Configuration Manual

MSc Research Project
Msc in Cloud Computing

Lakshmi Narasimha Naga Manikanta. Dasari
Student ID: x23301295

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Dasari. Lakshmi Narasimha Naga Manikanta……. ……
Student ID:	23301295……………………………………………………………………………………………………… …..…….
Programme:	MSc Cloud Computing…………………………………………………………… … **Year:** 2024-2025……………………… …
Module:	MSc Research Project………………………………………………………………………………………………….… ……
Lecturer:	Yasantha Samarawickrama ………………………………………………………………………………………….………
Submission Due Date:	11-08-2025………………………………………………………………………………………………….…… …
Project Title:	Boost-AGL Stack: A Scalable and Secure Ensemble Approach for Malicious URL Detection in the Cloud. ………………………………………………………………………………………….………
Word Count:	……………………………………… **Page Count:** 10……………………………….…….………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Dasari.Manikanta…………………………………………………………………………………………… ……………
Date:	11-08-2025………………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both	□

for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Lakshmi Narasimha Naga Manikanta. Dasari

Student ID: x23301295

Repository Link: https://github.com/manidasari27/thesis.git

INTRODUCTION

This project aims to develop a scalable and secure machine learning pipeline for detecting phishing websites by leveraging lexical and domain-based URL features. The core methodology involves training and evaluating multiple machine learning models, including a Stacking Ensemble approach that combines the strengths of several classifiers for improved accuracy. Once trained, the model is exposed through a Flask API, enabling real-time predictions via HTTP requests. To ensure portability and scalability, the application is containerized using Docker and deployed on the AWS Cloud, utilizing services such as EC2, ECR, and Cloud9. Furthermore, the entire deployment workflow is automated using GitHub Actions, streamlining the process of building, pushing, and deploying updated models seamlessly.

Dataset : http://cicresearch.ca/CICDataset/ISCX-URL-2016/Dataset/

About Dataset :

About URL Dataset:-

The internet has become a place for individuals url phishing . They often use web addresses, called URLs, as their main tool. To fight this security wise have mostly focused on making lists of bad URLs, like a 'blacklist'.

This blacklist works well for URLs we already know are bad. But it will not solve the whole problem. New bad URLs pop up all the time and they get a head start before we can add

them to our list. Also, even popular and trusted websites can be hacked and show harmful links. We call these 'defacement URLs'.

We're looking into a simpler way to find and sort these bad URLs based on what kind of attack they're trying to do. We believe that just by looking at the words and structure of a URL (this is called 'lexical analysis'), we can find these bad URLs quickly and effectively. We'll also study how criminals try to hide their bad URLs (this is called 'obfuscation') to understand which hiding tricks are used for different types of bad URLs. We'll mainly look at five different kinds of URLs.

. We study mainly five different types of URLs:

Benign URLs: Over 35,300 benign URLs were collected from Alexa top

websites. The domains have been passed through a Heritrix web

crawler to extract the URLs. Around half a million unique URLs are

First, we collected a lot of URLs. We cleaned this after removing any duplicate links and keeping only the main website addresses. After that, we checked these URLs using a service called VirusTotal to make sure we only kept the safe ones.

Here's where we got our different types of URLs:

- Spam URLs: We collected about 12,000 spam links from a public collection called WEBSPAM-UK2007.
- Phishing URLs: We got about 10,000 phishing links from OpenPhish, which is a place that keeps track of active fake websites designed to steal your information.
- Malware URLs: We found over 11,500 links to websites that spread harmful software (malware) from DNS-BH, a project that lists such sites.

For our project, we focused on two main types of URLs: safe ones (benign) and phishing ones.

Our Computer Setup (Hardware)

specifications of the setup we used:

- os: Windows 10 or 11
- Processor: Intel Core i3 (11th generation) running at 3.00 GHz.
- System Type: 64-bit, which means it can handle more complex tasks.
- Storage: 512 Gigabytes (GB) of hard drive space.
- Memory(RAM): 16 Gigabytes (GB), which is a good amount for running many programs smoothly.

Our Software Tools

software tools to develop our system:

a) Python (Version 3.10.12): popular and easy-to-use programming language. great for building applications quickly and for connecting different software parts together

b) Visual Studio Code (VS Code): open-source text editor from Microsoft very popular tool for writing computer code

c) Google Colab: This is a free online service from Google that lets anyone write and run Python code right in their web browser.

Software Libraries We Used

To help us build our URL detection system, we used several specialized software libraries (collections of pre-written code):

1. Pandas: Used for organizing and analyzing data, especially in tables.

2. .Numpy: Used for working with numbers and mathematical operations, especially large sets of data.
3. Matplotlib: Used for creating charts and graphs to visualize data.
4. Seaborn: Another library for making attractive and informative statistical graphics.
5. Plotly: Used for creating interactive charts and dashboards.
6. Scikit-learn: A very important library for machine learning, providing tools for building predictive models.
7. TensorFlow: A powerful library developed by Google for building and training machine learning models, especially deep learning models.
8. Flask: Used for building web applications, which could be used to make our URL detector accessible online.
9. Keras: A user-friendly library that works on top of TensorFlow, making it easier to build and experiment with neural networks (a type of deep learning model).

Implementation approach:

Implementation is split into three modules as mentioned below

Module 1: Getting Ready with Visual Studio Code

In this first step, we use a program called Visual Studio Code. Here, we bring in our lists of URLs (both good and bad ones). For each URL, we look at 19 different characteristics, or 'features,' that help us understand what kind of URL it is. We also add a 'label' to each URL, which tells us if it's good or bad (this is our 'target' information).

Once we've looked at all these features and added the labels, we save all this information into a file called final_dataframe.csv. This file is like a big spreadsheet that holds all the data we've prepared. (The original data came from the URL 2016 dataset by the Canadian Institute for Cybersecurity at UNB).

Module 2: Training Our Model with Google Colab

The next big step happens in Google Colab, which is an online tool that lets us run powerful computer programs. Here's what we do:

•Load Our Data: We take the final_dataframe.csv file we made in Module 1 and load it into Google Colab using a tool called Pandas.

•Clean the Data: We check our data for any missing information or parts we don't need. We remove unnecessary columns and make sure everything is tidy. We started with 2000 URLs, and for each, we had 19 features plus the one label (good or bad). We also convert any text-based information into numbers, because computers understand numbers better.

•See the Data (Data Visualization): We create charts and graphs to help us understand our data better. This is like drawing pictures of the numbers to see patterns.

•Prepare for Training: We split our data into two parts: 'features' (the 19 characteristics of the URLs) and 'target' (the label that says if it's good or bad). Then, we divide our data again: 90% of it is used to 'train' our computer models, and the remaining 10% is used to 'test' them. We always use more data for training so the models can learn well. For example, we used 1000 good URLs and 1000 bad (phishing) URLs for training.

•Teach the Models: We use this prepared data to teach our Machine Learning (ML) and Deep Learning (DL) models. The models 'learn' from the training data, and then we use the test data to see how well they can predict if a new URL is good or bad.

•Check Performance: After training, we evaluate how well our models performed. We use tools like the 'Confusion Matrix' and 'Classification Report' to see how accurate our predictions were and where the models might have made mistakes.

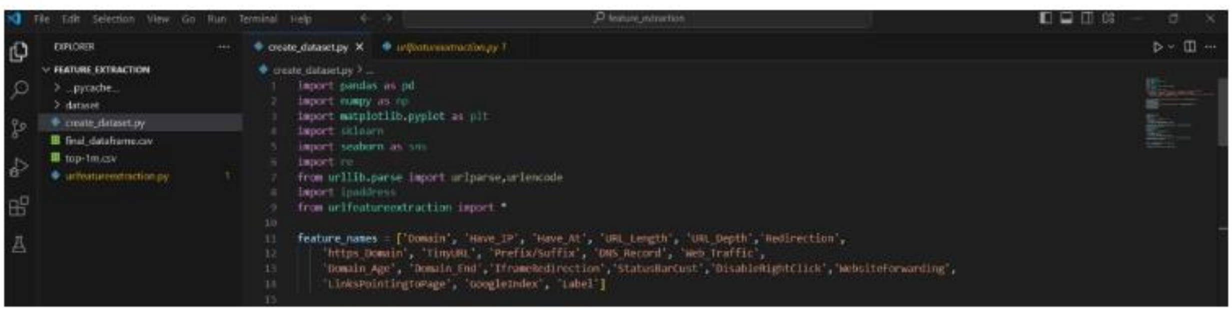
Module 3: Building a Web Tool to Detect Phishing URLs

This final part brings everything together into a user-friendly web application. We use a Python tool called Flask to build this web part, which combines the work from the previous two modules.

Here are the general steps involved:

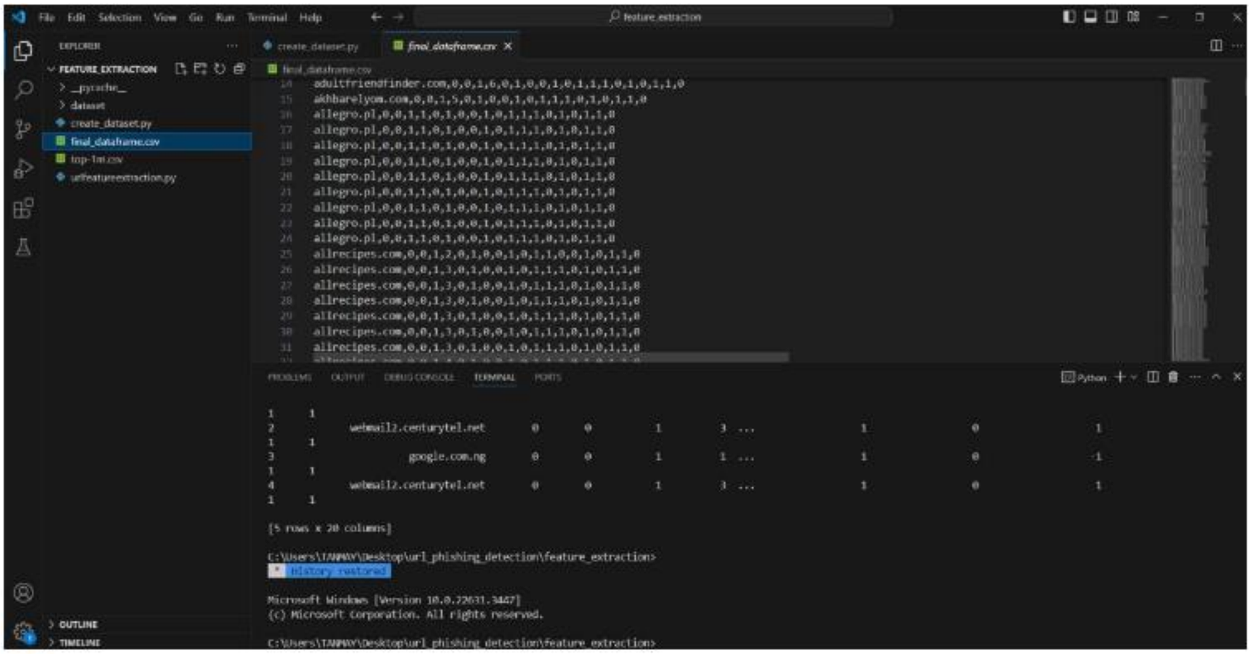
Step 1: Choose Your Data. We used a specific dataset that contains two types of URLs: good ones (benign) and bad ones (phishing). You can find this dataset at: https://www.unb.ca/cic/datasets/url-2016.html.

Step 2: Set Up Your Project. After installing Visual Studio Code, we create a new project. We then bring in all the necessary software libraries (like Pandas, Numpy, etc.) that help us extract the URL features and work with specific parts of our data.



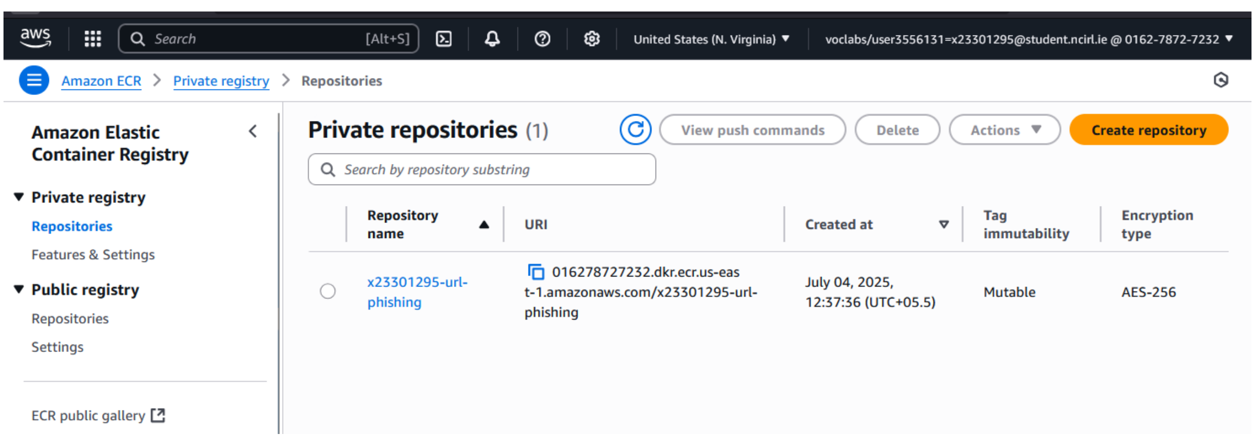
Step 3: Process the URLs. When we run a special program called create_dataset, it goes through each URL, checks its features, and then saves all the final processed data into a file, just like we mentioned for final_dataframe.csv in Module 1`.

the final data as mentioned in below image.



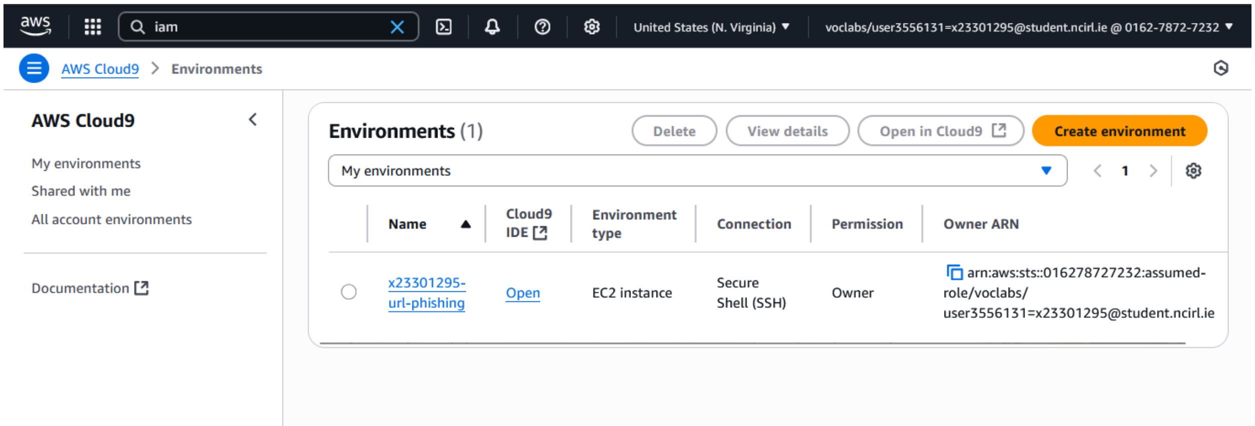
AWS Setup:

ECR Repository:

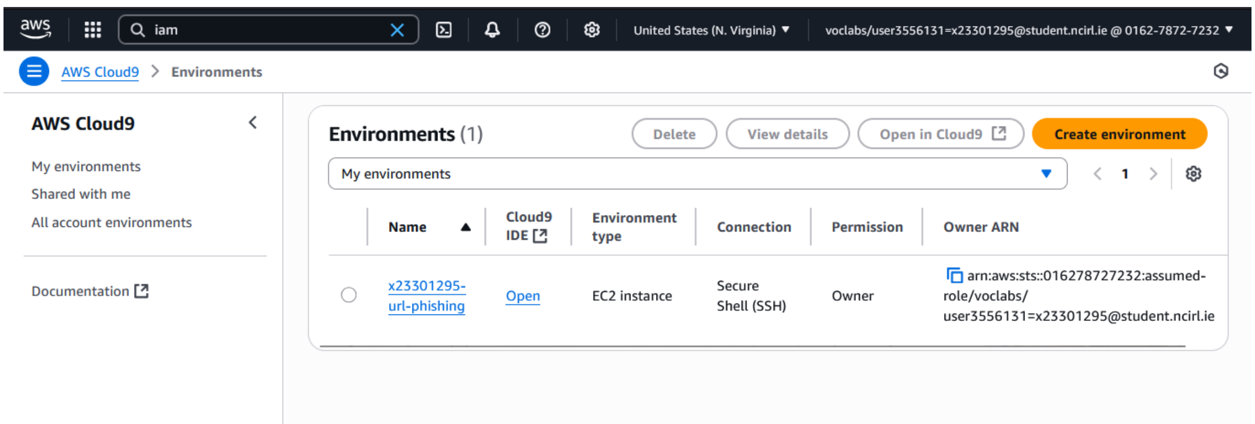


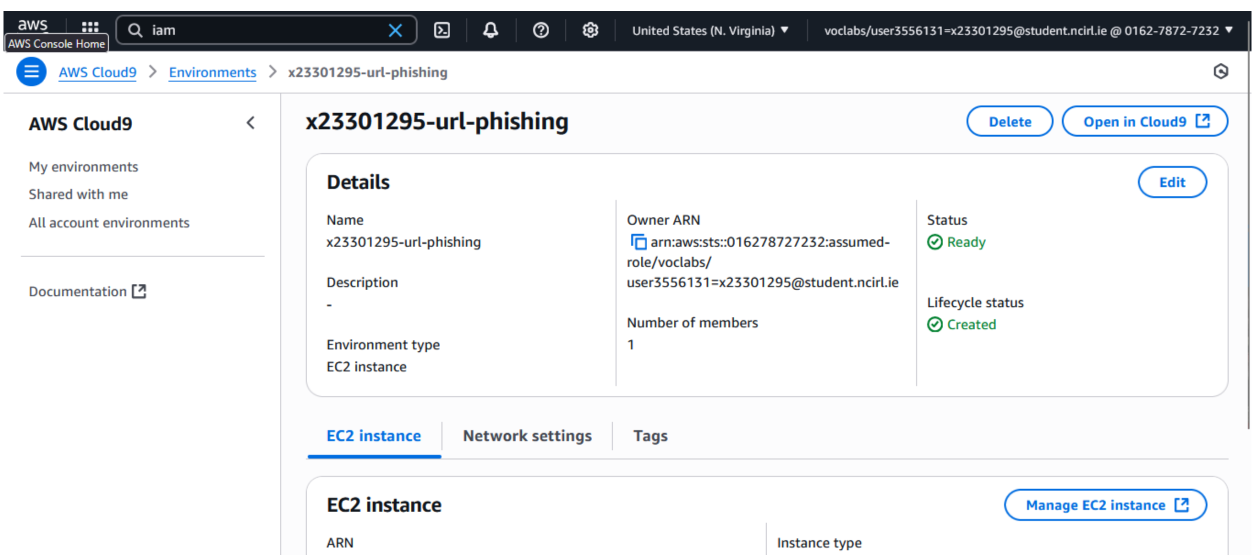
For Containerization on Web app

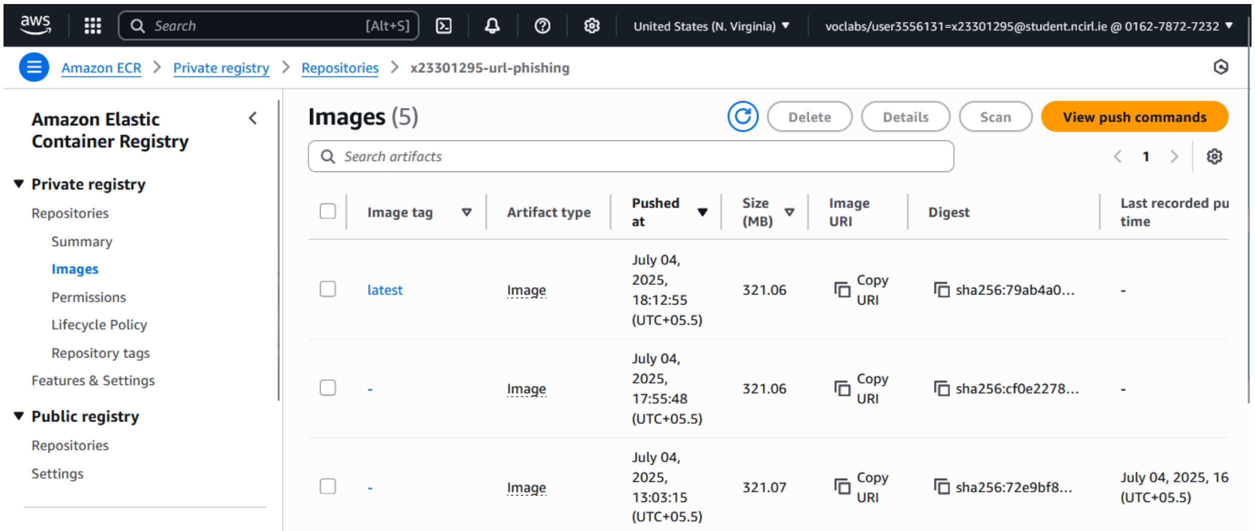
Cloud9:



For Pulling Container and running

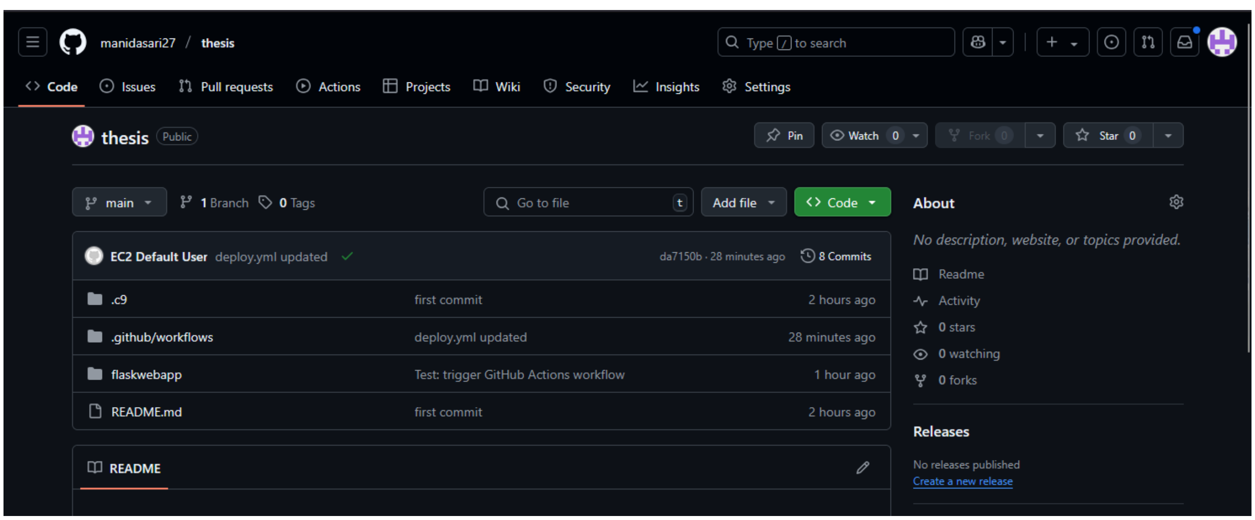


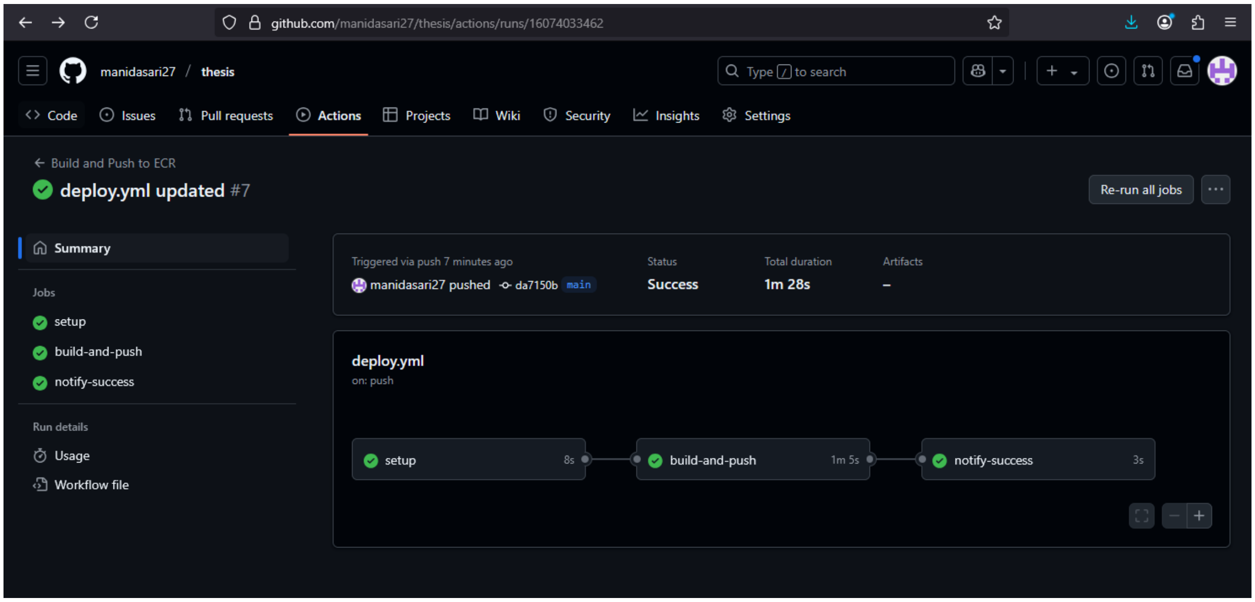




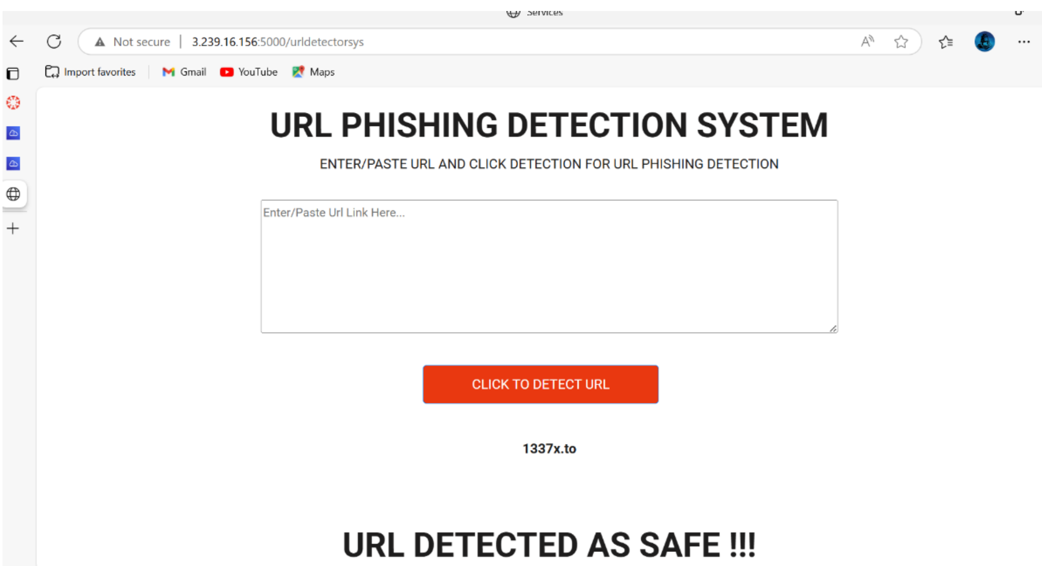
GitHub repository

For pipeline used GitHub actions it is an automated workflow that runs when specific events occur, like pushing code or opening a pull request.
It can include steps for building, testing, and deploying your application easily.

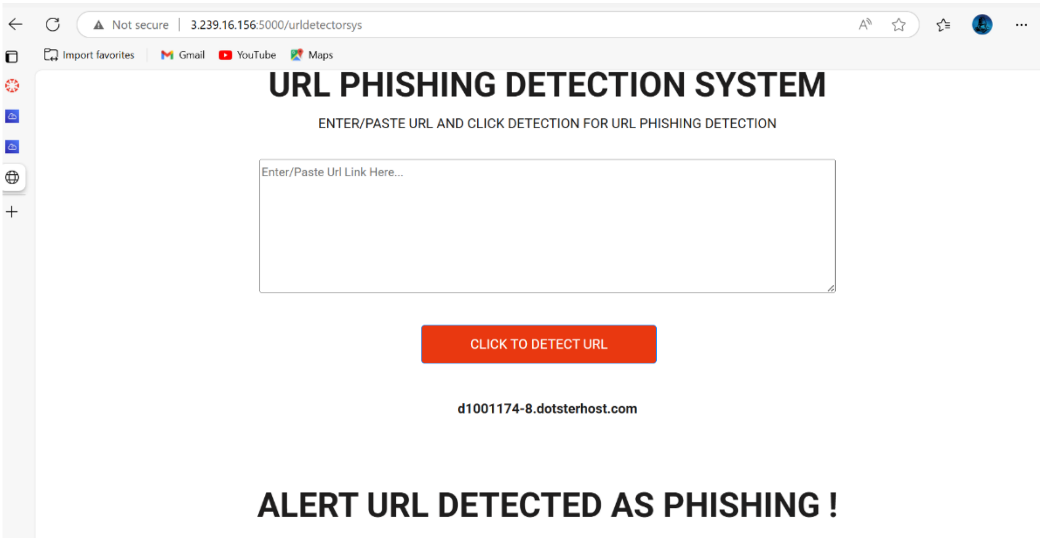




BENIGN URL: The URL has been detected Safe.



PHISHING URL: The URL has been detected as Phishing one, where these kind of URL'S should be blocked.



Importing all libraries

Importing Libraries

```python
import pickle
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
from sklearn import ensemble
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.figure_factory as ff
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.preprocessing import LabelBinarizer, LabelEncoder, MinMaxScaler
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from lightgbm import LGBMClassifier
from mlxtend.classifier import StackingClassifier
from sklearn.linear_model import LogisticRegression
```

```python
import warnings
warnings.filterwarnings('ignore')
```

Dataset loading

Data Loading

```python
dataframe = pd.read_csv('/content/drive/MyDrive/url_phishing/Data/new_final_dataframe.csv')
dataframe.head(5)
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	GoogleIndex	Label
0	1337x.to	0	0	1	3	0	1	0	0	1	0	1	1	1	0
1	1337x.to	0	0	1	3	0	1	0	0	1	0	1	1	1	0
2	1337x.to	0	0	1	3	0	1	0	0	1	0	1	1	1	0
3	1337x.to	0	0	1	3	0	1	0	0	1	0	1	1	1	0
4	1337x.to	0	0	1	3	0	1	0	0	1	0	1	1	1	0

References:

- Python Language Reference: https://docs.python.org/3/reference/index.html

- Visual Studio code: https://code.visualstudio.com/docs
- Google Colab: https://research.google.com/colaboratory/
- Docker for Containerization: https://docs.docker.com/
- ISCX-URL-2016 Dataset: https://www.unb.ca/cic/datasets/url-2016.html