Evaluation and Testing Report

Approach to evaluation

When evaluating the final state of our project, it was essential that we had met all our requirements. These requirements had previously been created based on the brief we were given at the start of the project and later modified for new assessments which meant they were up to date and comprehensive due to the number of meetings we had discussed our requirements with customers and the feedback given throughout the project. To meet the brief we were given, our project had to meet all the listed requirements.
As a group we went through our requirements and discussed whether we believed we had met them and what features in the game meant we had met the requirement, in order to have evidence for meeting the requirement explained by any member who had implemented the feature described. The results of this evaluation can be seen below. How the architecture of our game fitted the requirements can be found in the traceability spreadsheet. Another metric we used to evaluate our project was from our testing, described below. Our aim was to pass all of our tests, indicating the game works correctly and is of good quality, vital for the completeness of out project, found below.

Approach to testing

We felt working code, along with good documentation, would be of sufficient quality for the project. The results of testing can be found below.
Our N1 requirement made it necessary for our code to be kept 'consistent, readable and maintainable'. We have met our aim by including docstrings for every method and further comments where necessary throughout the game. Our code was clearly of good quality in the past based on feedback and how many teams adopted our game for assessment 3. During assessment 3, we did not follow this scheme due to the approach to testing taken by the team we inherited from. For assessment 4 we returned to using unit tests because, as explained in assessment 2 , we believe them to be more useful but also easier to carry out. The team we inherited from for assessment 4 had continued to use unit tests which meant we could expand and improve on what tests they had produced.
The majority of our testing in this project was done using unit tests. These tests were written alongside the features they are testing in order to keep on top of what needed implementing. Tests were written to test every feature in the game, each method in the game classes had at least one testing method in an accompanying testing class to test that feature (some methods were not necessary to test such as getters). Testing every aspect of the game allows us to gauge the quality of our code.
For assessment 4, the unit tests for the SaveGame class were completely rewritten as the test we inherited were not specific enough in our opinion. The previous test simply tested loading the game as a whole. This has now been updated to test saving and loading of all aspects of the game. Tests were also added for the new punishment cards. There was no

need to change previous tests as the game had only been added to, especially evident where unit level 5 was simply changed to postgrad in the UI.
New unit tests for assessment 4 are highlighted in the document below.

Updated unit tests and results: https://sepr-team-margaret.github.io/content/UnitT4.pdf

Meeting requirements

Updated requirements: https://sepr-team-margaret.github.io/content/Req4.pdf

C1 - Game completed 1/5/18
C2 - We never managed to get the game to crash
C3 - Games runs at 55 fps on average
C4 - Game is suitable for audience
C5 - Game is 54MB (final release version)
C6 - Build available for Window 10 on website <*link*>

N1 - All code has a good amount of comments that explain the code clearly, including docstrings for every method
N2 - Design and implementation was adapted for assessment 4 new requirements
N3 - Game can be player with either 4 human players or 3 human players and an AI player, selected from main menu
N4 - Map and 3d buildings on game map resemble university grounds and buildings
N5 - Sectors clear on game map with obvious borders, made even more clear with sector being colour
N6 - 4 landmark sectors on map, identified by symbols based on bonus, either attack or defence
N7 - Landmarks are placed to ensure balanced gameplay, the sports village, sport centre and central hall are significant landmarks on the university campus
N8 - Clear unit icons, coloured sectors to identify owners and card UI for stats
N9 - Random roll and bonus applied during conflict, player told results and winner
N10 - Skip turn button allows player to skip their turn
N11 - Main menu to load previous game or start new game
N12 - Our game was given to students outside computer science (who had never played the game)to get more of an average user base and they played a game with minimal input from team members who only showed them our manual <*link to manual*>, showing our game is easy to use for new players
N13 - Units move up a year after capturing a sector until 4th year then they level up to postgrad units, displayed in the mouse over GUI for each unit
N14 - 3 punishment cards are available - goose attack, hangover and industrial action

F1 - AI controlled player in 3 human player mode as the neutral player
F2 - A pro-vice chancellor is always present on the map but is not known to the players, its location is only known once a player captures the sector with the PVC in it
F3 - Mini game played once a player captures the sector with the PVC
F4 - Player who captures PVC gets base bonus and an increased bonus based on performance in minigame. We felt a multi awarding mini game would be too overpowering in the game
F5 - Players move units into adjacent sectors by clicking their unit and a nearby highlighted sector, only ever occupied by one player
F6 - Player has new unit spawn each turn if at their landmark if there is room for a new unit (not currently occupied)
F7 - Player can pause a game using the pause menu. Can also save and load a previous game after saving and quitting a game before its end
F8 - Can click punishment card before end of turn and select a card to use and against which player/unit