Software Requirements Specification for Whack A Prof Game

Contents

1 Introduction

1.1 Purpose

The purpose of this SRS document is to provide a description of the "Whack a Prof" game. This document is intended to be a guide for designers, developers, and testers (Quality Assurance), ensuring that the game is developed according to the specified requirements and that there is a clear understanding of its objectives, mechanics, and expected behavior.

The intended audience for this document includes:

- Backbone: Responsible for implementing the game's features and mechanics based on the requirements.
- Graphics Team: Responsible for developing the game's visuals, including characters, effects (e.g., blood splatter effects, hit effects), items (e.g., mallet), animations (e.g., mallet swing), sounds (e.g, mallet hit sound effects).
- QA Team: Responsible for verifying that the game functions as expected and meets the outlined specifications.

Each group will use this SRS to understand their roles and ensure that the game meets the overall project goals.

1.2 Scope

The software product to be produced followed by the SRS outline, should be named "Whack A Prof". It is an interactive time based gamed inspired by the original "Whac A Mole". In this game users will attempt to hit characters (i.e., Professors or Board of Trustee Members) that pop up randomly on the screen within a limited time frame. The player is rewarded with points for successful hits, and performance will be tracked through a score system. The player will be able to enter their name before playing the game, the player will be able to adjust settings in the game, the player will be able to pause/quit the game.

Benefits of Application

- User Engagement: The game will provide players with an engaging experience with a timed-based gameplay, to which encourages quick reflexes. Features such as real time score updates, random character appearances and background music will enhance user engagement as well.
- Entertainment: The game will provide players with entertainment that challenges the players' reaction time, decision making skills, and hand eye coordination that keeps players entertained and motivated to improve their performance.

Objectives of Application

- Timed Gameplay: Players have a limited time to which will be 90 seconds to hit as many professors and or board of trustees as possible.
- Accurate Scoring: Players will be scored based on how many characters they have successfully hit, the score system will track hits and misses providing real time feedback to which allows players to compare total scores across each game session.

The overall goal of "Whack A Prof" is to replicate and simulate the original "Whack A Mole" game while giving players an enjoyable experience.

1.3 Definitions, Acronyms, and Abbreviations

Definitions, Acronyms and Abbreviations used throughout this SRS document

- Whack A Prof: The game name, where players score points by hitting characters that randomly appear on the screen.

- Hit Detection: The system that detects when a player successfully hits a target and responds accordingly.
- Score System: A system that tracks the player's performance in the game, usually based on the number of targets hit or the accuracy of the hits.
- Target: Objects in the game, moles, that players need to hit to score points.
- **SRS**: Software Requirements Specification.
- **UI**: User interface, the elements through which players interact with the game.
- **QA**: Quality Assurance, the team responsible for testing the game to ensure its functionality and performance meet the specifications.
- **FPS**: Frames Per Second, referring to the refresh rate of the game's visuals, which ensures smooth gameplay.
- **WIP**: Work In Progress - optional/required requirements will be added soon.
- **OPT**: Optional - optional requirements - not necessary but optional requirement.
- **ESS**: Essential - An abbreviation for a essential requirement that would make it so that the software would not be acceptable unless these requirements are provided in an agreed manner.
- **COND**: An abbreviation for a conditional requirement that would enhance the software, but if absent would not make the software itself unacceptable.

1.4 References

- https://en.wikipedia.org/wiki/Whac-A-Mole
- Whack-A-Mole gameplay - Offline Version
- IEEE 830-1998 - https://ieeexplore.ieee.org/document/720574

1.5 Overview

This document is organized to guide the development, design, and testing of the game Whack A Prof ensuring that all teams involved clearly understand the objectives, scope, and requirements. It starts with the Introduction, which defines the purpose and intended audience of the document, including the Backbone, Graphics, and QA teams, each with specific roles to implement, design, and verify the game. The Scope outlines the game's core mechanics, modeled after the classic Whack A Mole game. It also lists the Objectives of the game, which include providing user engagement through timed gameplay and ensuring accurate scoring. The Definitions and Acronyms section clarifies key terms such as SRS, UI, and QA, which are essential for understanding the technical elements discussed throughout the document. The references section provides any material used or referenced in the creation of the Whack A Prof game. This document is structured to ensure that each team understands their role in fulfilling the game's design and functional goals, ensuring the game is developed according to the outlined requirements.

2 Product Perspectives

2.1 System interfaces

The game will consist of 3 things the player can access: Start/Title, Options, Game.

2.2 User interfaces

- **Start/Title Screen** The first thing the player sees when they load the app. It will have the buttons to start the game, exit to desktop, or options.
- **Options/Settings** Allows the user to configure the game to have the most comfortable experience. Options include brightness, quality, etc.
- **Game** The hammer will replace the mouse pointer, there will be 9 holes on screen in a 3x3 layout. Pressing Esc will bring up a menu for options or to return to the title screen.

2.3 Hardware interfaces

The game can be played in either fullscreen or windowed mode. It can be supported by most operating systems (i.e. Windows, Mac, Linux), or most browsers (i.e. Chrome, Firefox). The game will as well utilize the mouse and keyboard for gameplay.

2.4 Software interfaces

The game's name is "Whack A Prof". While the game is in development, there should be beta or alpha versions to make sure the game is at least usable before moving to the next step of development. An example would be "Beta 0.01". When the game is finished, it should be released as "Version 1.0".

2.5 Communications interfaces

The game itself should not require WiFi to play unless the game is on a website. If achievements and the ability to share achievements on social media (i.e. Steam) is developed, it will have to require WiFi.

2.6 Memory

The base game should aim to need 1-2GB worth of storage space due to the simplicity of the game. However, this is not a strict requirement and exceeding that amount is acceptable.

2.7 Operations

When the player starts the game, they are first met with the Title screen. If they need to change the settings to improve their experience, they can go to the Options/Settings button to change it. To actually start the game, the player must press the Start button.

To play the game, the player must press the hit button (default: Left-Click) when the hammer/mouse pointer is over a Prof to score points.

To leave the game, the player must press Esc and return to the Title screen.

2.8 Site adaptation requirements

The game will likely not need any specific site requirements other than the game being embedded.

3 Product Functions

3.1 User Interface Functions

- **Main Menu:** In the main menu, options will be displayed such as "Start Game," "Settings," and "Exit." This interface will feature clear visual cues, allowing users to effortlessly select their desired action.
- **Game Interface:** The game interface will showcase the playing field where characters, such as professors or board members, appear for players to hit. It will include a timer countdown to add urgency and a score display to keep players informed of their progress.
- **Settings Menu:** Players will have the option to customize their experience through the settings menu. Here, they can adjust sound levels, graphics settings, and gameplay preferences to suit their needs.

3.2 Gameplay Functions

- **Character Generation:** The game will randomly generate characters that pop up on the screen, ensuring a diverse and engaging gameplay experience. The variety in character appearance and frequency will keep players on their toes.
- **Hit Detection:** To enhance player interaction, the game will capture hits on characters through mouse or touch input. Immediate feedback will be provided via sound effects and visual cues, such as blood splatter effects, reinforcing the game's theme.
- **Scoring System:** The scoring system will award points for successful hits, with bonuses for speed and accuracy. This system will track the total score throughout the game and display it at the end of each session, encouraging competitive play.

3.3 User Interaction Functions

- **Player Input:** Players will be able to start, pause, or restart the game using keyboard or mouse inputs. Additionally, they will have the opportunity to input their name.
- **Feedback Mechanisms:** To enhance the gaming experience, auditory and visual feedback will be provided for both successful hits and misses. Messages will be displayed after each round, offering encouragement or constructive feedback, such as "Great Job!" or "Try Again!"

3.4 Data Management Functions

- **Score Tracking:** The game will save scores after each gameplay session, after completing a session users can view their score for that session.
- **User Profile Management:** (TBD/OPT) To personalize gameplay, the game will store player names and high scores associated with their profiles. This feature will facilitate continuity for players during future sessions.

3.5 Quality Assurance Functions

- **Error Handling:** To ensure a smooth gaming experience, the game will detect and report any gameplay errors or glitches, such as characters failing to appear. When issues arise, the game will provide instructions or suggestions for resolving them.
- **Testing Tools:** I will include tools for QA testers to simulate various gameplay scenarios. These tools will facilitate thorough testing, ensuring that all functions operate as intended before release.

3.6 Performance Tracking Functions

- **Statistics Display:** After each game session, players will be presented with statistics like hit accuracy, total hits, and average reaction time. This data will help players understand their performance and areas for improvement.
- **Gameplay Analytics:** To continuously enhance the game, I plan to collect data on player interactions. This gameplay analytics will inform future updates and adjustments, creating a more engaging experience.

3.7 Help and Support Functions

- **Tutorial Mode:** To support new players, I will implement a tutorial mode that introduces gameplay mechanics and controls through guided steps. This mode will ensure that all players feel confident as they start their gaming journey.
- **Help Section:** Lastly, a help section will be included, providing FAQs and troubleshooting tips for common issues. This resource will enhance user satisfaction and retention by addressing player concerns effectively.

4 User Characteristics

4.1 User Characteristics

The general characteristics of the users of 'Whack A Prof' are:

- **Educational Level:** Players can range from high school students to college graduates and beyond. The game's simplicity makes it accessible to a wide audience, regardless of their formal education.
- **Experience:**
  - **Casual Gamers:** Gamers who enjoy playing simple, quick games for relaxation or entertainment.
  - **Experienced Gamers:** Gamers who have extensive experience with various types of games, including both casual and complex games.
- **Technical Expertise:**
  - **Basic computer skills:** Players will have computer skills, such as using a mouse or keyboard, navigating the internet, and installing the software.
  - **Intermediate skills:** Some players might have intermediate technical skills, including familiarity with different gaming platforms and troubleshooting minor technical issues.
  - **Advanced skills:** A subset of players might have technical expertise, such as knowledge of game development, programming, or modding games.

5 Constraints

5.1 Regulatory policies:

- **Intellectual property rights:** 'Whack A Prof' avoids using copyrighted material without permission. This includes music, images, and characters.
- **Violence and ethical content:** 'Whack A Prof' ensures the depiction of violence is appropriate for the game's context and avoids promoting real-world violence or stereotypes.
- **Testing and quality assurance:** 'Whack A Prof' is put through testing, expressively handled by the Quality Assurance group of 'Whack A Mole'.
- **Fair play and cheating measures:**

  - Disabling Game Logic: Users can pause the JavaScript execution using developer tools, effectively giving themselves unlimited time to click on moles.

- **Hardware Limitations:**
  - **Memory Usage:** The use of 'setInterval' for the game timer could lead to memory leaks if not properly cleared, especially if the game is restarted multiple times without proper cleanup.
  - **Processing Power:** Frequent updates to the DOM (e.g., showing and hiding moles) can be CPU-intensive, especially if the game runs for a long time or if there are many elements to update.
  - **Compatibility:** The game relies on modern JavaScript features and DOM manipulation, which should be compatible with most modern browsers. However, older browsers might not support all features, leading to potential issues.
  - **Input Devices:** 'Whack A Prof' supports various devices such as keyboards, mice, and touchscreens.

- **Parallel Operation:**
  - **Single-Threaded Nature of JavaScript:** JavaScript handles asynchronous operations using mechanisms like 'setTimeout', 'setInterval', and event listeners. These allow the browser to manage tasks in the background and execute callbacks when certain conditions are met.

- **Control functions:**
  - **Start Game (Start):**
    * Initializes the game by resetting the score and timer.
    * Sets the game state to active and not paused.
    * Displays the first mole and starts the countdown timer.
    * Updates the UI to reflect the game state (e.g., hiding the start button, showing the pause button).
  - **Pause Game (Pause):**
    * Pauses the game by clearing the countdown timer.
    * Sets the game state to paused.
    * Updates the UI to reflect the paused state (e.g., changing the pause button to resume, showing the restart button).
  - **Resume Game (Restart):**
    * Resumes the game from a paused state by restarting the countdown timer.
    * Sets the game state to active and not paused.
    * Updates the UI to reflect the resumed state (e.g., changing the resume button back to pause).
  - **Whack Mole (whackMole):**
    * Handles the event when a mole is clicked.
    * Hides the clicked mole, increases the score, and shows a new mole in a random hole.
  - **Increase Score (increaseScore):**
    * Increases the score when a mole is successfully whacked.
    * Updates the score display in the UI.

- **Signal handshake protocols:**
  - **Single Handshake Protocols:** These are not relevant to 'Whack A Prof' since it does not involve any network communication or multiplayer interactions.
  - **Security Measures:** 'Whack A Prof' does not include any security measures for data exchange or multiplayer interactions.

- **Criticality of the application:**
  - **User Experience:** The game provides a seamless and enjoyable experience.
  - **Offline Play:** The game can be played offline since it does not depend on any external servers or online resources.
  - **Local Resources:** All game logic, assets, and interactions are handled locally within the user's browser.

6 Assumptions and Dependencies

6.1 Assumptions and Dependencies

- **Assumption:** The game will be on a web server that supports modern browsers for example (Chrome, Firefox, Safari) with JavaScript enabled.
- **Dependency:** If a user tries to play the game in an outdated browser or one that does not support JavaScript the game might not work as expected. In such a case, changes to the requirements would be needed, such as checking browser compatibility.
- **Assumption:** The game will be deployed on a website that has a stable internet connection and as well can handle multiple users accessing the website at the same time.
- **Dependency:** If the website hosting the game experiences downtime or poor performance due to limited bandwidth, the requirements might need to change to include performance optimization or the game might be able to be played offline.
- **Assumption:** The game will require users to interact via mouse clicks.
- **Dependency:** If the game needs to support additional and or more input methods, such as keyboard control, game controllers or touch screen, the SRS outline would need to change to accommodate the new input methods and there would need to be necessary changes in game mechanics.

7 Apportioning of Requirements or Optional Requirements

7.1 Apportioning of Requirements or Optional Requirements

- **Multiplayer Mode:** The initial release of the game will only support single-player mode. A future version of the game may include a multiplayer mode, allowing multiple users to compete simultaneously.
- **Delay:** (OPT) Multiplayer support will be postponed to the final version.
- **Additional Characters and Themes:** The initial game will feature a single theme with a limited number of characters (ie, professors, board of trustees). Future versions may introduce additional professors or in game background themes, such as new character designs, animations.
- **Delay:** Additional characters and themes will/may be part of Version 3.0.
- **Mobile App Version:** (OPT) The initial release of the game will be designed for web browsers, while a future version may include a mobile app version for Android and/or iOS devices.
- **Delay:** (OPT) The mobile app version will be postponed for a potential future release.

8 Specific Requirements

8.1 External interfaces

8.1.1 Player Name Input

- **Name of item:** Player Name Input
- **Description of purpose:** Allows the player to enter their name before playing the game which will be used to personalize the game experience
- **Source of input or destination of output:** Input comes from the user via a text field in the main game interface and the name will be stored in a database for later retrieval and use as needed.
- **Valid range, accuracy, and/or tolerance:** The name should be between 1 and 15 characters in length, it as well must consist of valid alphanumeric characters without special characters except spaces and hyphens.
- **Units of measure:** Character length should be 1 to 15 characters.
- **Timing (TBD):** The name input occurs during the initial game setup, it should be processed immediately after the player submits their name.
- **Relationships to other inputs/outputs:** The player's name is used throughout the software itself which includes usability in game stats, and/or in-game references to the player.
- **Screen formats/organization (TBD):** The input field is located at the start/setup of the game at the top section or in a separate input prompt screen that appears before gameplay begins.
- **Window formats/organization:** The name input box appears in a windowed or full-screen mode as part of the game interface during the setup phase.
- **Data formats:** Alphanumeric string (UTF-8 encoding).
- **Command formats:** User will input the name via keyboard input and the "Enter" key will be able to process and submit the name.
- **End messages:** A confirmation message will be displayed after the name is successfully submitted.

8.1.2 Mouse Input (Click/Movement)

- **Name of item:** Mouse Input (Click/Movement)
- **Description of purpose:** Allows the player to interact with the software itself, can be used for various software needed tasks, (ie, manipulating the menu/in game activities).
- **Source of input or destination of output:** Input comes from the player's mouse.
- **Valid range, accuracy, and/or tolerance:** Any mouse click/movement within the software itself is considered valid. While in-game accuracy can be measured if the player clicks directly on the professor or board of trustees, and or utilizing the mallet.
- **Units of measure:** Clicks per second (CPS), click events, mouse movements are/can be measured.
- **Timing:** Mouse clicks are processed in real-time during gameplay/software interaction. The response must be instantaneous to ensure smooth software performance.
- **Relationships to other inputs/outputs:** Mouse clicks can be tied to the game's scoring system. A successful click on a professor/trustee increases the player's score, while a miss may have no effect on the player's score.
- **Screen formats/organization:** The mouse clicks are recognized anywhere within the software itself.

- **Window formats/organization:** The game interface will be responsive to mouse clicks in the desired screen format.
- **Data formats:** Coordinates of the mouse click are recorded such as (x,y pixel positions).
- **Command formats:** Mouse left-click is used to interact with the game.
- **End messages:** No specific end message for mouse inputs are needed but feedback is provided in the form of animations or sound effects when a successful hit is registered on a trustee/professor, and mallet swing animation is played as well.

8.1.3 Keyboard Input

- **Name of item:** Keyboard Input
- **Description of purpose:** Allows the player to utilize their keyboard input to type in username and or pause the game and bring up the in-game menu by pressing the "ESC" key during gameplay.
- **Source of input or destination of output:** Input comes from the player's keyboard.
- **Valid range, accuracy, and/or tolerance:** The "ESC" key must be pressed during gameplay to trigger the menu, input can be valid at any point while the game is active. Other use cases for keyboard input can be considered as well, for example input username.
- **Units of measure:** Keypress event.
- **Timing:** Keyboard keypress events should be immediately processed for a specific task that is needed to be achieved where the keyboard input is necessary.
- **Relationships to other inputs/outputs:** Can be utilized for username input.
- **Screen formats/organization:** The software application will be responsive to keyboard inputs in the desired screen format.
- **Data formats:** Keypress events are tracked and processed.
- **Command formats:** Pressing any key on the keyboard can be described as a command to trigger some sort of action.
- **End messages:** When pressing the "ESC" key a menu message will be displayed that will have options for the end user to see such as "Go Back To Main Menu".

8.2 Functions

- **Validation Checks:** The system shall validate username input in a specific way:
  - Ensure the username is between 1 and 15 characters long, if the length is beyond this range the software will prompt the user with the message: "Please enter a name between 1 and 15 characters."
  - Ensure the username contains no special characters, if special characters are detected, the software will prompt the user with the message: "Special characters are not allowed."
- **Sequence of operations:**
  - Process the input for validation (length and special characters).
  - Return either a success message or the appropriate error message based on the input made by the user.
- **Error handling and recovery:**

- If the input fails validation (e.g., invalid length or special characters), the system will restrict the input and prompt the user to enter a valid username.
- Errors are handled in real-time immediately after submission to ensure that only valid usernames are accepted before proceeding to the game.

- **Relationship of inputs to outputs:**
  - Upon input submission the system generates an output in the form of either an error message (for invalid username input) or a confirmation to continue (for valid username input).

8.2.1 In-Game Score Tracking

- **Validation Checks:** The system shall validate in game score tracking in a specific way:
  - The system shall track mouse input events such as mouse clicks to determine if the player hits or misses a professor/board of trustee (target).
  - A hit box detection can be utilized to verify whether the mouse click falls within the valid range of the target, if the click is within this area, it is counted as a hit otherwise it is counted as a miss.

- **Sequence of operations:**
  - During gameplay the system will register mouse click input such as mouse clicks.
  - During gameplay the system will check whether the click is within the hit box of the professor/trustee.
  - If valid, increment the player's score and play a unique "hit" sound, if invalid click (ie, miss), no score will be awarded, and a distinct "miss" sound will play to inform the player.

- **Error handling and recovery:**
  - If the mouse click does not register within the target area the system will handle it as a miss and provide immediate feedback through sound.

- **Effect of parameters:**
  - The size and position of the hit box determines whether a mouse click is valid (i.e., a hit) or invalid (i.e., a miss), the system's response depends on this validation.

- **Relationship of inputs to outputs:**
  - The mouse input mouse clicks results in two possible outputs:
  - A hit (with a corresponding increase in score and "hit" sound effect).
  - A miss (with no score and a "miss" sound effect).
  - These outputs help guide the player during gameplay.

8.3 Performance Requirements

8.3.1 Static Requirements

- The type of data the software can handle will be in the form of keyboard inputs, mouse inputs

8.3.2 Dynamic Requirements

- Keyboard/Mouse events should be processed immediately less than 0.5 seconds
- Scoring data should be processed in the specified gameplay time during a game round of "Whack A Prof"
- (TBD) Professors/board of trustees should appear and disappear in intervals between 1 to 3 seconds.
- Audio cues for hits/misses must play within 0.2 seconds of corresponding mouse click.
- Scores must be saved within 1 second after game session ends.
- When the "ESC" key is pressed the game must bring up the specified menu within 1 second.

8.4 Software System Attributes

8.4.1 Reliability

- The software should be able to run smoothly and reliably when it has been delivered. To achieve this the game will undergo frequent and thorough testing by the QA team to identify and fix any bugs that arises. The games reliability can be measured by meeting specific benchmarks such as avoiding critical bugs that causes the game to malfunction or crash, achieving consistent frame rate throughout the game session, and providing quick responsive interactions for all user inputs.

8.4.2 Availability

- Factors that are required to guarantee a defined availability level for the entire system, such as recovery procedures, will ensure that if a critical bug occurs during a game session, the game will automatically stop and restart to which will reset the user's current session score.

8.5 User Class

8.5.1 External Interface Requirements

8.5.1.1 User Interfaces

- See 2.2

8.5.1.2 Hardware Interfaces

- See 2.3

8.5.1.3 Software Interfaces

- See 2.4

8.5.1.4 Communication Interfaces

- See 2.5

8.5.2 System Features

8.5.2.1 Introduction/Purpose of feature

- See 8.7.1.1

8.5.2.2 Stimulus/Response sequence

- See 8.7.1.2

8.5.2.3 Associated functional requirements

- See 8.7.1.3

8.5.3 Performance Requirements

- See 8.3

8.5.4 Design Constraints

- It is recommended to keep the game's memory size less than 1-2GB. The game must also be designed to be played on a website, and optionally, as an app.

8.5.5 Software System Attributes

- See 8.4

8.5.6 Other Requirements

- **N/A**

8.6 Object

- This specific part represents entities and components that contribute to gameplay. Each object possesses unique attributes and functions, facilitating interactions that engage users in the core mechanics of the game.

8.6.1 Professor

- **Attributes**: Position, appearance, state (visible/hidden), animation sequence.
- **Functions**: Appears at random intervals from holes, responds to whacking action, triggers animation (e.g., dizziness), and retracts after being hit.

8.6.2 Player

- **Attributes**: Score, accuracy, reaction time.
- **Functions**: Uses a "hammer" tool (controlled by left mouse button or spacebar) to whack professors as they appear, and records successful hits.

8.6.3 Hammer Tool

- **Attributes**: Position, hit detection range.
- **Functions**: Responds to user input, registers successful hits on professors, and plays hit animations.

8.6.4 Game Interface

- **Attributes**: Score display, timer, feedback prompts.
- **Functions**: Updates and displays real-time player score and time remaining, provides visual cues (like feedback on successful hits), and ends game when the timer expires.

8.7 Feature

8.7.1 Feature : Whacking

8.7.1.1 Introduction/Purpose of Feature

- The purpose of the whacking feature is to engage the user in a reflex-based challenge where they attempt to "whack" professors who randomly pop up from various holes. This feature is inspired by the classic "whack-a-mole" game, encouraging the user to respond quickly and accurately with a "hammer" tool.

8.7.1.2 Stimulus/Response Sequence

- Stimulus: A professor's head emerges from one of the holes, prompting the user to respond.
- Response: The user attempts to whack the professor over the head with the provided "hammer" (left mouse button or spacebar) before the professor disappears back into the hole.

8.7.1.3 Associated Functional Requirements

- The system must display professors who pop their heads up from random holes at random intervals.
- The user must be able to use a "hammer" to interact with the game by whacking (left mouse button/spacebar) the professors.
- The system should detect and register a successful whack when the hammer successfully (left mouse or spacebar input while professor animation is still out) hits a professor's head.
- Professors should have an animation of dizziness/getting hit then retreat back in their whole while/after the animation finishes.