ENG1 Group 2 Marlin Studios
Part 2

Implementation Report

SKYE FULLER

JONATHAN HAYTER

ALEXANDER MIKHEEV

MARKS POLAKOVS

FREDERICK SAVILL

HAOWEN ZHENG

Implementation

For the implementation of the requirements necessary for Assessment 2, no big changes were made to the architecture of the codebase. New features were designed beforehand to complement the architecture, and make them easier to test.

New Features Implemented

Change	Description	Requirement
Added Minimap	Minimap now shows up on the player's HUD	FR_MINIMAP

This was one of the smaller changes we made, however, we felt it was necessary to mention as it has a significant impact upon the player experience. The minimap was implemented as a new function in the **GameUI** class, which effectively contains the game overlay. It was implemented using 3 different sprites; a PNG of the map itself, a player indicator, and a system indicator. The player indicator moves across a still image of the map to show the player their location relative to the map. The system indicator sprite is rendered in the relative position of a system being attacked.
We felt this was a necessary feature to include as it was an unimplemented feature requirement from Assessment 1.

Change	Description	Requirement
Power Ups	Auber can find and use different power ups to aid them in arresting all of the Infiltrators	FR_POWER_UPS, FR_POWER_UPS _ABILITIES

The was the first major new feature implemented as per the requirements for Assessment 2, and its implementation consists of 2 new classes.
A class **PowerUp** was created to extend the abstract class **GameEntity**. The class **PowerUp** takes x and y coordinates as floats for its position, as well as a **World**, and an **Abilities**. The **World** it takes is used to identify the **Player** the power up applies to, and the variable of type **Abilities** is used to determine what power up to spawn in.
Abilities is a separate Enum which contains a method to return a randomly generated value from the possible range of power ups.
The **spawnBuff()** function in **World** is used to spawn in a random ability at the location of a random **Civilian**. If no **Civilians** are present in the current world, **spawnBuff()** will not spawn a power up. The function is called whenever an **Infiltrator** begins to attack a system, the number of times it is called is dependent upon difficulty.

Change	Description	Requirement
Menu UIs	Existing menus were redesigned, others were created	FR_MENU, FR_DEMO, FR_TUTORIAL

This was the second major new feature/update implemented into the project. We felt this was necessary in order to properly satisfy the new **FR_DIFFICULTY** requirement that was introduced as per Assessment 2.
We begin this by rewriting the **Button** class to display a new **Sprite** when the mouse was hovering over it rather than just enlarging the current **Sprite**. From there, we drew all of the necessary button textures in photo editing software.

In total, we created two new UIs and updated one.
The **MenuUI** was updated so that the user could load their save file, change difficulty, as well as start a new game. Upon pressing the Play button, the user is then shown a small tutorial explaining the basics of the game.
The first UI we created was **PauseUI**, this was the UI to be shown when the user had paused the game. It included 3 buttons, “Quit” to return to the menu, “Save & Quit” to save their game, and return to the menu, and a “Resume” button, to return the user back into the game.
The second UI we created was the **GameOverUI**, which was to be displayed when the game has ended in either a win or a loss. It includes 2 buttons, “Quit” to return the player to the menu, or “Play Again” to start a new game without the user having to look through the tutorial slides again.

Change	Description	Requirement
Added Difficulties	Added a Hard difficulty to make the game more of a challenge	FR_DIFFICULTY

This was implemented as per the requirements of Assessment 2. Having implemented the new **MenuUI**, this was fairly easy to implement. There is a button on the menu screen which when clicked, toggles between “Easy” and “Hard”. This value is represented in a new Enum class called **Difficulties**, and is passed into a **World** constructor. The game’s default difficulty setting is Easy, so changes are only made to the World when the difficulty is set to Hard. The changes made in Hard mode are exclusively to World constants such as **AUBER_BUFF_TIME**, **BUFFS_ON_ATTACK**, **INFILTRATOR_SABOTAGE_CHANCE**, and **INFILTARTOR_FIRING_INTERVAL**.

Change	Description	Requirement
Added Pause Functionality	The user is able to pause the game and continue it at a later time	UR_PAUSE

Again, this was one of the smaller changes we made, however, seeing as it has a large impact on player experience, we decided to include it.
If the player presses the ESC key while playing, the game will pause and render **PauseUI**. This was achieved by creating a boolean value “paused” in the **GameScreen** class which is true when the game should be paused, and false otherwise. If the boolean is true, the **GameEntity.Update()** function is not called, thus pausing their movements.

Change	Description	Requirement
Saving/Loading	Player can save a game, and continue it in another instance of the game	UR_SAVE

This was the final feature implemented into the game as per the requirements of Assessment 2. The system works by recursively serialising the state of the **World** (which contains all the entities and systems in play) to a JSON file in the player’s home directory, and deserialising it to load the game. We chose to use JSON (using LibGDX’s built-in serialiser) over Java’s native binary serialisation system, as it leads to easier debugging - if there is a bug in the logic we can simply inspect the generated file in a text editor.

Feature Requirements not Implemented

These are the requirements that are not implemented in the final version of the project.

Requirement	Consequence	Explanation
NFR_SCALABLE	Game does not function as intended when the application window is resized.	This requirement was not met due to time constraints.
FR_TELEPAD_DESTINATION	The player cannot pick which teleport pad to teleport to (Only an issue when teleporting from the Infirmary)	This was a low priority requirement, and it was decided that it would have such a minor impact upon the playing experience that we could not justify spending the time to implement it.
FR_SYSTEM_HEALTH	If Infiltrator is arrested 5 seconds after starting attack, the time needed to destroy the system is reset	Systems can either be WORKING, ATTACKED, or DESTROYED. A system only becomes DESTROYED when it is being ATTACKED for 5 seconds. This was noticed too late in the development process to make any changes.
FR_HOSTILES_SPAWN	Only the active Infiltrators are spawned in at the beginning of the game	There are only 3 Infiltrators maximum, active at any point in the game. This was a misled feature from part 1, and was not implemented due to time constraints