COMP2211 – NS Distributed Systems Summative Assignment

This coursework gives you an opportunity to implement a distributed replication system. There are various aspects for constructing such a system, including client and server implementation, operation workflow design, consistency control, message design, etc. For each aspect, different approaches could be chosen. You are expected to use RMI in Java/Python to implement a reliable distributed replication system based on the gossip replication architecture, which comprises at least three servers, one front-end server, supporting a client to retrieve and submit movie rating. The coursework allows you to demonstrate your competence in developing distributed systems. For movie rating dataset, you are not required to create your own one. You can use some public domain dataset, such as MovieLens. Submission deadline is 8th March 2019 (4pm). This assessment contributes 11.22% of the overall COMP2211 module marks.

Reference:

- Section 18.4: Case studies of highly available services: The gossip architecture
  George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. "Distributed Systems: Concepts and Design", Pearson, 5th edition, 2012.
- Movie rating dataset (MovieLens): https://grouplens.org/datasets/movielens/latest/

Requirements:

- Draw a simple diagram to depict the distributed replication system that you have implemented. The diagram should clearly illustrate major operation workflow flows among servers, a client and the front-end. [10 marks]

- Construct suitable replication servers to maintain movie rating information. At least 3 servers should be implemented. To allow the simulation of server availability and failure situations, each server should be able to report itself as "active", "over-loaded", or "offline" arbitrarily. [30 marks]

- Construct a suitable front-end server. It serves as the entry point for a client to access the distributed system. It also serves as the key distributed system component to maximize system availability by properly reacting to server availability. [25 marks]

- Construct a client program. It provides a handy interface for a user to retrieve, submit and update movie ratings. Since the main purpose of this coursework is to assess your competence in developing distributed systems, you are not required to develop any complicated movie rating system or user interface (e.g., text-based user interface is sufficient). [15 marks]

- Apply a suitable consistency control mechanism to support the gossip replication architecture. You are also required to include a suitable data message design to support such a mechanism. [20 marks]

Deliverables:

Develop a distributed replication system according to the above requirements using RMI in Java/Python. For submission, you should pack all program source codes and required files into a single .zip file for uploading to DUO. You should also include a readme file in the submission, showing essential instructions to run your program and providing a 200-word description to highlight the main application features you have provided.

COMP2211 – NS Distributed Systems Summative Assignment
Mark Sheet

Student ID: ____________

Total mark: ____________

Level of Achievement	System Architecture [10 marks]	Server Implementation [30 marks]	Front-end Server Implementation [25 marks]	Client Implementation [15 marks]	Consistency Control [20 marks]
81-100%	Effective architecture design providing comprehensive functionalities and high system availability	Optimal server implementation with high data integrity and wider scope of functionalities supported	Implementation supports high system availability and efficient server-client interaction	Implementation supports intuitive and effective user interaction	Implementation supports optimal consistency control and comprehensive data message design
60-80%	Good architecture design with sufficient coverage of functionalities	Robust implementation with sufficient functionalities supported	Robust implementation with sufficient functionalities supported	Robust implementation with sufficient functionalities supported	Robust Implementation with sufficient functionalities and good quality data message design
40-59%	Simple architecture design with limited functionalities supported	Simple implementation with limited functionalities supported	Simple implementation with limited functionalities supported	Simple implementation with limited functionalities supported	Simple implementation with limited functionalities and data message design
1-39%	Inadequate system architecture construction	Inadequate implementation	Inadequate implementation	Inadequate implementation	Inadequate implementation and data message design
0%	No / irrelevant submission	No / irrelevant submission	No / irrelevant submission	No / irrelevant submission	No / irrelevant submission
Marks Obtained	/10	/30	/25	/15	/20

Other comments: ____