

**Multithreaded Flight Arrival Client/Server Information System**



**Instructor**

**PRO. Mohammed Almeer**



**Course :** ITNE352: Network Programming **Semester :** S2 2024-2025

**Project Description:**

This project involves developing a client-server system using Python to exchange flight information for a specified airport. The system emphasizes client-server architecture, network communication, multithreading, API usage, and good coding practices. The server retrieves flight data from aviationstack.com using their API, manages multiple client connections, and responds to client requests. The client application allows users to query flight information, such as arrived flights, delayed flights, or details of a specific flight.

**Key Features and Functionalities:**

**Server Script:**

* Retrieves flight data from aviationstack.com based on the airport's ICAO code.
* Stores the retrieved data in a JSON file.
* Handles multiple client connections simultaneously (at least three).
* Extracts and sends relevant flight information to clients based on their requests.
* Displays connection status, client requests, and disconnections on the server screen.

**Client Script:**

* Connects to the server and sends the client's username.
* Sends requests to the server for flight information:
  + Arrived flights: flight code, departure airport, arrival time, terminal, and gate.
  + Delayed flights: flight code, departure airport, departure time, estimated arrival time, delay, terminal, and gate.
  + Details of a specific flight: flight code, departure and arrival details (airport, gate, terminal), status, scheduled departure and arrival times.
* Displays flight information neatly.
* Closes the connection when the user chooses to quit.

**Learning Objectives:**

The project aims to help students:

* Understand client-server architecture.
* Implement network communication using Python.
* Apply multithreading techniques.
* Utilize APIs to retrieve data.
* Practice good coding practices.

**Design Guidelines:**

* Implement a basic Python server to handle TCP connections.
* Use the aviationstack.com API to retrieve flight data.
* Implement multithreading to handle concurrent client connections.

**Evaluation Criteria:**

* Correctness and completeness of the implementation (30%).
* Design elegance, clarity, and robustness (20%).
* Code quality and style (10%).
* Report quality (10%).
* Demonstration (20%).
* Additional concepts (10%).
* Penalties for poor engagement, code style issues, redundancy, unclear output, and late submissions.

**Additional Concepts:**

Students are encouraged to explore and implement one of the following additional concepts:

* TSL/SSL for secure communication.
* Object-Oriented Programming (OOP) in Python.
* Graphical User Interface (GUI) for the client.

**GitHub Usage:**

* Groups are required to use GitHub for collaborative development.
* The repository should be named "ITNE352 Project - Group X".
* Both group members and the instructor should have access to the repository.
* All code and project-related activities should be managed through GitHub.

**Deliverables:**

* README.md (and PDF version).
* Client and server scripts, and the JSON file.
* Demonstration Video.

**Resources:**

The document provides a list of resources, including IEEE Explore, ACM Digital Library, ScienceDirect, and Google Scholar, for finding relevant research papers. It also lays out the proper format for references.

**Plagiarism Policy:**

The document emphasizes the importance of original work and outlines a strict plagiarism policy. All submissions will be checked for plagiarism, and any instances of copying or inadequate paraphrasing will be penalized.

Here are the steps to design the application, based on the "Recommended Design Guidelines" section in the Canvas document:

1. **Networking**
   * Implement a basic Python server that can accept incoming TCP connections and echo data sent on the connections back to the sender.
   * Test the server using the "telnet" program.
   * Use Wireshark to follow the exchange packets between the client and the server.
   * Implement the client and test the connection with the server.
2. **Retrieving the data from the online source**
   * Implement the API to retrieve the data from the online source.
   * Implement the extraction of wanted information.
3. **Threads**
   * Extend the networking code from Step 1 to accept multiple connections by creating a thread to deal with each incoming connection.
   * Ensure that functions are placed in their appropriate locations (main thread, loop, thread body).

**Technologies Used:**

Python 3

Socket Programming

Threading

aviationstack.com API

JSON

Terminal UI (Command Line Interface)

**Requirements Recap:**

Multi-threaded TCP server

JSON data fetch from aviation stack API

Serve multiple client queries

Clean user interaction and flight data display

**Optional Features**

Save session logs per client.

Add input validation.

Extend to GUI using T Kinter or Flask (for bonus).

**Tips for Professional Development**

Replace 'YOUR\_API\_KEY' with a valid API key from aviation stack.

Ensure JSON file is structured for easy parsing.

Use try/except blocks for robust error handling.

Use comments and logging for maintainability.

* + Attached with the cement are the codes required to create the application.

THANK YOU