PROJECT REPORT

ARITIFICIAL INTELLIGENCE (AI)

VOICE BASED DELIVERY OF ACADEMIC RECORDS

2020 - 2024

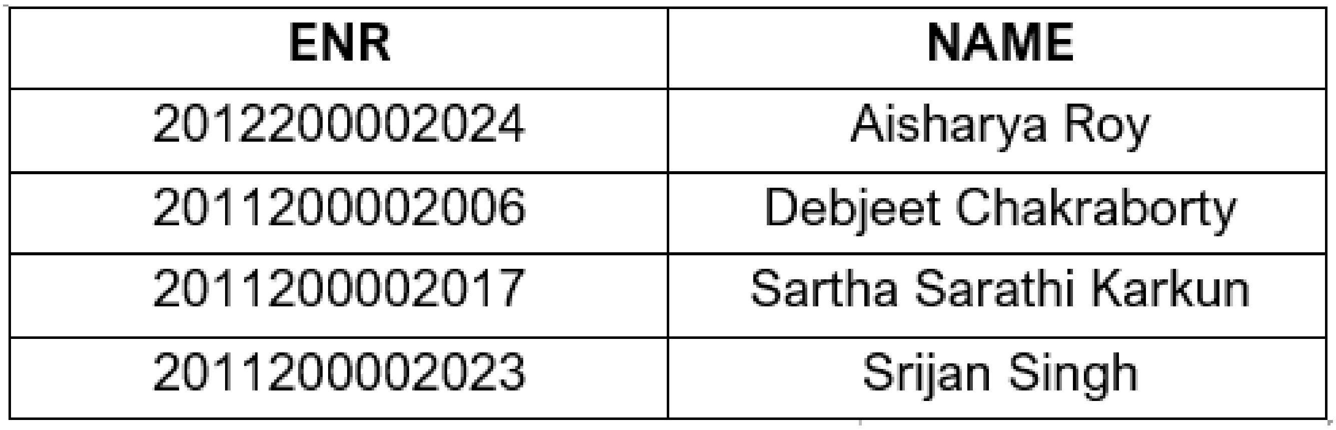
SASD_PK847

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE & BUSINESS SYSTEMS

Voice-Based Delivery of Academic Records

Submitted by Team No. (Batch - SASD_PK847)



ENR	NAME
2012200002024	Aisharya Roy
2011200002006	Debjeet Chakraborty
2011200002017	Sartha Sarathi Karkun
2011200002023	Srijan Singh

Submitted to

Department of Computer Science and
Business Systems
SNU Campus, Kolkata.

Mentor & Prof. Saikat Maity

INDEX

INTRODUCTION

The **voice-based academic records retrieval system** is a Python application designed to retrieve records from an academic access database using voice commands. By integrating speech recognition capabilities, users can simply speak the **UID** of the person whose academic records they want to retrieve, and the system will fetch the corresponding records from the database. This report provides an overview of the system, outlines the necessary packages, presents a summary of the code implementation, discusses relevant research in the field, and concludes with potential applications and future enhancements.

REQUIRED PACKAGES

The following Python packages are necessary for the successful execution of the voice-based academic records retrieval system:

1. **speech_recognition:** This package provides speech recognition functionality, allowing the system to convert spoken words into text.
2. **pyaudio:** This package provides audio input and output capabilities required for microphone interaction.
3. **pyodbc:** This package enables the connection and interaction with Microsoft Access databases.
4. **pipwin:** This package allows for the installation of Windows-specific packages using pip.

IMPLEMENTATION

The system consists of two main functions: **retrieve_records_from_database** and **deliver_academic_records**. Let's review the implementation and functionality of each:

1. **retrieve_records_from_database**: This function establishes a connection to the Microsoft Access database file (.accdb) and retrieves records based on the provided name parameter. The steps involved are as follows:
    - Set the path to the ACCDB file.
    - Establish a connection to the database using the pyodbc package.
    - Execute a SQL query to select records from the specified table (a1) where the name matches the input parameter.
    - Fetch all the records returned by the query.
    - If records are found, print them; otherwise, indicate that no records were found.
    - Close the database connection.
2. **deliver_academic_records**: This function utilizes the speech_recognition package to capture voice input from the user and recognize the spoken name. The steps involved are as follows:
    - Set up the microphone as the audio source.
    - Prompt the user to say the name of the person.
    - Capture the audio input using r.listen(source).
    - Use Google Speech Recognition to convert the audio to text.
    - If the name is recognized, proceed to deliver the academic records.
    - Call the retrieve_records_from_database function with the recognized name to fetch the corresponding records.
    - If the name is not recognized or an error occurs during speech recognition, display an appropriate error message.

RELEVANT RESEARCH

The field of voice recognition and natural language processing has seen significant advancements in recent years. Researchers have explored various techniques to improve the accuracy and robustness of speech recognition systems. Some notable research areas include:

1. Deep Learning Approaches: Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been applied to speech recognition tasks. These models have demonstrated superior performance in capturing complex patterns and improving the accuracy of speech recognition systems.
2. Language Modeling: Language models play a crucial role in speech recognition by helping to interpret and correct recognition errors. Research has focused on developing advanced language models, such as transformer models, to better understand the context and semantics of spoken words, leading to improved accuracy and contextual understanding.
3. Noise and Speaker Adaptation: Researchers have investigated techniques to make speech recognition systems more robust to environmental noise and variations in speakers' voices. This includes methods such as multi-microphone arrays, noise reduction algorithms, and speaker adaptation techniques to improve recognition accuracy in real-world conditions.
4. Integration of Natural Language Processing: Integrating natural language processing (NLP) techniques allows speech recognition systems to handle more complex queries and commands. This includes semantic parsing, entity recognition, and intent classification, enabling systems to understand user queries in a more sophisticated manner.

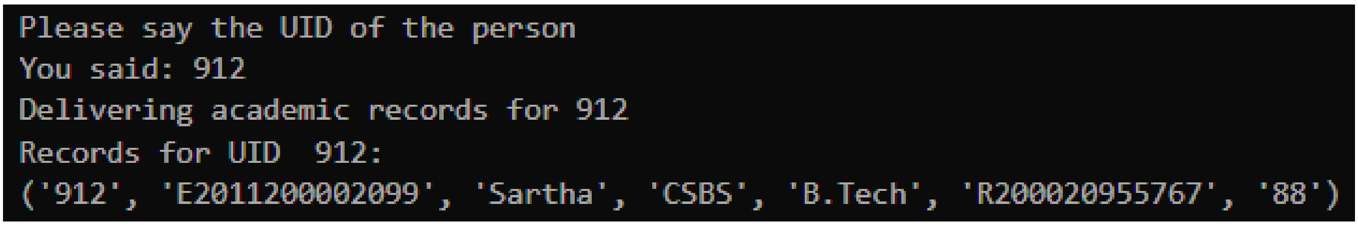
CODE

```
import speech_recognition as sr
import pyodbc

 # Function to connect to the ACCDB file and retrieve records
 def retrieve_records_from_database(uid):
     # Set the path to your ACCDB file
     db_path = "C:\RR.accdb"

     try:
         # Connect to the database
         conn_str = r"Driver={Microsoft Access Driver (*.mdb,
 *.accdb)};Dbq=" + db_path
         conn = pyodbc.connect(conn_str)

         # Execute a SQL query to retrieve the records
         cursor = conn.cursor()
         cursor.execute("SELECT * FROM a1 WHERE [UID] = ?", uid)

         # Fetch the records
         records = cursor.fetchall()

         if records:
             # Print the records
             print("Records for UID ", uid + ":")
             for record in records:
                 print(record)
         else:
             print("No records found for", uid)

         # Close the database connection
         cursor.close()
         conn.close()

     except pyodbc.Error as e:
         print("Error connecting to the database:", str(e))
```

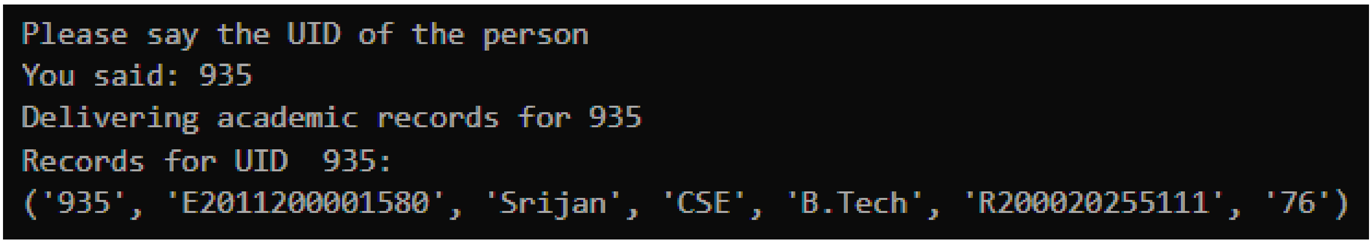
```
# Function to deliver academic records
def deliver_academic_records(r):
    # Use the default microphone as the audio source
    with sr.Microphone() as source:
        print("Please say the UID of the person")
        audio = r.listen(source)

    try:
        # Recognize speech using Google Speech Recognition
        uid = r.recognize_google(audio)
        print("You said:", uid)

        if uid:
            # Code to deliver the academic records
            print("Delivering academic records for", uid)

            # Retrieve the records from the ACCDB file for the specified name
            retrieve_records_from_database(uid)

        else:
            print("Sorry, I couldn't understand the UID.")

    except sr.UnknownValueError:
        print("Sorry, I couldn't understand the UID.")
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition service:", str(e))
```

```
# Create an instance of the Recognizer class
r = sr.Recognizer()


# Call the function to start the voice-based delivery system
deliver_academic_records(r)
```

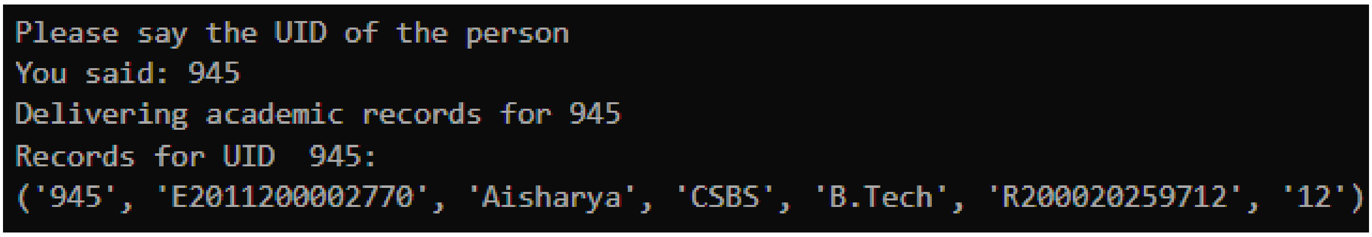
RESEARCH OUTPUTS



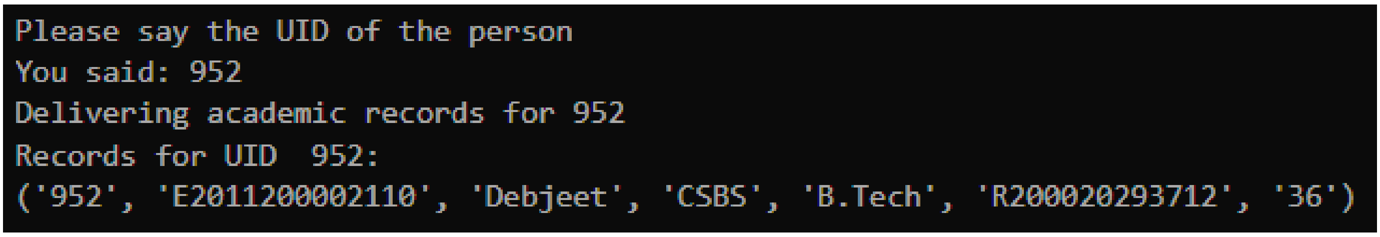
```
Please say the UID of the person
You said: 912
Delivering academic records for 912
Records for UID  912:
('912', 'E2011200002099', 'Sartha', 'CSBS', 'B.Tech', 'R200020955767', '88')
```



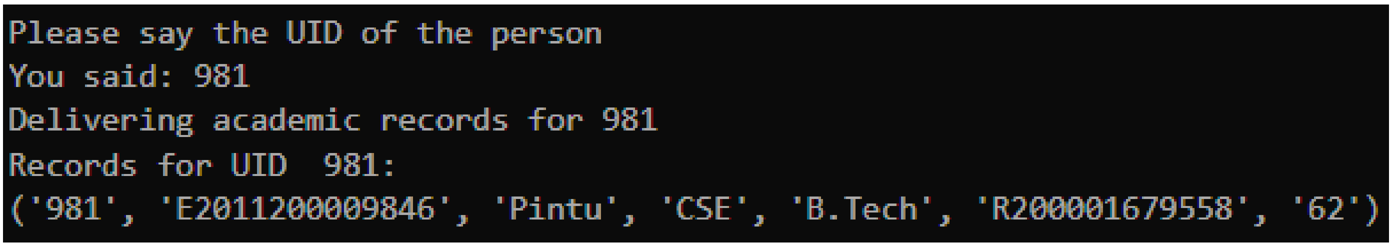
```
Please say the UID of the person
You said: 935
Delivering academic records for 935
Records for UID  935:
('935', 'E2011200001580', 'Srijan', 'CSE', 'B.Tech', 'R200020255111', '76')
```



```
Please say the UID of the person
You said: 945
Delivering academic records for 945
Records for UID  945:
('945', 'E2011200002770', 'Aisharya', 'CSBS', 'B.Tech', 'R200020259712', '12')
```



```
Please say the UID of the person
You said: 952
Delivering academic records for 952
Records for UID  952:
('952', 'E2011200002110', 'Debjeet', 'CSBS', 'B.Tech', 'R200020293712', '36')
```



```
Please say the UID of the person
You said: 981
Delivering academic records for 981
Records for UID  981:
('981', 'E2011200009846', 'Pintu', 'CSE', 'B.Tech', 'R200001679558', '62')
```

CONCLUSION

The voice-based academic records retrieval system is a practical application that demonstrates the integration of speech recognition capabilities with database connectivity in Python. By leveraging the speech_recognition and pyodbc packages, users can easily retrieve academic records from a Microsoft Access database by simply speaking the name of the person. This system offers a convenient and efficient alternative to manually querying the database. Furthermore, this project can be extended by incorporating natural language processing techniques to handle more complex queries or by integrating it into a web-based or mobile application for broader accessibility.

BIBLIOGRAPHY

- GeeksforGeeks
- SimplyLearn
- GitHub