Any changes that were made were done so in order to meet the requirements given, to improve the user experience or to increase code clarity. Any code change has the relevant requirement referenced by the unique ID and is emboldened. The requirements document that we followed for this is referenced below [1].

New Features:

Bad Weather

The tile map has been amended so that it includes patches of bad weather as per **F3.3**, this area earns more points if sailed through meeting **F6.3**. This implementation for this was quite efficient, bad weather was defined as a College ('Stormy') as college regions had all the features needed, for example, random battle encounters depending on regions were already implemented, and we wanted to add a storm-specific monster, while the region name label on the UI would update whenever the player would enter a college's region, which we used in making sure the player understood that they were in a region with bad weather. It also meant we were able to add the weather as RegionData in the Tiled map. As we had limited time, we felt that this was the most efficient way of implementing bad weather into the game in the time we had, and if we had more time, we would've created a new class for it instead. There were two new variables for SailingScreen, boolean inStorm to see if ship was in storm and integer weatherTimer for when the player takes damage. When in weather, set boolean to true and start counter, after the counter reaches a certain point, damage is taken and timer is reset. Once the player is out of the storm, the boolean and timer set to false/0.

An addition that affected the entire game was a label called healthLevel which was needed as during bad weather regions there was no way to see the current health. This made sense as previously there was no way to affect health levels in sailing mode. This was implemented specifically to make the game more user friendly and enhance the user experience as per **C3.1**.

Minigame

The minigame is a game of Cee-lo (rules explained in User Manual). This was implemented in order to meet the requirements, including **F12** as it is completely different from the main game, and **F12.1** because it allows the player to bet with their gold acquired from the game. To implement this, the minigame relies on 3 new classes, the first of which is MinigameScreen which relies on extending BaseScreen as do the other Screens within the game. It covers the display and interface of the minigame, whereas the actual logic of the game is within Minigame. The dice displayed are objects controlled by DiceImage, which is based off CombatShip allowing for easy change of the images. VictoryScreen is also used, and its usage is explained below.

The minigame offers the option to 'Sell Soul' if you have no gold. Besides being an interesting addition to the game it is also the only way to completely lose the game. A feature we felt would make the game more satisfying for the user as per **C3.1**. The added sense of risk accompanied with the chance to lose the game increases the user experience which we deemed extremely important.

Screens
There is now a Victory Screen which displays the points total once the main objective has been completed (therefore once the game is won) which satisfies **F2.1** and **F2.2**. The Victory Screen is a screen in the same style of the pre-existing screens within the game, inheriting BaseScreen. Additionally, in the rare instance in which a player attempts to use the minigame 'Sell Soul' option and fails, it is also used to display a 'Game Over' screen.

There is a ObjectiveScreen now which displays an overview of the map as well as the objectives that need completing in order for the game to be won. This was added to fulfil **F10** with regards to objectives and also to increase usability within the game as a map overview allows the player to increase their spatial awareness which allows them to focus on achieving their objectives, therefore achieving **C3.1** also.

Final Boss
A final boss was implemented into the game as per the requirements **F10.1** and **F10.2**, stating that there much be a final boss that needs to be defeated for the game to be won, which must be difficult enough that the player has to progress through the game and purchase upgrades to have a chance in defeating. This was implemented by adding additional checks on SailingScreen.java, when the player interacts with a college island that they're not allied with. It checks if the name of the college is equal to the boss name, in this case it's "Halifax", and a battle with a new ship that is stronger than all other enemy ships is created. This leads to the Victory Screen once it is defeated.

Obstacles
As per requirement **F3.3**, obstacles have been added to the sailing mode in the form of rocks that do not let the player pass through and force them to take a detour. These have been implemented as PhysicsData in the Tiled map, similar to the borders of the islands, which block the player and stop their momentum. We have only went as far as making rocks block the player rather than cause them to take damage as we thought that it would be harmful to the player experience and enjoyment, focusing on **C3.1**. Relating to this, we also added other various bits of detail to the islands, such as grass, rocks, and a fort specifically on the final boss' island. While this does not affect the gameplay, it was added for decoration to make the islands more appealing to the player, as per **C3.1**.

Sprites
As the requirement **F4.4**, we have implemented sprites that satisfy the requirement for a land-bound object. Previously, all battles would be against another ship sprite, including college battles which are land-based, and so we changed the sprite to a fort that varies for every college. This was added in CombatScreen.java, in the giveEnemyShipSprite() procedure, where it checks if the enemy is a boss, and if so, it creates the correct format for its filename and loads it. This was a minor change, but we felt like it helps improve user experience to make them feel as if they are fighting a much harder enemy, rather than just another ship, which reaches **C3.1**.

Significant Changes:

Colleges/Departments
As per **F5.1** there are now 5 colleges and 3 departments, the added colleges were Alcuin and Halifax. The tile map was edited so that the islands were closer together, which meant

the player did not have to travel for a long time before they could reach a college/department, this satisfies **C3.1** as we feel the game play has been improved by this change. The islands also contain a fort image so that the player can interact with land-bound objects as per **F4.4**

Gold
The gold rewards in the game were changed so that they can satisfy the requirement **F7.2**, where defeating stronger enemies rewards the player with more gold. This was changed in CombatScreen.java, where once the enemy ship dies, a check has been added to see if the enemy was a boss (enemy.getIsBoss()), meaning that it is a stronger enemy, and then rewards the player depending on the result. With this, as well as increasing the gold that the player gains from defeating random encounter enemies, we felt like it balanced the game to make it more enjoyable. Additionally, the update() function in SailingScreen.java was adjusted to include a gold counter, which implements ticking gold with relation to number of colleges captured per requirement **F8.5**, with more captured colleges meaning a faster tick rate.

Points
We've changed several factors regarding the points system. First of all, the requirement **F8.4** was reached, where CombatScreen.java was changed to make a check once the enemy ship has died to see if it is a boss and thus is a harder enemy, awarding the player with more points if it was. Additionally, as per requirement **F6.3**, the player now gains twice as many points per second when they are sailing through bad weather tiles to entice them to take more risks for more rewards.

The requirements **F6.2** and **F6.4** were both met by the inclusion of a point multiplier that increases as the player's game time increases. This was done by altering SailingScreen.java to include the float "pointMulti" that is used in the timer update section of the code. It starts at one, and increases to two or three depending on how much time has passed, stopping at three as the maximum multiplier.

Combat Screen
Moved starting graphics code from constructor to own function graphicInit for visibility. Also added further comments to this section to increase understandability. Additionally, a check has been implemented when the player defeats the enemy to see if they are a boss or not. If they are, the player is rewarded with a larger amount of gold and points. Regular enemies' point and gold rewards have also been increased.

Sailing Screen
Moved UI table and message table from constructor to own functions initUITable and initMessTable add for visibility. Added further comments to this section to increase understandability. Moved tile map initialization and individual MapObject data to their own functions for visibility (initTileMap, findPlayer, setCollegeHitboxes, setRegions).

Balancing
Upgrade strengths were increased to be twice as effective and gold cost for these have been decreased so instead of the base that is raised to the power being 2, it has been lowered to

1.5 in order to make the game more enjoyable for the user and provide a better user experience as per **C3.1**.

Not Fully Implemented:

Colleges/ Departments

This requirement **F5.2** is addressed partially, (and therefore is not fully implemented) as the islands are not completely geographically accurate. We felt that the requirement aimed to address the fact that students from York would recognise if the map was not representative of the university, however did not think it was relevant to place them exactly as the real campus is. This is supported by the fact that the map is an abstraction of the campus, and therefore is easily recognisable despite not being perfectly accurate.

[1] Requirements2