Esports Organizer

TO DO

Table of Contents

TO DO

Legend

TO DO

0 Introductory Part

0.1 Purpose

TO DO

0.2 Skeleton

TO DO

1 Informative Part

1.1 Team Members

Our project's team is composed entirely of students participating in the Esports Organizer project. Every team member is directly involved in the development, design, and decision-making processes. At this stage, we do not have external partners (such as external experts, sponsors, or client organizations). Since the project is academic in nature, it is fully carried out by the student team without additional stakeholders.

Roles and responsibilities are distributed among the team members as follows:

- Experience Design Team - Focuses on the user interface and overall user experience of the platform. This includes designing intuitive layouts, ensuring visual appeal, and optimizing how users interact with the system to create a seamless and pleasant experience.
- Identity and Data Systems Team - Manages the backend infrastructure, and the data handling. This team ensures that the system can securely store, process, and retrieve user and event information while maintaining performance and reliability.
- Events and Notifications - Handles functionality related to event management and notifications. This includes handling event creation, scheduling, and ensuring timely notifications and updates are delivered to participants and spectators.

- Player and Teams Profiles Team - Oversees features related to player and team identity within the platform. This includes profile creation, team registration, and ensuring accurate representation of players and teams across tournaments and events.

1.2 Current situations, needs, ideas

Current situation

Video games are a widely popular form of entertainment where people can either enjoy casual fun with friends or compete in more structured, competitive settings. However, the competitive gaming scene is broad and diverse, which makes it harder for players to navigate. Most of the information about esports events is scattered across multimple platforms and communities, and because genre preferences vary widely, a lot of events go unnoticed. This lack of variety prevents player from discovering tournaments, joining them, or finding a community they like, reducing opportunities for players to connect with others who share similar interests.

Needs

From this situation a few needs were identified.

- Players need a **clear, reliable and centralized** platform where they can easily discover and register foresports events.
- Organizers need a reliable and effective channel to promote their events and reach their audience.
- Communities need more visibility for their events so they can attract new members and maintain engagement.
- Spectators and fans need an accessible way to follow competitions, track results, and feel part of the event experience.

Ideas

Our project aims to address these needs by implementing a digital platform for esports tournamnets. This include the following features:

- Conducting domain engineering to describe how grassroots and student esports events currently operate.
- Implementing a web-based platform with UI/UX principles that ensure accessibility, adaptability, and usability.
- Centralized events hub where player can **browse and discover** tournaments by game, date, community.
- Registration and management system to simplify how participants sign up and how organizers handle teams and brackets.
- Notification and updates feature to keep participants and spectators informed in real time.
- Community-driven interface that allows players to connect with others, share interests, and follow ongoing competitions.

These ideas were designed to reduce fragmentation, improve the visibility of events, and create a stronger and more connected esports community.

1.3 Scope, Span, and Synopsis

Scope

This project belongs to the broad domain of entertainment, gaming and event organization, with a focus on the growing field of esports. This includes players, organizers, or simply spectators who participate or follow gaming tournaments and events, whether it's online or in person. The scope covers all phases of the software engineering required for the project, including domain description, requirements engineering, software architecture, component design, implementation, and finally testing. It also considers aspects of user experience, communication, and community engagement, as these are essential in the competitive gaming environment.

Span

While the domain of the project is the global esports community, the specific focus (span) of this platform is on small to medium scale tournaments organized by local communities, including student groups and other independent gaming organizations. The platform is designed to help these organizers manage registration, brackets, communication and visibility in a way that is easy to use. Unlike large-scale esport platforms, this project emphasizes inclusivity, adaptability, and simplicity making it suitable for semi-formal competitions.

Synopsis

This project aims to design and develop a web-based platform that facilitates the organization and participation of esports tournaments. The system will centralize key processes such as event discovery, player registration, tournament scheduling, and results publishing. It will also provide real-time notifications, and public results for spectators.

The project will be conducted through standard software engineering phases:

- Domain description to understand the current practices of student esport events.
- Requirements analysis to capture user and stakeholder needs.
- Software architecture and design to define the system's structure and components.
- Implementation of core features in line with the defined requirements.
- Testing and validation to ensure functionality, usability, and reliability.

In summary, this project delivers not only a working system but also serves as a solution to improve how other esports platform organize tournaments and shape user experiences.

1.4 Other activities than just developing source code

Our project is not limited to just source code. Through research, each team worked to ensure the system is meaningful and effective for the gaming community, addressing problems seen in other

platforms.

Domain Engineering

- Situation: The esports and gaming community is broad and diverse, with many different stakeholders (players, organizers, spectators). Without a clear description of the domain, the team risks misunderstanding the environment.
- Need: Developers require a structured understanding of the domain in order to identify relevant actors, workflows, and challenges.
- Implementation: Conducted research on existing platforms (e.g., Start.gg, Battlefy, Toornament) to understand current solutions.

Requirements Engineering

- Situation: There is no formal list of user needs or expectations for this system.
- Need: Developers need precise requirements to define what functionalities the system must provide.
- Implementation: Gather requirements through research, observation of other platforms, and discussions, and translate them into functional and non-functional requirements. Collecting and documenting needs from players, organizers, and communities.

Software Architecture and Component Design

- Situation: Without a defined architecture, the system could become inconsistent or unscalable.
- Need: A structured design that organizes the system into clear components and ensures maintainability.
- Implementation: Planning the system's structure, including front-end, back-end, and database interactions.

Implementation

- Situation: The functionalities identified in the requirements do not yet exist.
- Need: To build a working system that satisfies the requirements.
- Implementation: Setting up the React environment with Vite, researching rendering strategies (CSR, SSR, SSG), and creating UI components.

Testing

- Situation: Without testing, errors and usability issues may remain hidden.
- Need: To validate correctness, performance, and user experience.
- Implementation: Planning for future milestones to ensure usability, accessibility, and reliability.

Documentation

- Establishing AsciiDoc for documentation as code and summarizing research.

Collaboration & Team Organization

- Teams were divided into Experience Design, Communities & Social Features, Events & Notifications, Identity & Data Systems, and Player & Team Profiles. Each team performed initial research, role assignment, and task planning.

By addressing these activities alongside source code development, the project ensures a complete and professional software engineering process that increases the chances of success and usability of the final system.

The Milestone 1 accomplishments included: * Experience Design Team - branding, style guides, Figma mockups, user journeys, and feature page prototypes. * Communities & Social Features Team, Data Backend Team - React setup, research on JSX, Virtual DOM, props vs. state, localization, and documentation workflow. * All teams - established project repository, did their corresponding research, GitHub workflows, and cross-team communication for requirements.

1.5 Derived Goals

In addition to its main goals of centralizing event discovery, simplifying registration, and strengthening community engagement, the Esports Organizer project also seeks to achieve several secondary goals:

- Promote fair play and inclusivity in all competitive tournaments.
- Increase the visibility of local and student-led esports initiatives.
- Encourage partnerships among various gaming communities.
- Support the learning process and skill development for players and teams.
- Raise consciousness about esports culture and its values.

These derived goals add value beyond the platform’s main purpose, helping to strengthen communities, improve collaboration, and build long-term engagement in esports.

2 Descriptive Part

2.1 Domain Description

TO DO

2.1.1 Rough Sketch

- Players join a communitie and have the ability to create and join teams advertized on it.
- Organizers announce registration with eligibility rules (age, roster size, region).

- Teams submit rosters with captains before deadlines; some events allow roster changes.
- Seeding is set by rankings, past results, or qualifiers.
- Tournaments use formats like single-elimination, double-elimination, Swiss, or round-robin.
- Teams must check in before matches; missing a grace period may cause a forfeit.
- Results are recorded; tiebreakers decide standings in round-robins or Swiss.
- Spectators and media follow brackets, standings, and schedules.

Figure 2.1.1 – Tournament and Community Interaction Flow. This diagram illustrates how players, organizers, and communities interact in the platform — showing how tournaments and teams are created, advertised, and connected.

2.1.2 Terminology

- **Player**: An individual who registers on the platform to participate in esports tournaments.
- **Team**: A group of players who join together to compete in tournaments.
- **Tournament**: A competitive event where teams compete against each other in a structured format.
- **Match**: A single game or series of games played between two teams within a tournament
- **Organizer**: An individual or group responsible for setting up and managing tournaments.
- **Spectator**: An individual who watches the tournaments and follows the progress of teams and matches.

2.1.3 domain terminology in relation to domain rough sketch

The rough sketch captures raw specifics (e.g., “15-minute grace period,” “map veto order”). The terminology abstracts those specifics into stable concepts (e.g., Check-in, Forfeit, Map Pool, Tiebreaker Policy). Definitions arise after analyzing rough notes; they are not part of the rough sketch itself.

2.1.4 Narrative

Organizers announce a tournament with a published ruleset, format, schedule, and prize pool. Teams register during the registration window, declare a roster, and ensure all players meet eligibility requirements. Seeds are assigned by prior results, qualifiers, or draw. Matches are scheduled by round according to the format. Before each match, both teams check in; if a team fails to appear within the grace period, the opponent advances by forfeit. During play, referees enforce the ruleset, including pauses, coach timeouts, and server/side selections. Infractions result in penalties as outlined by the rules. After a match, the result is recorded and the bracket or standings advance accordingly. When group stages end, tiebreakers determine placement. The tournament concludes with final placements and prize payouts according to the payout schedule.

2.1.5 Events, Actions, Behaviors

- **Event** (instantaneous state change)
  - "Team A has just become eligible."
  - "Match 3 has just concluded 2–1."
  - "Forfeit recorded after grace period."
- **Action** (an act carried out once)
  - Register a team
  - Apply a penalty
  - Record a result
  - Perform map veto
- **Behavior** (multi-step process composed of actions/events)
  - Run a double-elimination bracket: seed → schedule rounds → play matches → move winners/losers → repeat until champion.
  - Conduct a Swiss stage: pair by score groups each round → play → update scores/OMW/SOS → repeat for R rounds.

2.1.6 Function Signatures

Function signatures describe what information is needed to perform an action and what result comes out of it. They make clear:

- **Inputs**: what the action needs (team, player, tournament, match, etc.)
- **Outputs**: what the action produces (confirmation, updated standings, match record, etc.)
- **Failures**: some actions may fail (e.g., late registration), so the signature should allow for that.

Examples

- `registerTeam(team, roster, tournament) → Confirmation | Failure`
- `scheduleMatch(teamA, teamB, time) → Match`
- `recordResult(match, score) → UpdatedStandings`

In short, function signatures show the inputs, outputs, and whether the action can fail.

2.2 Requirements

2.2.1 User Stories, Epics, Features

Epics

- As a gamer, I want to discover local tournaments and communities, so that I can participate on the tournaments and meet players with similar interests to myself.
- As a tournament organizer, I want to announce and manage events so that I can attract the maximun number of participants and grow the competitive scene.
- As a competitive player, I want to track my performance and rankings, so that I can measure progress and compare myself with others.
- As a casual player, I want to join or create teams for my favorite games so that I can play cooperatively and find new friends to play my favorite games.

User Stories

- As a gamer, I want to follow specific communities so that I receive notifications about upcoming events.
- As a gamer, I want to create a new community for a game without an existing one so that I can gather players with similar interests.
- As a competitive player, I want to view my local ranking so that I can see how I compare with others in my region.
- As a competitive gamer, I want to join a team for my favorite game so that I can participate in team-based competitions.
- As an organizer, I want to create tournament brackets so that matches are structured and easy to follow.
- As an organizer, I want to notify users about new tournaments so that they are aware and can sign up.

Features

- Tournament and event search by both game and location.
- Community following and customizable notifications.
- Local and regional ranking system based on tournament results.
- Team creation and management tools.
- Bracket generation for competitions.
- Organizer tools for posting and updating events.
- Option to create new communities for games without an existing competitive scene.

2.2.2 Personas

Persona 1: Alex the Competitive Player

- **Age:** 21
- **Background:** University student, plays multiple esports titles, highly motivated by rankings and performance.
- **Goals:**
  - Find tournaments to test and improve skills.
  - Track rankings and stats across games.
- **Frustrations:**
  - Difficult to keep up with scattered tournament announcements.
  - Lacks a centralized platform to measure performance.

Persona 2: Maria the Organizer

- **Age:** 34
- **Background:** Works in event management, organizes local gaming tournaments on weekends.
- **Goals:**
  - Announce tournaments easily to the right audience.
  - Manage brackets and notify participants quickly.
- **Frustrations:**
  - Promotion spread thin across many platforms.
  - Hard to build consistent communities for recurring events.

Persona 3: Liam the Casual Gamer

- **Age:** 26
- **Background:** Plays games for fun after work, sometimes interested in casual competitions.

- **Goals:**
  - Discover local communities for his favorite games.
  - Join teams to participate in friendly competitions.
- **Frustrations:**
  - Overwhelmed by too many platforms and event sources.
  - Wants simple notifications without constantly monitoring social media.

2.2.3 Domain Requirements

- Events must be associated with an existing videogame.
- Every event created must have at least one organizer.
- Only the event organizer should have permission to edit or cancel their events.
- Each event must have a starting date, ending date, and location, whether it is physical or online.
- Each team must consist of one or more users.
- Events must have a limit of participants.
- Once the limit is reached, no more users should be allowed to register for the event.
- Once an event is over, no registrations or modifications to the event should be allowed.
- The results of an event must be recorded once the event is finished.
- Appropriate ranking updates must be done based on the results of finished events.

2.2.4 Interface Requirements

- The system must allow for users to create and manage events.
- The system must allow users to search for events and communities within specific locations.
- The system must track, and update user rankings as needed.
- The system must allow users to join teams and communities of their choice.
- If a community or team does not exist for a certain game, the system must allow the user to create one.
- The system must allow event organizers to edit their events, start and end dates and location, until the event has started.
- Any registrations passed the start or end of an event must not be allowed.
- The system must allow event organizers to record event results once the event has finished.
- The system must notify users of new events relevant to their interests.

2.2.5 Machine Requirements

- The system must support at least 450 users at a time with an average response time of 2 seconds or less.

- The system must be available at least 99% of the time per month.
- The system must be able to keep user data secure within the website.
- The system must be able to handle at least 750 registered users without major performance decrease.

2.3 Implementation

The implementation stage translates the requirements described in Section 2.2 into a working system. While requirements define **what** the platform must achieve, the implementation details **how** these goals were realized.

The **software architecture** captures the big picture of the system: * Main modules include authentication, user profiles, tournament management, match reporting, and community management. * These modules communicate through well defined APIs and shared data stored in Firestore. * A web interface built with React ensures accessibility and usability for different types of users like players, organizers, communities.

The **software design** explains how each component is realized: * **Authentication:** Implemented with Firebase Authentication and OAuth2 providers such as Google, Discord, and Twitch. * **User profiles:** Stored in Firestore, including usernames, emails, stats, teams, and communities. * **Tournament management:** Collections store brackets, matches, and schedules, enforcing limits on participants like defined in Section 2.2.3 Domain Requirements. * **Match reporting:** Organizers and players update results, which automatically update user rankings. * **Communities and teams:** Users can create or join them, ensuring inclusivity and adaptability.

Diagrams and screen mockups complement the text: * Architecture diagrams highlight module interactions. * Sequence diagrams show workflows such as user login and event creation. * Figma mockups illustrate how the interface supports usage scenarios.

2.3.1 Selected Fragments of the Implementation

Selected fragments are only included when they clarify explanations and complement the documentation. By themselves, they are not useful. For this stage of the project, no fragments are presented since the system is still under development. Once code and designs are available, they will be added to illustrate how Section 2.2 requirements are met in practice.

Best practices followed in this documentation: * **No screenshots** of code — only properly formatted snippets. * **Scalable images** (SVG, PDF) for diagrams and mockups instead of raster images (JPEG, PNG). * **Fragments included only when they clarify**, never for their own sake.

3 Analytic Part

3.1 Concept Analysis

Based on our understanding of the esports domain and the problem space, we identify the following key concepts:

Game vs Gaming Community: Games are the software titles (Tekken, Valorant), while **Gaming Communities** are groups of players who compete in specific games within geographic regions.

Tournament vs Match: Tournaments are organized competitive events with multiple participants, while **Matches** are individual competitions between players/teams within tournaments. The bracket reference indicates tournaments contain structured match progressions.

Geographic Locality: Multiple references to "local," "in our city," and "geographic areas" reveal that competitive gaming operates within **Local Competitive Scenes** - geographically-bounded communities where players can feasibly attend in-person events.

Performance and Rankings: The "3rd place" and "ranking across events" statements show that **Competition Results** and **Player Rankings** are important domain concepts that currently exist in fragmented form.

Individual vs Team Competition: Some games support both individual and team play (e.g., fighting games typically individual, MOBAs typically team-based). Our domain model must accommodate both.

Casual vs Competitive Players: The distinction between someone who "plays games" and someone who "competes in tournaments" is crucial for our domain focus.

3.2 validation and verification

Validation determines whether stakeholders agree with our understanding of the esports domain as we have documented it.

Domain Concept Validation:

Present our identified concepts to stakeholders and ask for their agreement:

- Present our concept of **Gaming Community** as "groups of players who compete in specific games within geographic regions" to community members and verify this reflects their experience
- Share our understanding of **Local Competitive Scene** as "geographically-bounded communities where players can feasibly attend in-person events" with tournament organizers and participants

Domain Understanding Validation:

Present our domain analysis directly to stakeholders:

- Share our understanding that tournaments contain structured match progressions organized in brackets, and verify this reflects how competitive events actually operate
- Show our understanding that competition results and player rankings currently exist in fragmented form across different platforms, and ask stakeholders if this characterizes their current situation

Terminology Validation:

Present our terminology definitions to stakeholders and ask for confirmation:

- Confirm that our definition of **Match** as individual competitions within tournaments aligns with how stakeholders use this term
- Check that our concept boundaries between different domain entities match stakeholder understanding

Verification Strategy

All concepts in the domain are used consistently across documentation, requirements, and architecture. Requirements clearly trace back to domain properties, and every property that affects the system generates the right requirements. The software architecture covers all specified requirements without gaps, with components having clear responsibilities. The data model represents all domain concepts without conflicts. Implementation matches the design, with unit tests covering all components and interfaces working as specified. Finally, every requirement has a matching test case, and testing environments reflect the operational conditions defined.

Success Criteria

Validation Success Indicators: - Stakeholders recognize their experiences in our domain scenarios and confirm our understanding is accurate - Stakeholders agree with our concept definitions and the relationships we've identified between domain entities - When stakeholders suggest modifications, they represent refinements rather than fundamental misunderstandings of the domain

Verification Success Indicators: - All cross-references between project documents are accurate and consistent - Domain concepts are used consistently across all development phases - Requirements properly trace to domain properties without gaps or contradictions - Software architecture adequately addresses all specified requirements without conflicts

Application of Topics

TO DO

LogBook

Section Name	Member	Added or Modified	Description
Rough Sketch	Pedro Bonilla	Added	Added a Flowchart to better explain the way communities, players, teams, and organizers work.