Product Vision & Prototype (PVP)

Project Management Laboratory

21st March, 2025
LGP-17

António Augusto Brito de Sousa, 202000705
Daniel Cabral Bernardo, 202108667
Gustavo Nuno Ferreira Tarábbia, 202104655
Leonor Silva Santos de Azevedo Maia, up202302864
Pedro de Almeida Lima, 202108806
Rodrigo Campos Rodrigues, 202108847
Rodrigo José de Castro Pinheiro, 202403040
Tomás Alexandre Soeiro Vicente, 202108717
Tomás Eiras Silva Martins, 202108776

Index

Index **2**
1. Product Vision **3**
2. Prototype **4**
3. Requirements **4**
- Functional Requirements 4
- Non-Functional Requirements 6

4. User Stories **7**
5. Architecture **10**
6. Technologies **11**

1. Product Vision

As a way to present the reason for our scoreboard's existence and identify the main users, as well as to highlight what makes it different from existing solutions in the market, we present the Product Vision Statement below. This statement outlines key and crucial points and serves as a brief description of the desired future state that will be achieved through the development and deployment of our product.

For any indoor sports entities (clubs and federations)

Who need an affordable way to enhance game experiences and streamline event recording

The ZScore **is a** digital scoreboard and event management system

That modernizes indoor sports events by displaying rich game information and simplifying official record-keeping

Unlike traditional analog scoreboards and manual recording methods

Our product creates a more engaging experience for spectators, providing recognition for players, and increasing game officials' performance through an intuitive interface that enables rapid event recording - all at an accessible price point

Vision: Create an aesthetically appealing scoreboard and an efficient scorer's table.			
Target Group: Sports clubs, federations, event organizers, and sponsors aiming to improve match operations and audience experience.	**Needs:** Clubs need efficient, professional match control and an engaging display system, being also easy-to-watch for the audience.	**Product:** A smart, integrated digital scoreboard and scorer table connected to zerozero's platform, providing live statistics and event control.	**Business Goals:** Boost zerozero's presence in sports tech, generate revenue through hardware sales and sponsorships, and create value for clubs.

Tabel 1 - Product Vision Board

2. Prototype

For the prototype, we created the following video, to illustrate the functionality of our product.

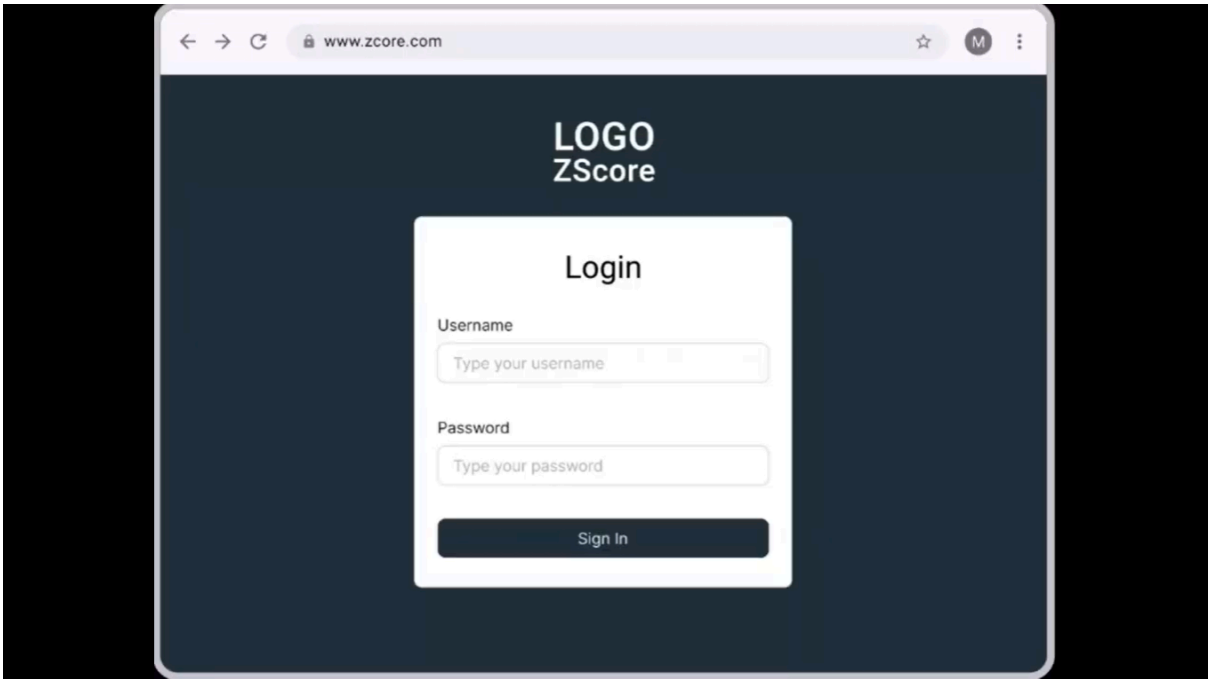

Figure 1 - Prototype video link

3. Requirements

- **Functional Requirements**

Code	Rule	Description
FR01	Intuitive touch-based input system	The interface must follow established touch UI/UX patterns with consistent button placement and sizing appropriate for fast-paced game environments. The system must provide tactile and visual feedback to confirm successful data entry. Sport-specific templates must automatically

		organize input controls relevant to the selected sport. Input areas must be sized and spaced to minimize accidental entries during high-pressure game situations.
FR02	Game setup wizard for pre-game configuration	The user must have access to a pre-game setup wizard so that the game information can be validated and updated by the user.
FR03	Login	The user responsible for the scoreboard must have an account which will then give them access to the games they are responsible for.
FR04	Main Page	The system must have a main page that allows users to search for a specific game. To facilitate this process, the page must display different filters (sports, leagues, time of the day, court). In addition, a settings button must be accessible.
FR05	Board Alert	The board shall provide visual alerts for critical game events (e.g., goals, fouls, timeouts, game end).
FR06	Multi-User Collaboration on the Table	The system shall allow multiple users to simultaneously operate the control panel from different devices or browser tabs. Each user can have specific responsibilities, such as one managing fouls while another controls the game clock. All updates, deletions, and new game events shall be synchronized in real-time across all connected devices, ensuring seamless coordination and accurate game management.
FR07	Manual Override of Any Value	Users must be able to manually change any game value (e.g., score, fouls, penalties) at any time without restriction.
FR08	Time Control	The system must allow users to start, stop, pause, and rewind the game clock and other timers (timeouts, ball possession timers).
FR09	Log Acess	A detailed history log should be available for review and accountability.

Table 2 - Functional requirements

- **Non-Functional Requirements**

Code	Rule	Description
NF01	Availability	The system must be available 99 percent of the time every day.
NF02	Performance	The system must display updates within 500ms of input and handle a rapid succession of events without lag.
NF03	Data integrity	The system must ensure the maintenance and assurance of data accuracy and consistency.
NF04	Usability and Compatibility	Interface learnable within 10 minutes for new officials. The system must be compatible with modern browsers, including Chrome, Firefox, and Edge. The control panel interface must be optimized for 10” tablets or larger. The system must be responsive and work correctly on mobile devices and desktops.
NF05	Security	Only authorized users should be able to edit game events and statistics.
NF06	Scalability	The system must allow for easy addition of new sports without the need for major code modifications. The database must be designed to support a large growth in the number of games recorded over the next years. System updates must be applied without impacting ongoing matches.
NF07	Maintenance	The source code must follow development standards (E.g. design patterns) to facilitate maintenance and have proper documentation.
NF08	Compliance with Sports Standards	The system must follow the official rules and standards of each sport (handball, basketball, volleyball, hockey, and futsal…) to ensure the accuracy and validity of the information displayed.

Table 3 - Non-Functional requirements

4. User Stories

User stories serve as a powerful tool in agile software development to capture essential functionality from the end user's perspective. By structuring requirements in this user-centric format ("As a [type of user], I want [some goal] so that [some reason]"), our team can prioritize features based on customer value, ensure shared understanding among stakeholders, and maintain flexibility throughout the development process as requirements evolve.

User Story	Priority
As a scoreboard operator, I want to log into my account, so that I can be authenticated.	**Must Have**
As a scoreboard operator, I want a guided pre-game setup wizard, so that I can validate and update game information before the match starts.	**Must Have**
As a scoreboard operator, I want to see a list of games I am assigned, so that I can easily find and select the game I need to officiate.	**Must Have**
As a scoreboard operator, I want to view the details of a game before the match starts (teams, time, location, other referees) so that I am prepared and informed.	**Must Have**
As a scoreboard operator, I want to have a clear indication of my role in a game if there are multiple people assigned to operate the scoreboard, so that we can coordinate our responsibilities.	**Could Have**
As a scoreboard operator, I want my officiating team to be able to access and update the same game scoreboard from different devices simultaneously, allowing each official to manage specific aspects of the game while maintaining a single, synchronized scoreboard display.	**Should Have**
As a scoreboard Operator, I want a main page with search and filtering options, so that I can quickly find and access a specific game.	**Should Have**
As a scoreboard operator, I want to add and edit goals/points, so that I can ensure the correct score is displayed when points are scored or scoring decisions are reversed.	**Must Have**

As a scoreboard operator, I want to record and modify player cards depending on the game rules, so that I can accurately track disciplinary actions and correct any recording errors.	**Must Have**
As a scoreboard operator, I want to log and edit team and individual fouls, so that the correct foul count is maintained throughout the game.	**Must Have**
As a scoreboard operator, I want to record and edit team timeouts, so that I can track remaining timeouts for each team and fix any incorrect entries.	**Must Have**
As a scoreboard Operator, I want to be able to view and edit a detailed history log, so that I can track changes.	**Should Have**
As a scoreboard Operator, I want the changes I introduced in the table to be shown in realtime in the scoreboard.	**Must Have**
As a scoreboard Operator, I want to be able to check and edit, if necessary, the game report after the match ends, including scores, penalties, and any relevant notes.	**Should Have**
As a scoreboard Operator, I want the app to work reliably and offline to a certain extent (e.g., basic game data cached), so that I can use it even if the internet connection is unstable in the venue.	**Could Have**
As a scoreboard Operator, I want to have a small indicator that alerts me if there are any connection issues between the app and the display screen, so that I can troubleshoot quickly.	**Should Have**
As a scoreboard Operator, I want an adapted interface so it fits the matches' needs.	**Should Have**
As a scoreboard Operator, I want to start, pause, and update (rewind or forward) the game clock, so that the game time can be accurately managed.	**Must Have**
As a scoreboard Operator, I want to identify the scoring player, so that goals are accurately attributed.	**Must Have**
As a volleyball scoreboard Operator, I want to signal Ace points, so the game is more dynamic, by showing a pop-up with the information and the player's image.	**Could Have**
As a volleyball scoreboard Operator, I want to be able to manage sets and track the current set score, so that the game progress is clearly displayed.	**Must Have**

As a volleyball and basketball scoreboard Operator, I want to be able to manage substitutions and potentially record player information if needed for game reports, so that player changes are tracked.	**Must Have**
As a futsal scoreboard Operator, I want to track accumulated fouls for each team, so that penalty situations are correctly identified.	**Must Have**
As a futsal scoreboard Operator, I want to be able to issue cards to players, so that disciplinary actions are recorded.	**Must Have**
As a futsal scoreboard Operator, I want to be able to manage penalty kicks and record the outcome (goal, miss, save), so that penalty situations are handled correctly.	**Could Have**
As a Spectator, I want the scoreboard to display visual alerts for critical game events so that I can quickly recognize key moments of the game.	**Could Have**
As a Spectator, I want the scoreboard to be easily readable from anywhere in the venue, so I can follow the game without straining.	**Should Have**
As a Spectator, I want to have a final view of the scoreboard when the game ends, that shows-off the winner and the most relevant events in the game.	**Could Have**

Table 4 - User Stories

5. Architecture

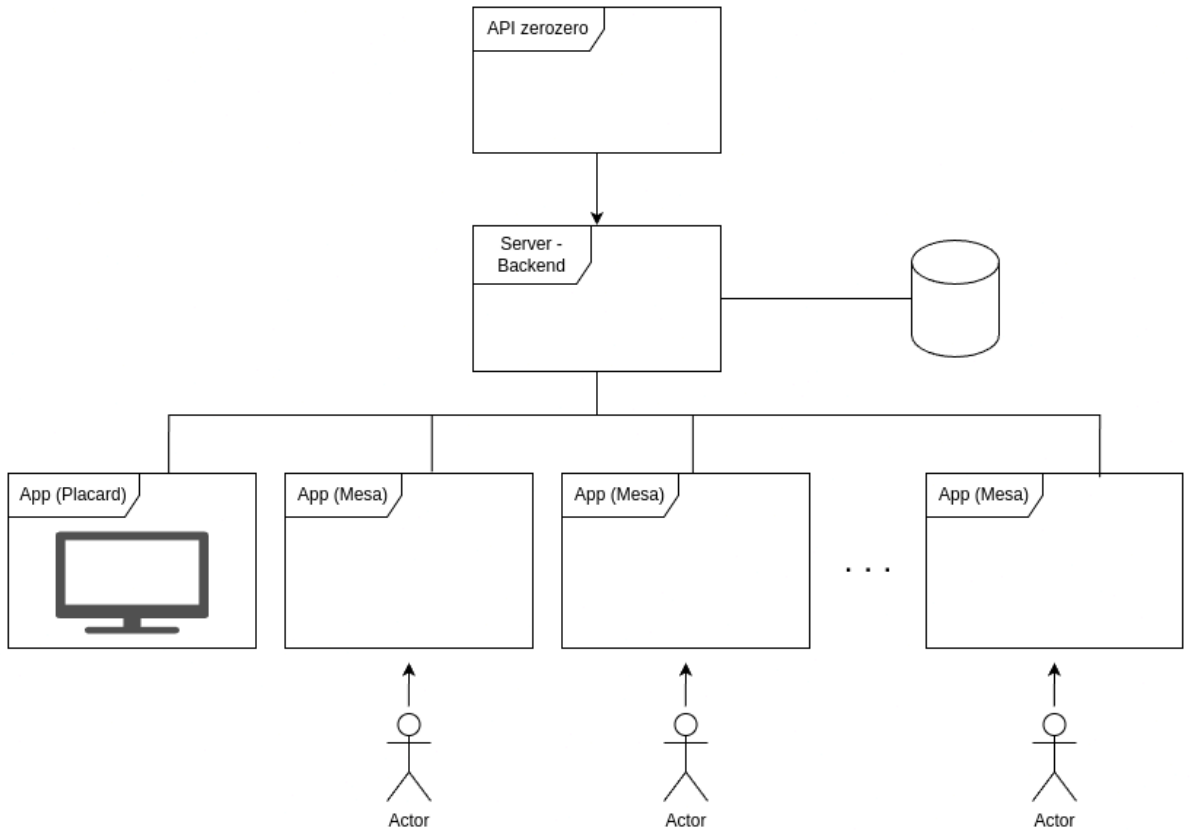

Figure 2 - System Architecture

The ZScore system employs a client-server architecture where a central server acts as the core component, retrieving pre-game information (teams, players, etc.) from the external zerozero API and maintaining persistent data storage through a connected database. The application presents two distinct interfaces: the "Placard" view, which functions as the scoreboard visible to spectators, and multiple "Mesa" views operated by scoring officials at the scorer's table. During gameplay, officials interact with the scorer's table interfaces to record events in real time, with these inputs transmitted to the central server for processing and database storage. All application instances, including both the scoreboard and other scorer's table views, update their information by polling the central server at regular intervals to receive the latest game state. This polling mechanism ensures that as scoring officials document game events through their user-friendly interfaces, the information is eventually reflected on the scoreboard display and synchronized across all scorer's table instances while maintaining data integrity and enabling efficient distribution of scoring responsibilities

among multiple officials. The streamlined workflow supports the system's primary goal: enabling quick decision-making and accurate event registration at the scorer's table while providing spectators with near real-time visual updates.

6. Technologies

For practical development, the following technologies will be used:

- **React**

  ReactJS is a component-based JavaScript library used to build dynamic and interactive user interfaces. We are using React to create a responsive and modular frontend for the project. The frontend is crucial as most computations will be handled on the client side, ensuring a smooth user experience. React's component architecture is particularly advantageous for our sports scoring system, as it enables efficient reuse and customization across different sports contexts. While various sports will require unique interface elements and functionality, many core components can be designed once and repurposed with sport-specific modifications. This approach significantly reduces development time and maintenance overhead while maintaining consistent user experience and visual identity across all implementations.

- **Vite**

  Vite is a fast development tool for React and other modern frontend frameworks. It quickly compiles code and updates the browser instantly. It also includes a built-in server for serving the frontend and supports fast reloading, making development smoother and more efficient.

- **PHP**

  PHP is a server-side scripting language designed for web development. Most of our backend will be built using vanilla PHP, which is responsible for setting up the REST API and WebSockets. There are multiple technologies that could fit this purpose, but PHP is one that we are more comfortable with and is the one that is easier for the client (zerozero) to expand upon.

- **MariaDB**

  MariaDB is an open-source relational database. We are using MariaDB to store and manage all the data from the games. Also, even after the game ends, zerozero can use this stored data in their website. Here, there are also multiple technologies that could fit this purpose, but MariaDB is the one that is easier for the client to expand upon.

- **REST API**

  A REST API is a web service that allows communication between client and server using HTTP requests. The first time a placard connects to the server, it will ask for data using a REST API. This is the easiest way to communicate between the frontend and backend and it has become the industry's standard for this.

- **Polling**

  Polling is a technique where the client periodically sends requests to the server to check for new updates. This means that, in this case, both the placard display and the table will repeatedly request the latest game information at set intervals. This approach is straightforward to implement using JavaScript, as it relies on simple HTTP requests through fetch() or XMLHttpRequest within a setInterval() function. Polling keeps the system updated without persistent connections but can cause unnecessary network traffic and slight delays. Despite these drawbacks, it is a useful initial step before implementing WebSockets for real-time updates.

- **WebSockets**

  WebSockets provide a real-time, full-duplex communication channel between app instances. After the first requests using the REST API, a WebSocket mesh network is established between all active instances. When any instance makes changes, these updates are broadcast directly to all other connected instances without server mediation, enabling instant peer-to-peer synchronization in real time. This direct communication ensures all users have access to the most current information with minimal latency, as updates flow directly from the source instance to all other instances simultaneously.

- **NGINX**

  NGINX is a high-performance web server and reverse proxy. We use it to serve the PHP backend, handle REST API requests, and manage WebSocket connections. It also ensures fast and secure communication by balancing loads and supporting HTTPS encryption.

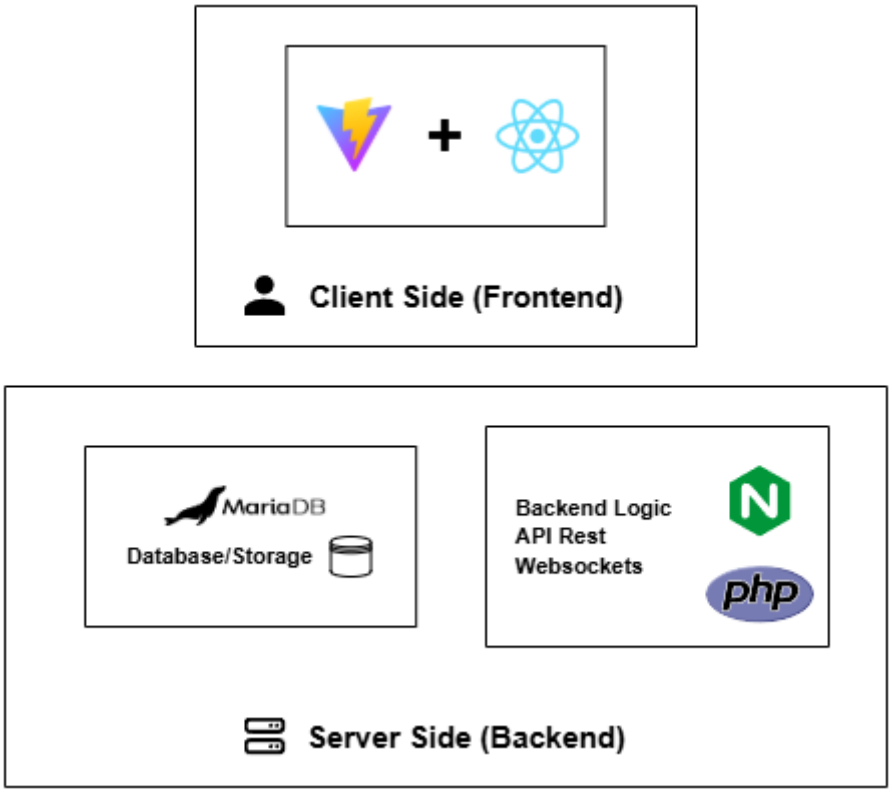

Figure 3 - Tech Stack Diagram