Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering

Iteration 2 - Project Analysis Report

Group 1D - Slay the Spire

Özge Yaşayan

Fatih Karahan

Gülnihal Koruk

Batuhan Özçömlekçi

Mehmet Bora Kurucu

Instructor: Eray Tüzün

Table of Contents

1. Introduction

Slay the Spire is a single player roguelike deck building game. The game is based on a character moving upwards on a tower while fighting enemies using the cards in their deck. The tower consists of three different acts, each end in a boss fight. Each act has a set of bosses that are different for each act. To reach the act boss, player has to climb the tower by clearing rooms. Each act usually consists of 45-55 rooms. Each room is connected to 1-3 different rooms, and a path of 15 rooms clears the floor, with the 16th room being the act boss. The player cannot move downwards or sideways on the level they are in, therefore the game forces the player to choose its path very carefully.

There are 7 different room types: Monsters, Elites, Bosses, Rest Sites, Stores, Treasures, unknown locations.

Monsters are the most common rooms in the map, they're rooms that have a random enemy that the player has to slay to continue, and each victory ends with a card and gold. Elite rooms are different in the way that the enemies spawn in those rooms are usually much stronger and harder to deal with, and a victory in this room also results in a relic being rewarded. Rest Sites are rooms that the player can rest and heal up, or upgrade a card from their deck. Stores are rooms where the player can buy cards, relics, potions, as well as remove a card from their deck by paying gold earned during the course of their run. Treasures are rooms where the player earns items.

Events are rooms that have something random happening in them. They can range from fighting an enemy, a heal, an upgrade for adding a curse card into their card, etc. There are numerous events that can happen, and not all of them are positive events.

The fights are rather simple, they end when each of the enemies' or the player's health decreases to zero. To achieve this, the player can attack, or block the incoming attacks. The fights happen in a turn based style, with each fight starting with the player. Each turn player has a certain amount of energy, that is by default 3, but can be changed further by different relics, cards or potions, that they use to play cards. Each card costs a predetermined amount of energy to play. For example, the basic Strike card deals 7 damage to a single enemy and costs a single point of energy to play. By default, the player can see the intent of the enemy, and act accordingly. Enemies, like the player, can do multiple things: they can attack, block, buff themselves or steal gold from the player. By the end of each round, the cards that the player has

in their hand gets put into the discard pile and the player draws new cards the next turn from the draw pile. When the draw pile is exhausted, the discard pile is shuffled back into the draw pile.

There are 5 different cards that can be added to the deck: Attack, Skill, Power, Status and Curse. Their results depend on the card description. On top of that, a card can also have a keyword, or an effect. For example, playing a certain card can cause the player to draw a new card, or a card can only be played once and then gets exhausted. Each card can also inflict a condition on the enemy or the player, like increasing their attacks or the damage they take from each attack. These conditions are temporary, and are reset at the end of each combat. Each card can also be upgraded to increase their effects or decrease their costs.

Relics, however are permanent effects that persist over each fight. A relic can give the player another point of energy, but can take the players ability to earn gold. Potions are rather similar to relics in the sense that they persist as well, but potions are consumables that don't have a cost that the player uses during their fights. For example, a block potion gives the player a certain amount of block for the turn it was used, or another potion can increase the energy player has for that turn.

2. Overview

2.1. Game Components

- Player
- Monsters
- Heroes
- Cards
- Relics
- Potions
- Buffs
- Debuffs
- Map

2.2. Player

The game can be played by 1 player at a time. The player can choose what path they are going to take. Every choice they make affects the gameplay and what they will encounter during the game.

2.3. Map

The player progresses the game through a map. A map is randomly generated, and it has various options as to how a player can choose to play the game. The map contains different locations that all have unique properties.

2.4. Items

The game has multiple items such as cards, relics, potions, and so on. The player fights the monsters using these components. Coming up with a good strategy for each character leads to more victories therefore the players have to decide what kind of items they are going to collect in order to complete the game.

3. Functional Requirements

3.1. Characters (Heroes)

The game can be played as 4 characters which will be explained in the following subsections.

3.1.1. Ironclad

Ironclad starts the game with 80 HP, which is the highest among the playable characters. He is known for his powerful attacks, and various defensive options. He starts with the *Burning Blood* relic, which means his HP gets increased by 6 after each combat. [1]

3.1.2. Silent

Silent is a huntress who starts the game with 70 HP which can be considered low. She damages her enemies by using potions, tricks and her agility to avoid their attacks. Her starting relic is *Ring of the Snake* which means that she draws 2 cards at the start of each combat. Her HP does not recover after combats unlike the Ironclad, so she is mostly focused on blocking the attacks. She has many cards that let her gain additional energy and the ability to draw more cards, which help her compensate for the lack of healing after combats. She is unlocked after defeating the Act III boss with the Ironclad [1].

3.1.3. Defect

Defect starts the game with 75 HP. He uses a unique game mechanic called *orb*s. His starting relic is *Cracked Core*, which means that he channels 1 Lightning upon entering a combat. He is unlocked after defeating the Act III boss with the Silent [1].

- **Orbs**

Defect starts with 3 orb slots, which can be increased or decreased by certain cards and relics. The maximum number of slots that a character can have is 10. Orb slots do not get carried over to the later combats. Orb slots have the capability to store elemental spheres. Spheres have passive effects which get activated each turn. They can also be *evoked* which means instant activation. The rightmost orb is considered to be the active orb. There are 4 types of orbs:

- **Lightning:** Gives damage to enemies (passive: 3 additional damage to a random enemy, evoke: 8 additional damage to a random enemy).
- **Frost:** Blocks incoming attacks to the player (passive: 2 additional block, evoke: 5 additional block).
- **Dark:** In order to increase the evoke effect, compromises Lightning's damage (has no passive effect on combat but the damage piles up by 6 each turn, evoke: deal an extra damage of piled amount to the enemy with the lowest HP).
- **Plasma:** Grants energy to the player (passive: 1 additional energy, evoke: 2 additional energy). [1]

Their effects depend on the *Focus* stat which is unique to the Defect. Focus has an influence on all orbs, except the Plasma Orbs, the same way that Strength influences Attack. It can be increased or decreased by using cards, potions and relics [1]. Channeling an orb means putting that orb in the player's first empty orb slot. If the player does not have any empty slots, the first orb is evoked automatically in order to make space [1].

3.1.4. Watcher

Watcher starts the game with 72 HP. She is a monk and she uses the unique game mechanic called *Stance*s. She uses scrying and retaining as well as card generation which is a unique strategy to her. Her starting relic is called *Pure Water*, which means that a *Miracle* is added to her hand upon starting a combat. She is unlocked after defeating the Act III boss with the Defect. [1]

- **Stances**

There are four stances:

- **Calm:** Gain 2 energy when leaving this stance.
- **Wrath:** While in this stance, receive and deal x2 damage.
- **Divinity:** Gain 3 energy while entering this stance and exit the stance at the end of your turn.
- **Empty:** The state of not being in any of the stances.

If the stance is changed, the player will be considered as exiting the old stance and entering the new stance. Stances can be entered by playing various cards or by consuming Mantra, depending on the stance. Other characters can also enter/exit stances, if they have the means to do it [1].

3.2. Cards

Cards are what a character uses to battle in fights. A card can be one of five types, with each card having a color and a rarity.

3.2.1. Card Types

- **Attack**

Attack cards are cards that deal direct damage to the enemy, with a possible effect. The Effect can be a buff, a debuff, a card draw, or a combo based on the previous played card.

- **Skill**

Skill cards are cards that do not deal damage directly. Effects a skill card can have is varied. It can vary from increasing the block of the character, adding or removing buffs and/or debuffs, drawing cards... A skill card can also be defined as every playable card that is not an Attack or a Power card.

- **Power**

Power cards are cards that add a permanent effect during the fight they were played. The effect can be a flat increase in one of the stats, a complex effect based on some conditions to be met, or a constant card that has an effect every turn. Power cards can only be played once, afterwards, they’re removed for the duration of the fight.

- **Status**

Status cards are unplayable cards that are added during the fight and removed once the fight ends. Their main purpose is to fill the deck of the character so during their turn, the cards they draw will not have as many usable cards as usual. They can also have varying effects like dealing damage at the end of the turn or decreasing the energy available to the character.

- **Curse**

Curse cards are cards added outside a fight. They can be added via relics, events, or game conditions. Most curse cards have a complex effect based on their description. Most of the curse cards can be removed, but some of them can't, based on its description.

3.2.2. Card Rarity

Every card, regardless of its type, has a rarity associated with it. The cards rarity can be Common, Uncommon or Rare. The rarity of a card only affects the chance that card can come up in after fight card choice.

- **Extra Rarity**

Some cards also have an extra rarity associated with them.

- **Basic:** Basic cards are cards the character starts the game with.
- **Special:** Special cards are cards that can't be obtained through fight rewards, only events.

3.2.3. Card Colors

Every card, regardless of its type, has a color associated with it. The card color can be the color of the character, Ironclad, Silent, Defect, Watcher, or it can be Colorless. The color of a card determines whether it can come up during that character's fight rewards, a character cannot have a card that is not its color in the fight rewards. Colorless cards can be obtained through other cards, events and shops.

3.2.4. Card Upgrades

Each card, unless specified by the card description, can only be upgraded once. An upgrade changes the numbers on the card without affecting its core mechanics. Usually, the upgrades come in the form of cost decrease, attack or block increase as well as the number of buffs/debuffs that the card applies. It can also change the keyword the card has.

3.2.5. Card Keywords

Card keywords are effects that change the way that card is played.

- **Unplayable:** An unplayable card can not be played in any way, unless character also has a specific.

- **Exhaust:** Exhaust means that the card can only be played once during the fight. After a card with exhaust is played, they are added to the exhaust pile, rather than the discard pile.
- **Ethereal:** Ethereal means that if that card is in the character's hand at the end of their turn, i.e. that card has not been discarded or played, that card will be added to the exhaust pile, and will not come up during that fight.
- **Innate:** Innate means that the card will always be in the hand of the character at the beginning of a fight.
- **Retain:** Retain means that the card will not be discarded at the end of the turn with the rest of the character's hand.

3.3. Map Locations

There are 7 types of map locations in the game:

- **Unknown Location:** When the character visits this unknown location, they will encounter an Event, Enemy, Shop or the Treasure room, each with different probabilities.
- **Shop:** A place where the player can buy potions, relics, cards as well as get a card removed from their deck in exchange for gold.
- **Treasure:** The player can obtain non-boss treasures in this room, such as 1 random relic and gold.
- **Rest Sites:** A place where the player can upgrade one of their cards or heal 30% of their maximum HP. The player always visits a rest site before facing the boss. [1]
- **Enemies:** The location where basic Monster (non-Elite) fights take place.
- **Elites:** The location where fights against Elites take place, resulting in a Relic reward in addition to normal rewards.
- **Boss Fight:** At the end of each map, there is a boss fight. The player can choose to be rewarded with one of 3 rare card options, and one of 3 Boss Relic options. Defeating a boss will recover the player's HP to its maximum level. [1]

3.4. Game Mechanics

3.4.1. Gold

Gold is a resource in the game, that can be used for purchasing various items and getting a card removed from the player's deck. Gold can be earned by defeating opponents as a reward, also by certain relics and cards, as well as some events. [1]

3.4.2. Gameplay

- **Neow the Whale**

When the player first starts playing the game, a whale called Neow will have 4 options for them to choose. The options will change based on whether the player has encountered any of the bosses before. If the player has encountered a boss before, the first 2 options will be something like a common card, relic, potion, gold, a card upgrade, the third option will be a trade-off, such as giving up gold in order to obtain rare relics. The fourth option will require giving up their starting relic in order to obtain a random boss relic. If the player has not encountered a boss, the first option will be adding 10% to their max HP, and the second one will be obtaining the *Neow's Lament* relic. [1]

- **Navigating through the map**

After the encounter with Neow, the player is located on the map of Act I. There are options to choose from on the map, and each choice affects what kind of rooms the player is going to interact with. There is a different map for every Act, and the maps are randomly generated so that they are different for each run. [1]

- **Building a deck**

Every player starts with a basic deck that consists of certain cards. The player adds more cards to their deck through completing events, fighting monsters, and collecting items. The player's goal is to refine the quality of the deck that they own by acquiring new cards, upgrading existing cards, removing cards that are not useful, and so on [1]. Having a better deck improves the player's performance in the game.

- **Combats**

The fights in the game are called combats. Combats are performed as turn-based card games. Players play with the cards from their deck that they build. Each turn, the players draw a default hand which consists of 5 cards, and they have 3 energy. In order to play a card, its energy cost must be paid by the player. What remains at the hand of the player is discarded at the end of each turn. The enemies play after the player. When the draw pile is empty, the discard pile is shuffled back into the draw pile. When the player finishes a combat successfully, they get rewarded with 1 of the 3 cards that are offered to them, as well as some gold, occasionally a potion too. Regular opponents are Monsters, however some opponents are harder to beat, also known as the Elites. When the player beats an Elite, they get rewarded with a relic in addition to

the regular rewards. There are also Bosses who are the hardest to beat and appear at the end of every Act [1]. Monsters show their intentions with a symbol on top of their head, which gives information about their next move to the player, their intention could be attack, block etc. with a value which signifies the intensity of their action as well, if the monster has the intention to attack and how many times they are going to attack.

- **Events**

While going upwards the Spire, the player encounters various non-combat events. Some of these events include offering free items such as gold, relics, card upgrades to the player, however, some of the events have a trade-off, making the player give up certain amounts of their items in order to obtain a strong bonus. Some of the events are strictly disadvantageous for the player, and damage them in many ways such as decreasing their maximum HP. [1]

3.4.3. Relics

Relics are items that are permanent and their bonuses last until the end of the run. Except for some relics, all relics are available to every character. Each character starts the run with a specific relic. There are different types of relics:

- **Starter:** The relics that the characters can start with.
- **Common:** Relics with weak effects that are commonly found.
- **Uncommon:** Relics with strong effects that are harder to find.
- **Rare:** Relics with powerful effects that are unique, and rarely found.
- **Boss:** Relics that can only be obtained through Boss chests.
- **Event:** Relics that can only be rewarded after events.
- **Shop:** Relics that can only be bought from the shop.
- **Special:** Relics that appear only in the case where the player has all of the other relics. [1]

3.4.4. Potions

Potions can be used only once during the combat, with some exceptions. Using a potion does not cost any energy. A player normally can hold 3 potions at the same time, however owning a *Potion Belt* increases this number to 5. Potions have different effects and can be classified by their rarity as common, uncommon and rare. Potions are occasionally obtained after combats or events, however they can also be purchased through the shop. [1]

3.4.5. Buffs

Positive effects on characters in the game are called buffs. These effects can be given to the player by potions, relics and cards. Enemies can also have buffs applied on them, sometimes upon starting the combat, or through relics and their moves. Each buff is removed at the end of the combat, hence their effects do not carry onto the following combats. Some buffs can be stacked according to their duration, increasing the number of turns that the buff will last for, or by their intensity, increasing the power of the buff's effect. Some buffs cannot be stacked. Most buffs can be stacked by their intensities. [1]

3.4.6. Debuffs

Debuffs are the negative effects that are applied on the characters by relics, potions and cards. The *Artifact Buff* has the ability to negate the effects of a debuff. Debuffs can be stacked similarly to buffs. Some cannot be stacked. [1]

3.4.7. Chests

Chests can contain relics, potions, gold. The player will always find a chest on the middle floor of an act. The chests can either be a Boss chest or a normal chest. Boss chests can only be obtained after boss fights, and they contain 3 Boss relics, one of which the player can choose to add to their deck, or they can be skipped. Normal chests can be classified as small, medium or large. Depending on their size, they will contain different rewards. They can be found at chest rooms on the map, or during events. [1]

3.4.8. Ascension Mode

Ascension mode is a game mode where different challenges are introduced in order to increase the difficulty of the game. After the player completed all 3 acts with a character, the first Ascension level is unlocked for that character. Completing an Ascension level unlocks the following one. Failing a level does not damage the progress. A player can play any Ascension level that they have unlocked. The negative effects of the levels stack up and carry over to the next levels. [1]

4. Innovations

- Playing with two or more characters can be added to the game so that two distinct draw-discard piles can be created for each character. Moreover, corollary to this kind of gameplay, relics, potions, cards can be enabled to be distributed between the characters by the

player. Some other entities such as money can be shared by these two or more characters in this kind of gameplay. The player can be enabled to choose which character/characters are going to enter the combat. In this kind of gameplay with two or more characters, the game difficulty can be increased.

- An animal companion (pet) might be added to the game to assist the main character in combats. This animal companion can be enabled to be modified utilizing pet-oriented items that can be purchased, looted or gained throughout the game. These pets can be added as default, that is, they can be given at the beginning of the run or they can be obtained (purchased) from the shops by the character's money.
- Optionally, more than one life (chance of trials in a stage) can be given to the players for a run so that the game can be eased for rookie players.
- Timer in the upper right corner can be removed from the game since it is an unnecessary detail calculating the total time of the run.
- There can be full visibility for the cards and relics in the card library and relic collection sections.

5. Non-Functional Requirements

- **Performance**

In the game, we will use some animations and visual effects for killing the enemy, playing the cards etc. We will provide background music for the game and different sound effects during the combat. The game will not require high system performance since we will try to keep simplicity and we will avoid using complex graphics and simulations for better performance. Since the software will be written using Java, it will be cross-platform so that many user can run the game. Additionally, response time will be considered so all user inputs can be acknowledged within at most 1.5 second in the computer system that has Intel(R) Pentium(R) CPU 4415U @2.30 GHz, Windows 10 64-bit, 8 GB RAM, Intel(R) HD Graphics 610. The game will be open within 2 clicks, button will be activated within 1 click. A text file will be used to store game data. Therefore, the database of the game will be local for the simplicity.

- **Reliability**

When the game is closed for any reason, the player will be able to continue to the game from the last node that he is completed. Thus, in case of unexpected termination of the game, the last node that the player completes will be considered as the checkpoint, and the game will save the

player profile and prevent the data loss in that case. The game will be considered reliable if the total number of bugs, or any kind of errors are exactly 0 in all tests. Our team will continuously test and fix the code to achieve reliability.

- **User-Friendly Interface**

Animations, design and graphics will be created as user friendly and UI design will be simple so that the game can be easy to play and the user can figure out the concepts easily.The client-user age will be 10 or more. So, user-friendly will imply good user experience; easy to use and understand. It will be straightforward and has some information about the game for user. In the main menu, there will be play, account settings, new account, statistics, options and compendium in which user may find info about cards. As in all projects, user interface will be under constant change. To verify that it is friendly enough, 10 selected client-users will play the game and deliver us their experiences, they will also rate it. If the average of rate is more than 9/10, user interface will be considered as friendly enough.Else, new changes will be made.

6. System Models

Abstract classes are not ideally indicated in dynamic diagrams. However, since most of the classes that extend the abstract classes also implement the functions of the abstract classes, we preferred to show them in one general case using the abstract classes. This way of solution is **previously discussed and approved.** Also, some concepts explained using the words “character” and “dungeon” etc., instead of AbstractCharacter and AbstractDungeon. These are **intended and do not correspond to inconsistency**.The naming character, dungeon etc. corresponds for the real entity objects that extends the abstract classes. For instance, the word “Character” corresponds to the actual characters in the game, like Ironclad, The Silent, The Defect. AbstractCharacter is the class that all real characters extend.To sum up, “dungeon”,”character”,”room”,”monster”, in the explanations are real entities that extends AbstractCharacter, AbstractRoom, AbstractMonster classes respectively. Same logic applies for all entities.

6.1 Use Case Diagram

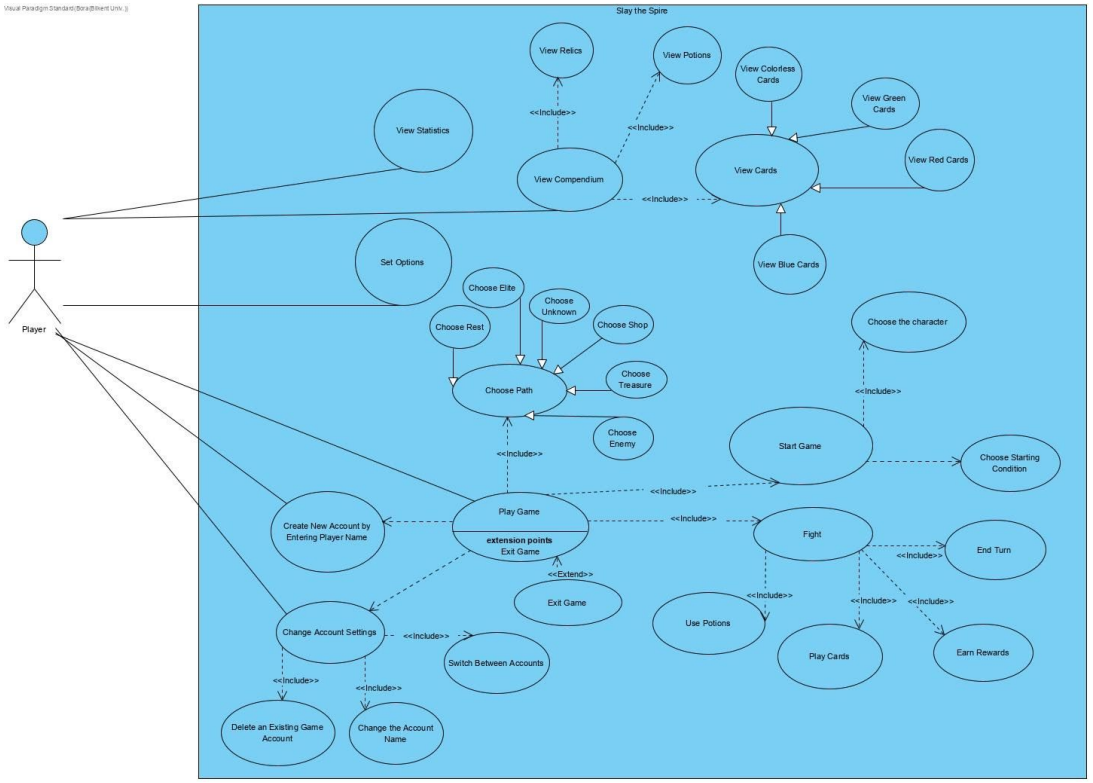

Figure 1: Use Case Diagram of Slay the Spire.

Use Case: `Play Game`

Participating Actor: `Player`

Flow of Events:

1. The player starts game by choosing a character and selecting starting condition
2. The player continuously chooses path and makes decisions to go up the floors
3. The player fights by monsters with his choice of actions

Entry Conditions: `Player` has already created an account by using `Create New Account by Entering Player Name`

`Player` clicks on the Play Game button

Exit Conditions: `Player` is eliminated

`Player` wins by defeating final boss

`Player` saves its current progress and exits from the game

Use Case: Create New Account by Entering Player Name

Participating Actor: Player

Flow of Events: Player creates an account by entering an account name

Entry Conditions: Player clicks on the Create a New Account button

Exit Conditions: A new account has been created

Use Case: Change Account Settings

Participating Actor: Player

Flow of Events:

1. Player selects an existing account
2. Player makes changes in his account

Entry Conditions: Player clicks on the Change Account Settings button

Exit Conditions: Player clicks on the Finish button

Use Case: Delete An Existing Game Account

Participating Actor: Player

Flow of Events: Player deletes an existing account

Entry Conditions:

Player has an existing account

Player uses Change Account Settings

Player clicks on the Delete button

Exit Conditions: The account is deleted

Player clicks on the Finish button

Special Requirements: Player cannot delete the account he/she logged in with

---

Use Case: Change The Account Name

Participating Actor: `Player`

Flow of Events: `Player` enters a new name for the already existing account

Entry Conditions:

`Player` has an existing account

`Player` uses `Change Account Settings`

`Player` clicks on the Change Account Name button

Exit Conditions: The name account is changed

Use Case: `Switch Between Accounts`

Participating Actor: `Player`

Flow of Events: `Player` chooses an account to login

Entry Conditions:

`Player` has two existing accounts

`Player` uses `Change Account Settings`

`Player` clicks on the Switch Account button

Exit Conditions: Current account is changed

Use Case: `View Statistics`

Participating Actor: `Player`

Flow of Events: `Player` views the recorded achievements and stats of his/her account

Entry Conditions: `Player` clicks on the View Statistics button

Exit Conditions: `Player` clicks on the Finish button

Use Case: `Set Options`

Participating Actor: `Player`

Flow of Events: `Player` creates new settings for the game

Entry Conditions: `Player` clicks on the Set Options button

Exit Conditions: Player clicks on the Finish button

Use Case: View Compendium

Participating Actor: Player

Flow of Events: Player selects to view relics, potions, cards in the compendium

Entry Conditions: Player clicks on the View Compendium button

Exit Conditions: Player clicks on the Finish button

Use Case: View Cards

Participating Actor: Player

Flow of Events: Player views his cards on the compendium

Entry Conditions:

Player uses View Compendium

Player clicks on the View Cards button

Exit Conditions: Player clicks on the Finish button

Use Case: View Colorless Cards

Participating Actor: Inherited from View Cards use case

Flow of Events: Player views his colorless cards on the compendium

Entry Conditions: Inherited from View Cards use case

Exit Conditions: Inherited from View Cards use case

Use Case: View Red Cards

Participating Actor: Inherited from View Cards use case

Flow of Events: Player views his red cards on the compendium

Entry Conditions: Inherited from View Cards use case

Exit Conditions: Inherited from View Cards use case

Use Case: View Blue Cards

Participating Actor: Inherited from View Cards use case

Flow of Events: Player views his blue cards on the compendium

Entry Conditions: Inherited from View Cards use case

Exit Conditions: Inherited from View Cards use case

Use Case: View Green Cards

Participating Actor: Inherited from View Cards use case

Flow of Events: Player views his green cards on the compendium

Entry Conditions: Inherited from View Cards use case

Exit Conditions: Inherited from View Cards use case

Use Case: View Relics

Participating Actor: Player

Flow of Events: Player views his relics on the compendium

Entry Conditions:

Player uses View Compendium

Player clicks on the View Relics button

Exit Conditions: Player clicks on the Finish button

Use Case: View Potions

Participating Actor: Player

Flow of Events: Player views his potions on the compendium

Entry Conditions:

Player uses View Compendium

Player clicks on the View Potions button

Exit Conditions: Player clicks on the Finish button

Use Case: Start Game

Participating Actor: Player

Flow of Events:

1. Player starts a new run
2. Player selects a character/characters by Choose the Character
3. Player decides starting conditions by Choose Starting Condition

Entry Conditions: Player uses Play Game

Exit Conditions: Player clicks on the Start Run button

Use Case: Choose the Character

Participating Actor: Player

Flow of Events: Player selects character(s)

Entry Conditions: Player uses Start Game

Exit Conditions: Player clicks on the Select Character button

---

Use Case: Choose Starting Condition

Participating Actor: Player

Flow of Events: Player answers the dialogue and decides the starting condition.

Entry Conditions: Player uses Start Game

Exit Conditions: Player clicks on the Start Run button

Use Case: Choose Path

Participating Actor: Player

Flow of Events:

1. `Player` continuously chooses paths
2. `Player` enters actions through the path
3. `Player` makes decisions according to actions

Entry Conditions:

`Player` uses `Play Game`

`Player` has already started a game by `Start Game`

Exit Conditions: `Player` clicks on the Save & Exit button

Use Case: `Choose Rest`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses a rest node on the map
2. The character(s) of `Player` are healed and thus their HP increased or alternatively the cards of the character(s) are upgraded by blacksmith
3. `Player` continues to choose paths on the map

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Use Case: `Choose Elite`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses an elite node on the map
2. The character(s) managed by `Player` fights against an Elite type of an enemy.
3. According to the result of the fight, `Player` continues to choose paths on the map or the run is ended.

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Special Requirements: If during the fight, `Player` saves and exits, his progress in fight will not be saved

---

Use Case: `Choose Unknown`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses an unknown node on the map
2. Among the actions shop, enemy, elite or treasure, `Player` encounters with a random action or alternatively `Player` encounters with a random event where he/she is going to make choices and take their consequences
3. `Player` continues to choose paths on the map

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Special Requirements: If during the fight, `Player` saves and exits, his progress in fight will not be saved

---

Use Case: `Choose Shop`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses a shop side on the map
2. With the current money, `Player` can purchases items, potions, cards and also can discard the unwanted cards inside the current deck(s)
3. `Player` continues to choose paths on the map

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Use Case: `Choose Treasure`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses a treasure node on the map
2. `Player` opens a treasure chest which may grant potion(s), relic(s), card(s) and money
3. `Player` continues to choose paths on the map

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Use Case: `Choose Enemy`

Participating Actor: Inherited from `Choose Path` use case

Flow of Events:

1. `Player` chooses an enemy node on the map
2. The character(s) managed by `Player` fights against an basic type of an enemy.
3. `Player` continues to choose paths on the map

Entry Conditions: Inherited from `Choose Path` use case

Exit Conditions: Inherited from `Choose Path` use case

Special Requirements: If during the fight, `Player` saves and exits, his progress in fight will not be saved

---

Use Case: `Fight`

Participating Actor: `Player`

Flow of Events:

1. `Player` starts the fight
2. `Player` plays cards and potions to defeat the enemy
3. `Player` ends his/her turn
4. The enemy makes its predetermined action and ends its turn
5. The above sequence in the stages 2-4 continues until one side wins and other side loses

6. If Player is the winner, he/she can get the earn rewards includes money, card(s), relic(s), potion(s); otherwise, the run of Player might be terminated.

7. If Player has won the fight or has achieved to survive, he continues to select his path on the map by Choose Path

Entry Conditions:

Player uses Play Game

Player has already started a game by Start Game

Exit Conditions: Player clicks on the Save & Exit button

Special Requirements: If during the fight, Player saves and exits, his progress in fight will not be saved

---

Use Case: Use Potions

Participating Actor: Player

Flow of Events: Player consumes potions in the fight

Entry Conditions: Player uses Fight

Exit Conditions: Player clicks on the Save & Exit button

Use Case: Play Cards

Participating Actor: Player

Flow of Events: Player play the cards of the characters

Entry Conditions: Player uses Fight

The character(s) must possess enough action points (mana) in that turn as the desired card wants

The character(s) must have drawn the card from draw-pile

Exit Conditions: Player clicks on the Save & Exit button

Use Case: End Turn

Participating Actor: `Player`

Flow of Events: `Player` ends his/her turn after completing his/her actions

Entry Conditions: `Player` uses `Fight`

Exit Conditions: `Player` clicks on the Save & Exit button

Use Case: `Earn Rewards`

Participating Actor: `Player`

Flow of Events:

1. `Player` earns the goods such as money, relic, potion and card after winning the fight
2. `Player` continues to select the path by `Choose Path`

Entry Conditions:

`Player` uses `Fight`

`Player` should win the fight

Exit Conditions: `Player` clicks on the Save & Exit button

Use Case: `Exit` **`extends`** `Play Game`

Participating Actor: `Player`

Flow of Events: `Player` exits from Slay the Spire

Entry Conditions: `Player` clicks on the Exit button

Exit Conditions: `None`

6.2 Dynamic Models

6.2.1 Sequence Diagrams

Scenario 1, Play Game: The game starts with the initialize() method which draws the initial screen and calls load() to load the game to the point where it is left. To load the game, character is loaded, which loads their deck, HP, power(s), gold, and relics. Then, initialize() method of the dungeon is called, which generate() s all the rooms by calling the initialize() method of the rooms. Then, game is ready to be played, while game is played, after each room is done**,** ascend() method of the dungeon is called, which precedes to the next room. When the game is finished, save() method is called automatically to save the progress.(Figure 2).

Scenario 2, Play Room: This scenario describes how the rooms are processed. Dungeon calls ascend() to process to the next room. The room start() s and perform different actions depending on the RoomType. If the RoomType is Fight, then preBattle() method is called to basically prepare the battlefield, including drawing the creatures, setting their lives,powers,etc.Then,until the fight is over, while !done, preTurn(), monsterPreTurn(), turn(), monsterTurn(), postTurn(), monsterPostTurn() functions are called respectively. When the fight is over, if it is win, if currentHp > 0, rewards are earned.If room type is Treasure, rewards are earned. If it Shop, potions,relics and cards can be bought, cards also can be removed. If room type is RestSite, either health gained with rest() or a card is upgrade() d. If roomtype is Event, event is triggered.(Figure 3).

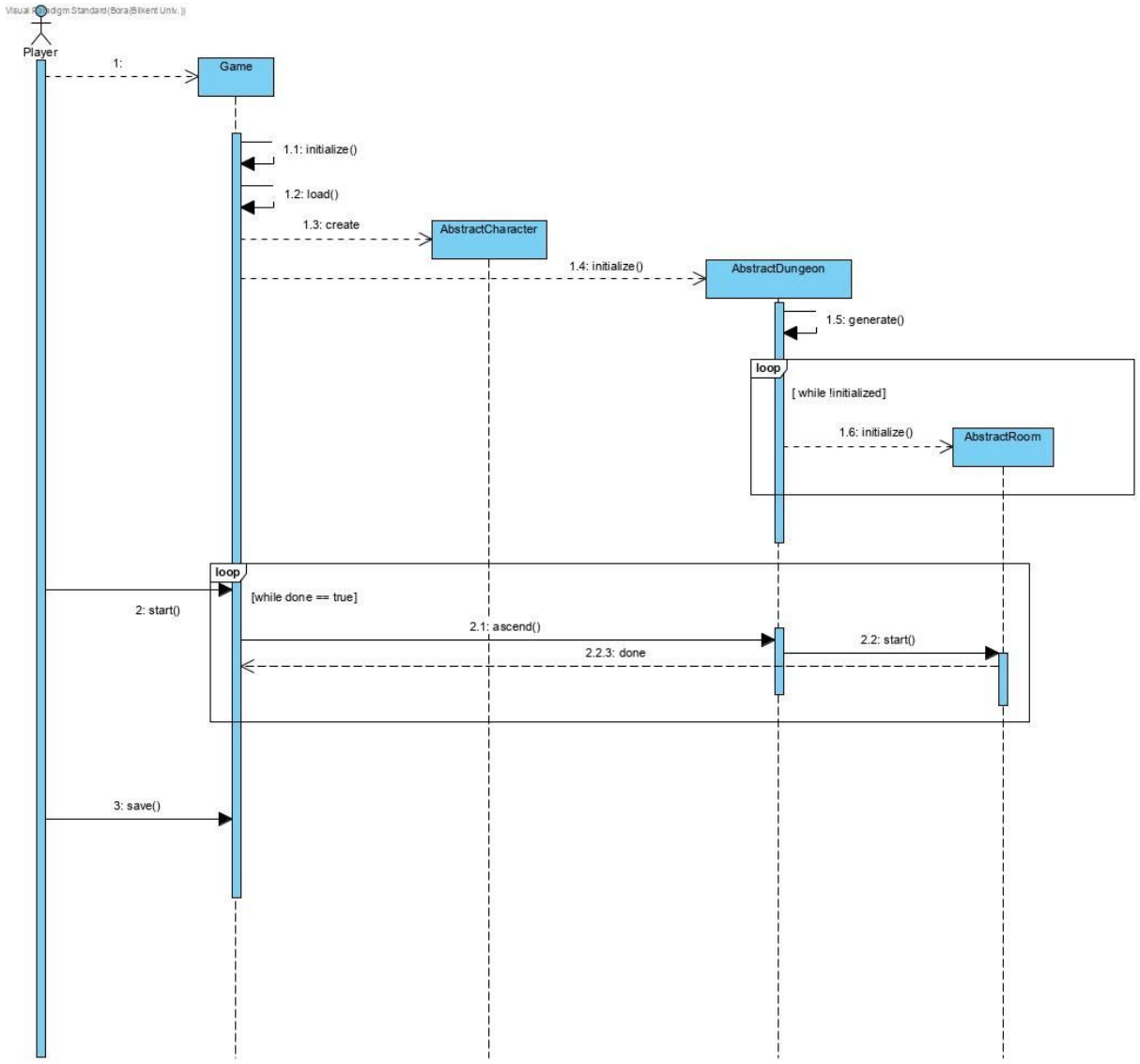

Figure 2: Sequence diagram of the “play game” scenario.

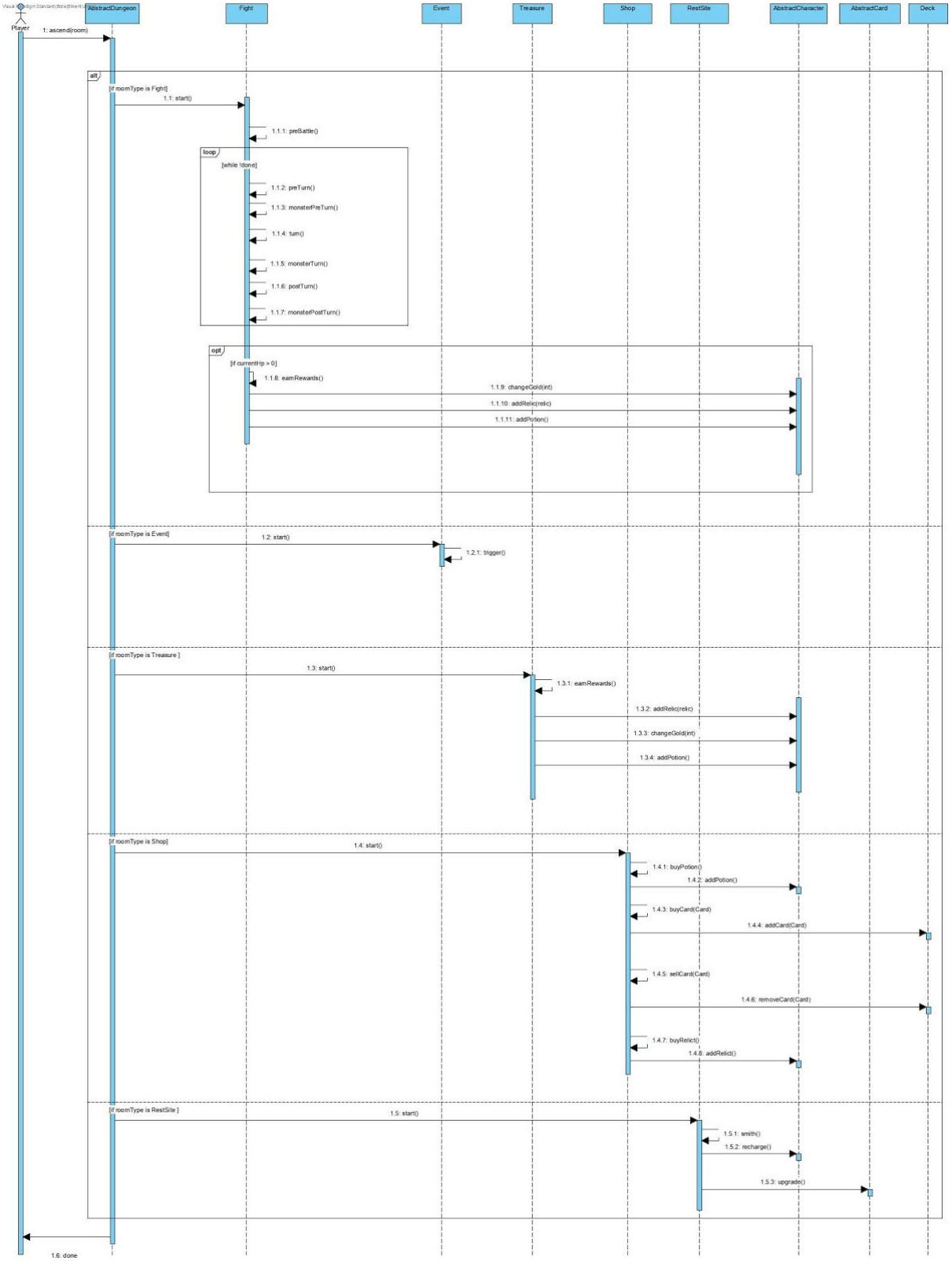

Figure 3: Sequence diagram of the “play room” scenario.

6.2.2 Activity Diagrams

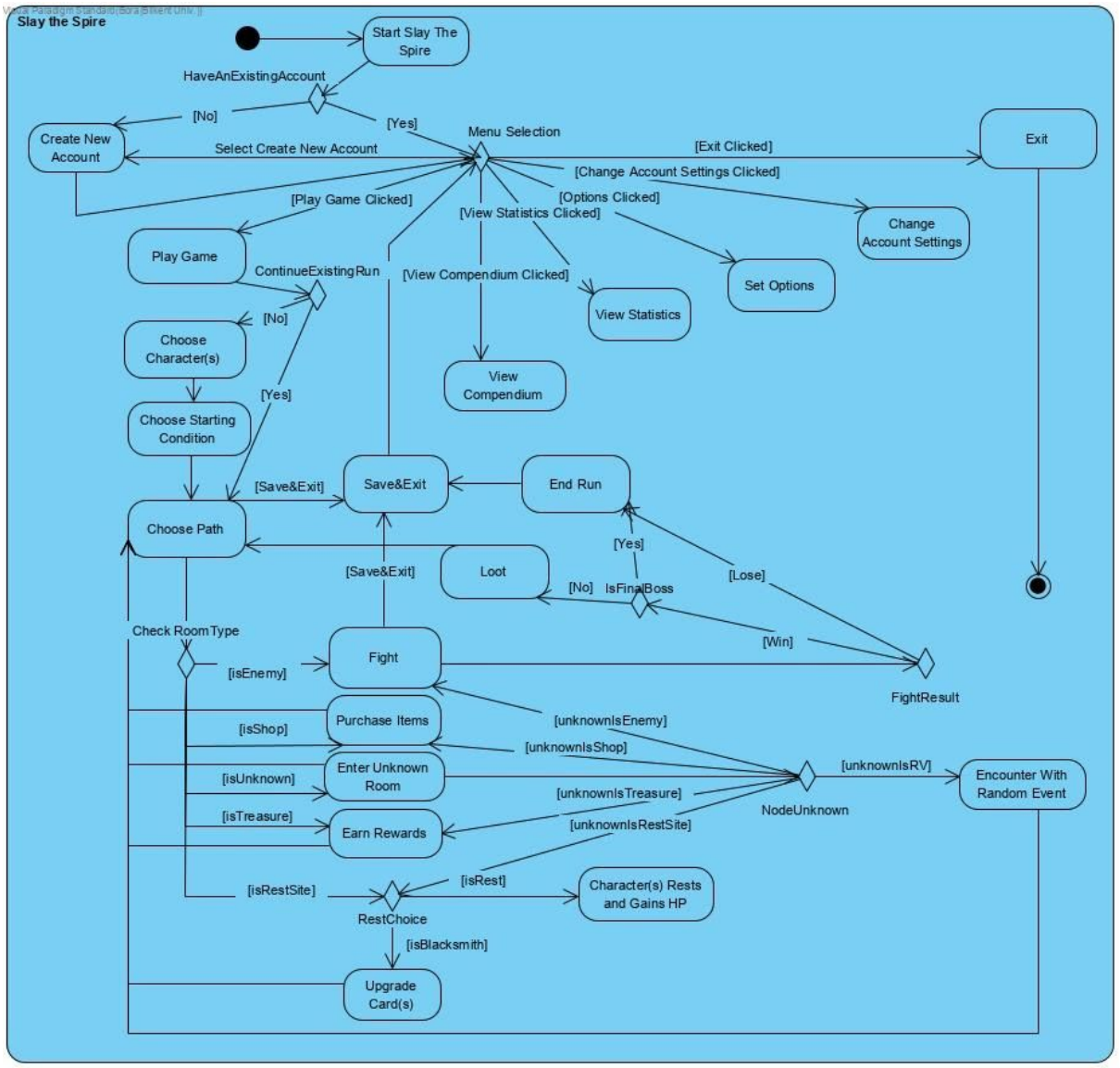

Figure 4: Activity diagram of Slay the Spire.

Slay the Spire as a whole activity starts with creating an account or checking whether there is an existing account or not. After the game ensures that there is an existing account, it directs the player to the main menu. Main menu has several functionalities to create an account, change account settings, view game statistics, set game options, view compendium or most importantly, play the game. Alternatively user can exit from the game by selecting the exit in the main menu.

By selecting play the game, one can start a new run or can continue with an existing saved run. Starting a new run leads to choosing a character and selecting starting conditions of the run.

After the screen showing the map appears, the player is able to choose a path to go up the spire. The nodes on the path can be a fight, a shop where you can purchase an item, a treasure to earn rewards, a rest site that leads the character to rest or the blacksmith to upgrade a card, an unknown node that opens up to any of the latter nodes. Moreover, unknown nodes can lead to random events that will grant the character an improvement, an item or a curse. After completing the node, the game returns the player to the path choice screen and the game continues in this loop until a fight is lost, a boss is defeated in the fight or the player saves the game and exits to the main menu. The player can perform save and exit during the fights yet he/she continues the game from the last finished node on the map when he/she restarts the game.

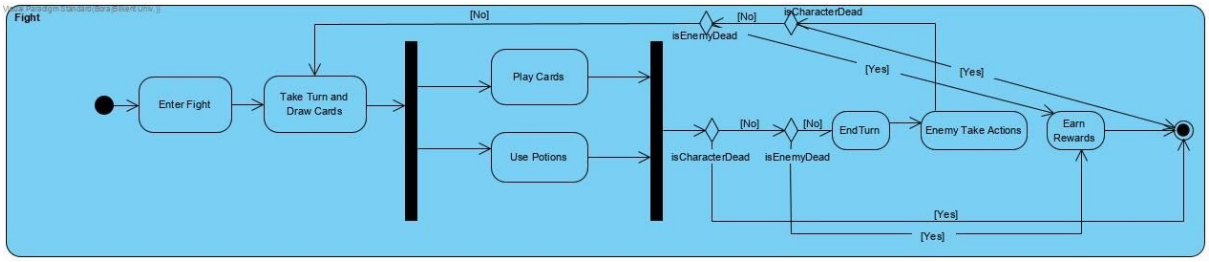

Figure 5: Activity diagram of the fight mechanism.

This activity diagram focuses on how the fight mechanism works on the Slay the Spire. Basically, the fights have a turn based mechanism that the loop continues until one side loses. First turn starts with the character the player control. The character managed by the player uses his potions and plays his cards spontaneously in his turns. After these moves, it is checked that whether one side has lost the game or not. The character managed by us ends his turn and the enemy takes his turn. After the moves of the enemy, it is again checked whether one side has lost or not. This loop of the fight continues until one side is dead and other side wins the game. Additionally, if the character controlled by the player wins the game, he earns rewards after the fight. After the fight activity ends, the game returns to the map or the main menu (if the character is dead).

6.2.3 State Diagram

The state diagram depicts how the game is run inside a room. If the RoomType is Fight,first the fight prepared for only one time in PrepareBattle state.Than, states change to PreTurn,Turn and to PostTurn until the fight is over.If the room is RestSite, user either rest() or smith(), then return to ChooseRoom state again.If the state is Shop, user enters the shop, buys/sells, than needs to

ChooseRoom again.If the room type is Treasure, after earning rewards,a new room is chosen. This process continues until there is no room to choose.

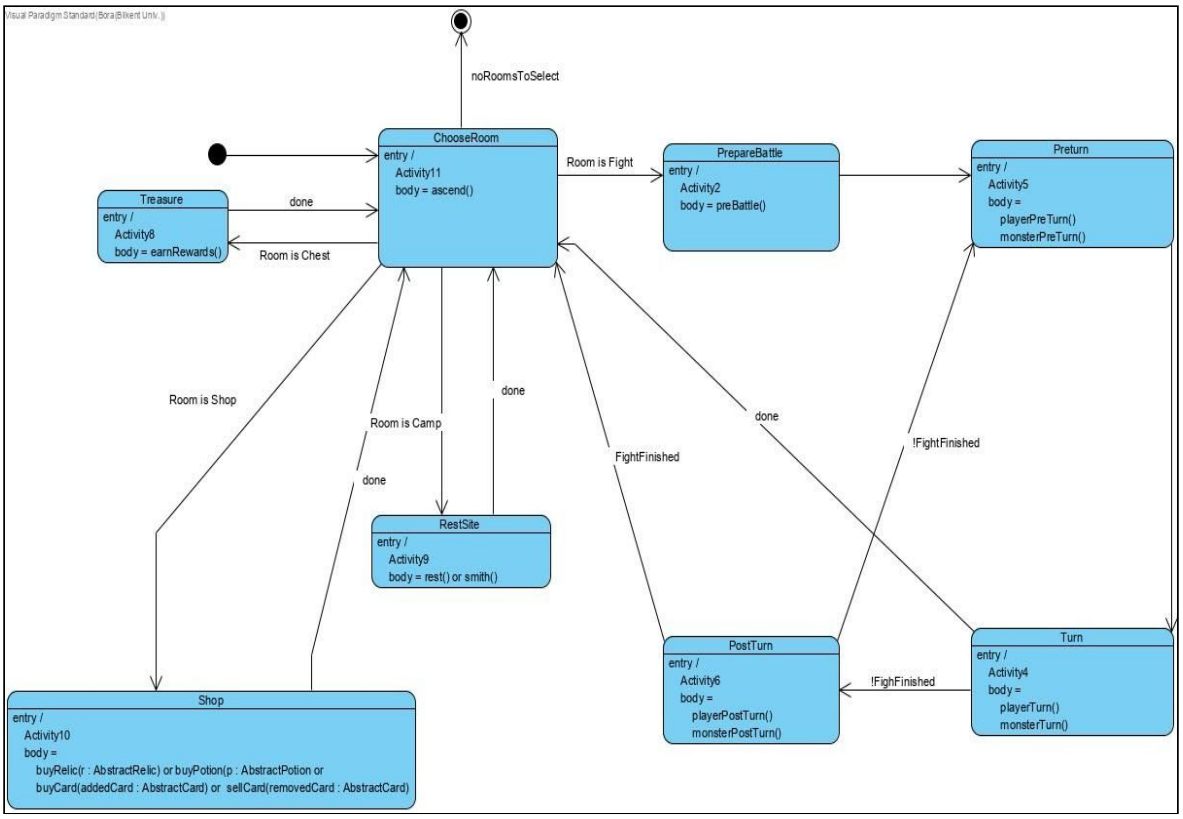

Figure 6: State diagram of Slay the Spire.

6.3 Static Models

6.3.1 Object and Class Model

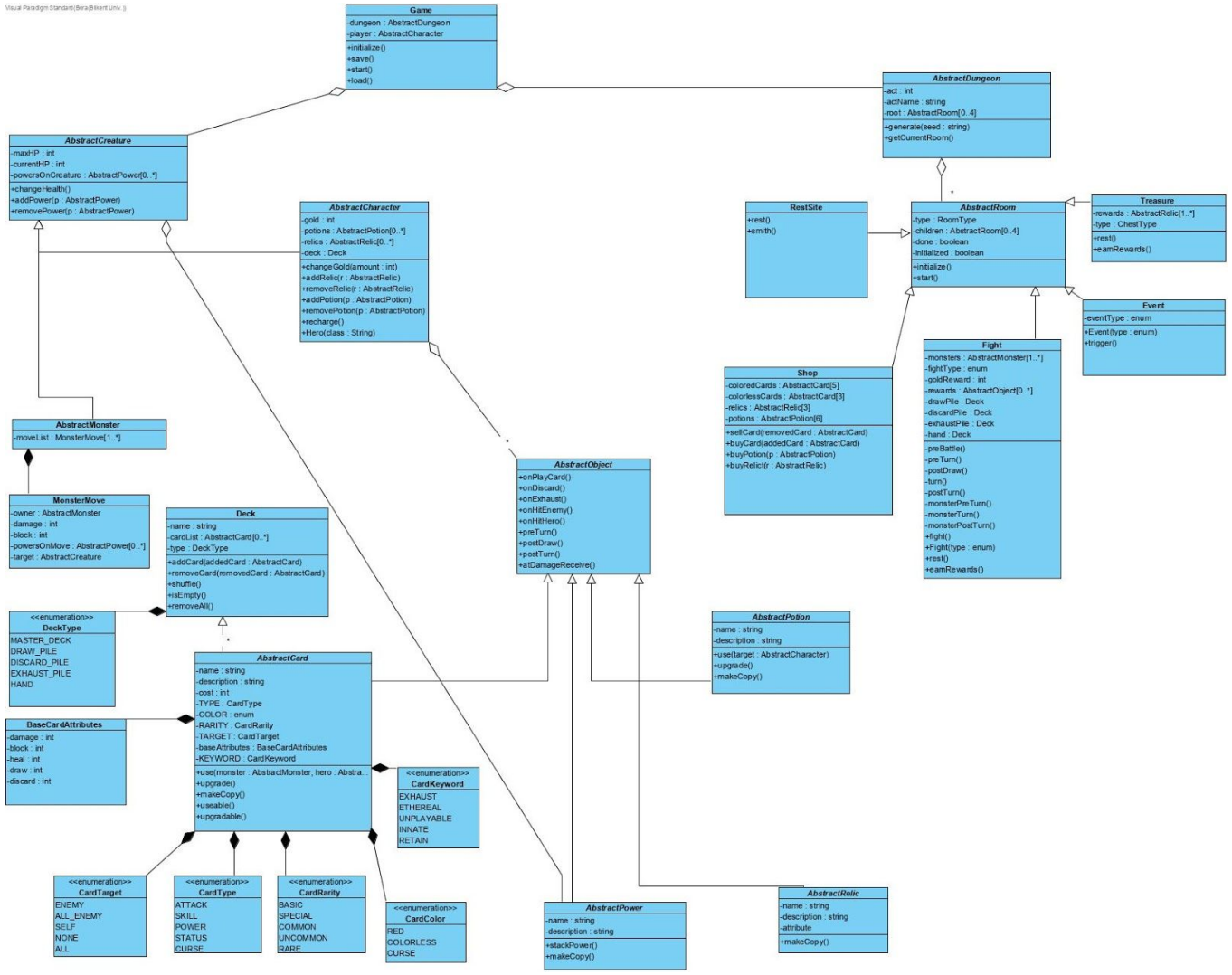

Figure 7: Class diagram of Slay the Spire.

The Game class is what manages the game's functionality, namely initializing the game, loading the game, saving the game and starting the game.

AbstractObject can be an AbstractRelic, AbstractPower, AbstractCard or an AbstractPotion. Each of them have their relevant properties.

Each card inherits from the AbstractCard class. AbstractCard class has properties such as CardColor, CardKeyword, CardType, CardRarity, CardTarget and BaseCardAttributes. They all provide the information necessary to identify a card and its effects.

A collection of cards make up the Deck class which can be of different types, such as draw pile, exhaust pile, master deck, discard pile and the hand of the player. This information is maintained through the DeckType class.

AbstractCreature is either a Monster, or a Character. The information about the moves that a monster is going to make are stored in a class called MonsterMove.

AbstractRoom can be a Fight, Shop, RestSite, Treasure, or an Event.The action of the Event is done when it is triggered().The random production of rooms and the progression on the map is handled by the AbstractDungeon class.

The aforementioned game elements, such as cards, relics, potions etc. will all have their individual classes and they will extend their respective abstract classes.

7. UI Mockups

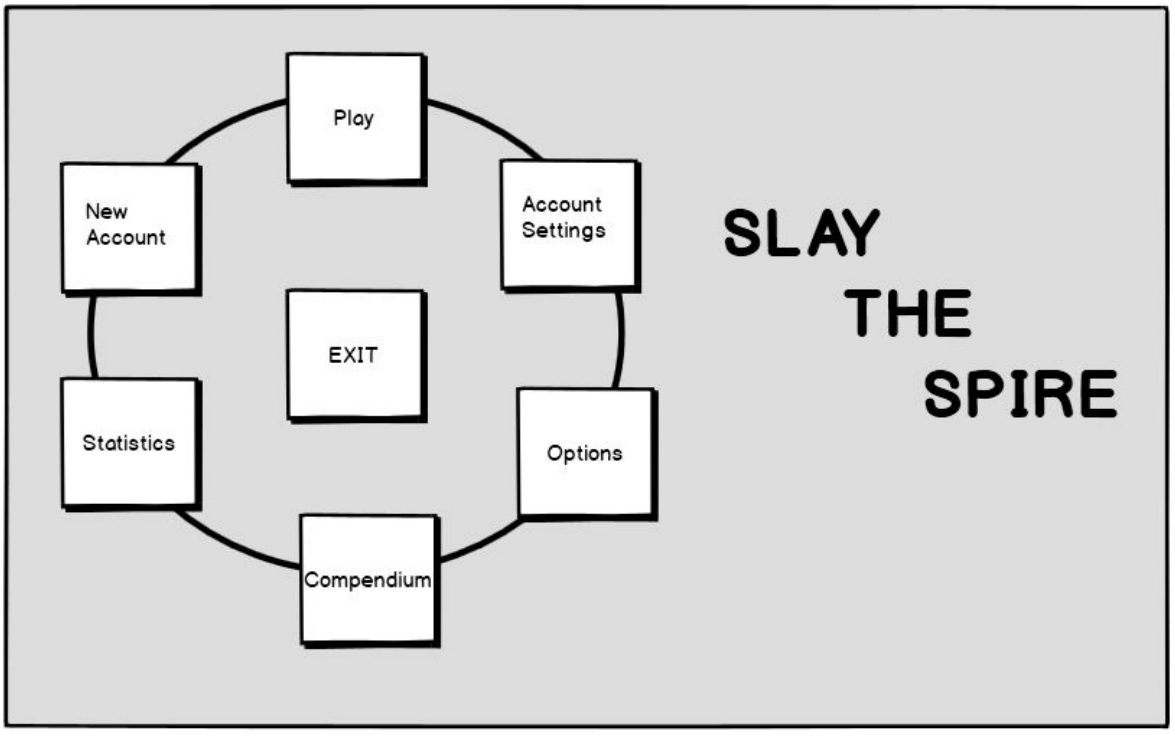

Figure 8: Main screen of the game.

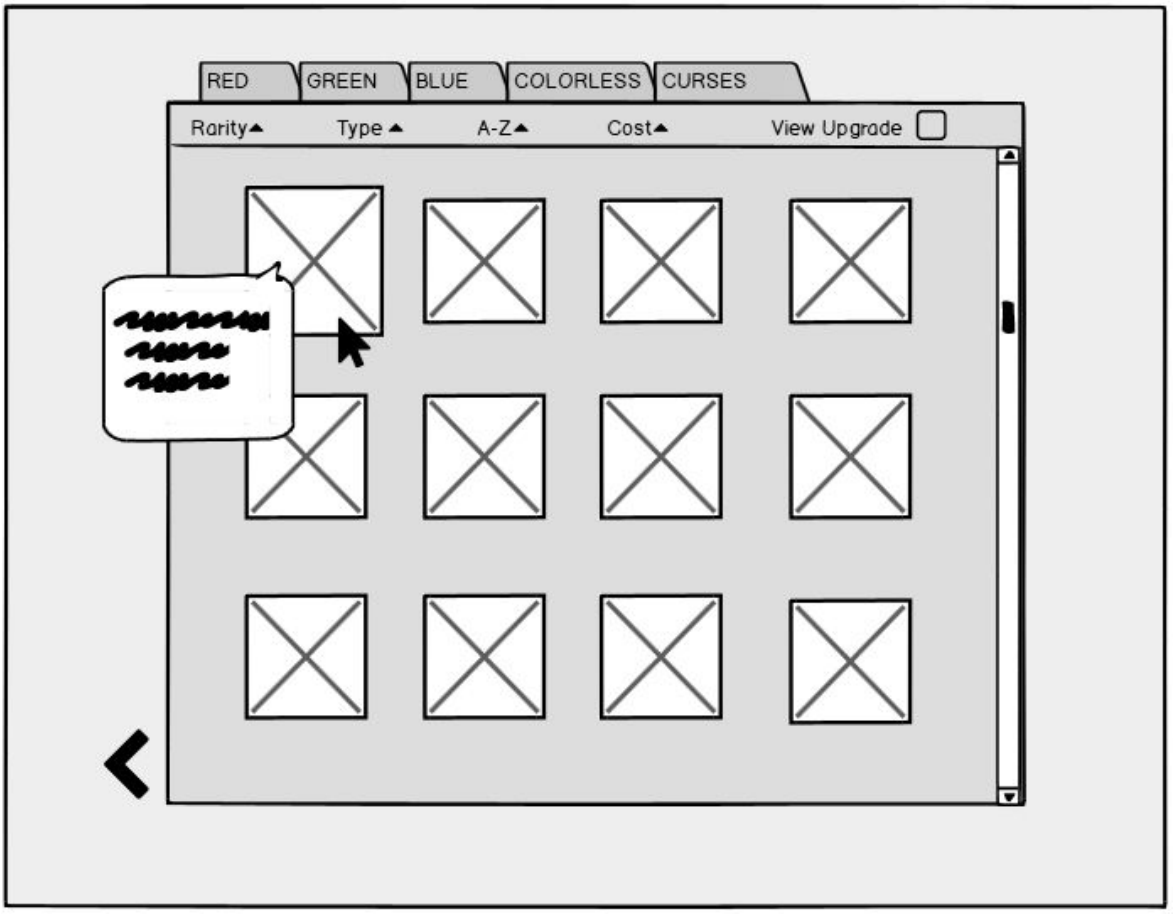

Figure 9: Card library screen of the game.

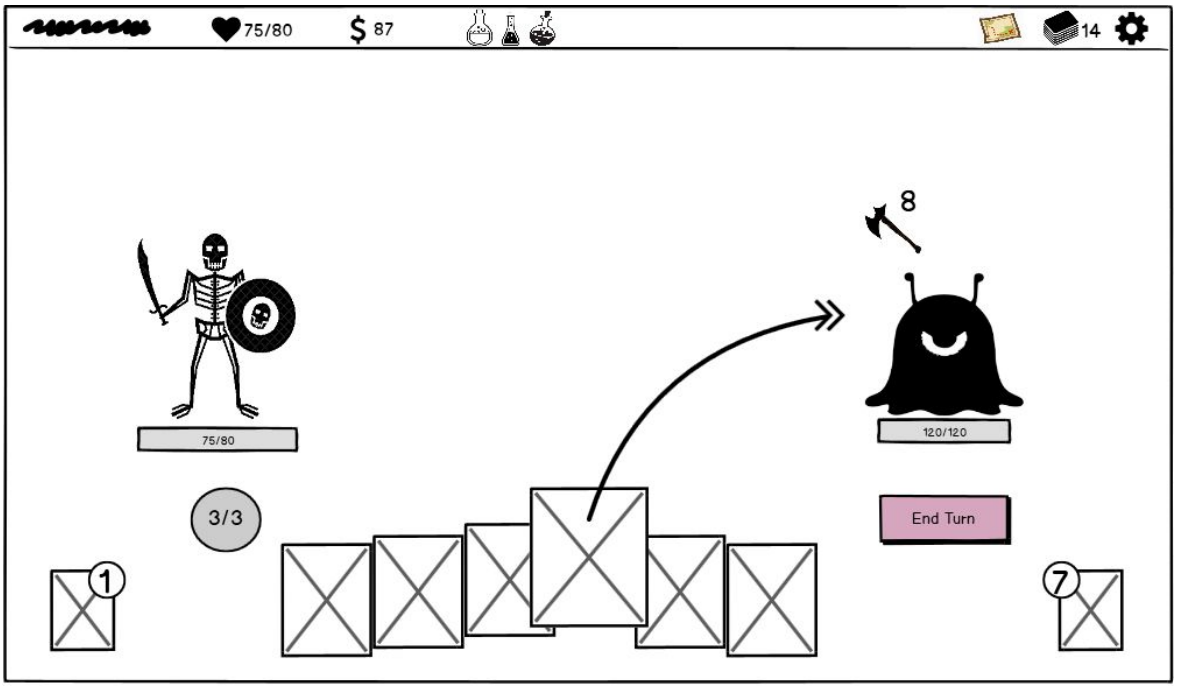

Figure 10: Combat screen of the game.

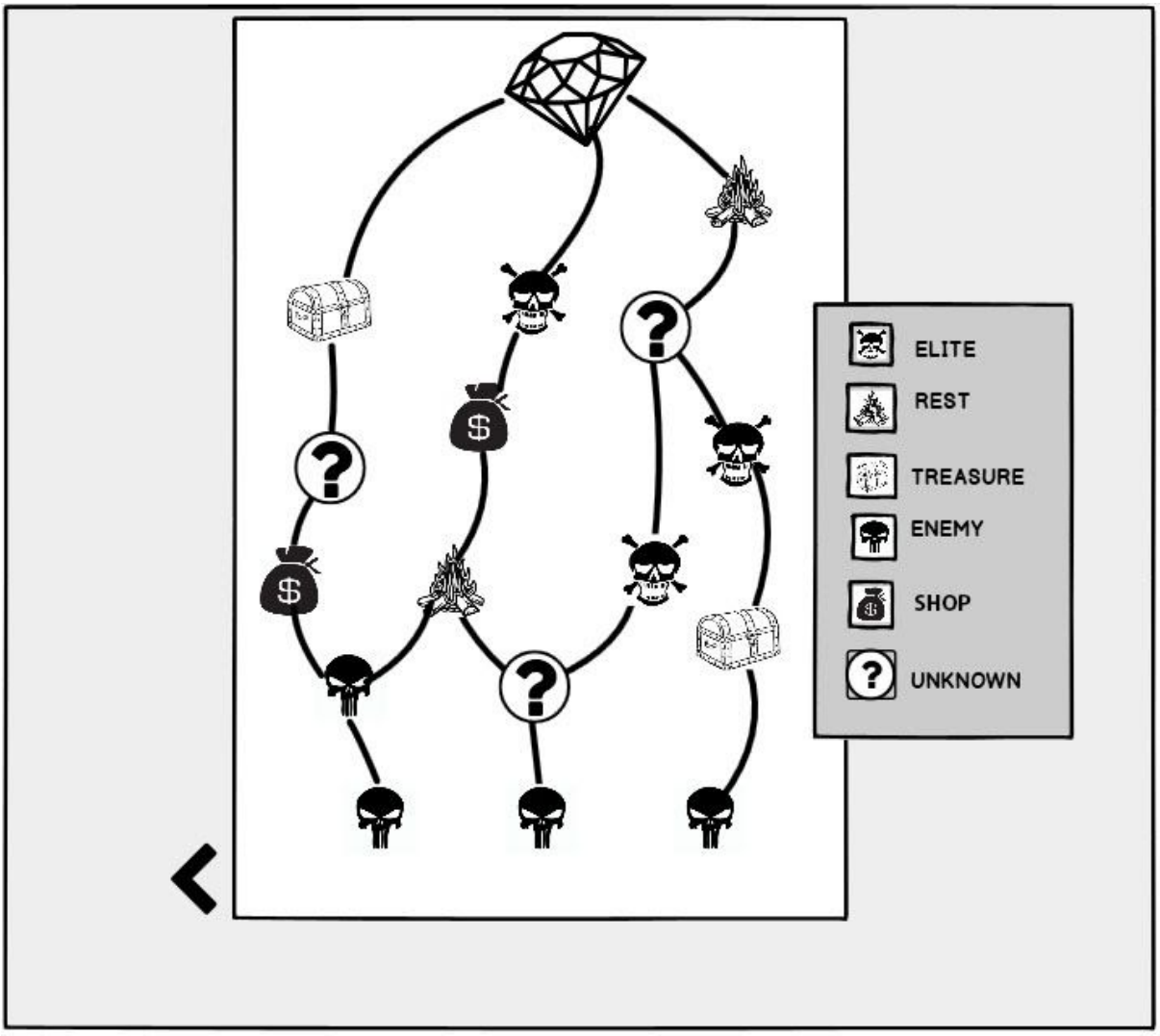

Figure 11: Map screen of the game.

8. Glossary

- **HP**: Health Point
- **Energy:** The mana point that cards require
- **Scrying:** A unique card mechanic used by Watcher class of character. Scrying enables the player to look at the cards of his/her draw-pile from the top and it doesn't reshuffles his/her deck.
- **Retaining:** Retained cards (except Ethereal cards) are kept in hand although the turn ends.

9. References

[1]"Slay the Spire Wiki," *Fandom*. [Online]. Available: https://slay-the-spire.fandom.com/wiki/Slay_the_Spire_Wiki. [Accessed: 04-Mar-2020].

[2]*Balsamiq*. [Online]. Available: https://balsamiq.com/. [Accessed: 04-Mar-2020].