جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

Spark

IT 496: Graduation Project Report
Product Release-1

Prepared by
Mariam Alahmed, 443204628
Nora Fisal Albyahi, 443200479
Raghad Sultan Aldajani, 443200551
Aljwharah Ali Alhowidy, 443201015

Supervised by
Dr. Sharefah Alghamdi

First Semester -1447
Fall 2025

Table of Contents

List of Tables:

List of Figures:

1 Introduction

The esports industry has emerged as one of the most rapidly growing sectors in the digital entertainment domain, attracting millions of players and audiences worldwide. In recent years, esports has witnessed remarkable development, strongly supported by initiatives aimed at enhancing digital transformation and creating a sustainable gaming ecosystem [1].

However, despite this progress, many talented players continue to face difficulties in gaining visibility and accessing professional opportunities. Teams and organizers also encounter challenges in identifying and evaluating players due to the lack of reliable, data-driven platforms [4]. This gap limits the growth of the esports community and slows the recognition of emerging talents. Addressing this problem through a technological solution is of great significance both locally and globally. Locally, it empowers players, promotes fair opportunities, and supports the national agenda for digital entertainment and economic diversification. Globally, it contributes to the development of standardized, intelligent systems for player evaluation and fosters cross-border opportunities in the international esports industry.

In this document, we present the overall problem, propose an AI-powered solution, outline the project's vision and roadmap, define its objectives and scope, and specify the tools, resources, and roles required to ensure successful implementation.

1.1 The Problem

The esports industry in Saudi Arabia is growing rapidly, driven by national initiatives and the rising interest of millions of players across the Kingdom [1]. Despite this growth, there remains a significant gap between talented players and professional opportunities. Many skilled individuals struggle to gain visibility or present their true performance level, largely due to the absence of localized platforms that provide fair recognition and objective skill evaluation [4].

This gap is especially evident when comparing the millions of amateur and semi-professional players with the very limited number of officially recognized professionals, estimated at around 100 full-time players in Saudi Arabia. Studies show that aspiring competitors frequently report challenges such as limited local competitions, unclear pathways to professionalism, and social stigma surrounding esports careers [3]. These barriers make it difficult for players to progress and for teams to reliably assess new talent.

Moreover, while participation in esports is widespread, only a small fraction of players succeed in transitioning to higher competitive levels. This highlights the lack of structured, data-driven tools that can evaluate player performance and support informed team formation and talent discovery [5].

Therefore, there is a clear need for a trusted, organized, and intelligent local platform that allows players to showcase their skills, gain visibility, and connect with teams and tournaments in a reliable and professional manner.

1.2 The Solution

SPARK focuses on addressing this gap by developing an intelligent platform that allows players to fairly showcase their skills and connect with other players to form teams in an organized and reliable manner. SPARK analyses team composition and predicts the probability of winning after forming a team, enabling teams to try alternative formations and improve their chances before entering competitions [5]. Additionally, SPARK offers a structured competitive environment that meets the growing demand for tournaments and helps present esports in a more professional light [6].

This solution directly contributes to addressing the issue of limited visibility and lack of organized opportunities for players [7]. SPARK enables both amateur and semi-professional players to gain fair recognition and supports the discovery of new talent by teams and organizers [7]. Furthermore, it paves the way for greater societal recognition of esports as a legitimate professional pathway. In the long term, SPARK strengthens the infrastructure for esports, aligns with the goals of Vision 2030 in the digital entertainment sector, and reduces the gap between aspiring players and professional pathways [7].

1.3 Product

1.3.1 Product Vision

Product Vision:

For Saudi esports players, teams, and organizers,

Who seek greater opportunities to gain visibility, join suitable rosters, and participate in professional tournaments,

The SPARK **is an** AI-powered esports platform,

That helps teams discover the right players, empowers players to showcase their skills, and provides reliable local competitive opportunities,

Unlike other competitors such as saudiesports.sa ,

Our product leverages intelligent insights to support decision-making, and richer player profiles, ensuring fair recognition and stronger team formations.

1.3.2 Product Roadmap

In this section, we will outline "SPARK" app roadmap. Figure 1-1 shows the product's development and delivery timeline.

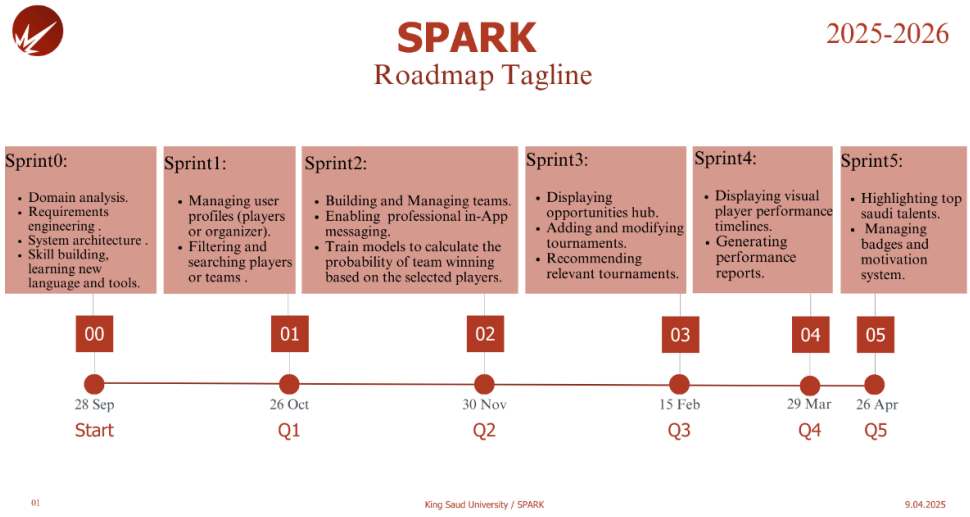


Figure 1: SPARK Roadmap

1.3.3 Objectives

In this section, we will outline the product, project, and learning objectives associated with the development of the "SPARK" platform. These objectives define the key goals and desired outcomes we aim to achieve by the completion of the project.

- **Product (customer focus-value)**:

These objectives focus on solving problems and delivering benefits through key features:

  - These objectives focus on solving problems and delivering benefits through key features:
  - Allow both players and organizers to manage their own profiles.
  - Facilitate talent discovery and connect players and teams with opportunities.
  - Enable players to create and manage their own teams within the platform
  - Analyze players' performance using AI to calculate the probability of a team winning.
  - Provide an integrated chat system that enables players to communicate directly, enhancing coordination and team bonding.
  - Allow players to link their in-game accounts by submitting their game tag, enabling the platform to automatically fetch and store their gameplay statistics for accurate performance tracking.

- **Project (solution focus-plan):**

These objectives define the tasks and stages necessary to complete the project:

  - Conduct interviews and surveys with esports players, teams, and organizers to gather requirements and feedback.
  - Collect and preprocess player performance data from the API such as Riot Games API [11].
  - Analyze and clean the collected data to prepare it for AI model training, then train models to calculate the probability of a team winning based on the selected players.
  - Identify user needs and ensure the mobile app features (profiles, teams, tournaments, reports) meet those needs.
  - Design user-friendly mobile interfaces using Figma [17] for both players and organizers.

- Develop the mobile application in Flutter [12], implementing:
  - Player and organizer profiles.
  - Team building and management.
  - In-app messaging.
- Create and connect a database (Firebase [15]/NoSQL [16]) to store user accounts, match history, teams, and tournament details.
- Test the application thoroughly to ensure functionality, usability, and accurate AI predictions.
- Deploy the application on devices and gather user feedback for improvements.

- **Learning (student focus):**

These objectives focus on new skills and tools to be learned:

- Gain hands-on experience in mobile application development using Flutter [12].
- Apply Artificial Intelligence techniques to real-world esports data to calculate team win probabilities.
- Gain practical skills in working with external APIs, such as the Riot Games API [11], for data collection and integration.
- Strengthen knowledge of cloud-based databases using Firebase [15]/NoSQL [16] for real-time data management.
- Improve agile project management and teamwork skills [19]by working in sprints and collaborating on a multi-module application.
- Enhance problem-solving and critical thinking [20]by addressing challenges in data preprocessing, AI modeling, and mobile app integration.

1.3.4 Scope

In this section, we will define the limitations of our platform and outline what is outside the scope of this project, which may be considered for future development.

The "Spark" platform is a mobile application for esports players, teams, and organizers. It allows profile management, team building, tournament browsing and applications, and AI-powered team win probability predictions. The AI model for predicting team win probability

will be applied exclusively to the game *League of Legends. The* app includes performance reports, visual timelines, and gamification features. It supports English on mobile devices using Android Flutter, with future expansion to additional languages or other platforms possible. Features such as live streaming, web support, third-party payments, or player tag verification are outside the current scope. Player tag verification will be considered in the future, potentially in collaboration with an external company to ensure secure and reliable validation.

1.3.5 Hardware/Software Tools and Cost

This section outlines the hardware and software resources required to develop and implement the **SPARK application**. It includes all necessary development environments, frameworks, APIs, and supporting tools for building, testing, and deploying the solution. Each tool's purpose and associated cost are summarized in Table 1.

Hardware Tools	
Name and Description	Cost
Computers for software development	Available
An android smartphone for testing a mobile application	500-900 SAR
Software Tools	
Name and Description	Cost
Python - [8] with libraries such as pandas, scikit-learn [9], and NumPy [10] – used for AI model training, data analysis, and feature engineering.	Free
API such as Riot API- [11] provides access to player match history, performance data, and statistics.	Depends on the chosen API
Flutter - [12] open-source frameworks for building cross-platform mobile applications from a single codebase.	Free

GitHub - [13] a version control and collaboration platform for managing the source code.	Free
Jira - [14] agile project management tools for task tracking and sprint planning.	Free
Database Management System – Firebase [15] / NoSQL [16] for storing user accounts, and tournament details	Free
Figma [17] cloud-based design tool for creating UI wireframes and prototypes.	Free

Table 1: Hardware/Software Tools and Cost

1.4 Scrum Team

1.4.1 Skill Set Requirements

This section identifies the technical skills required for developing the **SPARK application**, the team's initial proficiency levels at the start of the project, and the plan to address any skill gaps.

Table 2 provides an overview.

Technical Skill Required	What is the current level of the team (*beginner-intermediate- advanced*) for each skill? How will the gap be bridged? (*if necessary*) Learning plan
Python Programming	Intermediate – Team members had prior experience using Python for data processing and scripting.
Machine Learning	Intermediate – The team was familiar with supervised ML concepts and model training workflows.

Flutter	Beginner – Basic understanding of UI layout; additional practice was planned using documentation and official samples.
API Integration	Beginner – The team had limited prior experience; early tasks focused on connecting to RESTful APIs.
Front-End Development	Intermediate – Team members had experience with UI/UX prototyping and responsive design fundamentals.
Back-End Development	Beginner – Initial exposure to server-side logic and database operations, with improvement planned through hands-on implementation.
Data Collection & Visualization	Intermediate – The team was capable of handling datasets, performing analysis, and building visualizations using Python libraries.

Table 2:Skill Set Requirements

1.4.1.1 Learning

The team has worked on developing the required skills in a balanced manner, combining both theoretical learning and practical application. In terms of data collection and API integration, the team used the Riot Games API to successfully retrieve and initially process player and match data.

Regarding AI model development, the team carried out the full workflow of training an ML classifier. Using Python along with NumPy and scikit-learn, the dataset was cleaned, encoded, and structured to match the features needed for prediction. The data was then split into training and testing sets to ensure proper model evaluation. A Random Forest classifier was trained to predict team win probability based on team composition and player-level statistics. The team also conducted multiple experiments by adjusting model parameters, comparing performance metrics, and validating the model using accuracy and confusion matrix results. These steps allowed the team to understand how different features impact the final prediction and to identify the best-performing model variation.

For interface design, Figma was used to develop wireframes and test the user flow, with initial screens prepared for later implementation in Flutter. An ERD was created to document the database design based on the project requirements; however, the practical integration with Firebase/NoSQL has not yet been completed.

In addition, the team gained practical experience with project management tools such as Jira and GitHub to organize tasks, manage branches, and track progress. These efforts demonstrate that the team has gone beyond theoretical learning, engaging deeply in both the technical and organizational aspects of the project, which has helped build a strong foundation for completing the upcoming phases efficiently.

1.4.2 Roles and Responsibilities

Scrum Team	
Product Owner:	Dr. Sharefah Alghamdi
Developers:	Nora Fisal Albyahi Raghad Sultan Aldajani

	Mariam Alahmed Aljwharah Alhwiedy
Scrum Master (SM):	Dr. Sharefah Alghamdi
Stakeholders:	The Examiners Committee at King Saud University, Saudi esports players and teams.

Table 3: Roles and responsibilities

2 Background

Esports are growing rapidly worldwide, combining the excitement of sports with the strategy of video games. To understand the motivation for SPARK, this section introduces key concepts: what esports are, which games dominate the scene, the role structure in multiplayer games, the use of data and machine learning models, particularly tree-based ensemble methods such as Random Forest, which can handle nonlinear patterns and complex feature interactions.

2.1 Esports and Their Ecosystem

Esports are organized, competitive video-game contests played at amateur, semi-professional, and professional levels. Like traditional sports, they involve players, teams, coaches, organizers, broadcasters, and fans. Global tournaments now attract millions of viewers and significant sponsorship.

Popular game genres in esports include:

- **MOBA (Multiplayer Online Battle Arena)** — 5v5 team-based strategy games such as *League of Legends* and *Dota 2*. These emphasize roles, teamwork, and objectives spread across a map.
- **FPS (First-Person Shooter)** — titles such as *CS:GO* or *Valorant*, where teams compete in round-based shooting matches with tactical coordination.
- **Sports simulations** — such as *FIFA* or *NBA 2K*, which replicate real sports through digital competition.

In Saudi Arabia, esports are recognized under Vision 2030 as part of the digital economy and youth engagement strategy. The Saudi Esports Federation, established in 2017, hosts national and regional events, while Riyadh became home to the Esports World Cup, which was first announced in 2023 and officially launched in 2024.

2.2 Roles and Team Formation in MOBAs

In MOBAs like *League of Legends*, each team has five specialized roles:

- **Top** — durable solo laner, often absorbing pressure.
- **Jungle** — roams the map, controls objectives, creates surprise attacks.
- **Mid** — central playmaker with high flexibility.
- **ADC (Attack Damage Carry)** long-range damage dealer, strong in later stages.

- **Support** — vision control, protection, and initiation.

Each role has different responsibilities, so evaluating players without considering role leads to unfair comparisons. For example, "gold earned" is meaningful for an ADC but less so for a Support. Hence, role awareness is a cornerstone of modern esports analytics.

2.3 Data and Analytics in Esports

Game APIs (like the Riot Games API) expose structured data: match history, player stats, and outcomes. From this, analysts compute rate-based features such as:

- **KDA** = (Kills + Assists) ÷ max(1, Deaths).
- **CS/min** = farming efficiency.
- **Gold/min** = economy growth.
- **Vision score** = map control through wards/vision.
- **Kill participation (KP)** = share of team kills a player contributed to.

These metrics allow fair comparisons across games of different lengths and help track performance trends over time.

2.4 Machine Learning in Prediction

Machine learning allows computers to learn patterns from past data and use them to make informed predictions about new situations. In SPARK, the concept is applied to recognize which team and player patterns are typically linked to winning or losing in esports.

One widely used technique for this kind of task is the Random Forest model. A Random Forest is a machine-learning method that builds many simple decision trees, where each tree makes its own small prediction. The model then combines the results of all trees to produce a more reliable final prediction. This approach works well for esports because match outcomes depend on many interacting factors, such as kills, gold, farm, vision, and team coordination, and Random Forests are good at capturing these complex relationships without requiring deep mathematical assumptions.

3 Literature Review

This chapter reviews prior research relevant to SPARK, focusing on win/loss prediction in team esports. By analyzing key studies, we identify methods and insights that guide our system design.

3.1 Win/Loss Prediction in Team Esports

Win/loss prediction in multiplayer online battle arena (MOBA) games such as League of Legends (LoL) and Dota 2 has been extensively explored using a variety of machine learning techniques. Costa et al. [18] analyzed the champion picks and bans phase in LoL and demonstrated that tree based ensemble models, particularly Random Forest, achieve strong predictive performance using only pre-match drafting information. Their findings highlight the importance of team composition, role distribution, and champion interactions in shaping match outcomes.

Lin [19] further examined win/loss modeling in LoL by comparing pre-match and in-game features. While in-game signals provide direct indicators of momentum, Lin showed that aggregated pre-match indicators, such as historical KDA, gold efficiency, lane performance averages, and role consistency, still offer substantial predictive value when early decisions must be made before gameplay begins.

In Dota 2, Kinkade et al. [20] evaluated both “rich” (detailed) and “reduced” (simplified) feature sets and demonstrated that supervised learning models, including Random Forest, Naive Bayes, and Support Vector Machines, can effectively predict match outcomes. Their results emphasize the trade-off between computational efficiency and feature completeness.

Johansson [21] extended this direction by applying sequence based neural models (LSTMs) to draft ordering, showing that the temporal sequence of draft picks contributes meaningful predictive structure beyond individual hero selections. This reinforces the role of team composition synergy in determining the likelihood of winning.

Beyond specific machine learning models, several probabilistic frameworks provide useful foundations for interpreting match predictions. The Bradley–Terry model [22] and the Elo rating system [26] formalize how relative strengths between competitors can be quantified, offering theoretical grounding for understanding win probability outputs. Additionally, prior research highlights the need for probability calibration to ensure that predicted probabilities

correspond to real world frequencies. Techniques such as Platt scaling [23], isotonic regression [25], and evaluation metrics like the Brier score [24] provide established methods for improving probability reliability when necessary.

Collectively, these studies demonstrate that:

1. Pre-match data alone can be predictive, even in the absence of live in-game events.
2. Tree based ensemble models such as Random Forest are well suited for capturing nonlinear interactions in team based esports.
3. Role aware and team level features, not just individual player metrics, are crucial for accurate outcome prediction.
4. Probabilistic frameworks and calibration techniques offer tools for improving the interpretability and reliability of predicted win probabilities.

These insights closely align with SPARK's approach of using pre-match, role specific, and team aggregated indicators to evaluate the relative strength of different player lineups.

3.2 Relevance to SPARK

Taken together, these strands provide two foundations for SPARK. Research on win and loss prediction supports the use of Random Forest on pre-match, role aware, and rate normalized features to evaluate team strength before gameplay begins. Prior studies also highlight the importance of role interactions and team composition, which aligns with SPARK's focus on analyzing complete player lineups rather than individual metrics alone. By combining these insights, SPARK positions itself as a team oriented platform that offers practical, data driven support for lineup evaluation and team formation in its first release.

3.3 Competitive Product Analysis

This subsection reviews existing platforms to understand what they offer and what is missing in the esports ecosystem. We compare eight key features: player profiles, filtering and search, team management, in-app messaging, AI-based win prediction, tournaments, badges, and top-talent highlighting to position SPARK clearly within the competitive landscape.

Figure 2: OP.GG logo

1. OP.GG

OP.GG is one of the leading platforms specialized in tracking player statistics and analyzing their performance in competitive games in general. The platform is well-known for its large database and tools that help players review their results, compare their performance with others, and see their ranking on the server or within the game. [27]

Features:

- Player Profiles & Statistics: Provides comprehensive profiles including match history, win rates, and performance analysis by characters or roles. These statistics help players review their performance and improve their gameplay strategies.
- Leaderboards: Collects and analyses match data, then ranks players based on performance indicators such as KDA (Kills/Deaths/Assists), win rates, and overall team impact, making it easy to identify the top players on the server.
- Support for Multiple Games (Wide): Not limited to a specific game, but extends to support various competitive games.

Differentiation:

- Strengths: Large database, detailed analysis for each role within the team, accurate metrics like KDA, ability to track progress across multiple matches, and resource/performance analysis by game phase.
- Weaknesses: Interface can be difficult for new users, and lacks interactive features like chat, achievements, or pre-match predictions offered by other platforms.

Figure 3: Kafugames logo

2. Kafugames

Kafu Games is a Saudi platform specialized in organizing and managing e-sports tournaments. The platform focuses on providing a comprehensive experience for players and organizers through tournament management tools, support for a variety of games, and fostering community interaction. [28]

Features:

- Tournament Management: Provides easy-to-use tools for managing teams, creating schedules, and tracking results smoothly.
- Support for Various Games: The platform supports a wide range of competitive games.
- User-Friendly Experience: Intuitive interface, instant notifications, and customizable profiles.
- Interactive Community: Allows players to interact with each other and follow competitions directly.

Differentiation:

- Strengths: Strong focus on tournament organization, easy-to-use interface, and an active, engaging community.
- Weaknesses: Limited expansion beyond the region, less advanced statistical analysis compared to platforms relying on large databases, and does not provide detailed player performance analytics

Figure 4: Saudiesport logo

3. Saudiesports.sa

Saudi Esports is the official platform responsible for developing and organizing e-sports in Saudi Arabia. The platform focuses on hosting regional and international tournaments, developing talent, and fostering community interaction in a professional environment. [29]

Features:

- Tournament Management: Hosts major local and regional tournaments, with professional tools to manage teams, schedules, and match results.
- Talent Development: Offers specialized training programs in coaching, refereeing, and management, in collaboration with global academies.
- Strategic Partnerships: Collaborates with major companies to enhance digital infrastructure and provide opportunities for local talent.
- Professional Competitive Environment: Provides modern facilities for hosting tournaments and competitions, such as SEF Arena.

Differentiation:

- Strengths: Organizes regional and international tournaments, professional training programs, strategic partnerships with major companies, and modern facilities.
- Weaknesses: Lacks advanced AI-based tools and predictive analytics, does not provide detailed player performance analysis, and mainly focuses on large tournaments rather than offering advanced management or performance tools for beginners and smaller local competitions.

Feature / Platform	OP.GG	kafugames	Saudiesports.sa	SPARK
Type of System	Web & App	Web & App	Web	App
OS Support	All, IOS, Android	All, IOS, Android	All	Android
Player Profile Management & Statistics	✓	✓	✓	✓
Filtering and searching players or teams	✓	✓	✓	✓
Building and Managing teams	✓		✓	✓
App messaging				✓
Predicts the probability of a team winning using AI				✓
Tournaments & Opportunities		✓	✓	✓
Badges / Achievements			✓	✓
Highlighting top talents	✓	✓	✓	✓

Table 4: Competitive Product Analysis summary

After analyzing multiple e-sports platforms, it becomes clear that while they share core functionalities, such as player profiles, basic statistics, team discovery, and tournament participation, they differ significantly in the depth of their tools and the advanced capabilities they offer. Some platforms focus on community interaction or large-scale tournaments, while others, like SPARK, introduce AI-driven features that enhance prediction, analysis, and talent identification beyond standard platform functionalities (see Table 4).

Comparison:

SPARK offers a comprehensive experience that combines ease of use, social interaction, and intelligent performance analysis. The platform maintains a balance between organizational functionality and analytical depth through features such as an AI-based model that predicts team win probability, an in-app chat system, and detailed performance tracking for players.

In comparison with **OP.GG**, the latter has a larger database and provides more in-depth analytics across a wider range of competitive games, giving it a clear advantage in advanced statistics and data coverage. However, **SPARK** is also strong in analysis by offering accurate and simplified insights that help players easily understand their performance. Moreover, **SPARK** outperforms **OP.GG** in terms of user interface design, offering a cleaner, more interactive, and user-friendly experience compared to **OP.GG**, which is often seen as complex and visually crowded. Compared to **Kafu Games**, which organizes frequent small- to medium-scale tournaments using its professional management tools, **SPARK** distinguishes itself through an enhanced player experience. It provides more detailed individual performance insights, an AI-driven model for team win prediction, and an in-app chat feature that promotes communication among players, making the overall experience more interactive and socially engaging. As for Saudi Esports, the platform leads in professional infrastructure, offering training programs for talent development and organizing major regional and international tournaments with institutional support. However, **SPARK** stands out for being more socially engaging and user-friendly, enabling players to communicate through built-in chat, access AI-based win probability tools, and view comprehensive performance insights , features that Saudi Esports currently lacks.

Overall, **SPARK** can be seen as a well-balanced platform that bridges analytics, social interaction, and user experience. Its blend of simplicity, intelligence, and interaction makes it an appealing choice for players and organizers who seek a modern and dynamic esports environment without unnecessary technical complexity.

4 System Requirements

4.1 System Users

The **SPARK** platform targets two main categories of users, each with general characteristics in terms of educational level, experience, and technical expertise:

1. Players

- **Educational Level**: Mostly high school students or above.
- **Experience**: They possess moderate to strong experience in competitive gaming but lack professional tools to showcase their skills.
- **Technical Expertise**: Proficient in using smartphones, interactive applications, and online gaming platforms. The application provides players with detailed profiles, accurate statistics, and opportunities to join teams and tournaments, enabling them to present their skills professionally.

2. Organizers

- **Educational Level**: Typically university graduates in technical or administrative fields.
- **Experience**: Experienced in organizing local or regional esports events and tournaments.
- **Technical Expertise**: Moderate, requiring simple and intuitive interfaces to manage tournaments, publish announcements, and monitor teams and players. The application provides organizers with a dashboard to create tournaments, track registrations, and communicate with players and teams in an organized and efficient manner.

Overall, the SPARK platform is designed to be a comprehensive and user-friendly solution that meets the needs of all stakeholders in the esports ecosystem, enhancing visibility, collaboration, and sustainable growth.

4.2 Requirements Elicitation and Analysis

We conducted five interviews: three with players (each including seven questions) and two with organizers (each including six questions), aiming to explore the challenges they face in participating in or managing gaming tournaments, how they track performance, and their expectations for a better platform. Several common challenges emerged, including difficulties in finding suitable teams, late notifications of tournaments, communication issues, and registration hurdles, along with insights into how a modern application could improve the experience. The full list of interview and survey questions is provided in the Appendix section 10.

4.2.1 Interviews

In the first player interview, Omar Salem mentioned that proving himself to teams is difficult, especially when he doesn't have a track record. He sometimes misses tournaments because he finds out about them too late, and registering for events can take a long time. He has tried platforms like Saudiesports.sa and kafugames, but they provide only basic profiles and stats. He prefers performance to be displayed using a mix of numbers and visual charts, which would help teams understand his progress and compare him with other players. Full match history is extremely important for building trust, and direct communication with teams or organizers would save a lot of time. He also finds motivational features like badges useful because they give him goals beyond winning matches.

The second player interview was with Khalid Al-Mutairi. He shared that it is challenging to show he is a good player, and sometimes he gets ignored because he doesn't have much history. Signing up for tournaments can be confusing. He has used Saudiesports but finds it lacks detailed stats, charts, and performance comparisons. He prefers a combination of charts, numerical stats, and rankings compared to peers. Full match history is extremely important for him, and he values a built-in chat system for easier coordination. Motivational features are meaningful if they represent real achievements.

In the third player interview, Reem Al-Zahrani explained that sometimes registration is complicated or the information is unclear, but well-organized teams make it easier. She occasionally uses platforms to manage her profile, but she focuses more on overall skill

visibility rather than detailed stats. She prefers a balanced display of key numbers and visual charts showing trends over time. Full match history is somewhat important, while badges and achievements are nice but not essential. She is fine with communication happening through other channels if needed.

For the organizers, the first interview was with Sara AlMutairi. She said that communication is the hardest part of organizing tournaments because players contact them through different platforms, and it's difficult to keep everything in one place. They use Battlefy for registration, but she still needs to double-check player details manually. Evaluating players is usually done by looking at their profiles and past matches. She thinks it would be very helpful if the platform could show suggestions based on past performance and stats. Promotion is expensive and reaching new players is difficult. She needs performance reports after tournaments, highlighting top performers, win rates, and key moments.

The second organizer interview was with Fahad AlHarbi. He explained that scheduling is the biggest challenge because players often miss matches, and rescheduling causes delays. Verifying IDs is also a hassle. He usually relies on past matches to evaluate skill, but it can be hard if profiles lack detail. Suggestions from the platform based on stats would make organizing matches easier. Reaching the right audience is difficult, and he also wants performance reports during and after tournaments, including consistency, win rates, and head-to-head comparisons.

Overall, the interviews show that both players and organizers face challenges in communication, registration, and performance evaluation. Players want professional profiles, full match history, visual progress tracking, motivational features, and easy communication. Organizers want help in evaluating players, suggestions for suitable teams, and clear performance reports. Both groups are looking for a platform that makes tournament participation and management more efficient and transparent.

4.2.2 Survey

In parallel, a survey with 50 responses provides clearer insight into the needs of Spark's community. Most respondents are players (84%) with a near-even gender split (52% male, 48% female), predominantly aged 18–24 (68%). Overall, the ease of finding local tournaments is rated around average, with responses clustering in the middle. The most cited challenges are

difficulty joining teams (40%) and lack of available opportunities (30%), followed by limited information (18%). Interest in key features is high, especially tournament recommendations by skill level (60%), player/team search (58%), and a profile showcasing statistics (52%). There is also strong demand for a local leaderboard (96%) and solid motivation from rewards or digital badges (82%). The importance of AI for estimating winning probability is moderate but positive, indicating openness to AI provided it delivers practical value. These findings suggest Spark should focus on improving discovery and player, team matching, offering reliable tournament information, rich performance profiles, and motivational systems.

4.3 User Interactions

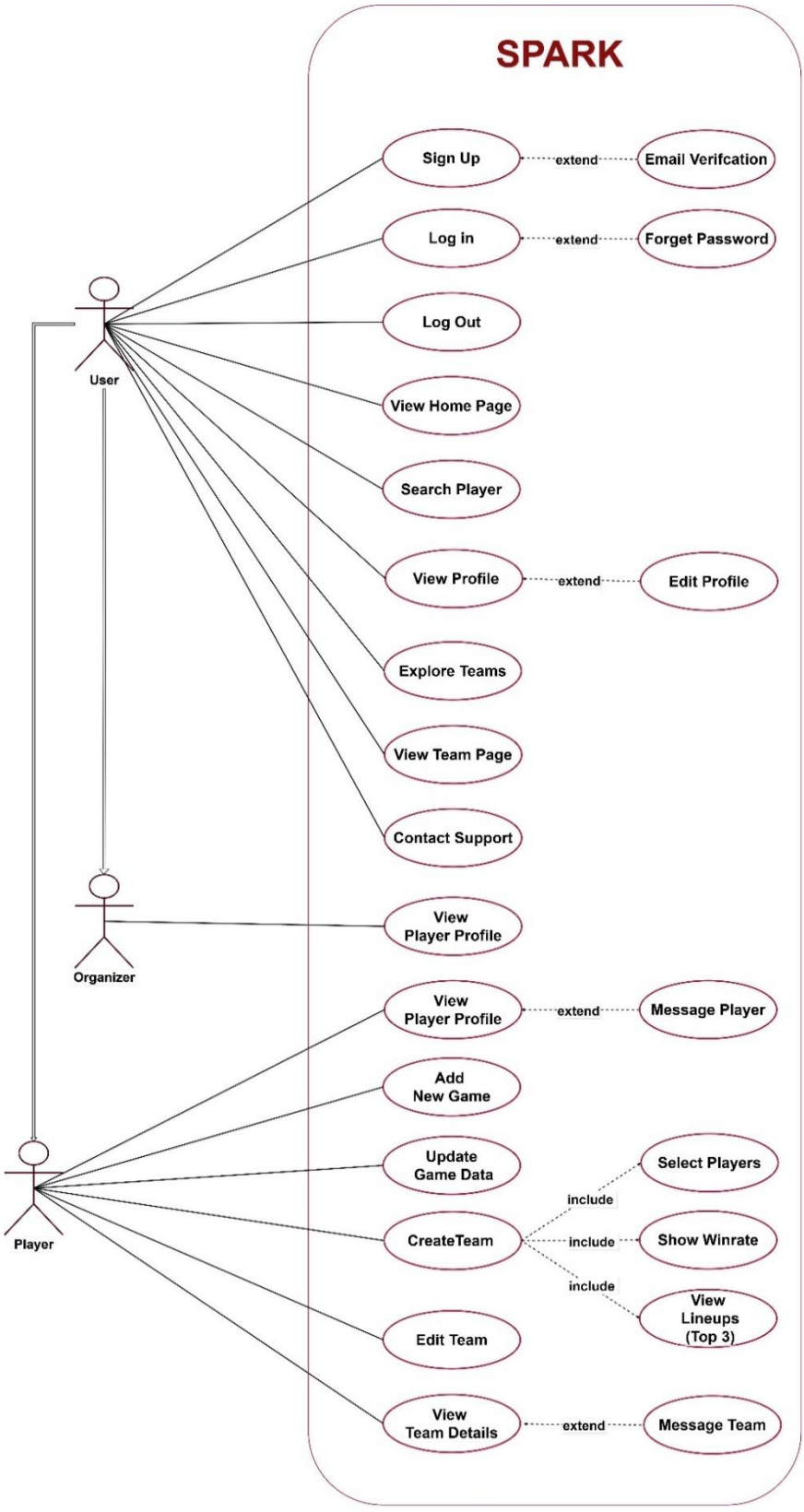


Figure 5 : Use Case Diagram

The Use Case Diagram provides a high-level visual representation of the interactions between SPARK's users and the system. It highlights the primary actors, Players and Organizers, and their main goals or tasks within the platform. The diagram illustrates how users engage with key functionalities such as profile management, team building, tournament adding and maneging, and win-probability insights, offering a clear overview of system usage without going into implementation details.

4.4 Product Backlog

The product backlog for the SPARK application is organized into two main categories: functional and non-functional requirements listed in Table 5; The functional requirements outline the core features and user interactions essential to the app's operation. The non-functional requirements describe the quality attributes and performance expectations that support the system's reliability and overall user experience. Together, these categories provide a clear structure for planning, development, and evaluation throughout the project lifecycle.

ID	PBI	Size	Type	Status	Acceptance Criteria
1	*As a user (player or organizer), I want to sign up with my email and password so that I can create an account and access the application's features.*	2	Feature	Done	1. As a nonregistered user (player or organizer), if I go to the sign-up page and enter my email and a password longer than 8 characters, then click the "Sign Up" button, I should be able to create an account and access the application. 2. If I try to sign up using an email that is already registered, I should see an alert telling me that the

					email is already in use, and the account should not be created. 3. If I leave any required field empty or enter invalid information, I should see an alert telling me to complete the required fields correctly before creating the account.
2	*As a user (player or organizer), I want to sign in with my email and password so that I can access my account and the application's features.*	2	Feature	Done	1. As a registered user (player or organizer), if I go to the login page and enter my email and correct password, then click the "Sign In" button, I should be able to access my account and the application. 2. If I enter an incorrect email or password, I should see an alert indicating that the login credentials are invalid. 3. If I leave any required field empty, I should

					see an alert indicating that all required fields must be completed before signing in.
3	*As a user (player or organizer), I want to log out of my account so that I can securely end my session and protect my account.*	2	Feature	Done	1. As a logged-in user (player or organizer), if I click the "Log Out" button, I should be signed out of the application immediately. 2. After logging out, if I try to access protected pages without signing in again, I should be redirected to the login page. 3. The session data should be cleared after logging out to ensure account security.
4	*As a player, I want to **edit** my account information so that I can update my personal details and keep*	3	Feature	Done	1. As a player, I should be able to update my name, age, profile picture, city and game from my account settings page.

	my profile up-to-date.				2. After saving changes, the updated information should be immediately reflected in my profile. 3. If any required field is missing or invalid (e.g., age not a number), I should see an alert prompting me to correct it.
5	*As an organizer, I want to **edit** my account information so that I can update my personal details and keep my profile up-to-date.*	3	Feature	Done	1. As an organizer, I should be able to update my name, profile picture, and info from my account settings page. 2. After saving changes, the updated information should be immediately reflected in my profile. 3. If any required field is missing or invalid, I should see an alert prompting me to correct it.
6	*As a user, I want to **search** for*	5	Feature	Done	1. As a user, I should be able to enter a search

	other players or teams so that I can be aware of the teams and players available on the platform.				query (player name, team name) in the search bar and see a list of matching results. 2. The search results should display relevant information such as player/team name and profile photo/team logo. 3. I should be able to click on a player or team to view their profile or team page.
7	*As a player, I want to **create** a team by giving it a name, logo, and description, and assigning players to specific roles so that the team is properly organized before sending invitations for approval.*	8	Feature	Done	1. As a player, I should be able to enter the team's name, upload a logo, and write a description. 2. I should be able to search for players by name in order to add them to the team. 3. After completing the team setup, I should be able to click the "Create" button to send invitation messages to each

					assigned player, showing: • The role assigned to them • The team's description • The team's current win percentage. 4. Players receiving the invitation should have the option to accept or reject the team invitation. 5. The team should not appear on my list or in any of the players' lists until all invited players accept the invitation. 6. If any player rejects, all the other players should receive a notification indicating that one of the players has declined and the team stats become declined. 7. Once all players accept, the team becomes visible to me

					and all players, and team stats become accepted.
8	*As a team member, I want to **edit** the team's page so that I can update the team's information and identity.*	3	Feature	Done	1. As a team member, I should be able to update the team's name, logo, and description from the team page. 2. After saving changes, the updated information should be immediately reflected on the team page visible to all team members.
9	*As a player, I want the model to analyze my selected players and generate the **top three optimal lineups** so I can choose the formation with the highest predicted win rate.*	10	Feature	Done	1. As a player, after adding the selected players to my team, I should be able to activate the AI model feature. 2. The AI model should analyze my team's players and generate the top three possible lineups, each showing the optimal role for

					every player based on win-rate potential. 3. I should be able to review the three AI model-recommended lineups and choose the one that I prefer.
10	*As a user, I want to see a preview of recently generated teams on the home page and be able to click "See All" to view the complete list of* ***all teams****, so I can quickly access and review team formations.*	3	Feature	Done	1. As a user, I want to see a preview of 3–4 most recently generated teams on the home page. 2. As a user, I want each team in the preview to show team name and logo. 3. As a user, I want to click the "See All" button to navigate to a page listing all generated teams. 4. As a user, I want to view all teams on the full list page with player names and role.
11	*As a player, I want to* ***send and receive***	8	Feature	Done	1. As a player, I should be able to open a chat section that lists all

	messages *with other players or teams that I am part of so that I can communicate and coordinate effectively.*				players and teams I am a member of. 2. I should be able to send text messages to: • Individual players • Entire team groups 3. When I send a message, the recipient(s) should receive it instantly and see it in their chat window. 4. Messages should be displayed with a timestamp and sender's name.
12	*As a player, I want to* ***add a game*** *to my profile so that I can see my statistics for that game displayed on my dashboard.*	3	Feature	Done	1. The player can select a game from a predefined list of supported games. 2. The player must enter their in-game username and tag when adding a game. 3. After saving, the new game is linked to the player's profile.

					4. The player's statistics are uploaded to the database. 5. An "Update Stats" button is displayed next to each game on the player's profile to refresh statistics on demand.
13	*As a player, I want to see the **timeline of my performance** so that I can track my progress over time and identify areas for improvement*	5	Feature	to do	1. As a player, I should be able to view a timeline chart showing my performance in past matches. 2. The timeline should display key stats such as: • Matches played • Wins / losses • Any other relevant performance metrics
14	*As a player, I want my **performance reports** to include personalized tips so that I can*	5	Feature	to do	1. The performance report should include personalized recommendations based on my match statistics, win/loss ratio, and performance trends.

	improve my skills.				2. Recommendations may cover areas such as: • Role-specific advice • Suggested practice focus • Strategies to improve the win rate.
15	*As a user, I want to see the **top-performing players** highlighted so that I can recognize high achievers and compare my performance against them.*	5	Feature	to do	1. The system should display a list or section of top players based on Badges. 2. The highlighted players should include key stats (username, rank, win percentage). 3. Users should be able to click on a highlighted player to view their profile or detailed stats.
16	*As a player, I want to earn **badges** and receive motivation feedback so that I feel rewarded for my achievements and stay*	6	Feature	to do	1. The system should award badges for specific achievements, including: • Consecutive play streaks (e.g., playing X days in a row) • Winning consecutive matches

	motivated to improve.				• Reaching 100 total wins 2. Players should be able to view all earned badges on their profile. 3. The system should provide motivational feedback or messages whenever a badge is earned.
17	*As a player, I want to receive* ***suggested tournaments*** *based on games I play so that I can easily discover tournaments that match my interests.*	6	Feature	to do	1. The system suggests tournaments only for games that the player has indicated they play. 2. Suggestions are displayed on the player's home in the Suggestions section.
18	*As a user, I want to* ***view the tournaments*** *published by organizers so that I can stay informed about*	3	Feature	to do	1. As a user, I should be able to see a list of tournaments that organizers have published. 2. Each tournament should display its

	upcoming competitions.				image, name and description clearly. 3. I should be able to click on a tournament to view its full details page. 4. If there are no tournaments available, I should see a message like: "No tournaments available at the moment."
19	*As an organizer, I want to **create tournaments** so that these tournaments become visible to players.*	7	Feature	to do	1. As an organizer, I should be able to upload a tournament image, enter the tournament name, description, details, game, date, and time. 2. After saving, the tournament should appear on the tournaments page and be visible to all players and teams. 3. The tournament should display its image, name, description, details,

					date, and time clearly formatted. 4. If any required field (e.g., name, date, time) is missing or invalid, I should receive an error message prompting me to fix it.
20	*As an organizer, I want to **edit** the tournaments I previously created so that I can update their information if needed.*	4	Feature	to do	1. As an organizer, I should be able to open any tournament I created and edit its information (image, name, description, details, date, and time). 2. After saving changes, the updated information should be immediately reflected on the tournaments page visible to players and teams. 3. Editing a tournament is locked 2 days before the scheduled start date the system should show a warning message if

					the organizer tries to edit after this period. 4. A clear confirmation message should appear after successful updates.
21	*As a user, I want the application to be available 99% of the time so that I can access my account, teams, and tournaments without interruptions.* [30].	-	Non-functional Feature	-	1. System uptime is monitored and logged 2. Scheduled maintenance notifications are shown to users in advance. 3. Scheduled maintenance notifications are shown to users in advance.
22	*As a player, I want the game or tournament pages to load within 3 seconds so that I can quickly access the information I need.*	-	Non-functional Feature	-	1. Page loads completely within 3 seconds under normal user load. 2. All images, text, and interactive elements are fully rendered on load.

					3. Page response is consistent across devices and browsers.
23	*As a player, I want the navigation and interface to be intuitive so that I can easily find tournaments, teams, and my profile without confusion.*	-	Non-functional Feature	-	1. Users can access all main features within 3 clicks or taps. 2. Menu labels and buttons are clear and consistent. 3. A usability test with at least 5 users shows 90% task completion rate without assistance.
24	*As a user, I want my personal data, login credentials, and team information to be securely stored so that my privacy and accounts are protected.*	-	Non-functional Feature	-	1. All sensitive data is stored encrypted in the database. 2. Passwords are hashed and never stored in plain text. 3. User sessions expire after inactivity. 4. Unauthorized access attempts are logged and blocked.
25	*As a player, I want the application to*	-	Non-functional Feature	-	1. All system errors are caught and handled

	handle unexpected errors gracefully without crashing so that I can continue using the system without disruption.				without crashing the app. 2. Error messages are user-friendly and provide guidance. Logging of errors is implemented for developer review and debugging

Table 5: Product Backlog

The user stories marked as "To Do" or "In Progress" are part of the planned development roadmap for the second term (Release 2). As outlined in the project plan, the work has been strategically divided across two terms to ensure systematic and manageable progress. remaining user stories are scheduled for completion in the second term according to the established roadmap and project timeline.

This phased approach aligns with the project milestones and academic requirements, allowing for iterative development, thorough testing, and quality assurance across both releases.

5 System Design

5.1 Architectural Diagram

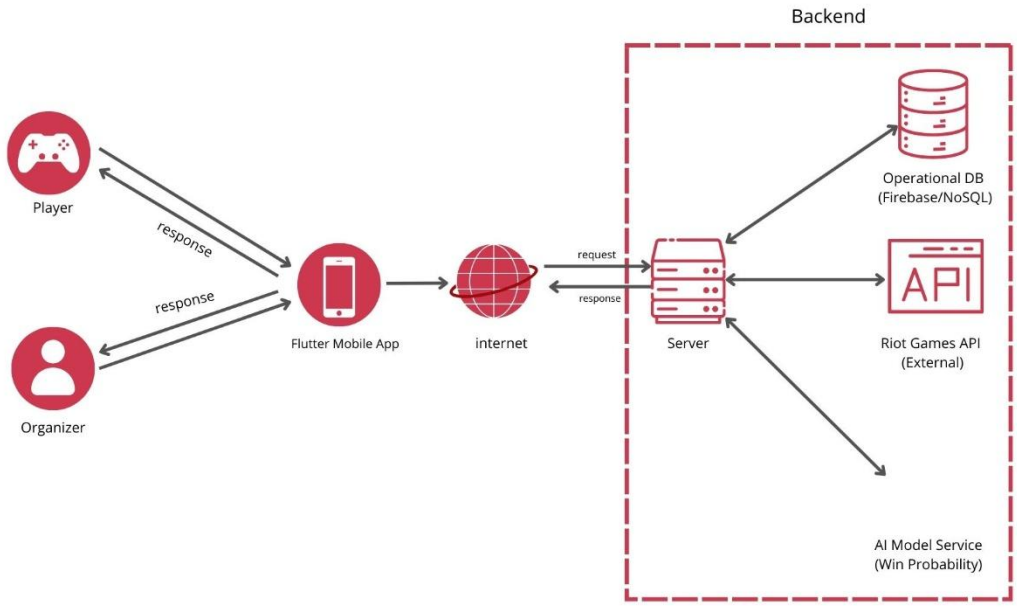


Figure 6: Architectural Diagram

The SPARK system follows a client–server architecture. The Flutter mobile app provides the user interface for players and organizers. The backend handles authentication, profiles, team management, tournaments, and reporting. It integrates with the external **Riot Games API** to retrieve match history and player statistics, stores operational data in a central database (Firebase/NoSQL), and invokes a separate AI Model Service to compute team win probability. A lightweight feature store is used to keep engineered features for inference. This decomposition separates concerns, improves scalability, and simplifies maintenance while supporting responsive user interactions.

5.2 Class Diagram /DFD

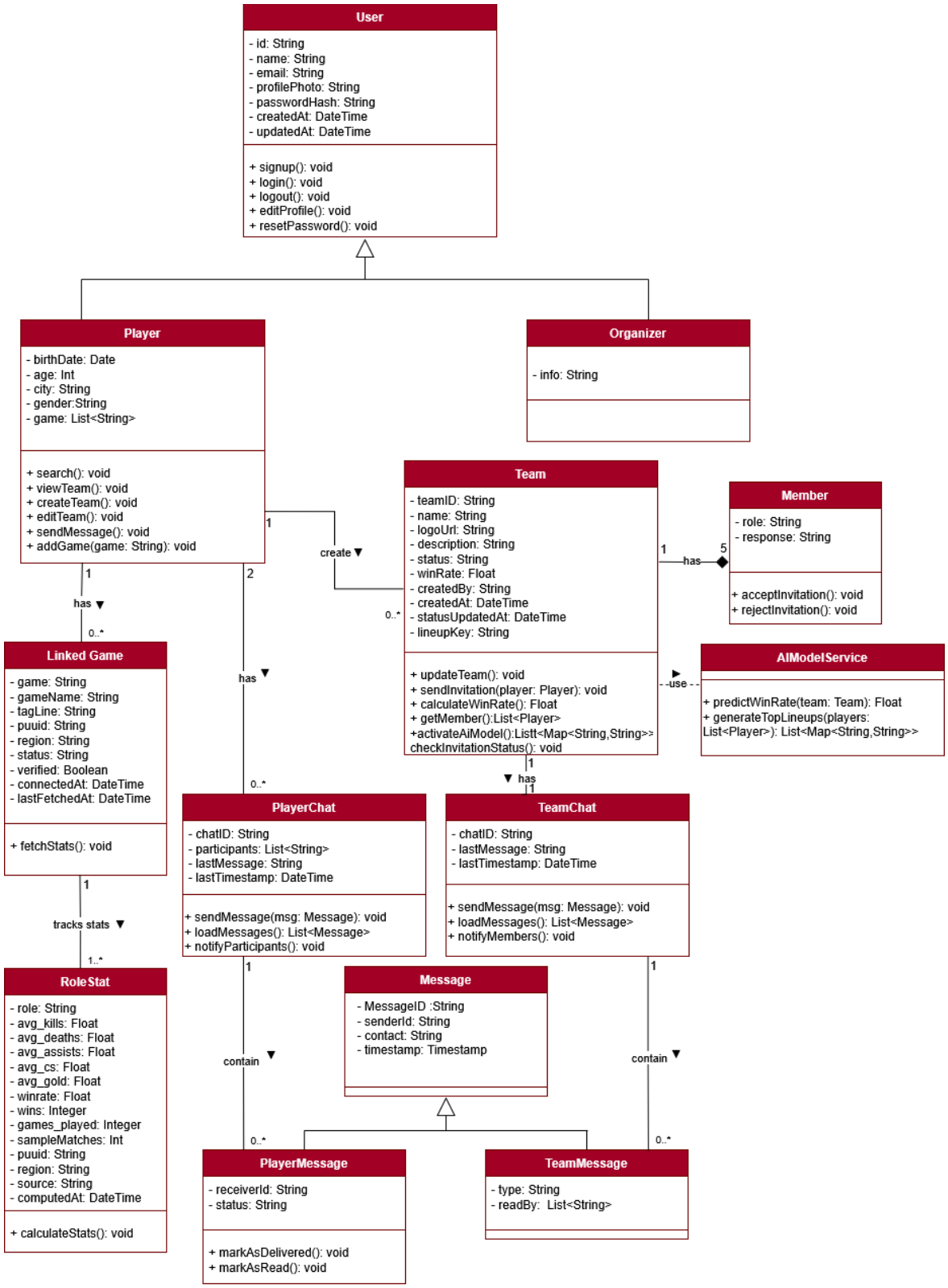


Figure 7:Class Diagram

5.3 Component Level Design

This section presents the key system components that support SPARK's core functionality. Each component describes the internal logic that transforms player data into meaningful features inside the app. Although these features appear simple to the user, they rely on structured processes, data flow, and model-based decisions in the backend. The following subsections explain how the system generates AI-recommended lineups, handles messaging between players and teams, and manages team creation and invitation approvals. Each component includes its purpose, core behavior, and supporting pseudocode or flowcharts to clarify how the system operates.

5.3.1 AI Top-3 Lineups Generation Component

ID	PBI	Size	Type	Status	Acceptance Criteria
9	*As a player, I want the model to analyze my selected players and generate the top three optimal lineups so I can choose the formation with the highest predicted win rate.*	10	Feature	Done	1. As a player, after adding the selected players to add to my team, I should be able to activate the AI feature. 2. The AI model should analyze my team's players and generate the top three possible lineups, each showing the optimal role for every player based on win-rate potential. 3. I should be able to review the three AI-recommended lineups and choose the one that I prefer.

Table 6:AI Top-3 Lineups Generation Component

Definition: This component is responsible for generating the top three lineups for the five selected players using the AI model. When the player activates the AI feature, SPARK retrieves

each player's stored role and performance statistics (originally pulled from the Riot API during game linking). The system then builds all valid role assignments for TOP, JUNGLE, MID, BOT, and SUPPORT based on the roles each player actually plays. For every possible lineup, SPARK constructs a feature vector following the model's required structure and sends it to the trained model to calculate the predicted win probability. These probabilities are produced by the Random Forest prediction model used in Spark. After evaluating all combinations, the system sorts them and returns the best three lineups, each showing the assigned roles and their estimated win rates. These three AI-recommended lineups are then displayed to the player, who can choose the one they prefer for team creation.

Pre-conditions:

- The player is logged in.
- All five selected players have linked their game accounts and have valid role stats stored (pulled earlier from Riot API).
- The AI model and feature list are available for prediction.
- The player has already selected which five players will be evaluated.

Post-conditions:

- The AI model evaluates all valid lineups and calculates their predicted win rates.

  A lineup is considered valid only if each assigned role matches a player who has played that role and has role-specific statistics stored in the database.
- The top three lineups are displayed to the player (not stored).
- The player selects one lineup.
- Only the chosen lineup is saved under the team creation process and used when sending invitations.

Flowchart:

Figure 8:flow chart 1

5.3.2 Team Creation & Invitation Approval Component

ID	PBI	Size	Type	Status	Acceptance Criteria
7	*As a player, I want to **create** a team by giving it a name, logo, and description, and assign players to specific roles so that the team is properly organized*	8	Feature	Done	1. As a player, I should be able to enter the team's name, upload a logo, and write a description. 2. I should be able to search for players by name in

	before sending invitations for approval.				order to add them to the team. 3. After completing the team setup, I should be able to click the "Create" button to send invitation messages to each assigned player, showing: • The role assigned to them • The team's description • The team's current win percentage or stats 4. Players receiving the invitation should have the option to accept or reject the team invitation. 5. The team should not appear on my list or in any of the players' lists until all invited players accept the invitation. 6. If any player rejects, all the other players should receive a notification indicating that one of the players has declined and the team stats become declined. 7. Once all players accept, the team becomes visible

					to me and all players, and team stats become accepted.

Table 7: Team Creation & Invitation Approval Component

Definition: This component lets a player create a team by entering a name, logo, and description, and assigning each player to a role. When the player clicks Create, the system sends an invitation to every assigned player with their role, the team details, and the team's win percentage. The team becomes official only if all invited players accept. If even one player rejects, the team is marked as declined and is not shown to anyone.

Pre-conditions:

- Player is logged in.
- Player has already chosen one AI-generated lineup.
- All invited players exist in the system.

Post-conditions:

- A team with status **pending** is created.
- Invitations are sent to all players.
- If all accept, the team becomes **accepted** and visible.
- If any reject, the team becomes **declined** and hidden.

Flowchart:

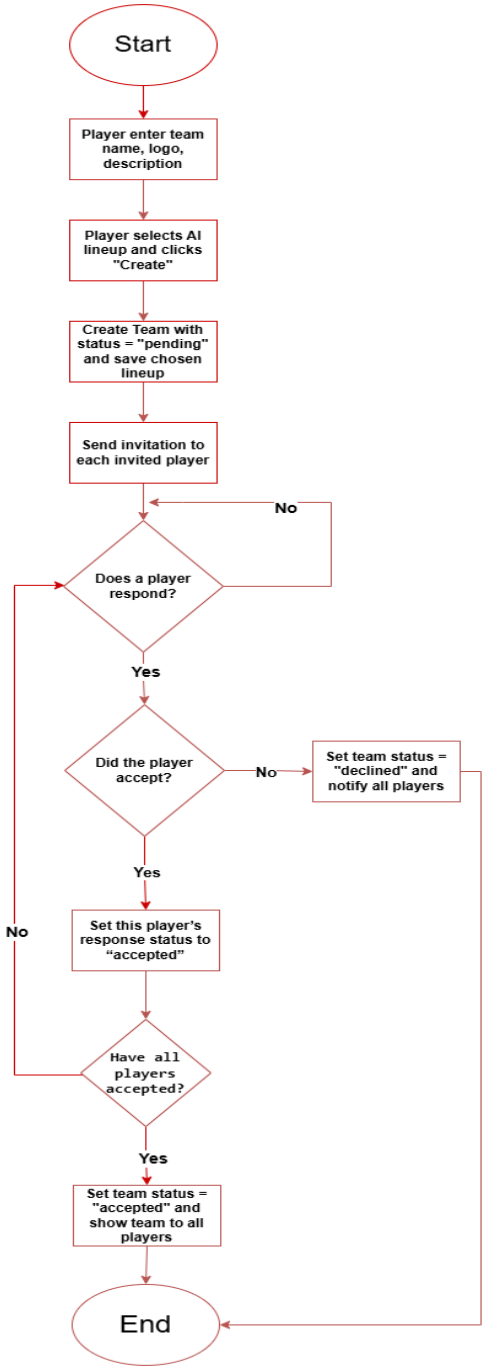


Figure 9:flow chart 2

5.3.3 In-App Messaging Component

ID	PBI	Size	Type	Status	Acceptance Criteria
11	*As a player, I want to* ***send and receive***	8	Feature	Done	1. As a player, I should be able to open a chat section that

	messages *with other players or teams that I am part of so that I can communicate and coordinate effectively.*				lists all players and teams I am a member of. 2. I should be able to send text messages to: • Individual players • Entire team groups 3. When I send a message, the recipient(s) should receive it instantly and see it in their chat window. 4. Messages should display with a timestamp and sender's name.

Table 8: In-App Messaging Component

Definition: This component provides a simple chat system inside SPARK. The system lets players chat with each other inside the app. A player can send messages privately to another player or talk with their whole team in a group chat. The app keeps every conversation in the database: one place for private chats and one place for team chats.

Pre-conditions:

- The player is logged in.
- The chat collections already exist in the database (PlayerChat for private chats, TeamChat for group chats).

Post-conditions:

- The player can see all their private chats and team chats.
- When they send a message, it is saved in the database and appears instantly to the other person or team.
- Messages show the text, the name of who sent it, and the time it was sent.

Pseudocode:

```
Start

1. Player opens a chat entry point.
  1.1 From the navigation bar:
    - Open the Messaging screen, which lists all chats.
  1.2 Or from a player profile:
    - Open or create a private chat with that player.
  1.3 Or from a team card or team header:
    - Open or create the group chat for that team.

2. When the Messaging screen is opened then
  2.1 Load private chats.
    - Read all PlayerChat documents where participants include currentPlayerId.
    - Order them by lastTimestamp in descending order.
    - For each chat, load the other player's name and profile photo.
    - Display avatar, name, lastMessage, and lastTimestamp.
  2.2 Load team chats.
    - Use TeamService.getUserTeams to get all teams of the current player.
    - For each team, read the related TeamChat document.
    - Display team name, logo, status, and an unread badge if newMessage is true.
  2.3 Wait for the player to select one chat.

3. When the player selects a chat then
  3.1 Determine the chat type.
    - If the selected item is a team, chatType ← "team".
    - Otherwise, chatType ← "player".
  3.2 Make sure the chat document exists.
    If chatType is "team" then
      - Look for the TeamChat document for this team.
      - If it does not exist then
        • Create a new TeamChat document for this team.
        • Set status to "pending" and newMessage to true.
        • Add an invitation message in TeamMessage with type "invitation".
    Else  (chatType is "player")
      - Try to find a PlayerChat document with both players as participants.
      - If it does not exist then
        • Create a new PlayerChat document with these two participants.
  3.3 Save the chat identifier as chatId.

4. Load existing messages and mark them as read.
  4.1 If chatType is "player" then
    - Read all PlayerMessage documents for this chat in time order.
    - For each message where the receiver is the current player
      and status is "sent", update the status to "read".
    - Start a real-time listener so any new message appears immediately.
```

4.2 Else (chatType is "team")
- Read all TeamMessage documents for this chat in time order.
- Start a real-time listener.
- For each displayed message, if the current player is not in readBy, add the current player to the readBy list.

4.3 Display all loaded messages in the chat screen (sender, text, time).

5. Handle special behaviour for team invitations (chatType = "team").
5.1 For each message with type "invitation":
- Show it as a special card at the top of the chat.
- Read the current player's response in the Team members collection.
- If the current player is the team creator, show "auto-accepted".
- Else if response is "Accepted", show "Invitation accepted".
- Else if response is "Rejected", show "Invitation rejected".
- Else (response is "none"), show Accept and Reject buttons.

5.2 When the player presses Accept or Reject then
- Update the member response to "Accepted" or "Rejected".
- Update acceptBy or rejectBy lists in the TeamChat document.
- Check all members' responses.
  • If any member rejected, set team status to "Rejected".
  • Else if all members accepted, set team status to "Accepted" and TeamChat status to "active".

5.3 If the team status is "pending" or "Rejected" then
- Hide the message input field and show a banner that the chat is locked.

6. While the chat screen is open, sending messages works as follows.
6.1 When the player types a message and presses Send then
- Set now to the current time.
- If chatType is "player" then
  • Create a new PlayerMessage document with sender, receiver, text, timestamp, and status "sent".
  • Update the PlayerChat document with lastMessage and lastTimestamp.
- Else if chatType is "team" and TeamChat status is "active" then
  • Create a new TeamMessage document with sender, text, timestamp, type "text", and readBy including the current player.
  • Update the TeamChat document with lastMessage, lastTime, and set newMessage to true.
- The new message appears immediately for all participants through the real-time listeners.

7. When a team chat is opened and the latest message is read by all members then
7.1 If all member IDs are included in the message readBy list then
- Set TeamChat.newMessage to false.

End

5.4 Data Design

5.4.1 Data Models

This section presents both the ER diagram and the NoSQL data model for the system. The ER diagram shows the main entities, players, organizers, teams, chats, and game stats, and their relationships. The non-relational data model translates this into a hierarchical structure in Firebase Firestore, using collections, documents, and subcollections. It also clarifies which relationships are embedded and which use referencing, providing a clear blueprint for implementing the database.

- ER diagram:

This part shows the main entities, players, organizers, teams, chats, and game stats, and how they interact. It highlights their roles in team creation, player communication, and game tracking, forming the basis for the Firebase Firestore design.

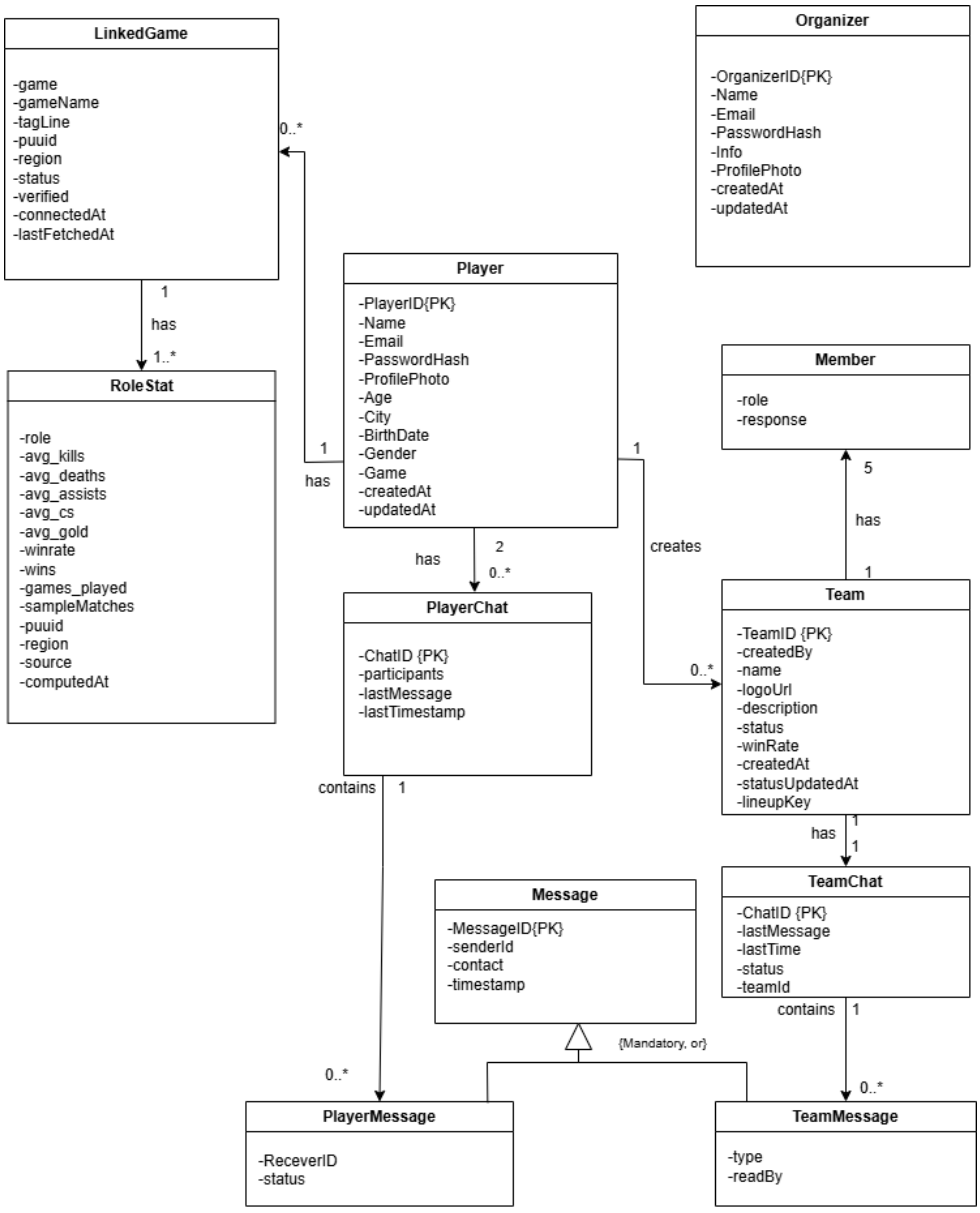


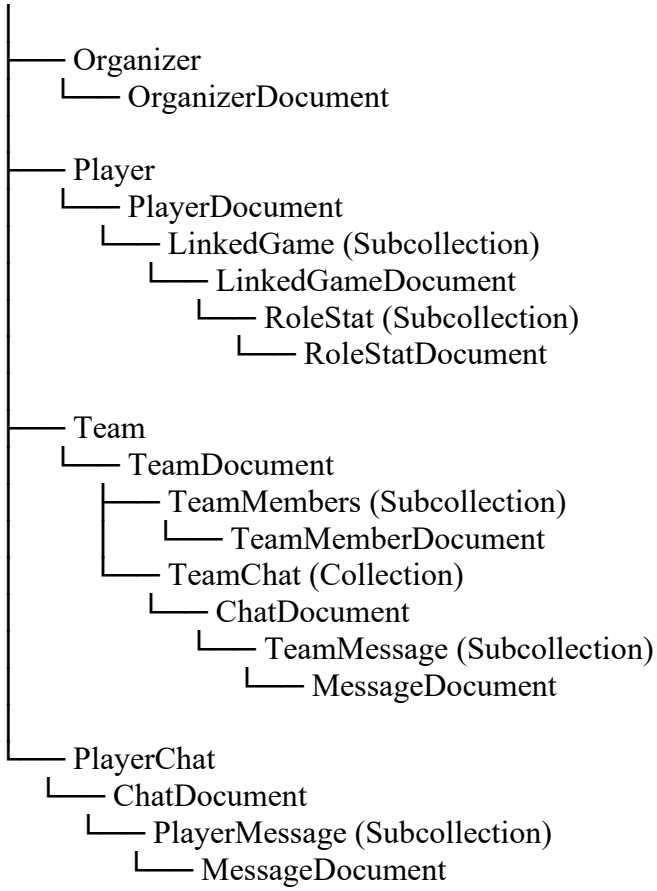
Figure 10::ER digram

5.4.2 Non-Relational Data Model (Firebase Firestore):

This section presents the hierarchical conceptual design of the system using a document-oriented NoSQL model (Firebase Firestore). The structure is represented as a tree of collections, documents, and subcollections, and it includes an explanation of whether each relationship uses embedding or referencing.

1. Overview of the Hierarchical Structure

Root



2. Detailed Non-Relational Data Model

Organizer Collection:

Each organizer is represented by a single document with fully embedded fields.

Organizer (Collection)
└── OrganizerID (Document)
- Name
- Email
- PasswordHash
- Info
- ProfilePhoto
- createdAt
- updatedAt

Relationship Approach: Embedding, All attributes are stored directly inside the document.

Player Collection:

1. Player Document.
Player (Collection)
└── PlayerID (Document)
- Name

- Email
- PasswordHash
- ProfilePhoto
- Age
- City
- BirthDate
- Gender
- Game (Array)
- createdAt
- updatedAt

2. Subcollection: LinkedGame.

LinkedGame
└── LinkedGameID (Document)
- game
- gameName
- tagLine
- puuid
- region
- status
- verified
- connectedAt
- lastFetchedAt

3. Subcollection: RoleStat (inside LinkedGame).

RoleStat
└── RoleStatID (Document)
- role
- avg_kills
- avg_deaths
- avg_assists
- avg_cs
- avg_gold
- winrate
- wins
- games_played
- sampleMatches
- puuid
- region
- source
- computedAt

Relationship Approach: Player → LinkedGame → RoleStat. Embedding (Subcollections) since the data belongs exclusively to the player. No reference is needed.

PlayerChat Collection :

1. PlayerChat Document.

PlayerChat (Collection)
└── ChatID (Document)
- participants (Array of PlayerIDs)
- lastMessage
- lastTimestamp

2.Subcollection: PlayerMessage.

PlayerMessage
└── MessageID (Document)
- senderId
- text
- timestamp
- ReceiverID
- status

Approach: Subcollections used for high-volume data (messages). senderId & ReceiverID are references.

Team Collection:

1.Team Document.

Team (Collection)
└── TeamID (Document)
- createdBy (PlayerID Reference)
- name
- logoUrl
- description
- status
- winRate
- createdAt
- statusUpdatedAt
-lineupKey

2.Subcollection: Member.

Member
└── PlayerID (Document)
- role
- response

Relationship Approach: Referencing , PlayerID used as reference inside Member. No embedding m, because players may belong to multiple teams.

TeamChat Collection:

1. TeamChat Document.

TeamChat
└── ChatID (Document)
- lastMessage
- lastTime
- status
-teamId

2. Subcollection: TeamMessage.

TeamMessage
└── MessageID (Document)
- senderId (PlayerID Reference)
- text
- timestamp
- type
- readBy (Array of PlayerIDs)

Relationship Approach: TeamChat stores chat information as a top-level document and references the related team using teamId → Referencing. TeamMessage documents are kept inside a subcollection under each chat → Embedding (child collection). Each message uses senderId and readBy to reference players → Referencing.

3. Embedding vs Referencing Summary.

Entity	Embedding	Referencing	Reason
Organizer	✓	✗	Stand-alone data
Player	✓	✗	Personal attributes
LinkedGame	✓	✗	Child data of Player
RoleStat	✓	✗	Belongs to game; large but structured
Team	✗	✓ createdBy	Connects Player ↔ Team
Member	✗	✓ PlayerID	Players exist independently
TeamChat	✓	✓ teamId	Chat data embedded, but team relationship via reference
TeamMessage	✓	✓ senderId	High-volume messages
PlayerChat	✓	✓ participants	References players
PlayerMessage	✓	✓ ReceiverID	Messaging system

Table 9: Embedding vs Referencing Summary

5.4.3 Data Collection and Preparation

In this section, we describe how the datasets used in our system were collected, created, organized, and preprocessed before model training. The project uses both Kaggle datasets and Riot API player data, and several data manipulation algorithms were developed to transform

raw match-level and player-level information into structured team-level features suitable for machine learning.

- Data Creation, Collection, and Organization

We utilized two external data sources:

1. **Kaggle dataset (multiple CSV files)** containing match information, player participation, and player performance statistics.
2. **Riot API dataset**, which provides real-time player-match statistics for live testing.

The Kaggle files (MatchTbl.csv, SummonerMatchTbl.csv, MatchStatsTbl.csv, TeamMatchTbl.csv) were merged to construct a unified player-match dataset. Riot API data (lol_full_clean.csv) was collected by querying player accounts, retrieving recent match IDs, and extracting per-match performance records. All data files were loaded, cleaned, and organized into structured DataFrames for further processing.

- Data Preprocessing Procedures

We performed a multi-stage preprocessing pipeline to transform raw inputs into a model-ready dataset:

1. Collecting Raw Player-Match Data.

We merged multiple Kaggle tables to create a single dataset representing each player in each match.

Processing steps included:

- Merging SummonerMatchTbl with MatchStatsTbl to link players with their in-match stats.
- Merging with MatchTbl to append match metadata.
- Renaming columns for consistency.
- Filtering to include only CLASSIC queue type.

Output **raw_df**:

	player_id	match_id	champion_id	role	win	kills	deaths	assists	MinionsKilled	DmgDealt
0	1	EUW1_7565751492	902	SUPPORT	0	0	2	12	30	4765

DmgTaken	TurretDmgDealt	total_gold	queue_type	rank_id	game_duration
12541	0	7058	CLASSIC	7	1751

Figure 11:raw_df

2. Building Player Role-Based Statistics.

We created **player_role_stats** by summarizing each player's historical performance by role.

Steps included:

- Filtering out rows without valid roles.
- Grouping by (player_id, role).
- Aggregating metrics such as wins, kills, assists, MinionsKilled, and total_gold.
- Computing winrate = wins / games_played.

This table represents each player's average performance when playing specific roles.

	player_id	role	games_played	wins	avg_kills	avg_deaths	avg_assists	avg_cs	avg_gold	winrate
0	1	SUPPORT	10	5	1.300000	2.90	17.100000	33.0000	8923.700000	0.500000

Figure 12:player_role_stats

3. Constructing the Player-Team Training Table.

We merged player-level data (raw_df) with team compositions (TeamMatchTbl.csv).

Steps included:

- Determining the player's team side (Blue/Red) from champion assignments.
- Computing team_win based on BlueWin.
- Merging historical role statistics (player_role_stats) into each player row.

The output:

	player_id	match_id	role	champion_id	kills	deaths	assists	total_gold	MinionsKilled	team_side
0	1	EUW1_7565751492	SUPPORT	902	0	2	12	7058	30	Blue

team_win	queue_type	rank_id	hist_games_played	hist_wins	hist_avg_kills	hist_avg_deaths	hist_avg_assists
0	CLASSIC	7	10	5	1.3	2.9	17.1

hist_avg_cs	hist_avg_gold	hist_winrate
33.0	8923.7	0.5

Figure 13:player_role_stats

4. Creating Team-Level Aggregated Dataset.

To produce machine-learning features at the team level, we aggregated player-level data:

- Grouping players by (match_id, team_side).
- Averaging team combat stats, economy stats, and historical stats.

- Computing multiple synergy metrics such as kills distribution, gold share variance, role entropy, and resource imbalance.

This generated **kaggle_team_full_df**, where each row represents one team in one match with full performance and synergy indicators.

	match_id	team_side	kills_mean	deaths_mean	assists_mean	total_gold_mean	MinionsKilled_mean	hist_winrate_mean
0	EUW1_6681382047	Blue	39.0	1.0	1.0	16025.0	144.0	1.0000

hist_games_played_mean	hist_games_played_sum	hist_avg_kills_mean	hist_avg_deaths_mean	hist_avg_assists_mean
3.0	3	24.0000	0.666667	3.333333

hist_avg_cs_mean	hist_avg_gold_mean	team_win	is_blue_team	syn_n_players	syn_kills_mean	syn_kills_std	syn_deaths_mean
128.666667	13402.333333	1	1	1	39.0	0.0	1.0

syn_deaths_std	syn_assists_mean	syn_assists_std	syn_minions_mean	syn_minions_std	syn_kda_mean	syn_kda_std
0.0	1.0	0.0	144.0	0.0	40.0	0.0

syn_role_nunique	syn_role_entropy	syn_role_max_count	syn_role_min_count	syn_role_imbalance	syn_gold_share_std
1	-0.0	1	0	1	0.0

syn_gold_share_max	syn_kill_share_std	syn_kill_share_max	syn_assist_share_std	syn_assist_share_max	syn_minion_share_std
1.0	0.0	1.0	0.0	1.0	0.0

Figure 14:kaggle_team_full_df

5. Processing Riot API Dataset.

We replicated the preprocessing pipeline for Riot data to ensure full feature alignment with Kaggle data.

The final output, **riot_team_full_df**, includes team-level averages and synergy metrics for Riot matches.

	match_id	team_side	kills_mean	deaths_mean	assists_mean	MinionsKilled_mean	csPerMin_mean	goldPerMin_mean
0	EUW1_7548714571	Blue	5.0	3.8	9.6	179.8	6.185780	419.917431

dmgPerMin_mean	kda_mean	team_gold_first	team_kills_first	team_deaths_first	team_assists_first	team_damage_first
913.128440	8.353333	61028	25	19	48	132708

team_minions_first	team_baron_kills_first	team_dragon_kills_first	team_tower_kills_first	team_inhibitor_kills_first
899	1	2	6	0

team_riftHerald_kills_first	team_objectives_win_first	team_win	is_blue_team	syn_n_players	syn_kills_mean	syn_kills_std
1	True	1	1	5	5.0	2.000000

syn_deaths_mean	syn_deaths_std	syn_assists_mean	syn_assists_std	syn_minions_mean	syn_minions_std	syn_kda_mean
3.8	2.315167	9.6	3.611094	179.8	84.712219	8.353333

syn_kda_std	syn_role_nunique	syn_role_entropy	syn_role_max_count	syn_role_min_count	syn_role_imbalance	syn_gold_share_std
7.650813	4	1.332179	2	0	2	0.035245

syn_gold_share_max	syn_dmg_share_std	syn_dmg_share_max	syn_kill_share_std	syn_kill_share_max	syn_assist_share_std
0.231369	0.059062	0.290940	0.080000	0.320000	0.075231

syn_assist_share_max	syn_minion_share_std	syn_minion_share_max
0.291667	0.094229	0.283648

Figure 15:riot_team_full_df

6. Combining Data and Training the Model.

We unified Kaggle and Riot datasets into **full_train_df** using shared feature columns. Preprocessing included:

- Label alignment (team_win as the target).
- Train-test split (80/20 stratified).
- Training a **RandomForestClassifier (500 estimators)**.
- Generating evaluation metrics (accuracy, classification report, confusion matrix).

This step produced the final trained model: **rf_model_v5**.

7. Testing the Model Using Real Riot Players.

To evaluate model usefulness, we fetched real player stats using Riot API and:

- Constructed player-level summaries.
- Aggregated features into a team row.
- Computed synergy metrics.
- Ensured feature alignment with training columns.

- Used the model to predict win/loss and win probability.

Software and Algorithms Used:

- **Python (Pandas, NumPy)** for data merging, cleaning, grouping, and feature aggregation.
- **Scikit-learn** for model training and evaluation (RandomForestClassifier, train_test_split).
- **Riot API** for live player and match data retrieval.
- **Custom preprocessing** to merge datasets, compute historical stats, assign team sides, build synergy metrics, and construct feature tables.

5.5 Interface Design

This section presents the overall structure of the SPARK application using clear site maps for both user types: **Player** and **Organizer**. The diagrams illustrate the main pages of the system and how users navigate between them.

5.5.1 Player Sitemap

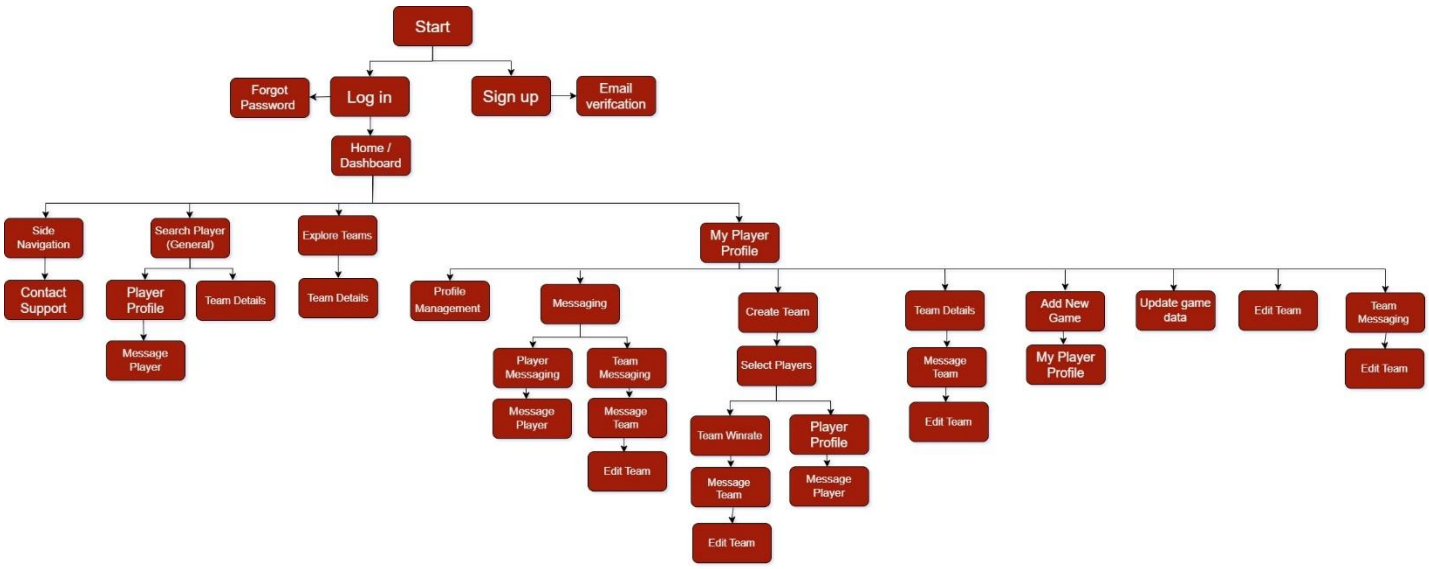


Figure 16:Player Sitemap

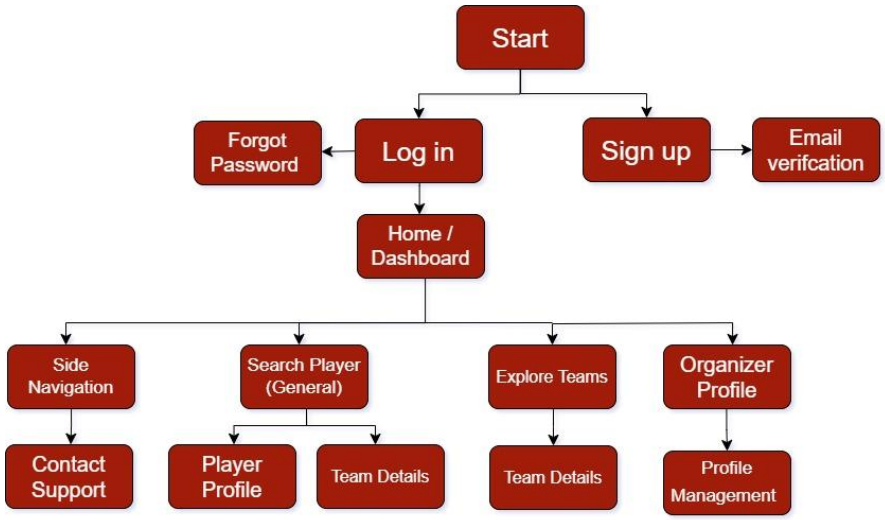


Figure 17:Organizer Sitemap

5.5.2 UX Guidelines

As a player, I want to edit the assigned roles of my team members so that the team is more organized and balanced.

1- Simplicity & Minimalism

The login screen follows a simple and minimalistic design, using clean layouts, limited colors, and clear text fields to help users complete the task quickly without distraction.

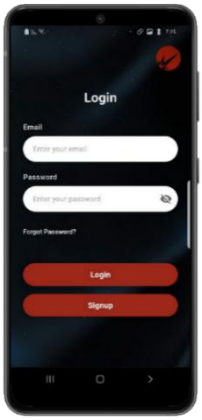


Figure 18: UX Guideline: simplicity

2- Error Prevention

The sign-up form uses clear validation messages (such as “Field is required”) to prevent users from submitting incorrect or incomplete information. The system highlights missing inputs in red, helping the user fix errors before proceeding.

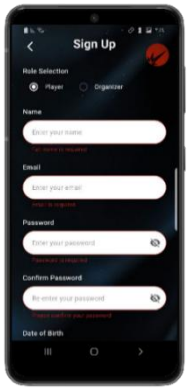


Figure 19:UX Guideline: Error Prevention

3- Clear Navigation

The side navigation menu provides a clear and consistent structure that allows users to easily access key sections such as Teams, Tournaments, Messages, Profile, and Contact. This helps users move smoothly through the app without getting lost or confused.

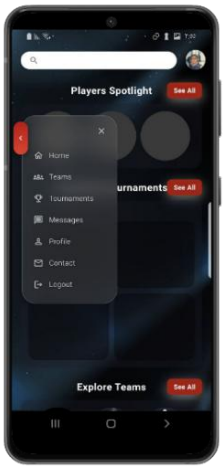


Figure 20:UX Guideline: Clear Navigation

4- Feedback & Status Visibility

The application provides clear system feedback by displaying the team's win rate and showing the top three recommended lineups immediately after the user selects teammates. This helps players understand system results in real time and compare different

Figure 22:Feedback & Status Visibility (1)

Figure 21:Feedback & Status Visibility (2)

team compositions effectively.

5- Consistency

The application maintains visual consistency across different screens by using the same color palette, typography, spacing patterns, and button styles. This unified design helps users navigate comfortably and recognize elements quickly.

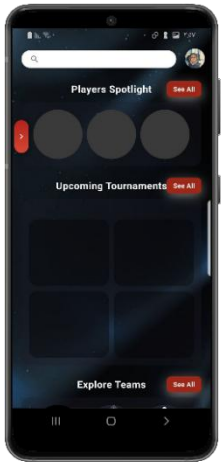


Figure 24:Consistent – Home Page

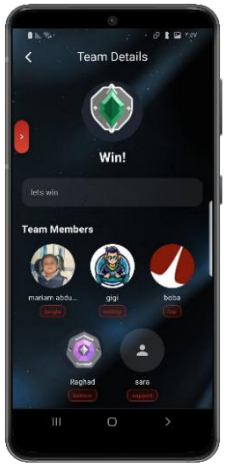


Figure 23:Consistent Layout – Team Details Page

6- User Control

The system gives users full control by allowing them to edit, save, or cancel actions, and by clearly indicating when updates are completed or require additional input.

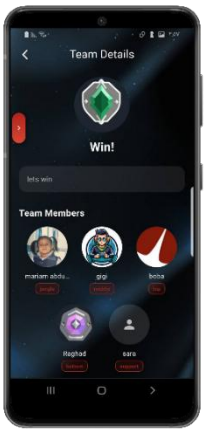


Figure 26:User Control (Team Management

Figure 25:User Control (Profile Editing)

7- Feedback

The application provides clear and continuous feedback during the team formation process.Users can see real-time invitation statuses, such as pending, accepted, or completed, which helps them understand exactly where the team stands. This transparency ensures players always know who has responded, who is still pending, and when the team becomes active and ready to use.

Figure 27:Feedback (Invitation Status – Pending Stage)

Figure 28:Feedback (Invitation Status – Accepted Stage)

6 System Implementation

6.1 Overview

The implementation of SPARK followed an iterative and modular development approach to ensure scalability, maintainability, and smooth integration between the machine learning model, Firebase backend, and Flutter frontend.

This section details the tools, technologies, datasets, model components, and implementation steps carried out by the team

6.2 Hardware & Software Components

- **Hardware Tools**

1. **Laptops:**
   Used for software development, model training, and backend integration.
2. **Android Devices:**
   Used to test the UI responsiveness, Firebase integration, and API-dependent features.

- **Software Tools:**

Tool	Category	Description
Firebase	Backend (BaaS)	Used for authentication, real-time database, storage, and managing user/team/tournament data.
Python	Programming Language	Used to build and run the AI model for win-rate prediction and role optimization.
Jupyter Notebook	Training Environment	Used for model experimentation, feature engineering, and evaluation.

Riot Games API	External API	Provides live player stats, match history, and performance metrics.
VS Code	IDE	Used to develop the Flutter application and manage Dart code structure.
Flutter (Dart)	Frontend Framework	Used to build the SPARK mobile interface for players and organizers.

Table 10 Hardware & Software Components

- **System Configurations**

  - Flutter SDK 3.x
  - Firebase CLI
  - Python 3.10
  - Firestore structured into: **Players, Teams, Invitations**
  - Riot API configured using authentication tokens and rate-limit handling

6.3 Key Implementation Steps

This section presents the major implementation steps followed during SPARK development. The process included gathering requirements, designing the system architecture, integrating Firebase services, handling Riot API data collection, preparing datasets, building the machine learning model, and linking the model output back to the Flutter application. Each step contributed to establishing a scalable, data-driven platform capable of generating optimized team lineups and accurate win-rate predictions.

Step 1: Requirements & System Architecture

- Identified **Player** and **Organizer** roles.
- Defined platform features using the Product Backlog.
- Designed Firebase collections:

- Player
- Team
- TeamChat
- PlayerChat
- Organizer

- Chat collections include subcollections (PlayerMessage, TeamMessage) to store messages efficiently.

Step 2: Firebase Integration

1. Authentication

Player & organizer account creation

Secure login using FirebaseAuth

Email-based password recovery

2. Realtime Database

Stores:

Player data

Team information

Player-to-Player chat messages

Team chat messages

2. Storage

Stores:

Player avatars

Team logos

Why Firebase?

Real-time synchronization

Secure authentication

Easy integration with Flutter

Scalable with user growth

Step 3: Riot API Integration

We fetched:

- Player PUUID
- Match history
- Role frequency
- Performance per champion
- Major Challenge API Rate Limits

Riot API allows only 24 requests per second, causing frequent 429 errors during data collection.

- Solution
- Added retry-after logic
- Applied sleep delays
- Implemented session caching
- Rotated API keys upon reaching limits

Step 4: Dataset Preparation

We combined:

- Kaggle dataset (~80,000 matches)
- Riot API data (live match history)

Processing Steps

1. Remove missing/corrupted entries
2. Normalize champion IDs
3. Create player-role performance profiles
4. Compute synergy scores
5. Encode categorical features
6. Split data into training/testing sets

Before / After Example:

- **Before (Raw Entry):**

```
player_id: 1283
role: JUNGLE
champion: khazix_001
win: "true"
cs: "150"
```

Figure 29:Data Transformation Example (Before)

This row shows the original match data as retrieved from the Riot API before any cleaning, normalization, or feature engineering.

- **After (Processed Feature Row):**

```
player_avg_cs: 152.3
player_role_winrate_jungle: 0.61
synergy_score: 0.78
team_comp_vector: [...]
```

Figure 30:Data Transformation Example (After)

This row represents the cleaned and transformed features after preprocessing, ready to be used for model training:

Step 5: Machine Learning Model Implementation

The SPARK ML Model performs:

1) **Best Combination Generation (Team Optimization)**

Generates the best possible team lineup by evaluating synergy and performance, not "best role."

2) **Team Win Rate Prediction**

Predicts win probability for the full 5-player squad.

3) **Top-3 Lineups**

Produces the strongest three-team formations.

Step 6: Core Algorithms

Here's a clean code snippet:

Example 1 : Player Role Profile Creation

```
# Create player role profile: calculate winrate per role
def build_role_profile(df):
    role_profile = (
        df.groupby(['player_id', 'role'])['win']
        .mean()
        .reset_index()
        .rename(columns={'win': 'role_winrate'})
    )
    return role_profile
```

Figure 31:Player Role Profile Function

This function builds each player's historical role-performance profile to support synergy and lineup scoring.

Example 2 : Predicting Best Role Using Logistic Regression

```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest model (Model v5)
rf_model_v5 = RandomForestClassifier(
    n_estimators=200,
    max_depth=20,
    random_state=42,
    class_weight="balanced"
)

rf_model_v5.fit(X_train, y_train)

# Predict win/lose for a team
pred = rf_model_v5.predict(team_features)[0]
```

Figure 32:Random Forest Model Training

This Random Forest model was trained on the combined Kaggle + Riot API dataset. Random Forest was chosen because it handles non-linear interactions, provides high stability, and delivered the best performance among all tested models.

Example 3 : Generating Top 3 Lineups

```
def get_best_lineups(all_combinations, model):
    scored = []
    for team in all_combinations:
        wr = model.predict([team])[0]
        scored.append((team, wr))
    scored.sort(key=lambda x: x[1], reverse=True)
    return scored[:3]
```

Figure 33:Role Prediction Function

This algorithm uses synergy scores and model predictions to generate and rank the strongest three possible team lineups.

Step 7: Integration With Platform (Flutter ↔ Firebase ↔ Python)

1. Process Summary
2. Flutter sends selected players and team IDs to Firebase.
3. Python reads data (players, stats, match history).
4. Model generates:
    - Best possible lineup
    - Predicted win rate
    - Recommended team combination
5. Python writes results back to Firebase as structured JSON.
6. Flutter fetches updated predictions and displays optimized results.

6.4 Out-of-the-box Components

We used the following external libraries and services to speed up development:

- **Pandas, NumPy** → data preprocessing
- **Scikit-Learn** → ML model training
- **Requests** → Riot API communication
- **Firebase SDK (Flutter)** → auth & database operations

- **Firebase Admin (Python)** → reading/writing model results
- **Matplotlib** → model evaluation

6.5 Implementation Challenges

1. **Riot API Rate Limits**
   Limited to 24 requests per second, causing interruptions and requiring retries.
2. **Merging Kaggle + API Data**
   Different formats required cleaning, normalization, and column mapping.
3. **Team Combination Explosion**
   Thousands of possible lineups; solved by pruning and optimized evaluation loops.
4. **Firebase Sync Conflicts**
   Simultaneous updates from multiple users caused conflicts in team and profile data, requiring additional validation during write operations.
5. **Model Integration**
   Required converting Python model output → JSON → storing it in Firebase for the Flutter app.
6. **Transforming Player-Level Data into Team-Level Data**
   The Kaggle and Riot API datasets were player-level, while the model required team-level rows (one row per 5-player squad).
   We had to group players by match ID, merge their roles, compute synergy, and restructure the data, which required extra cleaning and processing.

Consultations

During the implementation of SPARK, we consulted our project supervisor to validate the architecture and machine learning design. We also relied on official documentation for Flutter, Firebase, and Riot Games API to ensure correct integration and best practices.

6.6 GitHub Link

2025_GP_34: SPARK

7 System Testing

This section ensures that the SPARK application functions correctly, meets the documented requirements, and delivers a smooth experience for end users. Multiple testing methods were applied to evaluate usability, performance, and correctness of implemented features. This section summarizes the testing process, including User Acceptance Testing (UAT), the testing design, and the results gathered from real users.

7.1 User Acceptance Testing

User Acceptance Testing (UAT) is an essential phase for evaluating whether the SPARK application meets the expectations of its intended users. The UAT focused on assessing the platform's usability, ease of navigation, clarity of information, and overall user satisfaction. For this evaluation, data was collected exclusively through a structured online questionnaire, which provided both quantitative ratings and qualitative feedback from users. A total of **9 participants** completed the questionnaire, representing key segments of SPARK's target audience, including casual players, competitive players, and an esports organizer. The collected responses offer valuable insights into the system's strengths and areas for improvement. This section presents the UAT design, participant demographics, questionnaire results, and user feedback.

7.1.1 Demographics of Participants

To ensure clarity and consistency, the demographics of the nine users who completed the UAT survey are presented in this section. These results confirm that the participant sample reflects SPARK's intended audience, including esports players, competitive players, and organizers. The demographic information is shown in the following.

• Age Distribution:

Most participants were between 18–24 years old (77.8%), while the remaining respondents were evenly distributed between the 25–30 and above 30 age groups (11.1% each) (Figure 34).

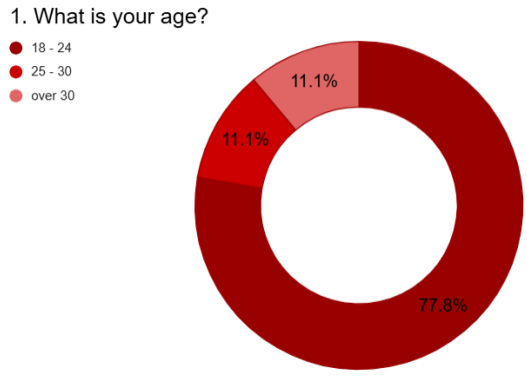


Figure 34:Questionnaire Results – age Distribution

• Gender:

The sample consisted of both male and female participants, with females forming a slight majority (55.6% female, 44.4% male) (Figure 35).

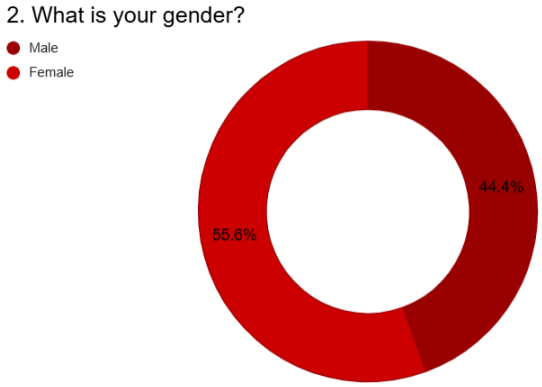


Figure 35:Questionnaire Results – gender Distribution

• Esports Community Role:

The participants represented a balanced mix of esports roles, including casual players (44.4%), organizers (44.4%), and one competitive player (11.1%) (Figure 36).

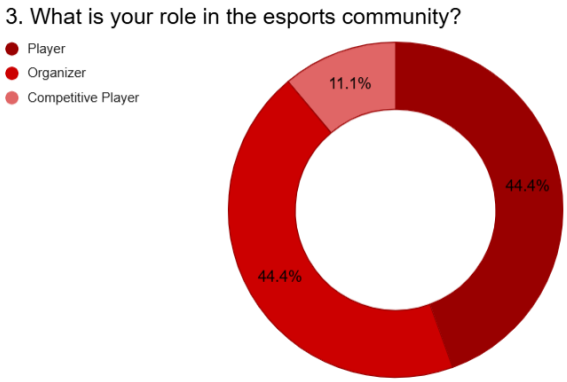


Figure 36:Questionnaire Results – role in the esports community

• **Gaming Experience:**

Most participants reported having between 1–4 years of gaming experience (a combined 88.8%), while one participant had over 5 years of experience (11.1%) (Figure 37).

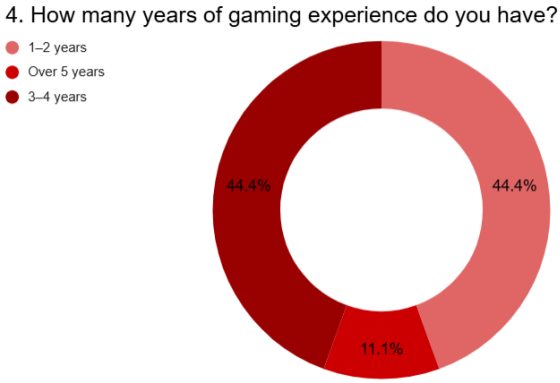


Figure 37:Questionnaire Results – gaming experience

• **Main Game Played:**

Participants reported playing a variety of main games. PUBG was the most frequently played game (66.7%), followed by League of Legends (22.2%) and Valorant (11.1%) (Figure 38).

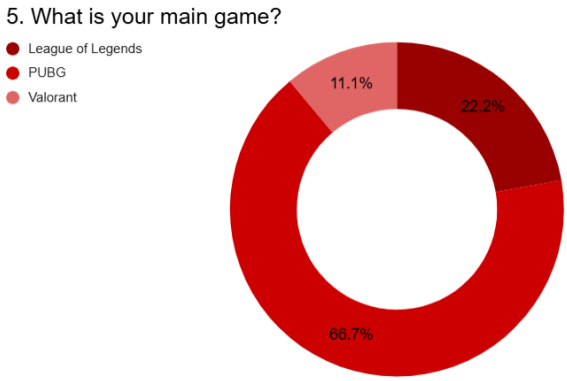


Figure 38:Questionnaire Results – main game

7.1.2 Questionnaire/Interview Results

A user questionnaire was conducted to gather feedback on the SPARK platform and evaluate its usability from the perspective of both players and organizers. The survey aimed to assess user demographics, gaming habits, and the effectiveness of key platform features. A total of 9 participants completed the questionnaire, providing valuable insights into the platform's strengths and areas for improvement. The usability evaluation of SPARK yielded positive results across multiple dimensions. When assessing whether the user interface was clear and easy to navigate, 44.4% strongly agreed and 33.4% agreed, with 22.2% remaining neutral (Figure 39). Similarly, the platform layout and feature organization received strong approval, with 44.4% rating it as excellent (5/5) and 33.3% rating it as good (4/5) (Figure 41). The team formation section also performed well, with most participants rating it as easy to understand and use (Figure 40). The AI win-probability feature was particularly well-received, with 88.8% of users rating it as easy or very easy to understand (44.4% rating it 5/5 and 44.4% rating it 4/5) (Figure 42). Furthermore, when asked about frequency of use, 88.8% indicated they would use SPARK frequently if it were officially released (44.4% rating 5/5 and 44.4% rating 4/5) (Figure 43), demonstrating strong user acceptance and platform viability.

These findings suggest that SPARK successfully addresses user needs in terms of interface design, feature accessibility, and overall user experience. The high satisfaction rates indicate that the platform is well-positioned for adoption within the esports community.

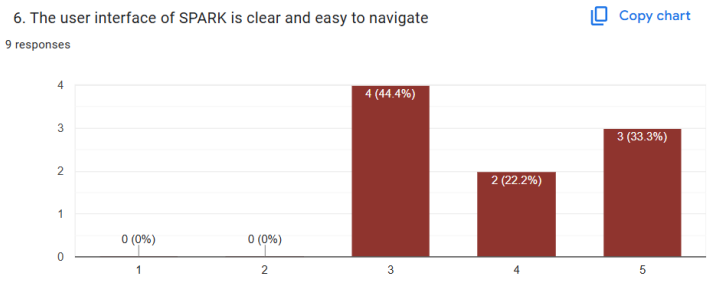


Figure 39:Questionnaire Results – user interface of SPARK is clear and easy to navigate

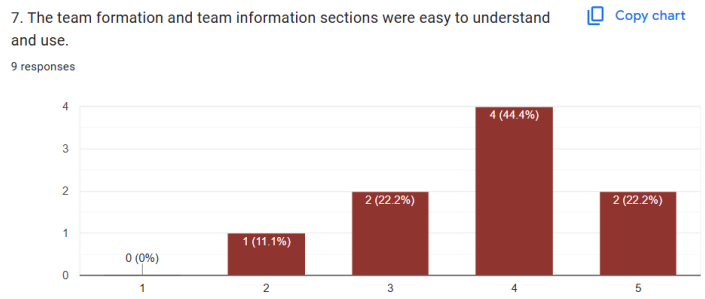


Figure 40:Questionnaire Results – team formation easy to understand and use.

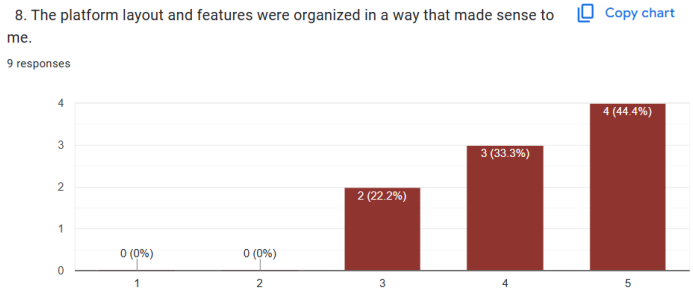


Figure 41:Questionnaire Results – platform layout and features organized.

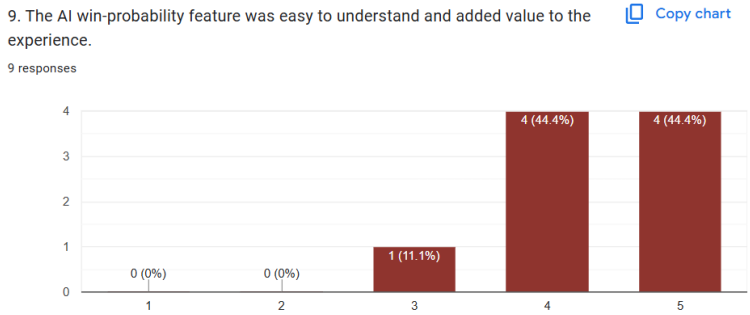


Figure 42:Questionnaire Results – AI win-probability feature was easy to understand

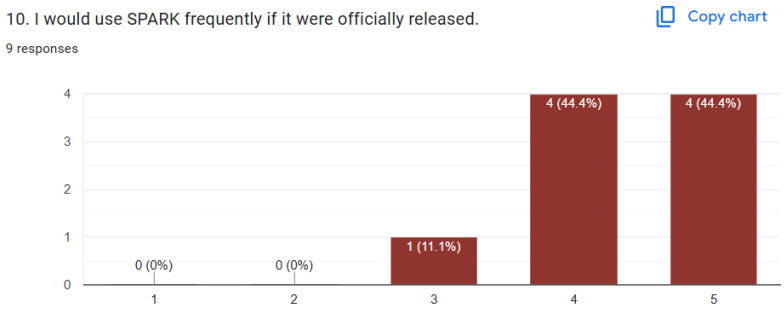


Figure 43:Questionnaire Results – use SPARK frequently.

7.2 Discussion

The results of the system evaluation suggest that SPARK is generally well-received by its users. Most participants, who were primarily young adults aged 18–24 with a slight majority of females, found the interface clear and the layout well-organized. This indicates that the design effectively supports usability, allowing users to navigate the system without confusion. The balance between different roles within the esports community, such as players and organizers, suggests that the system appeals to a broad range of users. Moreover, the fact that PUBG was the most played game among respondents highlights the relevance of the platform for popular esports titles. The AI win-probability feature was particularly appreciated, showing that users see added value in data-driven insights, which can enhance their gaming experience and decision-making. Additionally, the majority of participants expressed that they would use SPARK frequently if officially released, suggesting strong engagement potential. Taken together, these findings indicate that the system evaluation produced positive results, confirming that the current design and functionality meet user expectations to a large extent. Despite these positive results, there is room for improvement. The sample size was limited to 9 participants, which may affect the generalizability of the findings. Future evaluations should involve a larger and more diverse group of users to gain a more comprehensive understanding of user needs. From a design perspective, some interface elements could be refined to enhance accessibility and clarity, especially for users less familiar with esports platforms. Regarding implementation, the AI feature could be further developed to improve accuracy and provide more personalized recommendations. Collecting ongoing feedback after release could also help iteratively improve the system, ensuring that it continues to meet user expectations and maintains engagement.

8 Conclusions and Future Work

SPARK is an AI-powered esports platform designed to enhance visibility for Saudi players and bridge the gap between talent and opportunity in the local and global esports industry. Throughout this document, we presented the problem of limited recognition and structured support for players, reviewed existing platforms, and identified gaps that inspired SPARK's development. We then described the system requirements, architecture, and design process, followed by the implementation of data analysis, AI models, and the Flutter-based mobile application. In conclusion, SPARK demonstrates how artificial intelligence, gamification, and mobile technologies can be integrated to support Saudi Arabia's Vision 2030 goals and advance the esports ecosystem.

8.1 Global and local impact.

SPARK has a significant impact both locally and globally. Locally, it aligns with Saudi Arabia's Vision 2030 by promoting digital entertainment and empowering youth through structured opportunities in esports. Furthermore, it supports the development of the esports ecosystem by providing structured tools for talent evaluation, tournament organization, and communication between players. Globally, SPARK illustrates how intelligent systems and AI models can transform esports analysis and team formation processes. It drives a transformative change in how esports performance is analyzed, managed, and connected across the global gaming ecosystem. It provides a scalable framework that can be adapted to other regions, offering data-driven insights and promoting fair competition within the international gaming community.

8.2 Problems and challenges encountered during the software development

During the development of SPARK, the team encountered several challenges. One major issue was the Riot API rate limitation, which restricted the number of requests per second and required implementing caching, retry delays, and alternative data collection strategies. Another challenge was merging and cleaning data from multiple sources such as the Riot API and Kaggle datasets, which required extensive preprocessing and normalization. Additionally,

Firebase synchronization conflicts occurred when multiple users updated data in real time, requiring careful handling of concurrency and database updates. The team also faced difficulties in integrating the Python-based AI model with the Flutter application and Firebase, which demanded interdisciplinary coordination. Finally, time management and learning new technologies, including Flutter, Firebase, and AI integration, were key challenges the team overcame through collaboration and agile planning.

8.3 Limitations of the system.

Despite achieving its primary goals, the current version of SPARK still presents several limitations. The AI model is currently limited to predicting win probabilities for League of Legends only, making cross-game generalization unavailable at this stage. The platform is also implemented exclusively as an Android mobile application, with no web or iOS support yet. In addition, advanced functionalities such as player identity verification have not been implemented in this release. Although verifying whether the username or tag added by the user actually belongs to them is essential, Riot Games does not provide any permissions that allow for authenticating player accounts or confirming ownership. The model's accuracy also relies heavily on the availability and reliability of player data from external APIs, which may fluctuate due to updates or rate limits. Additionally, there is a delay in predicting the baseline when a player creates a team, as the system retrieves data for up to 50 matches, which can temporarily slow down the process. These limitations will be addressed in future iterations to enhance system reliability, scalability, and inclusivity.

8.4 The main contribution of the project

The SPARK project makes several key contributions to both the academic and esports domains. It integrates artificial intelligence with mobile development to create a data-driven environment that assists in team formation and performance prediction. The platform combines analytical rigor with a user-friendly interface and incorporates gamification elements such as badges and achievements to sustain player motivation. SPARK also contributes educationally by allowing the development team to gain hands-on experience in AI modeling, Flutter framework development, cloud integration using Firebase, and API-based data acquisition. From a societal perspective, SPARK bridges the gap between emerging gaming talent and

professional opportunities, providing an innovative and inclusive solution that supports Saudi Arabia's growing esports infrastructure.

8.5 Future work.

Future enhancements of SPARK aim to expand functionality, improve scalability, and enhance user engagement. Planned improvements include extending AI capabilities to support multiple esports titles beyond League of Legends and implementing a web-based version of the platform to increase accessibility. Additional features will include player ID verification, real-time match tracking, and advanced analytics dashboards. The AI model will be refined using deep learning and more diverse datasets to improve prediction accuracy and personalization. Moreover, Arabic language support will be integrated to better serve local users, and community-driven features such as sponsorship options and tournament funding may also be added. These improvements will help SPARK evolve into a comprehensive, intelligent ecosystem for players, organizers, and esports enthusiasts worldwide.

9 References

[1] Intenta Digital, "Video Game Industry: Statistics, Demographics and Trends in Saudi Arabia," 2022. [Online]. Available: https://intenta.digital/gaming-industry/video-game-industry-saudi-arabia/

[2] Boston Consulting Group, "Gaming & Esports: Media's Next Paradigm Shift," 2021. [Online]. Available: https://www.bcg.com/publications/2021/gaming-and-esports-sector-are-the-next-shift-in-media

[3] P. Xenopoulos, H. Doraiswamy, and C. Silva, "Valuing Player Actions in Counter-Strike: Global Offensive," *arXiv preprint* arXiv:2011.01324, 2020. [Online]. Available: https://arxiv.org/abs/2011.01324

[4] A. Smerdov, E. Burnaev, A. Somov, and A. Stepanov, "AI-enabled Prediction of eSports Player Performance Using the Data from Heterogeneous Sensors," *arXiv preprint* arXiv:2012.03491, 2020. [Online]. Available: https://arxiv.org/abs/2012.03491

[5] The Future of Commerce, "AI in esports: How generative AI + data analytics help Team Liquid win," May 8, 2025. [Online]. Available: https://www.the-future-of-commerce.com/2025/05/09/ai-in-esports/

[6] 360 Research Reports, *"Global Esports Market Report 2024–2031,"* 2024. [Online]. Available:https://www.360researchreports.com/market-reports/esports-market-203854

[7] A. Al Keaid, "Esports in Saudi Arabia: The New Frontier of Entertainment," Mar. 4, 2025. [Online]. Available: https://alwaleedalkeaid.com/2025/03/04/esports-in-saudi-arabia

[8] Python Software Foundation, "Python." [Online]. Available: https://www.python.org/

[9] Scikit-learn, "Scikit-learn: Machine learning in Python." [Online]. Available: https://scikit-learn.org/

[10] NumPy, "NumPy: The fundamental package for scientific computing with Python." [Online]. Available: https://numpy.org/

[11] Riot Games, "Riot Games API." [Online]. Available: https://developer.riotgames.com/

[12] Google, "Flutter." [Online]. Available: https://flutter.dev/

[13] GitHub, "GitHub: Where the world builds software." [Online]. Available: https://github.com/

[14] Atlassian, "Jira: The #1 software development tool used by agile teams." [Online]. Available: https://www.atlassian.com/software/jira

[15] Google, “Firebase.” [Online]. Available: https://firebase.google.com/

[16] Google Cloud, “What is NoSQL?” [Online]. Available: https://cloud.google.com/discover/what-is-nosql

[17] Figma, “Figma: The collaborative interface design tool.” [Online]. Available: https://www.figma.com/

[18] L. M. Costa *et al.*, “Feature Analysis to League of Legends Victory Prediction on the Picks and Bans Phase,” *IEEE Conference on Games*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9590790

[19] C. Lin, “League of Legends Match Outcome Prediction,” *Stanford CS229 Course Project Report*, 2016. [Online]. Available: http://cs229.stanford.edu/proj2016/report/Lin-LeagueOfLegendsMatchOutcomePrediction.pdf

[20] N. Kinkade *et al.*, “Dota 2 Match Outcome Prediction with Supervised Learning,” *Course Project Report*, 2015. [Online]. Available: https://cs229.stanford.edu/proj2015/029_report.pdf

[21] J. Johansson, “Predicting Game Outcome in Dota 2 from Draft Order Using Sequence Models,” *Master’s Thesis*, 2023. [Online]. Available: https://odr.chalmers.se/items/2eb89c52-8991-4aa8-8048-326e21f66a1b

[22] R. A. Bradley and M. E. Terry, “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons,” *Biometrika*, 1952. [Online]. Available: https://doi.org/10.2307/2334029

[23] A. Niculescu-Mizil and R. Caruana, “Predicting Good Probabilities with Supervised Learning,” *ICML*, 2005. [Online]. Available: https://dl.acm.org/doi/10.1145/1102351.1102430

[24] G. W. Brier, “Verification of Forecasts Expressed in Terms of Probability,” *Monthly Weather Review*, 1950. [Online]. Available: https://doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2

[25] B. Zadrozny and C. Elkan, "Transforming Classifier Scores into Accurate Multiclass Probability Estimates," *KDD*, 2002. [Online]. Available: https://dl.acm.org/doi/10.1145/775047.775151

[26] A. E. Elo, *The Rating of Chessplayers, Past and Present*, 2nd ed., 1978. [Online]. Available: https://archive.org/details/ratingofchesspla0000eloa

[27] OP.GG, "OP.GG Official Website," 2025. [Online]. Available: https://op.gg/

[28] Kafu Games, "Kafu Games Official Website," 2025. [Online]. Available: https://kafugames.com/

[29] Saudi Esports Federation, "Saudi Esports Official Website," 2025. [Online]. Available: https://saudiesports.sa/

[30] Amazon Web Services. *Reliability Pillar – Availability*. AWS Well-Architected Framework. Available at: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/availability.html

10 Appendix

10.1 APPENDIX A: Questionnaire for requirements elicitation

Questions:

1. Gender
2. Age
3. Role
4. What is the most difficult aspect you face when participating in tournaments?
5. How would you rate the ease of finding local tournaments?
6. Which features would you like to see in the platform?
7. How important is it for the platform to use AI to calculate your team's winning probability?
8. Do rewards or digital badges encourage you to use the platform?
9. Would you like to see a leaderboard of top local players?
10. What feature would you like to see on the SPARK platform?

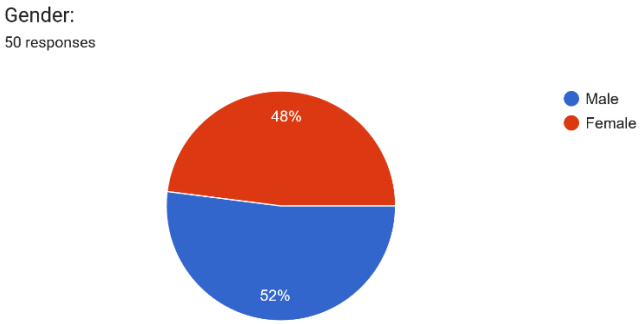


Figure 44:Questionnaire Results – Gender Distribution

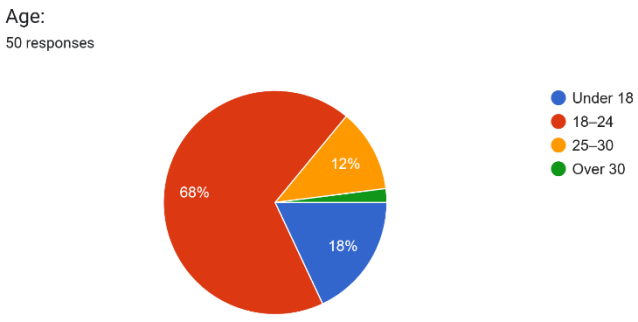


Figure 45:Questionnaire Results – Age Distribution

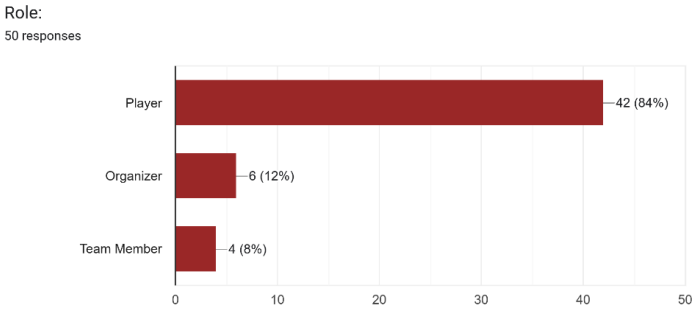


Figure 46:Questionnaire Results – Roles of Participants

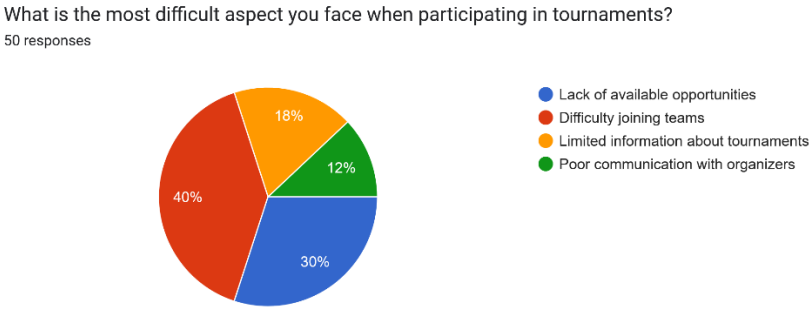


Figure 47:Questionnaire Results – Main Difficulties in Tournaments

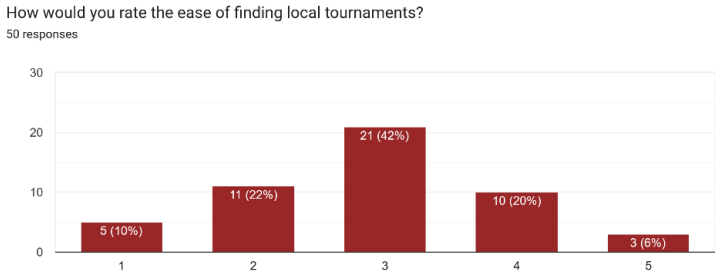


Figure 48:Questionnaire Results – Ease of Finding Local Tournaments

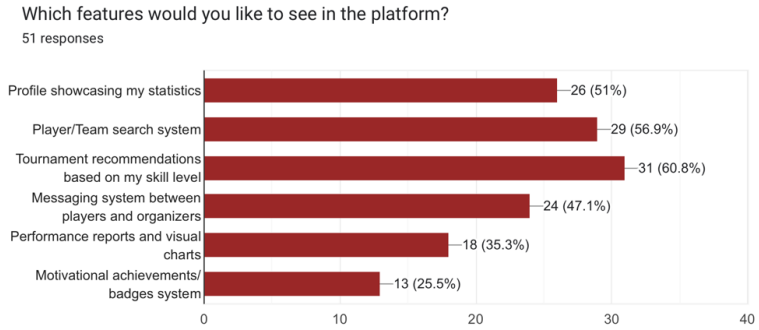


Figure 49:Results Importance of AI in Predicting Win Probability

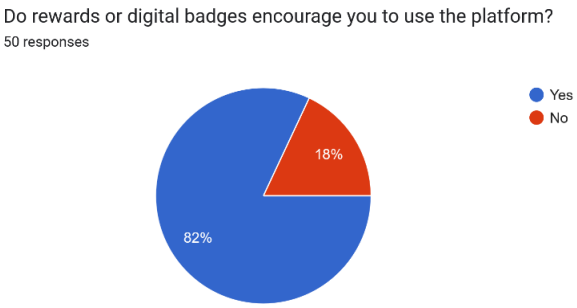


Figure 50:Questionnaire Results Impact of Rewards and Digital Badges on Motivation

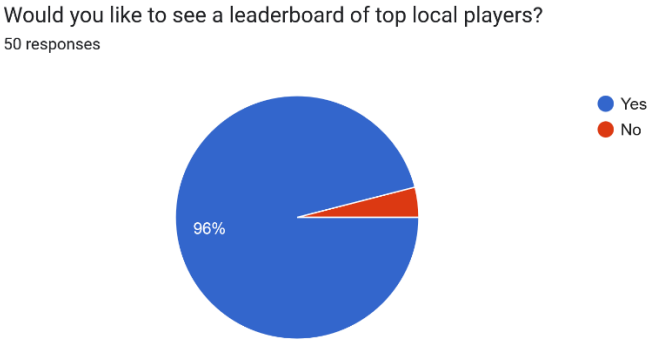


Figure 51:Questionnaire Results Preference for Local Leaderboard

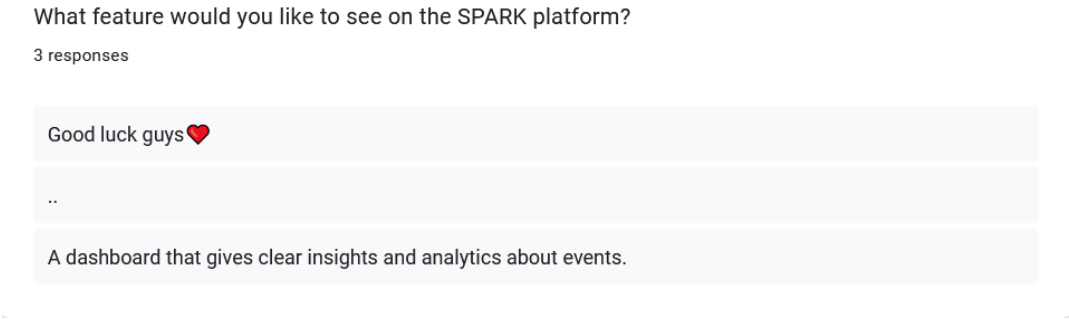


Figure 52:Questionnaire Results – Participants' Suggestions for Platform Features

10.2 APPENDIX B: Interviews for requirements elicitation

10.2.1 Player interviews

Interview 1 outline	
interviewee name: Omar Salem	Interviewers: Aljwharah Alhowidy
Location: Face to Face	Date:16/09/2025
Questions	Answers
Q1. How do you usually discover tournaments or events in the gaming community?	Most of the time through Twitter and Discord. I also hear about tournaments from friends or my team.
Q2. What difficulties do you face when trying to join teams or tournaments?	The hardest part is proving myself. Teams don't always trust me if I don't have a history. I remember missing a tournament once because I found out about it too late. And registering for tournaments can take a while.
Q3. Do you currently use any platforms to manage your player profile? If yes, what are their main drawbacks?	I've used platforms like Saudiesports.sa and kafugames They let me create a profile, but it feels too simple. Usually, it only shows my ID and some very basic stats. What I'm missing is a profile that looks more professional and shows detailed performance, not just basic info.
Q4. How would you prefer your performance to be displayed? (e.g., numerical statistics, visual charts, comparisons with other players)	I like a mix. Numbers are important, but visuals make it easier to understand. For example, charts showing my progress over time, or comparisons with other players, would really help teams see where I stand.

Q5. How important is it for you that the platform shows your full match history when applying for tournaments or teams?	Extremely important. Without a full match history, teams only see highlights, but they don't know if I'm consistent. A complete history builds trust and shows I'm serious.
Q6. Do you find it valuable to have a motivational system such as badges and achievements in the platform? Why or why not?	Yes, it's motivating. It gives me goals beyond just winning matches.
Q7. To what extent do you want direct communication with teams or organizers through the app?	That would be very helpful. Direct chat or a structured messaging system would save a lot of time.

Table 11: interview 1 outline

Interview 2 outline	
interviewee name: Khalid Al-Mutairi	Interviewers: Aljwharah Alhowidy
Location: Face to Face	Date:25/09/2025
Questions	Answers
Q1. How do you usually discover tournaments or events in the gaming community?	Mainly through friends and gaming forums. Occasionally I see ads on YouTube
Q2. What difficulties do you face when trying to join teams or tournaments?	It's hard to show that I'm a good player. Sometimes I get ignored because I don't have much history, and signing up for tournaments can be confusing.
Q3. Do you currently use any platforms to manage your player profile? If yes, what are their main drawbacks?	I tried Saudiesports. It's simple, but it doesn't provide detailed stats, charts, or performance comparisons

Q4. How would you prefer your performance to be displayed? (e.g., numerical statistics, visual charts, comparisons with other players)	I prefer a combination: charts to show progress, numerical stats for accuracy, and rankings compared to peers
Q5. How important is it for you that the platform shows your full match history when applying for tournaments or teams?	Extremely important. Full history builds trust and proves consistency
Q6. Do you find it valuable to have a motivational system such as badges and achievements in the platform? Why or why not?	Yes, if they represent real achievements. It keeps me motivated to improve and participate in more events
Q7. To what extent do you want direct communication with teams or organizers through the app?	Very useful. A built-in chat system would make coordination much easier.

Table 12: interview 2 outline

Interview 3 outline	
interviewee name: Reem Al-Zahrani	Interviewers: Raghad Aldajani
Location: Face to Face	Date: 25/09/2025
Questions	Answers
Q1. How do you usually discover tournaments or events in the gaming community?	Sometimes through Twitter and sometimes through friends from the teams I play with
Q2. What difficulties do you face when trying to join teams or tournaments?	Sometimes registration is complicated or the information is unclear, but other times it's easy if the team is well-organized

Q3. Do you currently use any platforms to manage your player profile? If yes, what are their main drawbacks?	I use them occasionally, but I don't focus much on detailed stats. I mainly want my overall skill level to be visible
Q4. How would you prefer your performance to be displayed? (e.g., numerical statistics, visual charts, comparisons with other players)	I prefer a balanced view: clear numbers for key stats combined with visual charts that show progress and trends over time
Q5. How important is it for you that the platform shows your full match history when applying for tournaments or teams?	Somewhat important, not essential. I think teams focus more on current skill rather than full history
Q6. Do you find it valuable to have a motivational system such as badges and achievements in the platform? Why or why not?	Not necessary. It's nice if it exists, but I don't rely on it.
Q7. To what extent do you want direct communication with teams or organizers through the app?	Preferably somewhat, but I'm fine if communication happens via email or other platforms

Table 13: interview 3 outline

10.2.2 Organizer interviews

Interview 1 outline	
interviewee name: Sara AlMutairi	Interviewers:Aljwharah Alhowidy
Location: Face to Face	Date:23/09/2025
Questions	Answers
Q1. How do you currently manage the tournament registration process?	We use Battlefy platform most of the time, but I still need to double-check player details manually

(Manually, using spreadsheets, through other platforms)	
Q2. What are the main challenges you face when organizing a tournament? (e.g., verifying players, scheduling, communication)	Communication is the hardest part. Players contact us through different platforms, and it's hard to keep everything in one place.
Q3. How do you currently select players or teams to participate? Do you find it difficult to evaluate their skill levels?	We usually look at their profiles and past matches. The data gives us a good idea.
Q4. Would you prefer the platform to recommend players or teams for you based on their skills and performance?	Yes, that would be really helpful. I think if the platform could show suggestions based on past performance and stats, it would make running tournaments a lot easier.
Q5. What difficulties do you face in promoting tournaments and attracting enough players?	Promotion is expensive. Ads cost money, and without them it's hard to reach new players.
Q6. Do you need performance reports before, during, and after the tournament? If yes, what type of data is most important to you? (e.g., KDA, win rate, consistency, etc.)	Yes, but I'd focus more on after the tournament. I want summaries like top performers, win rates, and key highlights.

Table 14: interview 4 outline

Interview 2 outline	
interviewee name: Fahad AlHarbi	Interviewers: Raghad Aldajani
Location: Face to Face	Date: 24/09/2025
Questions	Answers
Q1. How do you currently manage the tournament registration process?	Mostly through spreadsheets. We ask players to fill a form, then we manually

(Manually, using spreadsheets, through other platforms)	enter their data. It's time-consuming but at least we keep full control.
Q2. What are the main challenges you face when organizing a tournament? (e.g., verifying players, scheduling, communication)	Scheduling is the biggest challenge. Players often miss their matches, and rescheduling causes delays. Verifying their IDs is also a hassle.
Q3. How do you currently select players or teams to participate? Do you find it difficult to evaluate their skill levels?	We usually rely on their past matches. If a player has participated in recent tournaments, it's easier to judge. But when the profile doesn't show enough detail, it's difficult to know how consistent they really are.
Q4. Would you prefer the platform to recommend players or teams for you based on their skills and performance?	Yes, that would help. Sometimes it's hard to tell which players are reliable, so seeing suggestions based on stats would make organizing matches easier
Q5. What difficulties do you face in promoting tournaments and attracting enough players?	Reaching the right audience is tough. We rely on Twitter posts and word of mouth, but sometimes the turnout is lower than expected.
Q6. Do you need performance reports before, during, and after the tournament? If yes, what type of data is most important to you? (e.g., KDA, win rate, consistency, etc.)	Yes, especially during and after. I want to see consistency, win rates, and head-to-head comparisons.

Table 15: interview 5 outline

10.3 APPENDIX C: Jira & Github for manage the project

Jira link: https://mraiamalahmed.atlassian.net/jira/software/projects/SK/boards/35

Github link: https://github.com/NoraFisal/2025_GP_34

10.4 APPENDIX D: Questionnaire for User Acceptance Testing

Questions:

1. What is your age?
2. What is your gender?
3. What is your role in the esports community?
4. How many years of gaming experience do you have?
5. What is your main game?
6. The user interface of SPARK is clear and easy to navigate.
7. The team formation and team information sections were easy to understand and use.
8. The platform layout and features were organized in a way that made sense to me.
9. The AI win-probability feature was easy to understand and added value to the experience.
10. I would use SPARK frequently if it were officially released.

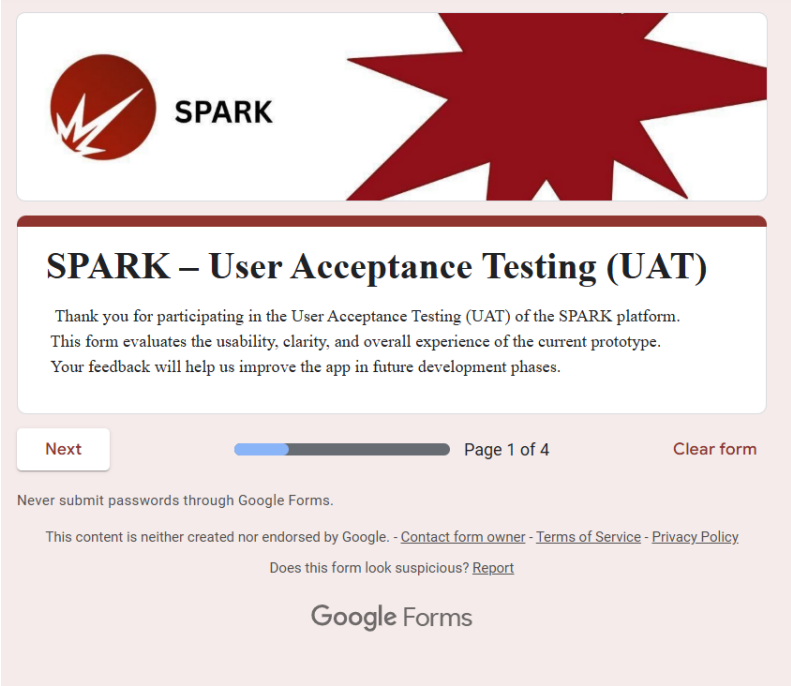


Figure 53:Appendix D – UAT Questionnaire Interface Preview