Web Scraper Communication Contract

This README will walk the user through setting up this microservice.

Fundamentally, the program waits for a successful detection of a CSV file that contains the game library.

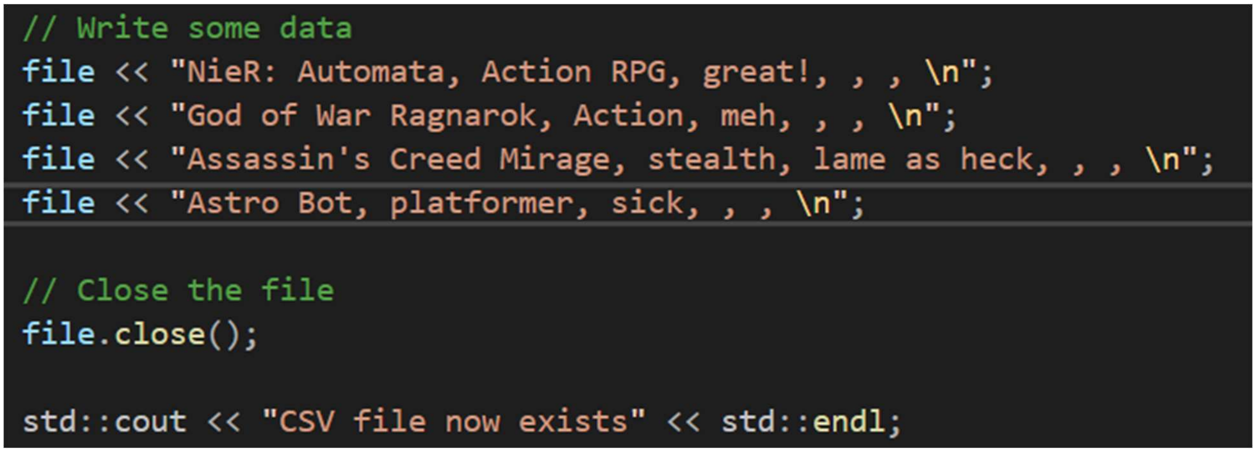
Once this CSV is detected, the program will process it, populating the needed 3 categories before either returning a new file or overwriting the existing CSV file.

It is critical that the user overwrites the desired file paths into the function calls at the bottom of the python file to their desired directory.

- Using the same file path for both the read and write will result in an overwritten file
- Using different file paths for the read and write will result in a new file

1.) How to request data

You will leave the microservice on as a consistent process which will wait for detections of a CSV file. In C++, writing and creating a file will trigger the process as detailed below.



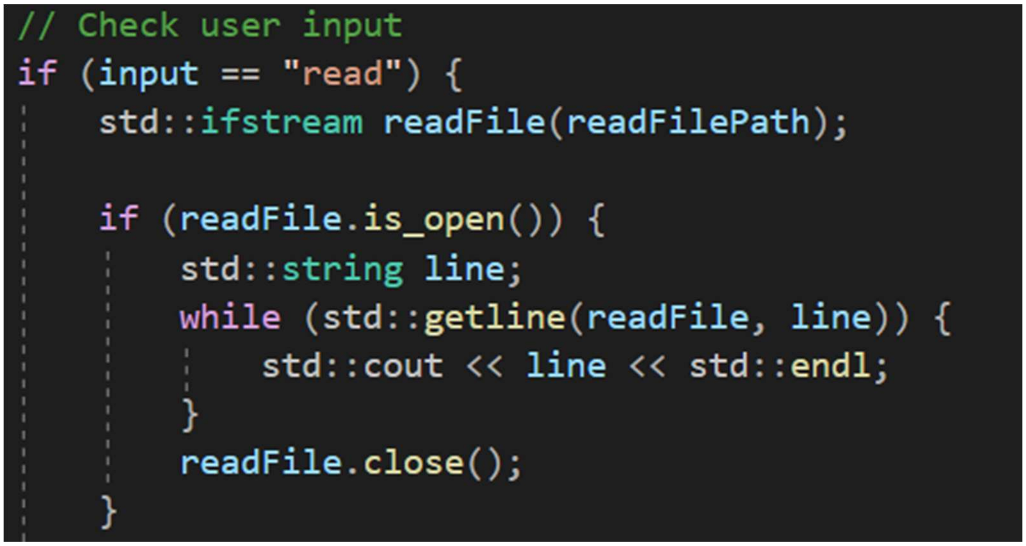
```
// Write some data
file << "NieR: Automata, Action RPG, great!, , , \n";
file << "God of War Ragnarok, Action, meh, , , \n";
file << "Assassin's Creed Mirage, stealth, lame as heck, , , \n";
file << "Astro Bot, platformer, sick, , , \n";

// Close the file
file.close();

std::cout << "CSV file now exists" << std::endl;
```

2.) How to receive data

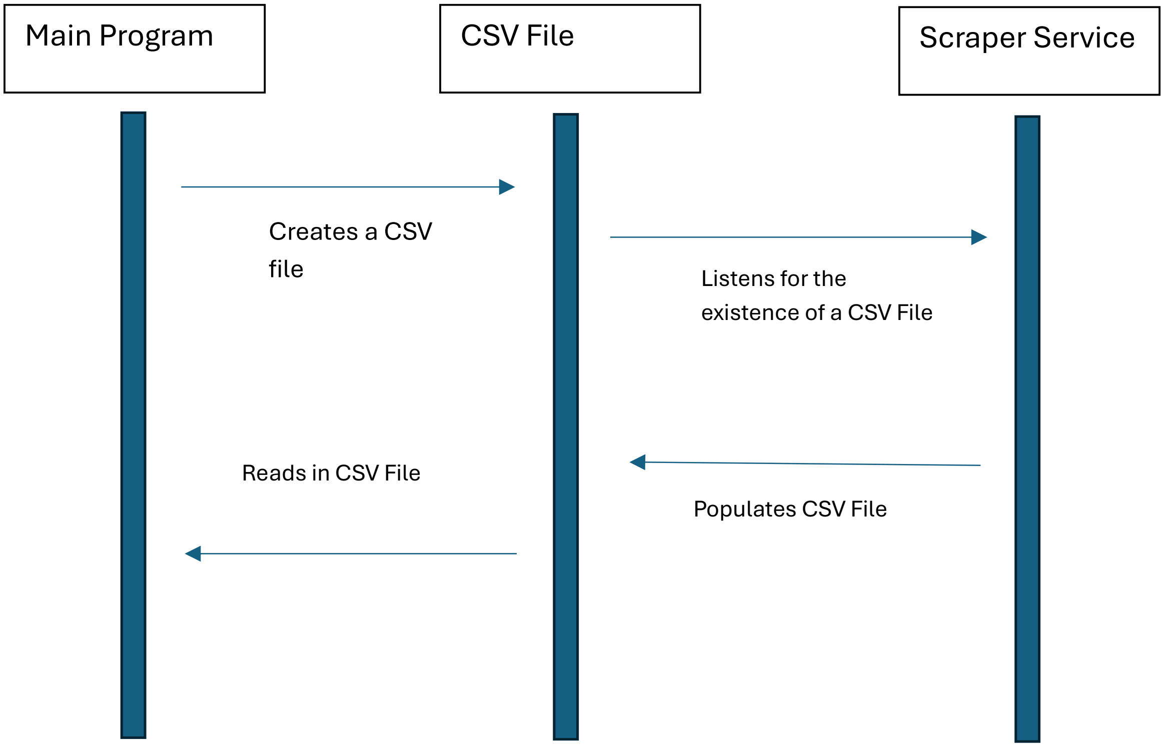
A listener will wait for the existence of the desired response file and process as needed.



```cpp
// Check user input
if (input == "read") {
    std::ifstream readFile(readFilePath);

    if (readFile.is_open()) {
        std::string line;
        while (std::getline(readFile, line)) {
            std::cout << line << std::endl;
        }
        readFile.close();
    }
```

3.) UML Sequence Diagram


Main Program
CSV File
Scraper Service
Creates a CSV file
Listens for the existence of a CSV File
Reads in CSV File
Populates CSV File