Selenium Prisoner Scraper – Documentation

# 📌 Overview

This project is a Python-based web scraper that extracts inmate details from a religious pen-pal listing website. It uses Selenium to:

- Automate login  
- Navigate paginated listings  
- Bypass reCAPTCHA audio challenges  
  
Extracted data is stored in Excel and appended to a historical master record.

# 🎯 Purpose

- Automate the collection of prisoner contact info from a public listing

- Maintain a historical record with detection of new entries

- Provide daily updated reports in Excel for analysis

# ✨ Key Features

✅ Automatic login with reCAPTCHA audio solving

✅ Handles dynamic elements and pagination

✅ Detects and saves only new entries

✅ Exports to both daily and master Excel files

✅ Built-in logging with timestamps for monitoring/debugging

# 🛠️ Technology Stack

Language:  
- Python 3

Libraries & Tools:  
- selenium – Browser automation  
- webdriver-manager – ChromeDriver management  
- pandas – Data processing  
- soundfile & speech\_recognition – reCAPTCHA audio  
- python-dotenv – Environment variable handling

# ⚙️ Environment Setup

1. Create a Virtual Environment (recommended):

python -m venv venv  
source venv/bin/activate # On Windows: venv\Scripts\activate

2. Install Dependencies:

pip install -r requirements.txt

# 🚀 How to Run

python your\_script\_name.py

Outputs are saved to prisoner\_data/:

- prisoner\_data\_YYYY-MM-DD.xlsx – Daily snapshot

- prisoner\_data\_all.xlsx – Historical record

# ⏰ Automate with Windows Task Scheduler

1. Open Task Scheduler → Create Basic Task

2. Set a name (e.g., Daily Prisoner Scraper)

3. Choose Daily, set a time

4. Select Start a program

5. For Program/script, input the Python path

C:\Users\YourName\AppData\Local\Programs\Python\Python39\python.exe

6. For Add arguments, input your script path:

"C:\path\to\your\_script\_name.py"

7. Enable Run with highest privileges (optional)  
  
Ensure .env and dependencies exist in the running environment.

# ❓ Why Selenium

- The site uses dynamic JavaScript-rendered content

- reCAPTCHA solving requires browser interactivity

- Selenium supports real-time interaction with web elements

# 🧠 Unique Skills Demonstrated

- Bypassing reCAPTCHA audio with speech recognition

- Handling dynamic wait conditions (WebDriverWait, expected\_conditions)

- Extracting nested HTML with CSS selectors

- Scalable data handling with Pandas

- Reliable logging & error management

# 📁 Project Folder Structure

project\_root/  
├── prisoner\_data/  
│ ├── prisoner\_data\_YYYY-MM-DD.xlsx  
│ └── prisoner\_data\_all.xlsx  
├── .env  
├── app.log  
├── your\_script\_name.py  
├── requirements.txt