Networked Software Development Assignment

Module Leader: Dr Abayomi(Yomi) Otebolaku		Level: 5
Module Name: Networked Software Development		Module Code: 55-502821
Assignment Title: Multithreaded Client-Server Messaging Assignment		
Individual	Weighting: 70	Magnitude: *wordcount/length of…2800*
Submission date/time: 5th-01-2023, 14:59	Blackboard submission Y Turnitin submission N	Format: source code, digital media, readme text file.
Planned feedback date: 27-01-2023	Mode of feedback: Written and verbal feedback	In-module retrieval available: No

Module Learning Outcomes

- employ multithreading to develop a simple (command-based) server with an associated GUI-based client (LO1)
- establish connection to a database and successfully execute SQL statements upon that database, returning the results of such executions (LO2)
- create a dynamic Web application, using high level, server-side programming (LO3)
- transmit multimedia files across a network and view/playing such files upon receipt (LO4)
- identify, explain, and apply the principal features of object orientation (LO5)

Assessment Brief

Introduction

In this assignment, you will design and implement a console-based multithreaded client-server messaging application for Sheffield Hallam University student community, using networked application development techniques. You will demonstrate your understanding of those techniques in the implementation of your prototype system. You are at liberty to decide and justify what kind of messaging application you want to design and implement. However, you will be provided with specific communication protocol for the messaging system (please see the server's specification below). This protocol must be strictly adhered to.

Task

A prototype of the multithreaded client-server messaging application for Sheffield Hallam University student community will be designed and implemented using client-server architecture and network software development techniques. As the architecture suggests, the prototype will consist of a multithreaded server and clients communicating over a network. The system will allow clients to publish messages, subscribe to channels, and read messages through the channels to which they have subscribed.

This assignment covers the core learning outcomes of the module as specified in the module descriptor and in the table above.

Server Specification

The server and clients will exchange message requests and responses using JSON format based on a fixed protocol specified in a separate document (included in the supplementary material). You will implement, using Java, a server that allows multiple clients to connect concurrently on port 12345 (you may choose a different port number) and that correctly responds to the requests as detailed in the protocol specification. You may assume that messages only have JSON encoded object with "from", "when" and "body" (please see the client specifications below). For the server, an optional persistence as an extension is recommended. In this regard, the server should be able to back up messages to a persistence storage such as a database or a file on the disk. If the server is killed, and restarted, it should be able to automatically recover the persisted messages.

Client Specification

The client to be implemented in Java should connect to the server above running on localhost and port number 12345 (you may choose a different port number). The client should allow a user to login to a particular channel, subscribe to and unsubscribe from other channels, publish messages, and read messages that have been published on the subscribed channels. This functionality must be implemented to communicate with servers using the JSON protocol in the server specification above. The users of your application should be able to interact with the client through a console. In addition, the application console should automatically display (without user's intervention) messages that have been published on their respective subscribed channels. For the client, you may decide

to implement an optional extension such as the capability to search messages, e.g., display all messages containing specific hashtag. Other extensions are allowed but you should discuss these with your tutors to clarify which feature counts as an extension.

Supplementary material

The folder *Assessment/Assignment* on Blackboard contains a zip archive with files that supplement this specification. These include a detailed specification of the request/response protocol and the JSON encoding. You are expected to test your client against your own server.

Assessment Criteria

Your system (both the client and server) will be assessed according to the following criteria:

C1. Core Functionality

- Working server: server compiles and runs, able to accept concurrent connections from multiple clients.
- Working client compiles and runs. Client does not crash nor deadlock, client can connect to server and communicate by exchanging information.
- Multiple clients connecting concurrently, publishing messages, subscribing to channels, unsubscribing from channels. Clients should be able to exchange at least 10 text-based messages without crash or deadlocks.
- A user-friendly console that displays messages published on subscribed channels automatically (without user intervention).

C2. Quality of your console: ease of interaction, user friendliness, allowing users with menu to perform main functionalities of the system, etc.

C3. Demonstration: Demonstration via recorded video/screencast (not more than 15 minutes) of your prototype's core functionalities and extensions (if implemented). Demonstration of system built explaining different aspects of core functionalities and extensions and any underpinning knowledge used in the process.

C4. Extensions

- If you implement persistence, your server should be able to automatically retrieve persisted messages when killed.
- If you implement message searching or any other functionality, this should be demonstrated.

C5. README File, the readme file should contain detailed descriptions of the contents of your submitted archives. It should also contain instructions on how to compile and run your application either from your chosen IDE or from the console.

	FAIL (insufficient)				THIRD (sufficient)			LOWER SECOND (good)			UPPER SECOND (very good)			FIRST (excellent)				
	Zero	Low Fail	Mid Fail	Marginal Fail	Low 3rd	Mid 3rd	High 3rd	Low 2.2	Mid 2.2	High 2.2	Low 2.1	Mid 2.1	High 2.1	Low 1st	Mid 1st	High 1st	Exceptional 1st	Perfect 1st
Criteria and weighting	**<19**		**20-39**		**40-49**			**50-59**			**60–69**			**70-84**		**85+**		
C1 60%	Server and client do not compile		Server and client compile, one runs, the other does not run. Server of client crashes		Server and client compile and run, server can accept concurrent connections with multiple clients, synchronisation is implemented, server or client does not crash			Server and client compile and run, server can accept concurrent connections from multiple clients, server and clients follow the specified protocol.			compile and run, server can accept concurrent connections from multiple clients, server, server, and clients follow the specified protocol, server can handle malformed requests.			compile and run, server can accept concurrent connections from multiple clients, server, server, and clients follow the specified protocol, server can handle malformed requests, server is able to handle additional message fields e.g. For receiving information from the server, well implemented client console.		compile and run, server can accept concurrent connections from multiple clients. Server, and clients follow the specified protocol, server can handle malformed requests, server is able to handle additional message fields e.g. for receiving information from the server, server can gracefully recover when killed, recovering all existing messages, well implemented client console menu, clients able to seamlessly communicate with other clients		
✓																		
C2 20%	Client console menu not implemented		Basic console implemented, giving limited options to users to interact with the system.		Client console menu implemented with some good options for users to interact with the application. Some issues such as not being able to navigate back to the initial options on the menu, etc.			Client console menu implemented with some good options for users to interact with the application. Users able to navigate back to the initial options on the menu, etc.			Client console menu implemented with some good options for users to interact with the application. Users able to navigate back to the initial options on the menu, novice users can easily interact with little or no glitches.			Very good Client console implemented, very good and intuitive menus options allowing users not only easy navigation of options but allowing to quit and restart the client application, very intuitive from an average user perspective.		Excellent, professional looking Client console implemented excellent in terms of user navigation of all options and functionalities of the application, excellent console menu in terms of look and feel, able to allow different users to login and log out of the application.		

✓							
C3 10%	No demonstration video submitted	Video demo /screen cast submitted but limited. A one- or two-minute demo video is limited	Video demo /screen cast submitted, demonstrates some of the core functionality of the system.	Video demo /screen cast submitted, demonstrates core functionality of the system, explains how the implemented system meets key assessment criteria	Video demo /screen cast submitted, demonstrates core functionality of the system, explains how the implemented system meets key assessment criteria, justification of key implementation decisions	Video demo /screen cast submitted, demonstrates core functionality of the system, explains how the implemented system meets key assessment criteria, justification of key implementation decisions, very good video quality	Excellent video demo /screen cast submitted, demonstrates core functionality of the system, explains how the implemented system meets key assessment criteria, justification of key implementation decisions, outstanding video quality.
✓							
C4 5%	Extension not implemented	At least an extension (server or client) is implemented but does not work as intended.	At least an extension (server or client) is implemented, and work as intended.	A good extension is implemented and works as intended	Client and server extension implemented, and work as intended; extension could be improved/enhanced	Excellent client and server extensions implemented; they work as intended	Excellent extensions such as those suggested in the brief. Additional extension(s) implemented.
✓							
C5 5%	No README file	README file submitted but no description of the contents of archive provided	README file submitted but limited description of the contents of archive provided	README file submitted, good description of the contents of archive provided, instruction on how to compile and run your system is provided	README file submitted good description of the contents of archive provided, instruction on how to compile and run your system is provided	README file submitted, excellent description of the contents of archive provided, instruction on how to compile and run your system is provided	Outstanding README file submitted. Excellent and succinct descriptions of archive contents, excellent instructions on how to compile server and client provided.
Overall mark: 100							

Submission Instructions

You must submit electronically to the module site on Blackboard by Thursday, 5th of January 2023, 2:59PM

You will find the submission point Coursework in folder Assessment. You must submit a zip archive with the following content

- all source files of your server,
- all jar files necessary to compile and run your server, and
- a README file detailing
  - the contents of your archive (one sentence per file suffices);
  - whether you implemented the persistence and chat searching extensions.
  - how to compile your server on the Java command line/IDE.

  - how to run your client from the Java command line/IDE.
- Demonstration Video

You must submit a demonstration video showing your running system. Please look at the marking criteria and prepare your demonstration accordingly. In case you face any challenge in video recording, please run your system and provide screenshot supporting the marking criteria,

****

All work must be your own. If evidence of collusion/copying is found, then such collusion will be penalised, severely if appropriate! If there is some doubt about the authenticity of a particular piece of work, then the person submitting it will be expected to give a detailed explanation of such work, including reasons for the programming decisions taken.

****