Project Documentation

Overview

This project automates various tasks using Appium, Celery, SQLite, and FastAPI. The tasks range from mobile automation and data scraping from APIs to APK analysis and creating a FastAPI server that provides real-time communication through WebSockets. The skills demonstrated include:

Mobile Automation using Appium and UIAutomator2.

Task Queueing and scheduling with Celery.

Data Management with SQLite.

WebSocket Communication using FastAPI for real-time updates.

Proxy Handling to rotate requests for scraping data from APIs.

Scripts Overview and Purpose

1. start_worker.py

Purpose: To launch both the FastAPI server and the Celery worker for asynchronous task processing.

Details: It starts the FastAPI server that serves a web dashboard for displaying device info, cryptocurrency data, and APK metadata. It starts a Celery worker, which runs background tasks such as fetching data from APIs or mobile automation tasks.

2. main.py

Purpose: To act as the main application, setting up the FastAPI routes, Celery tasks, and database interaction.

Details: Displays real-time data like device information, APK metadata, and cryptocurrency data using FastAPI's web interface. It processes background tasks using Celery for efficient data processing.

3. uiautomator_deviceinfo.py

Purpose: To automate device information extraction using UIAutomator2.

Details: It automates interactions with an Android device by scrolling through the settings menu and extracting detailed device information.

4. chrome_analysis.py

Purpose: To analyze an APK file and extract metadata such as permissions, activities, and package information.

Details: It uses the androguard library to analyze the APK's internal structure and extracts key details.

5. zeb.py

Purpose: To scrape cryptocurrency data from the ZebPay API using rotating proxies.

Details: Since the ZebPay API is restricted to Indian IP addresses, this script rotates through a list of Indian proxies to make requests.

6. mobile_automation.py

Purpose: To automate user interactions on a mobile app (YouTube) using Appium.

Details: It simulates user interactions like logging in, searching for a video, and submitting a comment.

Project Structure

```
Project Root
|
|- main.py                : Main FastAPI and Celery integration script
|- start_worker.py        : Script to start the FastAPI server and Celery worker
|- mobile_automation.py   : Appium script for automating mobile interactions
|- uiautomator_deviceinfo.py: Script to extract device info using UIAutomator2
|- chrome_analysis.py     : APK analysis using androguard
|- zeb.py                 : Cryptocurrency scraping with proxy rotation
|- requirements.txt       : List of dependencies
|- device_data.db          : SQLite database for device information
|- apk_metadata.db         : SQLite database for APK analysis
|- zebpay_data.db          : SQLite database for crypto data
|- device_info.json        : JSON file storing the extracted device information
```

Execution Workflow

1. Start the Project: Use start_worker.py to initiate the FastAPI server and Celery worker.

2. Device Information Extraction: Run uiautomator_deviceinfo.py to collect and save Android device details.

3. APK Analysis: Execute chrome_analysis.py to extract and store metadata from an APK file.

4. Cryptocurrency Data: Run zeb.py to scrape cryptocurrency data from the ZebPay API.

5. Mobile Automation: Use mobile_automation.py to simulate interactions in the YouTube mobile app using Appium.

6. Access the Dashboard: Navigate to http://127.0.0.1:8000 to view device info, APK analysis, and cryptocurrency data in real-time.

Database Setup

This project uses SQLite databases for persistent data storage:

device_data.db: Stores information about the Android device extracted using uiautomator_deviceinfo.py.

apk_metadata.db: Saves APK metadata, including package name, version, permissions, and activities, extracted using chrome_analysis.py.

zebpay_data.db: Stores cryptocurrency data scraped from the ZebPay API using zeb.py.

Install Dependencies

Install the required libraries:

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

Alternatively, install the libraries manually:

```
pip install fastapi celery plotly appium uiautomator2 androguard requests schedule
```

Running the Project

1. Activate Virtual Environment:

Ensure you are in the correct virtual environment.

```
source venv/bin/activate  # For Linux/Mac
. env\Scripts ctivate    # For Windows
```

2. Start the Celery Worker and FastAPI Server:

Run the following command to start the project.

```
python start_worker.py
```

3. Navigate to Dashboard:

Open your web browser and visit http://127.0.0.1:8000 to view the real-time dashboard displaying:

Device information

APK analysis

Cryptocurrency data