# Computer Systems 2B Assignment 3 Report

#### John Grossi, Caoilainn McCrory, Patrick Turton-Smith

## Approach

When we were first given the assignment, we agreed to meet to discuss the coursework specification together to ensure everyone understood the task we had been set. After this, we made a checklist of all the things we needed to complete. We discussed each team members strengths and assigned tasks accordingly, ensuring every member was comfortable with their task. During this process, we also discussed our approach to a couple of the more challenging aspects of the program, in particular implementing the multithreading aspect and how to fix the data race. We feel that this made us more comfortable with the assignment and meant we worked more effectively in completing it as a team.

## Problems & Solutions

Whenever we encountered an issue, we worked as a team to find a solution, meeting up if it could not be solved remotely. This maximised the work done in the available time.

We encountered several issues during our assignment, the main one being that we were unsure of how we could demonstrate the data race condition and how we could implement that demonstration. This affected large portions of the program – including withdrawals and deposits. Eventually, we solved this issue by adding a delay to emphasise the data race issue during the critical stages of each transaction.

We agreed that our first step was to modify the provided sample code to create a Windows Forms application with the functionality of an ATM with a basic GUI. This did not take long which allowed us to swiftly move on to multithreading the program – which we knew was going to be a complex part of the program. We met in labs to get these two aspects completed, which took a couple of days after tackling some small problems – one where a user was able to enter a mismatching account details and still access the account with the corresponding PIN (a clear security issue), solved by checking both account number and PIN against the database; and another issue where each ATM would directly modify the account details on the central database which was solved by returning a copy of the requested account to each ATM rather than just a reference.

## Features

We decided to make our ATM as authentic as possible, both by making the GUI look like a real ATM and by adding other services available at many ATM’s. The ATM’s contact a central database class to get the account information in order to carry out the transactions.

The final solution incorporates PIN validation, with the account being blocked after three incorrect attempts, which involved adding a ‘blocked’ field to the Account class. There is a check box on the login screen to toggle whether the data race problem should be demonstrated or whether the fix should run instead. We implemented the fix by using semaphores to control access to the critical sections of the program, such as during withdrawals, deposits and when changing the PIN.

We implemented and extended the withdrawals feature so that you can either choose one of a set of predetermined amounts (£10, £20, £40, £100, £500) or the user can type in a specific amount to withdraw, with checks to ensure there are enough funds in the account.

As an extra feature, we added the ability to deposit money into the account, with the user manually entering the amount to deposit.

Word count: 560 (excl. title and names)

unfortunately the user could lie about the amount with ATM’s in real life automatically identifying the amount to deposit based on the notes inserted.

and the ability to change the account’s PIN number.

We would discuss them in our group chat to notify the other members that we were have difficulties with parts we had been assigned. If the problem could not be solved though this we would all approach it together in the next lab. This worked out well and proved to be effective to make sure we could complete as much of the project as we could. As previously mentioned we had problems with our 1D arrays holding certain information from memory. So this was one of the things we all attempted in person. After looking at the code the person had done so far and researching online for inspiration we finally decide on an approach on how to achieve this. It was to use vectors (instead of using arrays like we were before) this helped greatly as they were able to be read into methods far easier and had lots of useful functions. This is what we decided would be the best solution and the person who was assigned this continued and completed this after the lab. Another problem that we had to tackle was also previously mentioned in the task of making our decimal to binary method to work. We figured out the program was reading the binary the wrong way around. So when we expected say “1” to be returned we actually received “16”. This was an easy fix when we noticed as we just had to make it read the binary the correct way around. As mentioned before we were having trouble knowing how to only extract what we needed from the assembler language file. We finally noticed that the file was separated by semi-colons so were able to use a delimiter to just extract the pieces of information before them as this was all we needed for the assembler to work. One of the best solutions we found to making sure all our code would work together and know what everyone had done was GitHub. It was easy to know if things weren’t going to be compatible and easier to share work we had completed. The other small problems we were able to solve by asking in the chat and researching (online and in the lecture notes) which ended with finding commands that we had either forgotten or did not know about.

--- - two random small problems

--- - 1st problem (bigger) 1)arrays to vectors 2)binary to decimal wrong way 3) separating the strings using for loop going through vector of strings, now ittarate through every word and select indivi depending on line numbers

--- - 2nd problem (smaller)

Word count: 600. needs to be 600-800 words. Needs to add a third problem to fit the word count