Software Requirements Specification (SRS)
Decrementor

Team: Group 1
Authors: Angel Chikumbirike, Abhiram Garre, Orion Golden, Nicholas Johnson, Senny Lu, Phyo Naing
Customer: Students in 6-8th 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 purpose of the software, the scope of the software, the definitions of important 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 reference for all of the stakeholders to ensure all parties share a common understanding of the software's capabilities and limitations. The intended audience of this document includes the developers of the software, the project managers, and the clients.

1.2 Scope

The software product to be developed is a 2D edutainment game with deck-building and roguelike aspects designed to teach algebraic problem-solving. The application of this software is 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 more difficult, forcing the user 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 can take before a game is over.
Enemy - An entity in the game that obstructs player progress and must be defeated in order to advance levels.
Level - A set of enemies that must be defeated in order to progress.
Score - A numeric scale representing player success, gained when defeating an enemy.
Card - A component of the player's attack capabilities. Can be played.
Operator - A type of card. Represents either a binary operation or a unary operation.
Number - A type of card. Represents a number that is acted upon by the operator cards in addition to other numbers.
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. It is created by the player 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.

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, as well as 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. This allows 4th to 8th graders a means to easily access Decrementor online through the project website, rather than downloading it. The game is designed with simple controls and objectives that make the game easy to pick up. Decrementor uses a card drawing system for the numbers and operators that users can play. The player will then use those cards to make equations, and if they satisfy requirements on the enemy, will damage/destroy it. Since progress in the game is tied to correctly forming these equations, this will encourage player engagement with the material, teaching them while they play. And on top of that, the further the player gets into their run, the more cards they receive. This means they can form more complex equations that can meet more complex requirements.

Decrementor is meant as a tool to help 4th to 8th graders become proficient with simple math equations through the engaging medium of gaming. By engaging and giving positive feedback, Decrementor gives 4th to 8th graders a reason to engage with mathematics outside of traditional learning environments. Users are gently encouraged to learn new mathematical concepts while training their speed and proficiency with mathematical fundamentals such as addition, multiplication, and division.

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

2.2 Product Functions

Decrementor will have a main menu screen and a pause menu, which allows the user easy control over when to quit the game or to adjust settings.

Decrementor's gameplay centers around several mechanics. A deck system, a card system for your hand, a battle sequence with an enemy, and a score counter. The deck will start out with a fixed set of cards from which the user will randomly draw. These cards will represent numbers and operators for the user to play in battle. When a battle is completed, the user will get to add new cards to the deck, which then have a chance of being drawn to the user's hand whenever the user draws cards.

Each battle will have one enemy with vulnerabilities and hit points. The battle sequence will consist of the user playing cards from their hand in order to form equations. The result of the equation will then be calculated, and the game will subtract hit points from the enemy according to the result of the equation and the enemy's vulnerabilities. If an enemy's hit points reach zero, the enemy is defeated, the score counter is increased, and a new enemy will be generated for the next battle.

2.3 User Characteristics

Users will be 4th to 8th graders who are able to use a mouse and keyboard. Users will have an understanding of how keyboards are used and know common buttons such as the “Esc” key. They will also need to have an understanding of how to use a mouse. Users will have a beginner or intermediate understanding of mathematical concepts such as multiplication and division. Users will have a basic understanding of websites and video games in order to access and play Decrementor.

2.4 Constraints

Since the game is aimed towards 4th to 8th graders, its content must be appropriate for these demographics. The mathematics of the game is also constrained to the Massachusetts Mathematics Curriculum Framework up to the 8th grade; therefore, it limits the complexity and difficulty of the initial levels of the game. The game is also constrained by the developer’s limited time and resources to complete Decrementor. Also, the game is constrained to simple mechanical controls, such that the users are able to easily play the game.

2.5 Assumptions and Dependencies

The game assumes that the user has access to a computer with a screen or monitor to display the game and either a trackpad or a mouse to interact with the game. The user must also have internet access to access the website with the game downloadable. 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. Lastly, the game assumes the user has a basic foundation in algebraic topics such as formulas containing addition, subtraction, multiplication, and division.

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:

- 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.

- Leaderboard enhancements, including online leaderboard synchronization or account-based scoring, may be reserved for future versions and are not required for the initial offline leaderboard.

3 Specific Requirements

1. The system shall implement four top-level game states: Main Menu, Gameplay, Pause, and Game Over.
   1.1. The Main Menu shall display the game title, a "Start Game" button, and a "Quit" button.
   1.2. During Gameplay, the system shall allow the player to open the Pause Menu.
   1.3. The Pause Menu shall suspend all enemy activity and turn progression.
   1.4. The Pause Menu shall include the following controls:
      1.4.1. A global audio volume slider.
      1.4.2. A "Main Menu" button.
      1.4.3. A "Quit to Desktop" button.
2. The system shall display three UI regions during gameplay: Hand Area, Equation Area, and Enemy Area.
   2.1. The Hand Area shall display all cards dealt to the player at the start of the turn.
   2.2. The Equation Area shall allow card placement and display the computed value of all placed cards.
   2.3. The Enemy Area shall display all active enemies, their criteria, and remaining turns.
3. The system shall maintain a Deck, a Discard Pile, and a Player Hand.
   3.1. At the beginning of each turn, the system shall deal a fixed number of cards from the Deck to the Player Hand.
   3.2. The system shall support three card types: Number Cards, Operation Cards, and Modifier Cards.
      3.2.1. Number Cards shall contain a whole integer value.
      3.2.2. Number Cards shall not be placed adjacent to another Number Card in the Equation Area.
      3.2.3. Operation Cards shall display a binary operator (+, −, ×, ÷).
      3.2.4. Operation Cards shall not be placed adjacent to another Operation Card.
      3.2.5. Modifier Cards shall represent unary operations (e.g., negate, square).
      3.2.6 Modifier Cards shall only be placed adjacent to a Number Card.
   3.3. At the end of each turn, all cards used in the Equation Area shall be moved to the Discard Pile.
   3.4. When the Deck is empty, the system shall reshuffle the Discard Pile to form a new Deck.
4. The system shall have an Equation Area that contains cards placed by the player.

4.1. The system shall parse the sequence of cards in the Equation Area from left to right to compute an integer value.
4.2. The system shall enforce adjacency rules during card placement.
4.3. The system shall detect invalid expressions (e.g., division by zero) and reject them with an error message.

5. The system shall have enemies.
   5.1. Each enemy shall contain a mathematical criterion that evaluates a computed integer as true or false.
   5.2. An enemy shall be eliminated when the computed value satisfies its criterion.
   5.3. Each enemy shall include a turn limit that decreases at the end of each turn.
   5.4. When an enemy's limit reaches zero, the system shall deal damage to the player.
6. The game shall become more difficult as the player clears more levels.
   6.1. The number of mathematical criteria on enemies shall increase.
   6.2. The difficulty of the mathematical criteria on enemies shall increase.
7. The player shall begin each run with a defined number of health points (HP).
   7.1. When an enemy deals damage, the system shall reduce HP by the enemy's damage value.
   7.2. The system shall end the run when HP reaches zero.
   7.3. At the end of a run, the system shall display the final score.
   7.4. The system shall track a numerical score for the player during gameplay.
   7.5. The system shall award points for eliminating enemies and completing levels.
   7.6. The system shall display the score during gameplay.
   7.7. The system shall store completed run scores in a local 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, and two sequence diagrams. The use case diagram is meant to illustrate user-visible functions and how they interact in the system.
The class diagram describes the structure of the system, 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 states of the game 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.

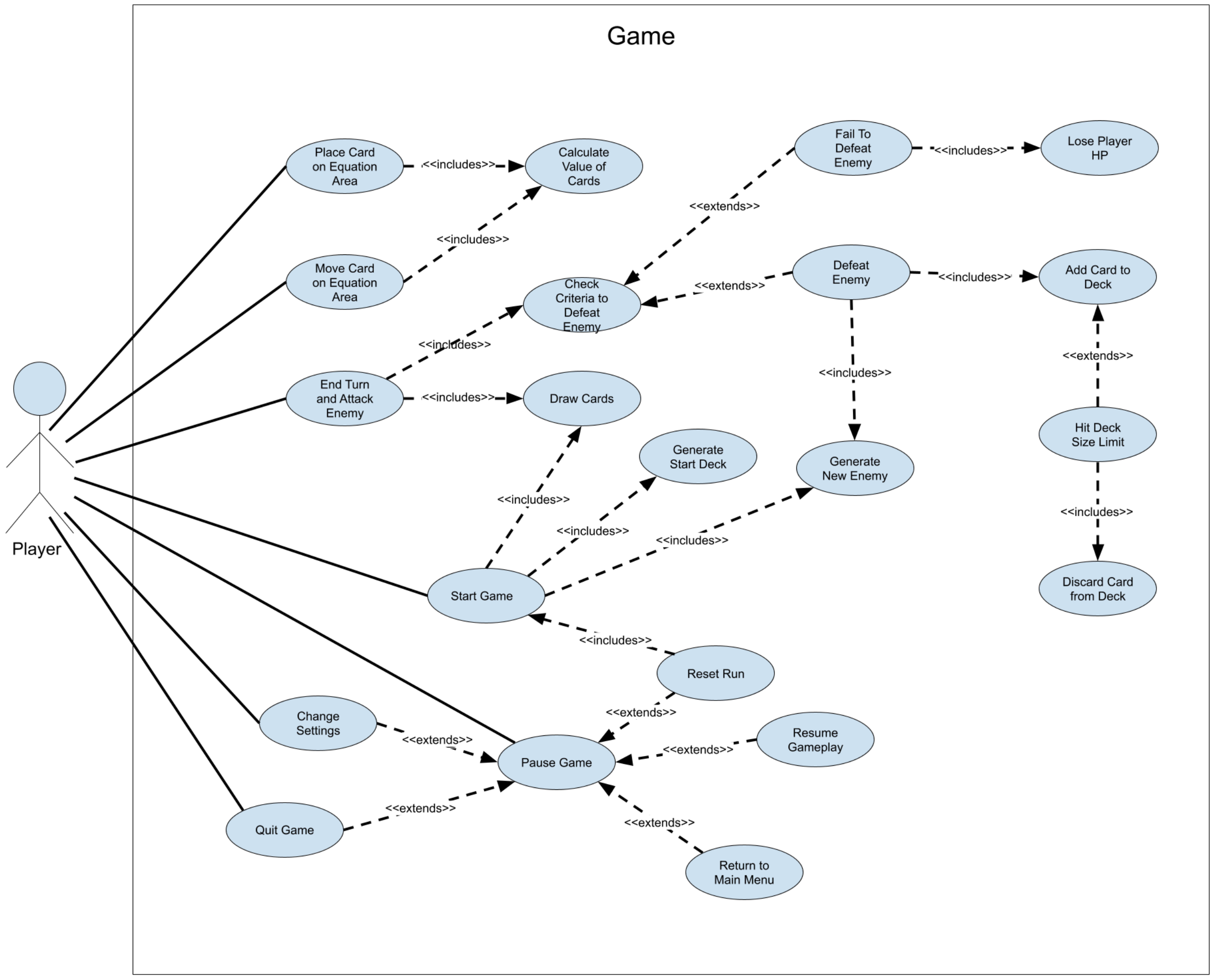


Figure 1: Use Case Diagram for Decrementor

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
Uses 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
Uses 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
Uses Cases:	Start Game, Defeat Enemy

Use Case Name:	**Draw Cards**
Actors:	Player
Description:	The player draws cards at the start of the game and after each turn.
Type:	Secondary and essential
Includes:	None
Extends:	None
Uses 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
Uses 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
Uses 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
Uses Cases:	None

Use Case Name:	**Resume Gameplay**
Actors:	Player

Description:	The player resumes play after pausing the game.
Type:	Primary
Includes:	None
Extends:	Pause Game
Uses Cases:	None

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

Use Case Name:	**Reset Run**
Actors:	Player
Description:	The player resets their run after pausing the game.
Type:	Primary
Includes:	Start Game
Extends:	Pause Game
Uses 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
Uses 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
Uses 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
Uses Cases:	Place Card on Equation Area, Move Card on Equation Area

Use Case Name:	**End turn and Attack Enemy**
Actors:	Player
Description:	The player ends their turn and attacks the enemy.
Type:	Primary and essential
Includes:	Check Criteria to Defeat Enemy, Draw Cards
Extends:	None

Uses Cases:	None

Use Case Name:	**Check Criteria to Defeat Enemy**
Actors:	Player
Description:	The game manager checks if the cards in the equation area meet the criteria to defeat the enemy.
Type:	Secondary and essential
Includes:	None
Extends:	None
Uses Cases:	End Turn and Attack Enemy, Defeat Enemy, Fail to Defeat Enemy

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

Use Case Name:	**Lose Player HP**
Actors:	Player
Description:	The player loses HP.
Type:	Secondary and essential
Includes:	None
Extends:	None
Uses Cases:	Fail to Defeat Enemy

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
Uses Cases:	None

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

Use Case Name:	**Hit Deck Size Limit**
Actors:	Player
Description:	The deck hits the size limit after adding a card.
Type:	Secondary
Includes:	Discard Card from Deck
Extends:	Add Card to Deck
Uses Cases:	None

Use Case Name:	**Discard Card from Deck**
Actors:	Player

Description:	The player chooses a card to remove from their deck.
Type:	Secondary
Includes:	None
Extends:	None
Uses Cases:	Hit Deck Size Limit

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 to one another through arrows labeled with their relationships to other classes.

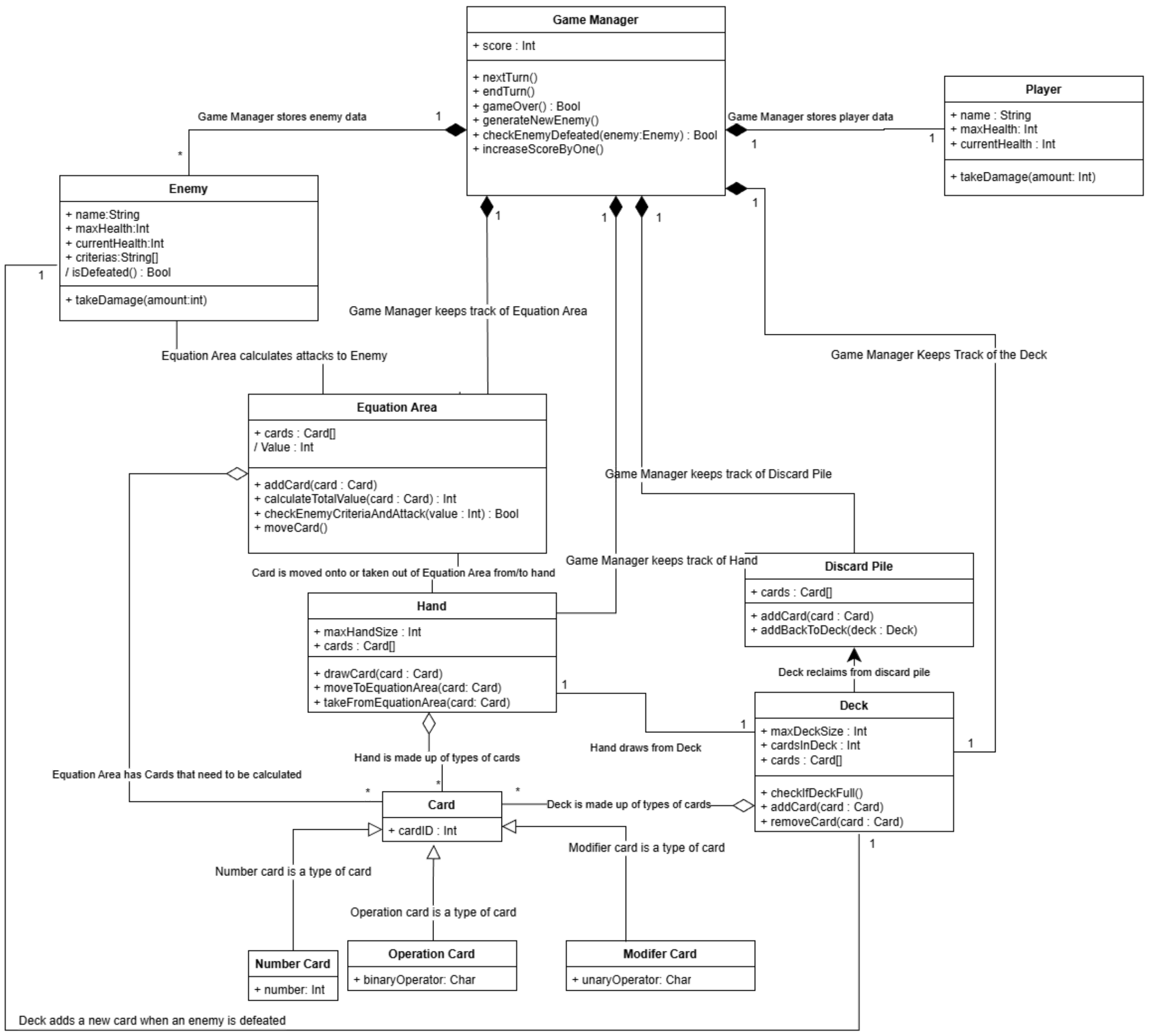

Game Manager
+ score : Int
+ nextTurn()
+ endTurn()
+ gameOver() : Bool
+ generateNewEnemy()
+ checkEnemyDefeated(enemy:Enemy) : Bool
+ increaseScoreByOne()
Player
+ name : String
+ maxHealth: Int
+ currentHealth : Int
+ takeDamage(amount: Int)
Enemy
+ name:String
+ maxHealth:Int
+ currentHealth:Int
+ criterias:String[]
/ isDefeated() : Bool
+ takeDamage(amount:int)
Game Manager stores enemy data
Game Manager stores player data
Game Manager keeps track of Equation Area
Equation Area calculates attacks to Enemy
Game Manager Keeps Track of the Deck
Equation Area
+ cards : Card[]
/ Value : Int
+ addCard(card : Card)
+ calculateTotalValue(card : Card) : Int
+ checkEnemyCriteriaAndAttack(value : Int) : Bool
+ moveCard()
Game Manager keeps track of Discard Pile
Game Manager keeps track of Hand
Card is moved onto or taken out of Equation Area from/to hand
Discard Pile
+ cards : Card[]
+ addCard(card : Card)
+ addBackToDeck(deck : Deck)
Hand
+ maxHandSize : Int
+ cards : Card[]
+ drawCard(card : Card)
+ moveToEquationArea(card: Card)
+ takeFromEquationArea(card: Card)
Deck reclaims from discard pile
Deck
+ maxDeckSize : Int
+ cardsInDeck : Int
+ cards : Card[]
+ checkIfDeckFull()
+ addCard(card : Card)
+ removeCard(card : Card)
Hand draws from Deck
Hand is made up of types of cards
Equation Area has Cards that need to be calculated
Card
+ cardID : Int
Deck is made up of types of cards
Modifier card is a type of card
Number card is a type of card
Operation card is a type of card
Number Card
+ number: Int
Operation Card
+ binaryOperator: Char
Modifer Card
+ unaryOperator: Char
Deck adds a new card when an enemy is defeated
1
*
1
1
1
1
1
1
1
1
1
1
1
1
*
*
*

Figure 2: Class Diagram for Decrementor

Element Name		**Description**
Game Manager		The central game object keeps track of score & turns. Controls level switching and spawning enemies.
Attributes		
	Score : Int	The score 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 keeps track of the Equation area.	
	The Game Manager keeps track of the Hand.	
	The Game Manager keeps track of the Discard Pile.	
	The Game Manager keeps track of the Deck.	
	Game Manager stores player data.	
UML Extensions	None.	

Element Name		**Description**
Player		Representation of the person playing the game.
Attributes		
	name: String	The name of the player, for the 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 need to 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 in order 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 equation area is kept track of by the Game Manager.	
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: Card)	Moves the selected card to the equation 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.	
	Hand is kept track of by the Game Manager.	

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.
	addBackToDeck(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 maximum number of cards the deck can hold.
	cardsInDeck: int	The number of cards in the deck, currently.
	cards : Card[]	The set of cards in the deck.
Operations		
	checkIfDeckFull()	Check if the deck is full.
	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 deck is kept track of by the Game Manager.	
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 was created for this card.
Operations		
	None.	The card has no operations.
Relationships	The Equation Area has cards that need to 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 a binary operation on it.
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. If the number of cards in the deck is greater than the maximum number of cards allowed, the player will be prompted to choose a card to remove from their deck. The player's turn ends after this sequence.

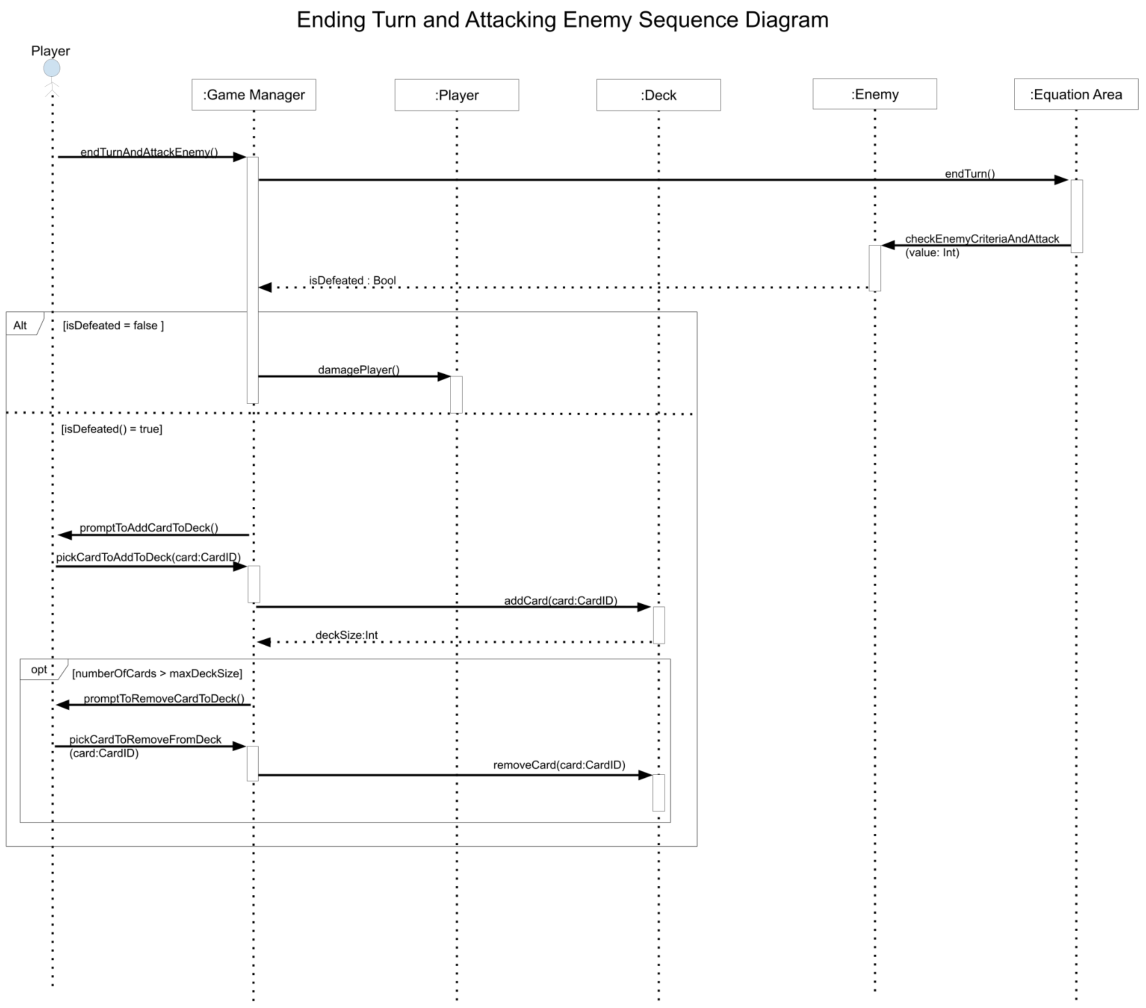


Figure 3: Sequence Diagram 1 for Decrementor

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.

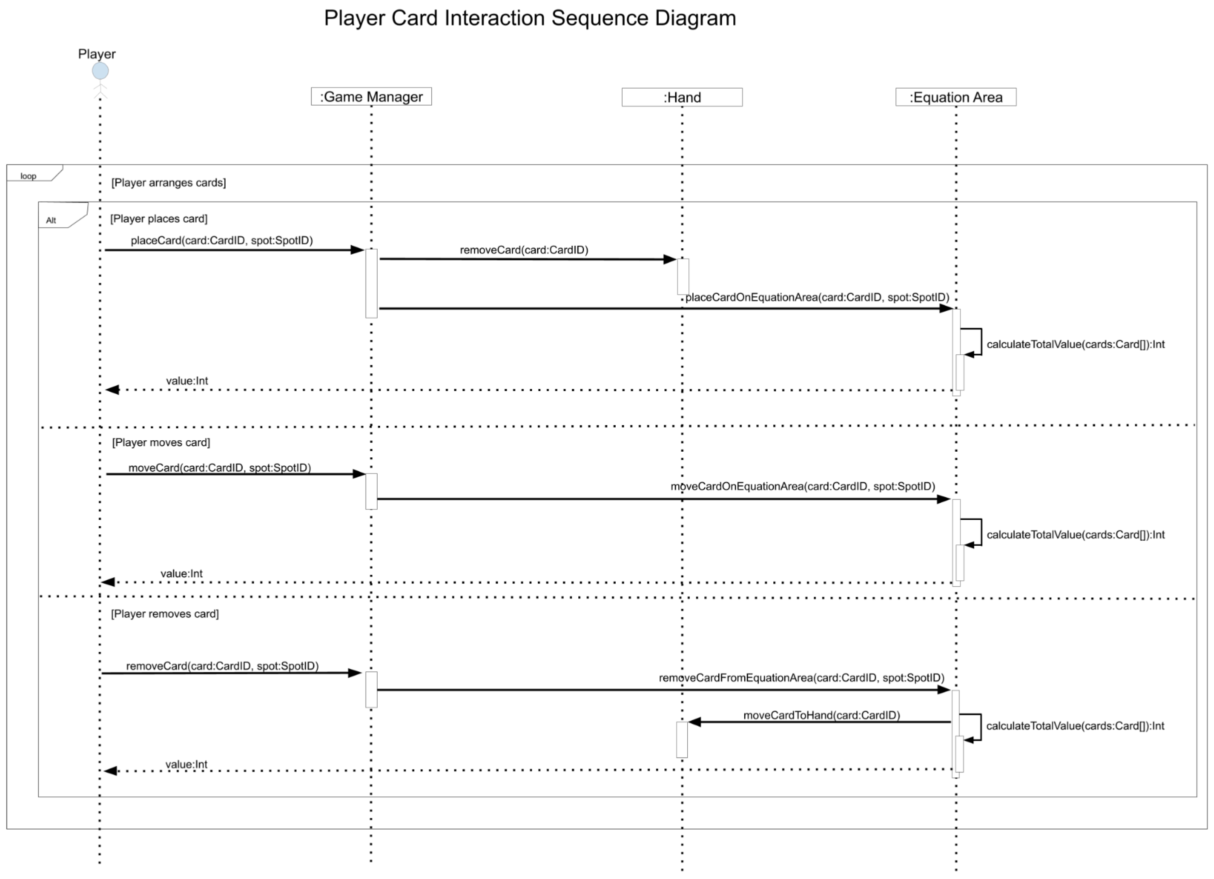

Player Card Interaction Sequence Diagram
Player
:Game Manager
:Hand
:Equation Area
loop
[Player arranges cards]
Alt
[Player places card]
placeCard(card:CardID, spot:SpotID)
removeCard(card:CardID)
placeCardOnEquationArea(card:CardID, spot:SpotID)
calculateTotalValue(cards:Card[]):Int
value:Int
[Player moves card]
moveCard(card:CardID, spot:SpotID)
moveCardOnEquationArea(card:CardID, spot:SpotID)
calculateTotalValue(cards:Card[]):Int
value:Int
[Player removes card]
removeCard(card:CardID, spot:SpotID)
removeCardFromEquationArea(card:CardID, spot:SpotID)
moveCardToHand(card:CardID)
calculateTotalValue(cards:Card[]):Int
value:Int

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 starts, the player will be able to pause the game, giving them access to main menu options and also changing the settings. When the game is unpaused, the player can move cards during their turn however much they want, and they can end their turn. 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.

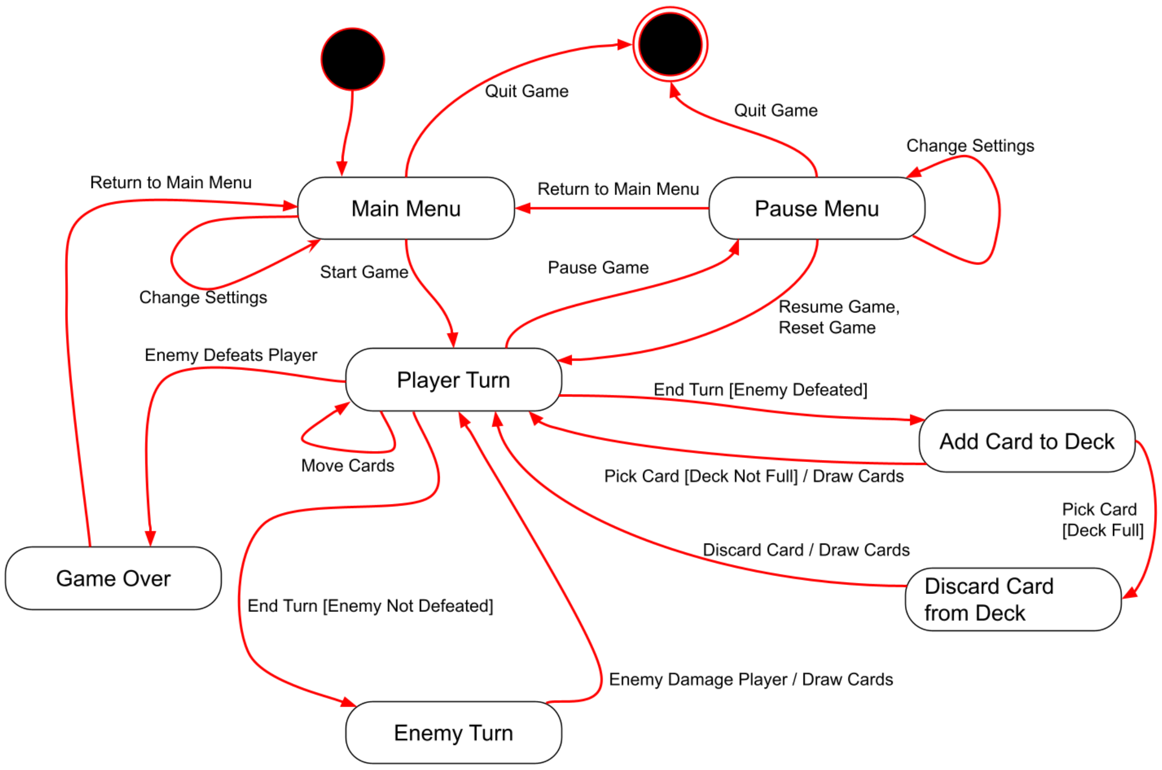


Figure 5: State Diagram for Decrementor

5 Prototype

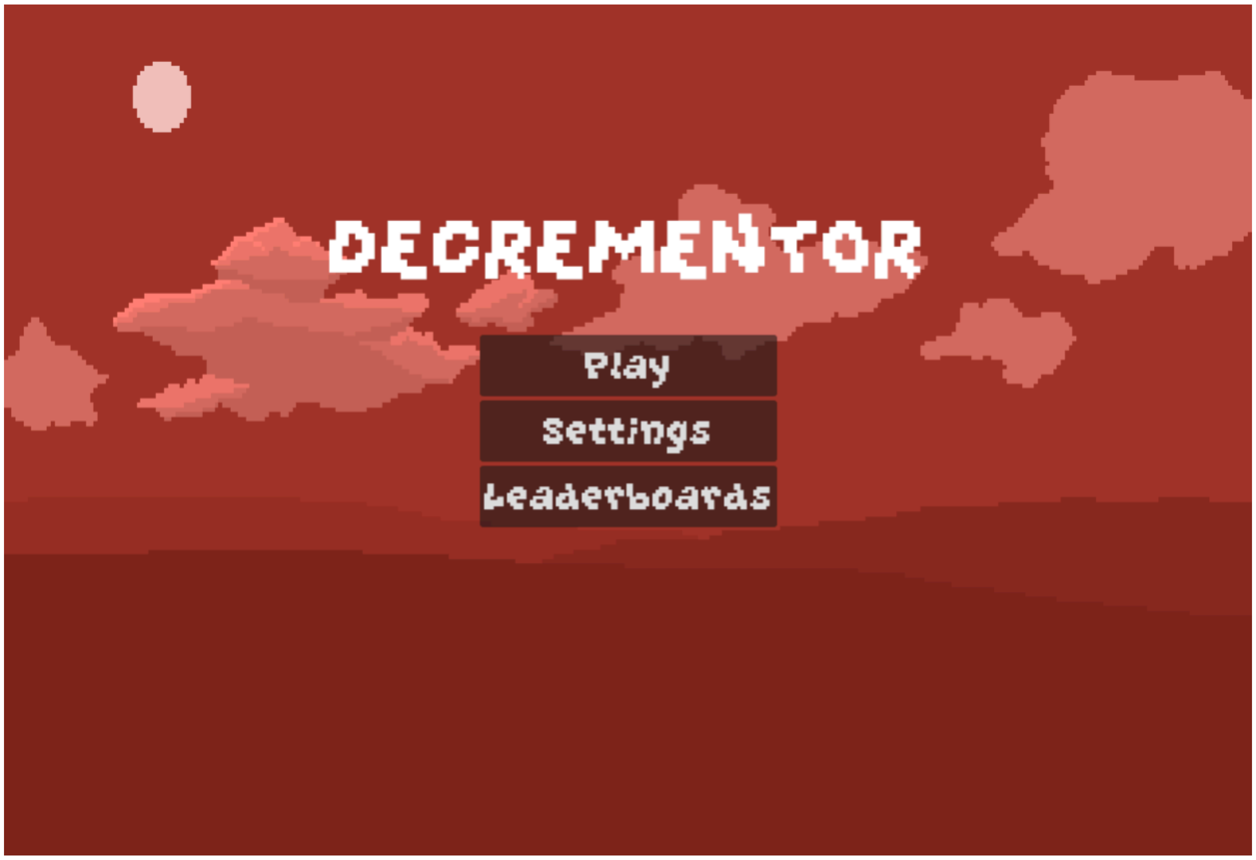
The prototype of the game will feature all functional requirements of the game. This means it will include a main menu, a game scene that advances the levels, and a pause menu on the game scene. It will also include the ability to form equations and defeat enemies.

5.1 How to Run Prototype

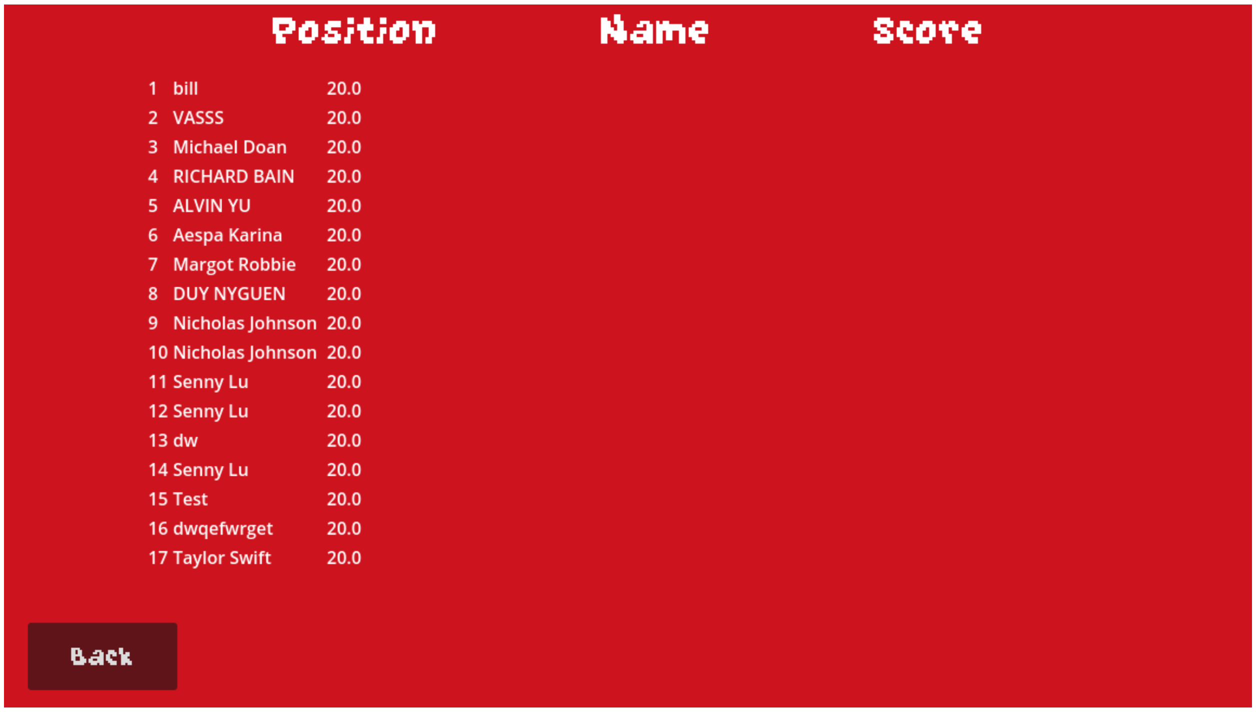
The prototype can be accessed on our website, Decrementor. When on the homepage, click on the prototype tab in the navbar. You should then be able to play the game on the website, no download needed.

5.2 Sample Scenarios

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

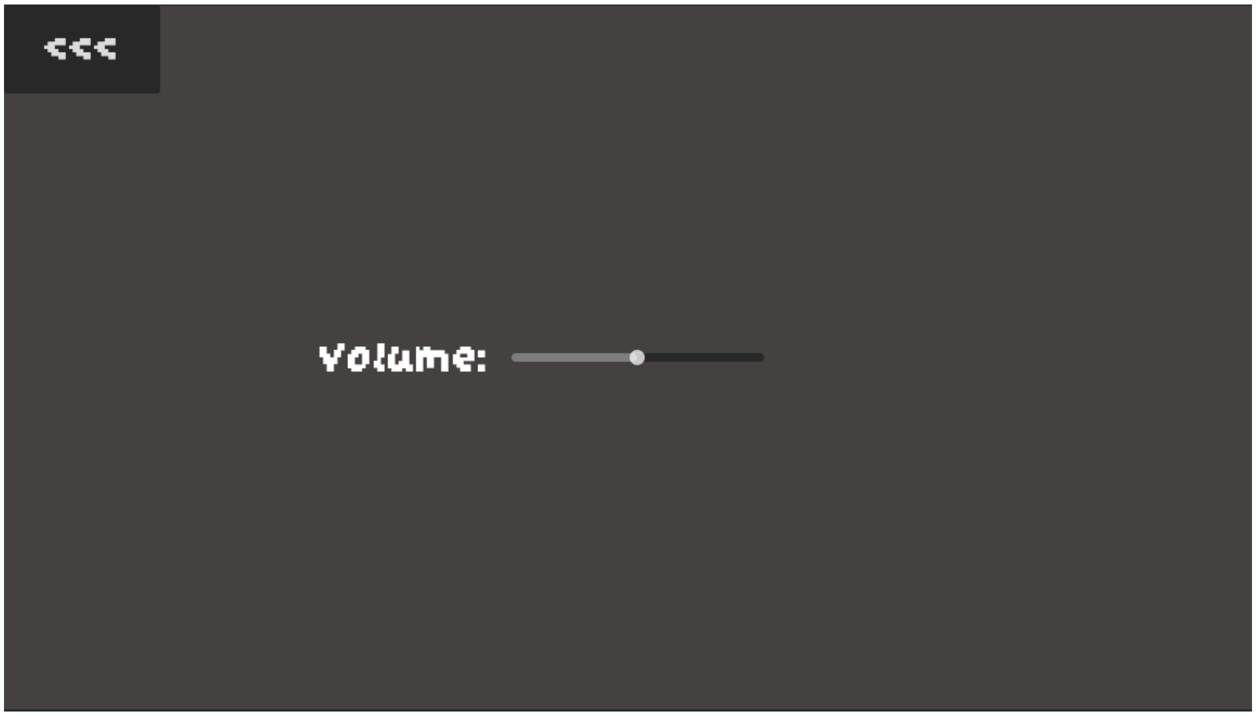


The player can navigate to the leaderboard for scores of past runs.

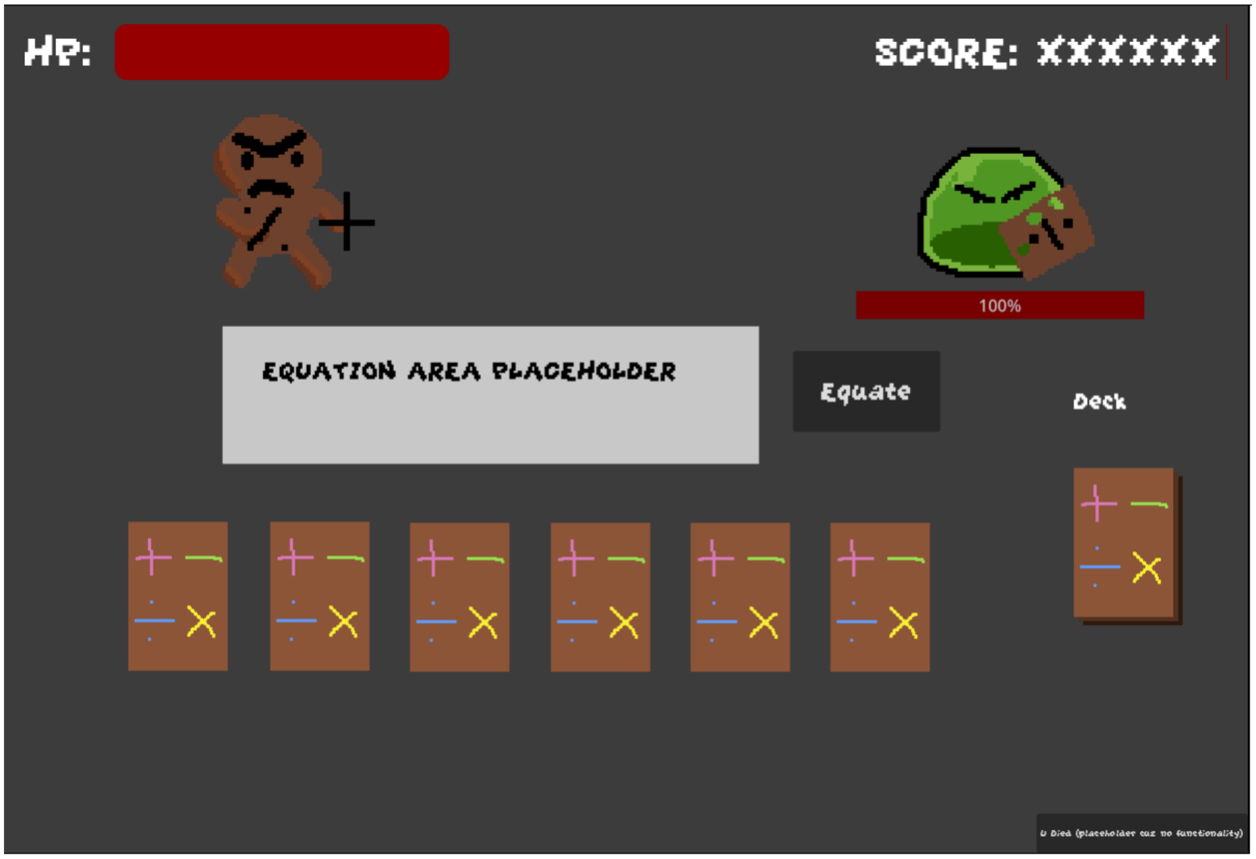


Position	Name	Score
1	bill	20.0
2	VASSS	20.0
3	Michael Doan	20.0
4	RICHARD BAIN	20.0
5	ALVIN YU	20.0
6	Aespa Karina	20.0
7	Margot Robbie	20.0
8	DUY NYGUEN	20.0
9	Nicholas Johnson	20.0
10	Nicholas Johnson	20.0
11	Senny Lu	20.0
12	Senny Lu	20.0
13	dw	20.0
14	Senny Lu	20.0
15	Test	20.0
16	dwqefwrget	20.0
17	Taylor Swift	20.0

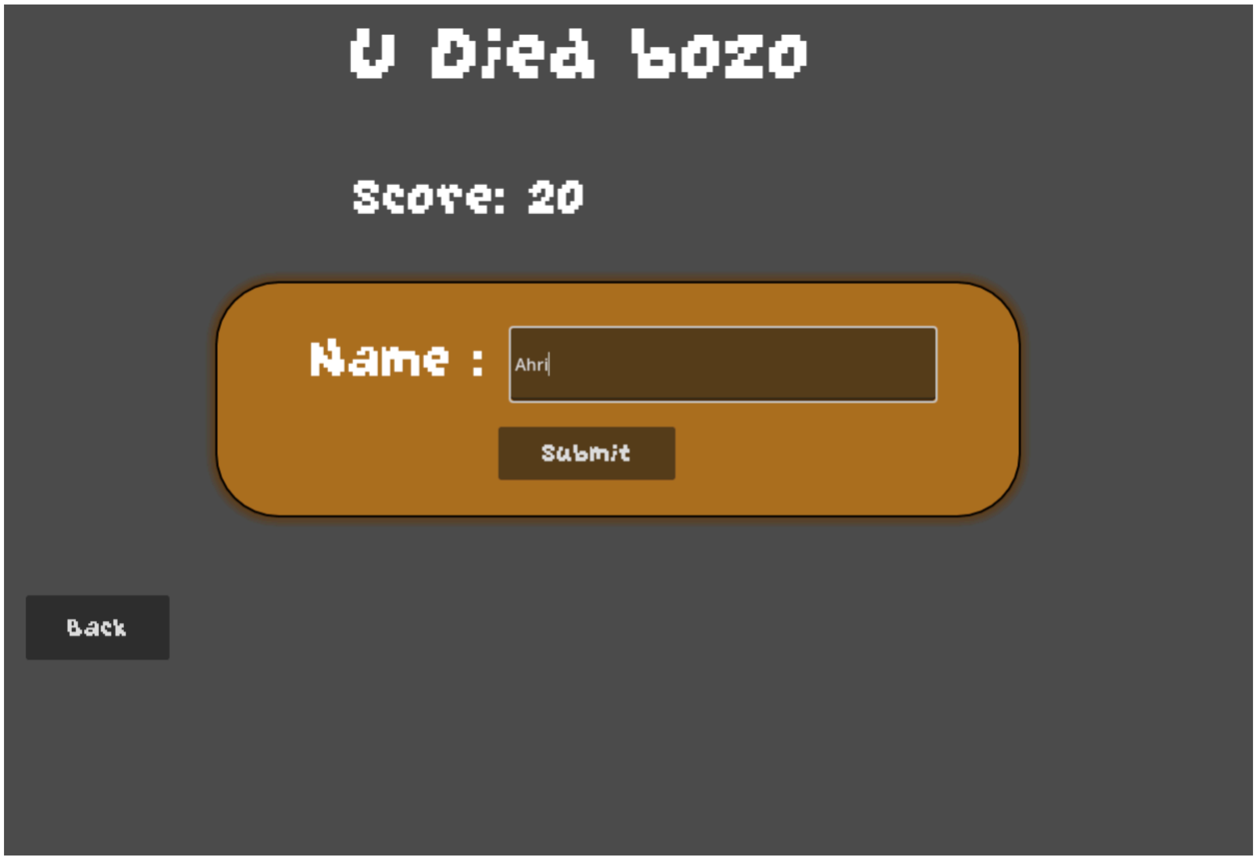
The player can navigate to the settings to change them.



This is a gameplay screen where the player interacts with cards to defeat enemies.


HP:
SCORE: XXXXXX
100%
EQUATION AREA PLACEHOLDER
Equate
Deck
U Died (placeholder cuz no functionality)

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



6 References

Massachusetts Curriculum Framework for Mathematics, Massachusetts Department of Elementary and Secondary Education, 2017, https://www.doe.mass.edu/frameworks/math/2017-06.pdf
GitHub: https://github.com/Software-Engineering-I-Group-1/Equation-Deckbuilder-Roguelike
Website: https://software-engineering-i-group-1.github.io/Equation-Deckbuilder-Roguelike/

7 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.