Kalyani Government Engineering College

Affiliated to

Maulana Abul Kalam Azad University of Technology

Department of Computer Application

Kalyani – 741235, Nadia, West Bengal

Project report on

Door Unlock System using Voice Authentication

Submitted by

Niladri Shekhar Dutta	10271021012
Arijit Ghosh	10271021019
Sutapa Maji	10271021034

Under the guidance of

Mrs. Arpita Nath (Assistant Professor)

Phone: 25826680 (PBX)
Fax : 25821309

কল্যাণী – ৭৪১ ২৩৫
নদিয়া, পশ্চিমবঙ্গ

Email : contact@kgec.edu.in

Kalyani – 741 235
Nadia, West Bengal, India

পত্রাঙ্ক/Ref. No.:
তারিখ/Date :

কল্যাণী গভঃ ইঞ্জিনিয়ারিং কলেজ

Kalyani Government Engineering College

(Govt. of West Bengal)

CERTIFICATE OF APPROVAL

This certifies that the project report titled "Door Unlock System using Voice Authentication" is a record of project work completed by the following students of Kalyani Government Engineering College:

1. Niladri Shekhar Dutta (Roll: 10271021012)
2. Arijit Ghosh (Roll: 10271021019)
3. Sutapa Maji (Roll: 10271021034)

The project work was carried out under the guidance of Mrs. Arpita Nath and was submitted as a requirement for the partial fulfillment of the Degree of Master of Computer Application from Maulana Abul Kalam Azad University of Technology (MAKAUT) for the 2nd year 4th semester examination in the subject “Major Project and Viva-voce (MCAN-482)” for the academic year 2022-23.

Head of Department Department of Computer Application Kalyani Government Engineering College	Supervisor Department of Computer Application Kalyani Government Engineering College

Examiner

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to our principal, Dr. Sourabh Kumar Das, our Head of Department, Dr. Surya Sarathi Das, and our project supervisor, Mrs. Arpita Nath, for providing us with the opportunity to work on the “Door Unlock System using Voice Authentication” project. Their guidance and support enabled us to conduct extensive research and learn about new technologies.

I would also like to extend my gratitude to all the faculty members who contributed to the successful completion of this project within the given timeframe.

Finally, I am grateful to our friends and family for their unwavering support throughout this project.

DECLARATION

We, the authors of this project, affirm that the content presented herein is our original work and that we have not plagiarized any content or ideas from other sources. We have adhered to all principles of academic honesty and integrity, and we have conducted thorough research to ensure that all information presented in this project is accurate and reliable.

We understand the importance of upholding the highest standards of academic honesty and integrity, and we are committed to demonstrating these values throughout the implementation of this project. We recognize that academic dishonesty undermines the credibility and integrity of the academic community and can have severe consequences for individuals, institutions, and society as a whole.

As such, we pledge to continue to uphold these principles and to act with honesty, integrity, and transparency in all our academic and professional endeavors related to this project. We will always attribute credit where credit is due, and we will acknowledge and cite all sources appropriately. We also pledge to hold ourselves and others accountable for upholding these principles, and to work towards creating a culture of academic honesty and integrity in all our academic and professional communities related to this project.

ABSTRACT

This project presents a secure and user-friendly door unlock system based on voice authentication. The traditional methods of using keys or codes for door unlocking are prone to security risks and inconvenience. To address these challenges, we propose a system that utilizes deep learning techniques, specifically TensorFlow and Keras, to build a robust model capable of accurately identifying individuals based on their voice. The system involves audio input, preprocessing, feature extraction using the Mel-Frequency Cepstral Coefficients (MFCC) technique, and training of the deep learning model. Additionally, the project incorporates the argparse library to provide separate functionalities for regular users and administrators, with the latter having privileges for managing user accounts and system configuration. To enhance the user experience, the system integrates the Google Text-to-Speech service, which delivers audio-based feedback for clear instructions and notifications during the authentication process. Through extensive evaluation, the system demonstrates high accuracy, speed, security, and usability, offering a convenient and reliable solution for door unlocking by leveraging the power of deep learning and voice biometrics.

TABLE OF CONTENTS

INTRODUCTION

With the increasing need for security systems in residential and commercial areas, the development of a reliable and user-friendly door unlock system has become a necessity. Traditional methods of using keys or codes can be inconvenient and easily compromised, while biometric authentication such as fingerprints or iris scans can be expensive and not accessible to everyone.

In this project, we propose a door unlock system that uses voice authentication to grant access to authorized individuals. The system takes advantage of the unique characteristics of human voice, which can be used as a reliable biometric identifier.

To implement this system, we use the Mel-frequency cepstral coefficients (MFCC) technique to extract 13 features from the recorded audio data. We then derive the first order and second order derivatives of these features, resulting in a total of 39 features. These features are then used to train a deep learning model that can accurately predict the identity of a person based on their voice.

The proposed system has several advantages over traditional methods of door unlocking. Firstly, it eliminates the need for physical keys or codes, which can be lost, stolen, or forgotten. Secondly, it provides a more convenient and user-friendly experience as users can easily unlock the door by speaking their passphrase. Lastly, it provides a higher level of security as voice authentication is difficult to spoof, making it more reliable than traditional methods.

In this report, we will discuss the details of the proposed system, including the data preprocessing, feature extraction, and deep learning model training. We will also present the results of our experiments and evaluate the performance of the system in terms of accuracy and speed. Overall, this project aims to provide a more secure and convenient method for door unlocking, using the power of voice biometrics and deep learning.

REQUIREMENTS

1) Hardware Requirements
   - Microphone: A microphone with good quality that can capture clear voice samples.
   - Speaker: A speaker with good quality that can provide feedback to the user.
   - Computer: A computer that can handle the processing required for the system.
   - Arduino board: To control the servo motor and receive input from the laptop.
   - Servo motor: To unlock the door when authorized.
2) Software Requirements
   - Python programming language: To write the code for the system.
   - Deep learning framework: A deep learning framework such as TensorFlow for model training and prediction.
   - MFCC library: A library such as librosa for calculating Mel-Frequency Cepstral Coefficients (MFCC) to extract voice features.
   - Filesystem: A filesystem to store the voice samples.
   - Database: A database to store user details.
3) Functional Requirements
   - Internet: A reliable and fast internet connection for sending emails and giving training to the model.
   - User authentication: The system should authenticate the user's voice sample and grant access if the user is authorized.
   - Voice recording: The system should be able to record the user's voice sample for feature extraction and training purposes.
   - Feature extraction: The system should extract MFCC features from the recorded voice sample and derive the first and second order derivatives.
   - Deep learning model: The system should use a deep learning model to predict the user's identity based on the extracted features.
   - User management: The system should allow the administrator to add, remove user accounts.
   - Feedback: The system should provide feedback to the user to confirm successful authentication or notify them of an error.
4) Non-functional Requirements
   - Security: The system should be designed with security in mind to prevent unauthorized access or data breaches.
   - Accuracy: The system should have a high accuracy rate for voice authentication to minimize false positives or negatives.
   - Usability: The system should be user-friendly and easy to use, with clear instructions and feedback.
   - Performance: The system should be able to handle multiple requests in real-time without any significant delay.

DESIGN

AUDIO INPUT

The audio input component of the proposed system is responsible for capturing audio signals from the user's voice and converting them into a digital format that can be processed by the system. To accomplish this, we have used a microphone that is connected to the computer. The microphone captures the audio signals and converts them into an analog voltage signal, which is then amplified and converted into a digital signal using an analog-to-digital converter (ADC).

Once the audio signals are captured, they are transmitted to the preprocessing component of the system for further processing. The audio input component plays a crucial role in the system, as it is the first step in the process of identifying the user's voice and granting them access to the system.

PREPROCESSING

The preprocessing component of the proposed system is responsible for cleaning and filtering the audio data to remove noise and enhance the signal quality. In our system, we have employed several techniques to preprocess the audio data, including noise removal, silence removal, normalization, pitch shifting, and time stretching.

To remove unwanted noise from the audio data, we have used a noise reduction algorithm that takes separate noise signal to reduce the noise from the original signal. This helps to remove ambient noise and other unwanted sounds that can interfere with the user's voice.

We have also employed silence removal to remove any periods of silence in the audio data. This is done by detecting periods of low energy in the audio signal and removing them from the recording.

To ensure that the audio data is consistent and uniform, we have normalized the audio data by scaling it to a standard range of values. This helps to ensure that the audio data is consistent across different recordings and users.

Overall, the preprocessing component plays a critical role in ensuring that the audio data is of high quality and suitable for input into the deep learning model.

FEATURE EXTRACTION

Mel-Frequency Cepstral Coefficients (MFCCs) are widely used in speech and audio signal processing as a feature extraction technique. They are derived from the short-time Fourier transform of the audio signal and aim to capture the characteristics of the human auditory system. MFCCs are computed by first converting the audio signal from the time domain to the frequency domain using a Fourier transform, and then mapping the frequency scale to the Mel scale, which is a non-linear frequency scale that approximates the human auditory system's perception of frequency. Next, the logarithm of the Mel power spectrum is computed and transformed using the Discrete Cosine Transform (DCT) to obtain a set of cepstral coefficients.

In the feature extraction stage, we extracted a total of 39 features from the preprocessed audio data. We started by extracting 13 Mel-Frequency Cepstral Coefficients (MFCC) from each audio sample. Then, we took the first and second order derivatives of these coefficients to obtain a total of 39 features. These derivatives capture the changes in the spectral content and provide additional information about the audio signal, which can improve the performance of the deep learning model. By extracting these features, we aim to capture the essential characteristics of the audio data in a compact representation that can be used as input to the deep learning model.

DEEP LEARNING MODEL

The deep learning model used in this project is a sequential neural network. It consists of an input layer with 39 units, representing the MFCC features extracted from the audio data. A Gaussian Noise layer is added for regularization, followed by a hidden layer with 30 units and a ReLU activation function. A Dropout layer is included to further prevent overfitting. An Activity Regularization layer with L1 and L2 penalties helps promote sparsity and prevent overfitting. The output layer uses softmax activation for multi-class classification. The model is trained with the sparse categorical cross-entropy loss function and the Adam optimizer for 30 epochs, with early stopping implemented. The design aims to optimize performance, enhance generalization, and mitigate overfitting.

DOOR CONTROL MECHANISM

The design of the door control system involves the utilization of Arduino and a servo motor to control the door's operation. Once authentication is confirmed, the Arduino sends a signal to the servo motor, which physically opens the door.

To ensure convenience and efficiency, the door will remain open for a predefined duration of 5 seconds. This allows the authorized user sufficient time to enter the premises without the need for additional authentication. After the 5-second interval, the Arduino triggers the servo motor to close the door automatically, ensuring the security of the premises.

The integration of Arduino and the servo motor provides a reliable and cost-effective solution for controlling the door. It enables seamless interaction between the authentication process, the physical door mechanism, and the timing functionality. Additionally, the Arduino platform offers flexibility for further enhancements and customizations to meet specific requirements.

By combining voice authentication, Arduino, and a servo motor, the door control system achieves a streamlined and secure access control process. It provides a user-friendly experience while maintaining the necessary security measures.

USER INTERFACE

In the user interface part of the system, we have provided two options for the user to run the program. If the user runs the program normally, they will be able to use their voice to unlock the door. However, if three continuous failed attempts occur, the system will lock, and a password will be required to unlock both the system and the door. In addition to that, an email will be sent to all the registered users for suspicious activity.

Alternatively, if the user runs the program using the --admin flag, they will be prompted to enter a password. Once authenticated, the user will be able to perform several administrative tasks, such as creating a new user, deleting an existing user, training the model, changing the password, and resetting the system.

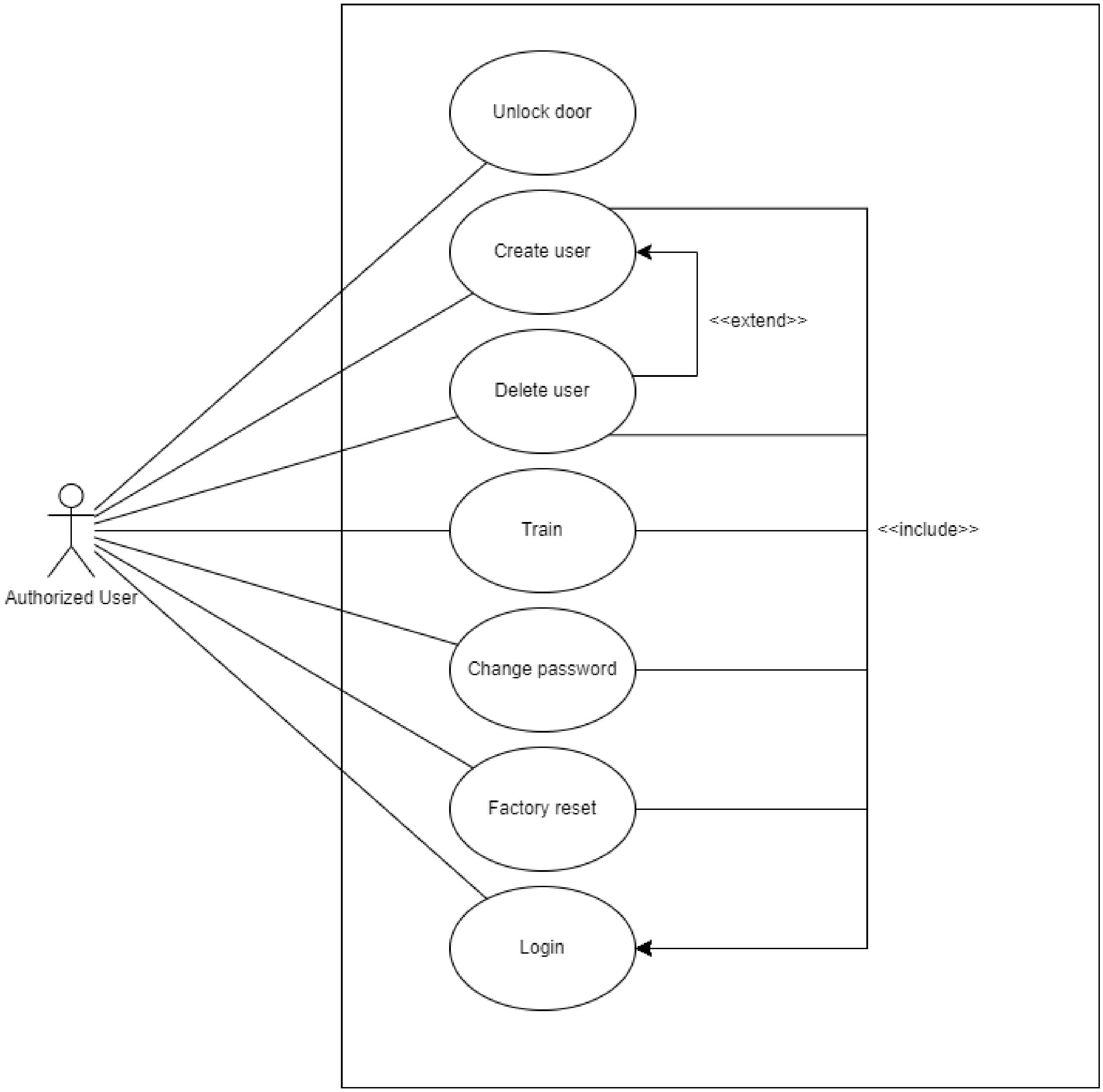
The user interface has been designed to be user-friendly and intuitive. The system provides feedback to the user through both text-based and audio-based feedback systems. Users are guided through the process of unlocking the door with their voice, and any errors or failed attempts are clearly communicated. Additionally, the admin features are only accessible with a password, providing an additional layer of security to the system.

DIAGRAMS

Diagrams are an essential tool for visualizing and communicating complex systems and their interactions. In this section, we present the diagrams used in our project, including use-case diagram, sequence diagram and deployment diagram. These diagrams provide a clear and concise representation of the various components of our system and how they interact with each other. Using diagrams not only helps in understanding the system design and architecture but also helps in identifying potential issues and making improvements.

USE CASE DIAGRAM

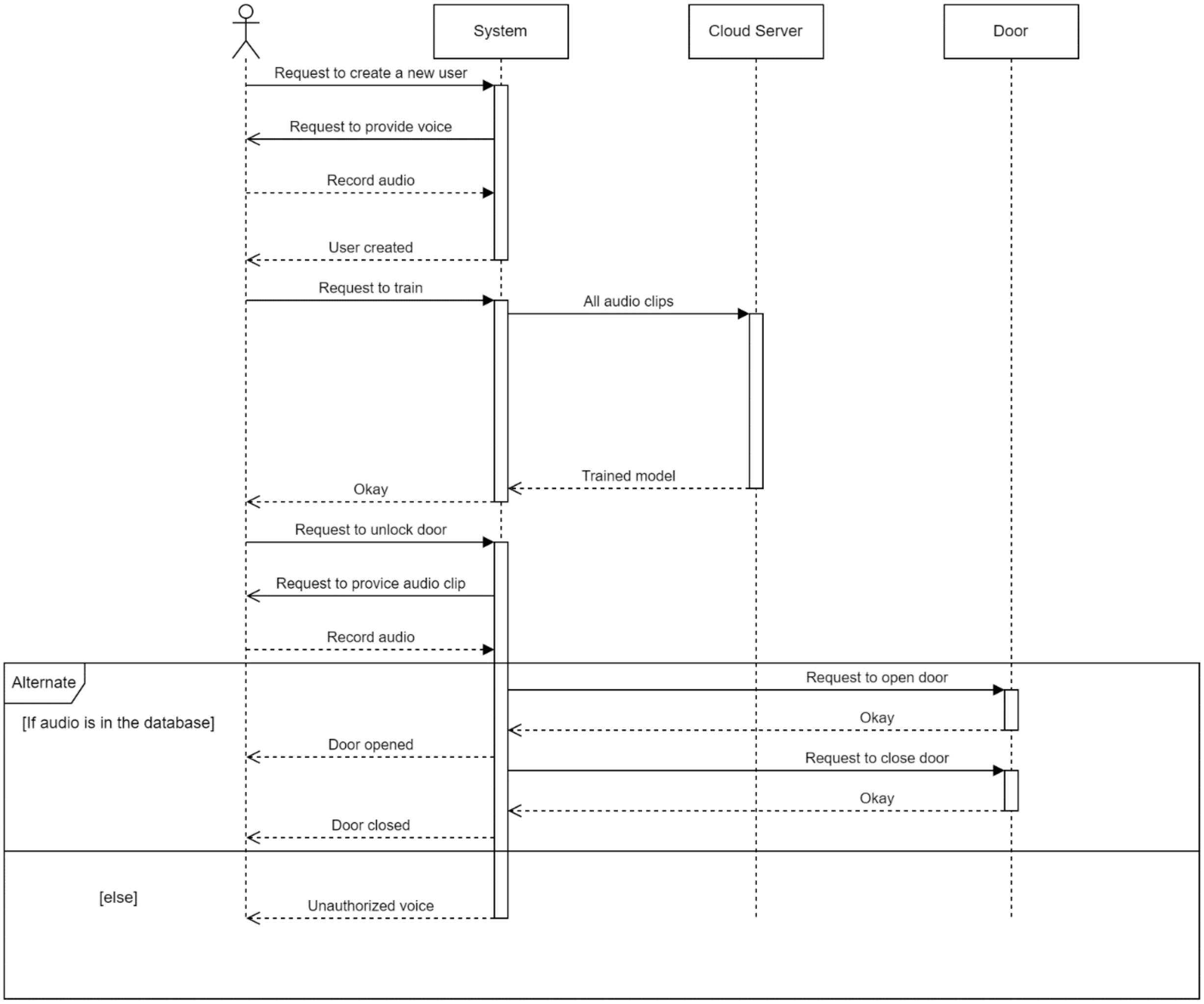
The use case diagram for the system shows the Authorized user as the only actor. The actor can perform six use cases, namely “Unlock door”, “Create user”, “Delete user”, “Train”, “Change password”, and “Factory reset”. The unlock door use case allows the user to unlock the door using their voice. The create user use case enables the user to create a new user in the system. The delete user use case allows the user to delete an existing user from the system. The train use case enables the user to train the machine learning model with new user voice samples. The change password use case allows the user to change their password for accessing the system. The factory reset use case enables the user to reset the entire system to its default state. There is an extension relationship between the create user and delete user use cases, which means that the delete user use case is an extension of the create user use case. Additionally, create user, delete user, train, change password, and factory reset shares a include relationship with login.



SEQUENCE DIAGRAM

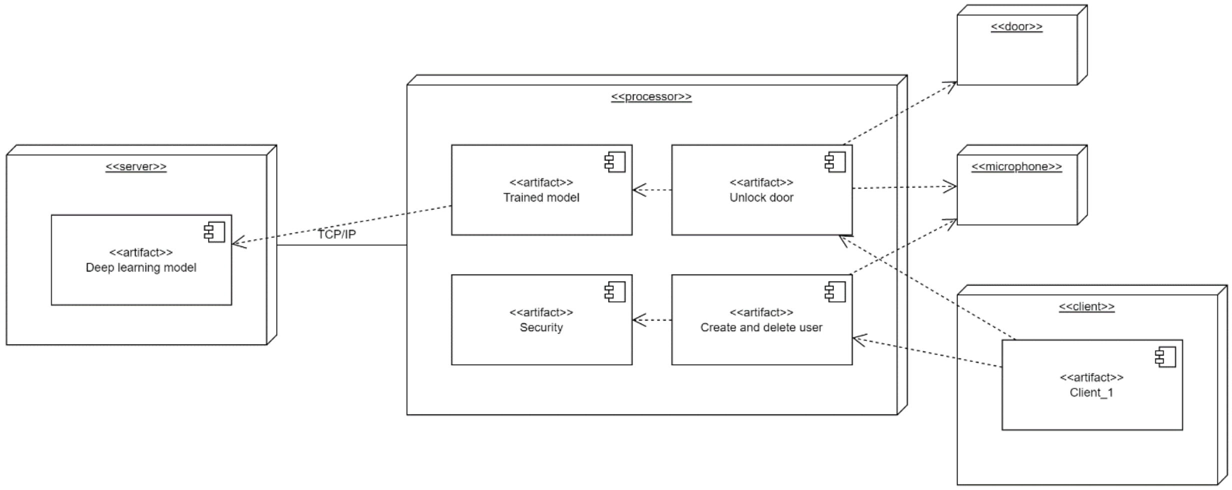
A sequence diagram is a visual representation that shows the order of interactions and messages exchanged between objects or components in a system. It helps depict the flow of communication and behavior of the system over time, highlighting the sequence of actions and method calls between entities. Sequence diagrams are useful for understanding system interactions, identifying bottlenecks, and facilitating communication and collaboration among stakeholders.

In this sequence diagram, we are showing how the admin will create a new user and train the model and use the model to predict and unlock the door.



DEPLOYMENT DIAGRAM

A deployment diagram is a visual representation that shows how software components are distributed across hardware devices in a system. It illustrates the physical deployment, connections, and relationships between nodes and components. Deployment diagrams help understand the system's architecture, identify points of failure, and assist in system planning and infrastructure design.


<<door>>
<<processor>>
<<server>>
<<artifact>> Trained model
<<artifact>> Unlock door
<<microphone>>
<<artifact>> Deep learning model
TCP/IP
<<artifact>> Security
<<artifact>> Create and delete user
<<client>>
<<artifact>> Client_1

IMPLEMENTATION

AUDIO INPUT

In the audio input subsection of the implementation section, we capture the audio input from the user using the sounddevice module. The system is designed to record the user's audio for a duration of 30 seconds, which is done twice to obtain a 1-minute audio clip. This duration provides sufficient time for the user to provide their voice sample for authentication.

The sounddevice module offers a simple and efficient way to capture audio from the microphone connected to the computer. It provides functions to start and stop audio recording, as well as options to specify the duration and sample rate of the recording.

To capture the audio input, we initialize the sounddevice with the appropriate settings, including the desired sample rate and the number of channels. We then use a loop to record the audio for the specified duration, which in this case is set to 30 seconds. By repeating the recording process twice, we obtain a 1-minute audio clip, which provides sufficient data for accurate voice authentication. This approach allows us to capture the user's voice sample and extract the necessary features for identification using the proposed deep learning model.

`env.py`

```
CHANNELS = 1
SAMPLE_RATE = 44100
```

`utils.py`

```
def record_audio(seconds: int) -> numpy.typing.NDArray:
    if sounddevice.default.samplerate is None:
        raise TypeError("Unable to determine the samplerate.")
    signal = sounddevice.rec(
        frames=seconds * env.SAMPLE_RATE,
        blocking=True
    )
    return signal.flatten()
```

`user.py`

```
def run_create_user_wizard() -> None:
    .
    .
    .
    count = 1
    signals = []
    while count <= 2:
        utils.wait_for_keypress(
            "Press any key when you are ready to record..."
        )
        utils.clear_screen_and_print_header()
        print(f"Progress: {count}/2")
        print("I am recording your beautiful voice... ", end='',
            flush=True)
        signal = utils.record_audio(30)
        print("Done.")
        choice = input("\nDo you want to keep this audio clip?
[Y/N]: ")
        if choice in ('y', 'Y'):
            signals.append(signal)
            count += 1
        else:
            print("I have discarded the audio clip.")
        print()
    .
    .
    .
    scipy.io.wavfile.write(
        filename=f"training_data/{id}.wav",
        rate=env.SAMPLE_RATE,
        data=signal
    )
```

PREPROCESSING

The preprocess function utilizes the following libraries and performs the following steps to preprocess the audio signal:

Libraries:

- noisereduce: Used for noise reduction in the audio signal.
- remove_silence: Used for removing periods of silence in the audio signal.
- librosa: Used for signal normalization.

Steps:

1. Noise Reference Extraction: If no noise signal is provided, the function extracts the last 4 seconds of the input signal as a noise reference.
2. Truncation: The function removes the last 4 seconds from the input signal to eliminate the noise component.

3. Noise Reduction: The function applies a noise reduction algorithm using the noise reference signal and the sample rate of the audio to reduce unwanted background noise.
4. Silence Removal: The function detects and removes periods of silence in the audio signal, enhancing the accuracy of feature extraction.
5. Signal Normalization: The function normalizes the signal by scaling its amplitudes to a standard range, ensuring consistent amplitudes across different audio samples.

These preprocessing steps collectively aim to enhance the quality and suitability of the audio data for subsequent processing stages in the deep learning model.

`utils.py`

```
def preprocess(signal, noise=None):
    if noise is None:
        noise = signal[-(4 * env.SAMPLE_RATE):]
        signal = signal[:-(4 * env.SAMPLE_RATE)]
    signal = noisereduce.reduce_noise(
        y=signal,
        sr=env.SAMPLE_RATE,
        stationary=True,
        y_noise=noise
    )
    signal = remove_silence.remove_silence(signal)
    signal = librosa.util.normalize(signal)
    return signal
```

FEATURE EXTRACTION

In our project, we have utilized the librosa library to extract Mel-Frequency Cepstral Coefficients (MFCC) from the preprocessed audio data. By using the librosa library, we are able to efficiently calculate the MFCCs from the preprocessed audio samples. These coefficients capture important information about the spectral content of the audio, such as the distribution of energy across different frequency bands. The MFCC features serve as a compact representation of the audio data and are suitable for input into our deep learning model. The use of librosa simplifies the process of extracting MFCC features and allows us to focus on building and training the model effectively.

`model.py`

```
def extract_features(signal):
    signal_mfcc = librosa.feature.mfcc(
        y=signal, sr=env.SAMPLE_RATE, n_mfcc=13)
    signal_mfcc_delta_1 = librosa.feature.delta(signal_mfcc)
    signal_mfcc_delta_2 = librosa.feature.delta(signal_mfcc,
order=2)
    return numpy.hstack((
        signal_mfcc.transpose(),
        signal_mfcc_delta_1.transpose(),
        signal_mfcc_delta_2.transpose()
    ))
```

DEEP LEARNING MODEL

In our project, we have utilized the TensorFlow and Keras libraries to build our deep learning model for voice authentication.

TensorFlow is a popular open-source library for numerical computation and machine learning. It provides a comprehensive set of tools and functionalities for building and training deep neural networks. We have utilized TensorFlow as the backend engine for our deep learning model.

Keras, on the other hand, is a high-level deep learning API that runs on top of TensorFlow (among other backends). It provides a user-friendly and intuitive interface for building neural networks. We have used Keras as a front-end API to construct our deep learning model. With Keras, we can easily define the architecture of our model using various layers, such as dense (fully connected) layers, activation functions, and regularization techniques.

By combining the power of TensorFlow and the simplicity of Keras, we have built an effective deep learning model for voice authentication. TensorFlow provides the underlying computational capabilities, while Keras simplifies the model-building process, allowing us to focus on designing and training our model efficiently.

`main.py (server)`

```
@server.post('/machine-learning')
def train(files: list[UploadFile]):
    .
    .
    .
    early_stop = tensorflow.keras.callbacks.EarlyStopping(
        monitor='val_loss', patience=5)
    model = tensorflow.keras.Sequential([
        tensorflow.keras.layers.Dense(
            39, input_shape=(39,), activation="relu"),
        tensorflow.keras.layers.GaussianNoise(0.2),
        tensorflow.keras.layers.Dense(30, activation="relu"),
        tensorflow.keras.layers.Dropout(0.2),
        tensorflow.keras.layers.ActivityRegularization(l1=0.01,
l2=0.01),
        tensorflow.keras.layers.Dense(len(files) + 1,
activation="softmax")
    ])
    model.compile(
        optimizer="adam",
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"]
    )
    .
    .
    .
```

DOOR CONTROL MECHANISM

The implementation of the door control mechanism involves the integration of PyFirmata, a Python library, to establish communication and control between our Python program and the Arduino board. PyFirmata enables us to send commands and receive feedback from the Arduino, allowing seamless coordination of the door control process.

In our Python program, we have implemented the voice authentication algorithm, which verifies the user's voice characteristics. Upon successful verification, the program sends a signal to the Arduino via PyFirmata to initiate the door opening sequence. Instead of abruptly opening the door, we have incorporated a gradual opening mechanism to ensure smooth and controlled movement.

After receiving the signal, the Arduino activates the servo motor, which, through its rotational movement, gradually opens the door. This gradual opening process enhances safety and prevents any sudden impact or noise that may occur with a rapid door movement.

To maintain the door open for a specified duration, we utilize a timing mechanism within our Python program. After the successful opening of the door, a timer is initiated for 5 seconds. During this time, the program continuously sends signals to the Arduino to keep the servo motor in the open position.

Once the 5-second interval elapses, the Python program sends a signal to the Arduino through PyFirmata to trigger the gradual closing sequence. The servo motor is controlled to gradually close the door, ensuring a controlled and secure operation.

By utilizing PyFirmata, our Python program seamlessly interacts with the Arduino board, allowing us to control the door's movement in a precise and synchronized manner. The gradual opening and closing mechanisms contribute to a smoother and safer user experience, while the integration of timing ensures the door remains open for the desired duration.

Overall, this implementation provides an effective and reliable solution for controlling the door using voice authentication and Arduino, with the support of PyFirmata for seamless communication and coordination between the software and hardware components.

`arduino.py`

```
import pyfirmata
import env
_board = None
def initialize() -> None:
    global _board
    _board = pyfirmata.Arduino(env.PORT)
    _board.digital[env.PIN].mode = pyfirmata.SERVO
def rotateservo(angle: int) -> None:
    global _board
    _board.digital[env.PIN].write(angle)
```

door.py

```
import Arduino
import time
def close() -> None:
    for angle in range(180, -1, -2):
        arduino.rotateservo(angle)
        time.sleep(0.001)
def open() -> None:
    for angle in range(0, 181, 2):
        arduino.rotateservo(angle)
        time.sleep(0.001)
```

main.py

```
.
.
.
door.open()
time.sleep(5)
door.close()

.
.
.
```

USER INTERFACE

In our project, we have implemented two sections using the argparse library to handle user and admin functionalities. The argparse library allows us to easily define and parse command-line arguments for our program.

For the user section, we have designed it to be the default mode. Users can interact with the system without any additional flags or parameters. They can use their voice for authentication and unlocking the door.

On the other hand, we have implemented an admin section that requires the --admin flag to be provided when running the program. This flag is used to enable administrative privileges. The admin section provides access to additional functionalities, such as creating new users, deleting existing users, training the model, changing the password, and resetting the system.

To provide feedback to the users, we have utilized the Google Text-to-Speech service. This service allows us to convert text into spoken audio, which is then played back to the users. By using audio-based feedback systems, we enhance the user experience and provide clear instructions and notifications regarding the authentication process and any errors that may occur.

main.py

```
import argparse
.
.
.
if __name__ == "__main__":
    .
    .

    .
    parser = argparse.ArgumentParser(
        description="This program authenticates users based on
previously stored audio profiles. When a user attempts to log in,
they provide a voice sample that is compared to their stored
profile using advanced signal processing and machine learning
algorithms. If the voice sample matches the profile, the user is
authenticated and granted access. This provides a secure and
difficult-to-spoof method of authentication."
    )
    parser.add_argument(
        '-a',
        "--admin",
        action="store_true",
        help="Enable admin privileges. Requires additional
authentication."
    )
    args = parser.parse_args()
    if args.admin:
        .
        .
        .
    else:
        .
        .
        .
```

utils.py

```
def text_to_speech(text):
    try:
        gtts.gTTS(text).save("speech.mp3")
        playsound.playsound("speech.mp3")
    except:
        pass
    finally:
        if os.path.exists("speech.mp3"):
            os.remove("speech.mp3")
```

LIMITATIONS

Our project has several limitations that need to be considered. Firstly, the functionality of the system heavily relies on the proper functioning of the microphone. Any malfunction or failure of the microphone can render the system non-operational, leading to inconvenience and potential access issues for users. To mitigate this, regular maintenance and monitoring of the microphone are necessary to ensure its optimal performance.

Secondly, the system may encounter difficulties in authenticating users who are unable to reproduce their previous voice accurately. Factors such as illness or changes in voice due to various reasons can affect the accuracy of authentication. To address this limitation, we have implemented a backup password mechanism to offer an alternative means of access for users who may encounter difficulties with voice authentication. This ensures that users can still access the premises with ease even if they are unable to rely solely on voice authentication.

Furthermore, it is important to note that our current project is in the prototype stage and is being developed using a computer. However, deploying a computer in the production environment may not be feasible due to factors such as cost, size, power consumption, and maintenance requirements. Therefore, further research and development will be required to design a more practical and dedicated hardware solution for deployment in real-world scenarios.

Additionally, the accuracy and reliability of the system may be affected by environmental factors such as background noise, echoes, and other acoustic conditions. Implementing appropriate noise cancellation or filtering techniques can help improve the system's performance in diverse settings.

Moreover, compatibility and integration with existing security systems or infrastructure can pose challenges. Ensuring seamless integration with other access control systems, databases, or security protocols may require additional effort and resources. Potential compatibility issues may arise during the implementation phase, which need to be addressed to ensure the system functions smoothly within the existing infrastructure.

Privacy and security concerns are also crucial considerations. Storing and protecting users' voice data requires robust encryption and stringent data protection measures to prevent unauthorized access or misuse. Regular security audits and compliance with relevant data protection regulations are necessary to mitigate these risks and ensure user privacy.

Furthermore, introducing a new authentication system may require user training and adjustment periods. Users accustomed to traditional access methods may need time to adapt to voice authentication and become familiar with the system's operation. Adequate training and clear instructions are essential to ensure a smooth transition and user acceptance.

Lastly, the system's accuracy in terms of accepting legitimate users and rejecting unauthorized individuals (false acceptance and false rejection rates, respectively) may vary depending on various factors. Fine-tuning the system's parameters and continuously monitoring its performance is crucial to maintain an optimal balance between security and user convenience.

FUTURE SCOPE

One potential avenue for future development of our project is the integration of Raspberry Pi as a platform for prototyping and deploying our system. Currently, our project is implemented and tested on a computer, but leveraging the capabilities of Raspberry Pi can offer several advantages and expand the project's practicality.

By transitioning to Raspberry Pi, we can create a more compact and portable solution that can be easily deployed in various environments. Raspberry Pi's small form factor, low power consumption, and GPIO (General Purpose Input/Output) pins make it an ideal candidate for embedded systems and Internet of Things (IoT) applications. This would allow our system to be integrated into devices or setups where a computer is not be feasible or practical.

Furthermore, Raspberry Pi provides native support for various peripherals and interfaces, enabling seamless integration with sensors, actuators, or other hardware components. This opens up possibilities for expanding the functionality of our system, such as incorporating additional sensors for real-time environmental monitoring or integrating actuators for interactive feedback.

In addition, Raspberry Pi's compatibility with popular software frameworks and libraries, including TensorFlow and Keras, ensures easy adaptation of our existing codebase. This enables a smooth transition from computer-based prototyping to Raspberry Pi-based implementation.

By embracing Raspberry Pi as the underlying platform for our project, we can enhance its versatility, portability, and potential for real-world deployment. This future development can make our system more accessible and applicable in various domains, opening doors to new use cases and opportunities for innovation.

CONCLUSION

In conclusion, our project aimed to develop a reliable and user-friendly door unlock system using voice authentication. We proposed a system that leverages the unique characteristics of human voice as a biometric identifier, providing a secure and convenient method for door unlocking.

Throughout the project, we successfully implemented and integrated various components to create a functional system. The hardware requirements included a microphone, speaker, computer, Arduino board, and servo motor. On the software side, we used Python programming language, TensorFlow deep learning framework, MFCC library, filesystem, and database to support the system's functionalities.

The system's design involved capturing audio input, preprocessing the audio data to remove noise and enhance signal quality, extracting Mel-Frequency Cepstral Coefficients (MFCC) features from the preprocessed data, training a deep learning model to predict user identity based on the extracted features, and providing a user-friendly interface for interaction and feedback.

We implemented these components effectively, utilizing sounddevice for audio input, applying noise reduction, silence removal, and normalization techniques for preprocessing, employing the librosa library for MFCC feature extraction, training a sequential neural network model with regularization techniques, and creating a user-friendly interface with text and audio-based feedback.

Our project successfully addressed the limitations of traditional door unlocking methods by eliminating the need for physical keys or codes, providing convenience and a higher level of security. Voice authentication proved to be a reliable biometric identifier, difficult to spoof, and accessible to a wide range of users.

The implemented system fulfilled the functional and non-functional requirements, ensuring security, accuracy, usability, and performance. We also presented diagrams, including a use-case diagram, sequence diagram, and deployment diagram, to visualize the system's architecture and interactions.

In conclusion, our project contributes to the development of innovative door unlock systems, leveraging voice authentication and deep learning techniques. The system offers a secure, convenient, and reliable approach to door unlocking, with potential applications in residential and commercial settings. Further improvements and optimizations can be explored to enhance the system's performance and expand its capabilities.

REFERENCE

Velardo, Valerio. Valerio Velardo - The Sound of AI. “Audio Signal Processing for Machine Learning” *YouTube*, uploaded by Valerio Velardo, https://youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0