RESPONSIBLE FOR THE DOCUMENT: Andreu Clotas, Xavier Rivera, Eduardo Tobia, Guillem Garcia, Marti Ponsa, Rita Alonso, Rui Lin

INDEX

1. INTRODUCTION

1.1 Objective and motivation

Our objective is to design a web page that gives the user a way to play minesweeper and improve their skills by checking at their statistics in their played games, watching what mistakes they've made in specific games, the right moves to do, and also by locking at their friends games.

1.2 Audience

This document is aimed at all those involved, including developers, testers, etc.

1.3 Definitions

Minefield: Grid where the mines are placed.

Action: An interaction performed by the user on the minefield.

Flag: The action of marking a field of the minefield with a marker to indicate a tile where the user thinks there is a mine.

Click: The action of uncovering a tile of the minefield.

Correct action: An action made by the user **with certainty** that there is no mine in that position.

Incorrect action: An action made by the user **without certainty** that there is no mine in that position, even if there turns out to be no mine. If the user had no logical way to determine the safety of the move, it is considered incorrect.

Stats: The statistics of the user, including their win rate and average number of clicks per game.

Mistake: Click that caused the user to lose the game

2. SYSTEM OVERVIEW

This project aims to provide a more personalised Minesweeper experience, providing the user with advanced statistics about their performance.

Context diagram:

Actor / system	Description
User	Is the one that is going to be interacting with the web app.
Web application	User interface designed to have a better experience playing Minesweeper.
Database	Will store the data. Some examples are: users, passwords, games, statistics for every player…
Chatbot	It will allow the user to ask questions when revising a played game.

Functional modules:

1. User Account Management
    - 1.1. User Registration
    - 1.2. Profile Modification
    - 1.3. Log In/Out
2. Core Gameplay
    - 2.1. Initiate Game
    - 2.2. Game Difficulty
    - 2.3. Reveal Cell
    - 2.4. Flag/Unflag Cell
    - 2.5. Game Win Condition
    - 2.6. Game Lose Condition
3. Game State Management
    - 3.1. Save Game Progress
    - 3.2. Quit Current Game
    - 3.3. Restart Current Game
    - 3.4. Resume Saved Game
4. Game History and Replay
    - 4.1. View Match History
    - 4.2. Replay Game Actions
    - 4.3. Show Optimal Move in Replay
    - 4.4. Identify Losing Move in Replay
    - 4.5. Resume from Replay Point
    - 4.6. Reveal All Mines
    - 4.7. AI Helper
5. Statistics and Analytics
    - 5.1. Clicks/Flags per Game Graph
    - 5.2. Average Solve Time Graph
    - 5.3. Win Rate Graph
    - 5.4. Moves Accuracy Graph

6. Social Features

   6.1. Send Friend Request

   6.2. Manage Friend Request

   6.3. Compare Statistics with Friends

   6.4. View Friend’s Games/Stats

3. FUNCTIONAL REQUIREMENTS

The solution must accomplish the following:

1. Let the user create an account with their own information:
    a. Username and password.
    b. User preferences on whether to share or not their profile statistics with their friends.

2. The user must be able to log in/out of an account

3. Let the user play games in various difficulties (changing the minefield size and number of mines) and then look them up in their match history. The games will be generated from a random seed.
    a. The user must be able to save a game locally.
    b. The user must be able to end a game (will count as lost).
    c. The user must be able to restart the current game (The game in progress will be quit and saved as lost, and a new one will be generated with the same seed).
    d. The user must be able to load and continue a saved game.

4. When looking a played game, the user must be able to:
    a. Watch all the actions they did.
    b. Press a button to show all the possible correct and incorrect actions at any point.
    c. Watch the mistake where they lost the game.
    d. Continue the game at that point (storing it as a new game).
    e. Show all mines (it will block the option to continue the game).
    f. Talk to a chat box to answer any questions.
    g. Let the user see their statistics.

5. Availability of different graphics relating variables (all of them separated by difficulty) such as:
    a. Clicks per game.
    b. Average time to solve.
    c. Winning rate.
    d. Average incorrect actions per game.

6. Availability to add friends to the user list via a friend code and a notification box to accept the friend request.

7. Availability to compare the user's statistics with friend's statistics (only in the case that they have the option to do so activated in their profile).

    .

8. Having the same functionalities for the match history and looking at a game played with their friends accounts (only in the case that they have the option to do so activated in their profile).

3.1 CATALOGUE OF FUNCTIONAL REQUIREMENTS

User Requirement Code	Functional requirement	*Version*	*Priority*	*Guest reviews*
	Create a username and password	1	High	

	Log in/Out	1	High	
	Changing profile preferences	1	High	
	Play games	1	High	
	Save game	2	Medium	
	Quit game	2	Medium	
	Restart game	2	Medium	
	Continue a saved game	2	Medium	
	Match history	1	High	
	Watch all the actions of a played game	1	High	
	Show the correct actions in any position of a played game	2	Medium	
	Watch the mistake where the user lost the game	1	High	
	Continue the game at any point of a played game	2	Medium	
	Show all the mines	1	High	
	Talk to a chat box while looking at a played game	3	Low	
	Graphic with the clicks/flags per game	2	Low	
	Graphic with the average time to solve per game	3	Low	
	Graphic with the winrate	2	High	
	Graphic with the correct/incorrect moves per game	3	Medium	
	Adding a friend with the friends code	3	Medium	
	Accepting a friend request invitation via a notification box	3	Medium	
	Comparing the user graphics with the user friends graphics	3	Medium	
	Having all the functionalities above for the user friends games	3	Low	

3.2 FUNCTIONAL SECURITY REQUIREMENTS

3.2.1 Treatment of access to personal data

The user password will be encoded (basic)

The user list of friends will only be visible by the user itself (basic)

3.3 User Account Management

3.3.1 User registration

The application must allow a new user to create an account by providing a unique username and a password.

3.3.2 Profile Modification

The application must allow a logged-in user to change their profile preferences or settings. I.e. username and passwords.

3.3.3 Log In/Out

The application must allow a user to log in or out in the application.

3.4 Core Gameplay

3.4.1 Initiate Game

The application should allow the user to start a new game of Minesweeper.

3.4.2 Game Difficulty

The application should allow the user to modify the difficulty of the game. Such as changing the number of cells and number of mines.

3.4.3 Reveal Cell

The game should allow the user to click on an unrevealed cell to reveal its content (a mine, a number indicating adjacent mines, or an empty area). Revealing an empty area should automatically reveal adjacent empty cells and number cells.

3.4.4 Flag/Unflag Cell

The game should allow the user to place a flag marker on a hidden cell suspected of containing a mine. The game shall also allow the user to remove a flag from a cell. Flagged cells cannot be revealed or interacted with by clicking unless it is explicitly unflagged by the user.

3.4.5 Game Win Condition

The game should detect when all cells that do not contain mines have been successfully revealed and declare the game won by the user.

3.4.6 Game Lose Condition

The game should detect when a user reveals a cell containing a mine and declare the game as lost by the user.

3.5 Game State Management

3.5.1 Save Game Progress

The application must allow the user to save the current state of an ongoing game session.

3.5.2 Quit Current Game

The application must allow the user to exit the current game session without saving.

3.5.3 Restart Current Game

The application must allow the user to abandon the current game session and immediately start a new game.

3.5.4 Resume Saved Game

The application must allow a user to load a previously saved game state and continue playing from where they left off.

3.6 Game History and Replay

3.6.1 View Match History

The application must allow the user to view a list or record of their previously completed games.

3.6.2 Replay Game Actions

The application must allow the user to select a game from their match history and watch a replay of all the actions taken during that game.

3.6.3 Show Optimal Move in Replay

During the replay of a past game, the application should provide functionality to show the user what the correct or optimal actions would have been at any specific point in the game.

3.6.4 Identify Losing Move in Replay

During the replay of a past game where the user lost, the application should provide functionality to highlight or indicate the specific action that resulted in the loss. The application should also provide an explanation as to why that specific action resulted in a loss.

3.6.5 Resume from Replay Point

While viewing the replay of a past game, the application must allow the user to start a new game instance that begins from the state at any selected point within that replay.

3.6.6 Reveal All Mines

After a game has ended or during a replay, the system must allow the user to view the locations of all mines on the board for that game.

3.6.7 AI Helper

While viewing the replay of a past game, the user should be able to communicate with an AI agent to resolve any doubts that the user may have. (Need to elaborate on this).

3.7 Statistics and Analytics

3.7.1 Clicks/Flags per Game Graph (Graphic?)

The application must generate and display a graphic visualization representing the number of clicks and flags used by the user during a specific completed game.

3.7.2 Average Solve Time Graph

The application must generate and display a graphical visualization showing the user's average time taken to successfully solve games, tracked over multiple games.

3.7.3 Win Rate Graph

The application must generate and display a graphic visualization representing the user's win rate (percentage of games won).

3.7.4 Moves Accuracy Graph

The application must generate and display a graphic visualization showing the number or ratio of correct versus incorrect moves made by the user per game over time. (Note: What is a correct or incorrect move? If an incorrect move is revealing a mine, then each game only has one incorrect move).

3.8 Social Features

3.8.1 Send Friend Request

The application must allow the user to send a friend request to another user through a unique identifier "friends code".

3.8.2 Manage Friend Request

The application must provide a notification area where users can view incoming and outcoming friend requests. The user must be able to modify outgoing friend requests (canceling them), and accept or reject incoming requests.

3.8.3 Compare Statistics with friends

The application must allow the user to view and compare their own game statistics graphics (as defined in section 3.7) with the statistics graphics of users on their friends list.

3.8.4 View Friend's Games/Stats

The application must allow the user to access and utilize the game history, replay features (section 3.6), and statistics viewing features (section 3.7) for games played by users on their friends list.

5. NON-FUNCTIONAL REQUIREMENTS

5.1 CATALOGUE OF NON-FUNCTIONAL REQUIREMENTS

User Requirement Code	Functional requirement	*Version*	*Priority*	*Guest reviews*
	Game must load fast	*1*	High	
	Must support different browsers	*1*	Medium	
	Must support different resolutions	*3*	Low	
	User's sensitive data must be hidden	*1*	High	
	UI must be easy to understand and use	*1*	High	
	Must support several people playing at once	*3*	Low	

5.2 INTEROPERABILITY REQUIREMENTS

The application must be able to connect easily with external services, such as a chatbot or a statistics tool. It should use secure and standard formats for communication, like HTTPS and JSON. Additionally, it must be compatible with other websites to ensure smooth data exchange. Finally, the system should be designed so that new tools or services can be added without needing to modify the core application.

5.3 RELIABILITY REQUIREMENTS

Achieving a seamless user experience requires the application system to comply with the following standards of reliability:

- Users should be informed in advance if there is any scheduled maintenance that will cause service unavailability.
- If an unexpected error or crash occurs mid-session, the user should be allowed to continue from the last correct point. The system should, however, save the state of the game at that point.
- User information (profile, statistics, history) needs to be stored in a non-volatile memory,with daily automatic backups to prevent data loss.
- The application should be functional and available at least for 99.5% of the month duration.

5.4 EFFICIENCY REQUIREMENTS

The application should provide a responsive and user friendly experience at all times. The application is expected to meet the following efficiency benchmarks:

- The game interface must load in less than 3 seconds when having a standard connection.

- Profile data, statistics and history requests must be served in under 500 milliseconds

- Actions taken by users are to elicit a response in under 100 milliseconds from the system.

- The server should not show performance deterioration when up to 100 players are engaging with the system concurrently

5.5 USABILITY REQUIREMENTS

1. Intuitive Usability:

   The app must be intuitive enough for a 10-year-old to understand how to start a game. Although statistics might be harder to understand, graphics should balance the difficulty of it.

2. Connection:

   The user must be able to play offline, and offline games should be considered also in the statistics. Any function that needs connection, such as friend's statistics, are excluded from the requirement.

5.6 SECURITY REQUIREMENTS

The minesweeper application must be protected against common web vulnerabilities prior to deployment. User authentication must use secure password practices, user sessions must expire after a period of inactivity, user data must be encrypted and stored securely and accessing other users statistics can only be done if enabled by the other user first.

It is necessary for the developer to test technical vulnerabilities of the web front-ends (publication and/or back office) and correct them before going into production (at least the vulnerabilities typified as high).

The developer must perform technical analysis of the source code and fix the identified vulnerabilities before production release (at least vulnerabilities typified as "blocker" and "critical").

5.7 PORTABILITY REQUIREMENTS

To maximize reach and flexibility for users, the software needs to be accessible and operational across different platforms and environments and be fully functional in the latest versions of the following web browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.

5.8 MAINTAINABILITY REQUIREMENTS

The application source code should be organized in a modular format, which facilitates easy updating and maintenance of the application. It is compulsory to follow clean coding practices as well as proper naming methodologies. Also, all the code must be tested locally before updating it to the shared code.

5.9 ACCESSIBILITY REQUIREMENTS

- 1.1.1 Non-text Content: Provide text alternatives for non-text content
- 1.2.1 Audio-only and Video-only (Pre-recorded): Provide an alternative to video-only and audio-only content
- 1.2.2 Captions (Pre-recorded): Provide captions for videos with audio
- 1.2.3 Audio Description or Media Alternative (Pre-recorded): Video with audio has a second alternative
- 1.2.4 Captions (Live): Live videos have captions
- 1.2.5 Audio Description (Pre-recorded): Users have access to audio description for video content
- 1.3.1 Info and Relationships: Logical structure
- 1.3.2 Meaningful Sequence: Present content in a meaningful order
- 1.3.3 Sensory Characteristics: Use more than one sense for instructions
- (Nuevo) 1.3.4 Orientation: Content does not restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.

- (Nuevo) 1.3.5 Identify Input Purpose: Input fields that collect certain types of user information have an appropriate autocomplete attribute defined.
- 1.4.1 Use of Colour: Don't use presentation that relies solely on colour
- 1.4.2 Audio Control: Don't play audio automatically
- 1.4.3 Contrast (Minimum): Contrast ratio between text and background is at least 4.5:1
- 1.4.4 Resize Text: Text can be resized to 200% without loss of content or function
- 1.4.5 Images of Text: Don't use images of text
- (Nuevo) 1.4.10 Reflow: No loss of content or functionality occurs and horizontal scrolling is avoided when content is presented at a width of 320 pixels.
- (Nuevo) 1.4.11 Non-Text Contrast: A contrast ratio of at least 3:1 is present for differentiating graphical objects (such as icons and components of charts or graphs) and author-customized interface components (such as buttons, form controls, and focus indicators/outlines).
- (Nuevo) 1.4.12 Text Spacing: No loss of content or functionality occurs when the user adapts text line height/spacing to 1.5 times the font size, paragraph spacing to 2 times the font size, word spacing to .16 times the font size, and letter spacing to .12 times the font size.
- (Nuevo) 1.4.13 Content on Hover or Focus: When additional content is presented on hover or keyboard focus: The newly revealed content can be dismissed (generally via the Esc key) without moving the pointer or keyboard focus, unless the content presents an input error or does not obscure or interfere with other page content; the pointer can be moved to the new content without the content disappearing; the new content must remain visible until the pointer or keyboard focus is moved away from the triggering control, the new content is dismissed, or the new content is no longer relevant.
- 2.1.1 Keyboard: Accessible by keyboard only
- 2.1.2 No Keyboard Trap: Don't trap keyboard users
- (Nuevo) 2.1.4 Character Key Shortcuts: Provide a mechanism to configure keyboard shortcuts
- 2.2.1 Timing Adjustable: Time limits have user controls
- 2.2.2 Pause, Stop, Hide: Provide user controls for moving content
- 2.3.1 Three Flashes or Below: No content flashes more than three times per second
- 2.4.1 Bypass Blocks: Provide a "Skip to Content" link
- 2.4.2 Page Titled: Use helpful and clear page titles
- 2.4.3 Focus Order: Logical order
- 2.4.4 Link Purpose (In Context): Every link's purpose is clear from its context
- 2.5.1 Pointer Gestures: All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.
- 2.5.2 Pointer Cancellation: To help avoid inadvertent activation of controls, avoid non-essential down-event (e.g., onmousedown) activation when clicking, tapping, or long pressing the screen.
- 2.5.3 Label in Name: For user interface components with labels that include text or images of text, the name contains the text that is presented visually.
- 2.5.4 Motion Actuation: Functionality that is triggered by moving the device (such as shaking or panning a mobile device) or by user movement (such as waving to a camera) can be disabled and equivalent functionality is provided via standard controls like buttons.
- 3.1.1 Language of Page: Page has a language assigned
- 3.2.1 On Focus: Elements do not change when they receive focus
- 3.2.2 On Input: Elements do not change when they receive input

- 3.3.1 Error Identification: Clearly identify input errors
- 3.3.2 Labels or Instructions: Label elements and give instructions
- 4.1.1 Parsing: No major code errors
- 4.1.2 Name, Role, Value: Build all elements for accessibility
- 2.4.5 Multiple Ways: Offer several ways to find pages
- 2.4.6 Headings and Labels: Use clear headings and labels
- 2.4.7 Focus Visible: Ensure keyboard focus is visible and clear
- 3.1.2 Language of Parts: Tell users when the language on a page changes
- 3.2.3 Consistent Navigation: Use menus consistently
- 3.2.4 Consistent Identification: Use icons and buttons consistently
- 3.3.3 Error Suggestion: Suggest fixes when users make errors
- 3.3.4 Error Prevention (Legal, Financial, Data): Reduce the risk of input errors for sensitive data
- 4.1.3 Status Messages: In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.

The generated front end code (html and css) must comply with compliance levels A and AA of the WCAG 2.1 guidelines.

Grammatical analysis

(X)HTML tags must be properly specified (open and close) and pages must be properly formed.

Accessibility analysis

Automatic analysis of the 30 criteria of level A and 20 of level AA.

http://accesibilidadweb.dlsi.ua.es/?menu=criterios-2.1-en

5.10 APPLICABLE STANDARDS AND STANDARDS

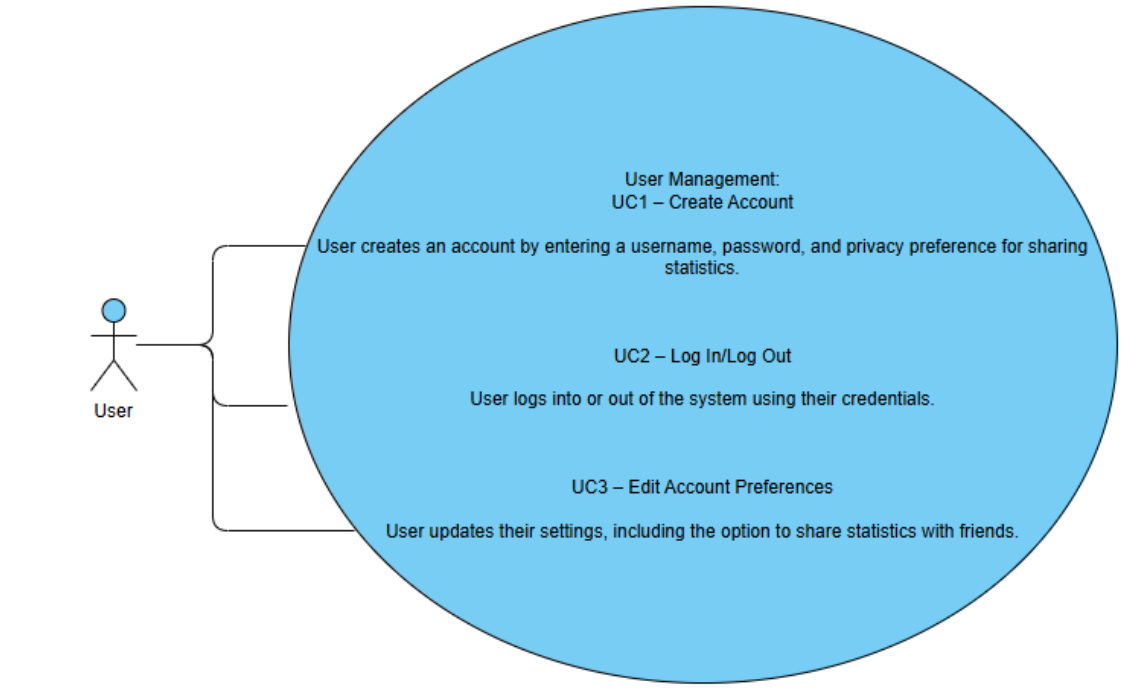
Formatting Guidelines:

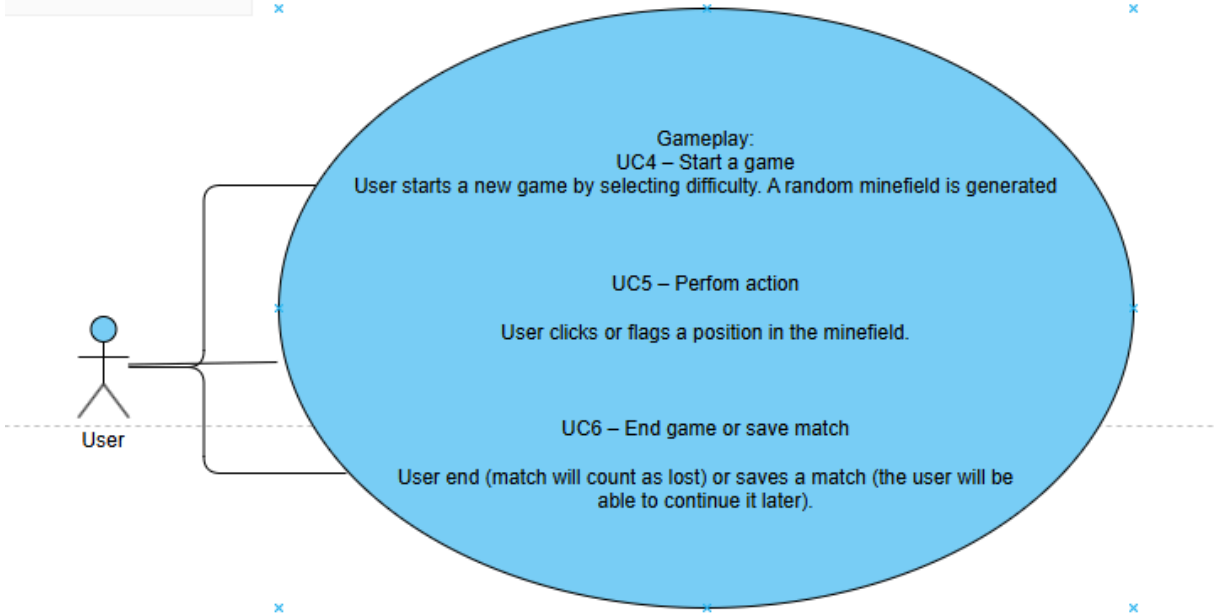
- All numbers will be used with either 0 or 2 decimals (0 in case that the number is an integer).
- All buttons that have the same objective will follow the same format (to be decided).
- User inputs (such as text boxes) must be clearly differentiated to normal texts.

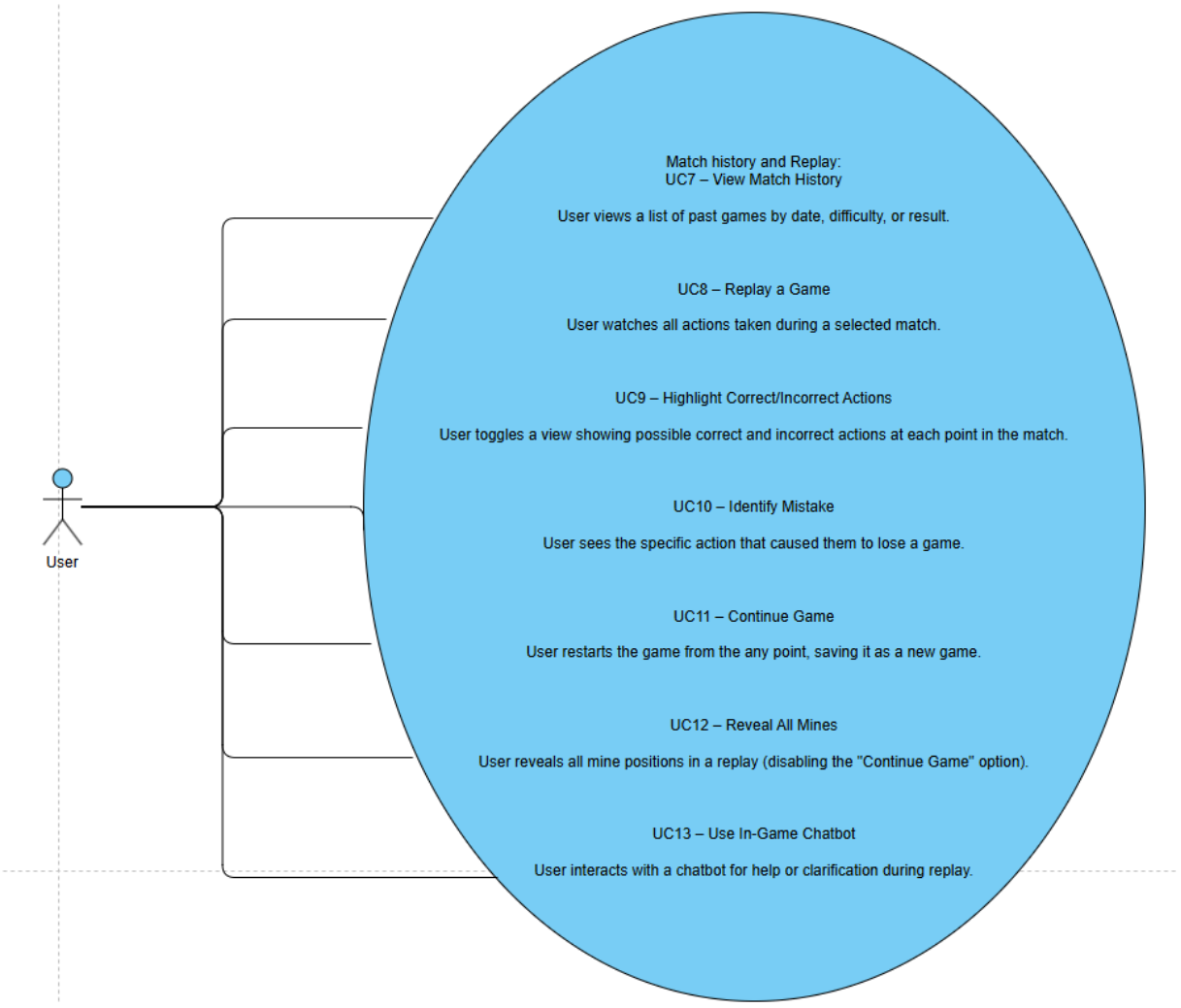
6. USE CASES

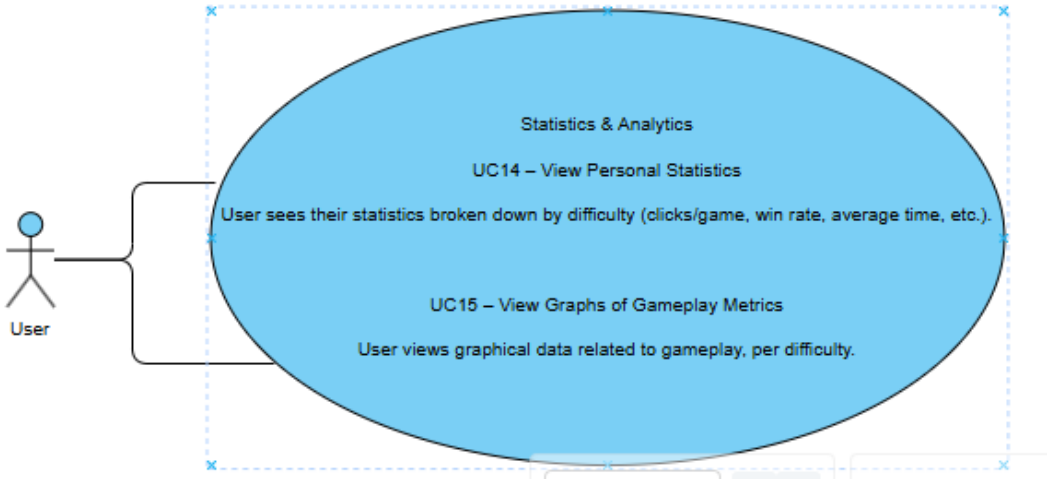
This section includes the set of use cases that describe the external behavior of the system.

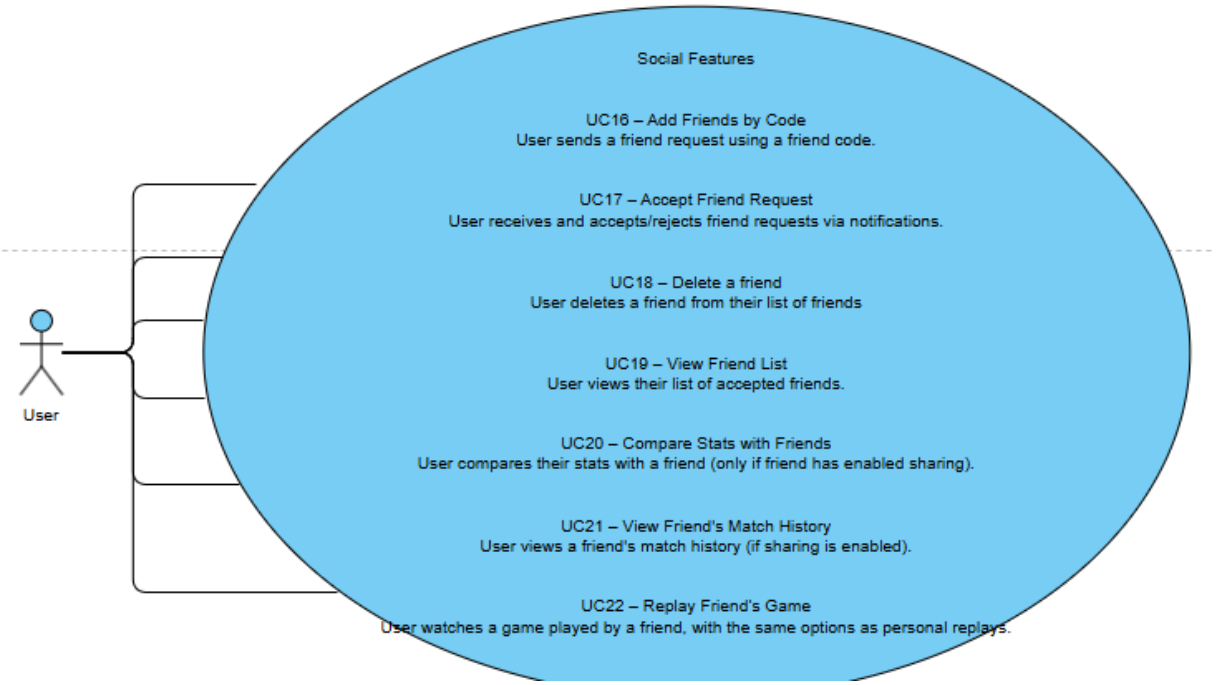
6.1 USE CASE DIAGRAMS AND ACTOR HIERARCHY






Match history and Replay:
UC7 – View Match History
User views a list of past games by date, difficulty, or result.
UC8 – Replay a Game
User watches all actions taken during a selected match.
UC9 – Highlight Correct/Incorrect Actions
User toggles a view showing possible correct and incorrect actions at each point in the match.
UC10 – Identify Mistake
User sees the specific action that caused them to lose a game.
UC11 – Continue Game
User restarts the game from the any point, saving it as a new game.
UC12 – Reveal All Mines
User reveals all mine positions in a replay (disabling the "Continue Game" option).
UC13 – Use In-Game Chatbot
User interacts with a chatbot for help or clarification during replay.
User


Statistics & Analytics
UC14 – View Personal Statistics
User sees their statistics broken down by difficulty (clicks/game, win rate, average time, etc.).
UC15 – View Graphs of Gameplay Metrics
User views graphical data related to gameplay, per difficulty.
User



6.2 CATALOGUE OF ACTORS

6.2.1 User

Type	Primary / Secondary
Description of their role	The User is the **main actor**. This actor interacts with all core functionalities of the Minesweeper application.
Objectives	To play and improve in minesweeper.
Related use cases	All of them

6.2.2 System

Type	Secondary
Description of their role	The system is an actor that is only used to do automatic tasks, no real person is involved
Objectives	To automate things such as creating minefields, looking at mistakes, ….
Related use cases	Indirectly, all of them

6.3 Catalogue of Use Cases

Turn your interest ID	*Use Case Name*	*Actors*
1	Create account	User, system

2	Log in/out	User, system
3	Edit account	User, system
4	Start a game	User, system
5	Perform Action	User, system
6	End/Save game	User, system
7	View match history	User, system
8	Replay game	User, system
9	Correct/Incorrect actions	User, system
10	Identify mistake	User, system
11	Continue game	User, system
12	Reveal mines	User, system
13	Use chatbot	User, system
14	See personal stats	User, system
15	View stats gameplay metrics	User, system
16	Add friends	User, system
17	Accept friend request	User, system
18	Delete a friend	User, system
19	View friend list	User, system
20	Compare stats	User, system
21	View friend's match history	User, system
22	Replay friend's games	User, system

6.4 DESCRIPTION OF USE CASES

6.4.1 Create account

Use Case Name	Create account
Descr. of the Use Case	Allows the user to create an account with a username and a password
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the create account button 2. The user writes the username and a password 3. The system responds if the account can be created a. If it is, it creates the account, logs in and redirects to the main page b. If it can't be created, it gives an error message
Alternate flows	In point 3, if the account can't be created, the user must change the data of point 2
Cond. entrance	The user has not previously created an account The system is available and functional
Cond. departure	The user created an account The user is redirected to the main page

Related requirements	Create a username and password Log in/out Changing profile preferences

6.4.2 Log in/out

Use Case Name	Log in/out
Descr. of the Use Case	Allows the user log in/out of an account
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the log in/out button a. If the user logs out, the system makes the user log out and redirects to the main page b. If the user logs in, redirects to log in page and continues 2. The user puts the username and password 3. The system validates the page a. If it does, it logs in and redirects to the main page b. If it doesn't, it gives an error message
Alternate flows	In point 3, if the data is incorrect, the user must change the data of point 2
Cond. entrance	The system is available and functional
Cond. departure	The user logged in/out to an account The user is redirected to the main page
Related requirements	Log in/out

6.4.3 Edit account

Use Case Name	Edit account
Descr. of the Use Case	Allows the user to edit the information of an account (the username, password and if the user wants to share their games data to their friends)
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the profile button 2. The user edits the desired information 3. The user selects the save button 4. The system responds if the account can be edited a. If it can, it edits the account b. If it can't, it gives an error message
Alternate flows	In point 4, if the account can't be created, the user must change the data of point 2
Cond. entrance	The user is logged in to an account The system is available and functional

Cond. departure	The user edited an account
Related requirements	Changing profile preferences

6.4.4 Start a game

Use Case Name	Start a game
Descr. of the Use Case	Allows the user to start a game with a specific difficulty
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the start new game button 2. The user selects a difficulty 3. The system responds with a randomly generated minefield (depending in the difficulty)
Alternate flows	No alternate flow
Cond. entrance	The system is available and functional
Cond. departure	The user created a minefield to play
Related requirements	Play game Save game

6.4.5 Perform an action

Use Case Name	Perform an action
Descr. of the Use Case	Allows the user to create an account with a username and a password
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects an action 2. The system responds by doing that action a. If the action is a click, it edits the minefield i. If the new edited minefield loses the game, the loss message is shown and the game is saved ii. If it wins the game, the win message is shown and the game is saved iii. If it doesn't win and it doesn't lose, the game countines b. If the action is a flag, it edits the minefield and continues the game
Alternate flows	No alternate flow
Cond. entrance	The user has a minefield to play The system is available and functional

Cond. departure	The minefield information is edited with the desired action
Related requirements	Save game

6.4.6 End/save game

Use Case Name	End/save game
Descr. of the Use Case	Allows the user to end or save a game
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects to end or save a game 2. The system responds by doing that action a. If the action is to save, it saves the minefield and redirects the user to the main page b. If the action is to end, it saves the minefield, maks the game as ended and redirects the user to the main page
Alternate flows	No alternate flows
Cond. entrance	The user has a minefield to play The system is available and functional
Cond. departure	The minefield information is saved and in the case that the option is ending the game, the game is marked as ended
Related requirements	Quit game Save game

6.4.7 View match history

Use Case Name	View match history
Descr. of the Use Case	Allows the user to view the match history
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the match history button 2. The system returns a list with all the matches the users did
Alternate flows	No alternate flows
Cond. entrance	The user is logged in to an account The system is available and functional
Cond. departure	The user has a list of the matches they did

Related requirements	Match history

6.4.8 Replay a game

Use Case Name	Replay a game
Descr. of the Use Case	Allows the user to replay all the actions of a game
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the replay a game button 2. The system responds by showing the replay with the following buttons a. Replay game (1 movement per second, can modify the speed with buttons next to it) b. Previous/Next move c. Got to mistake (message that was shown) d. Reveal mines (blocks the continue game button) e. Continue game (from the actual point) f. Show correct actions g. Chatbot
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The user has a game to select The system is available and functional
Cond. departure	The user sees the replay menu with the selected game
Related requirements	Watch all the actions of a played game Show the correct actions in any position of a played game Watch the mistake where the user lost the game Continue the game at any point of a played game Show all the mines Talk to a chat box while looking at a played game

6.4.9 Highlight correct actions

Use Case Name	Highlight correct actions
Descr. of the Use Case	Allows the user to see all the possible correct actions in the actual game state
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the Highlight correct actions button 2. The system responds by highlighting all the possible correct actions in the game state
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The user has selected a game The system is available and functional

Cond. departure	The user sees the replay menu with all the possible correct actions in the actual game state
Related requirements	Show the correct actions in any position of a played game

6.4.10 Identify mistakes

Use Case Name	Identify mistakes
Descr. of the Use Case	Allows the user to create an account with a username and a password
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the identify mistake button 2. The system responds by moving the replay to the action that lost the game
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The user has selected a game The system is available and functional
Cond. departure	The user sees the replay menu with the selected game at the moment it lost the game
Related requirements	Watch the mistake where the user lost the game

6.4.11 Continue game

Use Case Name	Continue game
Descr. of the Use Case	Allows the user to continue a game from the actual game state
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the continue game button 2. The system responds by creating a new game with the initial state equal to the actual game state of the replay and redirects the user to the menu to play the game
Alternate flows	No alternate flows

Cond. entrance	The user is logged in The user has selected a game The user hasn't clicked the button to show the bombs The system is available and functional
Cond. departure	The user is redirected to the game menu with the game having the same positions as the one where the button was clicked
Related requirements	Continue the game at any point of a played game

6.4.12 Reveal all mines

Use Case Name	Reveal all mines
Descr. of the Use Case	Allows the user to see all the mines in the minefield
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the create account button 2. The system responds by showing all the mines in the minefield
Alternate flows	In point 3, if the account can't be created, the user must change the data of point 2
Cond. entrance	The user is logged in The user has selected a game The system is available and functional
Cond. departure	The user can see all the bombs in the minefield The continue game button is blocked
Related requirements	Show all the mines

6.4.13 Use chatbot

Use Case Name	Use chatbot
Descr. of the Use Case	Allows the user to ask a question to the chatbot
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user asks a question to the chatbot 2. The system sends the question to an AI api 3. The system receives the answer from the api 4. The system gives the answer to the user
Alternate flows	In point 3, if the system can't receive an answer, an error message is shown

Cond. entrance	The user is logged in The user has selected a game The system is available and functional
Cond. departure	The user receives the answer to the question
Related requirements	Talk to a chat box while looking at a played game

6.4.14 See personal stats

Use Case Name	See personal stats
Descr. of the Use Case	Allows the user to see their personal stats
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the see personal stats button 2. The system responds by redirecting to the personal stats page
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The system is available and functional
Cond. departure	The user is redirected to the personal stats menu
Related requirements	Graphic with the winrate Graphic with the average time to solve per game

6.4.15 View stats gameplay metrics

Use Case Name	View stats gameplay matrix
Descr. of the Use Case	Allows the user to see the stats of gameplay metrics
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the see personal stats button 2. The system responds by redirecting to the gameplay metrics stats page
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The system is available and functional
Cond. departure	The user is redirected to the personal stats menu

Related requirements	Graphic with the clicks/flags per game Graphic with the correct/incorrect moves per game

6.4.16 Add friends

Use Case Name	Add friends
Descr. of the Use Case	Allows the user to add friends
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the add friend button 2. The user writes the friend code and selects the add friend button 3. The system searches if the code is correct a. If it is, it gives the request to the corresponding user and shows a message to show that it was correct b. If it is not, it gives an error message
Alternate flows	In point 3, if the the system can't find the friend, the user must change the data of point 2
Cond. entrance	The user has an account The friend has an account The user knows the friend code The system is available and functional
Cond. departure	The user sent a friend request to the friend account The user is redirected to the main page
Related requirements	Adding a friend with the friends code

6.4.17 Accept friend request

Use Case Name	Accept friend request
Descr. of the Use Case	Allows the user to accept a friend request
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user goes to the notification box 2. The user accepts or declines the invitation 3. The system gives a message with the decision to the request sender account
Alternate flows	No alternate flows
Cond. entrance	The user has an account The user received a request The system is available and functional

Cond. departure	The user accepts or declines an invitation The request sender receives the decision
Related requirements	Accepting a friend request invitation via a notification box

6.4.18 Delete friend

Use Case Name	Delete friend
Descr. of the Use Case	Allows the user to delete a friend
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user goes to a friend profile 2. The user selects the option to delete a friend 3. The system deletes the friend from the list
Alternate flows	No alternate flows
Cond. entrance	The user has an account The system is available and functional
Cond. departure	The user deletes a friend from their list of friends
Related requirements	Accepting a friend request invitation via a notification box

6.4.19 View friend list

Use Case Name	View friend list
Descr. of the Use Case	Allows the user to view all the friends that are added
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the view friend list button 2. The system responds by giving the list to the user
Alternate flows	No alternate flows
Cond. entrance	The user has an account The system is available and functional
Cond. departure	The user sees a list of all the friends added to their account

Related requirements	

6.4.20 Compare stats

Use Case Name	Compare stats
Descr. of the Use Case	Allows the user to compare their stats to the friends stats only if the friend has the option to show games to friends activated
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the view friend list button 2. The system responds by redirecting to the stats page with both the user and friend stats
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The user is in the friend profile The friend has the option to show games to friends activated The system is available and functional
Cond. departure	The user is shown their stats and the friends stats in the stats page
Related requirements	Comparing the user graphics with the user friends graphics

6.4.21 View friend match history

Use Case Name	View friend match history
Descr. of the Use Case	Allows the user to see the friend match history
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects the view friend match history button 2. The system responds by showing the match history of the friend
Alternate flows	No alternate flows
Cond. entrance	The user is logged in The user is in the friend profile The friend has the option to show games to friends activated The system is available and functional
Cond. departure	The user is shown a list with all the games of the friend

Related requirements	Having all the functionalities above for the user friends games

6.4.22 Replay friend's games

Use Case Name	Replay friend's games
Descr. of the Use Case	Allows the user to replay a game from a friend that has the option to show them activated
Actors participants	Initiated by user (P) system (S)
Basic event flow	1. The user selects a game from a friend 2. The system responds with the same flow as if it was replaying their own game
Alternate flows	No alternate flows
Cond. entrance	The user has logged in The friend has the option to show games to friends activated The user is in the match history of a friend The system is available and functional
Cond. departure	The user is shown the replay of a friend's game
Related requirements	Having all the functionalities above for the user friends games