Implementation Report

When beginning the implementation of the Assessment 4 changes, the inherited project was first tested to find and fix any bugs. The following report will outline the bug fixes carried out and will summarise the software modifications made when adding the Punishment cards and Postgraduate unit.
For a detailed list of all the changes made throughout Assessment 4, see the Changes table:
as2378.github.io/unlucky/files/Assessment4/Changes_Table.pdf
Updated Requirement Document: as2378.github.io/unlucky/files/Assessment4/Updated_Requirements.pdf

Bug Fixes:

1) The AI (non-human) players would become unsynchronised with the game, resulting in them using the first move of the next player's turn. This was fixed by adding restriction to the playComputerMove(), which made sure that the AI could not make a move if the game state was EndOfTurn or NULL.
2) The users could select the AI's units during the AI's turn. This was fixed by adding an if statement in the *OnMouseUpAsButton()* method within *Sector* that checks if the current player is human.
3) The back buttons of selecting torsos and legs within the minigame traversed the list of torsos/legs in the same direction as the forward buttons. This was fixed by rewriting the code for those buttons using an if statement.

Justification:

1) This bug was fixed because if the last player was an AI, then Player 1 could only move once during their turn since the AI had used Player 1's first move. This was not intended behaviour and so needed to be fixed.
2) Being able to select AI units during the AI's turn could confuse users playing the game. Also, the bug could have impacted the AI's ability to make moves.
3) The arrow buttons were fixed because when a user is scrolling through choices of torsos/legs, they would expect that the back button would traverse the choices in the opposite direction. Fixing this bug helped to improve the usability of the minigame.

Punishment Cards:

Code Changes:

1) *Card* and *CardDeck* classes have been added to the project along with *NothingCard, FreshersFluCard, LecturerStrikeCard* and *KillerHangoverCard*, which are implementations of *Card*.
2) *Game.Initialize()* has been modified to give a random punishment card to the first player at the beginning of a game.
3) *Game.NextPlayer()* has been modified to give a random punishment card to the current player at the beginning of their turn. The method also calls *CardDeck.DeactivatePunishmentCards*.
4) *GameControl.Save()* and *GameControl.Load()* have been modified to allow saving and loading of punishment cards.
5) The *Player* class has been given the attribute *punishmentCards,* a list of type *Card,* and accessors/mutators to manipulate it.
6) The *Player* class has an additional method, *ComputerPlayPunishmentCard,* which allows the AI to activate their punishment cards.

Justification:

Overall, this punishment card system has been added as part of the required Assessment 4 changes, specified by requirements F11, F12, F13 and N11. When implementing this system, it was important that the code was very modular (requirement N2) and had minimal interference with the inherited code. In change 3), *NextPlayer* calls *DeactivatePunishmentCards* because it was decided that the punishments should be temporary. Each time a player has their turn, *NextPlayer* is called and the *turnCount* values of the active cards owned by that player are decreased each turn.

Change 4) was implemented because requirement F8 states that the system shall allow for the saving and loading of punishment cards. Also, change 5) was implemented because the system needs a method of storing which cards are owned by each player, and creating a list of cards within each player was the easiest way of doing this. Change 6), which allows AI to play punishment cards, was carried out to satisfy requirement F13 because "both human and computer players must be able to activate punishment cards."

Architecture:
The punishment card system follows the concrete architecture. The *Game* only has a single *CardDeck* because the *Cards* that appear in the *CardDeck* are changed at the beginning of each player's turn, so that the player can only see their own cards. Also, restrictions within the *Player.AddPunishmentCard* method enforce the architectural requirement that each player can have up to five punishment cards in their deck.

GUI Changes:
1) A "Punishment cards" button has been added to the GUI. Every player can click on it at the beginning of their first move and see the card deck with their cards in it. The cards can be activated by clicking on them.
2) Within the card deck, unique images have been added to represent the different types of punishment card.

Justification:
The card deck GUI and "Punishment cards" button have been added to provide a mechanism for users to activate punishment cards, as explained in requirement F13. The card images have been added to improve the game's visual appearance, as well as improve playability because the images contain a description of the card's punishment.

Postgraduate Bonus:

Code Changes:
1) *Unit.Select()* has been modified. If a level-5 unit is selected, all the player's sectors are highlighted.
2) *Sector.OnMouseUpAsButtonAccessible()* has been modified to allow moving units to any of the player's sectors however far they are if the selected sector is a level-5 one.

Justification:
The postgraduate unit and postgraduate bonus have also been implemented as part of the Assessment 4 requirement changes. Level-5 units are implicitly seen as postgraduate units, meeting requirement N10, and can be 'teleported' to any sector owned by the player, which is different from undergraduate units because they can only move to adjacent sectors. This means that there is a difference in functionality between the two types of unit, and so requirement F10 is satisfied.
Also, requirement F9, "The saving and loading shall include both undergraduate and postgraduate units," is automatically satisfied. This is because Postgrad units use an existing representation of the base unit, meaning that the previous saving/loading system worked without need of change.

Architecture:
Postgraduate units are represented in the same way as undergraduate units, so no architectural changes were required. Therefore, the software remained consistent with the architecture.

GUI Changes:
1) Instead of Level-5, postgraduate units being represented by the number 5, they now have a graduate cap icon to represent them.

Justification:
This was done to make the postgraduate units look distinct from the normal, undergraduate units. Doing this improves playability because it is easy to distinguish postgrads from undergrads. This also helps satisfy requirement N8, which states, "The game map must offer a clear graphical representation of sectors, unit position, unit type and ownership of sectors."

Extra features that have been included in the software.

No additional features have been implemented as part of Assessment 4 because the team wanted to stick to the requirements as close as possible and the process of getting an extra feature approved by the client would have likely taken a long time.