Configuration Manual

MSc Research Project
M.Sc Cyber Security

Krithika Ramesh
Student ID: 23251361

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Krithika Ramesh
Student ID:	23251361
Programme:	M.Sc Cyber Security **Year:** 2024
Module:	Practicum Part 2
Lecturer:	Khadija Hafeez
Submission Due Date:	12th December
Project Title:	Phishing Detection and Mitigation: A Cybersecurity and Machine Learning Approach
Word Count:	1400 **Page Count:** 20

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:	Krithika Ramesh
Date:	12-12-2024

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

Krithika Ramesh
Student ID: 23251361

This project as two parts, first one focussing on developing and training the model on Google Colab, while the other involves integrating it into the Django web application for the presentation. In the first part, I utilized one dataset in csv format, with a machine learning implementation, coding, and evaluation. The single dataset has been used on thee model Naïve Bayes. In the second part, I integrated the Email Spam detection machine learning model into a web application called Django. This step is crucial as it transforms our model into a user-friendly web app that anyone can use, even without coding knowledge. I will be covering everything from setting up the model and creating forms and views to build the user interface. This manual provides a detailed overview and a step-by-step guide of the project execution.
(*Download Python*, n.d.; *Download Visual Studio Code - Mac, Linux, Windows*, n.d.; *GitHub · Build and Ship Software on a Single, Collaborative Platform*, 2024; *Kaggle*, n.d.)

1 System Requirements

- Operating System: Windows 11
- Processor: Intel i5
- RAM: 8GB
- Disk Space: 20 GB
- Internet Connectivity: Intel(R) Wi-Fi 6E AX211 160MHz

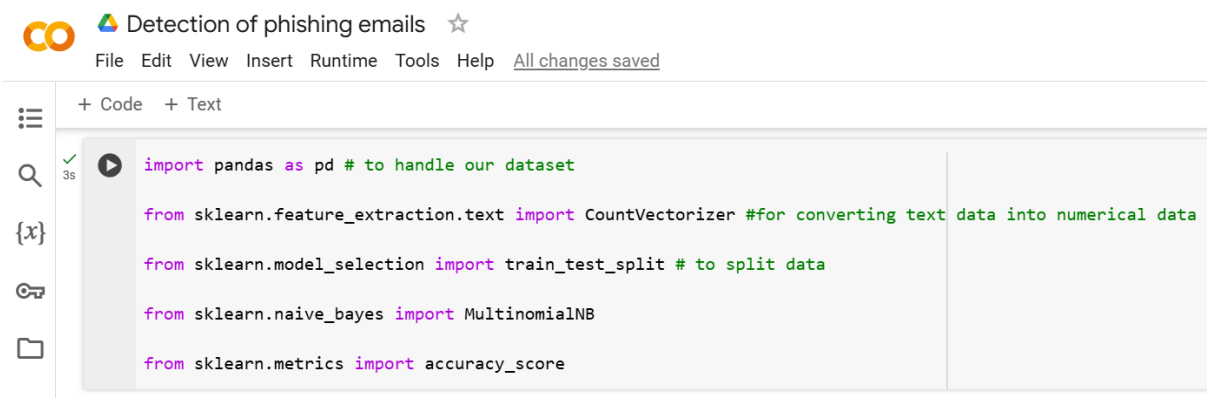
2 Tools and Frameworks

- Google Collab: For model creation and training
- Django: For building the web interface
- Python: Version 3.8

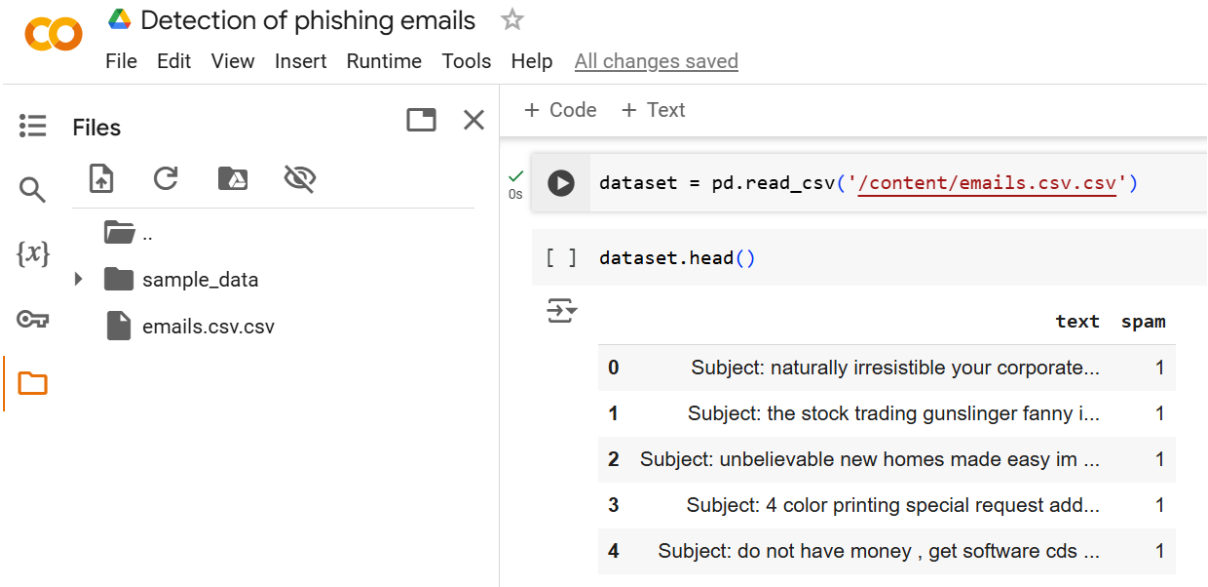
3 Machine Learning Model Configuration Step by step guide

3.1 Part I

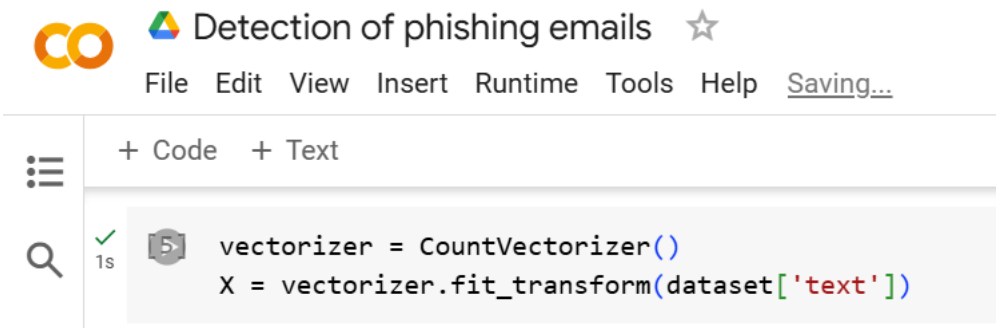
1. We start by importing necessary libraries, we will be using pandas to handle our dataset and sklearn for machine learning. Since the dataset file is in the form of text data, we need to first convert it into a format(numerical) that the machine learning model can process. To achieve this, we use sklearn. We will further split our data into training and testing set, as it allows us to train our model on one portion of data and test it on another portion to evaluate its performance. To build a spam detection classifier we will use naïve bayes algorithm. In order to assess the performance of the machine learning model, we need to measure its accuracy on the test data. Again use sklearn for this.



2. The next step is to upload the csv file and read the path of the file. We then view the csv file, for this we use the head function. Here we have two columns, one column depicts text of emails and the other column shows spam. In this case, 1 means it is spam, and 0 means it is ham which means it is not spam.



3. Now use Countvectorizer to convert text data into numerical format that the machine learning model can process. For this use vectorizer, then use and define variables as text.



4. Next, use train_test_split to split the data using variables and allotting size. In this case 80% data is for training and 20% is for testing.

Detection of phishing emails

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
[5] vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(dataset['text'])
```

```
[8] X_train, X_test, y_train, y_test = train_test_split(X, dataset['spam'], test_size=0.2)
```

6. Next, we will train the Naïve Bayes classifier using training data. For this, we will be using multinomialNB.

Detection of phishing emails

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[9] model = MultinomialNB()
    model.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

7. After training the model, it is now necessary to make predictions on the test set and evaluate its performance. In this step, the accuracy of the model created is achieved.

Detection of phishing emails

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[10] yPred = model.predict(X_test)

     #Evaluate the model

     accuracy = accuracy_score(y_test, yPred)
     print(accuracy)
```

```
0.9886561954624782
```

8. Finally, let us now check a message that uses our trained model to predict if a given message is spam or not. This function is specially useful, when we integrate our model into the web application. Firstly pass the variable, and the function is created. This function takes a message as input, vectorizes it using the same count vectorizer and uses the model to predict if it is a spam or not. After passing vairables and using print functions, enter the text on the text box. An example message is shown in the below screenshot.

Detection of phishing emails

File Edit View Insert Runtime Tools Help

+ Code + Text

```python
def predictMessage(message):
  messageVector = vectorizer.transform([message])
  prediction = model.predict(messageVector)
  return 'Spam' if prediction[0] == 1 else 'Ham'

userMessage = input('Enter text to predict: ')
prediction = predictMessage(userMessage)
print(f'The message is: {prediction}')
```

```
Enter text to predict:
```

Detection of phishing emails

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

17s

```python
def predictMessage(message):
  messageVector = vectorizer.transform([message])
  prediction = model.predict(messageVector)
  return 'Spam' if prediction[0] == 1 else 'Ham'

userMessage = input('Enter text to predict: ')
prediction = predictMessage(userMessage)
print(f'The message is: {prediction}')
```

```
Enter text to predict: Hey, You have won an Iphone! Click here to claim.
The message is: Spam
```

Detection of phishing emails

File Edit View Insert Runtime Tools Help

+ Code + Text

1m

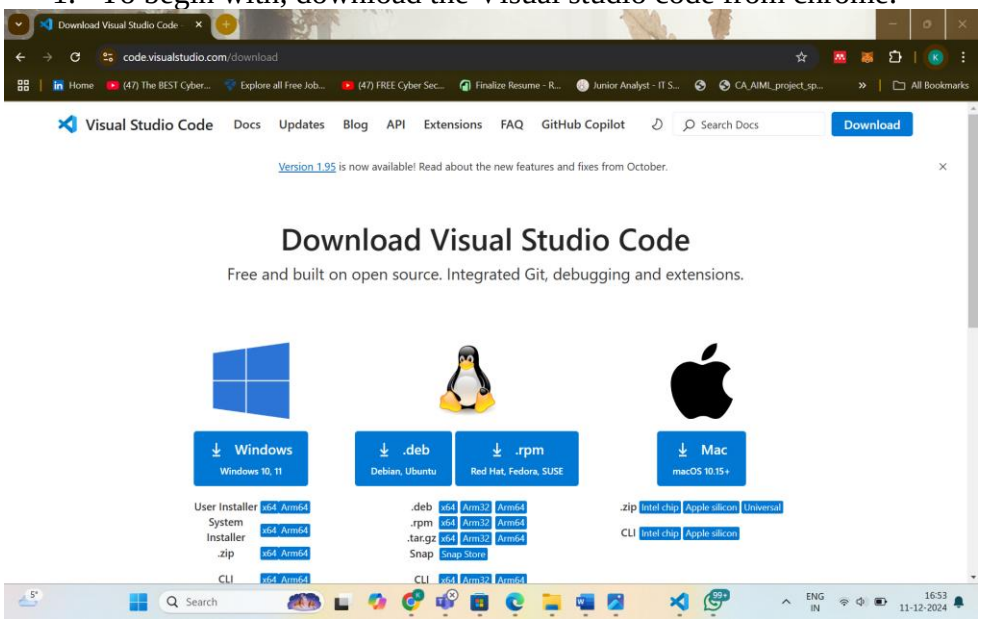
```python
def predictMessage(message):
  messageVector = vectorizer.transform([message])
  prediction = model.predict(messageVector)
  return 'Spam' if prediction[0] == 1 else 'Ham'

userMessage = input('Enter text to predict: ')
prediction = predictMessage(userMessage)
print(f'The message is: {prediction}')
```

```
Enter text to predict: Hey, are we meeting for dinner?
The message is: Ham
```

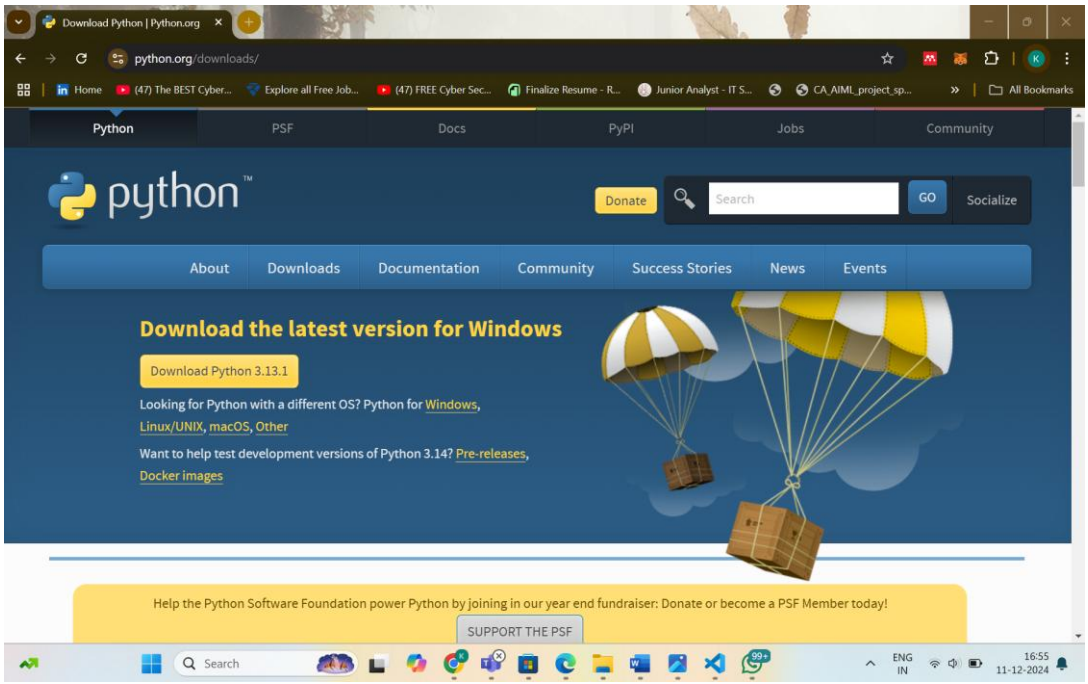
So, we have successfully built a spam detection classifier using a machine learning model, and in the next part, this model is integrated into a web application Django.

3.2 Part II

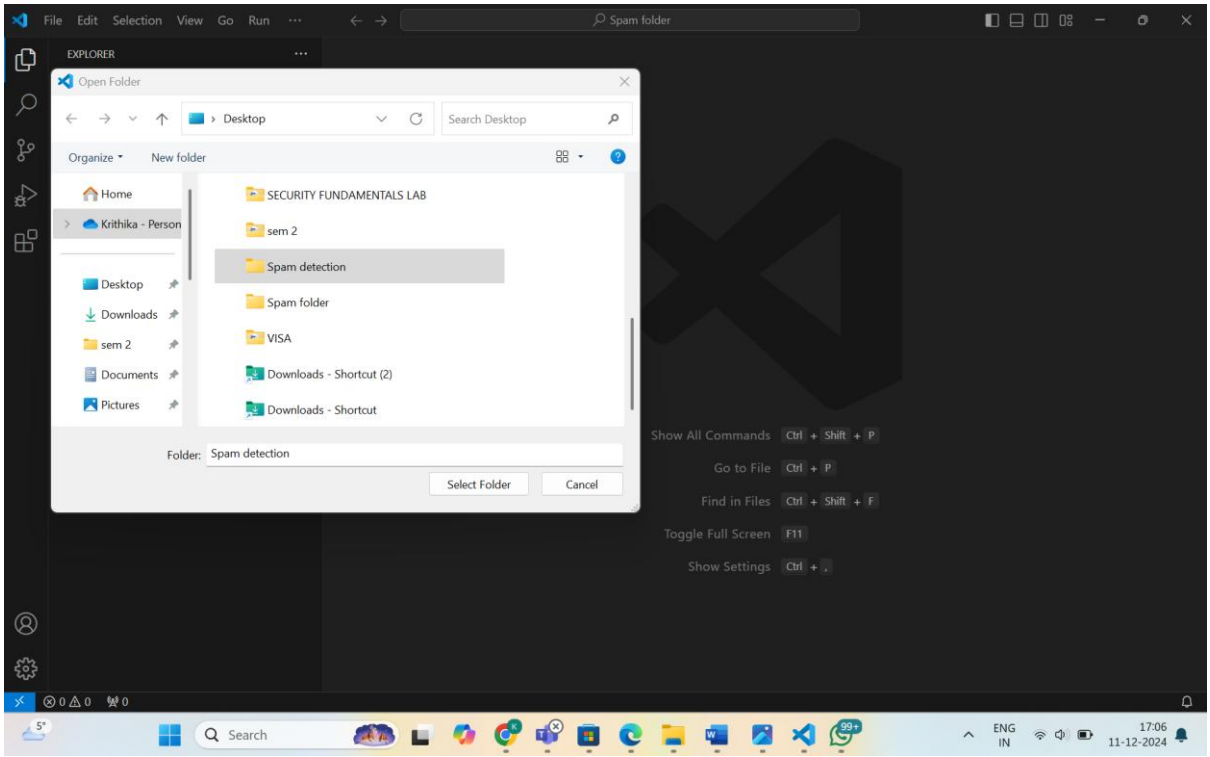
1. To begin with, download the Visual studio code from chrome.



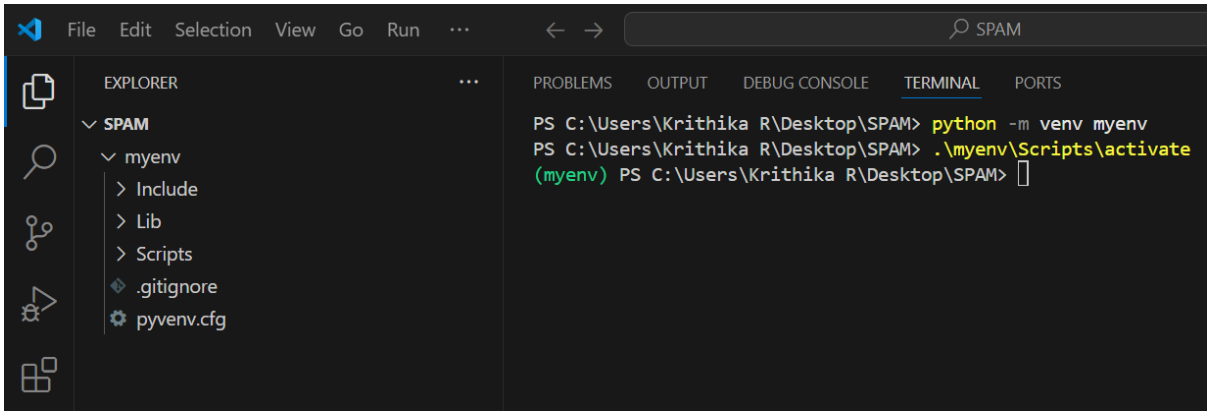
2. For Python, install the latest version of Python from chrome



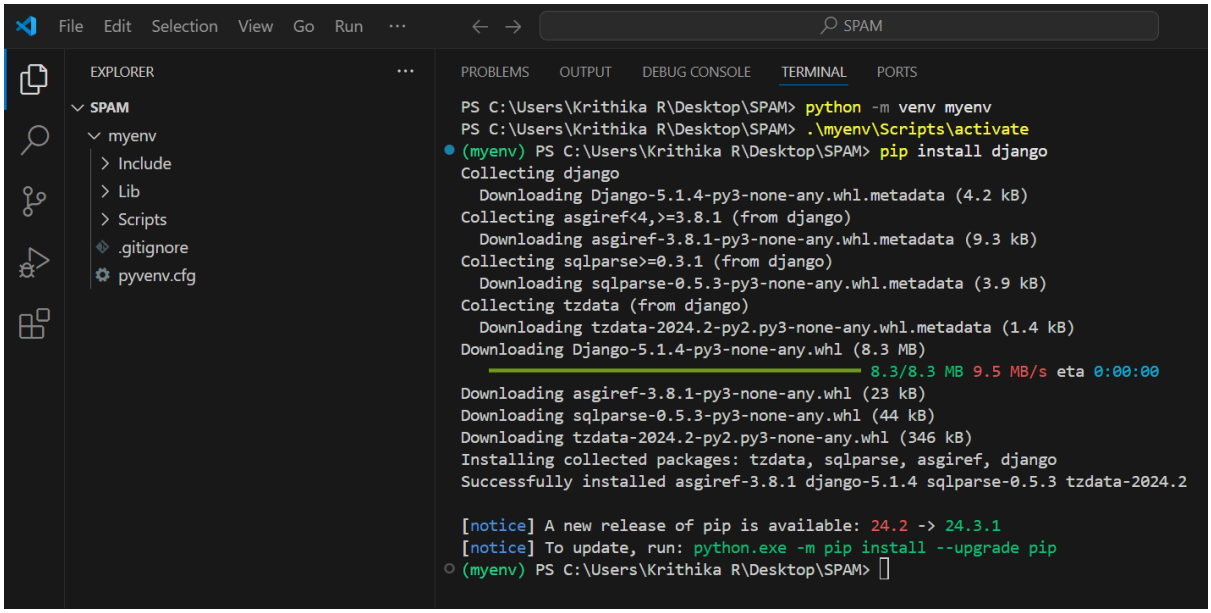
3. Now, let us get started with VS Code, first click on file and create an empty folder and select it.



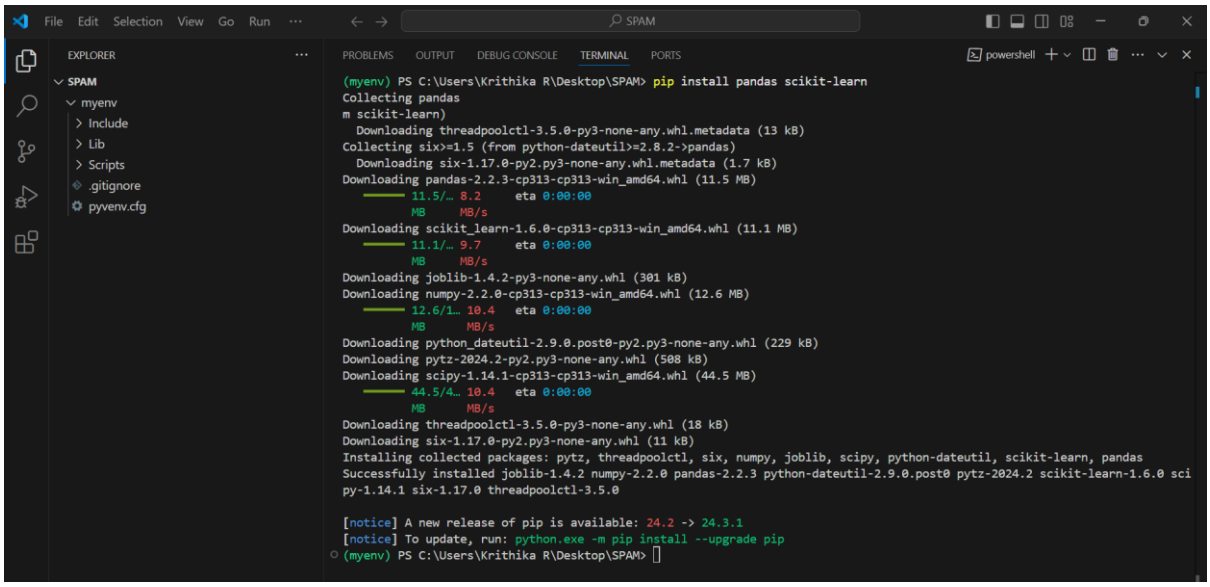
4. Open a new terminal, and the first step is to create a virtual environment as it helps many dependencies for the project to be isolated. For this use python -m venv myenv and activate it.



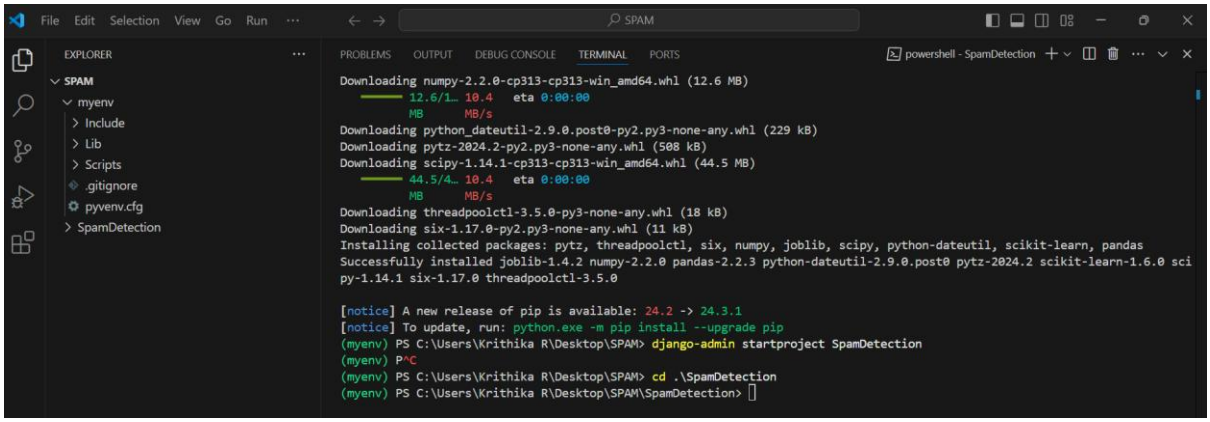
5. Now, we can setup the Django project, with the virtual environment activated, but first we need to install Django and other dependencies. Django is a python framework, so we use pip.

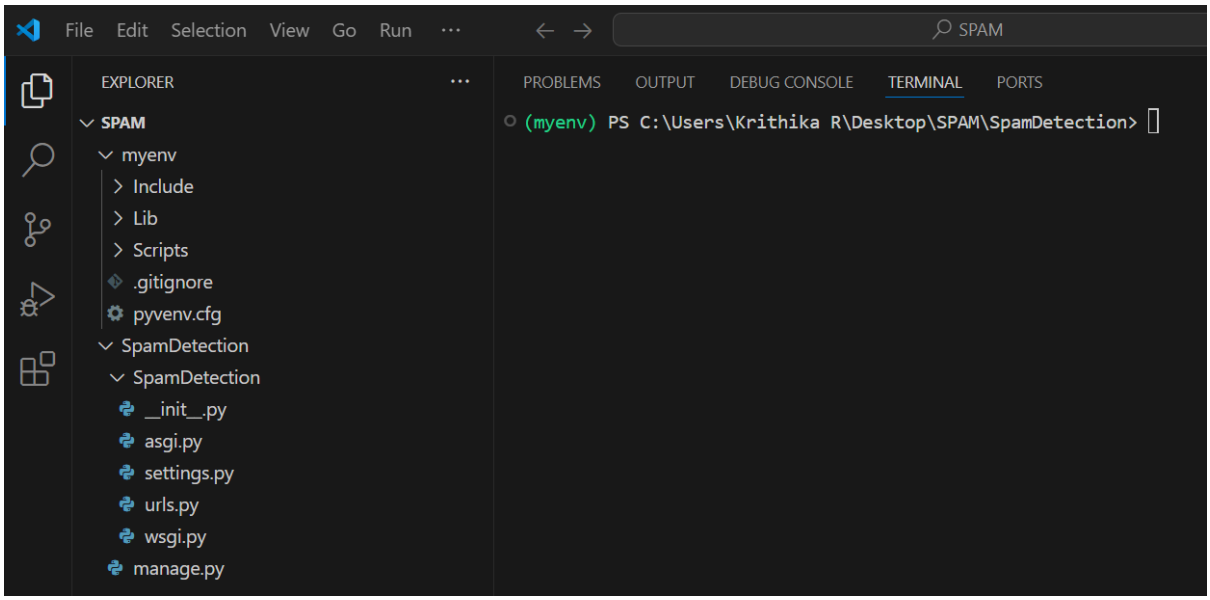


6. And in google Collab, we used pandas and sklearn, so we also need to download these two libraries.

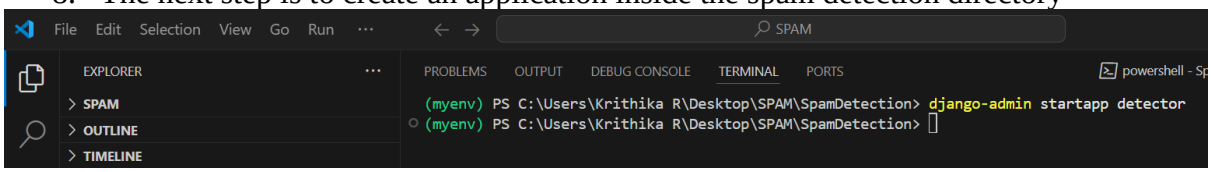


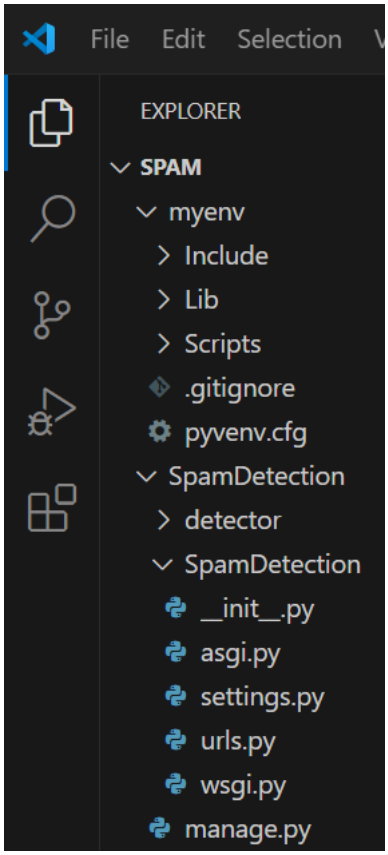
7. We can now set-up the django project, write django-admin startproject SpamDetection and further navigate to that directory, use cd command for navigation.



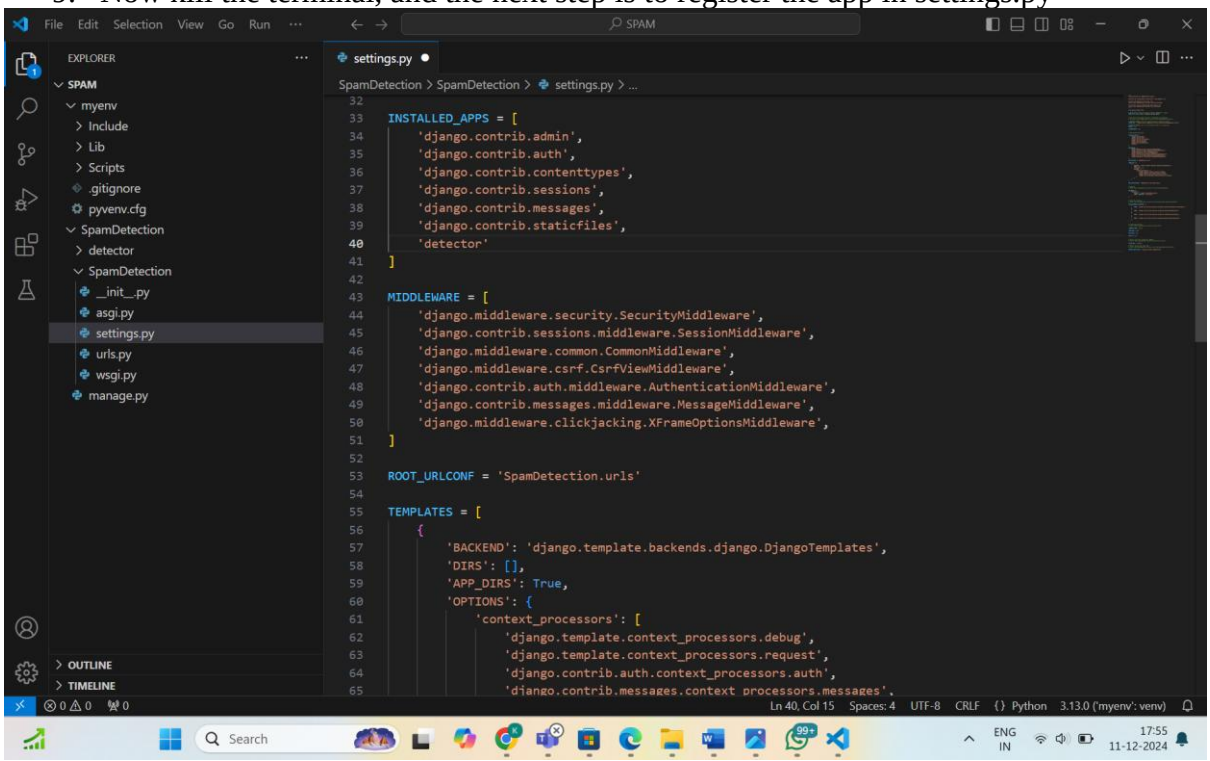


8. The next step is to create an application inside the spam detection directory



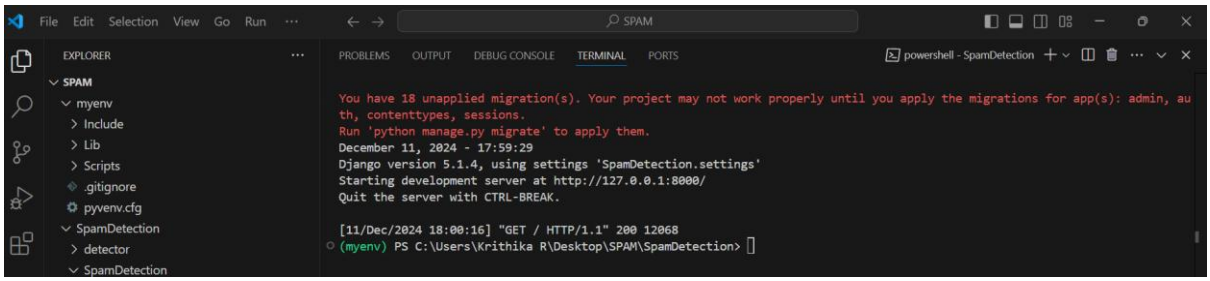


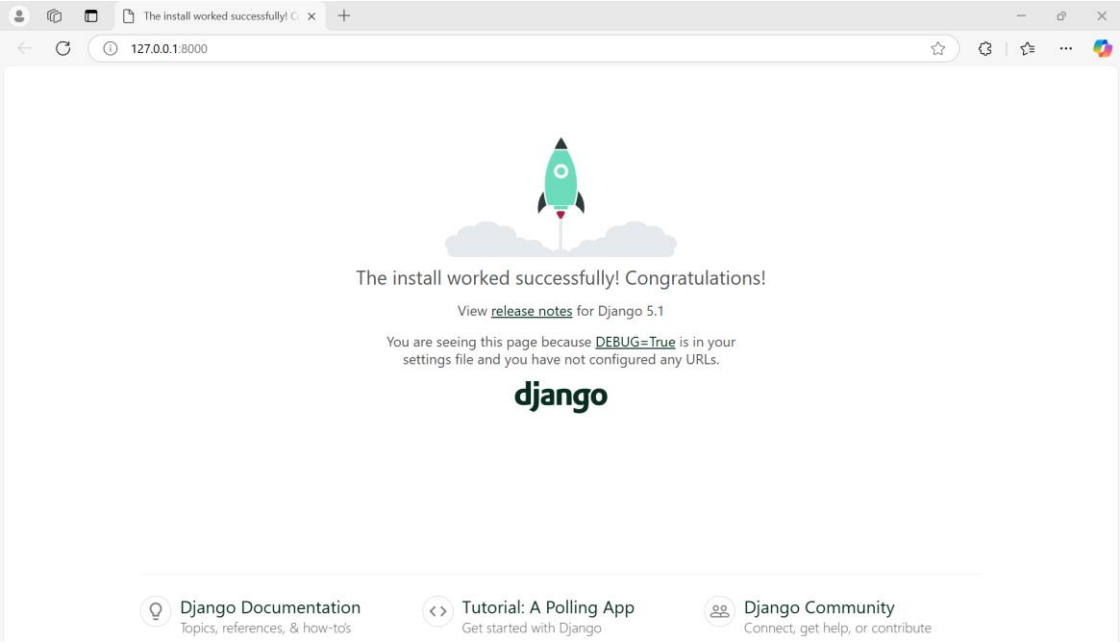
9. Now kill the terminal, and the next step is to register the app in settings.py



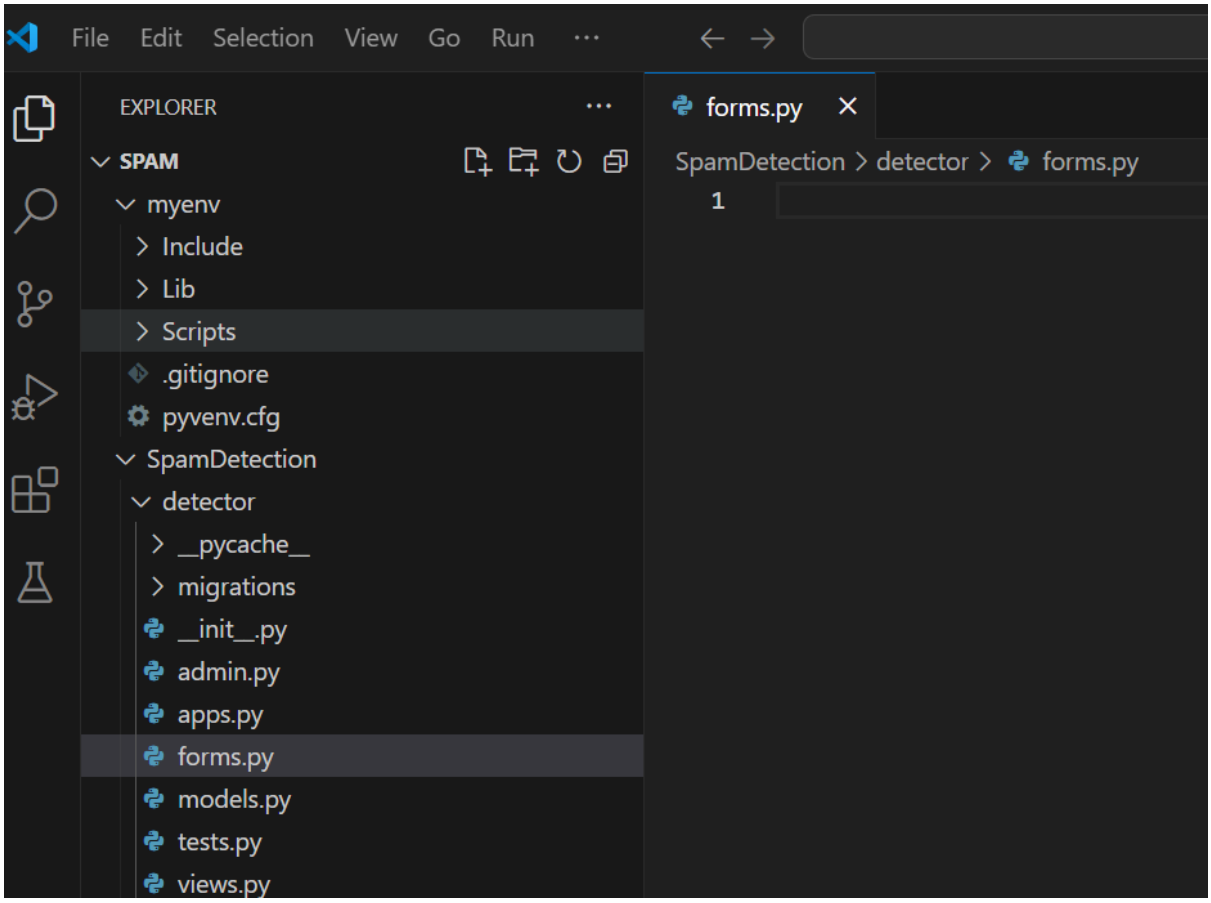
10. Save and close the file and test if the project is working fine or not by using .\myenv\Scripts\activate

11. Now, I am going to start the server, using python manage.py runserver

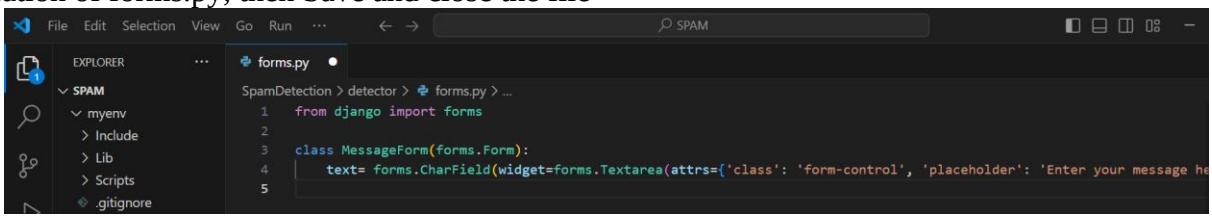




12. The nect step is to create a form and input a message to check if it is a spam or not. To do this, create a new file forms.py in the app directory



Creation of forms.py, then Save and close the file



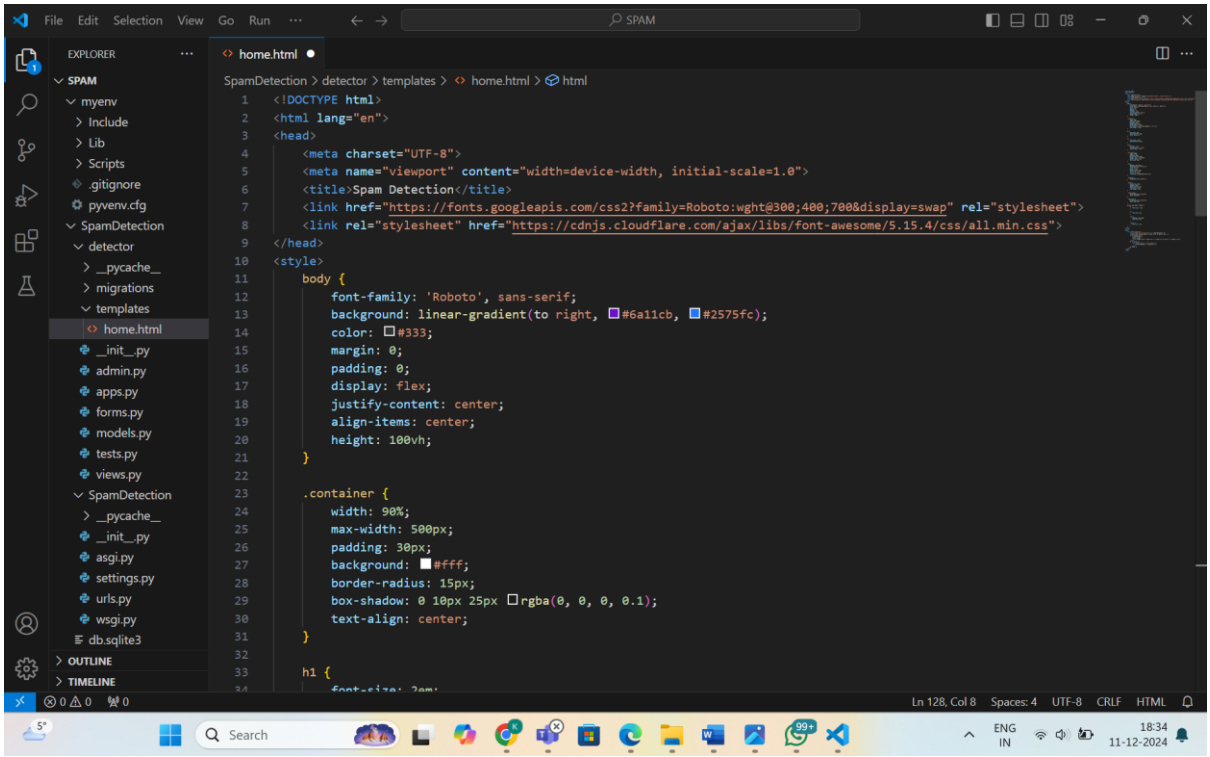
13. Now open the views.py file in the app directory. This step is very important as its where we write the code to handle the requests and responses. And we also integrate the machine learning model, so the same code has been used to make the model accessible to the web application. So, first paste the code from google colab and write the code to get message form.

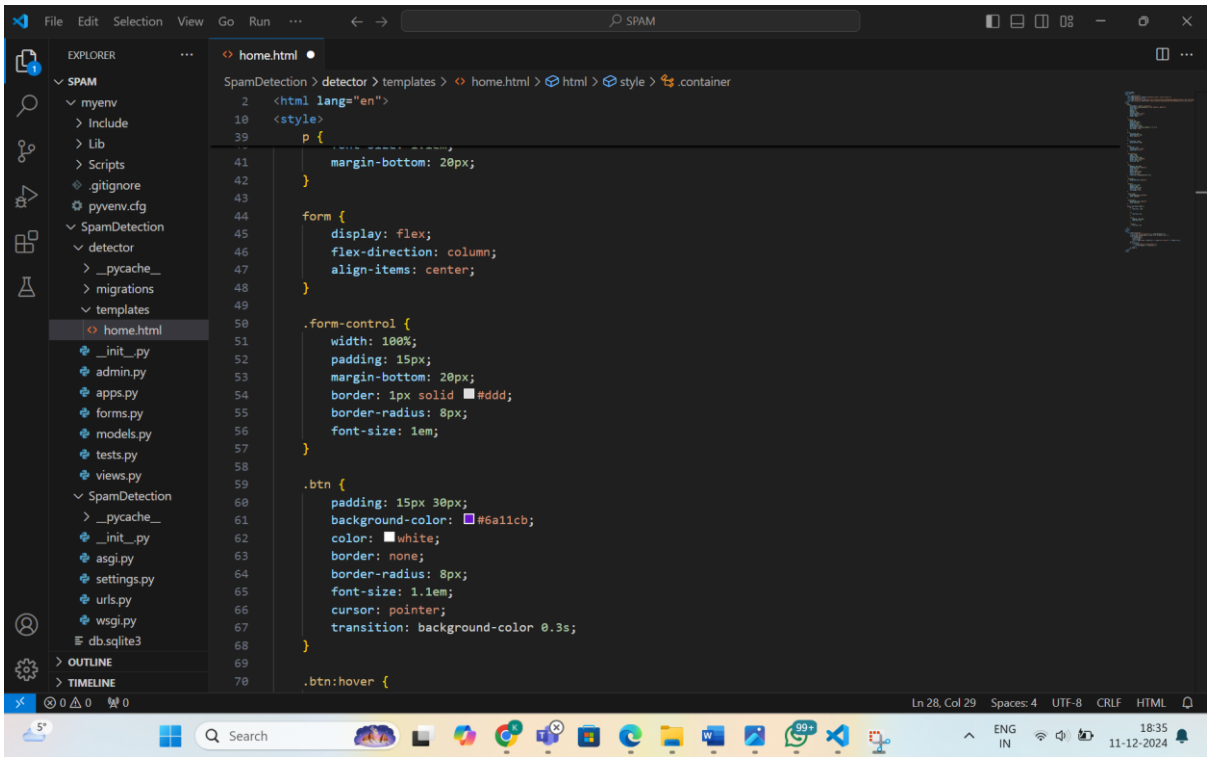
```python
from django.shortcuts import render

# Create your views here.
import pandas as pd # to handle our dataset

from sklearn.feature_extraction.text import CountVectorizer #for converting text data into numerical data

from sklearn.model_selection import train_test_split # to split data

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score

from .forms import MessageForm

dataset = pd.read_csv('C:\Users\Krithika R\Downloads/emails.csv.csv')

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(dataset['text'])

X_train, X_test, y_train, y_test = train_test_split(X, dataset['spam'], test_size=0.2)

model = MultinomialNB()
model.fit(X_train, y_train)

def predictMessage(message):
  messageVector = vectorizer.transform([message])
  prediction = model.predict(messageVector)
  return 'Spam' if prediction[0] == 1 else 'Ham'
```

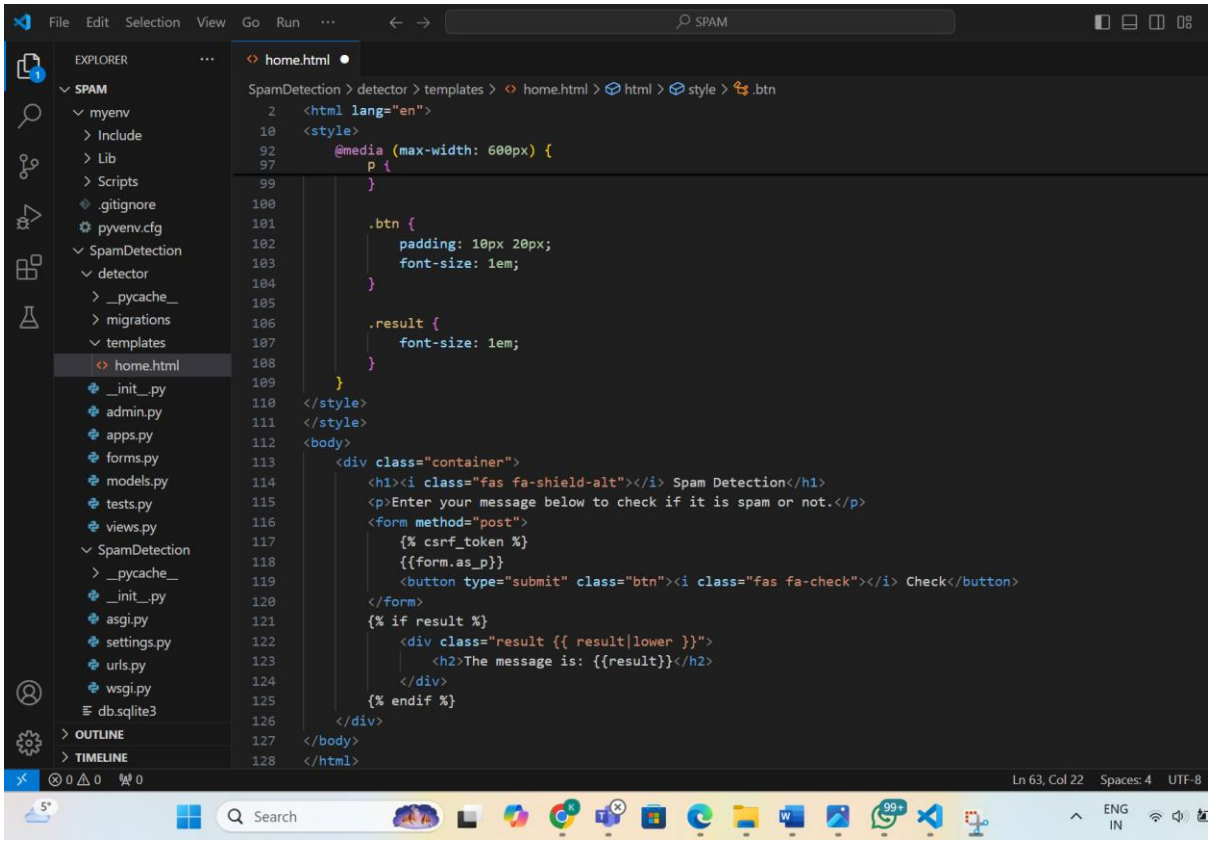
```python


def Home(request):
  result = None
  if request.method == 'POST':
    form = MessageForm(request.POST)
    if form.is_valid():
      message = form.cleaned_data['text']
      result = predictMessage(message)

    else:
      form = MessageForm()

      return render(request, 'home.html', {'form': form, 'result': result})
```

Save and close the file.

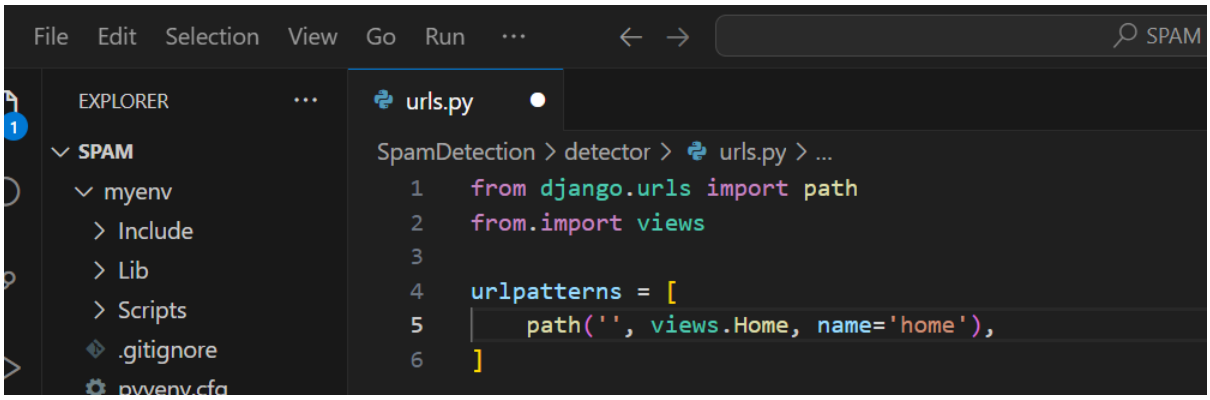
14. The next step is to create a user interface. First create a new folder named templates in the app directory and create a file inside this app directory called home.html. This is necessary because Django looks for HTML templates in the templates folder by default.


File Edit Selection View Go Run
SPAM
EXPLORER
home.html
SpamDetection > detector > templates > home.html > html
SPAM
myenv
Include
Lib
Scripts
.gitignore
pyvenv.cfg
SpamDetection
detector
__pycache__
migrations
templates
home.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
views.py
SpamDetection
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
OUTLINE
TIMELINE
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Spam Detection</title>
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
</head>
<style>
body {
font-family: 'Roboto', sans-serif;
background: linear-gradient(to right, #6a11cb, #2575fc);
color: #333;
margin: 0;
padding: 0;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
}
.container {
width: 90%;
max-width: 500px;
padding: 30px;
background: #fff;
border-radius: 15px;
box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
text-align: center;
}
h1 {
Ln 128, Col 8
Spaces: 4
UTF-8
CRLF
HTML
Search
ENG
IN
18:34
11-12-2024


File Edit Selection View Go Run
SPAM
EXPLORER
home.html
SpamDetection > detector > templates > home.html > html > style > .container
SPAM
myenv
Include
Lib
Scripts
.gitignore
pyvenv.cfg
SpamDetection
detector
__pycache__
migrations
templates
home.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
views.py
SpamDetection
__pycache__
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db.sqlite3
OUTLINE
TIMELINE
<html lang="en">
<style>
p {
margin-bottom: 20px;
}
form {
display: flex;
flex-direction: column;
align-items: center;
}
.form-control {
width: 100%;
padding: 15px;
margin-bottom: 20px;
border: 1px solid #ddd;
border-radius: 8px;
font-size: 1em;
}
.btn {
padding: 15px 30px;
background-color: #6a11cb;
color: white;
border: none;
border-radius: 8px;
font-size: 1.1em;
cursor: pointer;
transition: background-color 0.3s;
}
.btn:hover {
Ln 28, Col 29
Spaces: 4
UTF-8
CRLF
HTML
Search
ENG
IN
18:35
11-12-2024

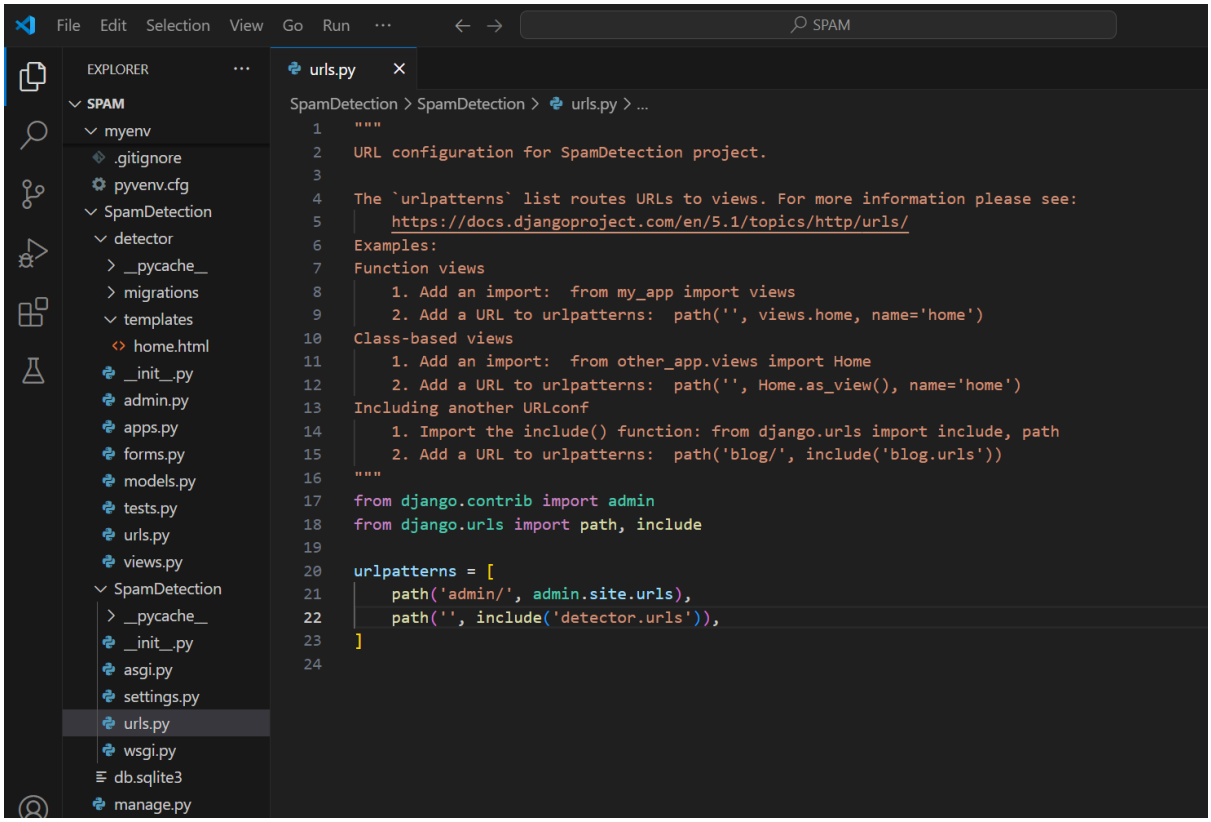


15. Now our last step is to set up a URL link for the website. For this, create urls.py file in the app directory

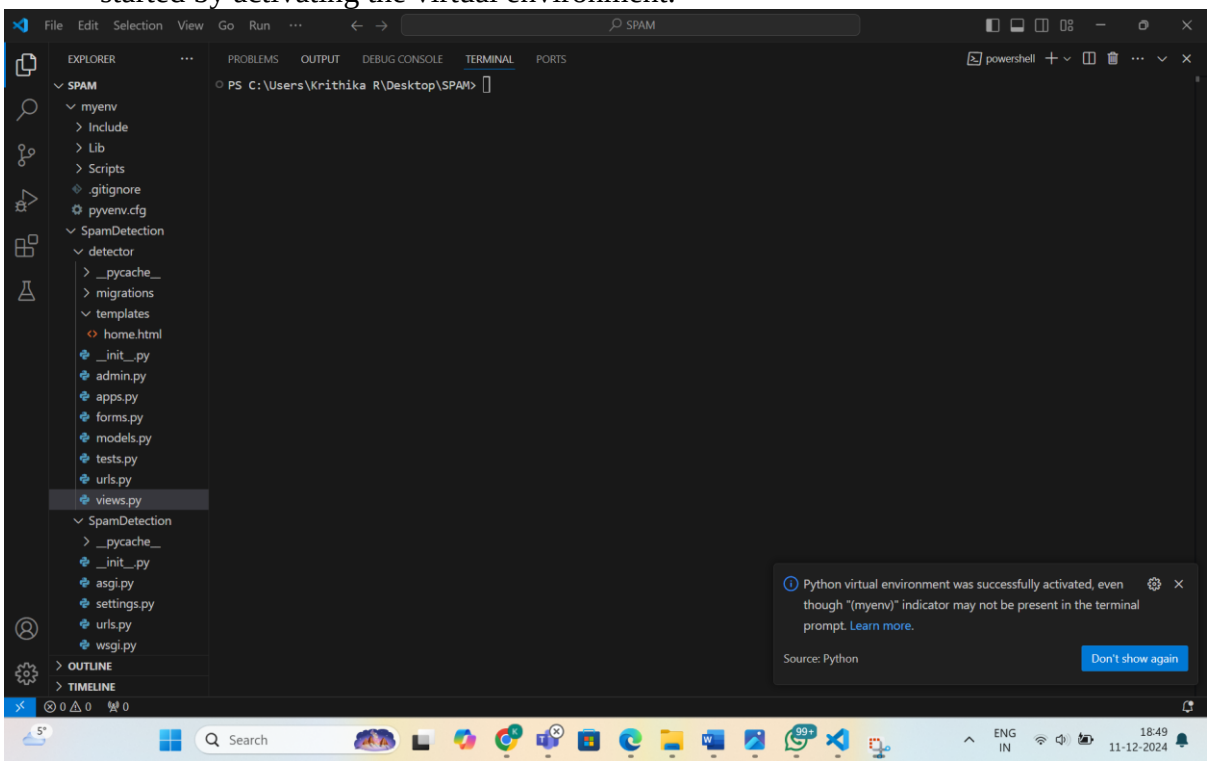


Save and close the file

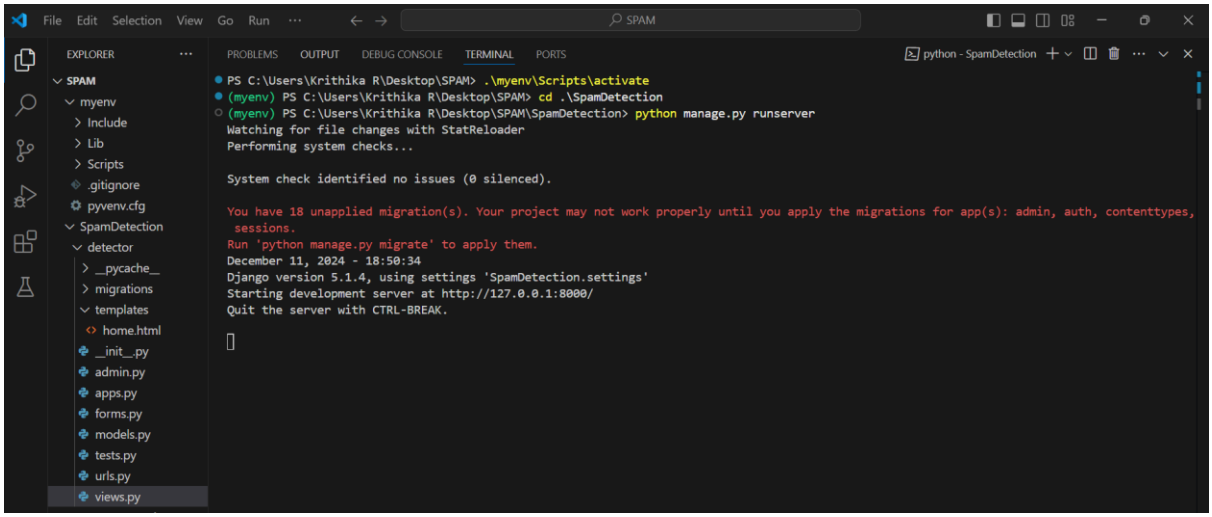
16. And next, in the project directory open the urls.py file. The app url is included here and edited in the code.

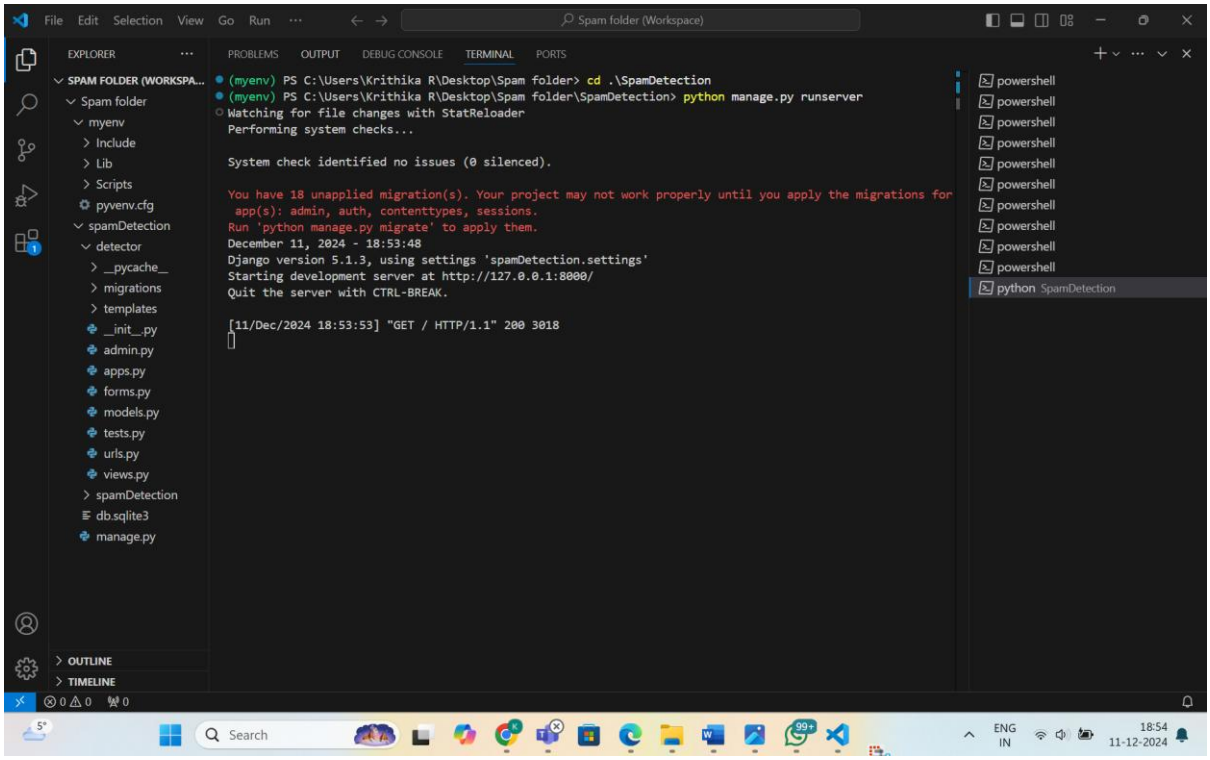


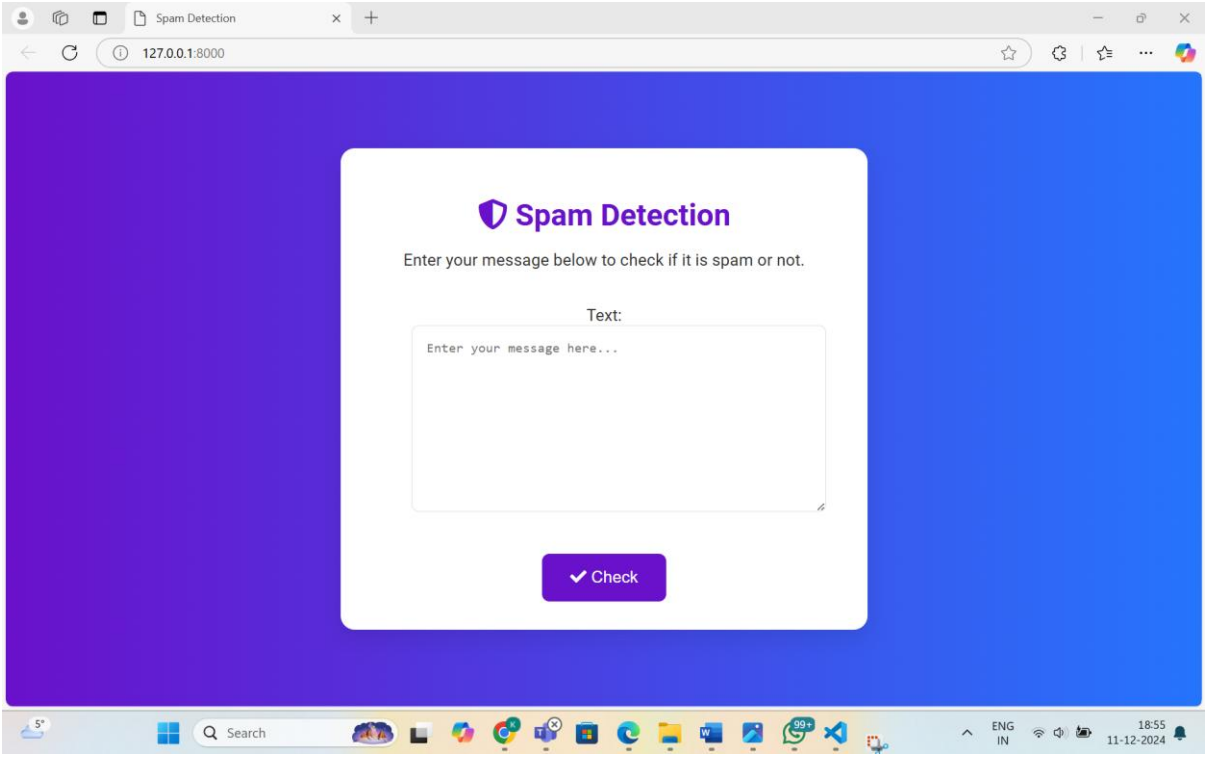
17. Finally, let us now run the django server and test our application. For this I again started by activating the virtual environment.



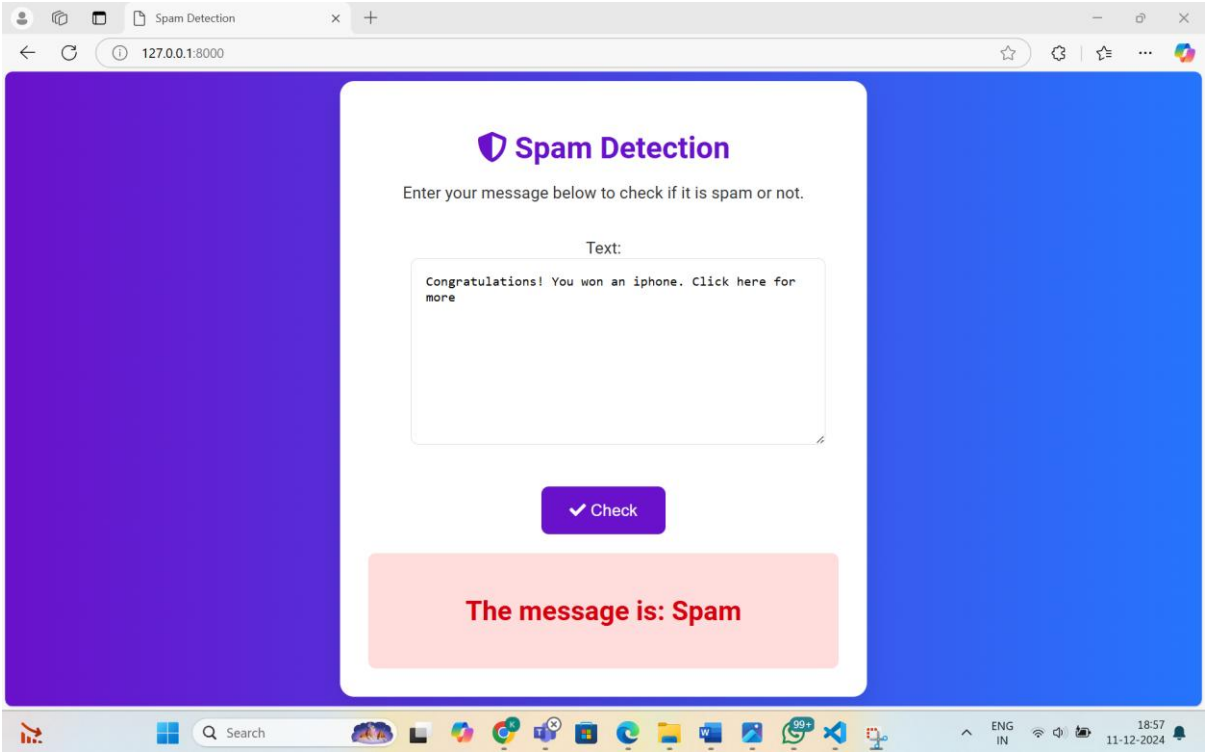
Congratulations! This is the final stage.


File Edit Selection View Go Run
SPAM
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python - SpamDetection
SPAM
myenv
Include
Lib
Scripts
.gitignore
pyvenv.cfg
SpamDetection
detector
__pycache__
migrations
templates
home.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
PS C:\Users\Krithika R\Desktop\SPAM> .\myenv\Scripts\activate
(myenv) PS C:\Users\Krithika R\Desktop\SPAM> cd .\SpamDetection
(myenv) PS C:\Users\Krithika R\Desktop\SPAM\SpamDetection> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 11, 2024 - 18:50:34
Django version 5.1.4, using settings 'SpamDetection.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

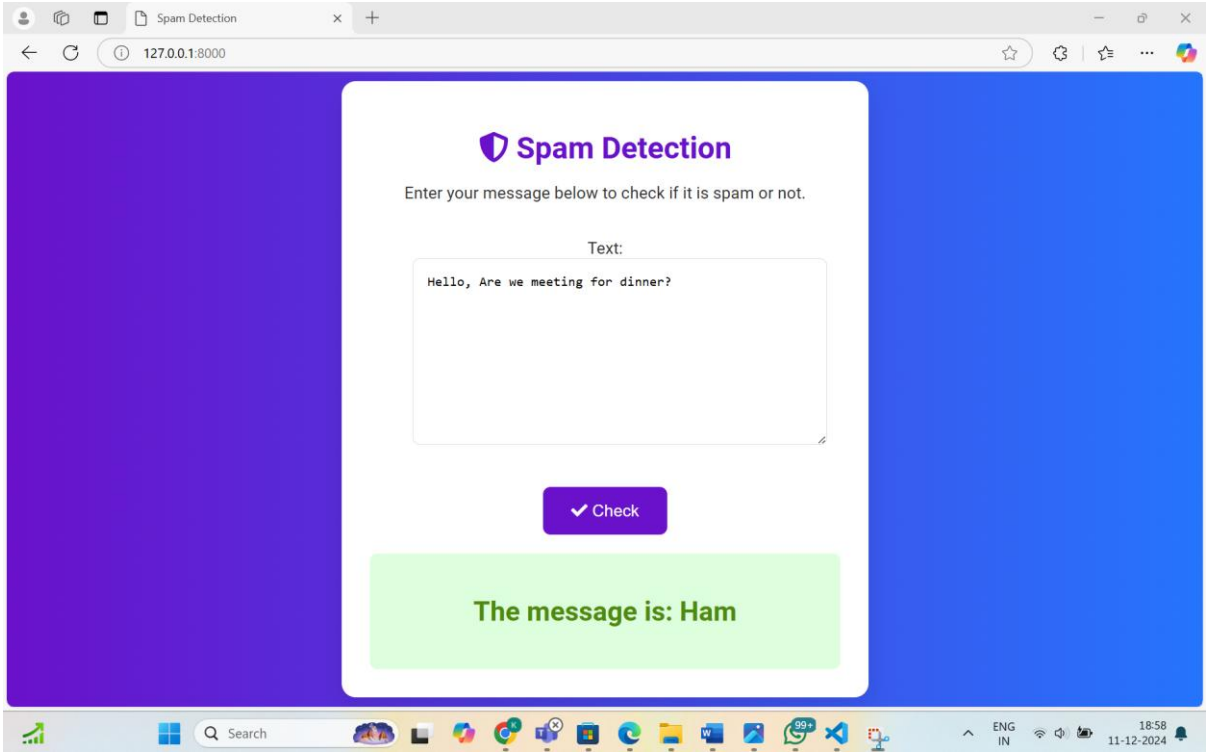

File Edit Selection View Go Run
Spam folder (Workspace)
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
SPAM FOLDER (WORKSPA...
Spam folder
myenv
Include
Lib
Scripts
pyvenv.cfg
spamDetection
detector
__pycache__
migrations
templates
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
spamDetection
db.sqlite3
manage.py
OUTLINE
TIMELINE
(myenv) PS C:\Users\Krithika R\Desktop\Spam folder> cd .\SpamDetection
(myenv) PS C:\Users\Krithika R\Desktop\Spam folder\SpamDetection> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 11, 2024 - 18:53:48
Django version 5.1.3, using settings 'spamDetection.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[11/Dec/2024 18:53:53] "GET / HTTP/1.1" 200 3018
powershell
powershell
powershell
powershell
powershell
powershell
powershell
powershell
powershell
python SpamDetection
Search
ENG IN
18:54
11-12-2024

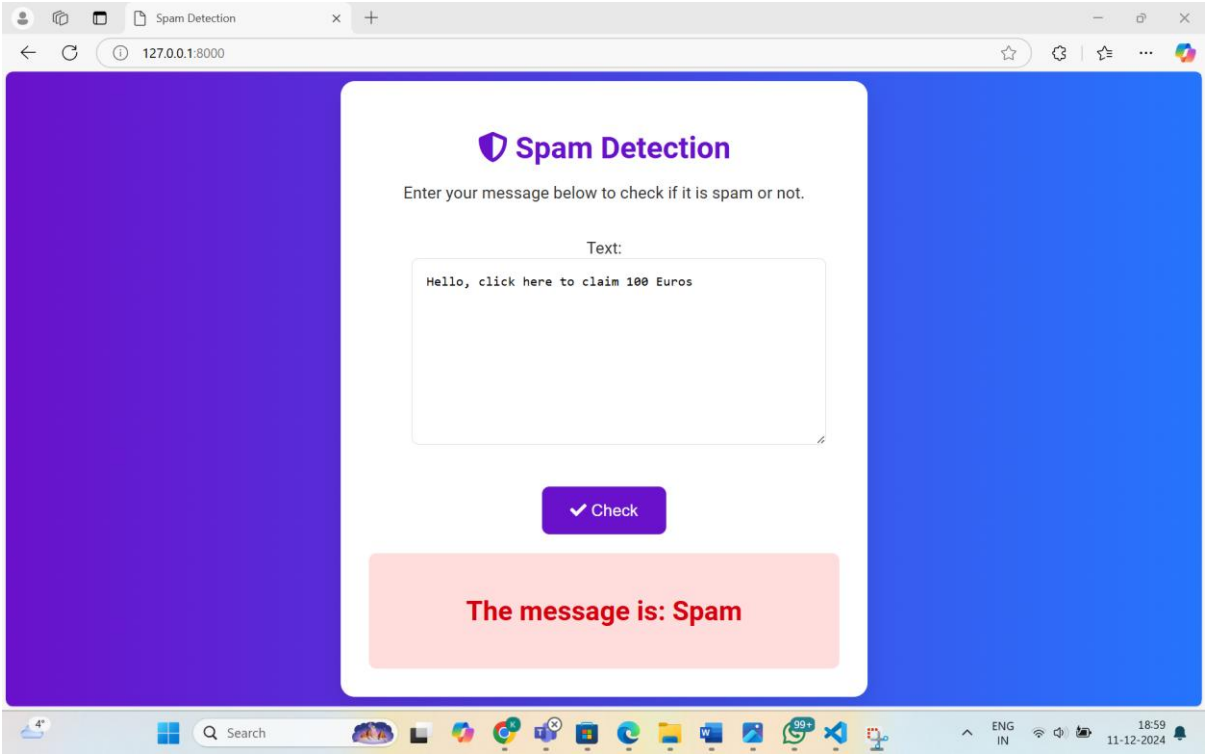


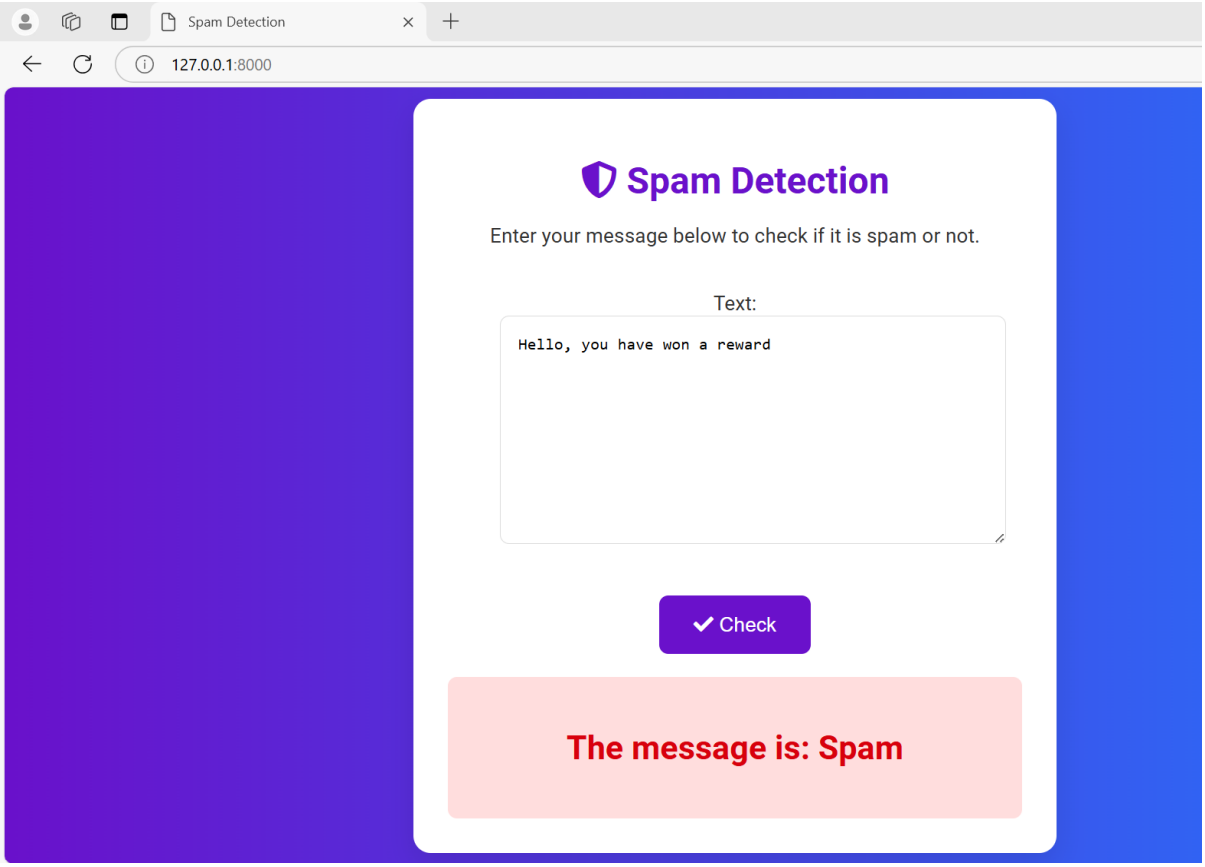
This is our web user interface, now first test it by adding any message.



As you can see this message is a spam.


Spam Detection
127.0.0.1:8000
Spam Detection
Enter your message below to check if it is spam or not.
Text:
Hello, Are we meeting for dinner?
Check
The message is: Ham


Spam Detection
127.0.0.1:8000
Spam Detection
Enter your message below to check if it is spam or not.
Text:
Hello, click here to claim 100 Euros
Check
The message is: Spam



Here are a few examples of messages and its screenshots are shown above.

References

Download Python. (n.d.). Python.Org. Retrieved December 12, 2024, from https://www.python.org/downloads/

Download Visual Studio Code—Mac, Linux, Windows. (n.d.). Retrieved December 12, 2024, from https://code.visualstudio.com/Download

GitHub · Build and ship software on a single, collaborative platform. (2024). GitHub. https://github.com/

Kaggle: Your Machine Learning and Data Science Community. (n.d.). Retrieved December 12, 2024, from https://www.kaggle.com/