**README for s12556828final**

**Author:** Xu Xiaochi

**Student Number:** 1255682

**File Description**

**Project\_server.py:**

This script is the core of the project, implementing a Flask web server with two main services: a Pi calculation service and a quote retrieval service. It handles user authentication, validates input data for simulations and concurrency levels, performs Monte Carlo simulations for Pi estimation, and fetches quotes using TCP/UDP protocols. Additionally, it manages a statistics file for tracking user requests, ensuring thread-safe operations.

**Quote\_server.py:**

This server application provides a service for delivering random quotes over TCP and UDP protocols. It initializes a pool of quotes from a compressed and encoded text block, then listens on specified ports for incoming requests. Upon connection, it simulates a delay to mimic real-world response times and then sends back a random quote prefixed with either "T" or "U" to indicate the protocol used. The server supports concurrent handling of clients through threading and can be started with custom host and port arguments via command line.

**test.py:**

The provided test.py script is a collection of unit tests for a project server. These tests check various functionalities of the server, including user authentication, simulation parameters for a Pi calculation service, concurrency handling, and a quote service using TCP/UDP protocols.

**Setup and Execution Instructions**

**Project Server Setup:**

**1. Required Libraries:**

List the third-party libraries and their versions required for this project. Example:

Flask

Requests

2. **Installation:**

Install the required libraries using pip:

pip install Flask

pip install requests

**3. Data Files Initialization:**

Our web service uses a simple text file named request\_stats.txt to track and persist the statistics of web service requests. This file records the usernames of clients who have made requests and the corresponding count of requests made by each user. It is important to note that only users who have made at least one request are included in this file.

**To initialize the request\_stats.txt file:**

1. **First Run:** Upon the first run of the Flask application, the init\_stats\_file() function checks for the existence of request\_stats.txt. If the file does not exist, the function creates it and initializes it with an empty JSON object {}. This is done to prepare the file for subsequent write operations.
2. **Updating Statistics:** Every time a user makes a request to either the /pi or /quote endpoint, the update\_statistics(username) function is invoked. This function reads the current contents of request\_stats.txt, updates the request count for the user, and then writes the updated statistics back to the file. This ensures that the request counts are always current.
3. **Concurrency Handling:** Access to request\_stats.txt is managed by a lock (lock = threading.Lock()) to prevent race conditions when multiple threads attempt to read or write to the file simultaneously. This guarantees that updates to the file are thread-safe and that the data integrity is maintained even under high concurrency scenarios.
4. **File Location:** The request\_stats.txt file is located in the same directory as the Flask application for ease of access. However, the location can be customized if needed by changing the STATS\_FILE variable in the Flask application's source code.
5. **Backup and Maintenance:** It is recommended to regularly back up request\_stats.txt to prevent data loss. In addition, monitoring the file size and archiving its contents may be necessary as part of regular maintenance, especially for applications with high traffic.

By following these steps, the request\_stats.txt file will be properly initialized and ready to record the web service request statistics accurately.

**4. Execution:**

To start the project server, run the following command:

>>python Project\_server.py

**Test Setup:**

Running Tests:

* Ensure the project server and quote server is running.
* Execute the test script using:

>>python test.py

**JSON Request and Response Formats**

**Pi Calculation Service**

Request Format:

| **Field** | **Type** | **Description** | **Example** |
| --- | --- | --- | --- |
| username | String | 4-digit user identifier | "0123" |
| password | String | Password associated with the username | "0123-pw" |
| simulations | Integer | Number of Monte Carlo simulations to run for calculating Pi | 1000 |
| concurrency | Integer | Level of concurrency for the request processing | 4 |

*Note: The ‘*simulations’ *field specifies the number of random points to generate for the Monte Carlo Pi estimation. The ‘*concurrency’ *field determines the number of parallel processes or threads to use for the calculation.*

Response Format:

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| calculated\_value\_of\_pi | Float | The calculated approximate value of Pi |
| number\_of\_simulations | Integer | The number of simulations used for calculation |
| level\_of\_concurrency | Integer | The concurrency level used for processing |
| measured\_time\_for\_processing\_request | Float | The time taken to process the request in seconds |

*Note: The response includes the result of the Pi calculation as well as metadata about the request processing.*

**Quote Service**

Request Format:

| **Field** | **Type** | **Description** | **Example** |
| --- | --- | --- | --- |
| username | String | 4-digit user identifier | "0123" |
| password | String | Password associated with the username | "0123-pw" |
| protocol | String | The protocol to be used: "tcp" or "udp" | "tcp" |
| concurrency | Integer | Level of concurrency for the request processing | 2 |

*Note: The protocol field specifies the network protocol to use when fetching quotes. The concurrency field determines the number of parallel requests to make to the quote server.*

Request Format:

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| quotes | Array | An array of quotes received from the server |
| protocol | String | The protocol used for the request |
| concurrency | Integer | The concurrency level used for processing |
| processing\_time | Float | The time taken to process the request in seconds |

*Note: The response includes an array of requested quotes along with request processing metadata.*

**Advanced Technologies Discussion**

**//To be Finished**