1. **Project Description**
2. **Research and Design**
3. **Requirements Specifications**
   1. Use case diagrams
   2. Activity/ Sequence Diagrams
   3. Class Diagrams
4. **Description of adopted approach**
   1. SDLC
   2. MosCoW analysis
   3. Analytical techniques used
   4. Proposed design method
   5. Object Oriented tech and C++ libraries
   6. Back up claims with sources
5. **Detailed explanation of data structure**
   1. Justify choices
6. **Algorithm**
   1. Pick one
   2. Say why
   3. Explain how it works
   4. Detail steps
7. **Comment on:**
   1. Software implementation
   2. Parameters
   3. Testing process and metrics (black box)
8. **Discuss Results**
   1. Reflect on testing approach
   2. Reflect on performance such as computational efficiency, reliability, security, portability, maintainability, scalability
   3. Analysis of system performance big-O notation
9. **Draw Conclusions**
   1. Reflect on the adopted method (and alternatives)
   2. Reflection on development and SWOT analysis
   3. Reflect on possible improvement
10. **Appendix**
    1. *Flow charts*
    2. *Design diagrams*
    3. *Printscreens of set-up windows*
    4. *images/tables of test cases*
    5. *Test images*
    6. *Results (tables, output images, performance histograms)*

**System Proposal**

The project is a match 3 puzzle game in which the player can progress through a series of battles against enemies. The enemies will be defeated when the player takes their health to 0. This can be achieved by matching symbols to cause damage or collecting enough of each resource to allow the player to use spells. The spells will have a different effect on the battle. As a reward for winning each battle the player will be rewarded with either currency, experience, or both. The currency will allow the player to purchase items in game to use during battles. The experience gained allows the player to level up, giving an increase in stats, which will make future battles easier and improve the effect the player's actions have. All progress made can be saved at any point and will allow the player to resume their game at any time.

**Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **REQUIREMENT** | **GOALS & ELABORATION** | **IMPLICATION** |
| **FR1:** Character Creation | The user must be able to create a character, specifying name, gender, and choosing from several preset images. | The character will be able to be given a name, and a range of avatars will be able to be selected by the user. These will need to be stored upon selection, and loaded whenever the user's character is displayed on screen |
| **FR2:** Save and Exit Application | The user must be able to save their progress in the game before exiting the application if they wish to. | The current information about the game must be implemented in a way that allows it to be stored. |
| **FR3:** Display Help Screen | The user must be shown the help screen on the first time playing the game. They must also be able to access the screen at any time if they need help. | Upon first starting the game, a splash screen showing the general idea of how to play the game will be displayed. This screen can be accessed at any point by the player by selecting the relevant option. |
| **FR4:** Load Existing Save | The user must be able to load past saves. The past saves should include the information generated during character creation, items, available currency, and level, and progress through the game. | The information stored when a game is saved (see **FR2**) must be read allowing the player to resume the game from where they last saved. |
| **FR5:** Match Three | The user must be able to select two tiles from the game area and swap them. This will give the user the resource that corresponds to the tiles matched | There will be an algorithm to detect when 3 identical/matchable tiles are adjacent to each other. These tiles will then be cleared and the appropriate effect will take place depending on the type of tile. The rest of the board will re-adjust to occupy the space of the cleared tiles, and new tiles will be generated and enter the game board at the top. |
| **FR6:** Match Four | The user must be able to match four tiles to create a special tile which will have greater effect when matched with other tiles. | The algorithm used to detect matches must be able to detect when four of the corresponding symbol are matched in order to create a new tile which will be added to the game screen. |
| **FR7:** Gain Experience | The user must be able to gain experience from matching certain tiles and defeating enemies. | Upon matching 3 or 4 of the experience symbol (see **FR5** and **FR6**) the player must receive experience equal to the amount of tiles matched. Experience must also be awarded upon victory (see **FR16**). |
| **FR8:** Collect Currency | The user must be able to gain currency from matching certain tiles, defeating enemies, and selling items in the shopping screen. | The system must be able to keep track of the user's total currency incrementing it when the user matches 3 or 4 corresponding symbols (see **FR5** and **FR6**) as well as upon completion of a battle or when an item is sold in the shop (see **FR12**). |
| **FR9:** Use “Spells” | The user must be able to spend “mana” in order to use “spells” which have an affect on the game. | The system will need to track the amount of each resourced gained when the corresponding symbol is matched (see **FR5** and **FR6**). This will then compare against the costs to use each spell and remove the amount of resources ‘spent’ from the player's total. |
| **FR10:** Battle Against AI | The user must be able to compete against AI which perform the same actions as the user in order to deal fatal damage to the user. | The match 3 system will be played in a way which pits the user against an AI. Both players will have health bars, and the first to deplete their opponent's health bars will be the winner. |
| **FR11:** Receive Items | The user must be able to receive items upon defeating an enemy or purchasing it at the shop. | Upon defeating an enemy (see FR 10 and FR 16), the player will have a chance to receive a specific item. This chance will vary depending on the type of enemy. The player will have a limited number of item slots, and therefore must choose whether or not they wish to take the item |
| **FR12:** Purchase/ Sell Items | The user must be able to purchase items (see **FR11**) in exchange for currency. The user must also be able to exchange items for currency. | Using the currency earned throughout the game, the user should be able to purchase items which will help them throughout the rest of the game. These items can also be sold for a percentage of the currency used to purchase them |
| **FR13:** In-Game UI | The user select all options from the game using the cursor. All information must be displayed on the screen and be available to the user either at all times, or on request. | The UI will have to support mouse input as well as keyboard inputs. These will have to be clearly told to the player during the help screen. |
| **FR14:** Progress Between Levels | The user is able to advance from one level to another after successfully completing the previous stage. | The system must be able to detect when each level has ended (see **FR16**) and provide the player with a choice between progressing to the next level or going to the shop. |
| **FR15:** Randomised Enemy Selection | The enemy the user faces in each stage will be selected from a pool of enemies at random to ensure slightly different gameplay experiences each time the game is played. | The system will need a store of enemies which includes stats and an image to display. These must then be scaled to the users level to provide appropriate challenge. |
| **FR16:** Defeat Enemy | If the enemy that the player is battling reaches 0 health the the game screen will end and the player will win. | The system must be able to detect when an enemy is defeated. This will then give the player experience based on the defeated enemy as well as an item from a predefined structure of items. |
| **FR17:** Main Menu | When the game is launched, the user will be taken to a main menu where they can start the game, change options, or exit the game | The menu must contain options to start a new playthrough of a game, load an existing save, or exit the game. |
| **FR18:** Player Defeated | When the player’s health reaches 0 the game must inform the player that they have been defeated and return them to the main menu. | The system must be able to detect when the player is defeated. It will then display their overall ‘score’ and return them to main menu screen. |

**2) Research And Design**

**9) Appendix**