Software Requirements Specification (SRS)

Decrementor

Team: Group 1
Authors: Angel Chikumbirike, Abhiram Garre, Orion Golden, Nicholas Johnson, Senny Lu, Phyo Naing
Customer: Students in 4th to 5th grade
Instructor: Dr. James Daly

1 Introduction

This Software Requirements Specification (SRS) provides a comprehensive overview of Decrementor, an educational math game. The subsections outline the software's purpose, scope, definitions of key terminology, and the organization of the rest of the SRS.

1.1 Purpose

The purpose of this SRS document is to describe the software's requirements and constraints. It serves as a benchmark for all stakeholders to ensure that all parties share a common understanding of the software's capabilities and limitations. The intended audience of this document includes the software developers, project managers, and clients, such as students from 4th to 5th grade. The audience will enjoy the game as a way to consolidate their foundation in mathematics.

1.2 Scope

The software product to be developed is a 2D edutainment game, Decrementor, with deck-building and rogue-like aspects designed to teach algebraic problem-solving. The application of this software is in the edutainment sector, focusing on educational and entertainment value. The main objective of this software is to teach and/or reinforce algebraic problem-solving skills by challenging the user to come up with unique formulas to meet certain mathematical criteria. As the user progresses, each level will become increasingly harder, forcing them to use their cards more strategically.

1.3 Definitions, acronyms, and abbreviations

SRS - Software Requirements Specification
HW - Hardware
SW - Software
Decrementor - The title of the game this SRS describes
Player - The user playing the game
Game Over - The state of the game that indicates failure.
HP - Health points, a numeric scale used to represent how much damage the player or enemy can take. If the player reaches 0 HP, it's game over. If the enemy reaches 0 HP, the player progresses.
Enemy - An entity in the game that obstructs player progress and must be defeated to advance to further levels.
Level - A stage to be cleared (including a set of enemies that must be defeated) to progress.
Score - A numeric scale representing player success, gained when defeating an enemy.
Card - A component of the player's attack capabilities. This card can be used during the player's turn to perform an attack action.

Operator - A type of card. It may be a binary operation, which takes two inputs (such as addition or subtraction), or a unary operation, which takes a single input (such as negation).
Number - A type of card. Represents a number that is used in combination with operator cards and other number cards.
Hand - The set of all cards the player can use right now
Deck - The set of all possible cards a player can have in their hand.
Equation - A set of cards that represents a valid attack. The player creates it when they play cards from their hand.
Hand Area - UI region where cards dealt to the player appear.
Equation Area - UI region where the player places cards to form a valid expression.
Enemy Area - UI region that displays enemies and their elimination criteria.
Criteria - A mathematical property an integer may satisfy (e.g., even, odd, prime, >30).
Run - A full playthrough from the start of the game until the player loses all HP.
Vulnerabilities - A buff to damage that is available based on whether a criterion is satisfied.

1.4 Organization

The rest of the document is organized in separate sections. Section 2 contains an overall description of the software, including project perspective, project functions, user characteristics, constraints, assumptions, and apportioning of requirements. Section 3 contains specific requirements of Decrementor: a list of functional and non-functional requirements for the software. Section 4 contains modeling requirements: diagrams to demonstrate the software's structure and capabilities. Section 5 contains software prototype descriptions, including their functionality and their running instructions. Section 6 contains references with a list of all resources used in the SRS document. Section 7 contains contact details for more information.

2 Overall Description

This section provides a high-level description of the product, its perspective and function, and the software's intended audience. This section will also cover the constraints of the software and the assumptions and dependencies made about the SW, HW, and user. Finally, this section will go over the requirements that may be addressed in future versions.

2.1 Product Perspective

Decrementor is a standalone web-based game, allowing the target audience to easily access it online through the project website rather than downloading it. The game is designed with simple controls and objectives, making it easy to pick up. Decrementor uses a card drawing system for numbers and operators that players can use to form equations. If the equations satisfy the enemy's requirements, the enemy takes damage or is destroyed. Since progress in the game is

tied to correctly forming these equations, players are encouraged to engage with the material, learning while they play. As the player progresses, they gain more cards, allowing them to form more complex equations to meet increasingly challenging requirements.

Decrementor is intended as a tool to help young learners become proficient with simple math equations through an engaging gaming experience. By providing immediate feedback, Decrementor motivates students to engage with mathematics outside traditional learning environments. Users are gently encouraged to practice and learn new mathematical concepts, including fundamentals such as addition, multiplication, and division, concepts that are typically introduced in 2nd through 5th grade and reinforced in later grades. Alongside this, they are expected to learn new concepts to their age, including prime numbers, multiples, powers, and PEMDAS through the criteria vulnerability system. This system is critical for achieving high scores in the game, encouraging players to learn how to meet these criterias.

Every enemy defeated through Decrementor's equation-based gameplay increases the user's score. A point system encourages players to beat their own high scores and engage in friendly competition. By recording high scores, Decrementor motivates users to return, further strengthening their understanding of mathematics.

2.2 Product Functions

The core gameplay revolves around a deck-building system, hand management, and mathematical battles. Players begin with a starter deck containing exactly one of each card: digits 0 through 9 and the four operators (+, -, ×, ÷). From this deck, they draw to form a hand and build equations. As players win battles, they can add duplicates of specific cards to their deck, strategically increasing the probability of drawing those numbers or operators in future turns.

Each battle features a single enemy with a set number of HP. Enemies also have vulnerabilities, which are available by meeting criteria, that deal extra damage when used against them. During a battle, the player plays cards from their hand to form equations. The game calculates the equation's result and reduces the enemy's HP according to the damage dealt and the enemy's vulnerabilities. When an enemy's HP reaches zero, the enemy is defeated, the player's score increases, and a new enemy appears for the next battle.

Finally, Decrementor features a main menu and a pause menu, giving players easy control over quitting the game or adjusting settings.

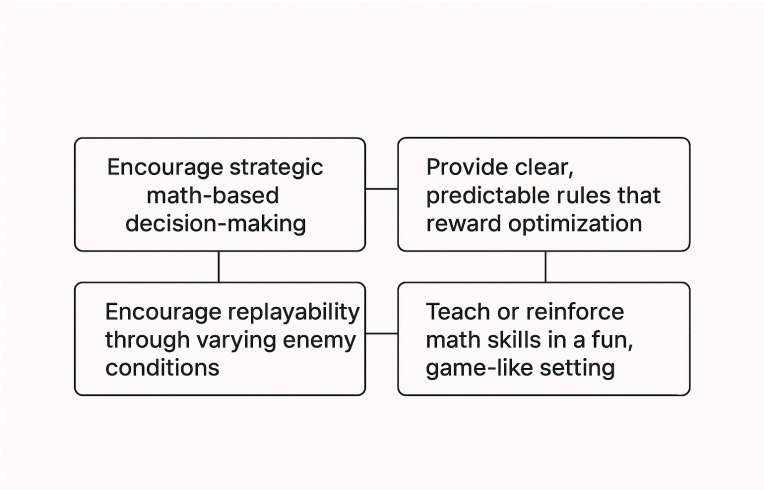

Figure 0: High-Level Goals

2.3 User Characteristics

Users will be American elementary and middle school students in the 4th to 5th grade, with a preferred narrowed range of 4th-5th graders due to the subject material being taught. A grade-level proficiency in English is expected. These students have a beginner or intermediate understanding of mathematical concepts such as multiplication and division, and how to form an equation. They must also have a basic understanding of websites and video games to access and play Decrementor, and experience using any form of computer.

2.4 Constraints

Since the game is aimed at students from 4th to 5th grade, all content, including visuals, language, and gameplay elements, must be suitable for this age group. This limits the types of challenges, themes, and scenarios that can be included in the game. This means the game must meet the standards for a PG-13 rating at the very least, and if possible, a lower rating. This rating limits the possible themes and imagery allowed to be displayed in this game. This also means there is a duty present to protect these children on any online services we offer to prevent them from being taken advantage of, which entails restrictions should that route be taken.

The game's mathematics is also constrained to the Massachusetts Mathematics Curriculum Framework up to the 5th grade; therefore, it limits the complexity and difficulty of the material taught. The game is limited to basic keyboard and mouse inputs so that target users can easily interact with it. This constraint affects the design of gameplay mechanics, preventing the inclusion of more complex control schemes, such as pressing multiple key combinations simultaneously or using gamepad buttons.

Finally, the game is also constrained by the developer's limited time and resources to complete Decrementor, as they may not have the time to fully implement all features desired.

2.5 Assumptions and Dependencies

The game assumes the user has access to a computer with a screen or monitor, along with a trackpad or mouse for interaction. Internet access is also required to visit the website where the game can be downloaded. If the user chooses to play the game in the browser, they must be using a modern browser that supports HTML5. The quality of the game will also be dependent on the browser and the user's computer capabilities.

2.6 Apportioning of Requirements

Some requirements are intended for later versions of the system and will not be fully implemented in the initial release. The following are some of the features:

- An in-game tutorial at the start of a game to guide players through their first enemy.
- Advanced difficulty-scaling features, such as dynamic generation of complex enemy criteria or adaptive difficulty based on player performance.
- Additional card types (such as multi-step modifiers, parenthesis cards, or special effect cards) may be postponed to a later stage of development depending on production resources.

3 Specific Requirements

1. The system shall provide a way for the player to begin a new play session.
2. The system shall provide a way for the player to stop playing and exit the application at any time before or during a play session.
3. The system shall allow the player to temporarily suspend gameplay.
4. When gameplay is suspended, all time-dependent and turn-based mechanics shall halt.
    - 4.1. While gameplay is suspended, the player shall be able to:
        - 4.1.1. Adjust global audio settings.
        - 4.1.2. Return to the initial application screen.
        - 4.1.3. Exit the application.
5. The system shall allow the player to interact with cards.
    - 5.1. The system shall provide the player with access to all cards available for use during their turn.
    - 5.2. The system shall allow the player to select and arrange cards to form an attack expression.
    - 5.3. The system shall provide feedback on the value or result of the player's constructed expression.
    - 5.4. The system shall provide the player with information about the enemy's HP and their vulnerabilities.

6. The system shall maintain a Deck, a Discard Pile, and a Player Hand.
    6.1. At the beginning of each turn, the system shall deal cards from the Deck to the Player Hand.
    6.2. The system shall support three card types: Number Cards and Operation Cards (Unary Operation Cards and Binary Operation Cards).
        6.2.1. Number Cards shall contain a whole integer value.
        6.2.2. Operation Cards shall display a binary operator (+, −, ×, ÷) or unary operator (e.g., negate, square).
    6.3. At the end of each turn, all cards placed in the Equation Area shall be moved to the Discard Pile.
    6.4. When the Deck is empty, the system shall reshuffle the Discard Pile to form a new Deck.
7. The system shall have an Equation Area containing cards placed by the player.
    7.1. The system shall parse the sequence of cards in the Equation Area from left to right to compute an integer value.
    7.2. The system shall not allow two operation cards or two number cards to be placed next to each other in the equation area.
    7.3. The system shall detect invalid expressions (e.g., division by zero) and reject them with an error message.
8. The system shall have enemies.
    8.1. Each enemy shall contain a mathematical criterion that evaluates a computed integer as true or false.
    8.2. The enemy shall be eliminated when its HP is zero.
    8.3. The enemy shall attack and decrease the player's HP every time the player's turn is over, and its HP is not zero.
9. The game shall become more difficult as the player clears more levels.
    9.1. The number of mathematical criteria on enemies shall increase.
    9.2. The HP of the enemy increases upon completion of each level.
10. The player shall begin each gameplay with a defined number of HP.
    10.1. When an enemy deals damage, the system shall reduce the player’s HP by the set value.
    10.2. The system shall end the game when the HP of the player reaches 0.
    10.3. At the end of a game, the system shall display the death screen with the score and input for the username.
11. The system shall track a numerical score for the player during gameplay.
    11.1. The system shall award points for eliminating enemies and completing levels.
    11.2. The system shall display the score during gameplay.
    11.3. The system shall store completed game scores in a global leaderboard.

4 Modeling Requirements

This section contains the models created to visualize multiple aspects of the project. This includes the use case diagram, the class diagram, two sequence diagrams, and a state diagram. The use case diagram is meant to illustrate user-visible functions and their interactions in the system.

The class diagram describes the system's structure, mainly data and methods. The sequence diagrams show how the objects in a system interact with each other in sequential order. The state diagram shows the game's states and the events that trigger those states.

4.1 Use Case diagram

The following diagram contains use cases for Decrementor. The player is denoted with a stick figure on the left. Use cases are denoted in ovals, and the lines represent how they are connected with other use cases. The player is connected directly to some use cases: "End turn and Attack Enemy", "Place Card on Equation Area", "Move Card on Equation Area", "Start Game", "Pause Game", and "Change Settings"; these use cases represent the actions the player can make in the software. The use cases are also connected to other use cases through arrows labeled with "<<includes>>" and "<<excludes>>" to show their relationships.

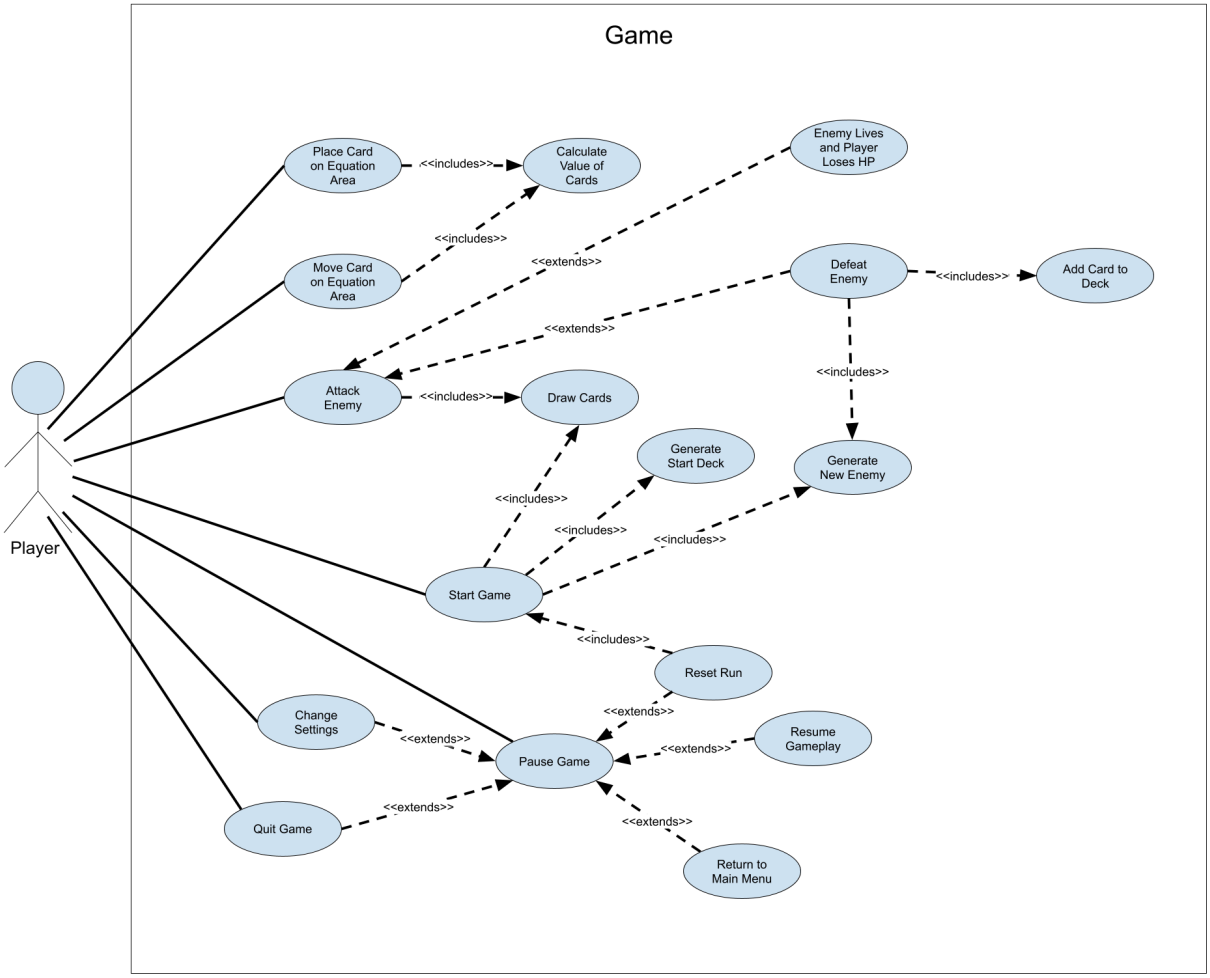

Figure 1: Use Case Diagram for Decrementor

Use Case Name:	**Start Game**
Actors:	Player
Description:	The player starts the game.
Type:	Primary and essential
Includes:	Generate Start Deck, Generate New Enemy, Draw Cards
Extends:	None
Use Cases:	Reset Run

Use Case Name:	**Generate Start Deck**

Actors:	Player
Description:	The game manager generates the starting deck for the player at the start of the game.
Type:	Secondary and essential
Includes:	None
Extends:	None
Use Cases:	Start Game

Use Case Name:	**Generate New Enemy**
Actors:	Player
Description:	The game manager generates a new enemy at the start of the game and after an enemy is defeated.
Type:	Secondary and essential
Includes:	None
Extends:	None
Use Cases:	Start Game, Defeat Enemy

Use Case Name:	**Draw Cards**
Actors:	Player
Description:	Players draw cards at the start of the game and after each turn.
Type:	Secondary and essential
Includes:	None
Extends:	None
Use Cases:	Start Game, End Turn, and Attack Enemy

Use Case Name:	**Pause Game**
Actors:	Player

Description:	The player pauses the game.
Type:	Primary
Includes:	None
Extends:	None
Use Cases:	Change Settings, Quit Game, Reset Run, Return to Main Menu, Resume Gameplay

Use Case Name:	**Quit Game**
Actors:	Player
Description:	The player quits the game on the main menu or after pausing the game.
Type:	Primary
Includes:	None
Extends:	Pause Game
Use Cases:	None

Use Case Name:	**Change Settings**
Actors:	Player
Description:	The player changes game settings, such as volume on the main menu or after pausing the game.
Type:	Primary
Includes:	None
Extends:	Pause Game
Use Cases:	None

Use Case Name:	**Resume Gameplay**
Actors:	Player
Description:	The player resumes play after pausing the game.

Type:	Secondary
Includes:	None
Extends:	Pause Game
Use Cases:	None

Use Case Name:	**Return to Main Menu**
Actors:	Player
Description:	The player returns to the main menu after pausing the game.
Type:	Secondary
Includes:	None
Extends:	Pause Game
Use Cases:	None

Use Case Name:	**Reset Run**
Actors:	Player
Description:	The player resets their run after pausing the game.
Type:	Secondary
Includes:	Start Game
Extends:	Pause Game
Use Cases:	None

Use Case Name:	**Place Card on Equation Area**
Actors:	Player
Description:	The player moves a card from their hand to the equation area.
Type:	Primary and essential
Includes:	Calculate Value of Cards

Extends:	None
Use Cases:	None

Use Case Name:	**Move Card to Equation Area**
Actors:	Player
Description:	The player moves a card from one spot on the equation area to a different spot on the equation area.
Type:	Primary
Includes:	Calculate Value of Cards
Extends:	None
Use Cases:	None

Use Case Name:	**Calculate Value of Cards**
Actors:	Player
Description:	The game manager calculates the value of cards in the equation area.
Type:	Secondary and essential
Includes:	None
Extends:	None
Use Cases:	Place Card on Equation Area, Move Card on Equation Area

Use Case Name:	**Attack Enemy**
Actors:	Player
Description:	The player ends their turn and attacks the enemy.
Type:	Primary and essential
Includes:	Draw Cards
Extends:	Defeat Enemy, Enemy Lives and Player Loses HP
Use Cases:	None

Use Case Name:	**Enemy Lives and Player Loses HP**
Actors:	Player
Description:	The player fails to defeat the enemy and loses HP.
Type:	Secondary and essential
Includes:	None
Extends:	Check Criteria to Defeat Enemy
Use Cases:	None

Use Case Name:	**Defeat Enemy**
Actors:	Player
Description:	The player defeats the enemy.
Type:	Secondary and essential
Includes:	Generate New Enemy, Add Card to Deck
Extends:	Check Criteria to Defeat Enemy
Use Cases:	None

Use Case Name:	**Add Card to Deck**
Actors:	Player
Description:	The player chooses a card to add to their deck.
Type:	Secondary and essential
Includes:	None
Extends:	None
Use Cases:	Defeat Enemy

4.2 Class Diagram

The following class diagram shows the classes that make up Decrementor. Each class consists of its names, attributes, and methods. The classes are connected through arrows labeled with their relationships to other classes.

The system modeled by this diagram is one spearheaded by the Game Manager. The game manager controls most things about the game, such as the turn queue, the levels, the spawning of new enemies, and checking whether the game is over. The rest of the diagram can be split into roughly three areas, those being the Player class, the Enemy class, and the various card container classes. The Player class stores most data about the player, like their name and HP (labeled in this diagram as currentHealth). The Enemy class is similar, storing the enemy's HP and the criteria that determine vulnerabilities. The various card containers represent how the cards are stored in the game, and they are all composed of cards, which is the superclass of the types of cards available in the game.

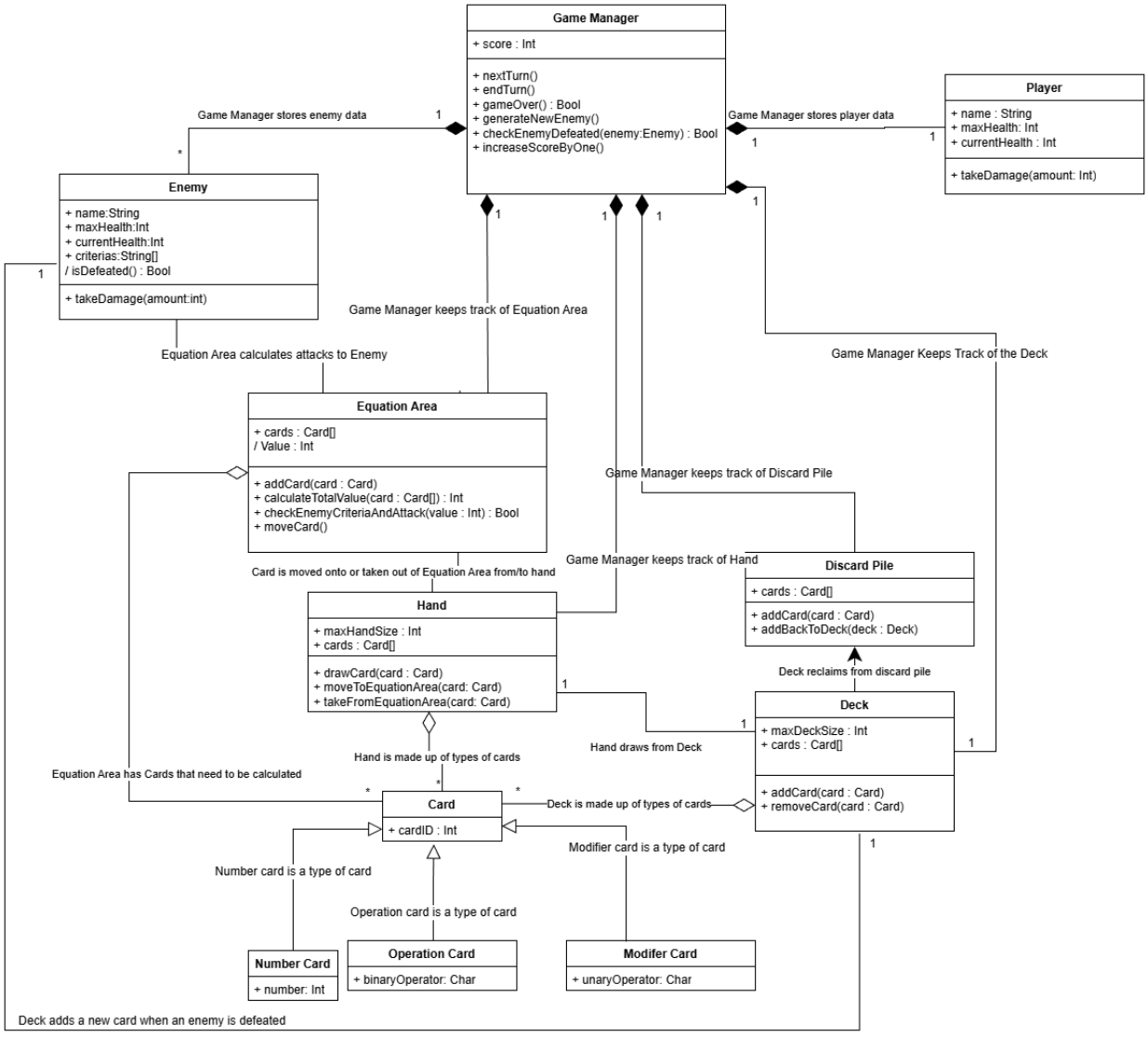

Figure 2: Class Diagram for Decrementor

Element Name		**Description**
Game Manager		The central game object, which keeps track of score & turns. Controls level switching and spawning enemies.
Attributes		
	Score: Int	The score that the player has accumulated is updated on enemy defeat.
Operations		
	nextTurn()	Advances to the next turn.
	endTurn()	Finishes current turn.
	gameOver(): bool	Determines if the game is over.
	generateNewEnemy()	Spawns a new enemy in the scene.
	checkEnemyDefeated(enemy: Enemy): Bool	Checks if the enemy is defeated.
	increaseScore()	Increases the player's score.
Relationships	Game Manager stores Enemy data.	
	The Game Manager tracks the Equation area.	
	The Game Manager tracks the Hand.	
	The Game Manager tracks the Discard Pile.	
	The Game Manager tracks the Deck.	
	Game Manager stores player data.	
UML Extensions	None.	

Element Name	**Description**
Player	Representing the user playing the game.
Attributes	

	name: String	Player name for leaderboard.
	maxHealth: int	The maximum possible health the player can have.
	currentHealth: int	The player's current health.
Operations		
	takeDamage(amount: int)	Decreases the player's current health by the amount.
Relationships	Player data is stored in Game Manager.	
UML Extensions	None.	

Element Name		**Description**
Enemy		A single enemy the player must defeat to progress to the next level.
Attributes		
	name: String	The name of this enemy.
	maxHealth: int	The maximum health of this enemy.
	currentHealth: int	The current health of this enemy.
	criterias: String[]	The criteria that must be met by an equation to damage the enemy.
	isDefeated: bool	Determines if the enemy is defeated.
Operations		
	takeDamage(amount: int)	Damages the enemy by an amount.
Relationships	Enemy Data is stored in the Game Manager.	
	The enemy is damaged by the Equation Area.	
	When the Enemy is defeated, the Deck gains a new card.	
UML Extensions	None.	

Element Name		**Description**
Equation Area		The area where cards are played to form equations that attack the enemy.
Attributes		
	cards: Card[]	The set of cards is currently played in the equation area.
	value: int	The calculated value of the equation.
Operations		
	addCard(card: Card)	Add the card to the play area.
	calculateTotalValue(card: Card) int	Calculates the value of the card.
	checkEnemyCriteriaAndAttack(value: int): bool	Checks if the equation meets the criteria and attacks with the value as damage.
Relationships	The equation area calculates the attacks on the Enemy	
	The equation area is made up of cards.	
	The equation area receives cards played by hand.	
	The Game Manager tracks the equation area.	
UML Extensions	None.	

Element Name		**Description**
Hand		The hand contains the cards currently playable by the player.
Attributes		
	maxHandSize: int	The maximum number of cards capable of being held by the hand.
Operations		
	drawCard(card: Card): void	Draw a card from the deck.
	moveToEquationArea(card:	Moves the selected card to the equation

	Card)	area.
	takeFromEquationArea(card: Card)	Take a card from the equation area.
Relationships	Hand plays to the Equation area.	
	The hand is made of types of cards.	
	Hand draws from the Deck.	
	The Game Manager tracks the hand.	
UML Extensions	None.	

Element Name		**Description**
Discard pile		Cards used in a turn are discarded into the discard pile until reclaimed by the Deck.
Attributes		
	cards : Card[]	The set of cards in the discard pile
Operations		
	addCard(card: Card)	Adds card to discard pile.
	returnToDeck(deck: Deck)	Adds cards from the discard pile back into the Deck.
Relationships	The Game Manager keeps track of the discard pile.	
	Cards in the Discard pile are reclaimed by the Deck.	
UML Extensions	None.	

Element Name		**Description**
Deck		The deck is all the possible cards that the player can draw into their hand.
Attributes		
	maxDeckSize: int	The number of cards in the deck,

		currently.
	cards : Card[]	The set of cards in the deck.
Operations		
	addCard(card: Card)	Adds a card to the deck.
	removeCard(card: Card)	Removes a card from the deck.
Relationships	Deck reclaims from the discard pile.	
	Deck deals to hand.	
	The deck is made of types of cards.	
	Deck adds a new card every time an enemy is defeated.	
	The Game Manager tracks the deck.	
UML Extensions	None.	

Element Name		**Description**
Card		A single card, which can be in the hand, deck, discard pile, or equation area.
Attributes		
	cardID: int	A unique ID is created for each card.
Operations		
	None.	The card has no operations.
Relationships	The Equation Area has cards that must be calculated.	
	The hand contains cards.	
	The deck contains cards.	
UML Extensions	None.	

Element Name	**Description**

Number Card		A type of card with a number on it.
Attributes		
	number: int	The number on the card
Operations		
	None.	This class has no operations.
Relationships	None.	
UML Extensions	Number card extends Card.	

Element Name		**Description**
Operation Card		A type of card with binary operation.
Attributes		
	binaryOperator: char	The operator on the card.
Operations		
	None.	This class has no operations.
Relationships	None.	
UML Extensions	Operation Card extends Card.	

Element Name		**Description**
Modifier Card		A type of card with a unary operation on it.
Attributes		
	unaryOperator: char	The operator on the card.
Operations		
	None.	This class has no operations.
Relationships	None.	

UML Extensions	Modifier Card extends Card.

4.3 Sequence Diagram 1

The following sequence diagram shows the process of the player ending their turn and attacking the enemy. This sequence is triggered when the player ends their turn. The game manager will send a request to the equation area, where the cards placed by the player during the turn are then used to attack the enemy. If the enemy is not defeated, the player will take damage. If the enemy is defeated, the player will be prompted to choose a card to add to their deck.

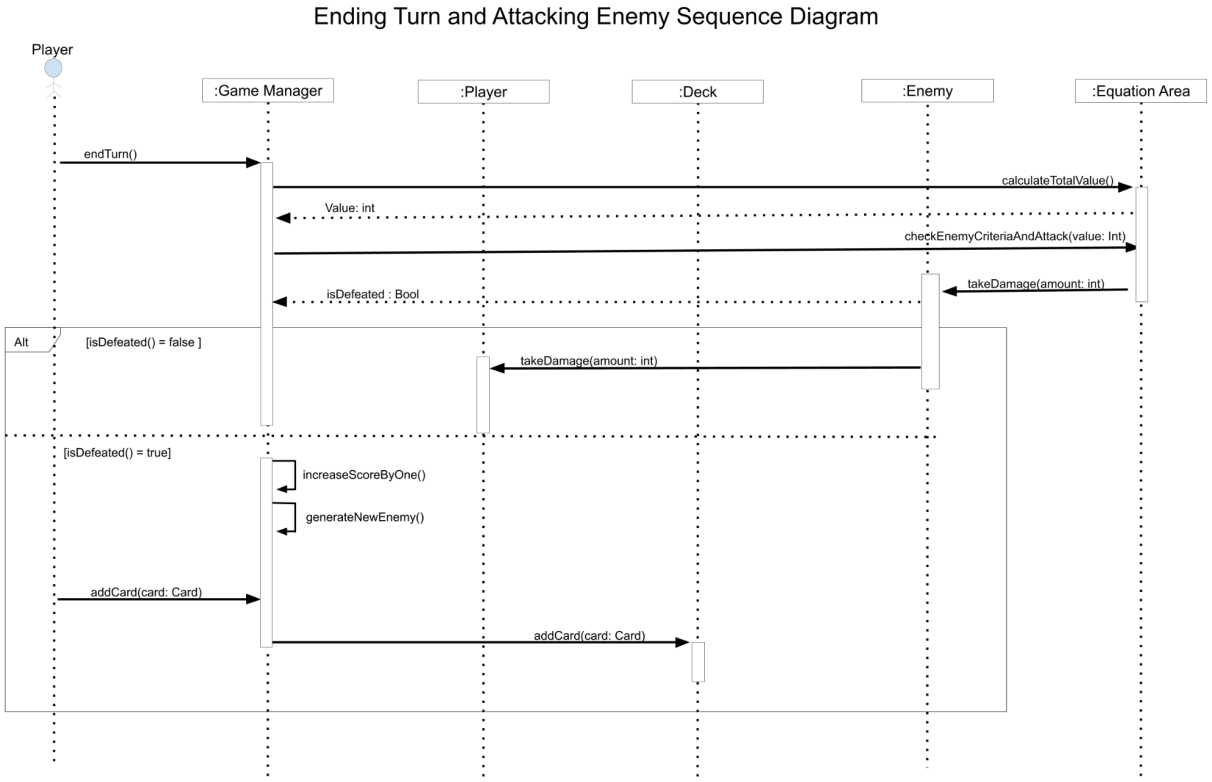

Figure 3: Sequence Diagram 1 for Decrementor

4.4 Sequence Diagram 2

The following sequence diagram shows the process of the player interacting with cards, specifically the process of the player moving cards between their hand and the equation area. The player can either place cards from their hand onto the equation area, move cards around on the equation area, or remove cards from the equation area. After each displacement of cards, the player will receive real-time information on the total value of the cards in the equation area.

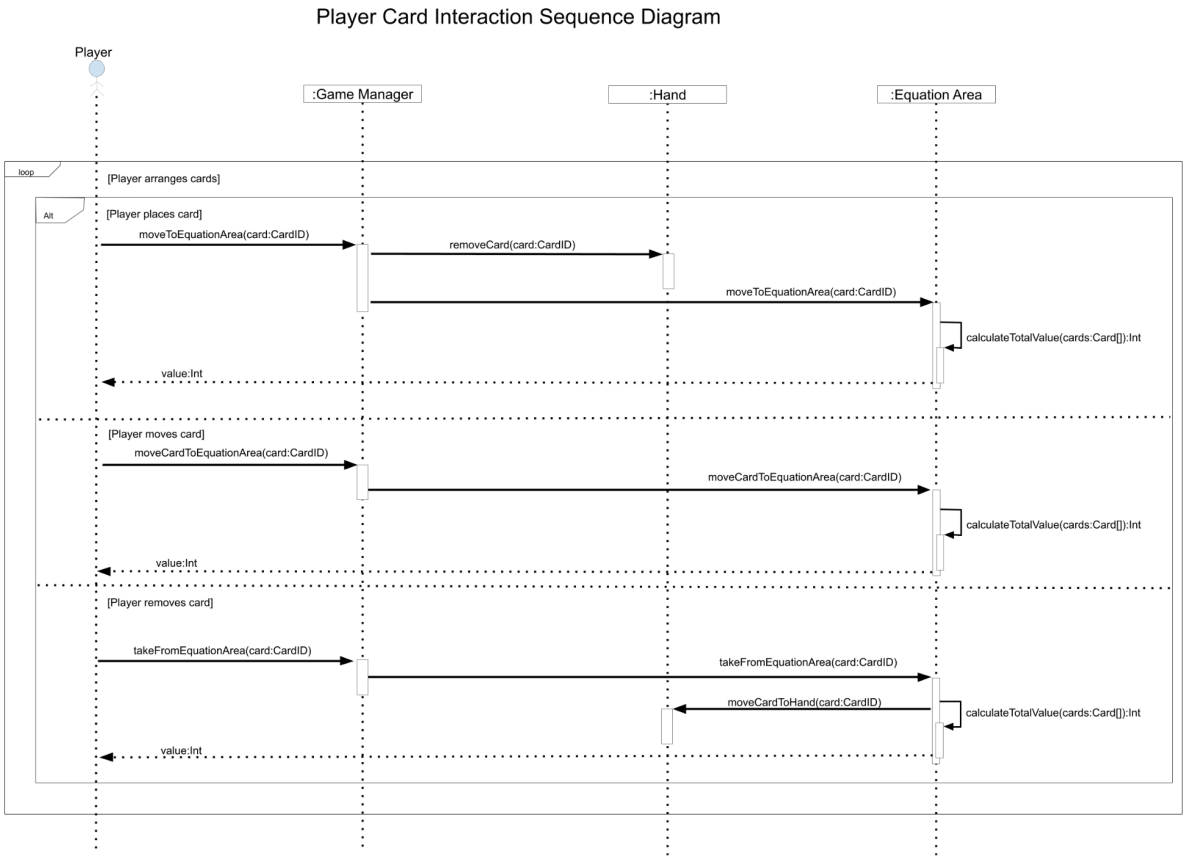

Figure 4: Sequence Diagram 2 for Decrementor

4.4 State Diagram

The following state diagram provides an overview of the game's states and the flow between them. The player starts on the main menu, where they can choose to start the game, change settings, or quit the game. Once the game begins, the player can pause at any time to access the main menu and adjust settings. When the game is unpaused, the player can freely move cards during their turn and may end their turn whenever they choose. If the enemy is defeated, the player will be able to add a card to their deck. If the enemy is not defeated, the player will take damage. When the enemy eventually defeats the player, the game-over screen will be shown, and the player may choose to continue.

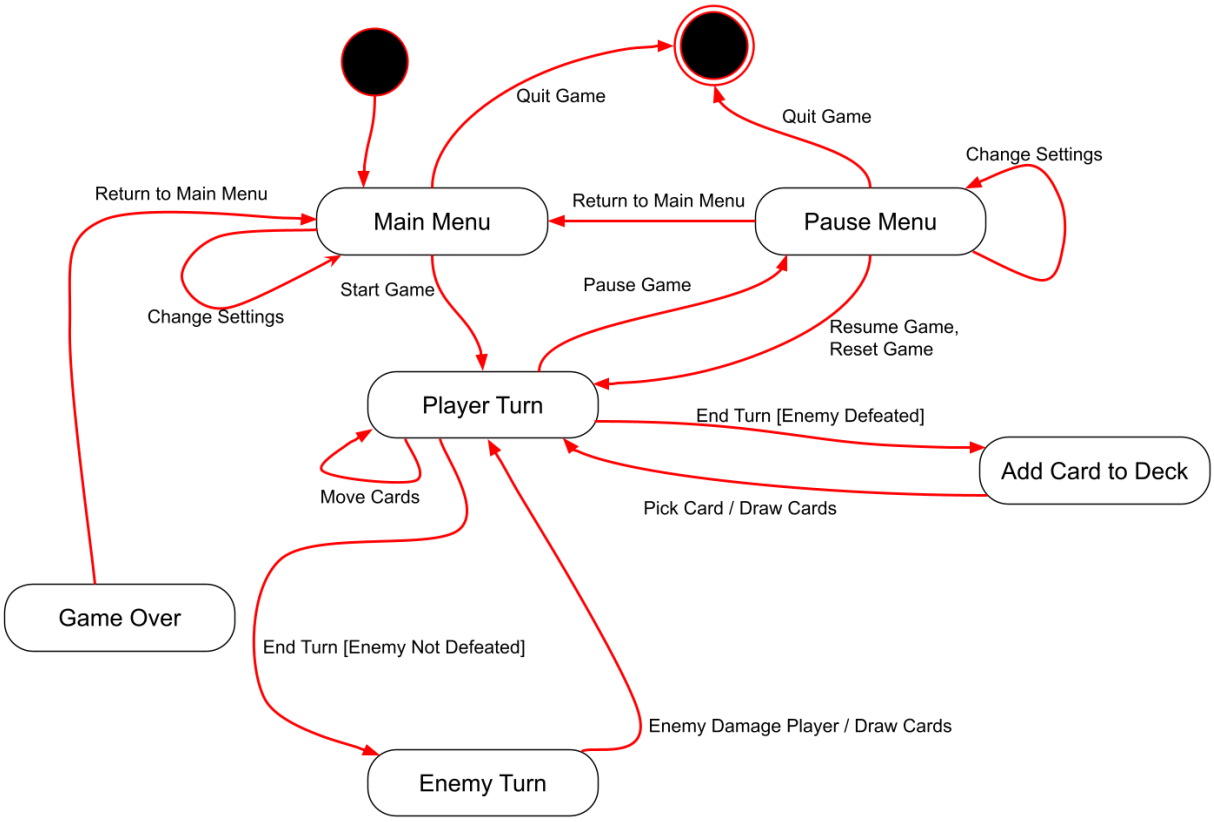

Figure 5: State Diagram for Decrementor

5 Prototype

The prototype of the game will feature all functional requirements of the game. This means it will include a main menu, a game scene that advances the levels, and a pause menu on the game scene. It will also include the ability to form equations and defeat enemies.

5.1 How to Run Prototype

Github Repo: https://github.com/Software-Engineering-I-Group-1/Equation-Deckbuilder-Roguelike
Website: https://software-engineering-i-group-1.github.io/Equation-Deckbuilder-Roguelike/

Building is not required, as the game can be accessed from the website. Simply click on the link next to the website, then click on the prototype tab present in the navbar.

This game also comes with an .exe version that can be played locally. To play it this way, simply download the .zip file from the Github repo, extract it, and navigate to Equation-Deckbuilder-Roguelike-main\decrementor\Decrementor.exe. Click on the .exe file.

If you want to build the game yourself, open the extracted project downloaded from the GitHub repo in Godot. Then click on Project, Export, and export it as Windows Desktop (change if not Windows Desktop). You should then be able to overwrite the Decrementor.exe file already present. Do not try to

export as Web (Runnable), as while this is the way the official website build was exported, it will not work on your local machine.

5.2 Sample Scenarios

The player is first presented with the main menu screen when opening the game.

Figure 6: Main Menu Screen

The player can navigate to the leaderboard for scores of past runs.

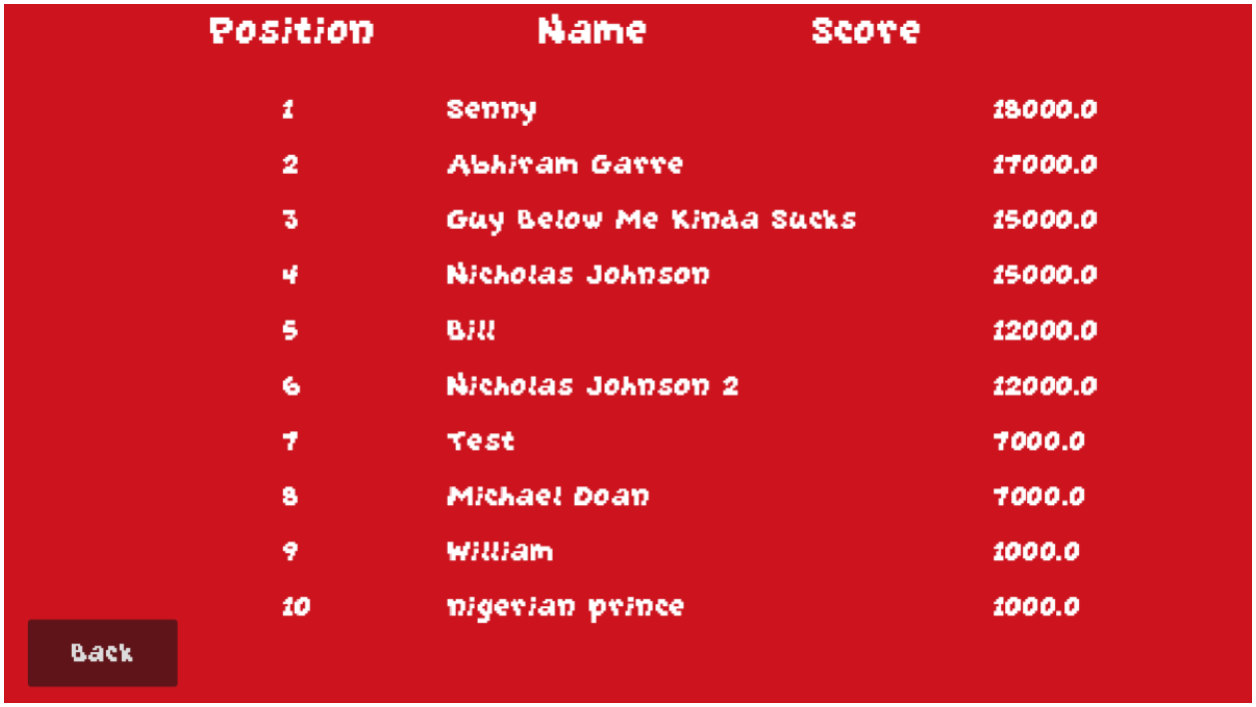

Figure 7: Leaderboard Screen

Players can navigate to settings to change them.

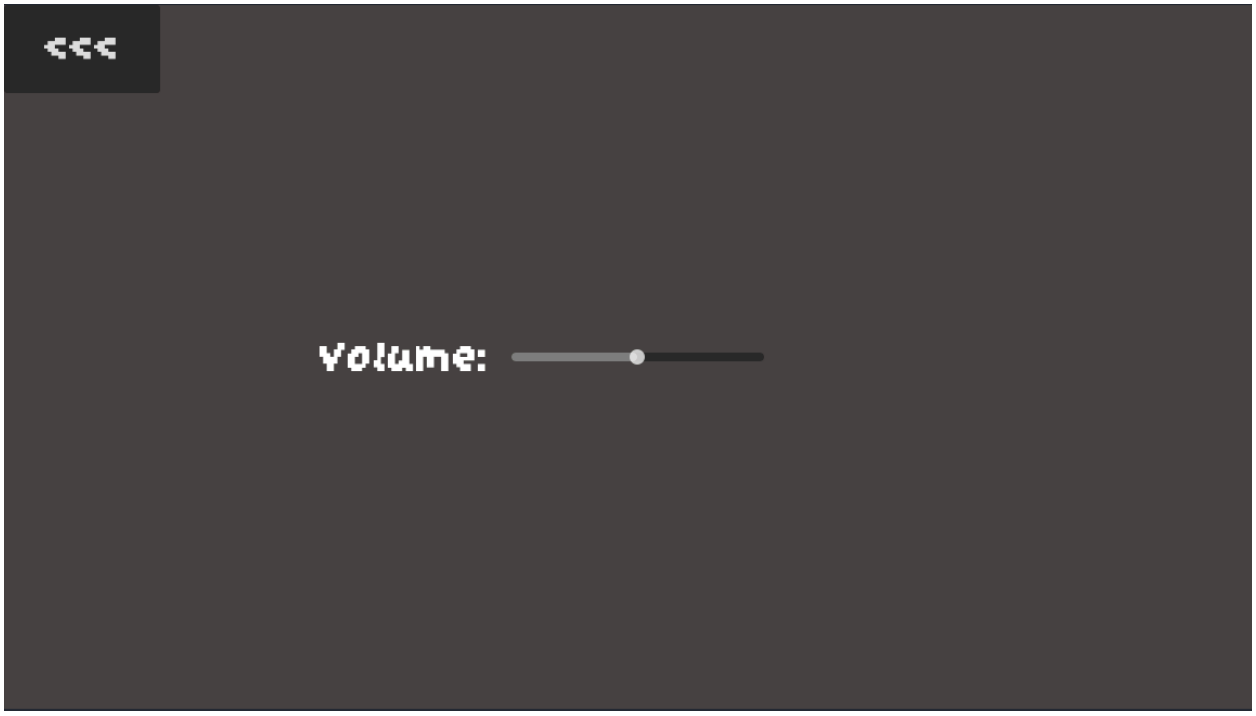

Figure 8: Settings Screen

This is the gameplay screen where the player interacts with cards to defeat enemies. At the gameplay screen, the player sees their hand of cards, the area where they build equations, and the enemy they must defeat. Each turn, the player chooses cards to form a valid equation that meets an enemy's mathematical condition. The player continues building equations each turn until the enemy is defeated:

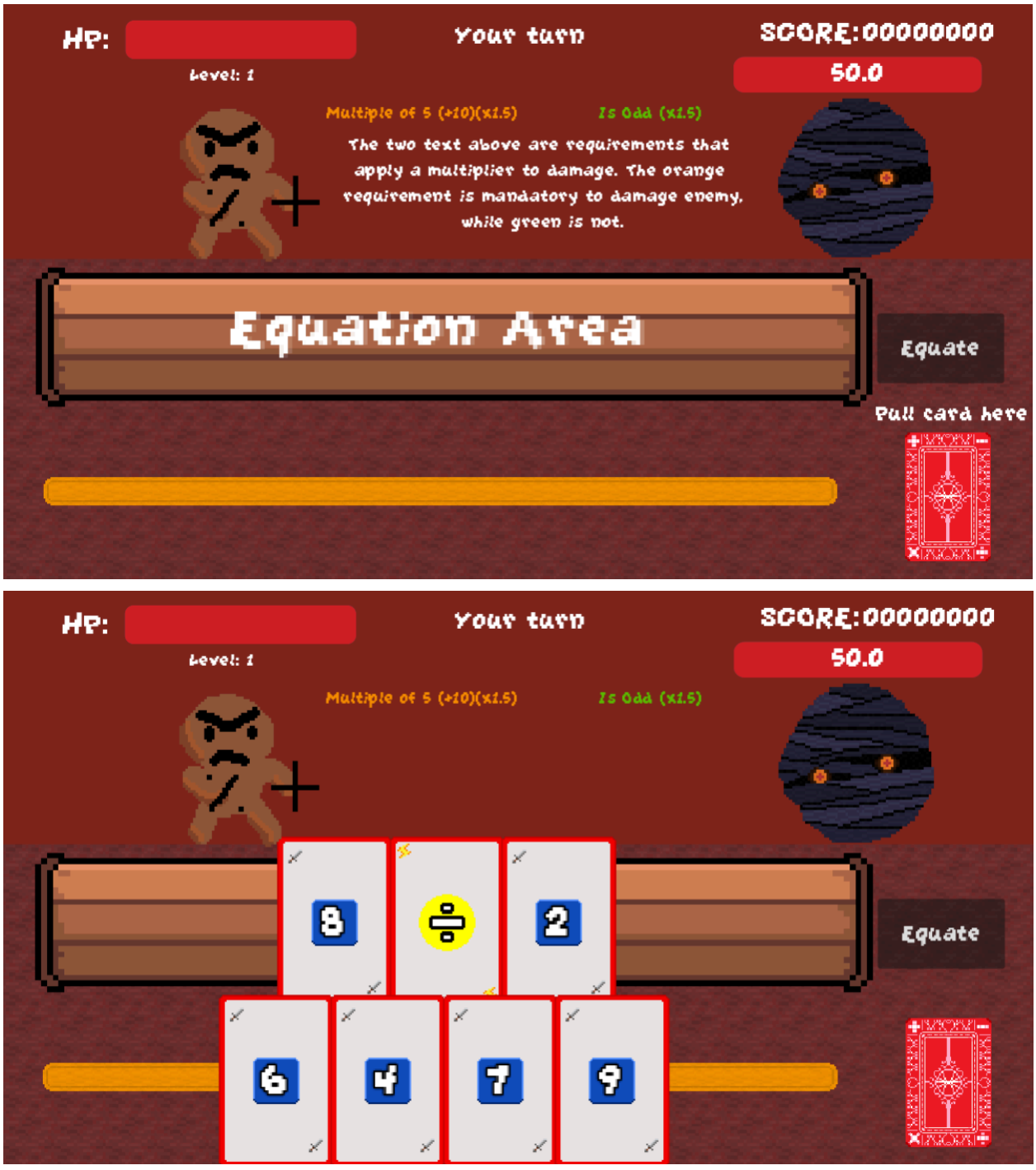

Figure 9-10: Gameplay Screens

This is the adding card screen, where the player can choose a card to add to their deck.

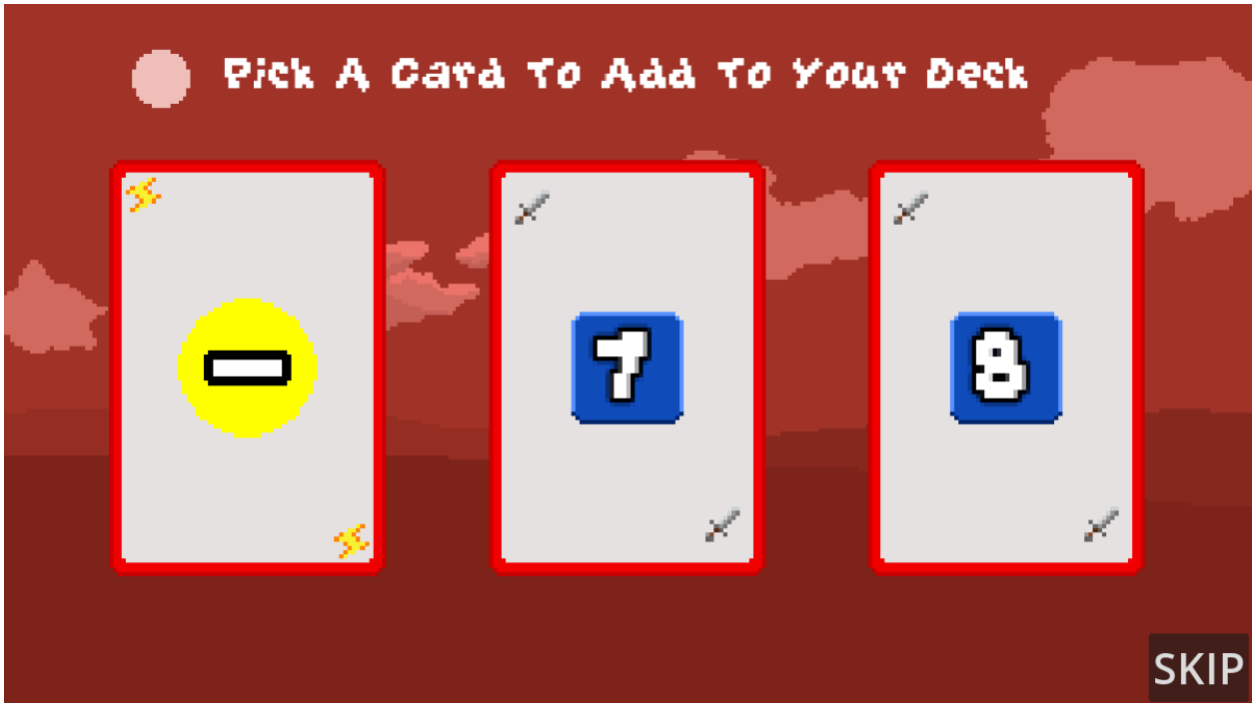

Figure 10: Adding Card Screen

This is the game-over screen where the player can submit their score to the leaderboard and go back to the main menu.

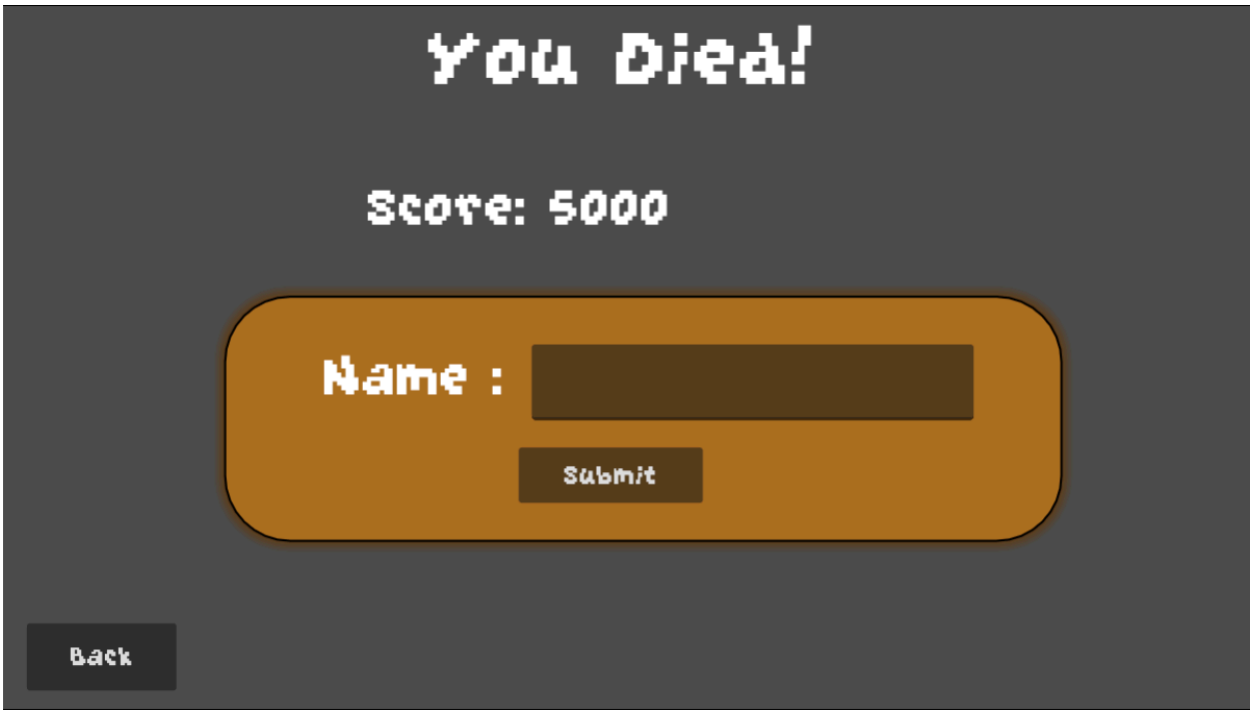

Figure 11: Ending Game Screen

6 References

[1] *Massachusetts Curriculum Framework for Mathematics*, Massachusetts Department of Elementary and Secondary Education, 2017, https://www.doe.mass.edu/frameworks/math/2017-06.pdf

[2] *Decrementor Github*, A. Chikumbirike, 2025, A. Garre, O. Golden, N. Johnson, S. Lu, P. Naing, 2025 https://github.com/Software-Engineering-I-Group-1/Equation-Deckbuilder-Roguelike

[3] *Decrementor Website*, A. Chikumbirike, A. Garre, O. Golden, N. Johnson, S. Lu, P. Naing, 2025 https://software-engineering-i-group-1.github.io/Equation-Deckbuilder-Roguelike/

[4] *jeti, "Grape Soda*", dafont.com, 20 December 2018, https://www.dafont.com/grapesoda-2.font
License: https://creativecommons.org/licenses/by/4.0/

[5] *Dance Queen."* White Records, Dudchyk Maksym, 2025, https://pixabay.com/music/beats-dance-queen-synth-pop-background-music-for-vlog-video-stories-short-2-427667/
License: https://pixabay.com/service/license-summary/
All art assets are made by Nicholas Johnson, Phyo Naing, and Orion Golden

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at the University of Massachusetts at 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.