**CS207 Final Workload and Lessons Learned Report**

**Team I (Monday 9 am -11 am)**

**Craig Griffin, Sean Waters, Stuart Lennon, Reece Jones**

**Functionality Overview**

As a group, we managed to implement most of the functionality which was requested in the requirements gathering stage and detailed in the user stories. Listed below is the functionality we managed to implement at each sprint and subsequently the functionality that we did not:

1. During **sprint one** we successfully managed to generate a cryptogram for the user to play and allowed them to perform basic procedures to the puzzle such as inputting a letter and removing a letter. We did not manage to allow the player to select between letters and numbers as a preferred cryptogram type. An issue which caused us to lose many marks at this stage was our lack of Junit testing.
2. During **sprint two** we successfully completed all the user stories from the product backlog. Our software was able to save a game and load it back and track the stats of players of the game. To ensure we did not lose basic marks at this stage we ensured that we had thorough testing for each user story. An issue that came up at this stage was the robustness of our software. If the file containing a player saved game or the file containing the details of all players was tampered with, it would result in our software crashing.
3. During **sprint three** we successfully completed all the user stories. Our software could give the player a hint, the player could view letter frequencies and give up and be shown the correct solution. The player could also compete against there friends on a leader board. We tested most of the functionality at this stage, however, during the demonstration when showing how the hint feature worked a bug that we missed managed to crash the software. Our frequencies were also slightly off because we were taking spaces and punctuation in to consideration.

Overall, our game completes the epic. A user can play a cryptogram puzzle. There is room for improvement in our solution however functionally it works.

**Workload Distribution**

The workload was distributed fairly amongst the team. Listed below is the tasks we identified at each stage and which member of the team completed it:

1. **Iteration 0** (User Stories and class diagrams): user stories were discussed as a team by all team members during a lecture slot and then written up by **Craig** and modified by **Stuart**. Class diagrams were produced by **Sean.**
2. **Iteration 1** (generate a cryptogram, enter and remove a letter):
   1. Create basic class layout with getters and setters - **Reece**
   2. Implement an adaption of HashMap to allow for One to One mapping - **Sean**
   3. Load and select a random quote from a flat text file – **Craig**
   4. Encrypt quote – **Craig**
   5. Input and remove a letter from the puzzle – **Stuart**
   6. Display cryptogram and empty boxes for user to enter letters into – **Sean**
3. **Iteration 2** (Load/Save game and track player stats)
   1. Create Model for Player and Players – **Reece**
   2. Save and load a game – **Craig**
   3. Testing of all new features – **Stuart**
   4. Adapt game controller to update stats of a current player - **Sean**
4. **Iteration 3** (Show solution, hints, frequencies and leader board)

If doing a project like this again, it would have been hugely beneficial to utilize GitLab issue boards and branching features to more efficiently work on the same project as a team.

**Teamwork Strategy**

For any successful group project good team work and communication is key. As soon as the finalised group listings were released a group chat was created with all group members. We used Facebook Messenger for convenience but as a team we agreed that if we were to do that project again we would use Slack for its extremely handy ability to connect to git and show recent commits.

**Lessons Learned**