CSI3344 Distributed Systems

# Assignment 2: Distributed System Project & Written Report

## Objectives from the Unit Outline

- Implement a small distributed system using RPC or RMI.
- Apply algorithms to the solution of complex distributed system problems.
- Discuss the structure and functionality of distribution algorithms for distributed systems.
- Present outcomes of the research and/or development of a distributed system.

## General Information

- This is an **individual** assignment.
- The major assessment of this unit is a project that includes the development of a small distributed system implementation, a written report, and a presentation/demonstration of the project.
- The major assessment is split into two parts, each forms an assignment (i.e., Assignment 2 and Assignment 3). You should not start Assignment 3 until you have completed your Assignment 2.
- In Assignment 2, you need to complete a small programming project and write a report on the programming project.
- The submission of the assignment includes submission of the project implementation (including source code/s, and executables if applicable) and a written report.

## Due Date

See due date in Assignments section on Canvas

## Marks

50 marks - 50% of the unit mark

## Background information

A banking transfer system is used by customers to log in to their bank account, view their account balance, and submit transfer requests to send money to another account. In practice, banking systems are complex distributed systems involving multiple clients and servers, security controls, persistent storage, and asynchronous processing (e.g., notifications).

From the technical point of view, a banking transfer system is a distributed system. It typically consists of a client (front end) and one or more servers (application and database). Some operations are naturally synchronous (request/response), while others are often implemented asynchronously (messages/events) to improve reliability and scalability.

Note: No real money is involved in this assignment. This is a teaching system with mock data only.

## System overview (three-tier architecture)

You are required to design a small banking system with a three-tier structure:

- Client tier: Banking Client (BC client)

Used by customers to sign in, query balance, and submit transfer requests.

- Application tier: Bank Application Server (BAS server)

Provides authenticated APIs, implements business rules (fee calculation, validation, state transitions), and coordinates transfer processing.

- Data tier: Bank Database Server (BDB server)

Stores persistent data such as user accounts, balances, transfer records, and audit logs.
The database server can only be accessed by the BAS server (not by the client).

Communication may be synchronous (e.g., login, view balance) or asynchronous (e.g., notification). You must justify your design choice per operation.

## System components

Consider the following simplified system: it consists of one client and two servers.

Client 1 — BC client:
This is the customer-facing banking client. A user can use it to:

- sign in,
- view their balance, and
- submit a transfer request and receive a transfer result (or a transfer status).

Details of the BC client behavior and requirements can be found in the “System behaviors and requirements” section.

Server 1 — BAS server:
This is the main application server of the bank. It offers primary functions such as:

- authentication and session/token management,
- balance query,

- transfer request validation,
- transfer fee calculation,
- transfer processing (sync or async),
- logging and persistence to BDB.

Server 2 — BDB server:
This is the database server used to persist data. It stores:
- users, accounts, hashed credentials (or mock secrets),
- account balances,
- transfer records and their statuses,
- fee records and audit logs.

## Transfer fee rules (per transfer)

Each transfer is charged on its own amount using the table below. Pick the tier by the transfer amount, compute the percentage on that amount, then apply the per-transfer cap.

| Transfer amount per transfer | Percentage on this transfer | Per-transfer cap | Notes |
|---|---|---|---|
| **$0 – $2,000.00** | 0% | — | Free tier |
| **$2,000.01 – $10,000.00** | 0.25% | $20.00 | Entry tier |
| **$10,000.01 – $20,000.00** | 0.20% | $25.00 | Mid tier |
| **$20,000.01 – $50,000.00** | 0.125% | $40.00 | Upper-mid tier |
| **$50,000.01 – $100,000.00** | 0.08% | $50.00 | High tier |
| **$100,000.01 and above** | 0.05% | $100.00 | Top tier |

- Cap = maximum fee for a single transfer.
- All monetary values must be rounded to 2 decimal places.

## System behaviors and requirements

*(A) Login and authentication*

The user runs the BC client and enters username/password (mock).
The client sends a login request to BAS server.
BAS server validates the credentials (mock validation) and returns an authentication result:
- success → return a session token / auth token
- failure → reject and log the attempt

Design requirement: Explain where authentication is handled and how the token is used in subsequent APIs.

*(B) Balance query*

The BC client sends a balance query request with the user's token.
BAS server authenticates the request, reads current balance from BDB, and returns it.
This operation is typically synchronous (request/response). Explain your choice.

*(C) Submit a transfer request*

A customer provides:

- recipient account ID (or payee ID),
- transfer amount,
- an optional reference message.

The BC client allows an authenticated user to submit a transfer request (recipient, amount, optional reference). The BAS server is responsible for enforcing banking rules and maintaining system consistency.

Your design must address (and justify):

- Authentication: how the request is authenticated/authorized.
- Validation: what input checks are required and how errors are reported.
- Fee policy: how the transfer fee is computed using the fee table (per transfer, with caps).
- Consistency: how you ensure balances and transfer records remain consistent under failures and concurrent requests.
- Processing mode: whether the transfer completes synchronously or is handled asynchronously (e.g., queued settlement), and why.
- Persistence & audit: what must be stored in BDB for traceability and later queries (e.g., transfer status/history).

The server must return an outcome that is meaningful to the user (e.g., a confirmed completion or a trackable request), but the exact mechanism is part of your design.

(D) Transfer status query

The BC client can query by transfer ID:

- PENDING (accepted but not settled yet)
- COMPLETED (funds moved + fee charged)
- FAILED (e.g., insufficient funds, invalid recipient, server error; no money deducted)

(E) Persistence and audit

Every transfer must be persisted with necessary information.

This is a teaching system with mock data:

- Create and use a small set of mock users/accounts in BDB.
- No integration with real banks.
- Focus on clear tier separation, authenticated APIs, persistence, and correct fee logic.

## Recommended procedure for implementation

It is recommended that you implement the project in two phases.

- Phase 1 focuses on a minimal two-tier client–server system (BC client ↔ BAS server).
- Phase 2 upgrades the system to a three-tier architecture by adding BDB server as the database tier (BAS server ↔ BDB server). The BDB server can only be accessed by the BAS server (not by the BC client).

### Phase 1: Two-tier implementation

In this phase, implement a basic banking system using BC client + BAS server only, *without a separate database tier (BDB)*. In this phase, you may maintain the system state in-memory on the BAS server (e.g., arrays/dictionaries), i.e., implement the required behaviours without involving the BDB server.

The goal is to demonstrate a clean remote invocation workflow and correct business logic.

Scope (Phase 1)
Your system should support the required functions, such as:

- Sign-in / authentication
- Balance query
- Transfer request submission
- Transfer fee computation using the provided fee table (per transfer, with caps)

Communication style (choose one):
*(i) Synchronous (RMI/RPC-style).*
Use a request/response pattern (e.g., RMI or RPC). Allowed third-party libraries for Style (i): Pyro5 OR gRPC (and their dependencies such as grpcio and grpcio-tools). Apart from these, only the Python standard library is allowed.
*(ii) Asynchronous or hybrid.*
Use asynchronous communication (or a hybrid of synchronous + asynchronous). In this case, you may use third-party libraries/packages. You should clearly define the message/event flow and how your system behaves under failures (e.g., retry, duplicate requests).

**Phase 2: Three-tier implementation**

Start Phase 2 only after Phase 1 is working. This phase has two sub-tasks:

(1) Testing (required)
Extensively test your Phase-1 system using various inputs and edge cases (e.g., boundary values of fee tiers, insufficient funds, invalid recipient, repeated requests).

(2) Add the BDB server and upgrade to three-tier
Create BDB server as the database tier and extend the Phase-1 system so that persistent data is stored and queried via the BDB server. *In Phase 2, you should use SQLite for the database implementation.* The SQLite database must be managed by BDB server program. The BC client and BAS server must not access the database directly.

Communication rule

- BC client has no direct access to BDB server.
- BAS server ↔ BDB server communication may use RMI/RPC (or your selected mechanism), and you must justify the choice.

**Testing**

For testing, manually create mock data for at least two users and store them in the database. *You should export your database tables (e.g., as .xlsx/.csv) and submit them as a supporting document.*

You must explain how your design maintains correctness under failures (e.g., partial updates, retries, duplicate requests).

**Note**: If you complete Phase 2, you do not need to submit Phase 1 separately. Your final submission (code and report) should contain the three-tier (Phase 2) system. In your report, present the complete system design and clearly explain the role of each component you developed, including how your Phase 1 design was extended to Phase 2 (e.g., after adding the BDB server).
If you do not complete Phase 2, submit your Phase 1 (two-tier) system as the final deliverable.

## The written report

Write a project report about the project you completed. Please also refer to the 'Format Requirement of the Assignment Report' section for more details.

The main body should include (but not limited to):

- Application/Technology background.
- Application requirements.
- Application design and implementation procedure.
- User manual: application setting-up and usage steps (with possible screenshots) (Note: this is to allow your tutor to install your project code/s to their/lab computer/s for testing your project/codes).
- Test cases that can be used to test your system.
- Example snapshots demonstrating application running status and input/outputs.

Notes: (1) The project source codes are required and must be included separately in the submission zip file.
(2) When forming test cases, please consider the following points:

- How does your code/s react to correct input?
- How does your code/s react to incorrect input?
- Can you exit the program/s safely after testing a chosen task (or sub-task)?
- Do the test cases cover all primary functions of the system?
- Do all menu options function properly?
- Is it formatted well enough?

## Other basic requirements

- The application/system should include all required components.
- The observable behaviors of your application should be consistent with what is described/required.
- The application should provide necessary error handling mechanisms.
- The application must be implemented using Python.
- The system must be runnable off-the-shelf, i.e., it can be executed from an OS shell (e.g., Windows Command Prompt) with Python installed, without requiring an IDE.
- The project report should be well structured, informative, and not contain codes of your project. All code files should be included, separately, in the submitted .zip file.

# Format Requirement of the Assignment Report

| | |
|---|---|
| Report content | **Cover/title page**<br>Must show unit code, unit title, assignment title, assignment due date, student ID and name, etc. |
| | **Executive Summary**<br>This should represent a snapshot of the entire report that your tutor will browse through and should contain the most vital information requested. This part should be placed in between Cover page and ToC. |
| | **Table of Contents (ToC)**<br>This must accurately reflect the content of your assignment/report and should be generated automatically in Microsoft Word with clear page numbers. |
| | **Introduction**<br>Provide background information on the project, defining its scope and assumptions if any; Use in-text citation/reference where appropriate.<br><br>**Main body** (this should be divided in sections, each with an appropriate title)<br>This part should contain (but not limited to):<br>• Understanding of concepts/techniques involved in the report.<br>• The problems being solved.<br>• Strategies used to solve the problem (e.g., an approach/es for developing solution/s, etc.).<br>• A *User Manual* for installing your project onto (two or more) computers and running your codes on different machines.<br>• Test cases that you may use to demonstrate the project in your video demo/presentation.<br>• Comments/discussions or criticism of the solutions developed /implemented, etc.<br>• Anything else you feel is necessary to include.<br><br>**Conclusion**<br>Outcomes or summary of the works done in this assignment.<br><br>**References**<br>A list of end-text reference, if any, formatted according to the ECU requirements using the APA format (Web references are also OK). |
| Other requirements | • The length of the assignment/report should be within 4,000 words (excluding references) and 12 pages. The text must use font **Times New Roman** and **be not smaller than 12pt**.<br>• Any use of AI tools must be clearly declared in the report, including the tool name, version (if known), and the date/time and purpose of use. |

## Submission Instructions

- Submit your Assignment 2 via Canvas assessment submission facility.
- Your submission should include the assignment main document (i.e., a report) and Python source codes. The main assignment document must be in report style, in Word or PDF format (see below for detailed format requirement). Pages must be numbered. Your source code file/s must be runnable in our lab environment.
- Your submission should be with two files.
  - A compressed file (e.g., in .zip), which contains your main document/report and Python source code file/s, and any other support documents. Rename the .zip file in a format of:

    *<student ID>_< LAST NAME>_<First name>_CSI3344_A2.zip*

    *As an example, if your student ID is 12345678 and your name is Ben SMITH, the submission file should be named 12345678_SMITH_Ben_CSI3344_A2.zip*

  - A report file in .docx or .pdf format with the same naming rules as:

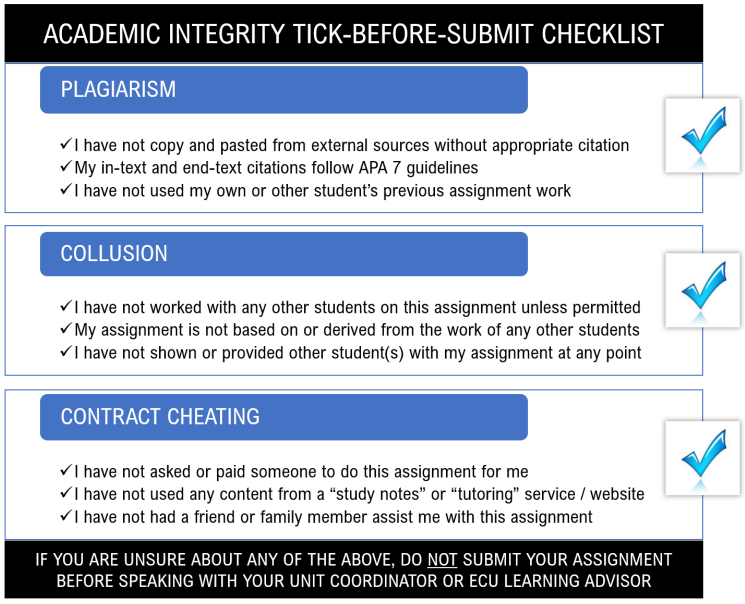
    *<student ID>_< LAST NAME>_<First name>_CSI3344_A2.<ext>*

    *where <ext> is pdf or docx.*

- No hard copy or email submission is acceptable.
- ECU rules/policies will apply for late submission of this assignment.

**Academic Integrity and Misconduct**

- This assignment is an **individual** work (not a teamwork). Your entire assignment must be your own work (unless quoted otherwise) and produced for the current instance of the unit. Any use of uncited content that was not created by you yourself constitutes *plagiarism* and is regarded as Academic Misconduct. All assignments will be submitted to plagiarism checking software, which compares your submission version with previous copies of the assignment, and the work submitted by all other students in the unit (in the current semester or previous years/semesters).
- *Never give any part of your assignment to someone else,* even after the due date or the results are released. Do not work on individual assignment with other students. – You can help someone by explaining concepts, demonstrating skills, or directing them to the relevant resources. But doing any part of the assignment for them or with them or showing them your work is inappropriate. An unacceptable level of cooperation between students on an assignment is *collusion* and is deemed an act of Academic Misconduct. If you are not sure about plagiarism, collusion or referencing, simply contact your lecturer or unit coordinator.
- You may be asked to explain and demonstrate your understanding of the work you have submitted. Your submission should accurately reflect your understanding and ability to apply the unit content and the skills learnt from this unit.
- Before submitting your assignment, make sure you have checked all items in the **Academic Integrity tick-before-submit checklist** below and watch the video by clicking the link next to the checklist image:




Watch this video before submitting your assignment

## Indicative Marking Guide

| | Description | Allocated Marks | Marks Achieved & Comments |
|---|---|---|---|
| **Phase I** | Phase 1: Two-tiered version completed | **16** | |
| **Phase II** | Phase 2: Three-tiered version completed | **16** | |
| **The Report** | Executive summary: abstraction of the report & vital information as a whole;<br>Thought/idea organization: conjunctions & cohesion;<br>Clarity: discussion flow and integrity etc.;<br>Citations to References;<br>User manual;<br>Test cases;<br>Conclusions achieved;<br>References;<br>Quality of the report: Report presented as per Format requirement;<br>Report length - not too long and too short. | **18** | |
| **Report** | Document presented as per format requirement? | * | |
| | Submitted as per submission requirement? etc. | ** | |
| **Penalty** | (*Academic misconduct? Late submission?*) | | |
| Total mark of 50 (50% of unit mark) | | **50** | |

* Also applies to each phase as an overall check on presentation and quality.
** Assessed independently of task marks; up to 2/50 of the total (2 mark) may be deducted separately for failing to meet submission requirements.