Instagram Impersonation Detection

Overview

Project Name	Instagram Impersonation Detection
Description	A modular **open-source investigation toolkit** designed to automatically identify potential **Instagram impersonation accounts**. It discovers Instagram profiles that match a person's name, collects public metadata via the Meta Graph API, downloads their profile photos, and compares those photos to a reference image (such as the real person's picture) to detect identical or visually similar profiles.
Features	• **Automatic account discovery — finds Instagram profiles matching a person's name.** • **Metadata enrichment — retrieves follower counts, bios, and other account details via Meta Graph API.** • **Local evidence storage — saves structured CSVs, URL lists, and profile images per investigated name.** • **Impersonation detection — compares a provided reference photo to downloaded profile pictures.** • **Safe re-runs — detects existing data and asks whether to overwrite or reuse it.**

Methodology & Design Decisions

1.Candidate Discovery (Name → Usernames)

- Uses **Google Custom Search API (CSE)** to perform queries such as:

site:instagram.com "<name>" -inurl:reel -inurl:reels -inurl:p/ -inurl:stories -inurl:explore -inurl:accounts -inurl:tags -inurl:tv

- Extracts usernames from result URLs, filters irrelevant paths, deduplicates, and ranks results using **RapidFuzz token_set_ratio** for fuzzy name matching.
- Keeps the best-scoring username per candidate profile.

2. Metadata Enrichment

- Calls the **Meta Graph API (Business Discovery)** to retrieve structured profile information:
  - account name
  - username
  - bio / description
  - followers / following count
  - number of posts
  - profile picture URL
- Results are exported to CSV and plain-text URL lists.

3. Artifact Organization

All results are saved to a structured directory:

Instagram-Impersonation-Detection/

```
├── instagram_osint_pipeline.py   # Main automation pipeline
├── find_image_match.py           # Image matching module
├── Main_Folder/
│   └── <Searched_Name>/
│       ├── <Searched_Name>.csv
│       ├── profile urls.txt
│       └── Images/
└── README.md
```

For example :

```
Main_Folder/
└── John Doe/
    ├── John Doe.csv
    ├── profile urls.txt
    └── Images/
        ├── johndoe.jpg
        ├── johndoe_97.png
```

4. Impersonation Detection (Image Matching)

- Implemented in a separate module find_image_match.py.
- The main script automatically calls it, passing the person's Images/ folder and letting the analyst provide a **reference image path from anywhere on the computer**.
- Matching pipeline:
  - **Exact match:** file SHA-256 and pixel comparison
  - **Near-duplicate:** perceptual hashing (phash, dhash, ahash)
  - **Robust fallback:** ORB feature matching (OpenCV, optional)
- Handles rotations and flips to detect reused or slightly edited profile photos.
- Returns any exact or similar matches and their corresponding account names and URLs.

Tools, Libraries, and APIs

Category	Library / API	Purpose
Discovery	Google Custom Search API	Finds public Instagram URLs that match a target name
Metadata	Meta Graph API (Business Discovery)	Retrieves public account info and profile picture URLs
Fuzzy matching	RapidFuzz	Computes name similarity between input and found profiles
Networking	Requests	HTTP client for API calls and image downloads
Data & Reporting	Pandas, csv, tqdm	CSV export, console tables, and progress bars
Image analysis	Pillow (PIL), ImageHash, OpenCV (optional)	Image normalization, hashing, and ORB feature comparison
Core language	Python 3.x	Automation

Limitations & Assumptions

Limitations	Description
CSE Coverage	Results depend on Google's index; some profiles may be omitted or regionally ranked. (Gets only active accounts with Posts and high number or followers and followings - Our target)
Graph API Access	Only public Business/Creator accounts can be queried. Private/personal accounts are excluded.
Token Validity	Requires a working long-lived access token; expired tokens must be refreshed.
URL Expiry	CDN image links can expire; older data may need re-download.
Matching Accuracy	Heavy edits or filters may reduce perceptual-hash accuracy. ORB fallback improves robustness but is optional.
Rate Limits	Google and Meta enforce API quotas; large runs may trigger throttling.
Security	Hard-coded API keys are for demonstration. In production, store credentials in .env or a secrets vault.
Intended Use	Google and Meta terms.