National College of Ireland

Configuration Manual

MSc Research Project
MSc in CyberSecurity

Avani Naidu
Student ID: 23285044

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Avani Chandrashekhar Naidu
Student ID:	23285044
Programme:	MSc in Cybersecurity **Year:** 2024-2025
Module:	Practicum
Lecturer:	Raza Ul Mustafa
Submission Due Date:	11-08-2025
Project Title:	Configuration Manual
Word Count:	1465 **Page Count: 12**

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:	Avani Chandrashekhar Naidu
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

Avani Naidu
Student ID: 23285044

1 Introduction

This configuration guide takes the user through deployment and configuration to the implementation of the project entitled Forensic Analysis of Synthetic speech: A Hybrid Approach to Audio Deep fake Detection. The presented project is based on machine learning, spectrogram analysis, and metadata forensics to offer an explainable and automated framework of identifying computer-generated (deepfake) audio.

The system uses the following tools in the following ways total audio feature detection with Librosa, total convolutional neural network classifications with TensorFlow/Keras, explainable AI with SHAP and optional VoIP monitor/Wireshark to capture VoIP traffic. This document serves not only to document but also to reproduce the process and it also gives implementers a step-by-step guide as to how to set things up to reduce configuration errors and permits the deployment of the detection pipeline to be effected seamlessly, this time both on standalone testing, as well as in the future deployment into real-time systems.

2 Prerequisites

2.1 Hardware requirements.

Processing Environment: Computing device that is able to execute machine learning tasks. To create and test this project a virtual machine (VM) was used.

Minimum Specifications:

- RAM: 4 GB or bigger (8 GB needed to generate spectrogram faster and run models on them).
- Storage: 40 GB of disk space on which to save the datasets (e.g., ASVspoof), images of spectrograms and trained models and logs.
- CPU: Dual core (Intel I5 or alike) or faster. The support of the GPU is not necessary but useful when training the CNN model.
- Network: Solid internet link to do dataset and package downloads, repositories pushes/pulls (e.g., GitHub).

2.2 Software Requirements

Operating System: Windows 10/11, Ubuntu 20.04+, or macOS 12+
Python 3.8 (Anaconda environment recommended)
Jupyter Notebook / JupyterLab for code execution (.ipynb file)
Required Python libraries:

- tensorflow / keras (for CNN model)
- scikit-learn (for Random Forest)
- librosa (audio processing)
- matplotlib, seaborn (visualization)

- shap (explainability for RF)
- opencv-python (Grad-CAM processing)
- pandas, numpy (data handling)
- streamlit (for interactive interface)

3 System Requirements

3.1 Installing softwares

Prior to the implementation of the project, Anaconda Python distribution that is compatible with the Ubuntu operating system has to be installed.

- **Figure 1** shows the Anaconda installer version that should be chosen correctly (64-bit, Python 3.x) in the case of Ubuntu.
- **Figure 2** displays the particular matplotlib version that should be used to support compatibility with the code used in the project and the visualizations.
- **Figure 3** shows how matplotlib can be installed in Ubuntu using the terminal as shown by the following command:

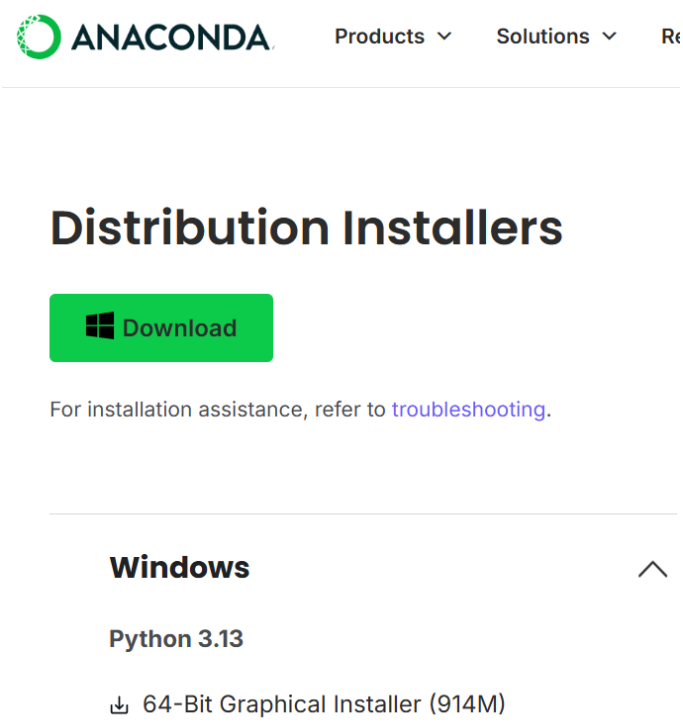

Figure 1: Anaconda Downloader.

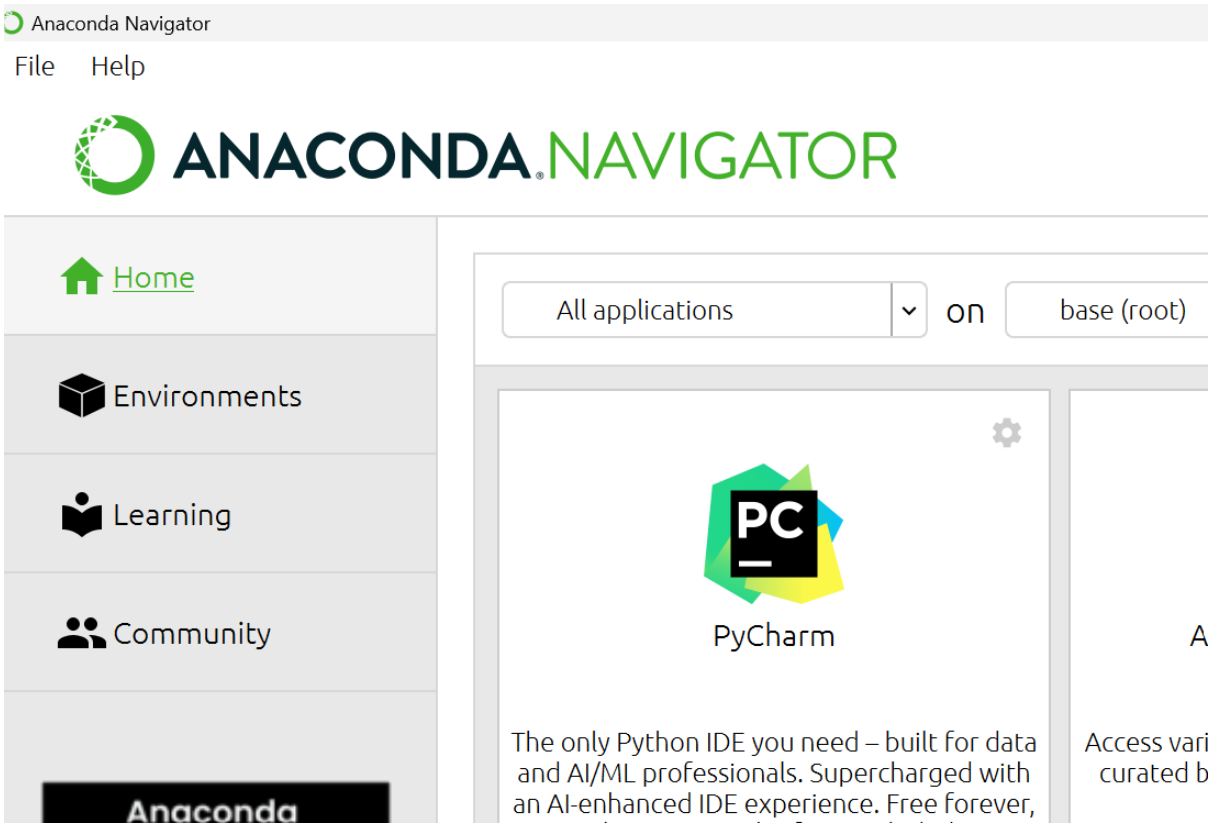

Figure 2: Home Page

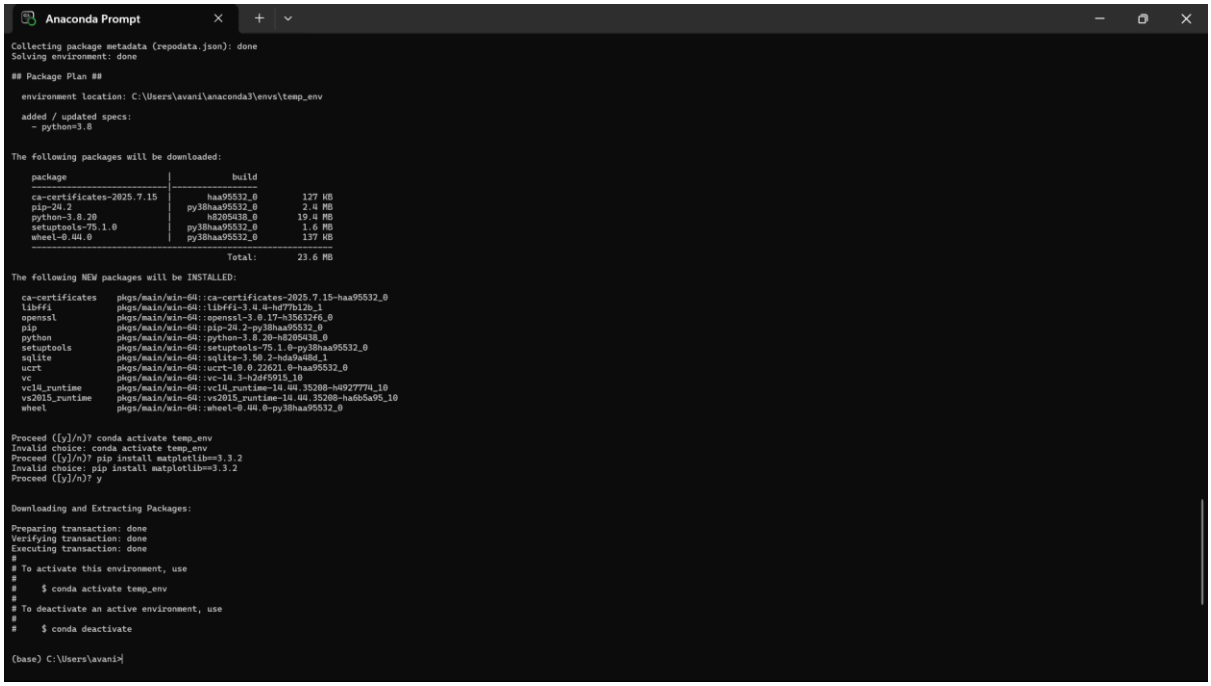

Figure 3: Installations required.

4 Installing Python packages/Libraries

```python
# Importing Required Libraries

# Core Libraries
import os
import re
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Audio Processing
import librosa
import librosa.display
from PIL import Image
# Deep Learning (TensorFlow/Keras)
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
# Machine Learning (Scikit-Learn)
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, StratifiedKFold
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    ConfusionMatrixDisplay
)
# Explainable AI
import shap
# Model Saving/Loading
import joblib
# Notebook Utilities
import nbformat
print("All libraries imported successfully")
```

```
All libraries imported successfully
```

Figure 4: Python Library and Packages

4.1 Implementation Flow and Performance Evaluation of Model

```
[3]: import numpy as np
     import os
     import librosa
     import librosa.display
     import matplotlib.pyplot as plt
     import random

     # === 1. Paths ===
     audio_dir = r'C:/Users/avani/Desktop/Assignment stuffs/Practicum/3rd sem/Datasets required/DS_10283_3336/LA/ASVspoof2019_LA_train/flac'
     labels_file = r'C:/Users/avani/Desktop/Assignment stuffs/Practicum/3rd sem/Datasets required/DS_10283_3336/LA/ASVspoof2019_LA_cm_protocols/ASVspoof2019.
     output_dir = 'mel_spectrograms_1000each'

     # === 2. Parse protocol file to get 1000 bonafide and 1000 spoof file IDs ===
     bonafide_files = []
     spoof_files = []

     with open(labels_file, 'r') as f:
         for line in f:
             parts = line.strip().split()
             file_id = parts[1]
             label = parts[-1].lower()
             if label == 'bonafide':
                 bonafide_files.append(file_id)
```

FIgure 5: Loading the Datasets and Preprocessing

```
    validation_data=val_gen,
    epochs=15,
    callbacks=[early_stop]
)

# === 5. Save the model ===
model.save("spoof_audio_detector_models_1000trainingnow.keras")
print("CNN model retrained and saved as 'spoof_audio_detector_models_1000trainingnow.keras'")
```

```
Found 1600 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
C:\Users\avani\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_s
nt to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\avani\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyData
super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do no
o `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 21s 97ms/step - accuracy: 0.7071 - loss: 0.6203 - val_accuracy: 0.8875 - val_loss: 0.3049
Epoch 2/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 16s 78ms/step - accuracy: 0.9039 - loss: 0.2703 - val_accuracy: 0.9200 - val_loss: 0.2469
Epoch 3/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 17s 83ms/step - accuracy: 0.9396 - loss: 0.1931 - val_accuracy: 0.9300 - val_loss: 0.2077
Epoch 4/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 16s 78ms/step - accuracy: 0.9373 - loss: 0.1579 - val_accuracy: 0.9350 - val_loss: 0.1911
Epoch 5/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 16s 78ms/step - accuracy: 0.9630 - loss: 0.1105 - val_accuracy: 0.9325 - val_loss: 0.1937
Epoch 6/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 15s 76ms/step - accuracy: 0.9817 - loss: 0.0757 - val_accuracy: 0.9350 - val_loss: 0.1948
Epoch 7/15
200/200 ━━━━━━━━━━━━━━━━━━━━ 15s 76ms/step - accuracy: 0.9831 - loss: 0.0667 - val_accuracy: 0.9325 - val_loss: 0.2056
CNN model retrained and saved as 'spoof_audio_detector_models_1000trainingnow.keras'
```

Figure 6: CNN Model Training Process and Performance

The figure demonstrates the definition of Convolutional Neural Network (CNN) architecture, compiling and training the model in detecting spoof audio using mel-spectrogram images. Training progress log illustrates improvement of accuracy and validation accuracy through 15 epochs and at that, as far as the model was saved under **spoof_audio_detector_models_1000trainingnow**, the validation accuracy of almost 93 percent was attained.

```
            score': prediction[0][0]
        })

# === Convert to DataFrame and save ===
df_results = pd.DataFrame(results)
df_results.to_csv('cnn_predictions.csv', index=False)
print("✅ Predictions saved to cnn_predictions.csv")
df_results.head()
```

✅ Predictions saved to cnn_predictions.csv

	filename	true_label	predicted_label	score
0	LA_T_1000406.png	bonafide	bonafide	0.013103
1	LA_T_1013597.png	bonafide	bonafide	0.006496
2	LA_T_1029184.png	bonafide	spoof	0.992430
3	LA_T_1049021.png	bonafide	bonafide	0.000537
4	LA_T_1052701.png	bonafide	bonafide	0.014088

Figure 7: CNN Model Predictions on Test Dataset

The graph shows the result of sample output of the trained CNN when utilized on the mel-spectrogram pictures in the test set. The table indicates what is the file name, the true label,

the predicted label, and the confidence of the prediction score. Most bonafide samples are correctly classified as such by the model and the probabilities scores of the bonafide samples are low, with some cases of misclassification (e.g., bonafide shares spoof results).

```
Accuracy: 96.75%
              precision    recall  f1-score   support

    bonafide       0.95      0.99      0.97      1000
       spoof       0.99      0.94      0.97      1000

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```

Figure 8: CNN Classification report

```
# === Save Model ===
joblib.dump(rf_model, "random_forest_mfcc_model.joblib")
print("✅ Model saved as 'random_forest_mfcc_model.joblib'")

📊 Classification Report:
              precision    recall  f1-score   support

    bonafide       0.00      0.00      0.00         3
       spoof       0.25      1.00      0.40         1

    accuracy                           0.25         4
   macro avg       0.12      0.50      0.20         4
weighted avg       0.06      0.25      0.10         4

✅ Model saved as 'random_forest_mfcc_model.joblib'
```

Figure 9: Random Forest model training on MFCC features for bonafide vs spoof classification

```
# --- Run the Batch Test ---
batch_test(
    audio_folder=r"C:\Users\avani\Desktop\Assignment stuffs\Practi
    protocol_path=r"C:\Users\avani\Desktop\Assignment stuffs\Pract
)

✅ Loaded label: LA_T_1138215 → bonafide
✅ Loaded label: LA_T_1271820 → bonafide
✅ Loaded label: LA_T_1272637 → bonafide
✅ Loaded label: LA_T_1276960 → bonafide
✅ Loaded label: LA_T_1341447 → bonafide

📋 Loaded 25380 labels from protocol

🔍 Checking: LA_T_1000137.flac
🔍 Checking: LA_T_1000406.flac
🔍 Checking: LA_T_1000648.flac
🔍 Checking: LA_T_1000824.flac
🔍 Checking: LA_T_1001074.flac
🔍 Checking: LA_T_1001114.flac
🔍 Checking: LA_T_1001169.flac
🔍 Checking: LA_T_1001718.flac
🔍 Checking: LA_T_1001871.flac
🔍 Checking: LA_T_1002656.flac
🔍 Checking: LA_T_1003665.flac
🔍 Checking: LA_T_9998739.flac
🔍 Checking: LA_T_9999048.flac
🔍 Checking: LA_T_9999170.flac
🔍 Checking: LA_T_9999337.flac
🔍 Checking: LA_T_9999560.flac
🔍 Checking: LA_T_9999757.flac
🔍 Checking: LA_T_9999995.flac

✅ Processed 25380 valid audio files
🎯 Accuracy: 0.5896375098502759
📋 Confusion Matrix:
[[14148  8652]
 [ 1763   817]]
📊 Classification Report:
              precision    recall  f1-score   support

       spoof       0.89      0.62      0.73     22800
    bonafide       0.09      0.32      0.14      2580

    accuracy                           0.59     25380
   macro avg       0.49      0.47      0.43     25380
weighted avg       0.81      0.59      0.67     25380
```

Figure 10: Loading the Trained Random Forest Model and Performing Predictions on New Audio Data

The Random Forest model which is trained and optimized is loaded in this step together with the saved indices of the features that are ranked highest. Each audio file as input has features

relevant to it, including MFCCs, chroma, zero-crossing rate to be filtered with only the most important ones being kept. These features can then be forwarded to the model to give predictions whereby the audio is classified as bonafide or spoof. The results are compared to ground truth labels in order to measure the actual model performance.

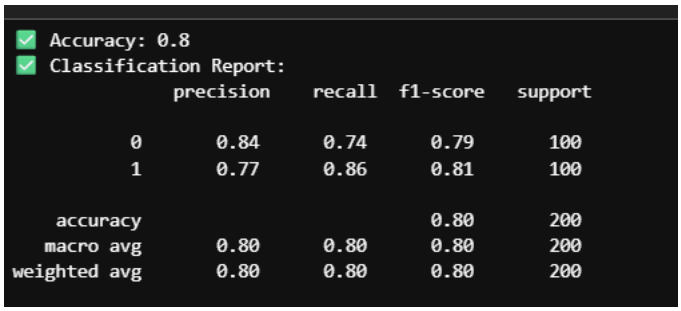

Figure 11. Performance metrics of the Random Forest model on MFCC features after tuning hyperparameters, achieving 80% accuracy with balanced precision and recall across bonafide and spoof classes

At First, Random Forest model with MFCC features had a comparatively low accuracy and this shows that there are default limitations built in the model. To overcome this, a systematic process was involved, i.e. balancing the data, feature selection and tuning of hyperparameters by grid search with cross-validation. Such optimizations had a great effect on the strength of the model to identify the authenticity of bonafide and spoof audio samples. This boosted the accuracy to 80% and the precision and recall proved to give a balanced performance between the classes.

5 Model Implementation and Deployment

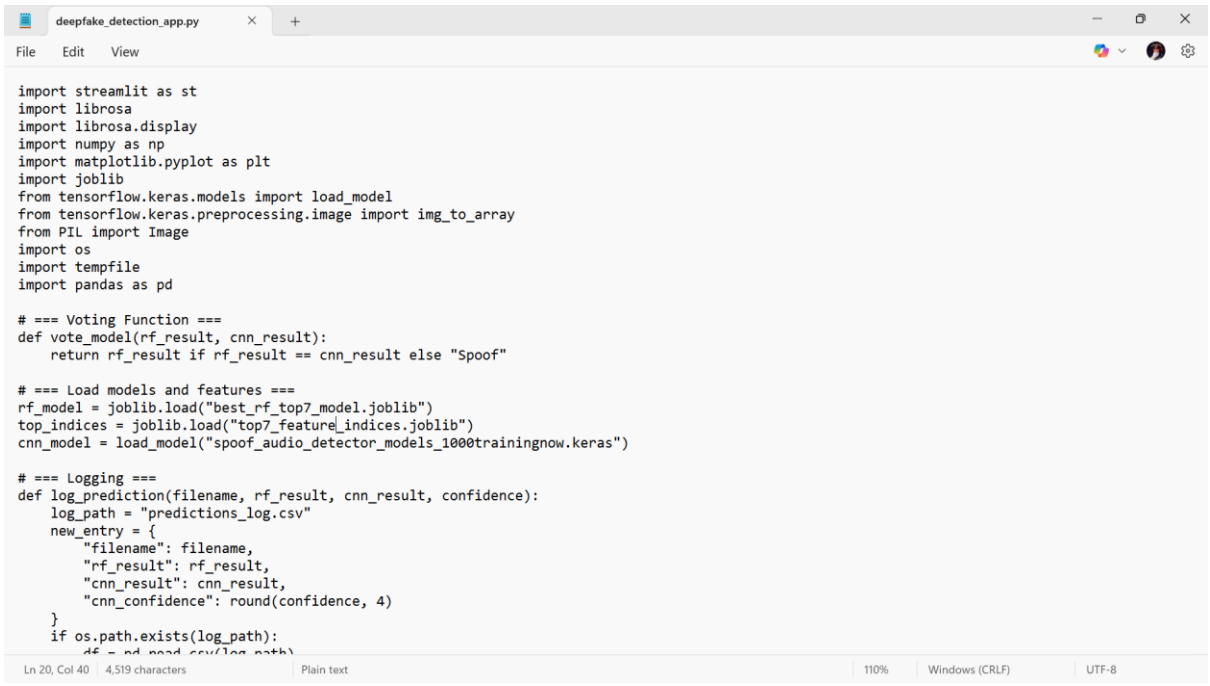

Figure 12: Final code for Deployment using Streamlit app

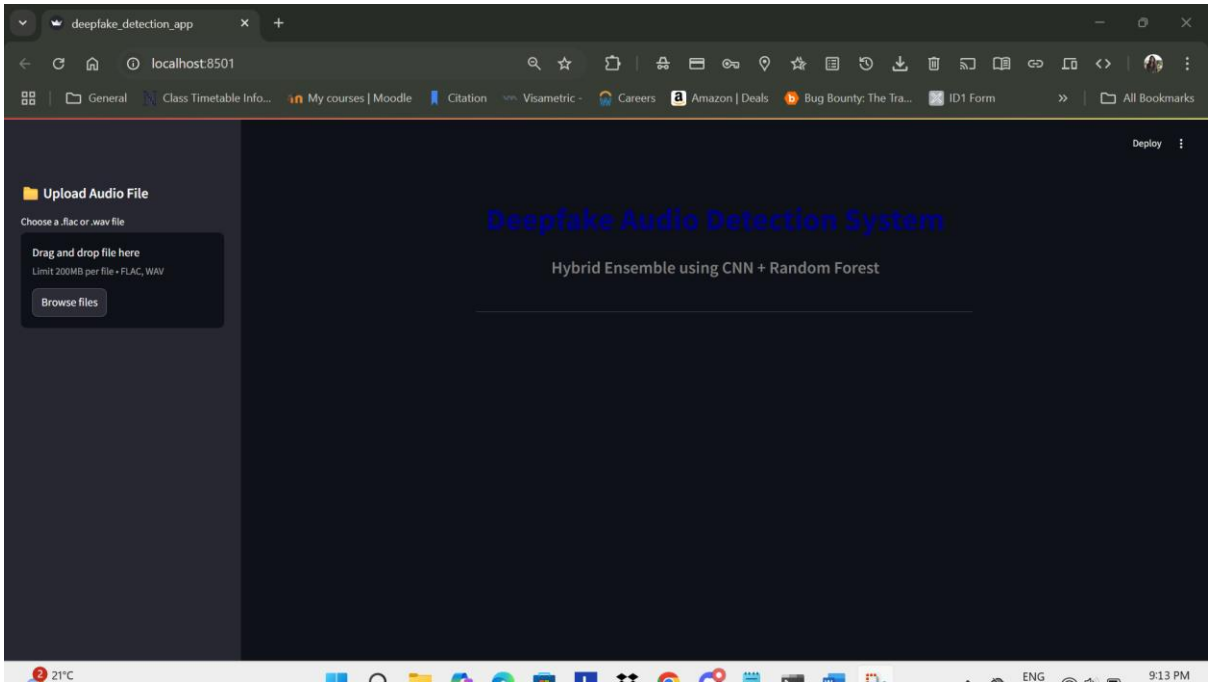

Figure 13: The Streamlit app UI.

The last implementation tool was that of Streamlit that was used to offer an interactive user interface that could detect audio deepfake. The drag-and-drop or browse option to upload the audio files takes .flac or .wav files. After uploading, any audio file is then automatically processed by the hybrid ensemble model, consisting of a CNN which was trained on the mel-spectrogram of audio images and Random Forest that was trained on handcrafted audio features. The model would then show the prediction outcome such as is classifying audio as bonafide or spoof and the confidence score made by the model. Such simplified interface permits quick and practical real-time analysis that can be done by anyone without technical knowledge.

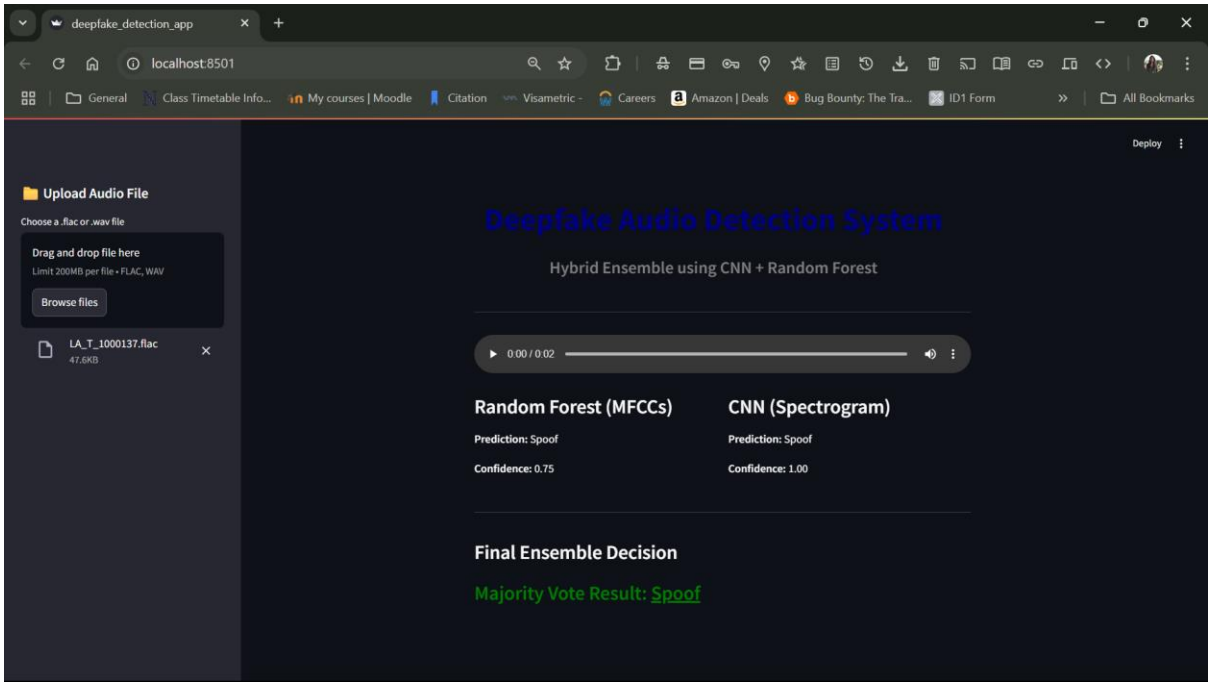

Figure 14: ***Prediction Output Screen of the Deepfake Audio Detection System***

On upload of an audio file, the system uses two separate models comprising: a Random Forest classifier based on handcraft MFCC features and a CNN based on mel-spectrogram images. All models give their own prediction (bonafide or hoax) as well as a confidence score. The results are then averaged together with a majority coalition to produce the ultimate ensemble decision. As can be seen in the example shown, both models have the audio predicted as spoof with a confidence of 0.75 (Random Forest) and 1.00 (CNN). Thus, giving it a final classification of spoof. This strategy is based on the complementary feature representations to boost the detection performance as well as reliability.

6 Testing and Validation

After deploying the app before giving this project a conclusion I tested my app on a few audio samples. And below are the results of those audio samples.

Audio File : LA_T_1001074
Actual Result: Spoof
Predicted result by Random Forest Model: Spoof
Predicted result by CNN Model: Spoof
Final Conclusion by both models: Spoof

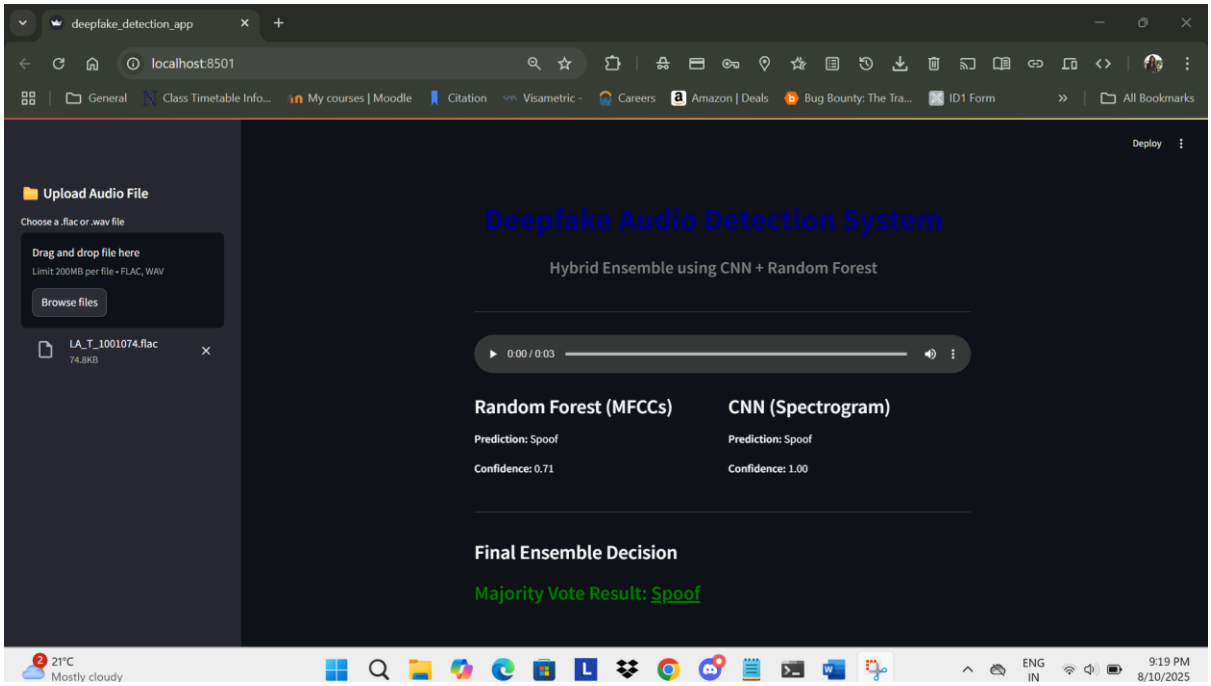

Figure 15: Predition of LA_T_1001074 File

Test 2: LA_T_9998739
Audio File : LA_T_9998739
Actual Result: Spoof
Predicted result by Random Forest Model: Bonafide
Predicted result by CNN Model: Spoof
Final Conclusion by both models: Spoof

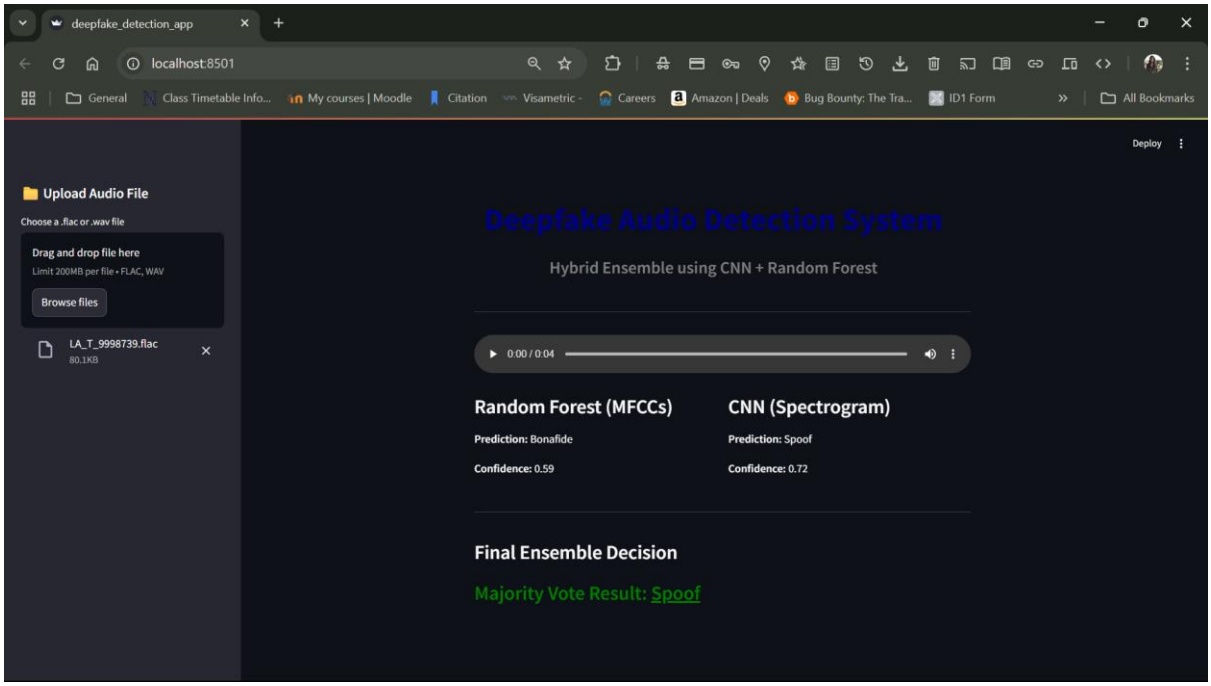

Figure 16: Test Results of LA_T_9998739.