Dungeons and Dragons: Deliverable #2
SE 3A04: Software Design II – Large System Design

Group6, Tutorial 2

Saad Ali - alis76
Aiyuan Liu - liua70
Harry Fu - fuh6
Francis Bajamunde - Bajamunf
Mohammend Mirajkar - Mirajkam
Shardool Patel - pates25

1 Introduction

1.1 Purpose

The purpose of this high level architectural design document is to provide the view of the Dungeons and Dragons system at an abstract level. It highlights the key components and how they interact at a high level which sets the outline for implementation.
Stakeholders and developers will have access to this document to communicate the high level design decisions regarding the product-to-be.

1.2 System Description

This document provides a high level design understanding regarding Dungeons and Dragons. This medieval themed game will be a twist on classic tower defense games by combining dungeon crawler elements to present a unique experience for users.
This game will be available for mobile devices running Android 6.0 and up. Users may download and update the app through the Google Play Store. It will require user memory access to store local save files and will not require an internet connection to play.

1.3 Overview

The remaining sections of this document provide a specific architecture design of this product using diagrams and architecture design strategies. Use case diagram is discussed in section 2 of this document. Section 3 provides an analysis class diagram for this product. Section 4 provides an overview of the overall architectural design of this system and been decomposed into subsystems with high cohesion and low coupling. The last section for this document is a class responsibility collaboration cards and provides the responsibility and collaborators of all classes for this product.

2 Use Case Diagram

1) The user selects options during which time the game play must be paused.

2) The user changes the audio of the level. The audio produced by the application must change to suit the current sound level.

3) The user modifies the contrast of the level. The contrast of all elements in the level must change to suit the current contrast.

4) The user selects a card. The card information (damage,effects,type, area) must be displayed.

5) The user casts the chosen spell and deals it's effect and damage in the area. The spell must only be able to cast on valid areas of the map.

6) The user places the chosen tower which needs a confirmation of the placement before being shown on the screen.

7) The user sells tower. The gold will be added into the account once the transaction is done and the user can no long use this tower.

8) The user upgrades the tower. The user's gold is compared to the amount user upgrades the tower. If the user has enough gold to upgrade tower then the appropriate amount of gold must be subtracted from the users gold amount and the tower is upgraded.

9) The user is attacked by enemies. The user will lose health according to the damage from the different enemies.

10) The user loads the previously saved game. This action will follow by loading the map and levels to the users using saved data.

11) The user starts a new game. This action will follow by directly loading the user into the first level.

12) The user selects a level from the map. The user will be asked to confirm their choice. Upon confirmation the level will be loaded. Upon cancellation the user will return to the map.

13) The user beats all the enemies of the level. This will be considered as a win and as a result, user will be given option to choose next level. The user will given an option to add a new card to their deck out of three options.

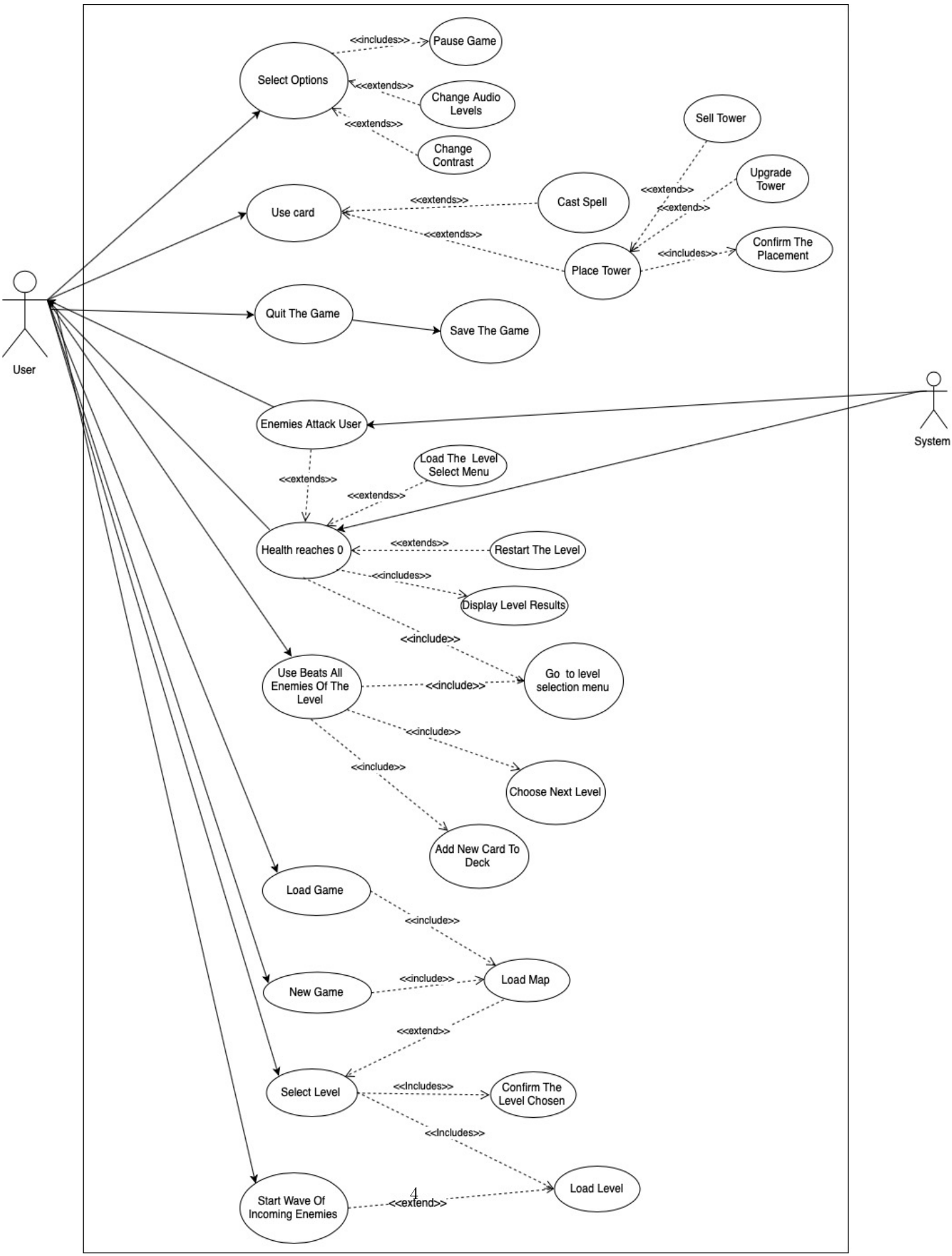
14) User wants to start the level. Upon starting the level. Enemies start advancing on the level path (from start to end).

15) The user is attacked by the enemies which may lead to health reaching 0 during level gameplay. The current level should end when this scenario occurs. The user should then be presented level results and the options to either restart the level or go to level selection menu.

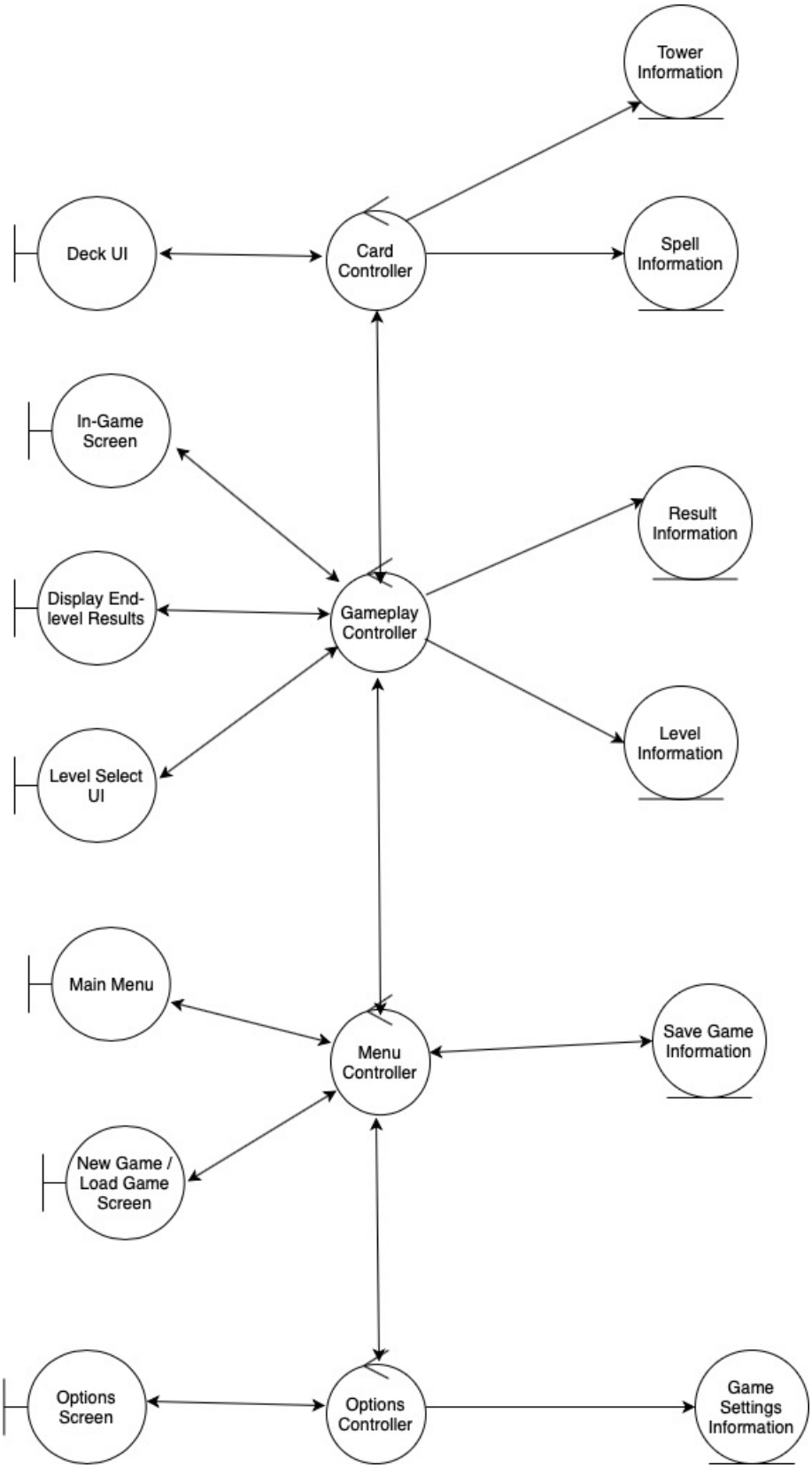
16) The user has loaded into the game but the level has not started yet. The user selects a tower card to start the game-play. When the user selects the tower card, the tower stats, available upgrades, and the area of defense. The user can then place the tower on the map. The user must be shown the valid areas and be given an option to confirm the placement.

17) The user ends the game-play. If they fail at this level, they have an option to replay the current level. Otherwise, the final result will be presented including the award and new towers and spells. After this, the user is presented with different levels and the chosen level will be shown.

18) The user wants to quit the game. This will save their current progress.


User
System
Select Options
Pause Game
Change Audio Levels
Change Contrast
Use card
Cast Spell
Sell Tower
Upgrade Tower
Place Tower
Confirm The Placement
Quit The Game
Save The Game
Enemies Attack User
Load The Level Select Menu
Health reaches 0
Restart The Level
Display Level Results
Use Beats All Enemies Of The Level
Go to level selection menu
Choose Next Level
Add New Card To Deck
Load Game
New Game
Load Map
Select Level
Confirm The Level Chosen
Load Level
Start Wave Of Incoming Enemies
<<includes>>
<<extends>>
<<extend>>
<<include>>
<<Includes>>
<<Includes>>

3 Analysis Class Diagram



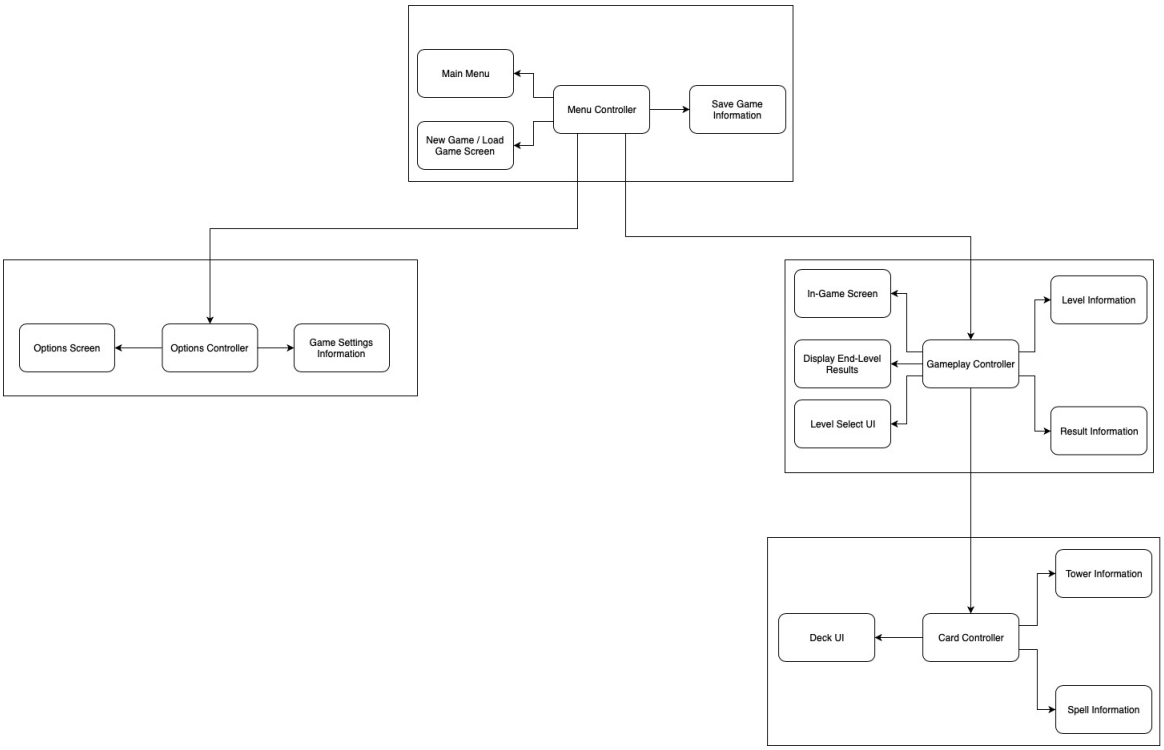
4 Architectural Design

4.1 System Architecture

1) We are going to use an interaction-oriented software architecture.

2) The overall architecture of the system is a Presentation-Abstraction-Control (PAC) architecture.

3) The system was chosen to be a PAC architecture because the structure is given in the analysis class diagram components of presentation, abstraction, and control. The controller classes act as the controllers of the system, the boundary classes act as the presentation components, and the entity classes act as the abstraction components which retrieve, store, and process data. Furthermore, there are no avenues for communication between the presentation and abstraction components. Only the controller components of the system allow for the transfer of data between the boundary classes and the entity classes.



4.2 Subsystems

a) **Menu Controller Subsystem** - The menu controller subsystem resembles a Model View Controller architecture style. This subsystem would be in charge of displaying the various menus the player needs to access various game options like New Game and Load Game. It also controls the user's access to the gameplay subsystem and the options subsystem.

b) **Options Controller Subsystem** - This subsystem would be in charge of allowing the user to manipulate in-game settings like sound and contrast.

c) **Gameplay Controller subsystem** - This subsystem is in charge of controlling the main gameplay of the application. It shows the user the in-game screen, the end level results, and the level select UI. It has access to specific level information, and has access to player results after gameplay is finished.

d) **Card Controller Subsystem** - This subsystem is in charge of displaying both the tower and spell decks to the user. The subsystem has access to information on the various towers and spells in the game.

5 Class Responsibility Collaboration (CRC) Cards

1. **Boundary Classes**

Class Name: Deck UI	
Responsibility:	**Collaborators:**
• Display available Towers in the current deck • Display available Spells in the current deck • Allow Spell and Tower selection • Display Tower and Spell meta-data (cost, description, etc)	• Card Controller

Class Name: In-Game Screen	
Responsibility:	**Collaborators:**
• Display current level • Render and Animate Enemy movements across the level • Render and Animate Towers • Show spell area of effect • Accept Tower placements • Allow Tower upgrades • Accept Spell use • Show Player's resources	• Gameplay Controller

Class Name: Results Screen	
Responsibility:	**Collaborators:**
• Show level results • Given the user the option to restart the level (if user did not win) • Given the user the option select a new card to add to their deck (tower card or spell card) if the user won the level.	• Gameplay Controller

Class Name: Level Select UI	
Responsibility:	**Collaborators:**
• Display all current levels that user can play. • Accept Level selection	• Menu Controller

Class Name: New Game/Load Game Screen	
Responsibility:	**Collaborators:**
• Display start game and load game options. • Accept appropriate selection.	• Menu Controller

Class Name: Main Menu	
Responsibility:	**Collaborators:**
• Display Start Game and Options. • If opened during gameplay display Save Game and Options. • Accept appropriate selection.	• Menu Controller

Class Name: Options Screen	
Responsibility:	**Collaborators:**
• Display game options • Accept options changes	• Options Controller

2. **Control Classes**

Class Name: Card Controller	
Responsibility:	**Collaborators:**
• Check if a given tower can be placed with available resources • Update gamestate (through gameplay controller) for placed Towers. • Give Deck UI appropriate Tower information. • Update resources based on the Tower placed. • Check if a given Spell can be used with available resources. • Update gamestate (through gameplay controller) after a Spell use. • Give Deck UI appropriate Spell information. • Update resources based on the Spells used.	• Deck UI • Tower Information • Spell Information • Resources • Gameplay Controller

Class Name: Gameplay Controller	
Responsibility:	**Collaborators:**
• Control Enemy spawns • Add Spells and Towers to gameplay screen • Start Gameplay based on level selected • Load level based on initial level information (number of waves, difficulty, starting player resources) • Update level information based on gameplay updates. • Control Enemy updates (health point change, enemy killed, spawn and despawns) • Show results at the end of the level • Accept next level selection after successful level completion. • Update game settings from options controller	• In-Game Screen • Results Screen • Level Select UI • Level Information • Card Controller • Options Controller

Class Name: Options Controller	
Responsibility:	**Collaborators:**
• Check changes to game settings • Update game settings • Switch view back to menu or gameplay from options view.	• Options Screen • Game settings information • Menu controller

3. **Entity Classes**

Class Name: Tower Information	
Responsibility:	**Collaborators:**
• Knows cost of Tower • Knows description of Tower • Knows attack range of Tower • Knows attack type of Tower • Knows model/image of Tower	

Class Name: Spell Information	
Responsibility:	**Collaborators:**
• Knows cost of Spell • Knows description of Spell • Knows area of effect of Spell • Knows model/image of Spell	

Class Name: Level Information	
Responsibility:	**Collaborators:**
• Knows level information: level design, difficulty, enemy type and wave frequency. • Stores current level state.	• Gameplay Controller

Class Name: Save Game Information	
Responsibility:	**Collaborators:**
• Knows last saved game state. • Accepts updates from Menu controller for new game save data.	• Menu Controller

Class Name: Game Settings Information	
Responsibility:	**Collaborators:**
• Knows optional settings for the game • Knows the current game setting • Stores game setting while it changed	• Options Controller

A Division of Labour

Harry Fu, 400065502 - CRC, Analysis Class Diag, Use Case Diag

Aiyuan Liu, 400223883 - Use Case Diag, User Case Scenarios

Francis Bajamunde, 400031789 - Introduction, Analysis Class Diagram, Architectural Design.

Shardool Patel, 400080843 - CRC, Analysis Class Diagram, Use Case Scenarios

Saad Ali, 400079576, Introduction, Use Case Diagram, Use Case Scenarios

Mohammnhed Mirage, 400088406, Use Case Diagram, Use Case Scenario, Editing.