NexusFetcher Web Scraping Bot

NexusFetcher is a professional, high-performance desktop application designed for automated web scraping. It features a modern graphical user interface (GUI), robust error handling, and powerful data management capabilities, including real-time Google Sheets integration and CSV export. The application is built to handle large-scale data extraction tasks efficiently and reliably.

(Note:)

Key Features

- **Modern & Intuitive GUI:** A clean, card-based user interface built with ttkbootstrap for a professional look and feel.
- **Automated Multi-Step Scraping:** Navigates complex website structures with a predefined sequence of clicks and data entry.
- **High-Performance Headless Mode:** Option to run the browser invisibly in the background for significantly faster scraping performance.
- **Robust Session Management:**
  - **Stop & Resume:** Gracefully stop the scraping process at any time and resume from the exact point you left off.
  - **File Updating:** Optionally remove processed entries from the source .txt file upon completion, perfect for batch processing.
- **Real-Time Progress Tracking:**
  - Live counters for Processed, Scraped, and Skipped records.
  - A dynamic progress bar showing overall completion.
  - An estimated time remaining (ETA) calculator.
  - A live status bar for immediate feedback on the bot's current action.
- **Flexible Data Extraction:**
  - Scrape mandatory data fields (Name, Email, Phone).
  - Optionally scrape additional data fields (Physical Address, Mailing Address) via UI checkboxes.
- **Powerful Data Output:**
  - **Real-time Google Sheets Integration:** Automatically appends each successfully scraped record to a specified Google Sheet in real-time.
  - **CSV Export:** Export all data collected in the GUI's table to a local CSV file with a single click.
- **Live** Resource **Monitoring:** An integrated "App Resources" panel displays the application's real-time CPU and RAM usage, as well as system-wide network speed.

Tech Stack

- **Language:** Python 3

- **GUI:** Tkinter with the ttkbootstrap library for modern styling.
- **Web Automation:** Selenium with ChromeDriver.
- **System Monitoring:** psutil
- **Google Sheets API:** gspread and google-auth-oauthlib
- **Image Handling (for UI):** Pillow (PIL)

Setup and Installation

Follow these steps to set up the development environment and run the application.

1. Prerequisites

- **Python 3.8+:** Make sure you have a recent version of Python installed.
- **Google Chrome:** The application is configured to use the Chrome browser.

2. Clone the Repository

https://github.com/muneebrehman-17/Portfolio/blob/1ff6aea5d72e39d967a359c37224f8a5fdfe9588/Nexus%20Fetcher%20Full

3. Install Dependencies

It's highly recommended to use a virtual environment.

```
# Create and activate a virtual environment (optional but recommended)
python -m venv venv
source venv/bin/activate  # On Windows, use `venv\Scripts\activate`

# Install the required packages
pip install -r requirements.txt
```

requirements.txt file:

```
selenium
ttkbootstrap
gspread
google-auth-oauthlib
psutil
Pillow
```

4. Google Sheets API Setup (for GSheets Integration)

To use the Google Sheets feature, you need to configure Google Cloud credentials.

1. **Google Cloud Project:** Create a new project in the Google Cloud Console.
2. **Enable APIs:** In your project, enable the **"Google Drive API"** and the **"Google Sheets API"**.
3. **Create Service Account:** Create a service account. When creating credentials for it, select the "Service Account" type.
4. **Get JSON Key:** Download the private key for the service account in JSON format.
5. **Embed Credentials:** Open the downloaded .json file and copy its entire content. In main_gui.py, paste this content into the SERVICE_ACCOUNT_INFO dictionary, replacing the placeholder data.
6. **Share Your Sheet:** Open your target Google Sheet. Click **"Share"** and share it with the client_email address found inside your .json file, giving it **Editor** permissions.

How to Run

Once all dependencies are installed, you can run the application from the project's root directory:

python main_gui.py

Code Structure

The application is contained within a single file (main_gui.py) for simplicity. The core components are:

- **Application Class:** This is the main class that builds and manages the entire Tkinter GUI. It handles user interactions, state management (like the resume index), and threading.
- **run_scraping_process() Method:** This is the "engine" of the bot. It runs in a separate thread and contains all the Selenium and Google Sheets logic. It communicates with the GUI via callback functions (update_status, add_data_to_table, etc.) to keep the UI responsive.
- **Helper Methods:** Various methods within the Application class handle specific tasks like updating stats, exporting to CSV, and managing the UI state.

Key Configuration Points for Developers

- **CSS Selectors:** All CSS selectors for the target website are hardcoded within the run_scraping_process method. To adapt the bot for a different website, these selectors are the primary values that need to be changed.
- **Fallback Logic:** The find_and_click_with_fallbacks method is designed to handle inconsistent UI elements. You can add more selectors to its call lists to make the bot even more robust.
- **SERVICE_ACCOUNT_INFO:** The Google Sheets credentials are embedded directly in this dictionary. For production, consider loading these from a secure file or environment variable.

Contributing

Contributions are welcome! Please feel free to fork the repository, make your changes, and submit a pull request.

License