Lab 2 Section 3 – ReasonED.io Product Specification

Team Crystal

CS 411W

Professor Thomas J. Kennedy

22 March 2024

Version 1

Table of Contents

3 Specific Requirements

3.1 System Features

3.1.1 Account Management

3.1.1.1 Introduction/Purpose of Feature

This feature allows students and teachers to create accounts on the ReasonED platform, granting personalized access to features and content.

3.1.1.2 Stimulus/Response

Stimulus: Users initiate the account creation process by navigating to the registration page and submitting their details.

Response: Upon submission of registration details via the website's registration page, the system promptly processes the information, verifies its accuracy, and securely stores the account in the database. Immediate confirmation of successful registration is then provided, followed by the dispatch of a verification email to facilitate account activation.

3.1.1.3 Associated Functional Requirements

3.1.1.3.1 Student Registration Form *(O: Morgan)*

The system must present a registration form for students to input account information, including only their email, username, date of birth, and password.

3.1.1.3.2 Student Registration Validation Checks *(O: Nabi)*

The student registration form must enforce validation checks to verify the uniqueness of usernames and emails, and compliance with age restrictions.

3.1.1.3.3 Teacher Registration Form *(O: Morgan)*

The system must present a registration form for teachers to input account information including only their email, password, and teaching level.

3.1.1.3.4 Teacher Registration Validation Checks *(O: Nabi)*

The teacher registration form must enforce validation checks to verify the uniqueness of emails.

3.1.1.3.5 User Type Selection Option *(O: Morgan)*

Users accessing the registration page must be provided with an explicit option to indicate whether they are creating a student or teacher account.

3.1.1.3.6 Dynamic Registration Form Adjustment *(O: Morgan)*

The registration form must dynamically adjust based on the selected user type (student or teacher), displaying relevant fields and validation criteria.

3.1.1.3.7 Strong Password Enforcement *(O: Nabi)*

The registration form must enforce strong password requirements during account creation and password change processes to enhance security.

3.1.1.3.8 Password Creation Guidelines *(O: Nabi)*

Users must be provided with clear guidelines for creating strong passwords containing a mixture of uppercase letters, lowercase letters, numbers, and special characters.

3.1.1.3.9 Password Encryption *(O: Morgan)*

Passwords must be encrypted before storing them in the database to ensure data security.

3.1.1.3.10 Username Whitelisting *(O: Morgan)*

The system must implement a username whitelisting mechanism to prevent the use of inappropriate or offensive usernames.

3.1.1.3.11 Username Whitelisting Validation *(O: Morgan)*

Upon account creation or username change, the system must validate proposed usernames against a predefined whitelist of prohibited terms, symbols, or patterns.

3.1.1.3.12 Prompt for Non-Compliant Username *(O: Morgan)*

If a proposed username matches any entry on the whitelist, the system must prompt the user to select a different username complying with whitelisting criteria.

3.1.1.3.13 Regular Whitelist Updates *(O: Morgan)*

The whitelist must undergo regular updates to include new inappropriate terms and patterns identified over time to maintain effectiveness.

3.1.1.3.14 Confirmation Email Dispatch *(O: Nabi)*

Upon successful registration, the system must automatically dispatch a confirmation email to the user's provided email address to verify its validity.

3.1.1.3.15 Confirmation Email Content *(O: Nabi)*

The confirmation email must include a verification link or six-digit code for users to click or enter to activate their account.

3.1.1.3.16 Confirmation Email Status Tracking *(O: Nabi)*

The system's database must track the status of confirmation emails and provide appropriate error handling for delivery failures or bounced emails.

3.1.2 Login

3.1.2.1 Introduction/Purpose of Feature

The login feature facilitates secure access for both students and teachers on the ReasonED platform, enabling entry to personalized accounts and platform features within a protected environment.

3.1.2.2 Stimulus/Response

Stimulus: Users initiate the login process by providing their credentials (email and password) on the dedicated login page.

Response: Upon successful validation of the provided credentials, users are directed to a Two-Factor Authentication (2FA) page.

Stimulus: Upon reaching the 2FA page, user receive a notification via email containing a randomly generated six-digit code.

Response: Users must input this code to complete the authentication process.

Stimulus: Additionally, a password change option is available on the login page.

Response: Selecting this option prompts users to enter their registered email address.

Stimulus: Upon clicking the password change link in the email.

Response: Users are directed to a password change page.

3.1.2.3 Associated Functional Requirements

3.1.2.3.1 Login Form *(O: Morgan)*

The system must feature a login page where users can input their email and password.

3.1.2.3.2 Two-Factor Authentication (2FA) Initiation *(O: Morgan)*

Upon successful authentication of credentials, users must be directed to the Two-Factor Authentication (2FA) page.

3.1.2.3.3 2FA Code Dispatch *(O: Morgan)*

Upon successful authentication of credentials, users must be directed to the Two-Factor Authentication (2FA) page.

3.1.2.3.4 2FA Code Input Prompt *(O: Morgan)*

Users must be prompted to input the received code to finalize the login process.

3.1.2.3.5 Password Change Option *(O: Morgan)*

The login page must include an option for users to change their password.

3.1.2.3.6 Password Change Initiation *(O: Morgan)*

When selecting the password change option, users must be prompted to enter their registered email address.

3.1.2.3.7 Password Change Email Dispatch *(O: Morgan)*

The system must send an email containing a link for password change to the user's registered email address.

3.1.2.3.8 Password Change Page Redirection *(O: Morgan)*

Clicking on the provided link must redirect users to a password change page where they can update their password.

3.1.2.3.9 Student Profile Access *(O: Morgan)*

Upon successful login, students must be granted access to their personalized profiles, including at a minimum games, leaderboards, and collectibles.

3.1.2.3.10 Teacher Resource Access *(O: Morgan)*

Teachers, upon login, must have access to additional resources tailored for instructional purposes and tools to monitor student progress.

3.1.3 Straw Manny: Gameplay Mechanics

3.1.3.1 Introduction/Purpose

This section describes the various gameplay mechanics within the "Straw Manny"game. In this game, players play as Manny and must build their combat skills by attacking real knights rather than straw men, teaching the concept of the "straw man" fallacy. Players will navigate through training fields as Manny, where they can run into knights and find combat abilities and power ups. When players run into a knight, the knight will present a claim and three response choices to that claim that the player must choose from. Two of the choices will be straw man responses, one

choice will not. Players must identify and select non-strawman responses to progress in the game. If the correct one is chosen, players can attack the knight and use any abilities that they have collected. If the player chooses a strawman response, Manny attacks a straw man instead and loses a life.

3.1.3.2 Stimulus/Response

Stimulus: Player collides with a knight.

Response: Scene changes to knight encounter (challenge scene).

Stimulus: Player clicks a straw man (incorrect) response

Response: Animation shows Manny attacking a straw man. A "life" is visibly deducted from the player. A dialogue box explains why the chosen response is a strawman fallacy.

Stimulus: Player clicks the correct response

Response: Player is rewarded with points for choosing the correct response. A combat scene is activated where the player can attack the knight and utilize any abilities they have obtained.

Stimulus: Player collides with a combat ability on the training field.

Response: The ability is added to the player's inventory.

Stimulus: Player collides with a power-up on the training field.

Response: The powerup's effects are instantly applied to the player.

3.1.3.3 Associated Functional Requirements

3.1.3.3.1 Tutorial Scene *(O: Cieslinski)*

The system must display a tutorial overlay that describes the objective of the game and the mechanics used to achieve this objective.

3.1.3.3.2 Tip Overlays *(O: Cieslinski)*

The application must:

- Display an overlay when the player first collides with an enemy knight that explains that the knight will make a claim that has to be contested.
- Display an overlay when the player first makes a correct response that explains that the player is in combat and can use abilities they have accumulated.
- Display an overlay when the player first makes an incorrect response that explains the player has chosen incorrectly and has lost a life for fighting a strawman.

3.1.3.3.3 Knight Spawning *(O: Louk)*

The application must generate knights at random locations only within the grass tiles of the training fields for players to find and run into.

3.1.3.3.4 Scoring (O: Louk)

The application must calculate the player's score based on their performance, rewarding 500 points for correct responses and 50 points for each use of a combat ability.

3.1.3.3.5 Adaptive Learning Algorithm *(O: Louk)*

The application must adjust the difficulty of each challenge according to the player's performance in the previous challenge and its difficulty rating.

3.1.3.3.6 Collectibles *(O. Dawe)*

The application must allow players to discover and collect combat abilities and power-ups dispersed throughout the training field scene. Combat abilities will be stored in the player's inventory for later use. Power-ups will be used immediately upon pickup.

3.1.3.3.7 Feedback *(O: Louk)*

The application must provide feedback specific to an incorrect response via the dialogue box when players choose an incorrect response.

3.1.3.3.8 Challenge Scene *(O: Louk)*

The application must switch to a challenge scene when the player collides with a knight in the training field scene. The challenge scene will initially display the knight character, a dialogue box displaying a claim, and a continue button. When the player clicks the button, the response choices for the claim will be shown.

3.1.3.3.9 Combat Scene *(O: Louk)*

The application must switch to a combat scene after the player clicks the correct response choice and the score is updated. The combat scene will display the knight character, the knight's health, Manny, and the combat abilities the player has obtained but not used. The Player can click the knight to perform a basic attack or click combat abilities to deal more damage.

3.1.4 Straw Manny: Player Controls

3.1.4.1 Introduction/Purpose

This section outlines the requirements for how the Straw Manny game responds to player input.

3.1.4.2 Stimulus/Response

Stimulus: The player presses the directional keys during gameplay.

Response: The player sprite moves in the direction indicated, with diagonal movement normalized.

Stimulus: The player presses the interaction key during a dialogue sequence.

Response: Depending on the state of the sequence, the dialogue box will either be filled with all the remaining text, move on to the next line of dialogue, or end the sequence.

Stimulus: The player presses an ability key during combat.

Response: The system executes the ability, potentially dealing damage to the enemy.

3.1.4.3 Associated Functional Requirements

3.1.4.3.1 Input during Dialogue Sequence *(O: Cieslinski)*

The system must:

- Prevent all game entities from moving when a dialogue sequence is on-screen.
- Allow dialogue that is still in-progress to automatically reach the end of the text on player input.
- Advance to the next line of dialogue on player input if all text in the current line has been displayed and there are more lines.
- End the dialogue sequence and return to the previous game state on player input if all text in the current line has been displayed and there are no more lines.

3.1.5 Straw Manny: Level Design

3.1.5.1 Introduction/Purpose

This section outlines the requirements for the level design of the Straw Manny prototype, such as object placements and item requirements. The purpose is to create unique but increasingly challenging environments that offer players a sense of progression and accomplishment.

3.1.5.2 Stimulus/Response

Stimulus: The player selects a level from the level selection screen.

Response: Load the selected level and initialize the player character.

3.1.5.3 Associated Functional Requirements

3.1.5.3.1 Spawn Location *(O: Marchione)*

Each level will have a static location to initialize the player character.

3.1.5.3.2 Opponent Locations *(O: Marchione)*

Each level will feature three opponents for the player to challenge. These opponent objects will randomly select an empty but accessible ground tile to spawn the opponent at. The opponent locations are selected when the level is loaded.

3.1.5.3.3 Level Requirements *(O: Marchione)*

The prototype will feature three levels each with their own unique layouts and three randomly located opponents.

3.1.6 Straw Manny: Graphics & User Interface

3.1.6.1 Introduction/Purpose

This section outlines the requirements for the on-screen elements of the user interface, including static scenes and interactive buttons and progress displays.

3.1.6.2 Stimulus/Response

Stimulus: Upon Launching the “Straw Manny” game, the user is presented with the main menu.

Response: The main game logo prominently at the top-center of the screen, accompanied by the image of the main character. Below the logo and the character image are three clearly labeled buttons.

3.1.6.3 Associated Functional Requirements

3.1.6.3.1 Dialogue Textbox *(O: Cieslinski)*

The system must:

- Add letters from a given dialogue string to the textbox on a timer to allow for text scrolling.
- Wrap text if it exceeds the width of the textbox.

- Display an animated indicator within the textbox when text scrolling is complete.

3.1.6.3.2 Character Dialogue Overlay *(O: Cieslinski)*

The system must:

- Display one or two high-resolution character images along with a dialogue textbox.
- Display the name of the character that is talking.
- Darken the image of the character that is not speaking if there are two characters in the dialogue.

3.1.6.3.3 Map Overlay *(O: Troyer)*

On the map screen, the system must:

- Display a numeric indicator of the current level.
- Display a numeric indicator of the total score.
- Display a strength bar to indicate the number of enemies defeated in the current level.
- Display a health bar indicating lives remaining.
- Display the inventory of abilities that can be activated.
- Display a button to mute the audio.

3.1.6.3.4 Main menu *(O: Troyer)*

When Straw Manny is loaded, the system must:

- Display the logo and main character.
- Display a button to start the tutorial.
- Display a button to start the main game.
- Display a button to mute the audio.

3.1.7 Straw Manny: Audio Design

3.1.7.1 Introduction/Purpose

This section outlines the requirements for the audio design in Straw Manny, including music, sound effects, and any other auditory elements.

3.1.7.2 Stimulus/Response

Stimulus: Players enter a new level.

Response: The background music changes to match the theme of the level, creating an immersive experience for the player.

3.1.7.3 Associated Functional Requirements

3.1.7.3.1 Volume Control *(O: Morgan)*

The system must include adjustable volume controls for both music and sound effects.

3.1.7.3.2 Dynamic Soundtrack Integration *(O: Morgan)*

The system must dynamically adjust music transitions based on in-game events and player actions.

3.1.7.3.3 Environmental Sound Effects *(O: Morgan)*

The system must incorporate environmental sound effects that reflect the player's surroundings, including at a minimum, footsteps on different surfaces, ambient wildlife noises, and weather effects.

3.1.7.3.4 Main Menu Soundtrack *(O: Morgan)*

The system must:

- Support the integration of a single soundtrack tailored for the main menu.
- Ensure that the main menu soundtrack loops without interruption.

- Allow players to adjust the volume levels of the main menu soundtrack to optimize their audio experience.

3.1.7.3.5 Training Ground Music Soundtrack *(O: Morgan)*

The system must:

- Support the integration of a single soundtrack tailored for the training ground.
- Enable the soundtrack to adapt dynamically to different areas within the training ground, reflecting changes in environment or player interactions.
- Ensure smooth and unobtrusive transition between different sections of the training ground.

3.1.7.3.6 Challenge Scene Music Soundtrack *(O: Morgan)*

The system must:

- Support the integration of a single soundtrack tailored for the challenge scenes.
- Ensure that the selected tracks align thematically with each challenge scene, enhancing immersion and coherence within the game world.
- Allow players to adjust the intensity and volume of challenge scene music according to their preferences without causing disruptions to gameplay.

3.1.7.3.7 Dialogue Sound Effects *(O: Cieslinski)*

The system must play a small beep each frame while text scrolling is happening, where the exact sound of each beep depends on the character speaking.

3.1.8 Hasty Harry: Gameplay Mechanics

3.1.8.1 Introduction/Purpose

This section outlines the core gameplay mechanics for Hasty Harry that will define the core game loop. The goal of Hasty Harry is to have the player find artifacts on a planet. These

artifacts help descript what things live on the planet. For example, if Harry finds an artifact with teeth that will indicate a carnivore lives there. Harry will deposit artifacts into his ship. Once the ship has enough artifacts the game will move onto the second portion which is the report. Harry will analyze the artifacts and sometimes write quick and inaccurate assumptions. It is up to the player to identify those inaccurate assumptions and correct them.

3.1.8.2 Stimulus/Response

Stimulus: Allow players to adjust their characters hair color, skin color, and spacesuit color in the main menu.

Response: The players character will be saved to their account and accessible anytime they play.

3.1.8.3 Associated Functional Requirements

3.1.8.3.1 Random Artifact Spawning *(O: Dawe)*

Artifacts will be spawned in random locations around the world.

3.1.8.3.2 Artifact Placement Limitation *(O: Dawe)*

No artifact should spawn closer than 20m from each other.

3.1.8.3.3 Inventory System *(O: Dawe)*

Harry will have an inventory that will contain pickups (artifacts) found in the world.

3.1.8.3.4 Artifact Collection *(O: Dawe)*

Harry will pick up artifacts by walking over them.

3.1.8.3.5 Artifact Transfer to Spaceship *(O: Dawe)*

When Harry enters the spaceship, any artifacts held on the character will be transferred to the spaceship.

3.1.8.3.6 Artifact Evaluation *(O: Dawe)*

When artifacts are transferred to the spaceship, the spaceship will evaluate to see if Harry brought enough artifacts to move onto the next section of the game.

3.1.9 Hasty Harry: Level Design

3.1.9.1 Introduction/Purpose

This section outlines the requirements for the level design of the Hasty Harry game, such as level transitions, item requirements and object placements.

3.1.9.2 Stimulus/Response

Stimulus: The player selects the level from the level selection screen.

Response: The selected level will load, initializing the player character.

3.1.9.3 Associated Functional Requirements

3.1.9.3.1 Spaceship Location *(O: Marchione)*

Each level requires a static location for the player's ship to land. This is used for level transitions and gameplay features.

3.1.9.3.2 Artifact Locations *(O: Marchione)*

Each level requires six artifact objects to be placed in random locations around their environment. The artifact objects will randomly select empty but accessible ground tiles to spawn at. These locations are selected when the level is loaded.

3.1.9.3.3 Level Requirements *(O: Marchione)*

The prototype will feature three levels with unique level layouts and six randomly located artifact objects. Each level will have a unique theme associated with it.

3.1.10 Educator Information & Resources

3.1.10.1 Introduction/Purpose

Information and resources for educators will be included on the website. This is to be transparent about the educational intentions of ReasonED and to provide additional critical reasoning materials for educators to use in their classrooms. The resources page will include a grid of downloadable educational materials such as posters, lesson plans, activity sheets, etc.

3.1.10.2 Stimulus/Response

Stimulus: The user clicks the "Teacher" link on the navigation bar

Response: The user is redirected to the general teacher information page.

Stimulus: The user clicks the "Resources" link on the Teacher dropdown menu in the navigation bar

Response: The user is redirected to the teacher resources page.

3.1.10.3 Associated Functional Requirements

3.1.10.3.1 Sorting and Filtering of Resources *(O: Louk)*

Every resource included on the Resources page must include two tags specifying the category it falls under and the grade level it is suited for. The categories are:

- Decorations
- Activities
- Lesson Plans
- Powerpoint presentations
- Infographics

3.1.10.3.2 Sorting and Filtering of Resources cont. *(O: Louk)*

Users must be able to filter resources by category and grade level.

3.2 Performance Requirements

3.2.1 Speed of Response

3.2.1.1 Homepage Loading Time *(O: Morgan)*

The average response time for loading the student and teacher homepage shall not exceed two seconds.

3.2.1.2 Account Creation Processing *(O: Morgan)*

The account creation process must be completed within five seconds from the user's submission of registration details to receive confirmation.

3.2.1.3 Game Level Loading *(O: Morgan)*

Game levels must load within three seconds to maintain engagement and prevent user frustration.

3.2.2 Throughout

3.2.2.1 Simultaneous User Sessions *(O: Morgan)*

The system must support a minimum of 500 simultaneous user sessions during peak hours without significant degradation in performance.

3.2.2.2 Account Registration Processing *(O: Morgan)*

The system must be capable of processing a minimum of 100 account registrations per minute to efficiently handle high-traffic periods.

3.2.3 Execution Time

3.2.3.1 Gameplay Interaction Execution *(O: Morgan)*

Game interactions and responses to user inputs must execute within 50 milliseconds to ensure smooth and responsive gameplay.

3.2.3.2 Database Query Execution *(O: Morgan)*

Database queries for user data retrieval must execute within 20 milliseconds to minimize latency in accessing personalized content.

3.2.4 Storage Capacity

3.2.4.1 Database Storage Capacity *(O: Morgan)*

The system's database must have a minimum storage capacity of 100 terabytes to accommodate user account data, gameplay logs, and other platform-related information.

3.2.4.2 File Upload Size *(O: Morgan)*

File uploads, such as user profile pictures or game assets, must support files up to 10 megabytes in size without impacting system performance.

3.3 Design Constraints

3.3.1 Compatibility Requirement *(O: Morgan)*

The system must be compatible with all major web browsers, such as Google, Chrome, Mozilla, Firefox, Microsoft Edge, and Safari.

3.3.2 Scalability Requirement *(O: Morgan)*

The system architecture must be designed to scale horizontally to accommodate increasing user loads without compromising performance or interrupting service during periods of high demand.

3.4 Software System Attributes

3.4.1 Reliability *(O: Morgan)*

The system must have a mean time between failures (MTBF) of at least 10,000 hours to minimize downtime and maximize operation.

3.4.2 Availability *(O: Morgan)*

The system must maintain an uptime of 99.99% over any given month to ensure that it is available to users whenever they require access.

3.4.3 Security *(O: Morgan)*

All user data must be encrypted using AES-256 encryption to prevent unauthorized access and ensure data confidentiality.

3.4.4 Maintainability *(O: Morgan)*

The codebase must adhere to industry-standard coding conventions and be well-documented to facilitate ease of maintenance and future enhancements.

3.4.5 User Portability *(O: Morgan)*

The software interface must be responsive and adaptable to various screen sizes and resolutions to provide a consistent user experience across desktop and mobile devices.

3.5 Other Requirements

3.5.1 Regulatory Compliance *(O: Morgan)*

The system must comply with relevant data protection regulations, such as HIPPA, to safeguard user privacy.

3.5.2 Copyright Compliance *(O: Morgan)*

All content, including images, audio files, and text, utilized within the software must possess proper licensing or be owned by the development team.

3.5.3 User Feedback *(O: Morgan)*

The system must provide a user feedback mechanism, such as a feedback form or survey, to gather user opinions and suggestions for improvement.