

Intel Gathering Platform And Analysis Using Machine Learning

A MINI-PROJECT REPORT

Submitted by
JERISH DANIEL J 312323205095

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



**St. JOSEPH'S COLLEGE OF
ENGINEERING (An Autonomous Institution)**

St. Joseph's Group of Institutions

OMR, Chennai 600 119

ANNA UNIVERSITY: CHENNAI 600

025 April-2024

ANNA UNIVERSITY: CHENNAI 600 025

ABSTRACT

In today's digital landscape, Law Enforcement Agencies (LEAs) face a daunting challenge: effectively harnessing the vast array of open-source intelligence (OSINT) available to combat crime. The necessity for a solution that streamlines the gathering, integration, and analysis of OSINT has never been more pressing. This mini-project endeavors to provide just that: a tailored unified platform designed specifically for LEAs, simplifying their access to the wealth of information dispersed across diverse OSINT repositories. By harnessing the power of automation, this platform excels at extracting data from a myriad of sources, seamlessly integrating it, and offering sophisticated analysis tools. The result? LEAs are presented with a comprehensive view of combined intelligence through an intuitive and user-friendly interface. This streamlined approach empowers law enforcement personnel to make informed decisions rapidly, expediting investigations and enhancing their overall efficacy. The significance of this initiative cannot be overstated. By facilitating proactive and targeted interventions, it equips law enforcement with the tools necessary to effectively address contemporary challenges in crime prevention. Ultimately, this unified platform represents a pivotal advancement in law enforcement's ability to leverage OSINT, ushering in a new era of efficiency and effectiveness in combating crime.

CHAPTER 1

INTRODUCTION

In the intricate web of contemporary digital crime and security challenges, Law Enforcement Agencies (LEAs) navigate a labyrinthine landscape, tasked with the formidable mission of harnessing the boundless expanse of open-source intelligence (OSINT) to combat the relentless evolution of criminal activities. The imperative for a transformative solution that not only streamlines but revolutionizes the intricate processes of gathering, synthesizing, and deciphering OSINT data has never been more urgent. This mini-project emerges as a beacon of innovation, poised to meet this pressing need head-on by presenting a meticulously crafted unified platform expressly tailored for LEAs, serving as a nexus to seamlessly tap into the expansive reservoir of information sprawled across myriad OSINT repositories.

At the heart of this initiative lies the fusion of cutting-edge automation technology with advanced analytical capabilities, culminating in a platform that transcends the conventional boundaries of data aggregation and interpretation. Harnessing the power of automation, this platform exhibits unparalleled adeptness in sifting through the labyrinth of digital data sources, distilling relevant insights, and orchestrating their seamless integration into a cohesive framework. Equipped with a suite of sophisticated analysis tools, LEAs are empowered to traverse the intricate web of intelligence with precision and agility, leveraging an intuitive and user-centric interface to navigate the complexities of the digital realm with ease.

The significance of this pioneering endeavor extends far beyond mere operational efficiency; it heralds a transformative paradigm shift in law enforcement methodology. By furnishing LEAs with the requisite tools to not only keep pace but stay ahead of the curve in the ever-evolving landscape of digital crime, this platform embodies a pivotal advancement in crime prevention efforts. Armed with proactive intervention capabilities and unprecedented insights, law enforcement personnel are empowered to confront the multifaceted challenges of contemporary criminality with unwavering efficacy and resolve. In essence, this unified platform stands as a testament to the relentless pursuit of innovation in law enforcement, ushering in a new era of resilience, adaptability, and efficacy in the ceaseless battle against digital crime.

CHAPTER 2

LITERATURE REVIEW

In developing the hands-free volume control system, a thorough literature survey was conducted to gather insights and methodologies from existing research and studies. The following summarizes the key references used to inform and guide various aspects of this project.

- [1] "Current Challenges and Future Research Areas for Digital Forensic Investigation" - David Lillis et al. - 2020

This paper delves into challenges in digital forensics, which intersect significantly with OSINT, particularly in data volume and complexity. It highlights issues of data privacy, the need for improved processing techniques for large datasets, and the difficulty of integrating data from diverse sources.

- [2] "Toward Automated Fact-Checking: Detecting Checkworthy Factual Claims by ClaimBuster" - Naeemul Hassan et al. - 2017

Addressing automated fact-checking, this paper focuses on techniques to identify checkworthy statements. It discusses limitations of natural language processing tools in detecting subtlety, nuance, and context within textual data.

- [3] "Machine Learning for Cybersecurity and Decision Support: a Survey of Trends and Techniques" - Talha Ongun et al. - 2020

This survey explores machine learning in cybersecurity, including its use for

3

OSINT. It highlights challenges such as adversarial attacks on machine learning models, the need for robust systems, and bias in training datasets.

- [4] "Social Media Data Analysis for Proactive Policing and Community Policing"

- Chuxu Zhang et al. - 2019

Exploring social media data in law enforcement via OSINT techniques, this paper raises ethical concerns about surveillance, data accuracy, and potential social biases affecting analysis outcomes.

- [5] "Automated Information Extraction from Multilingual Open Source Data for Proactive Event Detection" - José M. Fernández et al. - 2021

Addressing multilingual data extraction in OSINT, this study discusses limitations of language-dependent tools, challenges in cross-language data fusion, and semantic analysis complexities.

- [6] "The Limits of Artificial Intelligence in National Security" - Peter W. Singer and Gregory C. Allen - 2021

This paper discusses broader national security applications of AI, including OSINT, and highlights issues of reliability, ethical concerns, biases in AI decision-making, and understanding contextual nuances.

- [7] "Automating Open Source Intelligence: Algorithms, Approaches, and Applications" - Robert Layton, Paul A. Watters - 2021

Providing updates to their previous work, this paper focuses on recent algorithmic approaches and tools for OSINT. It addresses data veracity issues, real-time data analysis challenges, and misinformation handling.

- [8] "Challenges in Data-Driven Approaches to OSINT: The Case of Terrorist Content Online" - A. Edwards et al. - 2022

Examining data-driven approaches to detect terrorist content online, this paper discusses problems with false positives, privacy concerns, dynamic online content, and evolving legal frameworks.

[9] "OSINT in the Age of Information Overload: Challenges and Opportunities for Machine Learning" - Hanna Linder et al. - 2023

Discussing machine learning's role in filtering and analyzing vast amounts of data in OSINT, this paper highlights challenges like information overload, adapting models to new data, and computational resource needs.

[10] "Ethical Dilemmas in Automated OSINT Systems: A Social Science Perspective" - Julia Johnson - 2024

Focusing on the ethical aspects of OSINT automation, this paper addresses surveillance implications, consent issues, and impacts on public trust.

CHAPTER 3

SYSTEM ANALYSIS

3.1 SYSTEM REQUIREMENTS

In the development of our OSINT automation platform, consideration of system requirements was paramount to ensure optimal performance and compatibility. The following are the key system requirements for our platform:

3.1.1 HARDWARE REQUIREMENTS:

The platform requires hardware capable of supporting modern web development frameworks and tools. This includes sufficient RAM, CPU power, and storage space to handle the processing and storage demands of data-intensive operations.

- A PC or laptop with a minimum of 4GB RAM and a decent processor (i.e., Intel i5 or equivalent).

3.1.2 SOFTWARE REQUIREMENTS:

- Operating System: Windows 10, macOS, or Linux.
- Python 3.6 or higher installed.
- Web browsers for user interaction
- Development Environments like Visual Studio Code for coding and testing

3.1.3 NETWORK REQUIREMENTS:

Reliable internet connectivity is essential for accessing external

6

data sources and facilitating communication between the frontend and backend components of the platform. Adequate bandwidth is necessary to ensure smooth data transfer and real-time updates.

- A stable internet connection of at least 1Mbps speed suitable for scanning and analyzing the data on the platform.

3.2 TOOLS AND TECHNOLOGY USED

In the development of our OSINT automation platform, we carefully selected technologies and tools that would enable efficient gathering, integration, and analysis of open-source intelligence. Our platform leverages modern web development frameworks such as React.js for the frontend, Django for the backend, and Node.js with Express.js for server-side scripting.

- **React.js:** React.js is a popular JavaScript library for building user interfaces. Its component-based architecture allows for the creation of reusable UI components, providing a seamless and interactive experience for our platform users.
- **Django:** Django is a high-level Python web framework that promotes rapid development and clean, pragmatic design. It provides built-in features for authentication, database management, and URL routing, making it ideal for developing robust backend systems.
- **Node.js with Express.js:** Node.js is a runtime environment for executing JavaScript code outside of a web browser, while Express.js is a minimalist web framework for Node.js. Together, they enable the development of lightweight and scalable server-side applications, handling API requests and business logic efficiently.

3.3 MODULES USED

Integration of various OSINT tools into our platform to enhance intelligence gathering and analysis capabilities. These tools include:

- **Social Media Analyzer:**

This tool enables the collection and analysis of data from social media platforms, allowing law enforcement agencies to monitor online activities and identify potential threats or criminal behavior.

- **Dark Dumpus:**

Dark Dumpus is a specialized OSINT tool designed to search and retrieve data from dark web sources. It provides access to hidden forums, marketplaces, and other underground networks, facilitating the discovery of illicit activities and criminal networks.

- **Blackbird:**

Blackbird is a comprehensive OSINT platform that aggregates data from diverse sources, including public records, news articles, and online forums. It employs advanced algorithms for data mining and pattern recognition, assisting analysts in uncovering valuable insights and trends.

By integrating these tools into our platform, we ensure that law enforcement agencies have access to a wide range of open-source intelligence sources, enabling them to make informed decisions and take proactive measures to combat crime effectively.

Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software is determined, software developers will design or employ designers to develop a plan for a solution. It includes construction component and algorithm implementation issues which shown in as the architectural view. During this chapter we will introduce some principle.

4.1 ARCHITECTURE DIAGRAM

The architecture diagram succinctly illustrates the frontend, backend, database, OSINT tool integration, and deployment infrastructure of our platform. It offers a concise overview of the system's structure, facilitating effective gathering, analysis, and visualization of open-source intelligence data for law enforcement agencies

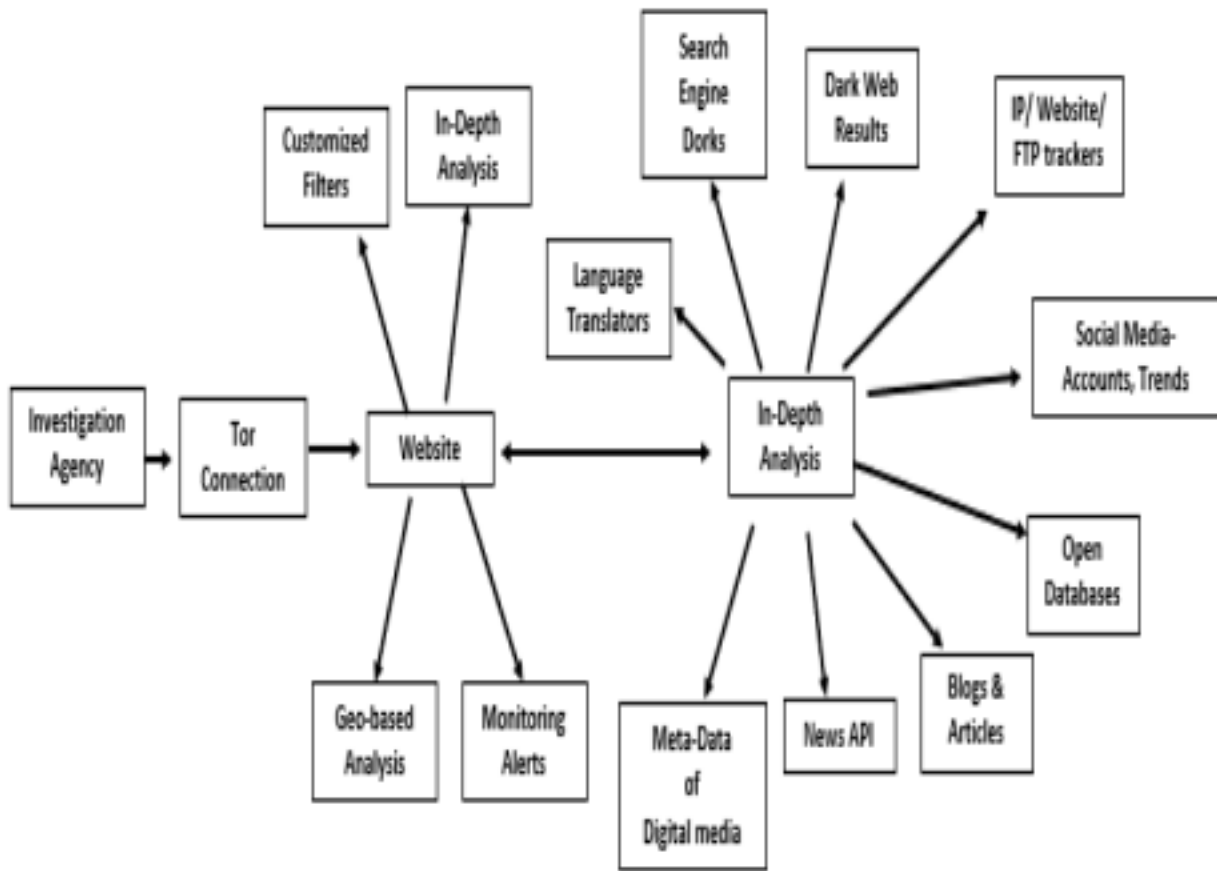


Fig 4.1.1

4.2 CLASS DIAGRAM

A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

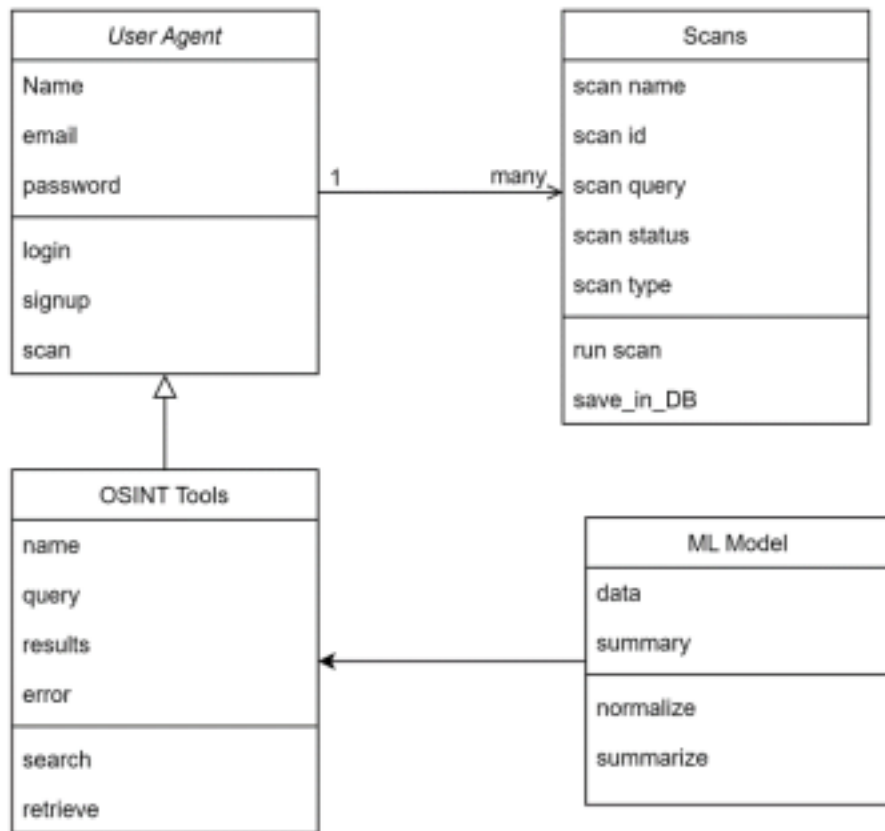


Fig 4.1.2

4.3 ACTIVITY DIAGRAM

Activity diagrams provide a visual representation of sequential and parallel activities within a system, aiding in understanding complex processes. They depict activities, transitions, decisions, and concurrency, offering a clear overview of workflow dynamics. These diagrams are valuable tools during software analysis and design, facilitating communication and decision-making among stakeholders.

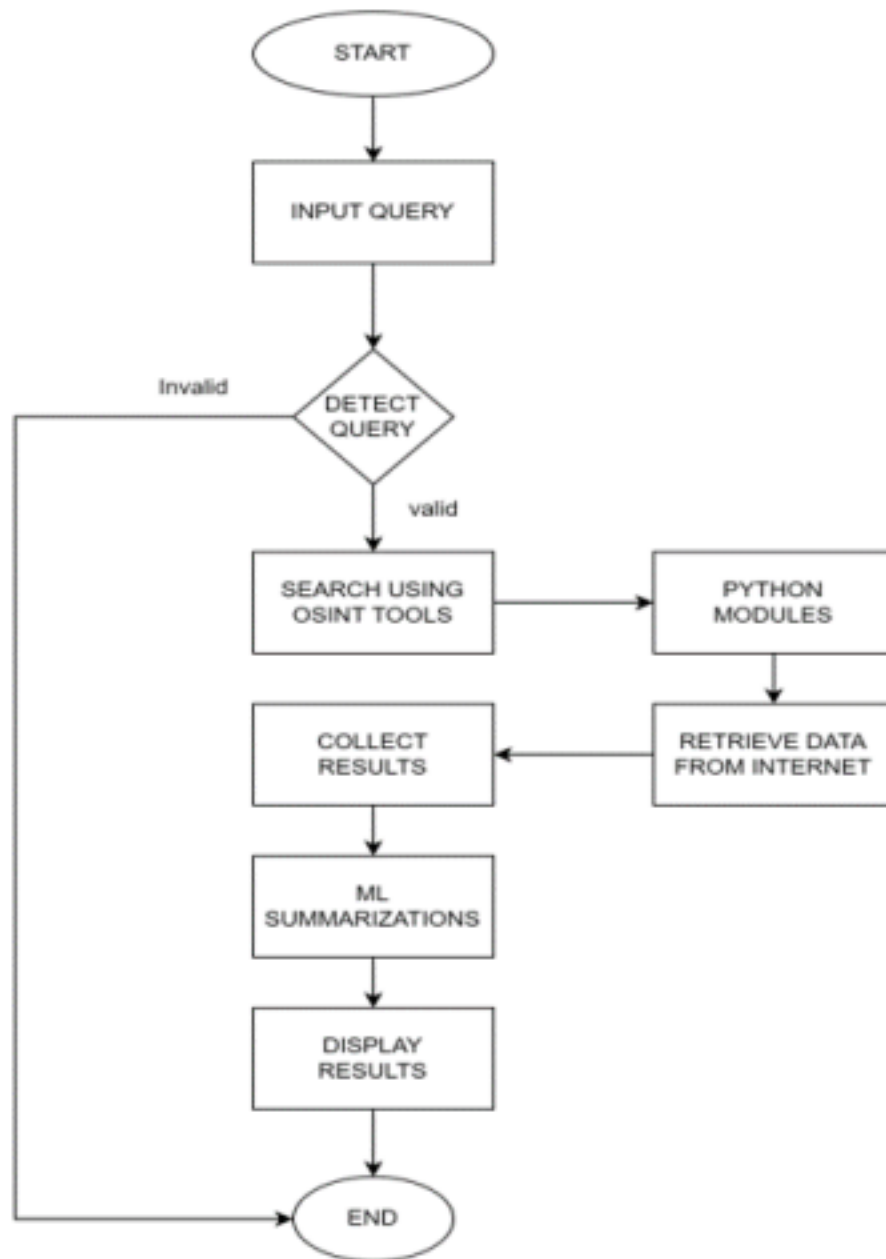


Fig 4.1.3

4.4 SEQUENCE DIAGRAM

A sequence diagram illustrates the interactions between objects or

components in a system over time. It displays the sequence of messages exchanged among these entities, showcasing the order of operations and the flow of control. Sequence diagrams are invaluable for visualizing system behavior, understanding communication patterns, and identifying potential design flaws or optimization opportunities.

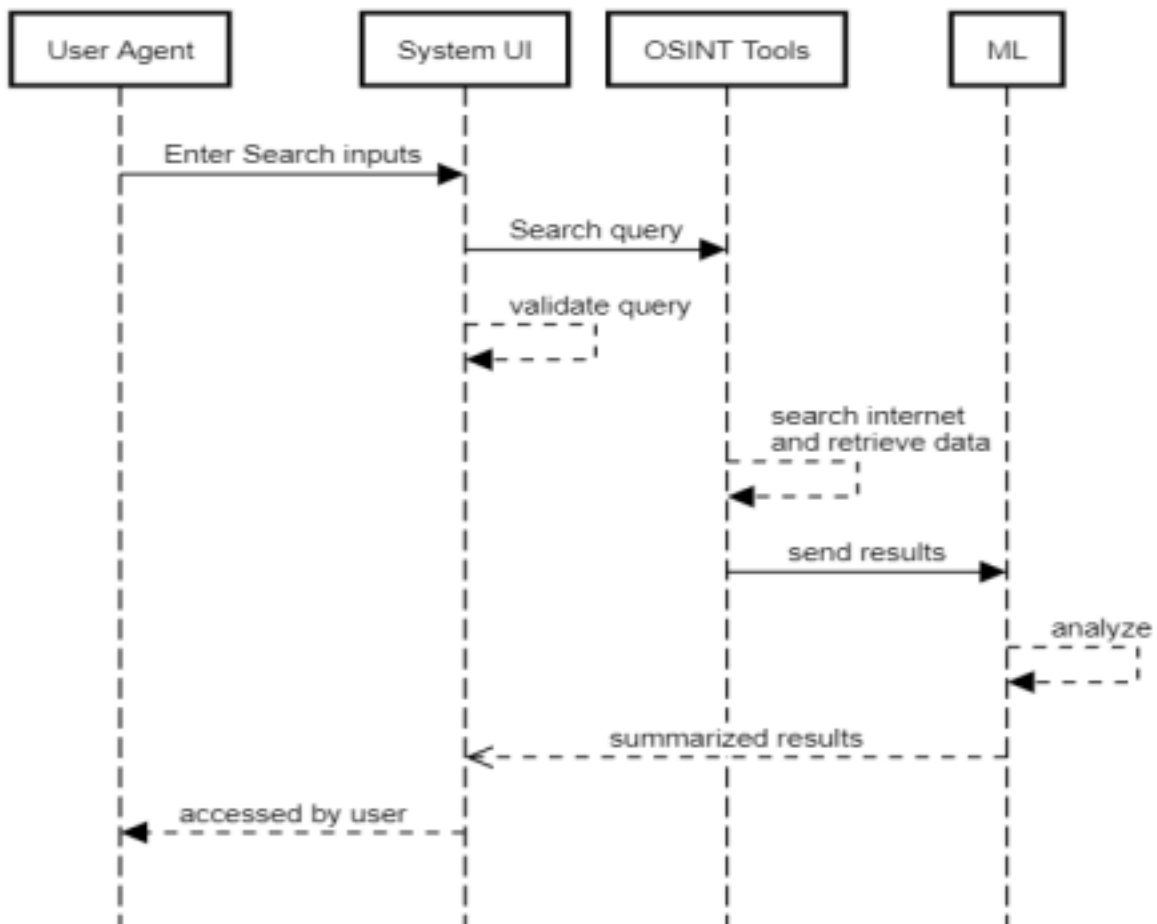


Fig 4.1.4

The OSINT automation platform has significantly enhanced intelligence gathering capabilities for law enforcement agencies, enabling them to collect, analyze, and act upon open-source intelligence data more effectively. Some key models for OSINT automations include:

- **Social Media Monitoring:** Automated monitoring of social media platforms has facilitated the detection of suspicious activities, criminal networks, and threats to public safety. Real-time alerts and notifications enable timely intervention and response by law enforcement authorities.
- **Dark Web Analysis:** Deep web and dark web analysis tools have provided insights into illicit activities, including cybercrime, illicit drug trafficking, and human trafficking. By monitoring dark web forums and marketplaces, law enforcement agencies can identify and disrupt criminal operations more proactively.
- **Geospatial Analysis:** Geospatial analysis tools have enabled the mapping and visualization of crime hotspots, criminal networks, and geographical trends. This spatial intelligence enhances situational awareness and informs strategic decision-making for crime prevention and enforcement efforts.
- **Sentiment Analysis:** Sentiment analysis of online content, including text, images, and videos, has provided insights into public perceptions, attitudes, and behaviors related to criminal activities. This sentiment intelligence helps law enforcement agencies gauge community sentiments and identify potential threats or risks.
- **Entity Recognition:** Automated entity recognition tools identify and extract relevant entities, such as names, locations, organizations, and events, from

unstructured text data. This entity intelligence enables law enforcement analysts to link related information and identify patterns or connections between disparate data sources.

Overall, the model for OSINT automations demonstrate the value and impact of leveraging advanced technologies and analytical techniques to augment intelligence capabilities and support law enforcement efforts in combating crime and maintaining public.

5.2 METHODOLOGIES

5.2.1 ENVIRONMENT SETUP

The first step involves setting up the development environment by installing necessary software and libraries. This includes Python for backend development and data processing, as well as frameworks like Django and Node.js for web development. Additionally, OSINT-specific libraries and tools such as BeautifulSoup for web scraping and NLTK for natural language processing are installed.

15

5.2.2 REQUIREMENT ANALYSIS

Conduct a thorough analysis of requirements gathered from stakeholders and literature surveys. Define functional and non-functional requirements for the OSINT automation platform, including data gathering, integration, analysis, and user interface features.

5.2.3 SYSTEM DESIGN

Design the architecture and components of the OSINT automation platform based on the defined requirements. Determine the structure of the frontend and backend components, as well as the data storage and processing mechanisms. Define interfaces for integrating OSINT tools and modules into the platform.

5.2.4 DEVELOPMENT AND IMPLEMENTATION

Develop and implement the core functionalities of the OSINT automation platform according to the design specifications. This includes building frontend interfaces for user interaction, backend services for data processing and analysis, and integration with OSINT tools and APIs.

5.2.5 TESTING

Conduct rigorous testing of the OSINT automation platform to ensure functionality, performance, and security. Perform unit testing, integration testing, and system testing to identify and resolve any defects or issues. Implement quality assurance processes to maintain code quality and reliability.

5.2.6 INTEGRATION

Deploy the OSINT automation platform to production environments, ensuring compatibility with existing infrastructure and systems. Configure integration with external data sources and APIs to enable seamless data gathering and analysis. Provide documentation and training for stakeholders on

platform usage and maintenance.

17

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 CONCLUSIONS

In conclusion, our project represents a significant advancement in leveraging open-source intelligence (OSINT) for law enforcement agencies. By integrating various technologies and tools, we've developed a robust platform for streamlined data gathering, analysis, and visualization. Moving forward, our

future work involves expanding the dataset, comparing XAI techniques, and developing real-time detection applications to better serve farmers and botany enthusiasts.

6.2 FUTURE WORKS

Another work that this study would like to pursue in the future is to provide a comparative study on different XAI techniques and implement a user study in order to find out which XAI technique provides the best explainability, transparency and interpretability. With the addition of data on Volatile organic compounds, soil types, environmental conditions and time of the month as mentioned by farmers through feedback from the user study, the user trust of the detection tool is expected to grow a little higher. As discussed earlier in the use case of this study, a working application that is capable of taking pictures of plants and detecting plant diseases in real-time is the

Looking ahead, we envision several opportunities for further innovation and improvement in the field of open-source intelligence:

- **Advanced Analytics:** Continued development of advanced analytics tools, including machine learning algorithms, natural language processing (NLP), and predictive modeling, to enhance intelligence analysis and prediction capabilities.
- **Global Collaboration:** Foster collaboration and information sharing among law enforcement agencies, intelligence communities, and international partners to address transnational threats and enhance collective security.
- **User-Centric Design:** Focus on user-centric design principles to enhance the usability, accessibility, and adoption of OSINT automation tools among law enforcement personnel and intelligence analysts.

- **Ethical and Legal Considerations:** Address ethical and legal considerations surrounding the collection, analysis, and dissemination of open-source intelligence data, including privacy concerns, data protection regulations, and human rights considerations.

In summary, our OSINT automation platform represents a powerful tool for empowering law enforcement agencies to combat crime effectively and safeguard public safety. By embracing innovation, collaboration, and user-centric design principles, we are committed to advancing the field of open-source intelligence and making a positive difference in the fight against crime.

Server.js :

```
const express = require('express');  
const mysql = require('mysql2');  
const cors = require('cors');  
const { exec } =  
require('child_process'); const fs =
```

```

require('fs');

const app = express();
const PORT = 3020;


const pool = mysql.createPool({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'osintika',
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0,
});

app.use(cors());
app.use(express.json());
app.get('/scans', async (req, res) => {
  try {
    let query = 'SELECT * FROM scanned';
    if (req.query.filter && req.query.filter !== 'None') {
      const filter = req.query.filter;
      query += ` WHERE status = '${filter}'`;
    }
    pool.query(query, (error, results) => {
      if (error) {
        console.error('Error fetching scans:', error);
        res.status(500).json({ error: 'Internal Server Error' });
        return;
      }
    });
  }

```

```

    res.json(results);
  });
  } catch (error) {
    console.error('Error fetching scans:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
app.delete('/scans/:id', async (req, res) => {
  const id = req.params.id;
  try {
    const query = 'DELETE FROM scanned WHERE id = ?';
    pool.query(query, [id], (error, results) => { if (error) {
      console.error('Error deleting scan:', error);
      res.status(500).json({ error: 'Internal Server Error' });
    }
    return;

```

21

```

    }
    if (results.affectedRows > 0) {
      res.status(204).send();
    } else {
      res.status(404).json({ error: 'Scan not found' });
    }
  });
  } catch (error) {
    console.error('Error deleting scan:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

```

```

app.get('/user/:userId', async (req, res) => {
  try {
    const userId = req.params.userId;
    console.log(userId);
    pool.query('SELECT data FROM scanned WHERE id = ?', [userId], async
(error, results) => {
      if (error) {
        console.error('Error fetching user data:', error);
        res.status(500).json({ error: 'Internal Server Error' });
        return;
      }
      if (results.length === 0) {
        res.status(404).json({ error: 'User not found' });
        return;
      }

```

22

```

const userData = results;
console.log(userData);
res.json(userData);
});
} catch (error) {
  console.error('Error fetching user data:', error);
  res.status(500).json({ error: 'Internal Server Error' });
}
});
app.post('/scans', async (req, res) => {
  const { scanName, scanTarget, selectedModules } = req.body;
  try {

```



```

const startTime = new Date();

pool.query(
  'INSERT INTO scanned (id, name, target, started, finished, status, elements,
data) VALUES (?, ?, ?, ?, ?, ?, ?, ?)',
  [
    null,
    scanName,
    scanTarget,
    startTime,
    '-',
    'Running',
    0,
    JSON.stringify({
      "socialMediaAnalyzer": [],
      "darkWebSearch": [],

      "metaDataAnalyzer": [],
      "newsGathering": [],
    }),
  ],
  async (error, results) => {
    if (error) {
      console.error('Error creating scan:', error);
      res.status(500).json({ error: 'Internal Server Error' });
      return;
    }
    const scanId = results.insertId;
    const shouldRunSocialMediaAnalyzer =
      selectedModules.includes('socialMediaAnalyzer');

```

```

if (shouldRunSocialMediaAnalyzer) {
  const username = scanTarget;
  const pythonCommand = `python blackbird/blackbird.py -u ${username}`;
  console.log("Running blackbird.py");
  const executePythonScript = () => {
    return new Promise((resolve, reject) => {
      exec(pythonCommand, (pythonError, stdout, stderr) => { if
(pythonError) {
        console.error(`Error executing Python script:
${pythonError.message}`);
        console.error(`Python script stderr: ${stderr}`);
        reject(pythonError);
      }
      return;
    }
  }

```

24

```

    console.log(`Python script output: ${stdout}`);
    resolve();
  });
});
};
await executePythonScript();
const jsonFilePath = `blackbird/results/${username}.json`;
const data = await new Promise((resolve, reject) => {
  fs.readFile(jsonFilePath, 'utf8', (readError, fileData) => { if
(readError) {
    console.error(`Error reading JSON file: ${readError.message}`);
    reject(readError);
  }
  return;
}

```

```

    }
    resolve(fileData);
  });
});
const jsonData = JSON.parse(data);
const sitesWithUrls = jsonData.sites.map(site => ({ site:
site.app,
url: site.url
}));
pool.query(
'UPDATE scanned SET data = JSON_SET(data,
"$$.socialMediaAnalyzer", ?) WHERE id = ?',
[JSON.stringify(sitesWithUrls), scanId],
(updateError) => {

```

25

```

if (updateError) {
  console.error('Error updating scan with module results:', updateError);
  res.status(500).json({ error: 'Internal Server Error' }); return;
}
res.status(201).json({ success: true, scanId });
}
);
} else {
  res.status(201).json({ success: true, scanId });
}
}
);

```

```
    } catch (error) {  
      console.error('Error creating scan:', error);  
      res.status(500).json({ error: 'Internal Server Error' });  
    }  
  });  
  
  app.listen(PORT, () => {  
    console.log(`Server is running on port ${PORT}`);  
  });
```

26

Blackbird.py:

```
import argparse  
import asyncio  
import json  
import os  
import random  
import subprocess  
import sys  
import time  
import warnings  
import csv  
from datetime import datetime  
import aiohttp
```

```

from bs4 import BeautifulSoup
from colorama import Fore, init

file = open("blackbird/data.json")
searchData = json.load(file)
currentOs = sys.platform
path = os.path.dirname(__file__)
warnings.filterwarnings("ignore")

useragents =
open("blackbird/useragents.txt").read().splitlines() proxy =
None

async def findUsername(username, interfaceType,
flag_csv=False): start_time = time.time()

27
timeout = aiohttp.ClientTimeout(total=20)
print(
f'{Fore.LIGHTYELLOW_EX}[!] Searching '{username}' across {len(searchData['sites'])}
social networks\033[0m"
)

async with aiohttp.ClientSession(timeout=timeout) as session:
tasks = []
for u in searchData["sites"]:
task = asyncio.ensure_future(
makeRequest(session, u, username, interfaceType)
)
tasks.append(task)
results = await asyncio.gather(*tasks)

```

```

now = datetime.now().strftime("%m/%d/%Y %H:%M:%S")
executionTime = round(time.time() - start_time, 1)
userJson = {
    "search-params": {
        "username": username,
        "sites-number": len(searchData["sites"]),
        "date": now,
        "execution-time": executionTime,
    },
    "sites": [],
}
for x in results:
    userJson["sites"].append(x)

```

28

```

pathSave = os.path.join(path, "results", username + ".json")
userFile = open(pathSave, "w")
json.dump(userJson, userFile, indent=4, sort_keys=True)
print(
    f'{Fore.LIGHTYELLOW_EX}(!) Search complete in {executionTime}
seconds\033[0m"
)
print(f'{Fore.LIGHTYELLOW_EX}(!) Results saved to {username}.json\033[0m") if
flag_csv:
    exportCsv(userJson)
return userJson

```

```

async def makeRequest(session, u, username, interfaceType):
    url = u["url"].format(username=username)

```

```

jsonBody = None
useragent = random.choice(useragents)
headers = {"User-Agent": useragent}
metadata = []
if "headers" in u:
headers.update(json.loads(u["headers"]))
if "json" in u:
jsonBody = u["json"].format(username=username)
jsonBody = json.loads(jsonBody)
try:
async with session.request(
u["method"], url, json=jsonBody, proxy=proxy, headers=headers, ssl=False ) as
response:

```

29

```

responseContent = await response.text()
if (
"content-type" in response.headers
and "application/json" in response.headers["Content-Type"] ):
jsonData = await response.json()
else:
soup = BeautifulSoup(responseContent, "html.parser")

if eval(u["valid"]):
print(
f {Fore.LIGHTGREEN_EX} [ + ] \033[0m - # {u["id"]}
{Fore.BLUE} {u["app"]} \033[0m {Fore.LIGHTGREEN_EX} account found \033[0m
- {Fore.YELLOW} {url} \033[0m [ {response.status} {response.reason} ] \033[0m' )
if "metadata" in u:

```

```

metadata = []
for d in u["metadata"]:
    try:
        value = eval(d["value"]).strip("\t\r\n")
        print(f" |--{d['key']}: {value}")
        metadata.append(
            {"type": d["type"], "key": d["key"], "value": value} )
    except Exception as e:
        pass
return {

```

30

```

    "id": u["id"],
    "app": u["app"],
    "url": url,
    "response-status": f"{response.status} {response.reason}",
    "status": "FOUND",
    "error-message": None,
    "metadata": metadata,
}
else:
    if interfaceType == "CLI":
        if showAll:
            print(
                f"[-]\033[0m - #{u['id']} {Fore.BLUE} {u['app']}\033[0m account not found -
                {Fore.YELLOW} {url}\033[0m [{response.status} {response.reason}]\033[0m' )
            return {
                "id": u["id"],
                "app": u["app"],

```



```

"url": url,
"response-status": f"{response.status} {response.reason}",
"status": "NOT FOUND",
"error-message": None,
"metadata": metadata,
}

```

```

except Exception as e:

```

```

if interfaceType == "CLI":

```

```

if showAll:

```

31

```

print(
    f'{Fore.RED}[X]\033[0m - #{u["id"]} {Fore.BLUE} {u["app"]}\033[0m error on request'
    f'({repr(e)})- {Fore.YELLOW} {url}\033[0m'
)

```

```

return {

```

```

    "id": u["id"],

```

```

    "app": u["app"],

```

```

    "url": url,

```

```

    "response-status": None,

```

```

    "status": "ERROR",

```

```

    "error-message": repr(e),

```

```

    "metadata": metadata,

```

```

}

```

```

def list_sites():

```

```

    for i, u in enumerate(searchData["sites"], 1):

```

```

        print(f'{i}. {u["app"]}')

```

APPENDIX 2

FRONTEND:

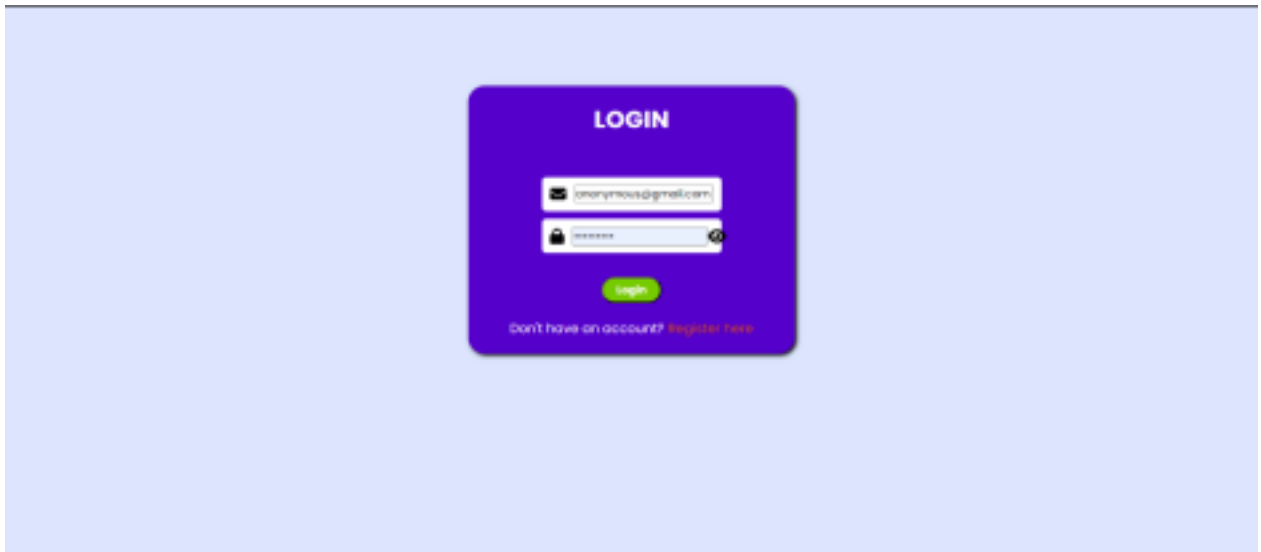
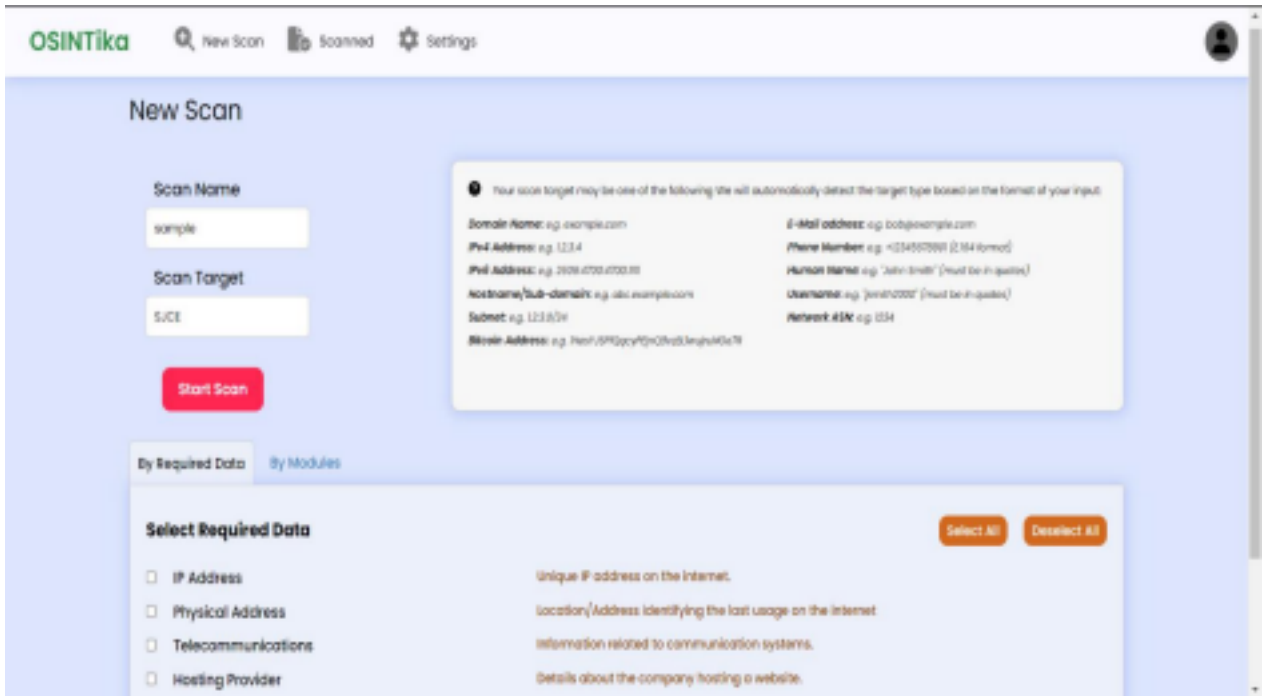


Fig A2.1



OSINTika New Scan Scanned Settings

New Scan

Scan Name:

Scan Target:

Start Scan

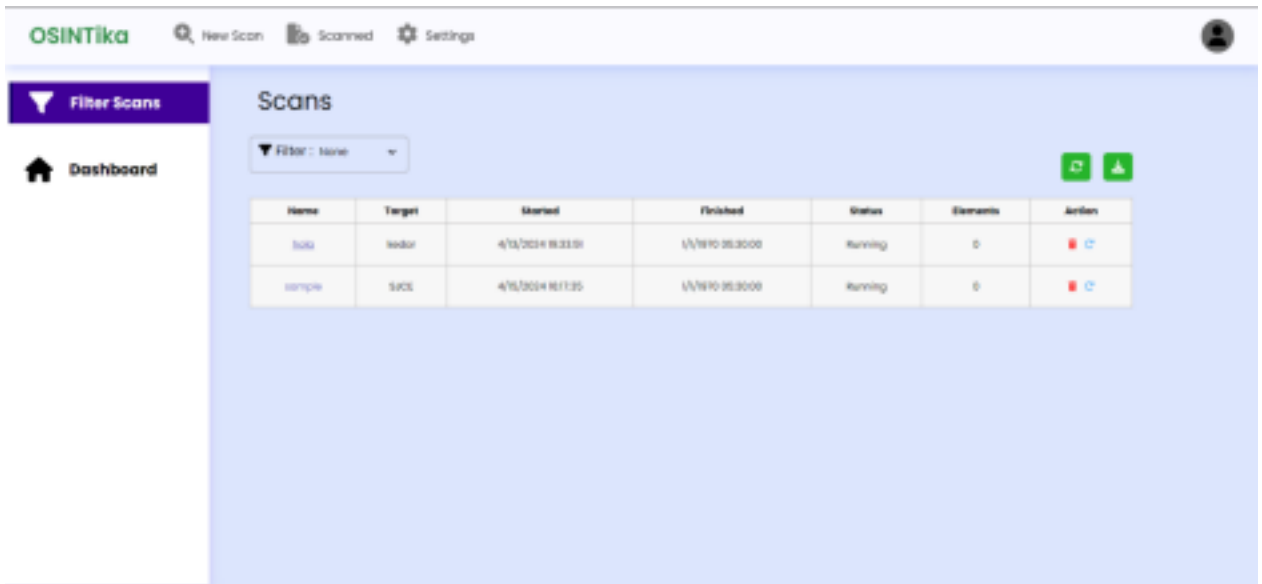
Select Required Data

- ☐ IP Address: Unique IP address on the Internet.
- ☐ Physical Address: Location/Address identifying the last usage on the Internet.
- ☐ Telecommunications: Information related to communication systems.
- ☐ Hosting Provider: Details about the company hosting a website.

Examples of Scan Targets:

- Domain Name: e.g. example.com
- IPv4 Address: e.g. 123.4
- IPv6 Address: e.g. 2001:db8::1
- Hostname/Sub-domain: e.g. abc.example.com
- Subnet: e.g. 123.0/24
- Bitcoin Address: e.g. 1Pz9VXGqcyDpC8v63bnghu6Gc78
- E-Mail address: e.g. bob@example.com
- Phone Number: e.g. +0234502990 (2344 format)
- Human Name: e.g. "John Smith" (must be in quotes)
- Username: e.g. "johnd0000" (must be in quotes)
- Network ASN: e.g. 1234

Fig A2.2



OSINTika New Scan Scanned Settings

Scans

Filter: None

Name	Target	Started	Finished	Status	Elements	Action
test	test	4/15/2024 10:33:51	1/1/1970 00:00:00	Running	0	
sample	SUCS	4/15/2024 10:11:35	1/1/1970 00:00:00	Running	0	

Fig A2.3

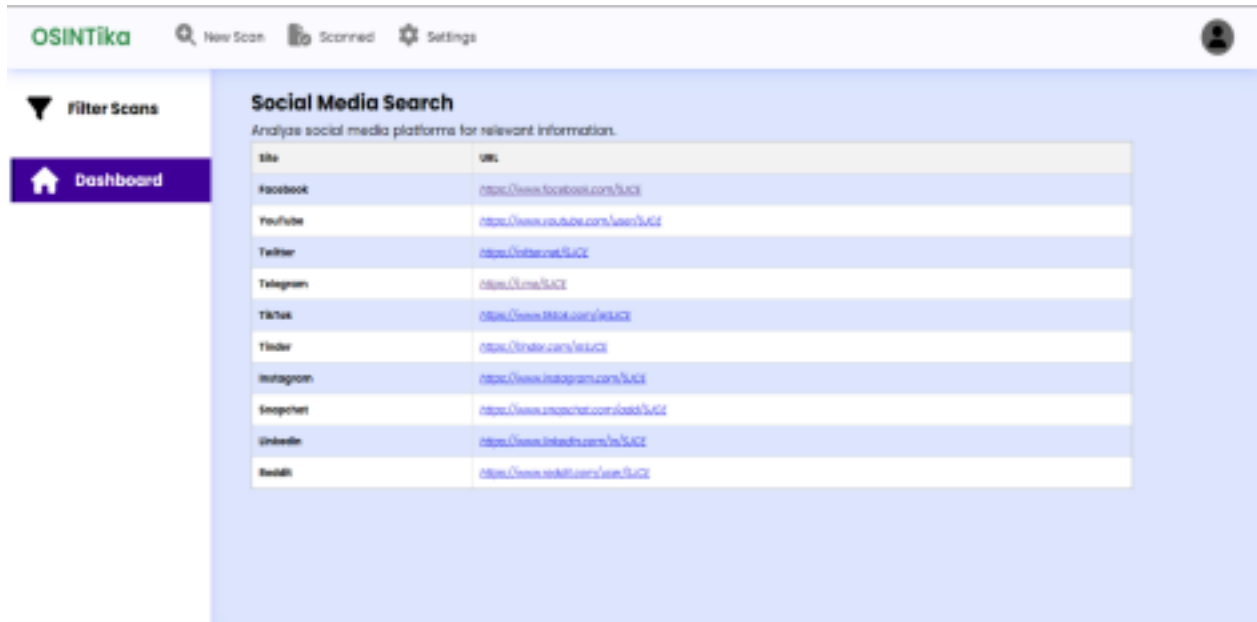


Fig A2.4

34

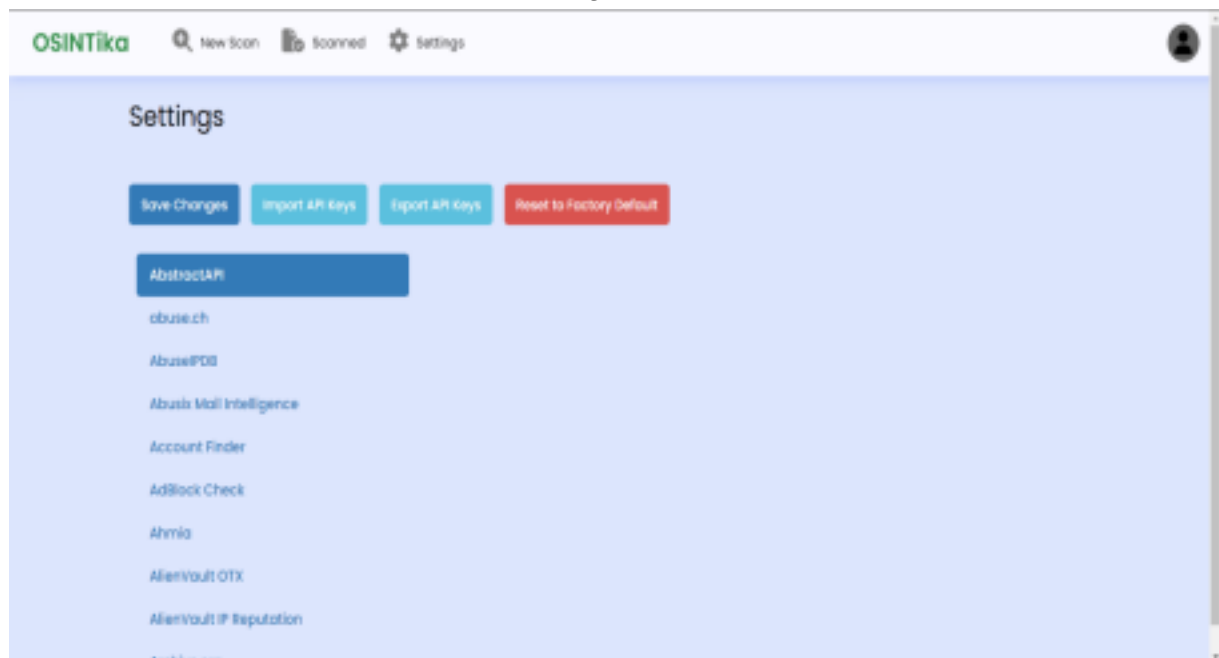


Fig A2.5

REFERENCES

- [1] "Current Challenges and Future Research Areas for Digital Forensic Investigation" - David Lillis et al. (2020)
- [2] "Toward Automated Fact-Checking: Detecting Checkworthy Factual Claims by ClaimBuster" - Naeemul Hassan et al. (2017)
- [3] "Machine Learning for Cybersecurity and Decision Support: a Survey of Trends and Techniques" - Talha Ongun et al. (2020)
- [4] "Social Media Data Analysis for Proactive Policing and Community Policing" - Chuxu Zhang et al. (2019)
- [5] "Automated Information Extraction from Multilingual Open Source Data for Proactive Event Detection" - José M. Fernández et al. (2021)
- [6] "The Limits of Artificial Intelligence in National Security" - Peter W. Singer and Gregory C. Allen (2021)

- [7] "Automating Open Source Intelligence: Algorithms, Approaches, and Applications" - Robert Layton, Paul A. Watters (2021)
- [8] "Challenges in Data-Driven Approaches to OSINT: The Case of Terrorist Content Online" - A. Edwards et al. (2022)
- [9] "OSINT in the Age of Information Overload: Challenges and Opportunities for Machine Learning" - Hanna Linder et al. (2023)
- [10] "Ethical Dilemmas in Automated OSINT Systems: A Social Science Perspective" - Julia Johnson (2024)