# Team members

Mitchell Sugden – N9136321  
Michael Lee – N8433518

Both students have contributed equally to the project.

# Statement of Completeness

|  |  |  |
| --- | --- | --- |
| TASK 1 |  |  |
| FUNCTIONALITY |  |  |
|  | Attempted | Completed |
| Client/server communication using BSD Sockets | Yes | Yes |
| Server command line parameter – configurable port & default port | Yes | Yes |
| Server loads & tokenises textfiles correctly | Yes | Yes |
| Server authenticates client using data in “*Authentication.txt”* file | Yes | Yes |
| Server exits gracefully upon receiving SIGNAL (ctrl + c) | Yes | Yes |
| Client command line parameters | Yes | Yes |
| Client menu driven implementation / Input Validation | Yes | Yes |
| Client exits gracefully when user selects option from menu | Yes | Yes |
| Client displays transaction details as per assignment specifications | Yes | Yes |
| Client displays transactions as per assignment specifications – single client | Yes | Yes |
|  |  |  |
| TASK 2 |  |  |
| FUNCTIONALITY |  |  |
|  | Attempted | Completed |
| Efficient Multithreaded implementation – Maximum of 10 concurrent connections | Yes | Yes |
| Efficient use of Synchronisation primitives & Shared data structure(s) | Yes | No |
| ATM client displays transactional details as per assignment specifications – multiple clients | Yes | Yes |
|  |  |  |
| TASK 3 |  |  |
| FUNCTIONALITY |  |  |
|  | Attempted | Completed |
| Thread pool management – Maximum of 10 concurrent connections | No | No |
| Synchronisation primitives – Shared data structure(s) | No | No |
| Synchronisation primitives – Thread pool | No | No |
| Transactional details stored in file as per specifications – multiple clients | No | No |

All functionality that has been marked as completed has been tested to be working correctly to the assignment specification. Note: We are reading the Authentication file name as ‘Authentication(1).txt’, this is how it appears on Blackboard so we assumed this is correct.

# Synchronization Primitives

Not implemented.

# Data Structures

## Client Details

We have stored client details in structs to allow easy access to variables. Having global structs for each client means that we can easily read account numbers when performing transactions.

## Accounts

Our accounts have been stored into a struct to allow easy access to values. This means that we can easily manipulate and read balances from each account.

## Transactions

We are storing our transactions in a singly linked list. This structure has been used to allow easy insertion of new transactions. We are adding new transactions to the end of the list to ensure that they are displayed in the correct order (newest transactions at the bottom). A linked list is also easy to iterate through and sort.

## Authentication

For authentication clients, we are simply reading the file into an array and iteration through to find the username and corresponding PIN. The PIN is always next to the username so this has proven to be a simple solution.