CRIMINAL RECORD CHECKER

NAME: PRADITA G

STEPS:

- **Input Acquisition:**
  When the user enters a name, the program automatically searches through official websites like Interpol**,** CBI, and FBI to check if there are any criminal records associated with that name. It retrieves details such as:
- The full name of the person listed in the records.
- Their charges (if applicable).
- The notice type (e.g., Red Notice, Yellow Notice)
- A direct link to their official notice on the respective agency's website, allowing the user to study their profile further if needed.

- **Result Handling:**
- If a match is found, all the details are displayed clearly for the user.
- If no record is found, the program prints a red-coloured statement indicating that the person is not listed in the official databases. This makes it visually clear to the user that no results were found.

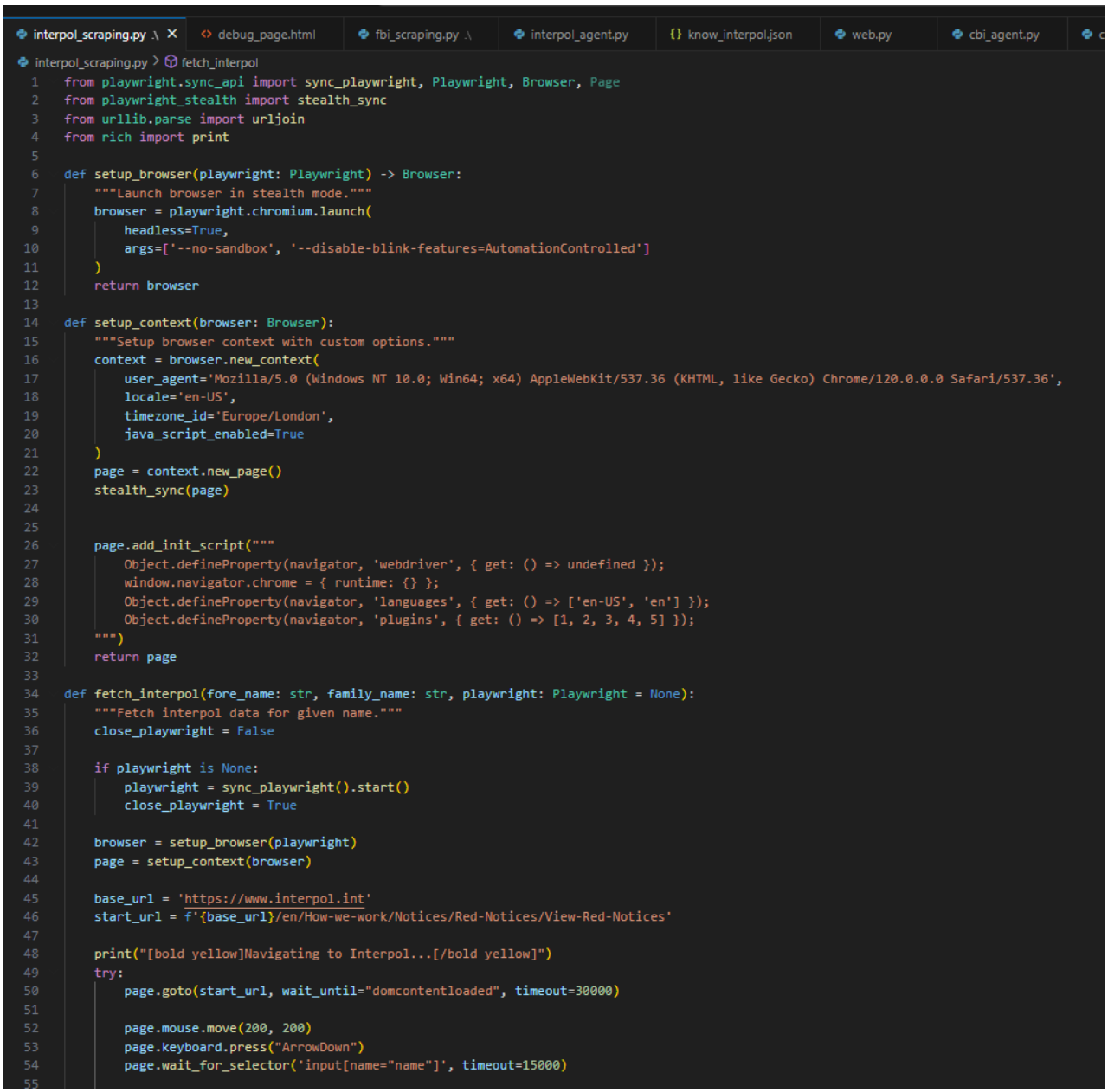
- **Parallel Checking for Speed:**
  The program is designed to query multiple sources (Interpol, CBI, FBI**)** in quick succession, simulating parallel checking. This significantly improves efficiency and reduces the time needed to gather data from all these official websites.

- **Reliable and Authentic Information:**
  Since the data is fetched directly from government and law enforcement websites, it is genuine and trustworthy**.** The system ensures that only accurate and up-to-date records are shown to the user, avoiding false positives.

LIBRARIES USED:

- Playwright
- Playwright_stealth
- Rich: formatting purposes
- Sync_playwright.

CODE SNIPPET OF INTERPOL CHECKER:



interpol_scraping.py .\ | debug_page.html | fbi_scraping.py .\ | interpol_agent.py | know_interpol.json | web.py | cbi_agent.py

interpol_scraping.py > fetch_interpol

```python
from playwright.sync_api import sync_playwright, Playwright, Browser, Page
from playwright_stealth import stealth_sync
from urllib.parse import urljoin
from rich import print

def setup_browser(playwright: Playwright) -> Browser:
    """Launch browser in stealth mode."""
    browser = playwright.chromium.launch(
        headless=True,
        args=['--no-sandbox', '--disable-blink-features=AutomationControlled']
    )
    return browser

def setup_context(browser: Browser):
    """Setup browser context with custom options."""
    context = browser.new_context(
        user_agent='Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36',
        locale='en-US',
        timezone_id='Europe/London',
        java_script_enabled=True
    )
    page = context.new_page()
    stealth_sync(page)


    page.add_init_script("""
        Object.defineProperty(navigator, 'webdriver', { get: () => undefined });
        window.navigator.chrome = { runtime: {} };
        Object.defineProperty(navigator, 'languages', { get: () => ['en-US', 'en'] });
        Object.defineProperty(navigator, 'plugins', { get: () => [1, 2, 3, 4, 5] });
    """)
    return page

def fetch_interpol(fore_name: str, family_name: str, playwright: Playwright = None):
    """Fetch interpol data for given name."""
    close_playwright = False

    if playwright is None:
        playwright = sync_playwright().start()
        close_playwright = True

    browser = setup_browser(playwright)
    page = setup_context(browser)

    base_url = 'https://www.interpol.int'
    start_url = f'{base_url}/en/How-we-work/Notices/Red-Notices/View-Red-Notices'

    print("[bold yellow]Navigating to Interpol...[/bold yellow]")
    try:
        page.goto(start_url, wait_until="domcontentloaded", timeout=30000)

        page.mouse.move(200, 200)
        page.keyboard.press("ArrowDown")
        page.wait_for_selector('input[name="name"]', timeout=15000)
```

interpol_scraping.py | debug_page.html | fbi_scraping.py | interpol_agent.py | know_interpol.json | web.py

interpol_scraping.py > fetch_interpol

```python
def fetch_interpol(fore_name: str, family_name: str, playwright: Playwright = None):
        page.fill('input[name="name"]', family_name)
        page.fill('input[name="forename"]', fore_name)
        page.click('button[type="submit"]')

        page.wait_for_load_state("networkidle")
        page.wait_for_timeout(3000)

        links = page.query_selector_all('.redNoticeItem__labelLink')

        if not links:
            print("[bold red]No results found in Interpol.[/bold red]")
            return

        results = []
        for link in links:
            name_text = link.inner_text().replace('\n', ' ').strip()
            full_url = link.get_attribute('data-singleurl')
            if full_url and family_name.lower() in name_text.lower() and fore_name.lower() in name_text.lower():
                safe_url = urljoin(base_url, full_url)
                notice_number = safe_url.split("/")[-1]
                results.append((name_text, notice_number, safe_url))

        if not results:
            print("[bold red]No matching results for the given name in interpol.[/bold red]")
            return


        selected_name, notice_number, selected_url = results[0]
        print(f"\n[bold yellow]Found:[/bold yellow] [bold yellow]{selected_name} in interpol!![/bold yellow]")
        print(f"[bold yellow]Notice Number:[/bold yellow] {notice_number}")
        print(f"[bold yellow]Profile URL:[/bold yellow] {selected_url}")

        page.goto(selected_url, wait_until="domcontentloaded", timeout=20000)
        page.wait_for_timeout(2000)


        try:
            json_data = page.evaluate('() => JSON.parse(document.body.innerText)')
            arrest_warrants = json_data.get('arrest_warrants', [])
            charges = arrest_warrants[0].get('charge', 'N/A') if arrest_warrants else 'N/A'
        except Exception:
            charges = "N/A"

        print(f"[bold yellow]Charges:[/bold yellow] {charges}")

    except Exception as e:
        print(f"[bold red]Error during Interpol fetch:[/bold red] {e}")

    finally:
        browser.close()
        if close_playwright:
            playwright.stop()
```

```python
if __name__ == "__main__":
    with sync_playwright() as pw:
        fore_name = input("Please enter the fore name: ").strip()
        family_name = input("Please enter the family name: ").strip()
        fetch_interpol(fore_name, family_name, pw)
```

EXPLANATION OF THE CODE SNIPPET:

- This Python script automates the process of searching Interpol Red Notices for a given fore name and family name using Playwright.
- It launches a Chromium browser in **stealth mode** and custom JavaScript overrides) to bypass bot detection and mimic human behaviour.
- The script navigates to the Interpol Red Notices page, fills in the search form with the user-provided details, submits it, and waits for results to load.
- It then extracts matching entries, retrieves key details such as the suspect's name, notice number, profile URL, and charges (if available), and displays them in a clean, styled format using the rich library.
- The use of stealth techniques and Playwright's headless automation ensures smooth data fetching while minimizing the chances of being blocked.

CODE SNIPPET OF FBI SCRAPING:

fbi_scraping.py > fetch_fugitives

```python
from playwright.sync_api import sync_playwright
from playwright_stealth import stealth_sync
from rich import print

def fetch_fugitives(name_input):
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        context = browser.new_context()
        page = context.new_page()
        stealth_sync(page)

        base_url = "https://www.fbi.gov"
        wanted_url = f"{base_url}/wanted/fugitives"

        print("[bold cyan]Accessing FBI Wanted Fugitives...[/bold cyan]")
        page.goto(wanted_url, wait_until="domcontentloaded")

        try:
            page.wait_for_selector("li.portal-type-person", timeout=15000)
        except:
            print("[bold red]Failed to load fugitive data.[/bold red]")
            return

        cards = page.query_selector_all("li.portal-type-person")
        results = []

        for card in cards:
            try:
                name_el = card.query_selector("p.name a")
                category_el = card.query_selector("h3.title a")
                link = name_el.get_attribute("href")
                name = name_el.inner_text().strip()
                category = category_el.inner_text().strip()

                if name_input.lower() in name.lower():
                    results.append((name, category, link))
            except:
                continue


        if results:
            print(f"[bold cyan]Found '{name_input}' in FBI!!:[/bold cyan]")
            for name, category, link in results:
                print(f"\n[bold cyan] {name}[/bold cyan]")
                print(f"Category: [green]{category}[/green]")
                print(f"Poster Link: [blue]{link}[/blue]")
        else:
            print(f"[bold red]No matches found in FBI!!.[/bold red]")

        context.close()
        browser.close()

```

EXPLANATION OF THE FBI SCRAPING:

- This Python script automates the search for wanted fugitives on the FBI website based on a user-provided name.
- Using **Playwright** for browser automation and Stealth to evade bot detection, it navigates to the FBI's "Wanted Fugitives" page, scrapes all fugitive entries, and filters them to find matches with the input name.
- For each matching fugitive, it extracts their full name, category (such as "Crimes Against Children" or "Violent Crimes"), and the link to their poster, then prints these details using the rich library for a styled output.
- The script runs headlessly, ensuring smooth, fast data retrieval, and closes the browser context after execution to maintain clean resource usage.

CODE SNIPPET FOR CBI SCRAPING:

```python
from playwright.sync_api import sync_playwright
from playwright_stealth import stealth_sync
from rich import print


def stealthify(page):
    """Apply stealth measures to evade bot detection"""
    stealth_sync(page)
    page.add_init_script("""
        Object.defineProperty(navigator, 'webdriver', { get: () => undefined });
        window.navigator.chrome = { runtime: {} };
        Object.defineProperty(navigator, 'languages', { get: () => ['en-US', 'en'] });
        Object.defineProperty(navigator, 'plugins', { get: () => [1, 2, 3, 4, 5] });
    """)


def safe_inner_text(page, selector, default="N/A"):
    """Safely get inner text from selector, return default if missing"""
    try:
        el = page.query_selector(selector)
        return el.inner_text().strip() if el else default
    except:
        return default


def fetch_cbi_notices(query: str, base_url: str, notice_type: str, context):
    """Fetch data from given CBI notice type URL"""
    page = context.new_page()
    stealthify(page)

    print(f"[bold purple]Checking {notice_type} Notices in CBI…[/bold purple]")

    try:
        page.goto(base_url, wait_until="domcontentloaded", timeout=30000)
    except Exception:
        print(f"[bold red]Failed to load {notice_type} Notice page. Skipping.[/bold red]")
        page.close()
        return

    try:
        page.wait_for_selector("#searchterm", timeout=15000)
    except:
        print(f"[bold red]Search box not found on {notice_type} page. Layout may have changed.[/bold red]")
        page.close()
        return

    if not query.strip():
        print("[bold red]Empty search query provided. Exiting.[/bold red]")
        page.close()
        return

    page.fill("#searchterm", query.strip())
    page.keyboard.press("Enter")
    print(f"[bold purple]Searching {notice_type} for: {query}…[/bold purple]")

    page.wait_for_timeout(3000)

```

```python
cbi_scraping.py > ...
def fetch_cbi_notices(query: str, base_url: str, notice_type: str, context):
    try:
        page.wait_for_selector("table#clipsid tbody tr", timeout=10000)
    except:
        print(f"[bold red]No results found for {notice_type} Notices in CBI!!.[/bold red]")
        page.close()
        return

    rows = page.query_selector_all("table#clipsid tbody tr")
    if not rows:
        print(f"[bold red]No results found in {notice_type} Notices table in CBI!!.[/bold red]")
        page.close()
        return

    found_any = False

    for idx, row in enumerate(rows, start=1):
        try:
            name_el = row.query_selector("td:nth-child(2) a")
            if not name_el:
                continue

            found_any = True

            name = safe_inner_text(row, "td:nth-child(2) a")
            link = name_el.get_attribute("href")

            print(f"\n[bold purple]{name}[/bold purple] ({notice_type} Notice)")
            print(f"Profile Link: [blue]{link}[/blue]")

            # Open detail page
            detail_page = context.new_page()
            stealthify(detail_page)
            try:
                detail_page.goto(link, wait_until="domcontentloaded", timeout=30000)
            except:
                print("[bold red]Failed to load detail page. Skipping.[/bold red]")
                detail_page.close()
                continue

            try:
                detail_page.wait_for_selector("div.wantedsingle__colright", timeout=15000)
            except:
                print("[bold red]Detail page failed to load properly.[/bold red]")
                detail_page.close()
                continue

            if notice_type == "Red":

                charges = safe_inner_text(detail_page, "#charge", default="No charges listed.")
                print(f"[bold purple]Charges:[/bold purple] {charges}")



            detail_page.close()
```

```python
def fetch_cbi_notices(query: str, base_url: str, notice_type: str, context):
                detail_page.close()

            except Exception as e:
                print(f"[bold red]Error processing row {idx} in {notice_type} Notices: {e}[/bold red]")

    if not found_any:
        print(f"[bold red]No results found for {notice_type} Notices in CBI!!.[/bold red]")

    page.close()


def fetch_cbi_interpol(query: str):
    RED_NOTICE_URL = "https://cbi.gov.in/interpol-red-notice"
    YELLOW_NOTICE_URL = "https://cbi.gov.in/interpol-yellow-notice"

    with sync_playwright() as playwright:
        browser = playwright.chromium.launch(
            headless=True,
            args=["--no-sandbox", "--disable-blink-features=AutomationControlled"]
        )
        context = browser.new_context(
            user_agent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36",
            locale="en-US",
            color_scheme="light",
            timezone_id="Asia/Kolkata",
            java_script_enabled=True
        )


        fetch_cbi_notices(query, RED_NOTICE_URL, "Red", context)
        fetch_cbi_notices(query, YELLOW_NOTICE_URL, "Yellow", context)

        browser.close()


if __name__ == "__main__":
    name_query = input("Enter name to search: ").strip()
    fetch_cbi_interpol(name_query)
```

EXPLANATION FOR THIS CODE SNIPPET:

- It opens the CBI website for **Red Notices** (criminal suspects) and **Yellow Notices** (missing persons), fills the search box with the given name, and fetches the results.
- For each match, it prints the person's name and profile link. If it's a **Red Notice**, it also extracts and shows their **charges**.
- The script uses stealthify() to make the browser look more human-like and avoid being blocked.
- It handles errors like no results found, page load failures, or missing elements gracefully.

COLLECTOR AGENT:

collector_agent.py > collector_agent

```python
import interpol_scraping
import cbi_scraping
import fbi_scraping
from rich import print
from concurrent.futures import ThreadPoolExecutor

def collector_agent():
    fore_name = input("Enter Fore Name (First Name): ").strip()
    family_name = input("Enter Family Name (Last Name): ").strip()
    full_name = f"{fore_name} {family_name}"

    print("[bold green]Checking for the name in three sites FBI/INTERPOL/CBI:[/bold green]")


    def check_site(site_name, scraper_func, *args):
        print(f"[bold green]Checking in {site_name}...[/bold green]")
        result = scraper_func(*args)
        if result:
            print(f"[bold green] FOUND in {site_name}![/bold green]")
            print(result)


    with ThreadPoolExecutor(max_workers=3) as executor:

        executor.submit(check_site, "FBI", fbi_scraping.fetch_fugitives, full_name)
        executor.submit(check_site, "INTERPOL", interpol_scraping.fetch_interpol, fore_name, family_name)
        executor.submit(check_site, "CBI", cbi_scraping.fetch_cbi_interpol, full_name)

    print("[bold yellow] Finished checking!![/bold yellow]")


collector_agent()

```

This collects the three together and provides the final result.

OUTPUT:

```
(.venv) PS D:\codings\machine_learning\agent_detection> python collector_agent.py
Enter Fore Name (First Name): Samantha
Enter Family Name (Last Name): lewthwaite
Checking for the name in three sites FBI/INTERPOL/CBI:
Checking in FBI...
Checking in INTERPOL...
Checking in CBI...
Accessing FBI Wanted Fugitives...
Checking Red Notices in CBI…
Navigating to Interpol...
Searching Red for: Samantha lewthwaite…
No matches found in FBI!!.

Found: LEWTHWAITE SAMANTHA LOUISE in interpol!!
Notice Number: 2018-89546
Profile URL: https://ws-public.interpol.int/notices/v1/red/2018-89546
No results found for Red Notices in CBI!!.
Checking Yellow Notices in CBI…
Searching Yellow for: Samantha lewthwaite…
Charges: 1. Being in possession of explosives
2. Conspiracy to commit a felony
No results found for Yellow Notices in CBI!!.
 Finished checking!!
(.venv) PS D:\codings\machine_learning\agent_detection>
```