Deliverable #2 Group #1 T02

SE 3A04: Software Design II – Large System Design

1 Introduction

1.1 Purpose

The purpose of the document is to provide a high level design for the system architecture and class composition. This document is intended for use by the stakeholders of the project as reference throughout the various stages of the SDLC. The primary stakeholders include Prof. Ridha Khedri, who has commissioned this project as part of Software Engineering course SE3A04 as provided by McMaster University, Tutorial Assistants for SE3A04, who are responsible with oversight and review of the deliverables of the project and members of the development team (ie. authors of the document). Other stakeholders include any future teams of developers and managers, that may maintain the application or use this project application as an open source reference.

1.2 System Description

Climatar is a world simulation software system illustrating the long term and short term effects of actions with respect to climate, greenhouse gas levels, economic stability and social relations with the governing bodies in the model world. The model world is based in the Avatar: The Last Airbender universe. Users are given governance of one of the four tribes in the Avatar universe: Air, Water, Fire, and Earth. Based on the state of the world and the regions that compose it, news events will be posed to the user that require action. Depending on their decisions the world will morph around the change and consequences will propagate throughout the simulation. All decisions have consequences associated with them, and the user can only react to events pertaining to their region, removing the omnipotent aspect of control, users must react to the changing world dynamically.

1.3 Overview

The remainder of the document pertains to the high level system design specifically outlining how the system will be structured as illustrated by the Use Case Diagram, Analysis Class Diagram, the inherent Architecture Design, and Class Responsibilities. The document first outlines the use cases for Climatar. The use cases illustrate the business events pertaining to the system and layout the system functionality. Descriptions of each use case aide in formalizing the system functionality from the view points identified previously in the SRS. From the Use Case Descriptions, a Noun-Verb Analysis was completed to create an Analysis Class Diagram. The Analysis Class Diagram acts as a starting point from which the major classes of the system will be extracted. The following section, Architecture Design, utilizes the information gained from the analysis class diagram to develop a system architecture to layout the system is a way such that each sub system has high cohesion and low coupling. Finally a Class Responsibility Collaboration (CRC) Card is developed for each class to formally outline the responsibilities, and dependencies for a given class.

2 Use Case Diagram

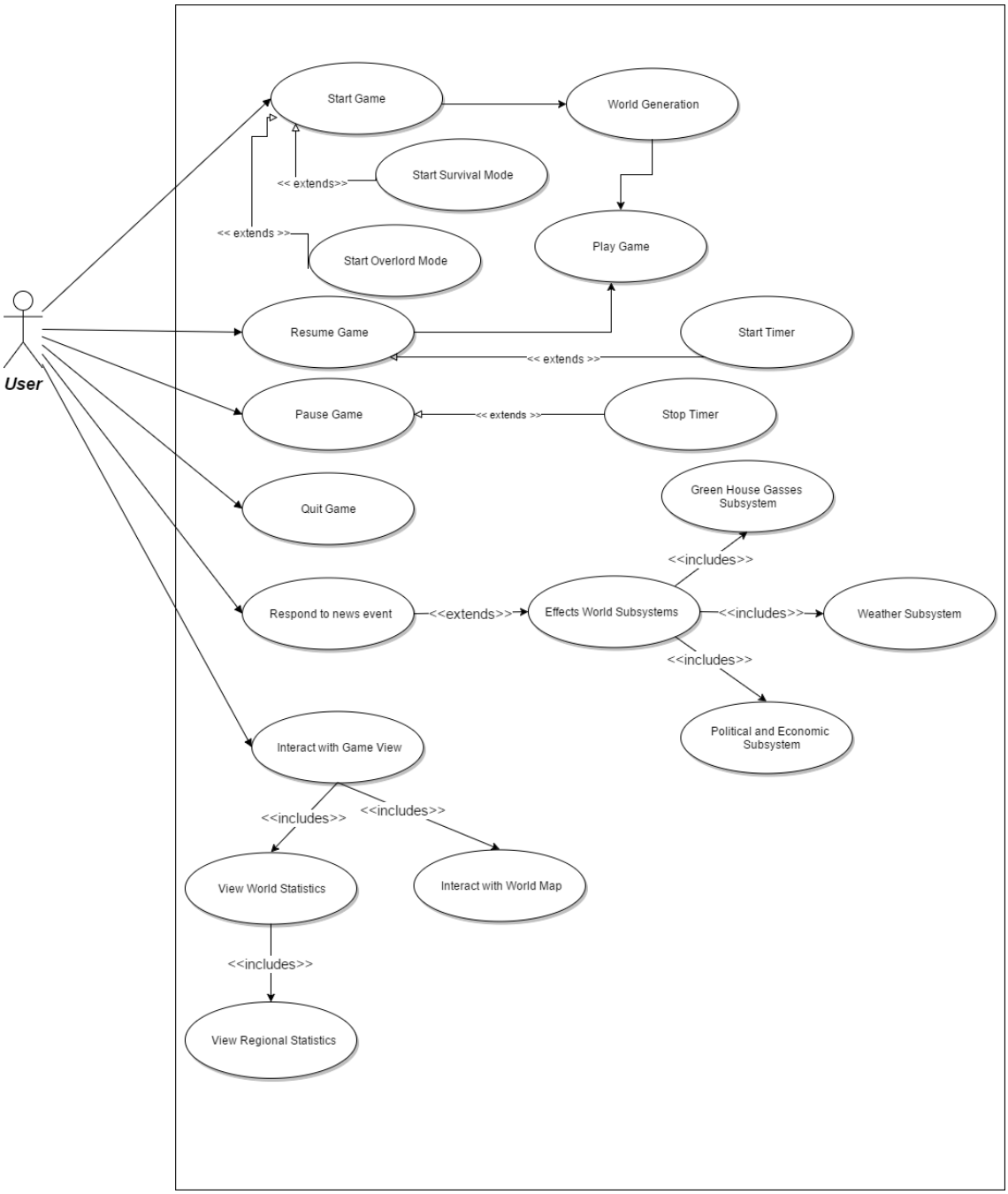

Figure 1: Use Case Diagram

a) the main menu screen, the user taps on the start game button which gives the user the option to select the survival game mode or overlord game mode. Selecting one of these options initializes the game world, and switches to the play screen where the user is presented with news events periodically.

b) Selecting one of these game modes, initializes the game world, and switches to the play screen where the user is presented with news events periodically.

c) The user taps the resume button on the play screen which resumes the game system to a playing state.

d) A user taps the pause button to pause the game. The ongoing game transitions to a paused state and timers controlling world simulation the news event generation are paused.

e) A user quits the game. They are prompted if they really want to quit the game early. If they select yes then the current game is destroyed and the game exits to the main menu. Otherwise the game continues unaltered.

f) The user responds to a news event generated by the system, by selecting one of the actions from the news event display. The action is sent back to the game world and it's implications are applied to the subsystems.

g) A user taps the pause button to pause the game. The ongoing game transitions to a paused state and timers controlling world simulation the news event generation are paused.

3 Analysis Class Diagram

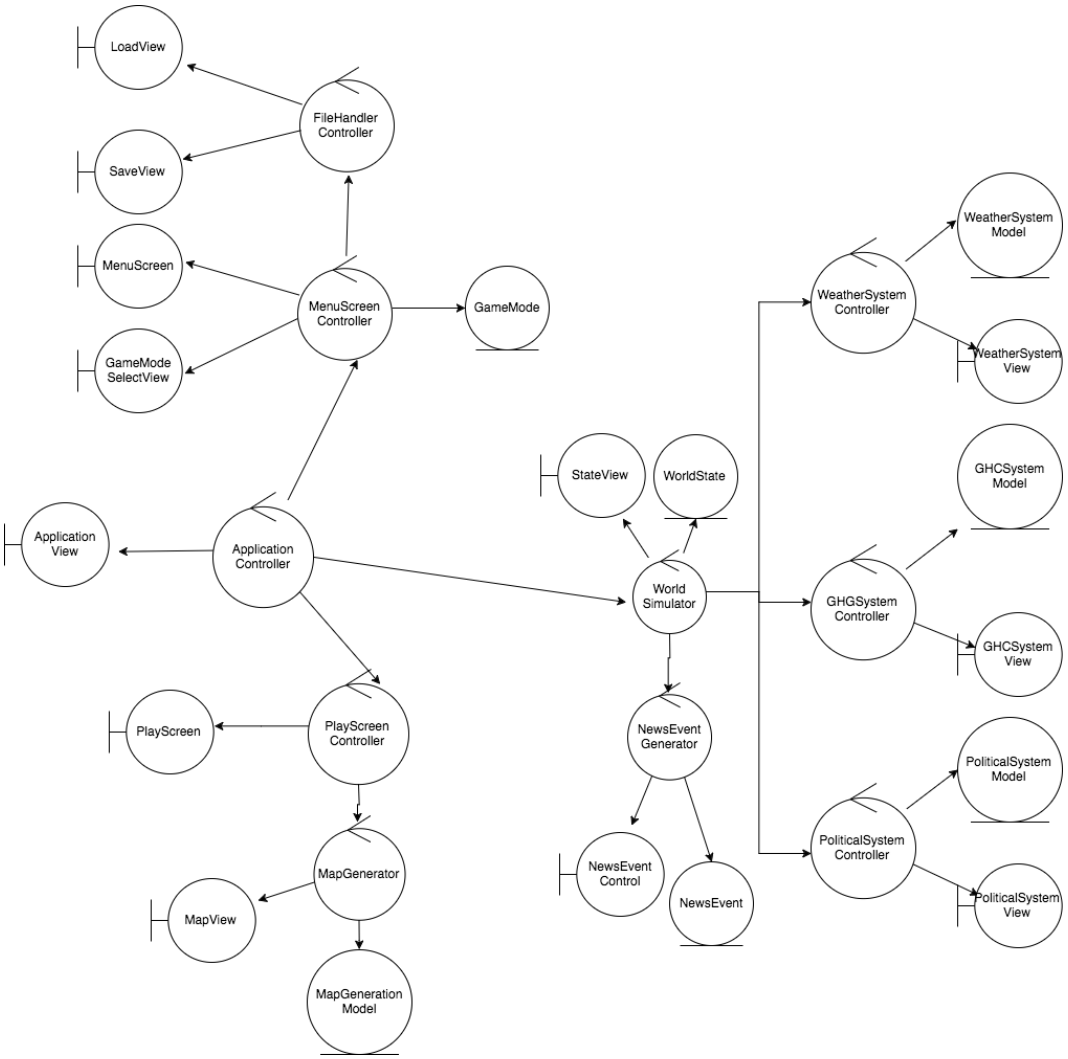

Figure 2: Analysis Class Diagram

4 Architectural Design

4.1 System Architecture

The system is divided into the GUI and World subsystems. The World subsystem further divides into Weather, Green House Gases (GHG), Political, and News Event subsystems. The GUI and World sub systems connect to one another but their controls run asynchronously to one another. The selected software architecture for the development of Climatar is Presentation-Abstraction-Control, PAC.

PAC is the optimal software architecture for Climatar as it helps abstract and modularize the subsystems, allowing also for concurrency between them. Using PAC will also ensure that modules are loosely coupled such that changes do not propagate throughout the system. This principle of loose coupling will make the system more maintainable and extensible in the future. Furthermore, the project has a development time line of approximately one week in which all code will be written. PAC's inherently modular nature allows developers to work in contained subsystems independent of other developers, thus the parallelism for development is optimal.

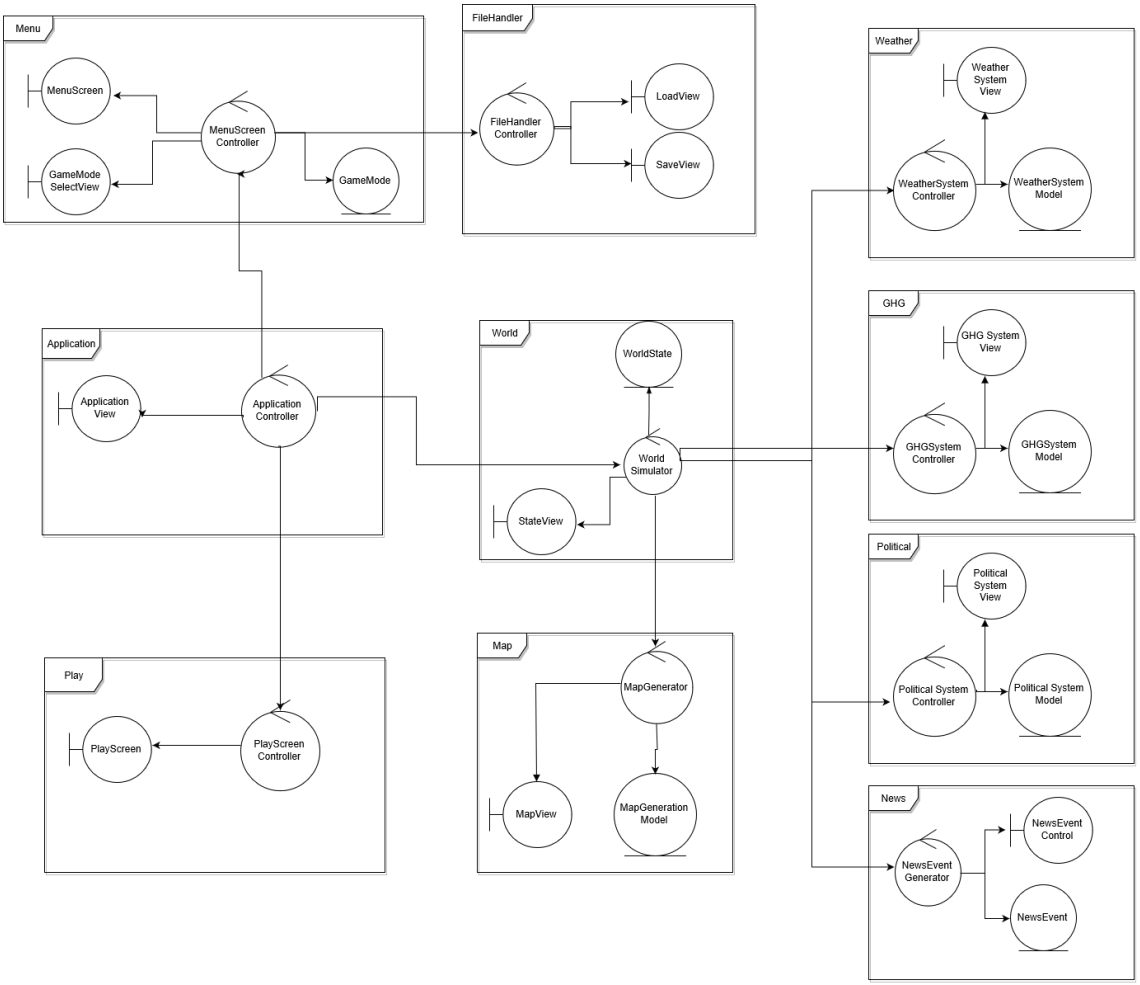

Figure 3: System Architecture, PAC

4.2 Subsystems

The following subsystems compose Climatar:

- **Application:** The Application subsystem is the entry point for the system, and acts as a top level controller for Climatar. This subsystem responds to user stimulus and passes command to the subsystem responsible for handling the stimulus, however this module can only command the World, Menu, and Play subsystems, which compose the control of the remaining subsystems.
- **Play:** The Play subsystem is responsible for displaying all elements composing the games view. Control is passed to this subsystem from Application.
- **Menu:** The Menu subsystem is responsible for all controls and UI components associated with the Main Menu/Title Screen for Climatar, and the game mode the game will be activated in. The Menu is given control by the Application subsystem and can control the FileHandler for loading and saving game instance requests.
- **FileHandler:** The FileHandler subsystem is controlled by the Menu for load and save game requests.
- **World:** The World subsystem is responsible for simulating the world in a Climatar game instance, this includes the information retrieval and interpretation from all world creating subsystems. Regions/Nations are linked to their corresponding subsystems.
- **News:** The News subsystem is responsible for storing possible news events, and responding to news event requests through passing an appropriate event given a specified criteria.
- **Map:** The Map subsystem is responsible for the generation and holding of the current map for a given game instance.
- **Weather:** The Weather subsystem is responsible for the simulation of the weather of a region. This subsystem can be disconnected from the major system and not effect its ability to work, weather will just not be a factor in the simulations.
- **GHG:** The GHG subsystem is responsible for the simulation of the green house gas levels and the rate of change of green house gas levels of a region. This subsystem can be disconnected from the major system and not effect its ability to work, green house gases will just not be a factor in the simulations.
- **Political:** The Political subsystem is responsible for the simulation of the economical aspects of a region and a nations relations with the other nations. This subsystem can be disconnected from the major system and not effect its ability to work, political factors will simply not partake the simulations.

A visual representation of the high level system composition can be seen in Figure 4.

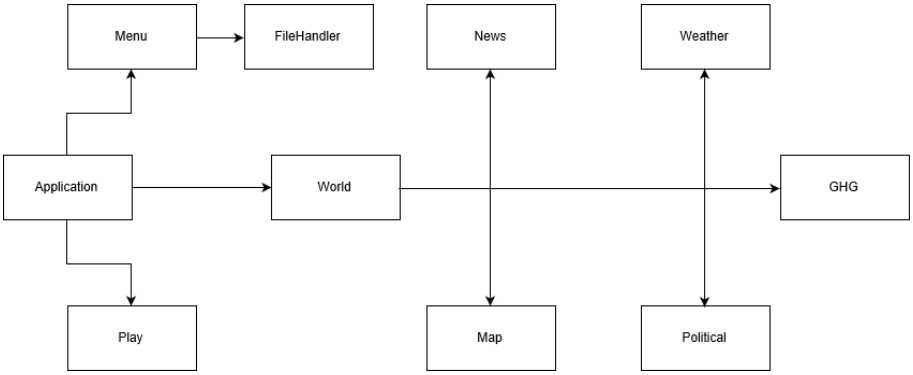

Figure 4: Module Uses Relation

5 Class Responsibility Collaboration (CRC) Cards

This section should contain all of your CRC cards.

1.

Class Name: ApplicationController	
Responsibility:	**Collaborators:**
Display the menu screen	MenuScreenController
Display the play screen	PlayScreenController
Change the application view	-
Start the world simulator	WorldSimulator
Stop the world simulator	WorldSimulator

2.

Class Name: ApplicationView	
Responsibility:	**Collaborators:**
Accept and respond to user inputs and preferences	-

3.

Class Name: FileHandlerController	
Responsibility:	**Collaborators:**
Maps a games state into text format	-
Saves a game file to the hardwares persistent storage.	-
Reads a game file from the hardwares persistent storage.	-
Maps a game files text data to a game state.	-
Displays a LoadView prompt.	LoadView
Displays a SaveView prompt.	SaveView

4.

Class Name: LoadView	
Responsibility:	**Collaborators:**
Displays a list of save game files.	-
Allows selection of a save game file.	-
Displays a success indicator.	-

5.

Class Name: SaveView	
Responsibility:	**Collaborators:**
Displays a success indicator.	-

6.

Class Name: GHGSystemController	
Responsibility:	**Collaborators:**
Modify current GHG levels	GHGSystemModel
Check if GHG levels are within safe levels	GHGSystemModel
Update GHG statistics in simulator	WorldSimulator

7.

Class Name: GHGSystemModel	
Responsibility:	**Collaborators:**
Maintain GHG level data	-

8.

Class Name: GHGSystemView	
Responsibility:	**Collaborators:**
Display GHG statistical data	-

9.

Class Name: MapView	
Responsibility:	**Collaborators:**
Retrieves user interaction with World Map	Play Screen Co ntroller
Display map	-

10.

Class Name: MapGenerator	
Responsibility:	**Collaborators:**
Can generate a new game world. Uses a MapGenerationModel to procedurally generate a game world.	- MapGenerationModel

11.

Class Name: MapGenerationModel	
Responsibility:	**Collaborators:**
Knows parameters that alter how a game world is generated.	-

12.

Class Name: MenuScreen	
Responsibility:	**Collaborators:**
Accepts and responds to Resume Game Request	MenuScreenController
Accepts and responds to Mute music request	MenuScreenController
Accepts and responds to Mute sounds request	MenuScreenController

13.

Class Name: MenuScreenController	
Responsibility:	**Collaborators:**
Opens the menu screen	MenuScreen
Closes the menu screen	MenuScreen
Mutes and unmutes sound	ApplicationController
Mutes and unmutes music	ApplicationController
Set game mode	GameMode
Display game mode selection view	GameModeSelectView
Accepts and responds to Save Game request	FileHandlerControler
Accepts and responds to New Game request	-
Accepts and responds to Load Game Request	FileHandlerControler
Accepts and responds to Quit Game Request	-

14.

Class Name: GameMode	
Responsibility:	**Collaborators:**
Receive users mode selection	MenuScreenController
Store current mode selection	-
Send constraint for user base on the mode to the MenuScreenController	MenuScreenController

15.

Class Name: GameModeSelectView	
Responsibility:	**Collaborators:**
Accept and respond to game mode selection	MenuScreenController
Display game modes	-

16.

Class Name: NewsEventControl	
Responsibility:	**Collaborators:**
Retrievers user interaction with news events	PlayScreenController

17.

Class Name: NewsEventGenerator	
Responsibility:	**Collaborators:**
Knows how to generate a news event.	NewsEvent
Can pass a news event to the NewsEventControl to be displayed.	NewsEventControl
Can forward a news events selected action to the WorldSimulator.	WorldSimulator

18.

Class Name: NewsEvent	
Responsibility:	**Collaborators:**
Knows the textual description of the news event	-
Knows a set of actions that the user can select in response to the news event.	-

19.

Class Name: PlayScreen	
Responsibility:	**Collaborators:**
Displays state of the World	PlayScreenController
Display Menu	PlayScreenController
Displays state of subsystems	PlayScreenController
Displays World Map	PlayScreenController
Accepts and responds to news events requests	-
Accepts and responds to world map requests	-
Accepts and responds to Pause Game Request	-

20.

Class Name: PlayScreenController	
Responsibility:	**Collaborators:**
Display GHG, Weather, Political and News statistic data	-

21.

Class Name: PoliticalSystemController	
Responsibility:	**Collaborators:**
Modify current political levels	PoliticalSystemModel
Check if political levels are within safe levels	PoliticalSystemModel
Update political statistics in simulator	WorldSimulator

22.

Class Name: PoliticalSystemModel	
Responsibility:	**Collaborators:**
Maintain political atmosphere data	-

23.

Class Name: PoliticalSystemView	
Responsibility:	**Collaborators:**
Display political statistics data	-

24.

Class Name: WeatherSystemController	
Responsibility:	**Collaborators:**
Modify current temperature levels	WeatherSystemModel
Check if temperature levels are within safe levels	WeatherSystemModel
Update Weather statistics in simulator	WorldSimulator

25.

Class Name: WeatherSystemModel	
Responsibility:	**Collaborators:**
Maintain temperature level data	-

26.

Class Name: WeatherSystemView	
Responsibility:	**Collaborators:**
Display weather/temperature statistics	-

27.

Class Name: WorldState	
Responsibility:	**Collaborators:**
Store current world information	-
Send world state information to the WorldSimulator	WorldSimulator
Retrieve calculated world state info from WorldSimulator	WorldSimulator

28.

Class Name: StateView	
Responsibility:	**Collaborators:**
Retrievers user interaction with World state	PlayScreenController
Retrieves user interaction with World Subsystems	PlayScreenController

Class Name: WorldSimulator	
Responsibility:	**Collaborators:**
Generate news events	WeatherSystemController, GHGSystemController, PoliticalSystemController, NewsSystemController

29.

A Division of Labour

Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

Contributor Name	Contributions
Wenbin Yuan	Uses Cases and CRC cards. Revised the document.
Haris Khan	CRC cards, initial draft of use cases.
Riley McGee	Original draft of 1.* and 4.* including figures for Module Uses Relation and Architecture
Vishesh Gulatee	Use Cases, CRC cards, and document revision.
James Taylor	Analysis Class Diagram, use case diagram refinement.

By signing below you agree to the work divisions stated above correctly representing all contributions made:

IMPORTANT NOTES

- Please document any non-standard notations that you may have used
  - *Rule of Thumb*: if you feel there is any doubt surrounding the meaning of your notations, document them
- Some diagrams may be difficult to fit into one page
  - It is OK if the text is small but please ensure that it is readable when printed
  - If you need to break a diagram onto multiple pages, please adopt a system of doing so and thoroughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask about it
- Please submit the latest version of Deliverable 1 with Deliverable 2
  - It does not have to be a freshly printed version; the latest marked version is OK
- If you do NOT have a Division of Labour sheet, your deliverable will NOT be marked