ITNE352 Project – Group GC5

News Service System (Client/Server) using Python, Sockets, JSON, APIs, File I/O

Group Information

- **Course Code:** ITNE352 (Network Programming)
- **Section:** 03
- **Group Name:** GC5

Student ID	Student Name
202204625	Abdulrahman Abdo Ahmed Shomaila
202208352	Kadhim Abdulla Kadhim Alhaddar

Semester

First Semester of 2025–2026

Project Title

News Service System – Client/Server Project

Project Description

This project is a **client–server news service system** where:

- The **server** accepts TCP connections (multi-threaded), receives client requests, calls **NewsAPI.org** endpoints, returns results in **JSON**, and saves the full API response into JSON files for evaluation.
- The **client** connects to the server, shows menus (Headlines / Sources), displays a list of results, then requests **full details** for a selected item.

✅ Output is limited to **15 results max** (matches `MAX_RESULTS = 15`).

Table of Contents

🗁 File Structure

File / Folder	Description
`server.py`	Multi-threaded TCP server + NewsAPI requests + validation + JSON saving
`client.py`	Client UI (menus) + sends requests + shows list + requests details
`requirements/`	Dependency files (example: `requirements.txt`)
`*.json`	Auto-generated result files: `<client>_<option>_<group_ID>.json`
`.env`	Contains `API_KEY=...` (not committed)
`.gitignore`	Prevents committing `.env` and generated files

Requirements (Setup)

1) Install Python

- Python **3.9+** recommended.

2) Install dependencies

Your server uses:

- `requests`
- `python-dotenv`

If you have `requirements/requirements.txt`, install like this:

```
pip install -r requirements/requirements.txt
```

Or install manually:

```
pip install requests python-dotenv
```

3) Add your NewsAPI key

Create a file named `.env` in the project root:

```
API_KEY=your_newsapi_key_here
```

▶ How to Run

Step 1: Start the server

```
python server.py
```

Expected:

```
[YYYY-MM-DD HH:MM:SS] Server started on 127.0.0.1:12345
[YYYY-MM-DD HH:MM:SS] Waiting for client connections...
```

Step 2: Start the client

```
python client.py
```

The client stays connected until you select **Quit**.

How to Use the Client Menus

According to the project specification, the client has **3 main menus** (Main, Headlines, Sources).

Main Menu

- **Search headlines** → opens Headlines menu
- **List of Sources** → opens Sources menu
- **Quit** → closes connection

Headlines Menu (examples)

- Search for keywords
- Search by category
- Search by country
- List all new headlines
- Back to main menu

Sources Menu (examples)

- Search by category
- Search by country

- Search by language
- List all
- Back to main menu

✅ Allowed parameters:

- Countries: `au, ca, jp, ae, sa, kr, us, ma`
- Languages: `ar, en`
- Categories: `business, general, health, science, sports, technology`

Protocol (Client ↔ Server Messages)

The server receives **JSON messages** ended by a newline `\n` (it reads until `\n` then parses JSON).

Message format

```
{"action": "headlines_keyword", "params": {"q": "bitcoin"}}
```

Examples (based on your server handlers)

- Headlines list actions start with `headlines_...`
- Sources list actions start with `sources_...`
- Detail requests:
  - `headlines_detail` with `{"index": 1}`
  - `sources_detail` with `{"index": 1}`
- Quit:

```
{"action":"quit","params":{}}
```

Scripts Description

server.py

Purpose: Multi-threaded news server that handles clients, validates input, fetches NewsAPI data, sends lists/details, and saves JSON files.

Used packages

- `socket`, `threading`, `json`, `datetime`, `os`
- `requests`
- `dotenv (python-dotenv)`

Main classes (from your code)

- `NewsDataFetcher` Calls NewsAPI endpoints:
  - `top-headlines`
  - `sources`

  Snippet:

  ```
  api_response = requests.get(endpoint, params=request_params, timeout=10)
  api_response.raise_for_status()
  return api_response.json()
  ```

- `ParameterValidator` Validates request params (country/language/category/keyword). Uses allowed sets.
- `ClientConnectionHandler (thread)` Handles each client on a separate thread, logs requests, caches results, supports detail view by index, and saves responses as:

  ```
  <client_name>_<option>_<group_ID>.json
  ```

  (Your code matches this format.)

  Snippet:

  ```
  filename = f"{self.username}_{sanitized_action}_{GROUP_IDENTIFIER}.json"
  json.dump(data, file, indent=4)
  ```

- `NewsServerApplication` Binds to `127.0.0.1:12345`, listens, and accepts clients (at least 3 supported by design).

---

client.py

Purpose:
Implements the required menus, sends JSON requests to the server, displays list results, and fetches details by index.

Used packages

- `socket`
- `json`

Main functionality

- Establishes a TCP connection with the server

- Sends the client username to the server on connection
- Displays three interactive menus:
  - Main Menu
  - Headlines Menu
  - Sources Menu
- Sends user requests to the server using JSON messages
- Receives and parses JSON responses
- Displays lists of headlines or sources
- Requests and displays detailed information by index

Additional Concepts Used

1. **Multithreading**: each connected client is handled in a separate thread (`ClientConnectionHandler`).
2. **Object-Oriented Programming (OOP)**: the server design uses multiple classes (`NewsDataFetcher`, `ParameterValidator`, etc.).
3. **Environment variables**: API key is stored in `.env` and loaded using `python-dotenv` (safer than hardcoding).

Acknowledgments

NewsAPI key used to fetch data for free : https://newsapi.org/

Conclusion

This project implements a complete client/server architecture for a news system using Python sockets, JSON messaging, API integration, multi-threading, and file saving for evaluation, matching the required menus, responses, and output limits.

Resources

- Python OOP Concepts: https://www.geeksforgeeks.org/python/python-oops-concepts/
- Python and REST APIs: https://realpython.com/api-integration-in-python/
- NewsAPI: https://newsapi.org/