.js + Express for API layer
- PostgreSQL as the primary database
- Redis for caching, rate limiting, queue pooling
- Optional MongoDB for flexible content fields (e.g., blog metadata)

Architectural Principles

- REST API (GraphQL-compatible in future)
- Modular service-based organization (e.g., auth, tournaments, blog)
- Rate limiting and brute-force protection
- JWT-based stateless auth with refresh tokens

Scalability

- Horizontally scalable APIs (containerized)
 - Match queue and result processing optimized for async job workers
 - CDN delivery for assets (e.g., Cloudflare, AWS CloudFront)
-

5.3 Database Design Principles

Data Layers

- **Layer 1:** Core user data (validated on platform level)
- **Layer 2:** Game-related and optional profile fields
- **Layer 3:** Tournament-specific fields (project-defined)

Entities

- Users, Teams, Matches, Tournaments, Leagues, BlogPosts, Tickets
- Use UUIDs for IDs
- All timestamps stored and read in **GMT+3 (Saudi Time)**

Data Integrity

- Foreign key enforcement
 - Soft deletion and audit trail fields (createdBy, updatedAt, deletedAt)
-

5.4 Security & Permissions Framework

Authentication & Identity

- Passwords hashed with bcrypt
- Email and phone verification required
- Device/session management optional in future

Authorization

- Role-based access control:
 - Admins, Operators, Players (fully configurable)
- Rule-based permission matrix by project

Other Security Layers

- Input validation and sanitation
 - Secure file uploads (MIME checking, virus scanning)
 - Captcha and rate-limiting on public endpoints
 - HTTPS enforced across all environments
-

6. Non-Functional Requirements

6.1 Performance

- The platform must respond to user interactions within **300ms** for 95% of actions.
 - Match queues should process and form teams within **60 seconds** or less.
 - Admin dashboard must handle operations on datasets with **High number of records** (e.g., players, matches) without lag.
-

6.2 Scalability

- Backend must support burst traffic during major events (e.g., Prime qualifiers).
 - DB and queue layers must auto-scale (horizontal scaling via containerization).
 - Modular microservice-friendly structure for eventual decomposition.
-

6.3 Availability

- Target uptime of **99.9%** (excluding maintenance windows).
 - Health checks, retry logic, and fallback UIs must be in place for API failures.
 - Use load balancers and multiple instances to ensure availability during peak traffic.
-

6.4 Security

- Enforce HTTPS and secure cookie policies.
 - Use **bcrypt** or better for password storage.
 - Apply role-based access control for all admin views.
 - Run **regular vulnerability scans** and dependency audits.
 - All user data (especially personal identifiers) encrypted at rest and in transit.
-

6.5 Maintainability

- Code must follow clean architecture principles (e.g., separation of concerns).
 - Full API documentation generated and versioned.
 - Linting, formatting, and pre-commit hooks integrated in CI/CD.
-

6.6 Compliance

- GDPR/KSA data privacy compliance:
 - Clear user consent for data usage
 - Ability to export, delete, or update data upon request
- Log all changes to user data (who/when/what)

- Visible, accessible **privacy policy** on platform
-

7. Use Case Scenarios

7.1 Tournament Registration Flow (Player)

Actor: Registered Player

Flow:

1. Player browses available tournaments.
2. Selects a tournament and clicks "Register".
3. System checks profile completion and eligibility.
4. If incomplete, redirects to /profile/edit with required fields highlighted.
5. Upon completion, redirects back to registration and confirms entry.

Edge Cases:

- Player is underage → Blocked with custom message.
 - Player hasn't verified phone → Registration blocked until verification.
-

7.2 Admin Creating a Multi-Stage Tournament

Actor: Project Admin

Flow:

1. Admin logs in and goes to Competitions > Create Tournament.
2. Inputs general details (game, rules, registration deadline, etc.).
3. Uses stage builder to add Round Robin, then Elimination stages.
4. Publishes tournament for public registration.
5. Monitors registrations and configures seeding manually.

Edge Cases:

- Admin adds duplicate players → System flags and blocks stage setup.
-

7.3 Resolving Match Result Conflicts

Actors: Two Team Captains + Admin

Flow:

1. Each captain submits different match results with screenshots.
 2. System flags conflict and marks match as “Disputed”.
 3. Notifies both teams and opens score edit window.
 4. If scores match after correction → Conflict closed.
 5. If not resolved → Admin receives alert and manually decides winner.
-

7.4 Solo Player Joins Prime Queue

Actor: Individual Player

Flow:

1. Player enters Prime solo queue during open hours.
2. System checks ELO, previous matches, and queue conditions.
3. Team formed using average ELO of matched players.
4. Match is created and players notified.
5. After game, ELO is updated per result.

Edge Cases:

- Player leaves queue before match → Removed with log
 - System fails to match → Retry logic triggers after timeout
-