# 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: 576