*Provisional*

CS3004 Network Computing

Assessment/Coursework for 2021/22

**Table of Contents**

[Main Objective of the assessment 1](#_Toc527913751)

[Description of the Assessment 1](#_Toc527913752)

[Learning Outcomes and Marking Criteria 3](#_Toc527913753)

[Format of the Assessment 4](#_Toc527913754)

[Submission Instructions 4](#_Toc527913755)

[Avoiding Plagiarism 4](#_Toc527913756)

[Late Coursework 5](#_Toc527913757)

|  |  |
| --- | --- |
| **Assessment Title** | CS3004 Network Computing |
| **Module Leader** | Simon Taylor |
| **Distribution Date** | 18-10-21 |
| **Submission Deadline** | 03-12-21 |
| **Feedback by** | 17-12-21 |
| **Contribution to overall module assessment** | Threshold-based assessment (see below) |
| **Indicative student time working on assessment** | 35 Hours |
| **Word or Page Limit (if applicable)** | 3000 Words/Pages (not including references) suggested |
| **Assessment Type (individual or group)** | Individual |

# Main Objective of the assessment

The main objectives of this assessment are to create and report on a network computing application as described below. These objectives satisfy all the learning outcomes of the module as re-stated below:

* Demonstrate a thorough understanding of the main issues related to network computing
* Critically evaluate requirements and problems that arise when advanced network computing applications are designed and implemented
* Demonstrate practical ability in implementing an advanced network computing application

# Description of the Assessment

Read carefully the following scenario then follow the instructions given to complete the task.

***The WLFB Bank Application***

**Scenario and Instructions for Completing the Task**

The primary goal of this assignment is to develop a demonstrator of a concurrent client-server system that shows how transactions issued by bank application clients correctly update the data held by the banking application server. Remember this is a demonstrator to show that you understand issues in network computing (not banking!)

***The WLFB Bank Application***

The WLFB Bank Application (WBA) that you will design and implement is very simple as it is only used by three people at the moment (it’s a small bank!) The amount of money that each user has is represented in the server by a variable (i.e., there are three variables in the server!) The users are very busy and (1) frequently add and subtract money from their account and (2) transfer money to other user’s accounts (i.e., all these operations run at the server). Each user has their own client and they use their client to instruct the server to run the operations. You can call the clients A, B and C. The operations are as follows:

Add\_money(account, value) – this adds *value* (virtual) money to the specified account (i.e. the user’s account – you can assume that they only add money to their own account).

Subtract\_money(account, value) – this subtracts *value* (virtual) money from the specified account (i.e. the user’s account – you can assume that they only subtract money from their own account).

Transfer\_money(account1, account2, value) – this transfers *value* (virtual) money from account1 to account2.

The users start with 1000 units[[1]](#footnote-1) in their accounts. Accounts can go below zero.

You are required to create a multi-threaded client-server system that uses locking and has:

* One client used by each user to run the operations at the server as above (three in total)
* One server that holds the account variables and executes the operations as instructed by the clients

Demonstrate that the three clients can add/subtract money to and from their own accounts. Also demonstrate two transfer money actions between two different sets of (e.g., A/B and A/C). As you will need to show how your application works, you might also consider implementing simple log files. You are not required to implement any kind of user interface (so don’t!) and you will not get any credit for it (you have other assignments to work on).

**NOTE THAT DUE TO PROBLEMS IN PREVIOUS YEARS YOU MUST DEVELOP THIS IN JAVA USING THE SOCKET AND THREAD APIs AS PRESENTED IN THE TUTORIALS.**

# Format of the Assessment

Students are expected to submit one report individually. The format is as follows:

***Title page****:*

Be sure to include your student ID number, the title of the assignment, the name, module code and year of the module and the submission date.

***Body:***

You must use the following format:

1. Introduction.

2. Requirements

3. Design (Architecture and Protocol Sequence)

4. Implementation

5. Testing

6. Conclusions

Appendix A: Code

The sections are as follows:

**Introduction**

This section gives a short overview of your assignment.

**Requirements**

Briefly specify the requirements of your program with a short justification of how these relate to the assignment.

**Design**

This should be followed by the design discussion. This should capture the overall architecture of your system (the processes and their communication links) and the design of your program that clearly shows relevant details for the two types of client and the one server as well as the **protocol** (the sequence of messages between the clients and the server and message format). You can use one of the techniques in the design and implementation lectures to document this (e.g. algorithm, protocol table, etc.) It is critical that this section discusses how you created the design of this network computer application so that the marker can understand the approach that you took. A protocol table can be used to specify the protocol.

**Implementation**

Using code snippets discuss how you implemented the design. Rather than being a description of the code, you should concentrate on how key elements of the design were implemented. The purpose of the section is to demonstrate that you understand how a network computing application can be implemented.

**Testing**

Show the following in your testing section (supported by screen shots):

* 2 x successful execution of Add\_money between one client and the server (two shots – one for the client, one for the server) with different clients (e.g., one example with A and one with B)
* 2 x successful execution of Subtract\_money between one client and the server (two shots – one for the client, one for the server) with different clients (e.g., one example with A and one with C)
* 2x successful execution of Transfer\_money between one client and the server (two shots – one for the client, one for the server) with different clients (e.g., one example with A transferring between A and B, one with C transferring between B and C)

Remember that these should show the connection, the protocol, correct locking and updates of relevant values.

**Conclusions**

A short summary to close your report.

**Appendix A**

A formatted listing of all your code with a short commentary describing each class.

***Word Limit:***

The suggested word limit for the main text of the report to be around 3000 words. There is no penalty for exceeding the word limit but it is strongly recommended to stay within these limits as you have a finite time in which to complete this report.

# Learning Outcomes and Marking Criteria

These objectives satisfy all the learning outcomes of the module as re-stated below:

LO1: Demonstrate a thorough understanding of the main issues related to network computing

LO2: Critically evaluate requirements and problems that arise when advanced network computing applications are designed and implemented

LO3: Demonstrate practical ability in implementing an advanced network computing application

It does this by giving you the opportunity to understand how to apply the main issues of network computing to a practical application and to critically evaluate those through a report. For this assignment (Task 1) the LOs are marked together as follows:

**Task 1: Threshold Coursework** – The coursework specification will be distributed on the date shown above with a submission deadline also noted above. The assessment will require you to design and implement a network computing application as described above. There will be structured opportunities to get advice on the assessment between it being set and you submitting during the workshops.

The assessment of Task 1, which will be undertaken in Term 1, will be confirmed by the Panel of Examiners and, if necessary, the Board of Examiners will offer a reassessment of Task 1 in Term 2, subject to the re-assessment limitations of SR2.

**A student who fails to achieve grade D- in Task 1 at both the first and second attempt will not be eligible for any further assessment/re-assessment in the module.**

|  |  |
| --- | --- |
| **Learning outcomes for the assessment** | **Assessment and marking criteria** |
| There is no evidence that the threshold requirements could be met by the work presented. | F grade |
| There is some evidence that the threshold requirements are close to being met, however, fall short in attaining the expected standard, e.g. failure to create a working network computing application and no clear explanation of how the application works. | E grade |
| Threshold (D): The student has built and demonstrated a working network computing application that satisfies the functionality described in the assignment and has produced an adequate report. | D- grade |

# Submission Instructions

You must submit your coursework as a PDF file on Wiseflow by \*\*\*\*\*\*\*\*\* at 11am. You can follow the link to Wiseflow through the module’s section on Blackboard Learn or login in directly at https://uk.wiseflow.net/brunel. The name of your file should follow the normal convention set out in the student handbook, and must therefore include your student ID number (e.g., 0612345.pdf). It can also include the module code (e.g., CS2001\_0612345.pdf).

# Avoiding Academic Misconduct

Before working on and then submitting your coursework, please ensure that you understand the meaning of [plagiarism, collusion](https://www.brunel.ac.uk/life/library/SubjectSupport/Plagiarism), and cheating (including [contract cheating](https://www.brunel.ac.uk/about/quality-assurance/documents/pdf/Contract-Cheating-Guidance-for-Students-v1.pdf)) and the seriousness of these offences. Academic misconduct is serious and being found guilty of it results in penalties that can reduce the class of your degree and may lead to you being expelled from the University. Information on what constitutes academic misconduct and the potential consequences for students can be found in [Senate Regulation 6](https://www.brunel.ac.uk/about/documents/pdf/Senate-Regulation-6-2020-07-01.pdf).

You may also find it useful to read this [PowerPoint presentation](https://blackboard.brunel.ac.uk/bbcswebdav/pid-1426008-dt-content-rid-6875120_1/xid-6875120_1) which explains, in plain English, the different kinds of misconduct, how to avoid (even accidently) committing them, how we detect misconduct, and the common reasons that students give for engaging in such activities.

If you are experiencing difficulties with any part of your studies, remember there is always help available:

* Speak to your personal tutor. If you’re not sure who your tutor is, please ask the Taught Programmes Office ([TPOcomputerscience@brunel.ac.uk](mailto:TPOcomputerscience@brunel.ac.uk)).
* Alternatively, if you prefer to speak to someone outside of the Department you can contact the [Student Support and Welfare](https://www.brunel.ac.uk/life/supporting-you) team.

# Late Coursework

The clear expectation is that you will submit your coursework by the submission deadline stated in the study guide. In line with the University’s policy on the late submission of coursework (revised in July 2016), coursework submitted up to 48 hours late will be accepted, but capped at a threshold pass (D- for undergraduate or C- for postgraduate). Work submitted over 48 hours after the stated deadline will automatically be given a fail grade (F).

Please refer to the [Computer Science student information pages](https://students.brunel.ac.uk/study/cedps/welcome-to-computer-science) and the [Coursework Submission Procedure](https://students.brunel.ac.uk/study/coursework-submission-procedure) pages for information on submitting late work, penalties applied and procedures in the case of Extenuating circumstances.

1. A “Unit” is an arbitrary and artificial currency. [↑](#footnote-ref-1)