**COMP 90015 Distributed System**

**Assignment 1 Report**

Given Name: Jiali

Surname: Ying

Student ID: 1346717

**Introduction**

The multi-threaded Dictionary is designed for users to search, add and remove words efficiently. The system uses Transmission Control Protocol(TCP) for reliable communication between the client and the server and implements a worker pool architecture to manage threads. The worker pool architecture improves performance by reducing the overhead of creating and destroying threads for each request. The dictionary also provides a graphical user interface (GUI) to make interactions more user-friendly.

**Architecture**

The server implements a **worker pool architecture**. A pool of 10 worker threads handles incoming requests. This architecture is chosen because it helps to manage high concurrency requests effectively while preventing thread exhaustion. Compared to other models like thread-per-request or thread-per-connection, the worker pool model minimises thread creation and destruction overhead, leading to better resource utilization and improved performance.

**Communication Method**

All communication between the client and the server is achieved through **TCP** sockets. TCP is chosen because it is a reliable and connection-oriented communication protocol that ensures data is transmitted in the correct order and without loss. Unlike UDP, TCP provides automatic error recovery, making it more suitable for our system where data integrity and reliability are crucial.

**Class Design**

The system includes several key classes:

• **DictionaryClient**

Connects to the dictionary sends user inputs, and receives the processed results from the server.

• **DictionaryServer**

Runs the server, listens for incoming client connections, and handles multiple requests concurrently.

• **DictionaryGUI**

Provides a user interface for interacting with the dictionary. It displays a headline, an input field, a search button, the results area and the instruction areas.

• **Processor**

Handles all user input processing. It takes commands from the client, updates the dictionary, and sends appropriate responses back to the client.

**Message exchange protocol**

• **DictionaryClient** is started using the command:

java –jar DictionaryServer.jar <port> <dictionary-file>

• **DictionaryServer** is started using the command:

java –jar DictionaryClient.jar <server-address> <server-port>

• **User input**

1. **QUERY word**: Searches for the meaning of an existing word.

2. **ADD word [meaning]**: Adds a new word with one or more meanings.

3. **REMOVE word**: Removes a word and its meanings.

4. **ADDITIONAL word [meaning]**: Adds additional meanings to an existing word.

5. **UPDATE word [old meaning] [new meaning]**:

Updates an existing meaning for a word.

This approach was chosen because:

1. **Easy to parse**: The consistent format makes it easier to parse the user input. With the predefined structure for all commands, the server can correctly split and identify different parts of the input. Without the standardized structure, it would be difficult to split different parts of the user input, leading to errors.

2. **User-Friendly**: The format is intuitive for users as it clearly shows different types of supported commands and relevant formats. This reduces the chance of mistakes or confusion and the user can enter commands following a simple and recognizable pattern. If meanings include “ ”, the user will need to replace it with ‘ ’. If users don’t use [] for adding meanings, they can still be read if meanings are in “”.

**Failure Model**

The server is designed with robust error-handling mechanisms. It can detect and report various errors such as input errors, network communication failures, I/O issues, and word processing errors. These errors are caught and appropriate error messages are returned to the client to ensure the system operates smoothly and users are informed about issues.

A screenshot of a dictionary

AI-generated content may be incorrect. A screenshot of a dictionary

AI-generated content may be incorrect.

***Figure 1*** *Network communication failures* ***Figure 2*** *Invalid input format*

A screenshot of a dictionary

AI-generated content may be incorrect. A screenshot of a dictionary

AI-generated content may be incorrect.

***Figure 3*** *Duplicate word error* ***Figure 4*** *Word processing error*

**Supported Functions**

1. **Query the meaning(s)**

• Command: QUERY word

• Example: QUERY apple

• If the word is not found, an error message is returned.

2. **Add a new word and meaning(s)**

• Command: ADD word [meaning]

• Example: ADD grape ["purple fruit”]

• Example: ADD grape ["purple fruit", "sweet”]

• If users did not use [], the command can still be read if the meaning is in “ ”.

• If the word already exists or the input format is incorrect, an error message is returned.

3. **Remove an existing word**

• Command: REMOVE word

• Example: REMOVE apple

• If the word is not found, an error message is returned.

4. **Add additional meaning(s) to an existing word**

• Command: ADDITIONAL word [meaning]

• Example: ADDITIONAL apple ["red fruit”]

• If users did not use [], the command can still be read if the meaning is in “ ”.

• If the word does not exist or the input format is incorrect, an error message is returned.

5. **Update existing meaning of a word**

• Command: UPDATE word [one existing meaning] [one new meaning]

• Example: UPDATE apple ["fruit"] ["red fruit”]

• If the word or old meaning does not exist, an error message is returned.

**Data persistence**

The dictionary does not implement data persistence, which means all the user modifications will not be updated in the input dictionary file and are only valid within the current session. The modification will be lost upon server restart. The decision is based on the following considerations:

1. **Security**

As the dictionary allows multiple users to connect and use instead of a personal dictionary, freely modifying meanings and adding or deleting words can lead to security risks. Malicious users could exploit the system to add inappropriate content, such as sensitive information or fraudulent data. This could compromise the integrity of the system.

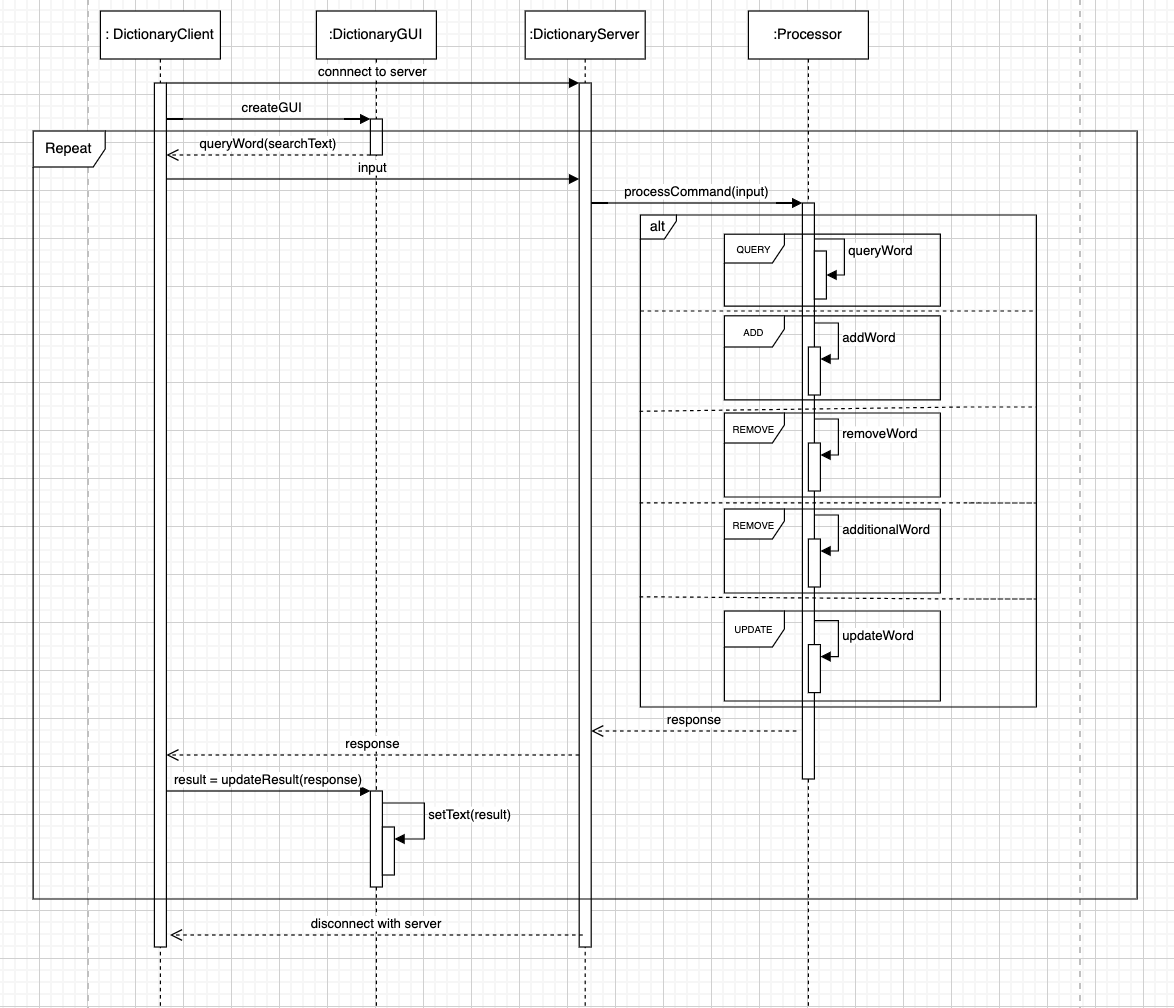
2. **Lack of reviewing system**

The current design does not have a review or approval system to review user modifications. This leads to the risk of unauthorised changes, especially in a system that allows multiple users to modify content. This makes persistence dangerous without additional safeguards.

3. **Scalability Considerations**

While the system does not support data persistence now, it could be extended in the future to include such functionality if the need arises. Implementing persistence (e.g., using a database) will not affect the core features of the system and can be added as a future enhancement.

**Interaction Diagram**

**

***Figure 5*** *Interaction Diagram – Sequence Diagram*

**GUI**

The GUI contains a Headline, Search Bar, Search Button, Result Area and Supported Commands.

• **Headline**: displays “Dictionary” to highlight the application’s purpose.

• **Search** **Bar**: Allows users to enter input.

• **Search** **Button**: When the button is clicked, the message in the search bar will be sent to DictionaryClient for processing.

• **Result** **Area**: Displays the result of the user’s command. It can be error messages, success messages or the meaning of a word.

**A screenshot of a dictionary

AI-generated content may be incorrect.**

***Figure 6*** *Interaction Diagram – Sequence Diagram*

**Critical Analysis of the Work Done**

• Choice of Worker Pool Architecture

Worker pool Architecture is used to handle concurrent requests because it can effectively reduce the overhead of creating and destroying threads for each request. This improves system performance and resource management compared to other methods like thread-per-request.

• Future scalability and improvements

Data persistence is currently not implemented because of security concerns, as there is no review process for all changes to the dictionary. However, it could be added in the future with appropriate safety methods. This would allow a persistent storage of data.

• Use of TCP

TCP is used for communication because the system requires reliable message transmission. TCP ensures that all client-server communication is error-free and in the correct order. Although TCP may be slower than UDP, its reliability outweighs the drawbacks in this case.

• GUI Design and Layout Choices

The GUI is aimed to be simple and user-friendly. The search bar is designed to stretch with the size of the GUI so that when the window size is too large or small the search bar could still have an appropriate size. The result area will show one output meaning or instruction per line to make the output clear to see. Supported commands are also listed in different colours, with explanations and examples to ensure that users can understand the format easily.

• Safe Concurrent Operations

In order to achieve safe concurrent operations and protect critical regions, I used the *synchronized* keyword and *ConcurrentHashMap*. This ensures that multiple threads do not concurrently modify shared resources and lead to inconsistent or erroneous states. By using *synchronized*, only one thread can edit the dictionary, preventing race conditions and ensuring the integrity of the operations. With *ConcurrentHashMap,* it uses “locking” to control concurrent access to ensure thread-safe operations, making sure the dictionary system can deal with users’ commands correctly.

**Summary**

In this project, I have developed a multi-threaded dictionary system that allows users to search, add, remove and modify words with a client-server architecture. The system uses a worker pool to handle concurrent client requests for better performance and resource management. TCP is used to ensure reliable communication and accurate data transmission between the client and the server. The GUI is designed to be intuitive and user-friendly, with clear layout and instructions to enhance user experience. While the system currently does not currently support data persistence because of security concerns, this could be implemented in the future with safeguards.