Department of Computing
School of Mathematical, Physical and Computational Sciences

Assessed Coursework Set

Module Title: Blockchain and Security
Module Code: CS3BC
Lecturer responsible: Prof. Atta Badii
Teaching Assistants: Dr Udeni Jayasinghe, Dr Asad Hussain, Mr. Abdullah Sarwar
Type of Assignment: Coursework
Individual/group Assignment:
Individual **Weighting of the**
Assignment: 50%
Page limit/Word count: 7 pages excluding the appendix (screen-shots and/or code segments)
Expected time spent on this assignment: 20 hrs
Items to be Submitted: 1 report plus link of GitLab or similar repository for codes
Work to be submitted online via Blackboard Ultra by: 12:00 noon, Monday 19th of May 2025
Artificial Intelligence Tools May be used to support work

NOTES

By submitting this work, you are certifying that you have read the assessment guidelines, which are displayed in the folder of Assessment on the Blackboard course for this module, and that you have conformed to and understand the associated policies and practices, including those on:

- Submitting your own work, not that of other people or systems (including those using artificial intelligence), and the associated penalties for Academic Misconduct
- Submitting by the specified deadline, and the penalties associated with late submission (if allowed)
- The exceptional circumstances system
- For students with relevant needs, attaching with a green sticker

Assessment classification

First Class (>= 70%)
Outstanding quality correct and highly efficient solution with excellent analysis and rationalisation of methods used and tested with the approach to validation of correctness and efficiency thoroughly justified; excellent quality and presentation of the technical report in terms of clarity, coherence, and other aspects of the presentation such as English grammar.
Upper Second (60-69%)
High quality correct and efficient solution with good analysis and rationalisation of methods used and tested with the approach to validation of correctness and efficiency well-justified; high quality of presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar.
Lower Second (50-59%)
Good quality correct and moderately efficient solution with reasonable analysis and rationalisation of methods used and with the approach to validation of correctness and efficiency reasonably justified; good quality and presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar.
Third (40-49%)
satisfactory solutions with moderate/low quality and efficiency but adequate analysis and rationalisation of methods used and tested with the approach to validation of correctness adequately justified; satisfactory presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar but with some shortcomings.
Pass (30-39%)
Basic (minimally acceptable quality) solution with barely adequate analysis and rationalisation of methods used and tested with the approach to validation of correctness not fully justified, adequate quality and presentation of the technical report in terms of clarity, coherence, and other aspects of presentation but with many shortcomings.
Fail (<30%)
Inadequate solution with inadequate analysis and rationalisation of methods used and tested; inadequate or missing validation of correctness of the approach justified; inadequate quality and presentation of the technical report lacking in clarity, coherence, and other aspects of presentation- below the minimum acceptable standards.

Assignment implementation requirements

C#, Visual Studio (v# 4.7.2): https://visualstudio.microsoft.com/vs/community/
Code Project: BlockchainAssignment.zip (downloaded from the CS3BC Blackboard “Assessment” → “Assignment Starter Resources” folder Titled: Practical and Assignment Application Starter Project/Initial Implementation Environment)

Assignment Submission requirements

The following information is required on the front sheet of your submission:
Module Code: CS3BC
Assignment report Title: Blockchain Coursework Assignment
Student Number (e.g.,25098635):
Date (when the work was completed):
Actual hrs spent for the assignment:
Which Artificial Intelligence tools used (if applicable):

Practical Guided Exercises and Assignment

This integrated Practical's & Coursework Assignment set for this module is intended to enable every student to implement a basic offline Blockchain. The practicals are structured to allow guided step-by-step development of the required components for the basic blockchain implementation.

It is expected that you will complete and document the first 5 parts of the Coursework Assignment as a structured sequence of 5 exercise steps as supported through step-by-step lab demonstrations as illustrated in the practicals support document which is called "Practical Exercises and Coursework Support Lab Script" and that you can download from:

Blackboard ⟶ Modules ⟶ Blockchain and Security

And in the assessment Tab:
Content → Assessment → Assignment Starter Resources

The assignment task (**Part 6)** is based on the application that you will have built through the first 5 parts of the practical lab exercises. For your submission you will submit your solution for the first 5 parts as worked out through demonstrated lab sessions plus your solution to the three tasks out of 4 of the following 4 tasks that form Part 6.

To earn **full marks for implementation** (**15**) for this **Part 6** of this coursework you only need to submit correct implementations of **three** out of **four** of these assignment tasks. As you can see in the below marking scheme table, this **Part 6 carries 35 marks overall**, **15 for implementation and 20 for the report**. **You can earn the full 20 marks for the report section if you only implement three sections**. However, you may be able to pick up lost marks due to incomplete implementation of any three if you choose to implement all four, although in any case, **for Part 6, one cannot obtain more than the maximum allowed marks of 35.**

Research and reference to other Blockchain implementations will be valued highly for the Part 6 tasks; especially if you incorporate their logic into your solution.

Marks for the assignment are allocated as follows:

Coursework Parts	**Marks for evidence of implementation**	**Marks for the Report**	**Total Marks**
Part 1 – Project Setup	3	2	**5**
Part 2 – Blocks and the Blockchain	5	5	**10**
Part 3 – Transactions and Digital Signatures	5	5	**10**
Part 4 – Consensus Algorithms (Proof-of-Work)	10	10	**20**
Part 5 – Validation	10	10	**20**
Part 6 – Assignment Tasks (1-4) Equally divided	15	20	**35**
	48	**52**	**(100)**

For each part in this coursework assignment, you are expected to submit evidence of implementation alongside a report which details your understanding of the steps. Evidence of implementation can be provided by screenshots of results for parts 1-5 and by a link to the code to be made accessible to the assessors.

To reconfirm: For this Part 6 (the assignment), you are expected to complete 3 out of 4 of the tasks to achieve the maximum of 35 marks. However, in cases where a student may have attempted all four tasks, but not scoring the maximum on each task, the fourth task is also assessed and this could raise their overall score up to a maximum of 35 marks which is the total mark that could be earned for Part 6.

The 4 tasks in Part 6 of this assignment are as follows:

Task 1: Extending the Proof-of-Work algorithm

Extending the Proof-of-Work algorithm to parallelise this task such that multiple threads are in use. Currently when a node performs Proof-of-Work to find a hash to satisfy the difficulty level, only one core in the CPU is working, the rest are idle. For the report, document the implementation and prove that there is a performance efficiency gain when using multiple threads as opposed to one.

Guidance for this task:

Take multiple samples of mining times and compare them; make use of *system.Diagnostics .Stopwatch*. Do not forget to consider the fact that threads may repeat work (hashing the same data and creating hashes that have already generated). Provide a solution that avoids such duplicated work and explain in the report how it prevents work from being repeated.

Tip: The following C# resources may help implementation: Callbacks/Delegates and Threading.

If you have problems updating the UI using threads the following line may be useful.

```
textBox1.Invoke(new Action(() => textBox1.Text += message))
```

Task 2: Adjusting the Difficulty Level in Proof-of-Work

As previously mentioned in the practicals support document, the difficulty level was a static type so far and as such did not change for each new cycle of block mining. In state-of the art Blockchain (Proof-of-Work) implementations a **dynamic difficulty level** is used. In Proof-of-Work crypto-currencies 'Block Time' is considered as the average amount of time required until the next Block is added to the chain. In Bitcoin this is 10 minutes and in Ethereum this is 10-20 seconds.

For this task, please decide your own 'Block time' (with justification provided in the report) and implement an **adaptive difficulty level algorithm**. Prove that the implementation works in the report and discuss how you developed your solution for your implementation.

Guidance for this task:

In our current implementation increasing the difficulty by one would increase the amount of work by a factor of 16. This is not suitable for dynamic difficulty level setting. You would need to consider another approach; you may wish to review the existing approaches as adopted by others.

In your report, please detail how you developed your implemented dynamic difficulty level and why you chose a particular approach while also providing evidence that it works. Also state what the duration of your 'Block Time' was and justify why.

Task 3: Implementing Alternative Mining Preference Settings

Currently, when Blocks are being formed by including transactions within them to be added to the chain, the transactions are selected from the transaction pool arbitrarily. In reality, there are many ways that a node (a miner) may wish to select transactions to mine from the transactions pool. They may wish to be altruistic and pick the transactions that have been waiting the longest. They may wish to be greedy and pick the transactions with the largest fees. They may wish to be unpredictable and pick the transaction entirely randomly. They may have their own transactions pending and choose to pick up transactions involving their address first. For this task, please implement a setting within the UI, as has been developed during the practical sessions, to enable the node to decide how it wishes to pick transactions from the pool. Include "Greedy" (highest fee first), "Altruistic" (longest wait first), "Random" and "Address Preference".

Guidance for this Task:

In your report discuss what you believe would be the optimal setting to choose and why. Also discuss how to choose to implement the settings and provide evidence that each setting works.

Task 4: Your own Task

You are welcome, within the scope of this work, to come up with your own extension/modification of the application developed through parts 1-5 of the practical sessions. Such an extension as you may design and implement for this task 4 should not be trivial; it would need to require an amount of effort comparable to what is needed to complete each of the first 3 tasks listed above.

Guidance for this Task:

Some ideas for this task 4, for example, include:

- The implementation of a different consensus algorithm (Proof-of-Stake),
- Creating multiple nodes running in a local network, automating the generation of transactions,
- Smart contracts (if you are really ambitious).

You will need to justify your choice to add any extension to the application, document your implementation and provide proof that it works as intended.

****