INTERNSHIP PROJECT

Optical Character Recognition and Text Analysis for Extraction of Sensitive Information

Submitted by: Buğra Çayır

Session: August 22, 2023

Subject of Report: Cyber Threat Intelligence

Word Count: 3651

Reference Style: APA 7th edition

Table of Contents

Abstract 3

1) Research Topic 4

2) Introduction 4

3) Background Information 5

3.1) Optical Character Recognition 5

3.1.1) OCR Software and Libraries 6

3.2) Selection of Technologies and Tools 9

3.3) Development Environment Preparation Procedure 11

3.3.1) GitHub 11

3.3.2) Poetry 11

3.3.3) Redis 12

3.3.4) FastAPI 12

3.3.5) Docker & Docker Compose 12

3.3.6) Other Libraries 13

4) Architecture 14

5) Methodology 15

6) Evaluation 17

6.1) Conclusion 17

6.1.1) Example Run & Result 18

6.2) Strengths and Weaknesses 20

7) Extension of the Investigation 21

References 23

Abstract

This essay presents a comprehensive exploration of the topic of Optical Character Recognition (OCR) and text analysis. The project aims to develop an Application Programming Interface (API) to extract textual content from images and identify sensitive information within the text. The significance of cybersecurity in today's digital landscape is highlighted, emphasizing the need to preemptively protect against potential cyberattacks. The project's purpose is to contribute a proactive defense mechanism by automatically analyzing images for sensitive data, aiding individuals and organizations in preventing inadvertent data exposure. The essay outlines the background of OCR technology and its evolution, describing its principles and applications across diverse industries. The selection of technologies, tools, and libraries such as pytesseract, FastAPI, Redis, and Docker, along with their respective justifications, is detailed. The architecture of the project is presented, demonstrating adherence to clean code principles and Pythonic idioms. The development methodology is elucidated, from project setup to the incorporation of sensitive information validation mechanisms. The evaluation of the project's success in achieving its goals is discussed, highlighting its strengths, limitations, and potential for expansion. Future directions for extending the project's capabilities, such as incorporating machine learning techniques and developing user interfaces, are also explored. Overall, this project contributes to the advancement of automated sensitive information detection and establishes a foundation for proactive data security measures.

1) Research Topic

This project is focused on the topic of Optical Character Recognition (OCR) and text analysis. Through the utilization of Application Programming Interface (API), the aim is to extract textual content from images provided by the user, along with the identification and extraction of sensitive information contained within the text.

2) Introduction

Cyber security is one of the most crucial fields of the entire information technologies and its value increases dramatically day by day. Malicious actors are always trying to delve into the every little detail of people and companies. Even a single sensitive information found may lead to a complex cyberattack which can cause irreparable damage (*Why Is Cybersecurity Important? |University of Nevada, Reno*, 2022). To prevent such cases and protect the reputation and revenue, cyber security intelligence must be one step further than the threat actors.

The aim of this project acts as one piece of the puzzle of a barrier or a precaution for people or companies to take appropriate actions before any cyberattacks occur. In daily life, there are lots of images shared with the public in different forms such as a post in social media or a commit to GitHub repository. Unfortunately, some of these images might contain any sensitive information and be mistakenly shared with everyone else. For an automated scanning process, this project can analyze and verify the sensitive data in the textual content of an image submitted by a user via an API, and the response is returned as JSON format. Furthermore, a cache mechanism is implemented to expedite responses when a previously submitted image is resubmitted to the system. This project can be one step of a more complex automated service that scans every image in various websites and detects if there is any sensitive information.

3) Background Information

3.1) Optical Character Recognition

Optical Character Recognition (OCR) stands as a transformative technology that bridges the gap between physical and digital realms by enabling the conversion of printed or handwritten text within images into editable and machine-readable formats. The core principle behind OCR lies in its ability to recognize and interpret characters, symbols, and patterns present in images, ultimately transforming them into text that can be processed, analyzed, and manipulated by computer systems. OCR technology has witnessed significant advancements over the years, driven by the increasing demand for efficient data digitization, text extraction, and document management. Initially developed as a tool to automate the digitization of printed texts, OCR has evolved to encompass various languages, fonts, and styles, enabling its application across diverse domains.

The process of OCR involves several key steps, including image acquisition, preprocessing, text recognition, pattern matching, feature extraction, and post processing (*What Is OCR? - Optical Character Recognition Explained - AWS*, n.d.). Image preprocessing techniques, such as noise reduction, contrast enhancement, and skew correction, are applied to ensure the optimal quality of the input image. Subsequently, characters within the image are identified and separated through segmentation, allowing for individual character recognition. Feature extraction involves identifying distinguishing attributes of characters, which are then utilized to match characters with corresponding textual representations. The final step, text recognition, entails mapping the recognized characters to their corresponding Unicode or ASCII values, resulting in the conversion of the image's textual content into a machine-readable format (Education, 2022).

OCR has found wide-ranging applications across various industries, including finance, healthcare, education, legal, and more. In finance, OCR is used for automating data entry from invoices, receipts, and financial statements. In healthcare, it aids in digitizing patient records, prescriptions, and medical reports. The education sector benefits from OCR by facilitating the conversion of printed textbooks and handwritten notes into accessible digital formats. Additionally, OCR plays a pivotal role in aiding the visually impaired by converting printed text into speech or Braille (Dalip et al., 2022). The integration of OCR with advanced technologies, such as machine learning and artificial intelligence, has further elevated its accuracy and versatility. As a result, OCR has become an indispensable tool for extracting valuable information from images and enabling efficient data-driven decision-making processes. In the context of this research project, OCR serves as the foundation upon which the extraction and analysis of textual content from images are built, contributing to the project's overarching goals of enhancing information retrieval and analysis capabilities.

3.1.1) OCR Software and Libraries

If any user or company wants to use the advantages and benefit from the opportunities of the OCR technology, there are numerous tools for various purposes. For OCR software, some examples can be listed as Nanonets, ABBYY Flexicapture, Kofax Omnipage, IBM Datacap, AWS Textract, Klippa, and Docparser (S, 2023). Instead of using a specific software for this aim, there are some built-in properties of some common applications such as Microsoft OneNote, LightPDF, SingleView, and OCR with Google Docs (Sieber, 2023).

As a developer, in this project, it was not possible to use such a service or software since the image scanning process should be automated. For this reason, a python library named

“pytesseract” is used (*Pytesseract*, 2022). Python-tesseract serves as an interface for Google's Tesseract-OCR Engine. It can also function independently as a command-line script for invoking Tesseract. It possesses the ability to process various image formats, including jpeg, png, gif, bmp, tiff, and more which are supported by the Pillow and Leptonica imaging libraries. For complex operations, it can be also used with other libraries like OpenCV. For developers, there are other open source OCR libraries such as EasyOCR, Kraken, and Textract (Mousa, 2022). In this project pytesseract is used due to its usage frequency, community, support of various image input formats of jpeg, jpg, png, gif, bmp, tiff etc. and output formats of PDF, TEXT files, TSV, and read-only text.

While using only pytesseract, some problems occurred while reading the image. For instance, in the image it was written “5394219505” however, it was taken as “9394219505”. There were more occurrences of such situations and as a solution, another OCR library is used in the project. At first, the aim was to use as many libraries as possible but most of them were not suitable.

While downloading the Kraken library, scikit-image is needed but cannot be downloaded. To investigate the issue, the scikit-image library is downloaded itself with the version of 0.21.0. Then, the following error is given:

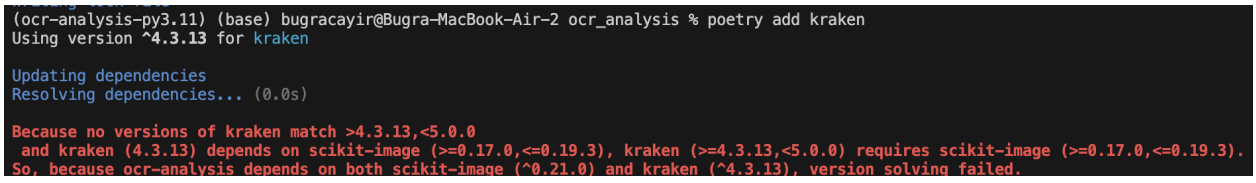


Figure 1. Kraken installation error

To solve this issue, scikit-image is tried to downgraded to the requested version and another problem occurred:

```
(ocr-analysis-py3.11) (base) bugracayir@Bugra-MacBook-Air-2 ocr_analysis % poetry add scikit-image==0.19

Updating dependencies
Resolving dependencies... (0.5s)

Package operations: 0 installs, 1 update, 0 removals

    • Updating scikit-image (0.21.0 -> 0.19.0): Failed

  ChefBuildError
```

Figure 2. Scikit-image version 0.19 installation error

When the new error is investigated, it is seen that:

```
Note: This error originates from the build backend, and is likely not a problem with poetry but with scikit-image (0.19.0) not supporting PEP 517 builds
. You can verify this by running 'pip wheel --use-pep517 "scikit-image (==0.19.0)"'.
```

Figure 3. Scikit-image version 0.19 installation error reason

In the end, it was not possible to use Kraken for this project, therefore EasyOCR was tried. Installation process was done successfully but while using this library, the image process took so long. Here is an example run:

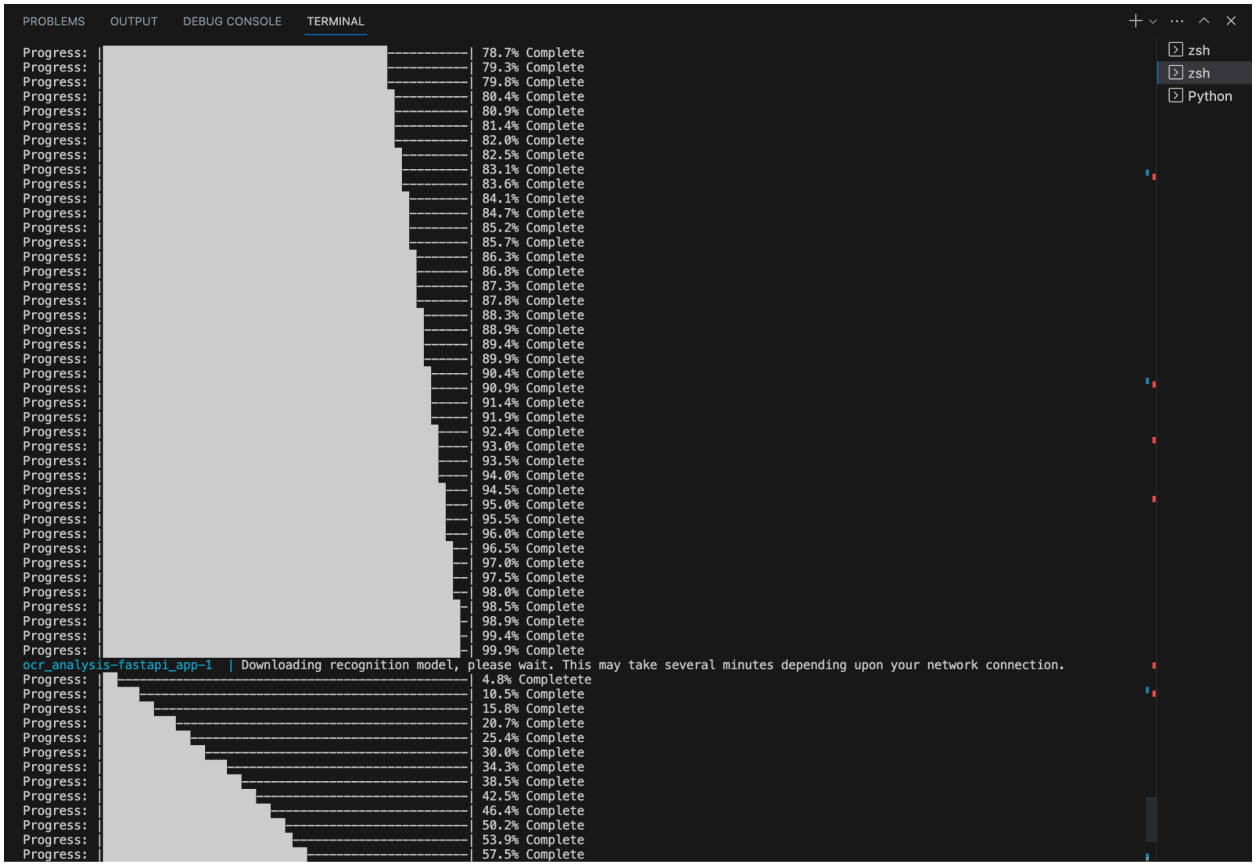


Figure 4. EasyOCR example run

Since this process took approximately 2 minutes, it was not efficient. Then, as the third option to be the second OCR library, the textract library is used. It was easy to install and implement with no errors.

3.2) Selection of Technologies and Tools

For the development IDE, Visual Studio Code is used due its simplicity, easy-to-use design, and lightweight. The project is published and shared with other developers via GitHub, and GitKraken is used as an alternative to Git command line interface.

As a tool for dependency management and packaging, Poetry library is chosen. Although there are alternatives to Poetry such as pyenv or virtualenv; Poetry makes it easier to create, manage and maintain a simple, isolated, and efficient codebase, as a newer dependency management tool, with more modern functionalities than the older pip and pipenv tools (*What Is Python Poetry? Why Use It? (Dependency Management)*, n.d.). Firstly, Poetry offers an inherent virtual environment that is more streamlined to configure. In contrast, both pip and pipenv relied on the 'venv' module, which could pose a slight challenge during setup. Secondly, Pipenv utilizes a Pipfile to oversee dependencies, as opposed to Poetry's usage of the pyproject.toml file. The Pipfile shares similarities with the requirements.txt file used by pip, but incorporates additional functionalities, such as support for development dependencies and environment-specific packages (*What Is Python Poetry? Why Use It? (Dependency Management)*, n.d.). While these features are advantageous, they also introduce complexity and lead to an increased number of files to manage over time. Thirdly, Pipenv and poetry diverge in their approach to Python environments. Pipenv relies on the system-provided Python environment, whereas Poetry automatically generates and administers a project-specific virtual environment. This automated functionality of Poetry aids in isolating packages and mitigating potential version conflicts within Python files and projects.

For the web framework and API usage, FastAPI is chosen. The alternatives to FastAPI would be Django REST Framework or Flask but as it is claimed in the FastAPI official documentation, there are various advantages of FastAPI over these options since Django framework inspired FastAPI to have an automatic API documentation web user interface, and Flask inspired FastAPI to be a micro-framework making it easy to mix and match the tools and parts needed (*Alternatives, Inspiration and Comparisons - FastAPI*, n.d.).

For the caching mechanism, Redis is used. There is an option of Memcache, but Redis offers similar speed with much more functionality.

Docker is implemented as the containerization service because it is really common in the information technology industry with a great support from its community. It is an open source platform and lets any code run on any computing environment. Additionally, Docker Compose made it possible to run different services or servers to run at the same time with ready to use Docker implementations. For alternatives, there is LXC but it is an OS-based container that allows the app to run on multiple Linux systems, and CRI-O is a container runtime interface for Kubernetes, a container cluster management platform; so these two options were not suitable for this project (Dhaduk, 2022).

3.3) Development Environment Preparation Procedure

In this project, the Python version of 3.11 is used.

3.3.1) GitHub

The project cannot be reached from the GitHub repository due to the security restrictions.

3.3.2) Poetry

After the Poetry is installed to the local, in the main project directory, the following functions are called one by one and poetry.lock and pyproject.toml files are added:

```
poetry init
poetry install
```

Afterward, use the following command to enter the Poetry shell to write any command:

```
poetry shell
```

From now on, everything is ready to use with Poetry and see official documentation for more information.

3.3.3) Redis

Redis is directly used from Docker. The implementation on docker-compose.yml file is:

```
redis:
  image: redis:latest
  ports:
    - "6379:6379"
```

3.3.4) FastAPI

FastAPI is added via Poetry, and uvicorn is used as the ASGI server.

3.3.5) Docker & Docker Compose

Dockerfile for the project itself is written and it is prepared by being compatible with Poetry which means requirements.txt file is not used. Docker Compose is used to run both the Redis and FastAPI server together. Inspect the Dockerfile and docker-compose.yml files in the GitHub repository.

3.3.6) Other Libraries

Table 1.
Libraries used in the project

Library Name	Definition & Reason to Use
Dynaconf	Dynaconf is a library designed to excel as the preferred solution for handling configuration in Python. It possesses the capability to retrieve settings from diverse origins, encompassing environment variables, files, configuration servers, vaults, and more. In this project, it is used with dotenv configuration (Rocha, n.d.).
Jinja2	Jinja is a web template engine utilized in the Python programming language. It bears resemblance to the Django template engine, yet it furnishes Python-like expressions while guaranteeing that template evaluations occur within a sandbox (*Jinja — Jinja Documentation (3.1.x)*, n.d.).
Black	Black makes the code formatting process in a faster way and developers can save time and mental energy for other topics. It is able to reformat the blank spaces, blank lines in an appropriate way so that everyone can easily read and review the code (*Black 23.7.0 Documentation*, n.d.).
Bandit	Bandit is a utility crafted to detect prevalent security concerns within Python code. To accomplish this, Bandit examines individual files, constructs an Abstract Syntax Tree (AST) from each file, and employs relevant plugins to analyze the AST nodes. Upon completing scan across all files, Bandit generates a comprehensive report (*Welcome to Bandit — Bandit Documentation*, n.d.).
Isort	Isort stands as a Python tool and library for arranging imports in alphabetical order, while also automatically organizing them into distinct sections based on their type. This solution offers a command-line interface, a Python library, and add-ons for different editors, enabling efficient sorting of all import statements (*Isort*, n.d.).
Python-multipart	python-multipart is an Apache2 licensed streaming multipart parser for Python (*Python-Multipart — Python-multipart 0.0.1 Documentation*, n.d.).
Pillow	The Python Imaging Library serves as a cost-free and open-source supplementary toolkit for the Python programming language. It introduces the capability to access, manipulate, and store a diverse range of image file formats (*Pillow*, n.d.).

Validators	This library is used to validate sensitive urls and domains. There are other validator library options in Python but they require defining a schema or form but this library does not (*Validators — Validators 0.11.2 Documentation*, n.d.).
Python-whois	This library has a function that sends a whois request to the established domain (*Python-whois*, 2022).
Requests	Requests stands as an HTTP client library intended for the Python programming language. Renowned for its refined alignment of the HTTP protocol with Python's object-oriented nature, Requests is among the most widely used Python libraries. It holds distinction as it's not native to Python's standard library (*Requests: HTTP for Humans™ — Requests 2.31.0 Documentation*, n.d.).

4) Architecture

The code structure and file hierarchy in this project are organized according to the clean code principles since it is easy to understand, more efficient, easier to maintain, scale, debug, and refactor. Throughout the project, Python Enhancement Proposal (PEP8) is regarded (Tomazic, 2021). For the naming convention as meaningful and intention-revealing names; variable names, function names, and constants are all written in snake_case and all lowercase. Ambiguous abbreviations are avoided. Line formatting such as avoiding multiple statements on the same line and imports being on separate lines, isort library is used. Moreover, the set of idioms adapted by the Python community is called Pythonic code which includes variable tricks, list manipulation, dealing with functions, and explicit code. The code is written according to these principles, and for instance, list comprehensions are used. Since Python is an OOP language and in this project there are interfaces and encapsulation, SOLID principles are also applied. In the below image, the architecture can be seen:

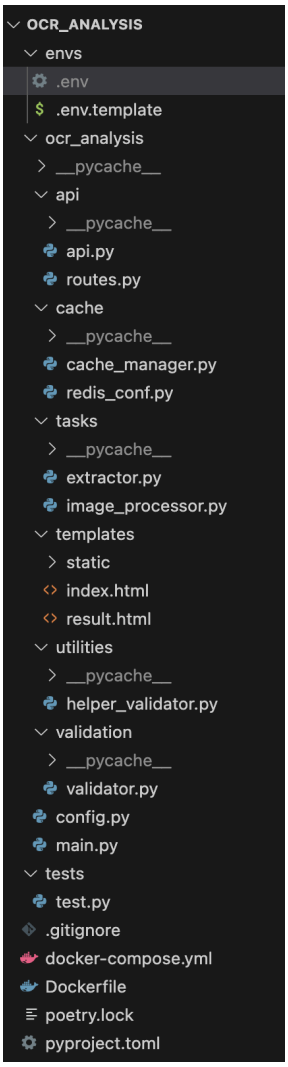


Figure 5. Project architecture

5) Methodology

The development process is completed step by step.

1. Project architecture created.
2. Poetry added.
3. FastAPI added with an example run code to see if it is working properly.

4. Dockerfile is created. Poetry and pytesseract are installed via Dockerfile due to an error of pytesseract as “Image To String: Tesseract.exe Is Not Installed Or It's Not In Your Path”. Additionally, libgll-mesa-glx is also installed via Dockerfile due to the error of “Import error: libGL.so.1: cannot open shared object file: No such file or directory.”
5. Docker-compose.yml file is configured to run both the FastAPI server and Redis.
6. After the image processing and text extraction functions are completed, sensitive information extraction is handled with regex and an interface is used for this purpose.
7. For the sensitive information verification process, it is possible to verify URLs, domains, credit cards, plates, hashes of md1 sha1 sha256, Turkish IDs, IP addresses, btc wallets, IBANs and email addresses. URLs and domains are verified via the validators library. Credit cards are verified according to the Luhn algorithm, and also the Bin List of Amex, Visa, Mastercard, Diners Club, Discover, and JCB are used to detect the company of the credit card. For email address verification, hunterio and Verifalia services are used via API requests and this is the reason why it is needed for dotenv configuration with Dynaconf library.
8. HTTP204 and HTTP400 cases are handled.
9. For Captcha cases, both pytesseract and textract were unable to directly read the image and HTTP204 was returned. To fix this issue, the following solutions were implemented:
    a. https://stackoverflow.com/questions/60028463/read-text-from-an-image
    b. https://stackoverflow.com/questions/74642350/simple-captcha-solver-with-python
10. Caching implementation is completed with Redis.
11. Code is reviewed and made better and concise with libraries such as Black and isort.
12. Ngrok is used to expose the local development server to the Internet with minimal effort.

6) Evaluation

6.1) Conclusion

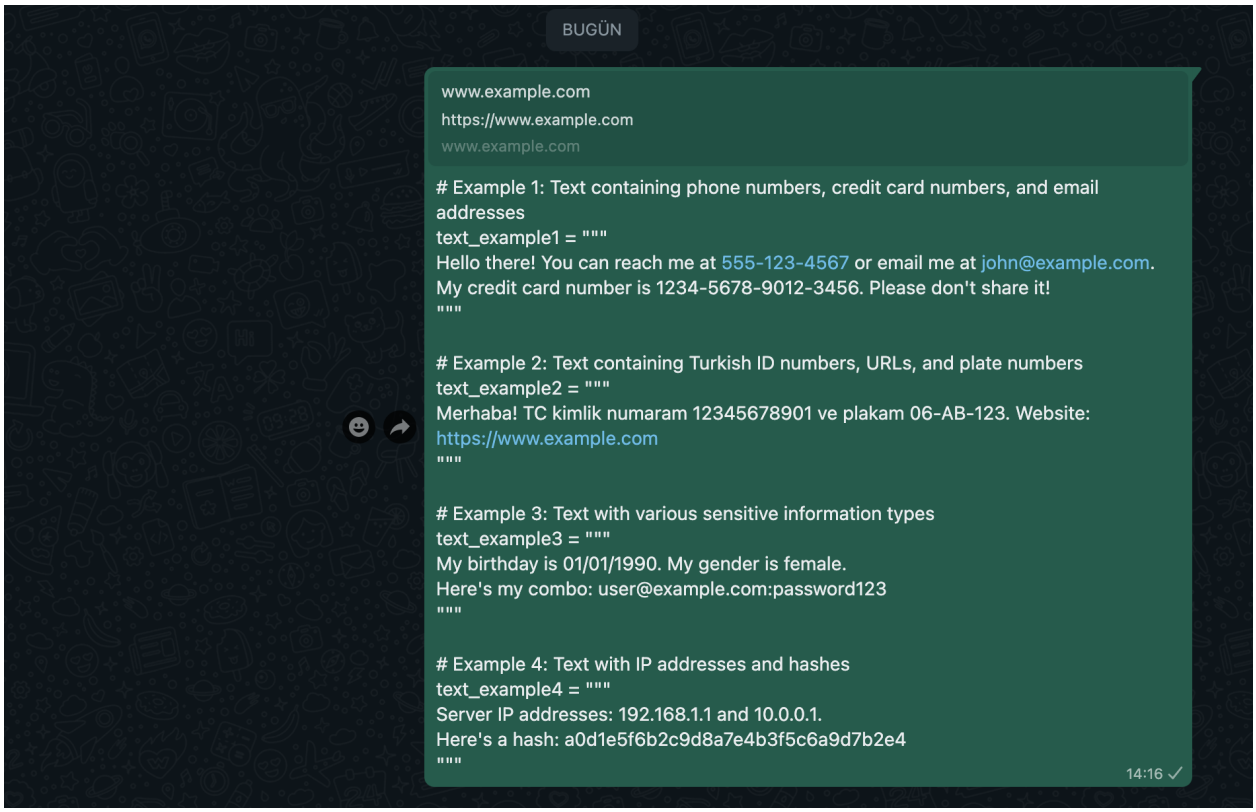
In conclusion, this internship project has successfully addressed the challenge of automating the extraction and analysis of sensitive information from images using Optical Character Recognition (OCR) and text analysis techniques. The project's main objectives were to create an API that receives images, processes them to extract text, identifies sensitive information within the text, and responds with the findings in JSON format. This goal was achieved through the careful selection and integration of various technologies and libraries.

Throughout the development process, the project leveraged key technologies such as FastAPI for creating the API endpoints, pytesseract for OCR capabilities, and Redis for caching to enhance response times. Additionally, the use of external services like hunterio and Verifalia facilitated email verification and validation of sensitive information. By combining these tools, the project established a streamlined workflow for processing images, extracting textual content, and validating sensitive information.

The project also addressed challenges related to efficiency, accuracy, and security. The implementation of caching using Redis significantly improved response times for previously processed images, enhancing user experience. The incorporation of validation mechanisms for sensitive information, such as credit card numbers and email addresses, helped ensure accurate identification and verification.

6.1.1) Example Run & Result

Note that only valid sensitive information values are displayed. The content of the image is taken from both pytesseract and textract but since in many cases pytesseract was better, its content is shown. The below image contains some example cases:

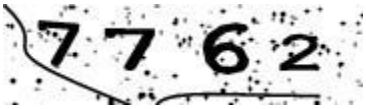


The above image is uploaded to the service and here is the response:


```
{
  "content": "BUGUN\n\nwww.example.com\n\nhttps://www.example.com\n\n# Example 1: Text containing phone numbers, credit card numbers, and email\naddresses\n\ntext_example1 = \"\"\"\n\nHello there! You can reach me at 555-123-4567 or email me at john@example.com.\nMy credit card number is 1234-5678-9012-3456. Please don't share it!\n\n# Example 2: Text containing Turkish ID numbers, URLs, and plate numbers\ntext_example2 = \"\"\"\n\nMerhaba! TC kimlik numaram 12345678901 ve plakam 06-AB-123. Website:\nhttps://www.example.com\n\n# Example 3: Text with various sensitive information types\ntext_example3 = \"\"\"\n\nMy birthday is 01/01/1990. My gender is female.\n\nHere's my combo: user@example.com:password123\n\n# Example 4: Text with IP addresses and hashes\ntext_example4 = \"\"\"\n\nServer IP addresses: 192.168.1.1 and 10.0.0.1.\n\nHere's a hash: a0d1e5f6b2c9d8a7e4b3f5c6a9d7b2e4\n\n14:16 Y\n",
  "status": "successful",
  "findings": [
    {
      "value": "555-123-4567",
      "type": "PHONE_NUMBER"
    },
    {
      "value": "1990-01-01 00:00:00",
      "type": "DATE"
    },
    {
      "value": "user@example.com:password123",
      "type": "COMBOLIST"
    },
    {
      "value": "female",
      "type": "GENDER"
    },
    {
      "value": "https://www.example.com",
      "type": "URL"
    },
    {
      "value": "www.example.com",
      "type": "DOMAIN"
    },
    {
      "value": "example.com",
      "type": "DOMAIN"
    },
    {
      "value": "john@example.com",
      "type": "EMAIL"
    },
    {
      "value": "user@example.com",
      "type": "EMAIL"
    },
    {
      "value": "a0d1e5f6b2c9d8a7e4b3f5c6a9d7b2e4",
      "type": "HASH"
    },
    {
```

```
    "value": "06-AB-123",
    "type": "PLATE"
  },
  {
    "value": "10.0.0.1",
    "type": "IP_ADDRESS"
  },
  {
    "value": "192.168.1.1",
    "type": "IP_ADDRESS"
  }
]
}
```


Furthermore, the below Captcha cases are also handled:

6.2) Strengths and Weaknesses

Table 2
Evaluation of method & limitations

Evaluation	Limitation	Realistic and Achievable Suggestion
Number of sensitive information fields	In this project; phone number, ID number, credit card, plate, date, email, domain, url, hash, combolist, gender, BTC wallets, IBANs, and ip address fields are extracted. Since regex is used, more fields would cause false positive results.	There can be more fields of sensitive information. In fact, this project can be extended into divisions where country-specific versions extract data since every country uses various phone number starters, plates etc.
Number of people working on the project	Since the project is developed by one person, it is hard to implement every best practice.	Other developers and people with no background about cyber security would look at the development process to see other perspectives and achieve excellent results.

Credit card verification bin list	For the bin lists, only Amex, Visa, Mastercard, Diners Club, Discover, and JCB are used.	In every country there are also other credit card suppliers and more bin lists can be added
Email address verification services	Only hunterio and Verifalia are used as email verification services.	The number of services used can be increased.

7) Extension of the Investigation

In the future, the project could be extended in various directions to further enhance its capabilities and applicability. One avenue for expansion is the incorporation of machine learning techniques to improve the accuracy of sensitive information extraction and validation. Training a model to identify additional types of sensitive information and implementing custom validation methods could broaden the project's scope. Furthermore, the project could benefit from enhanced support for multiple languages and scripts, allowing it to process and analyze text in various languages. This could involve integrating language-specific OCR models and leveraging multilingual validation mechanisms. Moreover, the best-use of this project would be combining the idea of the project with another service that automatically detects and finds images from social media accounts of employees, GitHub, or dark web; and then sending these images to this project to extract the sensitive information.

Another valuable extension could involve the development of a user interface for uploading images, interacting with the API, and viewing the results. This would enhance user experience and make the project more accessible to non-technical users.

In conclusion, this internship project serves as a foundational step towards automating the extraction and analysis of sensitive information from images. By leveraging Optical Character Recognition and text analysis techniques, the project has demonstrated the potential to enhance data security and facilitate efficient data processing. With further development and expansion,

the project could evolve into a powerful tool for safeguarding sensitive information in various domains.

References

Alternatives, inspiration and comparisons - FastAPI. (n.d.). https://fastapi.tiangolo.com/alternatives/

Black 23.7.0 documentation. (n.d.). https://black.readthedocs.io/en/stable/

Dalip, F., Yildiz, K., Ulku, E. E., & Büyüktanir, B. (2022). Raspbraille: Conversion to Braille Alphabet with Optical Character Recognition and Voice Recognition Algorithm. *Hitite Journal of Science and Engineering*, *9*(4), 253–261. https://doi.org/10.17350/hjse19030000278

Dhaduk, H. (2022, October 18). Containerization Technology: types, advantages, applications, and more. *Simform - Product Engineering Company*. https://www.simform.com/blog/containerization-technology/

Education, I. C. (2022, January 5). *What is optical character recognition (OCR)? - IBM blog*. IBM Blog. https://www.ibm.com/blog/optical-character-recognition/

isort. (n.d.). https://pycqa.github.io/isort/

Jinja — Jinja Documentation (3.1.x). (n.d.). https://jinja.palletsprojects.com/en/3.1.x/

Mousa, H. (2022). 11 OCR libraries and projects. *MEDevel.com*. https://medevel.com/os-ocr-libraris-and-frameworks/

Pillow. (n.d.). Pillow (PIL Fork). https://pillow.readthedocs.io/en/stable/

pytesseract. (2022, August 16). PyPI. https://pypi.org/project/pytesseract/

Python-Multipart — python-multipart 0.0.1 documentation. (n.d.). https://andrew-d.github.io/python-multipart/

python-whois. (2022, July 3). PyPI. https://pypi.org/project/python-whois/

Requests: HTTP for Humans™ — Requests 2.31.0 documentation. (n.d.). https://requests.readthedocs.io/

Rocha, B. (n.d.). *DynaConf - 3.2.1*. https://www.dynaconf.com/

S, P. (2023). 10 Best OCR Software of 2023 (Free & Paid Tools). *Nanonets AI & Machine Learning Blog*. https://nanonets.com/blog/ocr-software-best-ocr-software/

Sieber, T. (2023). The 8 best Free OCR Software apps to Convert Images into Text. *MUO*. https://www.makeuseof.com/tag/top-5-free-ocr-software-tools-to-convert-your-images-into-text-nb/

Tomazic, N. (2021). Clean code in Python. *TestDriven.io*. https://testdriven.io/blog/clean-code-python/

validators — validators 0.11.2 documentation. (n.d.). https://validators.readthedocs.io/

Welcome to Bandit — Bandit documentation. (n.d.). https://bandit.readthedocs.io/en/latest/

What is OCR? - Optical Character Recognition Explained - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/ocr/

What is Python Poetry? Why Use it? (Dependency Management). (n.d.). FavTutor. https://favtutor.com/blogs/poetry-python

Why is cybersecurity important? |University of Nevada, Reno. (2022, July 22). University of Nevada, Reno. https://onlinedegrees.unr.edu/ms-in-cybersecurity/resources/why-is-cybersecurity-important/