**AC22005 – Assignment 3 Report– C# ATM Simulator**

**Team 3**: Laura Clark, Mark Cockerill, Abz Mohamed

**Word Count: 487**

We decided to approach this assignment by splitting up the tasks between group members to evenly split the work and using a GitHub to share our work. We used the sample code given to us to get a basic version of an ATM working in a C# form and then made more sophisticated updates to the GUI, and added additional ATM functionality and threads. Once we had two ATMs running on separate threads we were then able to add in different versions for race or non-race conditions. We then split the additional functionalities of the ATM such as locking invalid accounts and being able to log in and out of different accounts, and updates to the GUI between us for the duration of the project. Once we were satisfied that we had met the requirements of the project we thoroughly tested it and resolved any errors we found.

When the program is run, there is the option to launch the non-data race or the data race version by clicking the corresponding version. A different version can be run simply by re-running the code and choosing the alternative option. The non-data race version uses semaphores to prevent the section of code where money is withdrawn being run on the two threads simultaneously, while the data-race version does not. The ATM screen prompts the user to enter account and password, validating this and locking the user out for too many attempts. Once logged in there is the option to withdraw money, view balance, or log out. The ATM can be controlled either by using the buttons on the screen or the side buttons next to the screen. There will be a time-out if the user is inactive for one minute.

Our main form of communication for this task was out Teams chat which we used to communicate throughout. To split up the tasks initially, we met in person and created a GitHub repository and set up issues. We split the initial tasks between us with one person working on creating a form, another on the GUI, and another on starting to set up the threads. One issue we did face was lack of communication between team members which led to work being duplicated as there was a miscommunication about who was doing what. This might have been easier if we had been meeting in person more regularly to discuss things. However we did resolve this issue by combining the different pieces of code and using parts of both. In the future we made use of GitHub Issues to assign people specific tasks and communicated more clearly in our Teams chat to make sure that this did not happen again. We also found that creating pull requests on a GitHub repository was the most efficient way for us to review each others work and always make sure we were working on the most up to date version of the project.