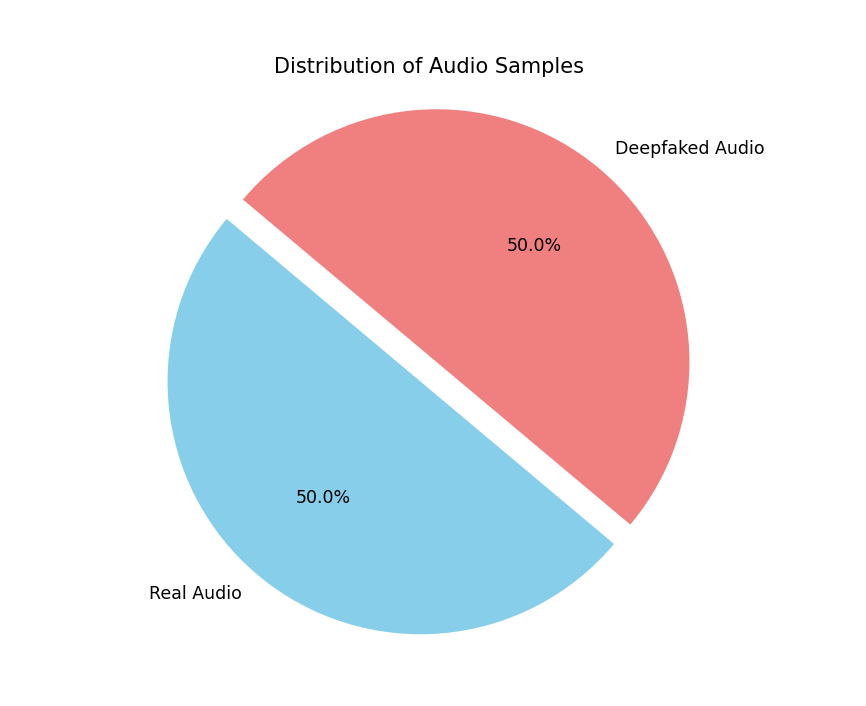
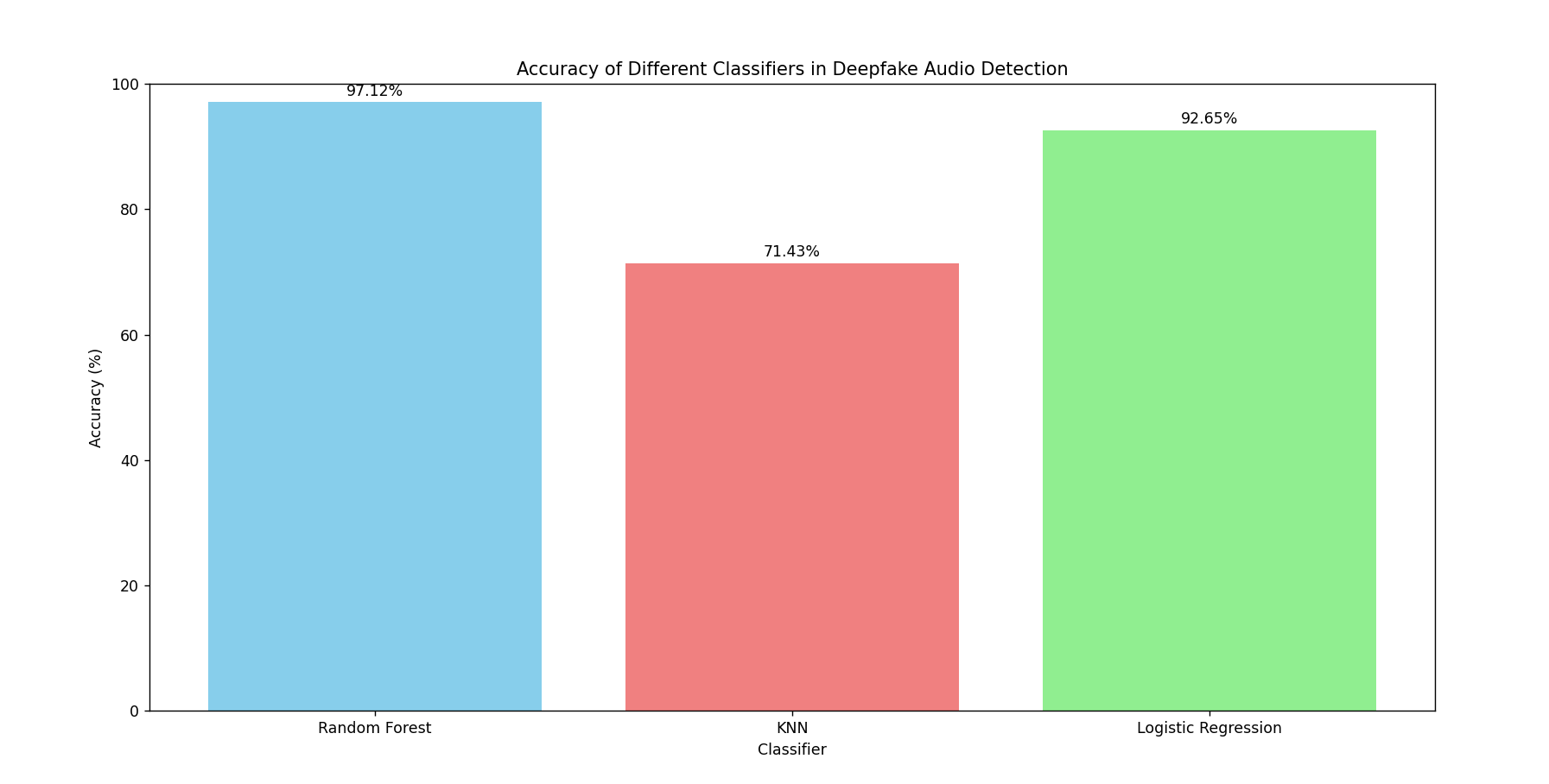
**PIE CHART :**

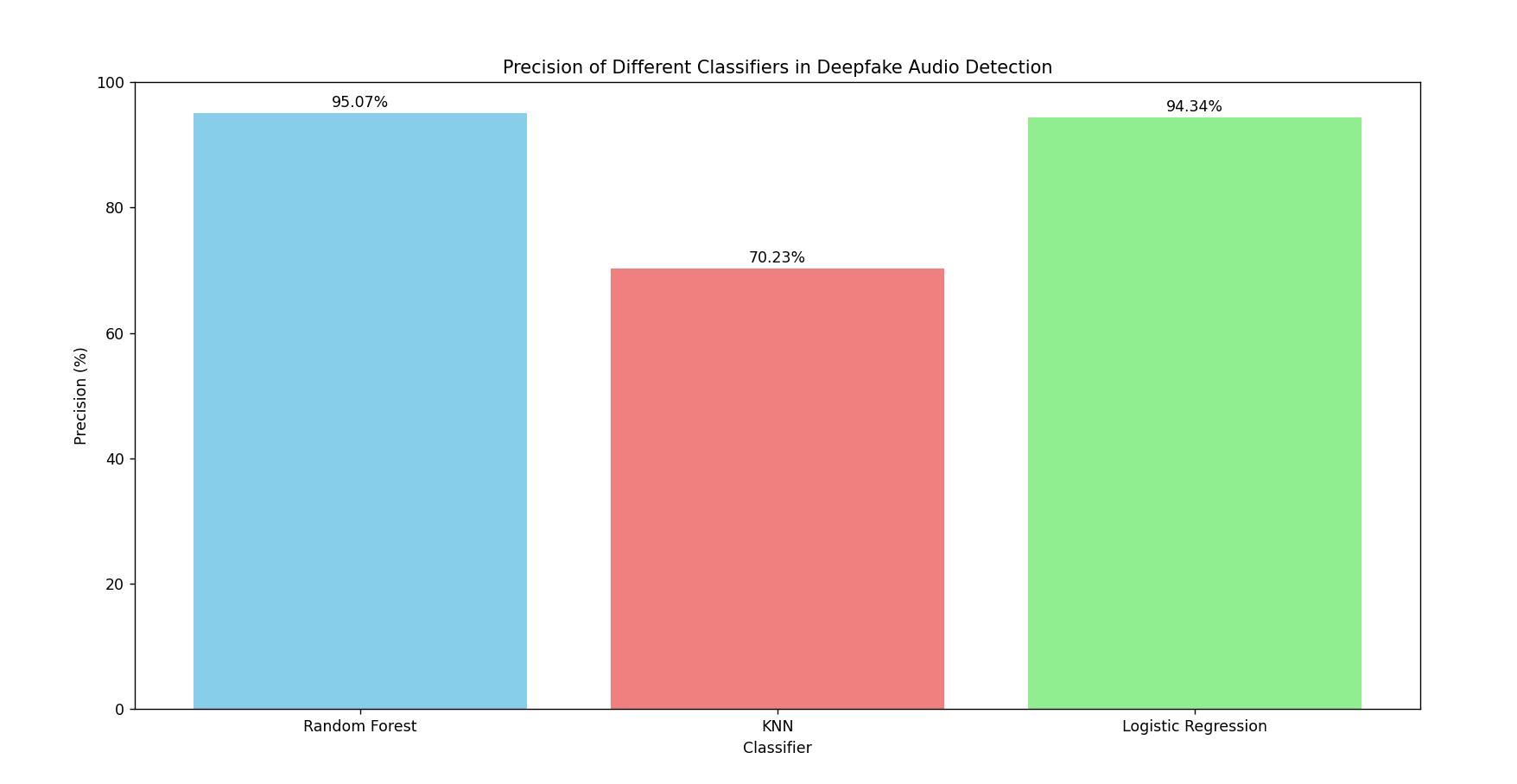


**BAR GRAPHS :**

*Accuracy :*



*Precision :*



*Recall :*

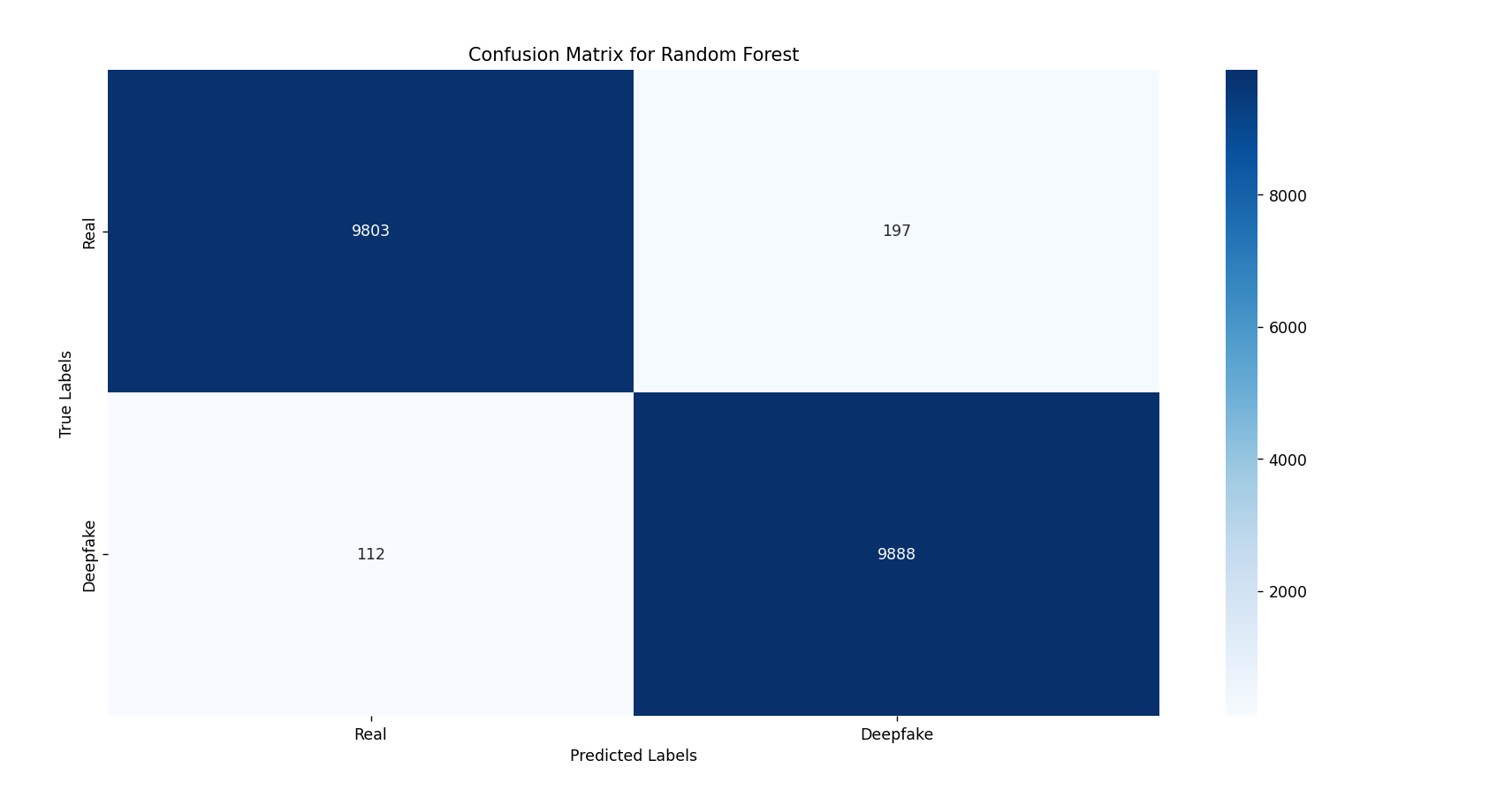
A graph showing different colored rectangles

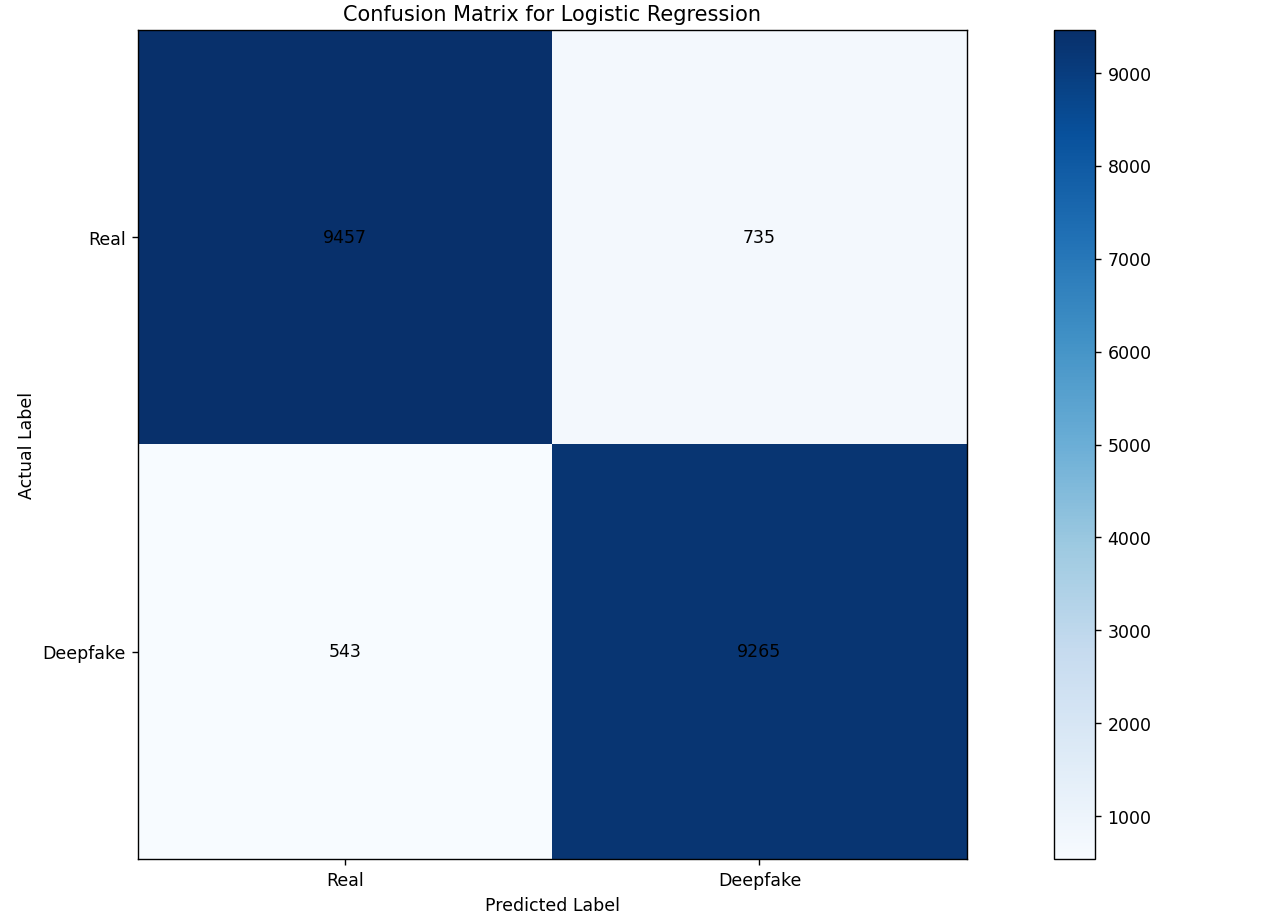
Description automatically generated

A graph of different classifiers

Description automatically generated

**CONFUSION MATRIX :**





A diagram of blue squares

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | F1 Score | Recall |
| Logestic Regression | 92.65 | 94.34 | 89.47 | 81.32 |
| KNN | 71.43 | 70.23 | 68.69 | 91.04 |
| Random Forest | 97.12 | 95.07 | 96.47 | 90.01 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int | No | PRI | NULL | Auto\_increment |
| caller\_number | varchar(15) | Yes |  | 7416622564 |  |
| receiver\_number | varchar(15) | No |  | NULL |  |
| call\_type | enum(‘incoming’, ‘outgoing’) | No |  | NULL |  |
| call\_duration | time | No |  | NULL |  |
| call\_timestamp | timestamp | Yes |  | Current\_timestamp | DEFAULT\_GENERATED |
| deepfake | tinyint(1) | No |  | NULL |  |
| contact\_name | varchar(255) | Yes |  | Unknown |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int | No | PRI | NULL | auto\_increment |
| Name | varchar(255) | No |  | NULL |  |
| Number | varchar(15) | No | PRI | NULL |  |
| Block | tinyint(1) | No |  | 0 |  |
| Call\_history | text | Yes |  | NULL |  |

|  |  |
| --- | --- |
| Output | Call |
| Value | Real / Deepfake |
| Type | Bool |
| Size | Audio file size |
| Others | Audio record path Bit Rate Duration Sample rate |

**ALGORITHM :**

Start the VeriCall app:

1. Initialize the application.

2. Load the trained deepfake detection model from the joblib file.

3. Define a function to validate the presence of a phone number in the contacts database.

4. Define a function to initiate a call:

a. Prompt the user to input a phone number.

b. Validate if the provided phone number exists in the contacts database.

c. If the phone number is found, display the associated contact name; otherwise, show "Unknown".

d. Initiate the call by dialing the phone number.

e. Select a random audio file to play.

f. Utilize the deepfake detection model to predict whether the audio is genuine or a deepfake.

g. Record call details, including contact name, phone number, deepfake prediction, and timestamp.

5. Define a function to handle incoming calls:

a. Simulate an incoming call from a randomly selected phone number in the database.

b. If the incoming phone number is present in the contacts, display the corresponding contact name; otherwise, show "Unknown".

c. Play a randomly chosen audio file.

d. Employ the deepfake detection model to predict if the audio is a deepfake or genuine.

e. Record call information, encompassing contact name, phone number, deepfake prediction, and timestamp.

6. Define a function to log call details:

a. Record pertinent information such as contact name, phone number, deepfake prediction, and timestamp.

7. Implement the main menu loop:

a. Present the user with options: Make a call, Receive a call, Call history, and Exit.

b. If the user opts to make a call:

i. Invoke the make\_call function.

c. If the user chooses to receive a call:

i. Invoke the receive\_call function.

d. If the user selects to view call history:

i. Display recorded call information.

e. If the user decides to exit, terminate the program.

End the VeriCall app.

**PSEUDO CODE :**

# Start: Import necessary libraries

import joblib

import random

# Step 1: Load the trained model from the joblib file

model = joblib.load('trained\_model.joblib')

# Step 2: Define a function to check if the phone number is in contacts

def check\_contact(phone\_number):

"""

Check if the phone number is in the contacts database.

Args:

phone\_number (str): The phone number to be checked.

Returns:

str: The contact name if found, else 'Unknown'.

"""

return contacts.get(phone\_number, 'Unknown')

# Step 3: Define a function to make a call

def make\_call(phone\_number):

"""

Make a call to the given phone number.

Args:

phone\_number (str): The phone number to call.

"""

contact\_name = check\_contact(phone\_number)

print("Initiating call...")

# Step 4: Play a random audio

selected\_audio = random.choice(audio\_files)

# Step 5: Predict if the audio is deepfake or real

prediction = model.predict(selected\_audio)

# Step 6: Record call information including deepfake result and timestamps

record\_call(contact\_name, phone\_number, prediction)

# Step 7: Define a function to receive a call

def receive\_call():

"""

Simulate receiving a call from a random phone number.

"""

random\_phone\_number = random.choice(contacts.keys())

contact\_name = check\_contact(random\_phone\_number)

print("Incoming call from:", contact\_name)

# Step 8: Play a random audio

selected\_audio = random.choice(audio\_files)

# Step 9: Predict if the audio is deepfake or real

prediction = model.predict(selected\_audio)

# Step 10: Record call information including deepfake result and timestamps

record\_call(contact\_name, random\_phone\_number, prediction)

# Step 11: Define a function to record call information

def record\_call(contact\_name, phone\_number, prediction):

"""

Record call details including contact name, phone number, deepfake result, and timestamps.

Args:

contact\_name (str): The name of the contact.

phone\_number (str): The phone number involved in the call.

prediction (str): The prediction result of the call audio.

"""

call\_history.append({

"Contact Name": contact\_name,

"Phone Number": phone\_number,

"Deepfake Prediction": prediction,

"Timestamp": current\_timestamp()

})

# Step 12: Main menu loop

while True:

print("1. Make a call")

print("2. Receive a call")

print("3. Call history")

print("4. Exit")

choice = input("Enter your choice: ")

if choice == "1":

phone\_number = input("Enter the phone number: ")

make\_call(phone\_number)

elif choice == "2":

receive\_call()

elif choice == "3":

print("Call history:")

for call in call\_history:

print(call)

elif choice == "4":

break

else:

print("Invalid choice. Please try again.")