UEEN3123/UEEN3433 TCP/IP Network Application Development

ASSIGNMENT

June 2024 Trimester

Course Outcomes

In this assessment, you are assessed based on the following **course outcomes**:

- **CO2**: Build client-server network applications using TCP and UDP transport-layer protocols.
- **CO3**: Develop multicast and broadcast network applications.
- **CO4**: Utilize a scripting language to automate networking tasks.

Assessment Contribution

This assignment contributes **15%** to the overall assessment for this course with equal contribution for each CO.

Membership

This assignment must be attempted in an **individually**.

Deadline

The deadline for this assignment is **Friday, September 06, 2024**.

Part A: Weather Monitoring System - Web API Development

GreenEarth Inc., an environmental monitoring organization, is in the process of developing a cloud-based application that will serve as a central control hub for collecting and managing real-time weather data from various weather stations across the country. The data collected includes temperature, humidity, wind speed, and atmospheric pressure, which are crucial for environmental analysis and forecasting.

Your task is to create a Web API using Flask that allows weather station devices to submit their data and allows authorized personnel to retrieve and manage this data. The data will be stored in an SQLite database, and the Web API will provide the necessary endpoints to perform database operations.

The Web API should support the following operations:

1. **Retrieve all weather data entries:**
   Endpoint: `GET /api/weather`

2. **Retrieve specific weather data by station ID:**
   Endpoint: `GET /api/weather/<int:station_id>`

3. **Add a new weather data entry:**
   Endpoint: `POST /api/weather`

4. **Update existing weather data by station ID:**
   Endpoint: `PUT /api/weather/<int:station_id>`

5. **Delete a weather data entry by station ID:**
   Endpoint: `DELETE /api/weather/<int:station_id>`

Database Structure:

The SQLite database will have a table named ***weather_data*** with the following structure:

id	station_id	temperature	humidity	wind_speed	pressure	timestamp
1	ST1001	27.5°C	60%	12 km/h	1013 hPa	2024-08-14 10:00:00
2	ST1002	29.0°C	65%	15 km/h	1010 hPa	2024-08-14 10:05:00
3	ST1003	26.0°C	55%	8 km/h	1012 hPa	2024-08-14 10:10:00
4	ST1001	28.0°C	62%	10 km/h	1011 hPa	2024-08-14 10:15:00
5	ST1002	30.0°C	568%	18 km/h	1008 hPa	2024-08-14 10:20:00

Table 1.1: Sample partial data of *weather* table

Submission Requirements:

- Source code for the Flask application.
- SQLite database file with some sample data.
- Documentation describing how to set up and run the application.

Part B: Python Scripts for Weather Monitoring System

Develop the following Python scripts that connect to the Web API created in Part A to perform the following operations:

1. **Upload Weather Data from CSV to Server via Web API**:
   - Write a Python script that reads weather data from a CSV file and uploads the data to the server using the Web API.
   - The CSV file should contain columns corresponding to the `station_id`, `temperature`, `humidity`, `wind_speed`, and `pressure` fields.
   - Each row in the CSV file represents a new weather data entry that needs to be uploaded to the server.
2. **Download All Weather Data from Server to CSV**:
   - Write a Python script that connects to the Web API, retrieves all weather data from the server, and saves it into a CSV file.
   - The CSV file should include the same columns as the database: `station_id`, `temperature`, `humidity`, `wind_speed`, `pressure`, and `timestamp`.
3. **Update Weather Data Based on Naming Convention Changes**:
   - A script is required to update specific weather station IDs due to a rebranding initiative. For example, station ID `ST1001` may be renamed to `ST2001`.
   - The script should read these updates from a CSV file where each row contains the old `station_id` and the new `station_id`.
   - The script should then update the corresponding records on the server via the Web API, changing the `station_id` in the database.

Assignment Details:

Script 1: Upload Weather Data from CSV

- The script should parse a `CSV` file with the following structure:

```
station_id,temperature,humidity,wind_speed,pressure
ST1001,27.5,60,12,1013
ST1002,29.0,65,15,1010
```

- For each row, the script should send a `POST` request to the `/api/weather` endpoint to add the data to the server.

Script 2: Download All Weather Data to CSV

- The script should send a `GET` request to the `/api/weather` endpoint to retrieve all weather data.
- It should then save the data into a `CSV` file with the following structure:

```
station_id,temperature,humidity,wind_speed,pressure,timestamp
ST1001,27.5,60,12,1013,2024-08-14 10:00:00
ST1002,29.0,65,15,1010,2024-08-14 10:05:00
```

Script 3: Update Weather Station IDs

- The script should parse a `CSV` file with the following structure:

```
old_station_id,new_station_id
ST1001,ST2001
ST1002,ST2002
```

- For each row, the script should send a `PUT` request to the `/api/weather/<old_station_id>` endpoint to update the station_id to the new value.

Submission Requirements:

- Three Python scripts corresponding to each of the tasks described above.
- Sample CSV files for testing each script.
- A brief README file explaining how to run each script and any dependencies required.

Part C: Weather Alert Broadcast System

As part of the Weather Monitoring System, GreenEarth Inc. wants to implement a real-time broadcast application that can disseminate urgent weather alerts (such as extreme temperature or high wind speed warnings) to all connected clients on the same subnet. The goal is to ensure that these alerts are promptly delivered to all monitoring stations or devices within the network.

You are required to develop a broadcast application that includes a Broadcast Sender and a Broadcast Receiver to handle these alerts.

1. **Broadcast Sender**:
   - The sender application should prompt the user to enter the following information:
     - **Date & Time of Occurrence**: The specific time when the event was detected.
     - **Type of Alert**: The type of warning (e.g., "High Temperature Warning", "High Wind Speed Warning").
     - **Location**: The location of the weather station that detected the event.

    - **Description**: A brief description of the event (e.g., "Temperature exceeded 40°C at Station ST1003").
  - The application should automatically add the server's current date and time to the data.
  - The combined data should be formatted in **JSON** and broadcasted to all receivers on the subnet.
2. **Broadcast Receiver**:
  - The receiver application should continuously listen for broadcast messages on the subnet.
  - Upon receiving a message, the application should parse the JSON data and display the alert information on the screen in a user-friendly format.

Deliverables

You should submit the following items in your report:

1. Softcopy of all Python scripts, arranged accordingly by parts.
2. Softcopy of all Python scripts, each part should be grouped within the same project.
3. Softcopy of all SQLite and CSV files used, appropriately placed within the same project for which they belong to.

Submission & Originality Policies

1. This assignment must be submitted completely by the abovementioned deadline.
2. All work must be original and if, taken from any works other than yours must be properly referenced.
3. Any plagiarized content will be penalized, including the award of zero marks.

Marks Allocation

Part / No.	Course Outcome	Assessment Criteria	Marks
A / 1	CO2	Data Access & Storage	5
A / 2	CO2	Functionalities	20
A / 3	CO2	Data Exchange	8
B / 1	CO3	Use of HTTP Client module	10
B / 2	CO3	Script Logic	15
B / 3	CO3	Data Exchange	8

C / 1	CO4	Sockets Usage	10
C / 2	CO4	Program Logic	15
C / 2	CO4	Data Exchange	8
		TOTAL	99