ITNE 352 Project Solution - Multithreaded Flight Information System

Introduction

This project involves developing a multithreaded client/server system using Python to manage and display flight information using the aviationstack API, The system is designed to handle multiple clients concurrently and process three types of requests regarding flight data, Object-Oriented Programming (OOP) was used in the implementation.

Server Code Description

• The server starts by requesting the airport code (ICAO) from the user.
• It uses the aviationstack API to retrieve 100 flight records.
• Stores the retrieved data in a JSON file named group_ID json.
• Uses multithreading to handle multiple client connections.
• Processes requests based on type (arrived, delayed, specific flight details).

Client Code Description

• Connects to the server and sends the username.
• Displays a menu of options for the user:
  1. Show all arrived flights.
  2. Show all delayed flights.
  3. Get details of a specific flight.

4. Quit.

• Sends the selected request to the server and displays the response.

• Stays connected until the user selects the quit option.

Used API

The aviationstack API is used to retrieve real-time flight data. A free API key can be obtained by registering at aviationstack.com.

Object-Oriented Programming (OOP)

OOP was implemented in the code using several classes to organize logic:

• FlightData: To parse and hold flight data.

• ClientHandler: To handle each client connection in a separate thread.

• Server: To run the server and manage connections.

How to Run the System

1. Install required libraries using:
   pip install requests

2. Run the server script:
   python server.py

3. Run the client script:
   python client.py

4. Test the system by sending different requests from the client to the server.

GitHub and Report

• A GitHub repository was created to manage the development process.
• The README.md file includes project description, requirements, setup guide, usage, and code examples.
• This document can be included as part of the final report submission.

Important Notes and Penalties

• Ensure clear display of outputs in both client and server terminals.
• Avoid redundancy and maintain clean, readable code.
• Adhere to deadlines to avoid penalty points.
• Avoid plagiarism – all work must be original or properly cited.