Project Documentation

Informative Document

1.1 Informative Section

1.Name, Place and Date

- E-sports Team
- University of Puerto Rico Mayagüez
- Fall 2025

1.1 Team Members

Our project team is composed entirely of students participating in the E-sport 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 - Responsible for 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 - Responsible for managing 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 - Responsible for the 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 - Responsible for managing 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 Informative Section

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 E-sports events is scatted across many platforms and communities, and because preferences in games genres vary widely, a lot of events go unnoticed. This lack of variety prevents player from discovering tournaments, joining them, or join the community of that game. 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** where they can easily discover and register for E-sports events.
- Organizers need a reliable and effective channel to promote their events and reach their audience effectively.
- 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 E-sports tournamnets. This include the following features:

- Conducting domain engineering to describe how grassroots and student E-sports events currently operate.
- Implementing a web-based platform with UI/UX principles that ensure accessibility, adaptability, and usability.
- A centralized events hub where player can **browser and discover** tournaments by game, date, community.
- A registration and management system to simplify how participants sign up and how organizers handle teams and brackets.
- A notification and updates feature to keep participants and spectators informed in real time.
- A 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 E-sports community.

Countermeasures:

- Domain engineering: analyze the e-sports ecosystem, players, and tournament organizers.
- Requirements engineering: clearly define the minimum functionalities that the platform requires.
- Software design and architecture: design a modular architecture that allows for scalability and future integrations.
- Implementation: frontend (UI/UX) and backend (event management, users, notifications) development.
- Testing: unit and integration tests to ensure quality.
- Deployment: deployment on a server accessible to test users. = 1.3 Informative Section

1.3 Scope, Span, and Synopsis

Scope

This project belongs to the big domain of entertainment, gaming and event organization, with a focus on the growing field of E-sports. This includes players, organizer, or just spectators who participate or follow gaming tournaments and events, whether its 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 important for the competitive gaming environment.

Span

While the domain of the project is the global E-sports community, the specific focus (span) of this web is on small to medium scale tournaments organized by local communities, including the student's group and other independent gaming community. The platform is designed to help these organizers manage registration, brackets, communications and visibility in a way that is easy to use. Unlike large scale E-sport platforms, this projects emphasis inclusivity, adaptability, and simplicity making it suitable for semi-formal competitions.

1.3.2Synopsis

This project aims to design and develop a web-based platform that facilitate the organization and participation of E-sports tournaments. The system will centralize key process like event discover, player registration, tournament registration and schedule. 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 E-sport events.
- Requirements analysys 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 serve as a solution to improve the way other E-sports platform organize tournaments and user's experiences. = 1.4 Informative Section

1.4 Other activities than just developing a source code

Our project is not limited to just source code. Behind everything each team did the respective research in order to develop a system that is meaningful and effective for the gaming community, helping resolved problems that affect other platforms.

Domain Engineering

- Situation: The E-sports 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: Conduct domain research and analysis before implementation.

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 others platform, and discussions, and translate them into functional and non-functional requirements.

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: Define an architecture/workflow that supports tournament management (data team), communication and events, and user access in a modular way.

Implementation

- Current Situation: The functionalities identified in the requirements do not yet exist.
- Need: To build a working system that satisfies the requirements.
- Implementation: Implement the modules incrementally, following good coding practices and version control.

Testing

- Current Situation: Without testing, errors and usability issues may remain hidden.
- Need: To validate correctness, performance, and user experience.
- Idea: Perform unit testing, integration testing, and user testing before deployment

Deployment and maintenance

- Current situation: The system must eventually be made available to real users.
- Need: Ensure the system can be accessed by the intended audience and remain functional over time.
- Idea: Plan deployment (initially in a limited scope, etc, for a student or local gaming community) and establish a process for updates and maintenance.

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.

1.5 Informative Section

Descriptive Document

2.1 Descriptive Section

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. = Analytic Document

3.1 Informative Section

3.1 Concept Analysis

Concept Formation and 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 Informative Section

3.2 Validation and Verification

Validation Strategy

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