Software Requirements Specification (SRS)

Dungeon Quest

Team: Group 7

Authors: Greg Kaplowitz, Nicholas Miceli, Derrick Lor, Wyatt LaRose, Alberto Chavez

Customer: 6th Graders

Instructor: Dr. James Daly

**1 Introduction**

This is a software requirements specification document (SRS) outlining the necessary details about our proposed educational game.

In this document, we will have detailed subsections that will cover the following: 1.1 Purpose, 1.2 Scope, 1.3 Abbreviations, 1.4 Organization

2 Overall Description, 2.1 Product Perspective, 2.2 Product Functions, 2.3 User Characteristics, 2.4 Constraints, 2.5 Dependencies, 2.6 Apportioning Requirements

3 Specific Requirements

4 Modeling Requirements, 4.1 Use-Case Diagram, 4.2 Class Diagram, 4.3 Sequence Diagram, 4.4 State Diagram

5 Prototype, 6 References, 7 Contact.

## Purpose

The purpose of this document is to establish a basic requirement list between the clients, users, stakeholders, and developers of our proposed educational game. All participants involved in this software project must agree to this document, and hereby follow the established outline to the creation, documentation, and distribution of the proposed educational game.

## Scope

The software product being developed is titled “Dungeon Quest”. Dungeon Quest is aimed at the target audience of 4th to 6th grade students. It is an educational game meant to be a learning aid, not replacement, of core curriculum taught at public schools regulated by the Massachusetts Department of Elementary and Secondary Education.

The benefits of Dungeon Quest include the ability to have students be asked a wide or narrow variety of questions depending on the desires of the educator. Players will have the option to decide which core subjects they would like the questions to be drawn from.

The objective of Dungeon Quest is to provide a simple yet engaging experience for students to be able to reinforce the topics that are covered in their classroom. This includes reasonably paced gameplay for students in the 6th grade age range as well as questions drawn from the core curriculum provided by the Massachusetts Department of Elementary and Secondary Education.

The application domain for Dungeon Quest is that of an educational video game.

Dungeon quest will include game sessions that can reasonably be completed within a standard class mns block. Game sessions are meant to be played during the class and progress is not to be carried over throughout multiple class sessions. The game will give players the option to select what subject they would like to answer questions for and will provide more difficult questions the deeper the player is into their current session. Dungeon quest will keep track of player’s score throughout the session and display it upon completion (game over or game won) so users can aim to better their scores during future sessions.

Dungeon Quest does not intend to replace core curriculum teaching techniques but rather supplement them in order to reinforce the topics covered in a traditional class setting.

## Definitions, acronyms, and abbreviations

Definition list of all the terms, acronyms, and abbreviations in this document.

Educational game to be created is Dungeon Quest.

\*Map = the selection screen where players choose which dungeon to play.

\*Dungeon = the level and the specific genre covered in that level.

\*Path = the direction the player can move to.

\*Encounter = the battle between players and enemies.

\*User = the person interacting with the software

\*Enemy = the entity that the user is fighting against

\*SRS = Software requirement specification

\*UI = User Interface

\*SW = Software

\*HW = Hardware

\*MDoE = Massachusetts Department of Education

\*OS = Operating System

\*CPU = Central Processing Unit

\*RAM = Random Access Memory

\*GPU = Graphics Processing Unit

## Organization

Going forward, the rest of the SRS will cover topics pertaining to the edutainment game’s interface design, functionality, user expectations, possible constraints, requirements, models and diagrams, and then finally points of contact.

Section 2 -

2.0 - Description of Section 2

2.1 - Product Perspective: The context for the product

2.2 - Product Functions: The functions that the software will perform

2.3 - User Characteristics: Expectations about the user

2.4 - Constraints on the development of the game

2.5 - Assumptions and Dependencies

2.6 - Apportioning of Requirements

Section 3 -

3.0 - List of requirements for the software

Section 4 -

4.0 - Modeling requirements

a. Use Case Diagram

b. Class Diagram/Data Dictionary

c. Sequence Diagrams

d. State Diagram

Section 5 -

5.0 - Description of what functionality the prototype will display

5.1 - How to run the prototype

5.2 - Sample Scenario

Section 6 -

6.0 List of references

Section 7-

7.0 Contact information

# Overall Description

This section will cover the context of the product, the major functions it will perform, the characteristics of the expected users, constraints on the software’s design, assumptions during production, and apportioning of requirements.

* 1. Product Perspective

Dungeon Quest is designed to provide a fun environment where students can test their knowledge while playing an interactive game. Research has been done to support the fact that “using games in teaching can help increase student participation, foster social and emotional learning, and motivate students to take risks” [2] and Dungeon Quest will be no exception. By including questions based on 6th grade core curriculum set by the Massachusetts Department of Education and gameplay that incorporates fantasy elements kids are known to love, Dungeon Quest is the perfect game for students to improve their skills.

The interface needs to be kept simple so that children will have an easy time interacting with it. Dungeon Quest’s creation is constrained by Unity as that is the platform that has been chosen to develop with, but it has many different options that can be used to achieve this. There is data that should be present on different screens so a certain UI element will persist between scenes. Unity teams is being used to facilitate production which supplies 25GB of cloud storage space to develop collaboratively which constrains possible memory; however, Dungeon Quest will fit within these bounds. Finally, the current build of Dungeon Quest is being produced to run on Windows, however future iterations will include support for other platforms.

Graphical user interface, application, website

Description automatically generated

* 1. Product Functions

The player will start Dungeon Quest and choose the difficulty level they would like to experience. From there, they will be given a map that consists of branching paths. Starting from the root of this tree-like structure they will pick one of the child paths. Choosing a path will take the user into the battle screen of the game where the educational aspect takes place.

During each battle, the user will be asked questions relating to the 6th grade core curriculum (MDoE). By answering correctly, the user’s character will attack the enemy and cause them to lose health. On the other hand, if the user answers incorrectly, the enemy will attack the user’s character causing them to lose one life. The battle continues until either the user’s character or the boss is out of health. In the former case, game over will be displayed to indicate that the user has lost, in the latter case the user will be returned to the map to select their next level or if they just beat the final level the game won screen will be displayed indicating that the user has completed a successful playthrough.

User’s scores will be tracked throughout a gameplay session and will increase for each question the user answers correctly (more difficult questions will award more score per question). Once the user reaches either the game over or game won screen, the score will be displayed to them.

* 1. User Characteristics

The expected grade level of the users of Dungeon Quest are 6th graders (Approximately 11 years old). The users are expected to have enough foundational knowledge through typical teaching styles to have the ability to answer the questions asked of them in Dungeon Quest. The user ideally should have enough experience with multiple choice style questions to rule out incorrect answers and arrive at a reasonable answer to the questions provided. Users will be provided with different difficulty level options to suit 6th grade students at all levels of educational advancement.

Users are expected to have enough experience with computers to be able to play the game and will require the motor functions needed in order to interact with Dungeon Quest’s interface (through mouse or trackpad).

Finally, users should have the decision-making abilities necessary to navigate Dungeon Quest’s maps to seek out the end of the dungeon.

* 1. Constraints

One constraint on the project is the memory allotted. Unity teams provide 25 GB of cloud storage to be used for Dungeon Quest which constrains how large the game can be. That said, Dungeon Quest will fit within these bounds. Also, only 5 members are allowed on a Unity team which constrains how many members can work on Dungeon Quest.

Another constraint is the hardware that the game will run on. The current build of Dungeon Quest is intended to be run on a Windows machine. Future iterations may provide builds for other platforms.

Finally, Dungeon Quest’s production is constrained by time. Given that only a couple of months are given to complete the project, there are certain features that will need to be trimmed to meet deadlines.

* 1. Assumptions and Dependencies

Once assumption being made is that the computer running Dungeon Quest will have a mouse or trackpad to interact with the game. Also, this computer will have Windows operating system to run the game. The device must have the hardware specifications detailed in section 5.1 of this SRS document to successfully run the game.

The production of Dungeon Quest is dependent on Unity’s game development environment. Unity was selected because of its collaborative abilities using Unity teams as well as it being free to use.

Users of the game will not need to have access to Unity, as the game will be packaged as an executable.

* 1. Apportioning of Requirements

Our biggest requirement that we probably won’t be able to do because of time is implementing a powerup system to alter how combat works. Currently our combat system is just answering questions, we might have both multiple choice and short answer but currently the requirement is for multiple choice. In future iterations of this game, we could make combat more interesting for users so that it isn’t simply based on answering questions.

Another requirement that is beyond the scope of the current iteration of our project is the ability for teachers to input their own questions for their students to be asked when playing through Dungeon Quest. This functionality would allow educators to fine tune the game to ensure that the questions being asked fit the scope of their lesson plan and would allow teachers of other grade levels (other than 6th) to utilize Dungeon Quest in their classrooms as well.

# Specific Requirements

1. There shall be a main menu from which the game: “Dungeon Quest”, will begin when the user launches it
2. The menu will be dungeon themed with the name of the game as well as the buttons to navigate to other screens labeled “start game” and “about”
3. There will also be a “Quit” button that will exit the game
4. The “start game” button will display different options for different game modes
5. The user must then select which difficulty they would like to play (easy, medium, hard)
6. After the user selects their difficulty, the game will begin.
7. The game will begin with the user’s character in the dungeon selection screen with 5 lives (maximum).
8. The user will choose which path in the dungeon to take
9. Paths will determine the player’s route through the dungeon and what they encounter
10. Paths can have encounters that have different types of questions (topic, difficulty, etc.)
11. Once a path has been selected, the user will enter an encounter with an enemy
12. Within the encounter the user will have a running life total
13. The user will be asked questions in order to activate their attacks if answered correctly
14. The monster will attack which prompts the player with a question, if the player gets it correct, they dodge, otherwise they are hit
15. Once enough questions have been answered, the enemy will be defeated, and the encounter will end.
16. The number of questions needed to defeat an enemy will depend on how far the user is into the dungeon, and the difficulty selected.
17. Once the encounter ends, the user will be returned to the dungeon screen to select their next path.
18. If the user runs out of lives at any point, they will enter a final stand question
19. If the user answer this correctly they will be able to continue the encounter, if they answer incorrectly, they will be sent to a “game over” screen.
20. If the user reaches the end of the dungeon without running out of lives, a “Congratulations” screen will be displayed along with the user’s score
21. The score will be calculated based on how many questions the user gets right, it will be higher the fewer questions answered incorrectly
22. The “about” button will link the user to the website for the project
23. A bank of questions for each subject level
24. Questions should have a range of difficulties
25. Questions can be partitioned to different types of enemies
26. Questions are for 6th graders

# Modeling Requirements

Use Case Diagram

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Use Case Name: | View About Page |
| Actors: | Player |
| Description: | The Player clicks on the “About” button on the main menu and is directed to the project’s website |
| Type: | Primary |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 4 |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Start Game Session |
| Actors: | Player |
| Description: | The Player clicks on the “Start” button on the main menu and begins a session of the game |
| Type: | Primary and Essential |
| Includes: | Select Difficulty/Dungeon |
| Extends: | None |
| Cross-refs: | Requirement 2 |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Exit Game |
| Actors: | Player |
| Description: | The Player clicks on the “Quit” button on the main menu and the game is exited |
| Type: | Primary |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 1b |
| Uses cases: | N/A |

|  |  |
| --- | --- |
| Use Case Name: | Select Difficulty/Dungeon |
| Actors: | Player |
| Description: | The Player selects which difficulty of dungeon they would like to play |
| Type: | Primary and Essential |
| Includes: | Encounter Enemy/Answer Question |
| Extends: | None |
| Cross-refs: | Requirement 2a |
| Uses cases: | Player must first click the start button to be able to select their difficulty |

|  |  |
| --- | --- |
| Use Case Name: | Encounter Enemy/Answer Question |
| Actors: | Player |
| Description: | The Player battles an enemy by answering questions |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | None |
| Cross-refs: | Requirement 3b |
| Uses cases: | The player must have started a run of the game and selected their difficulty and a level to play |

|  |  |
| --- | --- |
| Use Case Name: | Answer Correctly |
| Actors: | Player |
| Description: | The Player chooses the correct answer to a question asked in an enemy encounter |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | Encounter Enemy/Answer Question |
| Cross-refs: | Requirement 3bii |
| Uses cases: | The player must have started a run of the game and selected their difficulty and a level to play |

|  |  |
| --- | --- |
| Use Case Name: | Answer incorrectly |
| Actors: | Player |
| Description: | The Player chooses the incorrect answer to a question asked in an enemy encounter |
| Type: | Primary and Essential |
| Includes: | None |
| Extends: | Encounter Enemy/Answer Question |
| Cross-refs: | Requirement 3biii |
| Uses cases: | The player must have started a run of the game and selected their difficulty and a level to play |

Class Diagram:

Graphical user interface, application, Teams

Description automatically generated

Data Dictionary:

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| MainMenu | | This class represents the main title screen that will be displayed upon launch of the game |
| Attributes | background : Image | The image that will be displayed behind the text and buttons on the Main Menu |
|  | title : String | The title of the game “Dungeon Quest” that will be displayed on the Main Menu |
|  | startGame : Button | A button on the Main Menu that will begin the actual gameplay |
|  | about : Button | A button on the Main Menu that will open the website for our project |
|  | quit : Button | A button on the Main Menu that will exit the game altogether |
| Operations | startGameClicked() : void | Called when the StartGame button is clicked, begins the gameplay |
|  | aboutClicked() : void | Called when the About button is clicked, displays project’s website |
|  | quitClicked() : void | Called when the Quit button is clicked, exits the game |
| Relationships | The main menu begins an instance of “Player” when the game is started. It is also displayed when a player quits during/after their run of the game. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Player | | This class represents the player in the game and contains attributes that would carry with them throughout the encounters. |
| Attributes | numLives : int | The current amount of lives a player has, how many more times they can be hit by the boss before game over |
|  | score : int | The current score that a player has |
| Operations | numLivesZero() : Bool | Checks to see if the current value of NumLives is 0 |
|  | gameOver() : void | Displays game over screen if num lives reaches 0 |
| Relationships | Is created when start game is selected from the main menu. Views the map of the levels and displays the game over screen when necessary. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Map | | This object represents the map of levels that the player will go through during the game |
| Attributes | mapBackground : Image | The image that will be displayed behind the levels and other map icons |
|  | currNode : LevelNode | The node of the map that the player is currently standing on |
|  | winNode: LevelNode | The final node of the map that once beaten indicated the player has won |
|  | nodeList: list | Structure to hold all level objects of the map |
| Operations | playCurrNode() : void | Begins the level associated with the node that the player is currently standing on |
|  | moveLeft() : Bool | Moves the player to the node to the left of them in the dungeon, returns true if move was successful |
|  | moveRight() : Bool | Moves the player to the node to the right of them in the dungeon, returns true if move was successful |
|  | atWinNode() : Bool | Checks if the current node is the win node |
|  | currNodeBeated() : Bool | Checks if the current node’s “Beaten” flag is set to true. |
| Relationships | The map is composed of a tree of multiple LevelNodes. It is viewed by the Player and plays the levels associated with its nodes. Finally, it displays the GameWon screen if the current node is the win node and it has been beaten | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| GameOver | | This object represents the game over screen that will be displayed if the player runs out of lives |
| Attributes | gameOverText : String | The main text that is displayed on the GameOver screen |
|  | score : int | The player’s score at the time of GameOver being displayed |
|  | background : Image | The background image displayed behind the other attributes on the GameOver screen |
|  | playAgain : Button | A button that begins a new gameplay loop |
|  | quit: Button | A button that returns the player to the MainMenu where they can exit the game |
| Operations | playAgainClicked() : void | Begins a new gameplay loop by creating a new Player object |
|  | quitClicked() : void | Returns the player to the main menu |
| Relationships | The GameOver screen is displayed by the Player object. It has the option to exit to the main menu or begin a new player instance and restart the game. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| GameWon | | This object represents the game won screen that will be displayed if the player successfully completes the WinNode level |
| Attributes | gameWonText : String | The main text that is displayed on the GameWon screen |
|  | score : int | The player’s score at the time of GameWon being displayed |
|  | background : Image | The background image displayed behind the other attributes on the GameWon screen |
|  | playAgain : Button | A button that begins a new gameplay loop |
|  | quit: Button | A button that returns the player to the MainMenu where they can exit the game |
| Operations | playAgainClicked() : void | Begins a new gameplay loop by creating a new Player object |
|  | quitClicked() : void | Returns the player to the main menu |
| Relationships | The GameWon screen is displayed by the map object. It has the option to exit to the main menu or begin a new player instance and restart the game. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| LevelNode | | This object represents a single node on the tree that makes up the map of the dungeon |
| Attributes | parent : LevelNode | The LevelNode that leads to this node |
|  | left : LevelNode | The left child of this LevelNode |
|  | right : LevelNode | The right child of this LevelNode |
|  | beaten : Bool | Indicates if the level associated with this node has been successfully completed by the player |
| Relationships | The map has a tree of LevelNodes that the player can traverse throughout the gameplay. Each LevelNode is associated with a Level object that can be accessed and played from the node. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Level | | This object represents encounter where the player will be asked questions in order to defeat a boss |
| Attributes | bossHealth : int | The current health level of the boss within the encounter |
|  | player : Sprite | The sprite displayed on screen representing the player |
|  | boss : Sprite | The sprite displayed on screen representing the boss |
|  | background : Image | The background image displayed behind the sprites and other on screen attributes. |
| Operations | askQuestion() : void | Asks a question and displays it on the screen for the player to respond |
|  | answerCheck() : Bool | Called once an answer is provided, returns true is the answer is correct |
| Relationships | Each Level is associated with a specific LevelNode on the Map. The Map plays a Level when the player clicks on the LevelNode that it is associated with. The Level asks questions from the bank of questions to ask. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| Question | | This object represents one of the questions that a Level can ask |
| Attributes | class : string | The class subject that this question is pertaining to |
|  | difficulty : enum | The difficulty level of this question |
|  | text : String | The actual text to display on screen in order to ask the question |
|  | answer : String | The correct answer for the question |
| Relationships | Question objects are asked by Level objects. | |

|  |  |  |
| --- | --- | --- |
| **Element Name** | | **Description** |
| MultipleChoice | | This object represents one of the questions that a Level can ask |
| Attributes | optionA : String | The first possible answer that can be selected |
|  | optionB : String | The second possible answer that can be selected |
|  | optionC : String | The third possible answer that can be selected |
|  | optionD : String | The fourth possible answer that can be selected |
| Relationships | MultipleChoice is a subclass of the Question class. | |

Sequence Diagrams:

Sequence 1: User loses game by answering incorrectly.

The user launces the game and on the main menu chooses to start the game. This creates a new player object with a full set of lives. The user then clicks on the LevelNode they are currently at (CurrNode) and the level for that node begins. Within the level, the user is asked questions if the boss’s health and the player’s number of lives remaining are above 0. Each answer provided by the user is then checked and is wrong so the Player’s number of lives decrements. Once NumLives is 0, the level exits and returns. Upon return, the Player checks their life count to see if it is 0. Because it is 0, the Player then displays the GameOver screen, and the user clicks quit which returns to the Main Menu.

Diagram

Description automatically generated

Sequence 2: User views about page and then quits game

Graphical user interface, text

Description automatically generated

User Launches the game and then on the Main Menu they click on the “About” button. This button opens the projects webpage in their browser. Once they return to the game, they click the quit button which exits the game.

State Diagram:

Diagram

Description automatically generated

# Prototype

The first prototype is a test of user interface design (UI) of the project. It will show an early build focusing on the basic start game loop and select dungeon/difficulty screen and quit game functionality. The only hardware component used is the mouse.

Upon loading the game, it will direct the user to the main menu page of the game. The main menu page consists of the start game button, which leads to a scene changer subroutine that directs to the gameplay loop of the game, and the quit game button, which exits out of the application.

Upon clicking the start game button with the left mouse click, a scene changer event listener will activate the next scene to be loaded. The map select scene is loaded, where the user will select the difficulty of the dungeon presented to them. To enter the dungeon, all the user must do is press the left mouse click while hovering over the icon of the dungeon. This will load the scene specific to the dungeon.

Included on every page except for the main menu is the back button. The back button will allow users to revert back to the previous scene. This will come in useful if the user ever finds themselves miss clicking on accident, or potentially soft locking the user into a gameplay loop where they can’t escape.

The quit game button, as it suggests, will close the game by calling scene changer event listener to exit the application. The quit game button discards any progress the user makes in the dungeons. Each game start up is a new clean save.

* 1. How to Run Prototype

Minimum system requirements:

OS = Windows 7 or higher

CPU = Single core 2.0 GHz

RAM = 2 GB

GPU = Intel UHD graphics or equivalent

Size = 80MB

Hardware = Mouse

Prototypes require a mouse as the minimum hardware component required to play.

Version one of the prototype will have one executable (.exe) that only runs on Windows operating systems. All the necessary files associated with the game will be packaged into a zip file (.zip) to allow for compression and easy distribution. The zip file will be listed on the project website, available for download.

To run prototype one, download the zip file from the project website. Unzip the zip file into the current directory. Popular unzipping software tools are WinRar and 7zip, which are free to download online and use. Run the game by double clicking the .exe application.

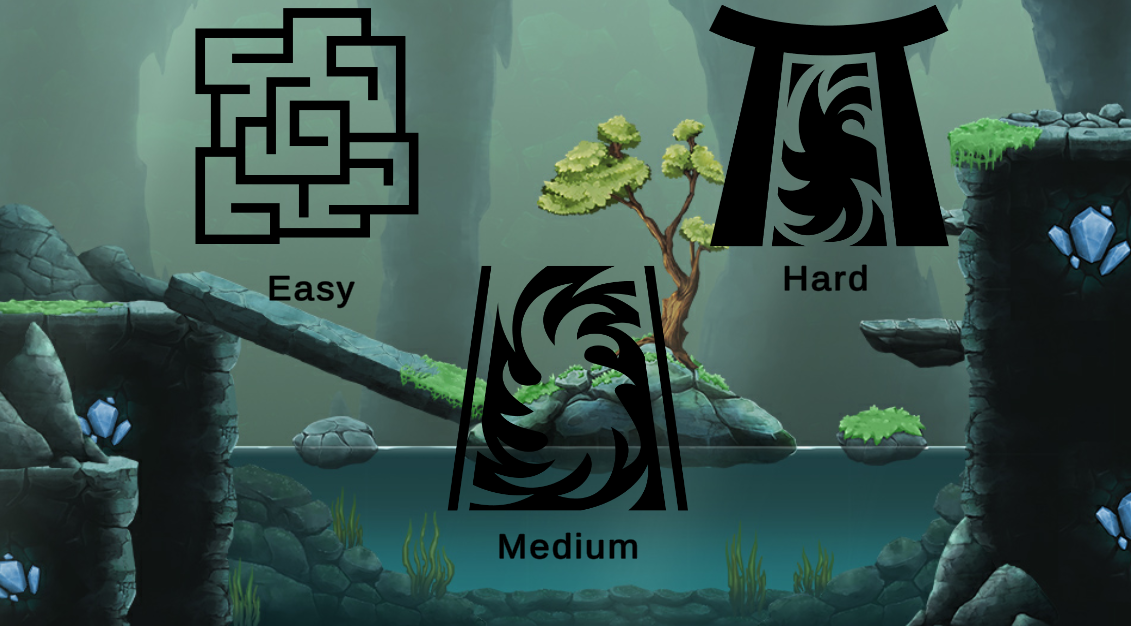
Download version one prototype at:

URL : https://derricklor.github.io/SWEProject/Website/documents.html

* 1. Sample Scenarios



## When the game started the users will see this page where they can start by pressing start or exit by pressing exit, in this case lets assume the users presses the start button.



The user will then see this screen where they can choose between 3 separate difficulty options. The user can press any button which will determine what types of enemies will generate in the dungeon. Lets assume the user is playing for the first time and chooses Easy.

Diagram

Description automatically generated with medium confidence

The user is then brought to this screen where they see the map. The whole map is visible the user can plan out what route they want to take to try and best make it to the end alive. Since the user is at the start there is only one possible encounter option so they will click the sword in the top left of the map.

Graphical user interface

Description automatically generated

This is an example encounter screen with a skeleton enemy. The user answers the questions by selecting the multiple choice and submitting. User can flee if the challenge is too difficult.

# References

1. D. Thakore and S. Biswas, “Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks,” Proceedings of IEEE Military Communication, Atlantic City, October 2005.
2. G. Kaplowitz, N. Miceli, D. Lor, W. LaRose, A. Chavez, “Dungeon Quest” (2021). https://derricklor.github.io/SWEProject/Website/index.html
3. H. Nguyen, “How to Use Gameplay to Enhance Classroom Learning,” Edutopia.org, March 2021.
4. Massachusetts Department of Elementary and Secondary Education. (2017). “Massachusetts Mathematics Curriculum Framework — 2017”. https://www.doe.mass.edu/frameworks/math/2017-06.pdf
5. Squire, Kurt. "Video games in education." *Int. J. Intell. Games & Simulation* 2.1 (2003): 49-62. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.543.5729&rep=rep1&type=pdf>

# Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james\_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.