**SnoreScan: A Smart App for Sleep Apnea Detection**

by

**Jones M**

2248009

Under the guidance of **Dr.Apash Roy**

A Project report submitted in partial fulfillment of the requirements of IV Semester Master of Science (Data Science) of CHRIST (Deemed to be University)

May - 2024

CERTIFICATE

*This is to certify that the report titled* ***SnoreScan: A Smart App for Sleep Apnea Detection*** *is a bonafide record of work done by* ***Jones M(2248009)*** *of CHRIST (Deemed to be University), Bengaluru, in partial fulfillment of the requirements of IV Semester MSc(Data Science) during the year 2023-24.*

**Head of the Department Project Guide**

Valued-by:

Name : Jones M

1. Register Number 2248009

Examination Centre : CHRIST (Deemed to be University)

1. Date of Exam :

**Acknowledgements**

This project was successfully completed with the help and support of a lot of people. Firstly, I express my sincere gratitude to our Vice-Chancellor Dr. Fr. Joseph C C, Pro-Vice Chancellor Dr. Fr. Viju P D, Head of the Department Dr. Saleema J S , Coordinator Dr. Rupali Sunil Wagh and the faculty for providing us with this priceless opportunity to put in and improve my technical skills and knowledge through this project.

I would like to express my gratitude to my project guide Dr. Apash Roy for taking keen interest in helping and guiding me throughout the project work by providing all the necessary information, giving ideas, continuous feedback and supervision and helping me in coming up with a good result.

I’m thankful and fortunate enough to get constant encouragement, support and guidance from the faculty of Masters of Science Data Science program who helped me in successfully completing my project work. Finally, my deep and sincere gratitude to my friends, classmates and family members for their continuous and unparalleled love, help and support.

**Abstract**

The SnoreScan project aims to develop an intelligent mobile application that utilizes audio data analysis and machine learning techniques to detect snoring patterns and assess the risk of obstructive sleep apnea (OSA) in users. Sleep disorders, such as OSA, pose significant health risks and can impact overall well-being. However, early detection and monitoring of sleep patterns can help individuals take proactive measures to improve their sleep health.

The SnoreScan application employs advanced signal processing algorithms to analyze recorded audio data and identify snoring events. Leveraging machine learning models, the application evaluates the severity of snoring and assesses the user's risk of OSA based on various factors, including snoring patterns, sleep duration, and other relevant metrics. By providing personalized recommendations and educational resources, SnoreScan aims to empower users to make informed decisions about their sleep health.

The development of SnoreScan involves addressing several technical challenges, including audio signal processing, machine learning model training, and mobile application design. By integrating cutting-edge technologies and ensuring user-friendly interface design, SnoreScan aims to deliver a seamless and intuitive user experience.

Overall, the SnoreScan project represents a significant advancement in the field of sleep health monitoring and demonstrates the potential of mobile technology and machine learning in improving healthcare outcomes. Through ongoing research and development efforts, SnoreScan seeks to enhance its capabilities and contribute to the early detection and management of sleep disorders on a global scale.

**TABLE OF CONTENTS**

**Acknowledgments**

**Abstract**

**List of Figures**

1. **Introduction**

1.1. Project Description

1.2. Existing System

1.3. Objectives

1.4. Purpose, Scope and Applicability

1.4.1. Purpose

1.4.2. Scope

1.4.3. Applicability

2. **System Analysis and Requirements**

2.1. Problem Deﬁnition

2.2. Requirements Specification

2.3. Block Diagram

2.4. System Requirements

2.4.1. User Characteristics

2.4.2. Software and Hardware Requirements

2.4.3. Constraints

2.5. Conceptual Models

3. **System Design**

3.1. System Architecture

3.2. Module Design

3.3. Database Design

3.3.1. Tables and Relationships

3.3.2. Data Integrity and Constraints

3.4. Interface Design and Procedural Design

3.4.1. User Interface Design

4. **Implementation**

4.1. Implementation Approaches

4.2. Coding Standard

4.3. Coding Details

4.4. Screen Shots

5. **Testing**

5.1. Test Cases

5.2. Testing Approaches

6. **Conclusion**

6.1. Design and Implementation Issues

6.2. Advantages and Limitations

6.3. Future Scope of the Project

**References**

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figure Name** | **Page No.** |
| 1 | Architecture Flow Diagram | 17 |
| 2 | ER Diagram | 21 |
| 3 | App Screenshots | 25 |
| 4 | Code Screenshots | 28 |

**1.INTRODUCTION**

In the realm of healthcare technology, the intersection of artificial intelligence and mobile applications has paved the way for innovative solutions to address critical health issues. Among these, SnoreScan emerges as a groundbreaking project, focusing on the development of an Android app designed to analyze snoring sounds during sleep, offering insights into potential sleep disorders, with a primary emphasis on sleep apnea.

Sleep disorders, often underestimated in their impact, can have profound effects on an individual's overall well-being and health. The prevalence of conditions like sleep apnea underscores the importance of accessible tools for early detection and monitoring. SnoreScan aims to fill this gap by leveraging advancements in machine learning, specifically Convolutional Neural Networks (CNNs), to interpret audio data captured during sleep.

Traditional methods of diagnosing sleep disorders often involve elaborate and expensive procedures conducted in specialized sleep clinics. SnoreScan seeks to democratize this process by providing a convenient and user-friendly solution that individuals can use in the comfort of their own homes. The Android app empowers users to take an active role in monitoring their sleep health, receiving real-time feedback on potential sleep disorders through the analysis of their snoring patterns.

**1.2 Project Description**

SnoreScan is an innovative mobile application designed to address the common issue of snoring and its potential correlation with obstructive sleep apnea (OSA). Leveraging machine learning and signal processing techniques, SnoreScan aims to provide users with insights into their snoring patterns and assess their risk of OSA. By analyzing recorded audio snippets of snoring, the app offers a convenient and accessible tool for individuals to monitor their sleep health and take proactive steps towards better sleep quality.

**1.3 Existing System**

There are some mobile application that records voices and detects snoring and non-snoring sounds. These platforms include:

1. SnoreLab: Record Your Snoring - SnoreLab records, measures and tracks your snoring and helps you to discover effective ways to reduce it. The app is very easy to use: simply set SnoreLab running next to your bed whilst you sleep. In the morning you will discover your Snore Score, exactly when and how loudly you snored, and listen to some highlights. SnoreLab lets you log and track lifestyle factors and any snoring remedies so you can see how they impact your snoring.
2. Snore Control - The data and history are well managed and arranged in the app. This provides you with easy access to these whenever required. You get an option to ‘record the snoring,’ The application then detects your snores and records only your snoring. The app also analyses your sleeping pattern and presents it in charts. Adjustable sensitivity to the snoring sound. You can select to play a chosen sound or vibrate your device whenever snoring is detected.
3. myNight - The app uses your microphone and speakers to record your snoring sounds. Using this app, you can begin recording when you head to sleep and stop when you wake up. You can later analyze and hear the recording to learn about your snoring habits in detail. The app also monitors the duration and number of times you take breathing or snoring pauses. It tracks the complete course in which you snore and presents a comprehensive report.

**1.4 Objectives**

* Develop a mobile application capable of accurately detecting snoring events.
* Implement machine learning algorithms to analyze snoring patterns and assess the user's risk of obstructive sleep apnea.
* Provide users with a user-friendly interface to record and monitor their snoring episodes.
* Offer educational resources and recommendations for improving sleep hygiene based on individual sleep patterns.
* Collaborate with healthcare professionals to enhance the app's diagnostic capabilities and provide users with access to expert advice and support.

**1.4 Purpose, Scope, and Applicability**

**1.4.1 Purpose**

The purpose of SnoreScan is to empower individuals to monitor their snoring patterns, assess their risk of obstructive sleep apnea, and take proactive steps towards improving their sleep health. By offering a user-friendly mobile application with advanced analysis capabilities, SnoreScan aims to raise awareness about the importance of sleep quality and facilitate early detection of sleep-related disorders.

* + 1. **Scope**

The primary focus of SnoreScan lies in collecting audio recordings of individuals during their sleep to assess the presence of sleep apnea and other sleep disorders. Users will generate these recordings through the SnoreScan Android app, creating a diverse dataset encompassing various sleep patterns.

The project will employ machine learning techniques, particularly Convolutional Neural Network (CNN) models, to analyze the audio data and identify distinctive patterns associated with sleep disorders. The scope includes data preprocessing to extract relevant features from the audio recordings, laying the foundation for effective model training.

The development of the Android app itself is a crucial component of the project's scope. The app will offer an intuitive and user-friendly interface, allowing users to effortlessly record and analyze snoring sounds. Real-time feedback will be provided, and the app will generate alerts to notify users of potential sleep disorders based on the model's analysis results.

Model evaluation will be a key aspect of the project scope, utilizing metrics such as accuracy, precision, recall, and F1 score. Cross-validation on diverse datasets will ensure that the trained CNN model generalizes well to different sleep patterns and demographics, enhancing its reliability and applicability.

**1.4.3 Applicability:**

SnoreScan is applicable to:

* Individuals concerned about their snoring patterns and sleep quality.
* Healthcare professionals seeking diagnostic tools for sleep-related disorders.
* Sleep clinics and medical facilities offering sleep health services.
* Researchers studying sleep disorders and their implications for overall health.

**1.5 Overview of the Report**

This report provides a comprehensive overview of the SnoreScan app, including its development objectives, features, market analysis, competitive landscape, business model, target audience, and future prospects. By examining each aspect of the project in detail, this report aims to highlight the significance of SnoreScan in promoting sleep health awareness and facilitating early detection of obstructive sleep apnea.

**2. SYSTEM ANALYSIS AND REQUIREMENTS**

The SnoreScan system utilizes audio data analysis and machine learning models to detect snoring patterns and assess the risk of obstructive sleep apnea (OSA) in users. Leveraging real-time audio processing and machine learning algorithms, SnoreScan aims to provide personalized insights into sleep health and facilitate early detection of potential sleep disorders.

**2.1 Problem Definition**

The objective of SnoreScan is to develop a mobile application that accurately detects snoring patterns and evaluates the likelihood of obstructive sleep apnea (OSA) based on user recordings. The system faces several challenges, including:

* Developing robust algorithms for snoring detection: Implementing signal processing techniques and machine learning models to accurately identify snoring events in recorded audio data.
* Assessing OSA risk: Utilizing machine learning algorithms to analyze snoring patterns and other relevant factors to assess the user's risk of obstructive sleep apnea.
* Providing personalized insights: Tailoring recommendations and educational resources based on individual sleep patterns, risk factors, and user preferences.
* Ensuring user-friendly interface: Designing an intuitive and user-friendly mobile application interface for recording, analyzing, and interpreting snoring data.

The ultimate goal of SnoreScan is to empower users to monitor their sleep health, identify potential sleep disorders, and take proactive steps towards improving their overall well-being.

**2.2. Requirements Specification**

**2.2.1. Functional Requirements**

* Snoring Detection: Implement audio processing algorithms to detect snoring events in recorded audio data.
* OSA Risk Assessment: Develop machine learning models to analyze snoring patterns and assess the user's risk of obstructive sleep apnea.
* User Authentication: Provide secure user authentication mechanisms to ensure data privacy and confidentiality.
* Audio Recording and Playback: Enable users to record and playback audio samples for snoring analysis.
* Personalized Recommendations: Generate personalized recommendations and educational resources based on individual sleep patterns and risk factors.
* User Profile Management: Allow users to create and manage their profiles, including personal information, sleep history, and preferences.
* Notification System: Implement a notification system to alert users about significant changes in their sleep patterns or risk factors.
* Data Visualization: Provide visualizations of snoring patterns, sleep metrics, and OSA risk assessments to help users understand their sleep health.
* Feedback Mechanism: Incorporate a feedback mechanism for users to provide input, suggestions, and report issues to improve the application.

**2.2.2. Non-Functional Requirements**

* Performance: Ensure fast and efficient performance of the application, especially during audio processing and machine learning model inference.
* Security: Implement robust security measures to protect user data and ensure compliance with data privacy regulations.
* Usability: Design an intuitive and user-friendly interface for easy navigation and interaction with the application.
* Reliability: Ensure the reliability and stability of the application, minimizing downtime and potential errors.
* Scalability: Design the application to scale with increasing user base and data volume, ensuring continued performance and efficiency.

**2.3. Block Diagram**

Figure 2.1: SnoreScan System Architecture

**2.4. System Requirements**

The SnoreScan application is designed to run on mobile devices running Android OS version 10.0 and above. The following are the system requirements:

**2.4.1. Software Requirements**

* Operating System: Android OS version 7.0 (Nougat) or above
* Development Platform: Android Studio IDE for application development
* Programming Language: Java/Kotlin for Android app development
* Libraries/Frameworks: TensorFlow Lite for machine learning model inference, Android MediaRecorder API for audio recording, Firebase Authentication for user authentication, Firebase Cloud Messaging for push notifications
* Backend Services: Firebase Realtime Database or Firestore for storing user data and preferences
* External APIs: Google Maps API for location-based services, Firebase ML Kit for machine learning capabilities

**2.4.2. Hardware Requirements**

* Processor: ARMv8-A architecture or higher
* RAM: 2GB or higher
* Storage: Minimum 100MB of available storage space for application installation and data storage
* Microphone: Built-in microphone for audio recording
* Internet Connectivity: Wi-Fi or mobile data connection for accessing backend services and downloading updates

**2.4.3. Constraints**

* Limited computational resources on mobile devices may affect the performance of audio processing and machine learning algorithms.
* Privacy concerns regarding the collection and storage of sensitive user data, such as audio recordings and health information.
* Dependence on external APIs and services for certain functionalities, which may introduce latency and reliability issues.
* Compliance with regulatory requirements, such as GDPR for data privacy and security, may impose constraints on data handling and storage practices.

**2.5. Conceptual Models**

The SnoreScan application incorporates signal processing techniques and machine learning models to analyze recorded audio data and detect snoring patterns. The following conceptual models are used:

* Audio Processing Pipeline: Preprocessing audio data to remove noise and extract relevant features for snoring detection.
* Machine Learning Model: Training a machine learning model, such as a convolutional neural network (CNN) or recurrent neural network (RNN), to classify audio segments as snoring or non-snoring.
* User Feedback Loop: Collecting user feedback and annotations to improve the performance and accuracy of the machine learning model over time.

By combining these conceptual models, SnoreScan aims to provide users with accurate and personalized insights into their sleep health and facilitate early detection of potential sleep disorders such as obstructive sleep apnea.

**3.SYSTEM DESIGN**

**3.1 System Architecture**

The SnoreDetect project employs a comprehensive system architecture aimed at classifying snoring sounds and identifying episodes of obstructive sleep apnea (OSA) within those snoring events. The architecture encompasses several key stages, each contributing to the overall functionality and performance of the system. Let's explore the system architecture in detail:

1. Data Acquisition:

The system starts by acquiring audio recordings containing snoring sounds from various sources, such as microphones or recording devices. These audio recordings serve as the primary input data for the subsequent stages of the system.

2. Preprocessing:

Once the audio recordings are acquired, they undergo preprocessing to enhance the quality of the data and prepare it for feature extraction and classification. Preprocessing steps may include noise reduction, normalization, and segmentation to isolate individual snoring events.

3. Feature Extraction:

In this stage, the preprocessed audio data is analyzed to extract relevant features that capture important characteristics of snoring sounds. Features such as spectral features, temporal features, and frequency domain features are extracted from the audio signals, providing valuable information for classification.

4. Model Training:

The system utilizes Convolutional Neural Network (CNN) models to classify snoring sounds and identify episodes of obstructive sleep apnea. Two CNN models are employed: one for classifying snoring and non-snoring sounds, and another for distinguishing between apnea and non-apnea snoring events.

5. CNN Model for Snoring Classification:

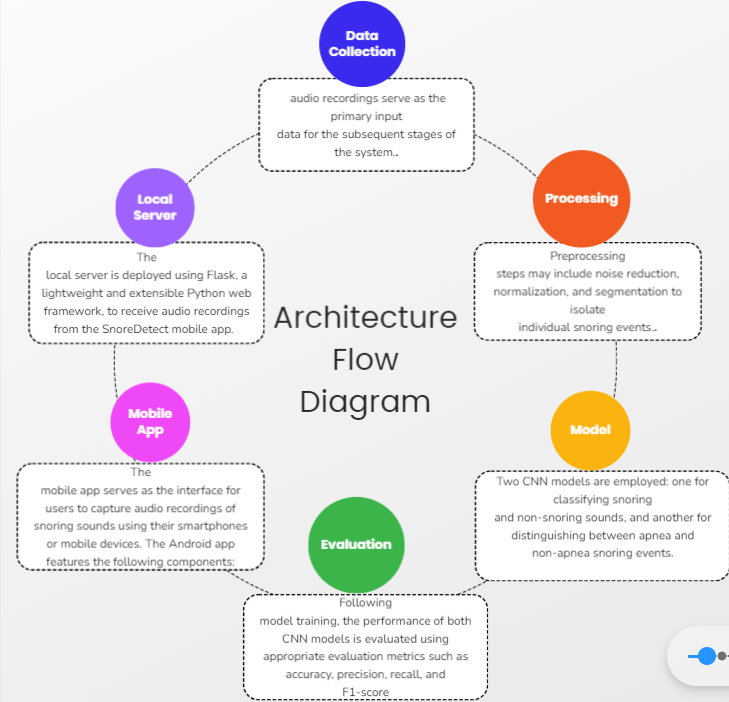
The first CNN model is trained to classify audio segments as either snoring or non-snoring. This model consists of convolutional layers followed by max-pooling layers to extract relevant features from the audio data. The extracted features are then passed through fully connected layers with dropout regularization to classify the audio segments.

6. CNN Model for Apnea Detection:

The second CNN model is designed specifically to identify episodes of obstructive sleep apnea within the snoring events. This model also comprises convolutional layers followed by max-pooling layers for feature extraction. The extracted features are then fed into fully connected layers to predict whether a snoring event corresponds to an episode of obstructive sleep apnea.

7. Model Evaluation and Optimization:

Following model training, the performance of both CNN models is evaluated using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. The models are then optimized through techniques such as hyperparameter tuning and regularization to improve their accuracy and generalization capability.



**Fig 1. Architecture Flow Diagram**

**3.2. Module Design**

1. Mobile Application (Android Studio):

The SnoreDetect mobile application is developed using Android Studio, a popular integrated development environment (IDE) for Android app development. The mobile app serves as the interface for users to capture audio recordings of snoring sounds using their smartphones or mobile devices. The Android app features the following components:

* User Interface (UI): The UI of the mobile app includes buttons for recording audio, stopping the recording, and initiating the transmission of audio data to the local server. Additionally, the app may include features for user authentication, settings configuration, and displaying real-time feedback during audio recording.
* Audio Recording Module: The mobile app utilizes built-in functionalities of Android devices to capture audio recordings of snoring sounds. This module incorporates features such as audio sampling, noise reduction, and real-time visualization of audio input to aid users in capturing high-quality recordings.
* Data Transmission Module: Upon recording a snoring sound, the mobile app initiates the transmission of audio data to the local server for further analysis. This module establishes a connection with the server and sends the recorded audio data in a suitable format for processing.

2. Local Server (Flask Python Script):

The local server is deployed using Flask, a lightweight and extensible Python web framework, to receive audio recordings from the SnoreDetect mobile app. The server is responsible for processing the received audio data, performing snoring detection and obstructive sleep apnea (OSA) classification, and providing feedback to the mobile app. The local server architecture includes the following components:

* Flask Application: The Flask application serves as the core of the local server, handling incoming requests from the mobile app, processing audio data, and generating classification results. The Flask application consists of routes for receiving audio data, performing analysis using machine learning models, and returning results to the mobile app.
* Audio Data Processing Module: Upon receiving audio data from the mobile app, the Flask application preprocesses the audio recordings to enhance their quality and extract relevant features for snoring detection and OSA classification. This module may include functionalities for noise reduction, feature extraction, and signal processing.
* Machine Learning Model Integration: The Flask application integrates machine learning models, such as Convolutional Neural Networks (CNNs), trained for snoring detection and OSA classification. These models analyze the preprocessed audio data to identify snoring sounds and detect episodes of obstructive sleep apnea. The integration of machine learning models allows for accurate and real-time analysis of audio recordings.
* Response Generation Module: After processing the audio data, the Flask application generates classification results indicating whether snoring sounds are present and if they correspond to episodes of obstructive sleep apnea. These results are formatted into a suitable response format (e.g., JSON) and sent back to the mobile app for display to the user.

3. Communication Protocol:

Communication between the SnoreDetect mobile app and the local server is established using Hypertext Transfer Protocol (HTTP) or Transmission Control Protocol (TCP). The mobile app sends HTTP requests containing audio data to the server, and the server responds with classification results. This communication protocol ensures seamless data exchange between the mobile app and the local server, facilitating real-time analysis of snoring sounds.

**3.3.Database Design**

**3.3.1. Tables and Relationships**

1.Snoring Sound Recording Table:

* Structure: recording\_id (Primary Key), user\_id (Foreign Key to Users), recording\_path, recording\_duration, recording\_timestamp.
* This table stores information about individual snoring sound recordings. Each recording is uniquely identified by a recording ID (recording\_id) and is associated with the user who made the recording (user\_id). Details such as the file path (recording\_path), duration (recording\_duration), and timestamp (recording\_timestamp) are included.

2.Users Table:

* Structure: user\_id (Primary Key), username, email, password\_hash, registration\_date.
* This table contains user information, including their unique ID (user\_id), username, email address, hashed password (password\_hash), and registration date (registration\_date). Each user is assigned a unique ID as the primary key.

3.Snoring Classification Table:

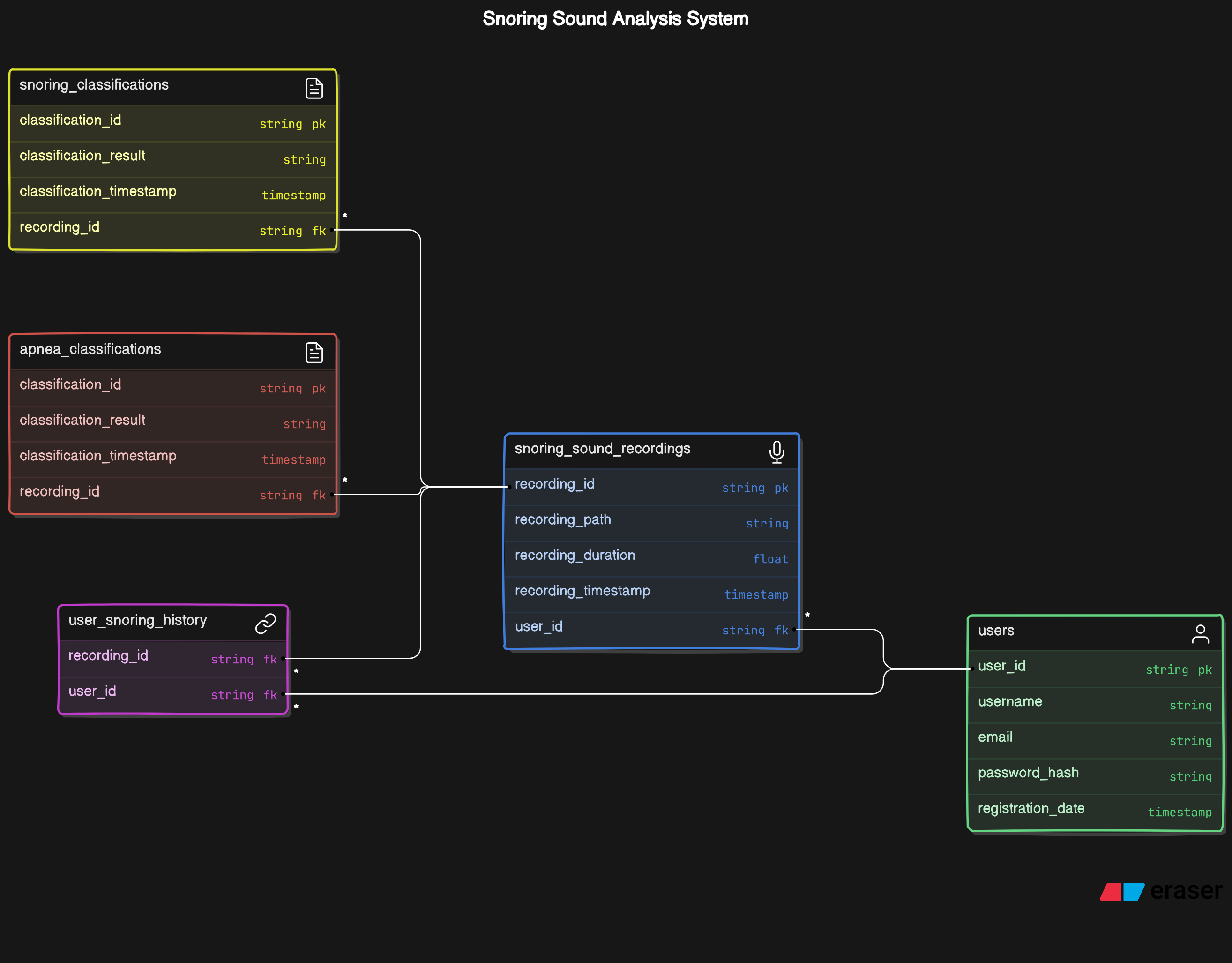
* Structure: classification\_id (Primary Key), recording\_id (Foreign Key to Snoring Sound Recording), classification\_result, classification\_timestamp.
* This table records the classification results of snoring sound recordings. Each classification result is uniquely identified by a classification ID (classification\_id) and is associated with the corresponding snoring sound recording (recording\_id). The classification result (classification\_result) and the timestamp of the classification (classification\_timestamp) are stored.

4.Apnea Classification Table:

* Structure: classification\_id (Primary Key), recording\_id (Foreign Key to Snoring Sound Recording), classification\_result, classification\_timestamp.
* This table stores the classification results of snoring sound recordings for apnea detection. Each classification result is identified by a classification ID (classification\_id) and is linked to the respective snoring sound recording (recording\_id). Details such as the classification result (classification\_result) and the timestamp of the classification (classification\_timestamp) are recorded.

5.User Snoring History Table (Many-to-Many Relationship):

* Structure: user\_id (Foreign Key to Users), recording\_id (Foreign Key to Snoring Sound Recording).
* This table establishes a many-to-many relationship between users and snoring sound recordings. It allows multiple users to be associated with multiple recordings, representing the history of snoring recordings for each user. Each record in this table links a user ID (user\_id) to a recording ID (recording\_id), indicating which users have recorded which snoring sounds.



**Fig 2 ER Diagram**

**3.3.2.Data Integrity and Constraints**

1. Audio Data Integrity:

* The dataset consists of audio recordings of snoring sounds, categorized into snoring and non-snoring samples.
* Number of Snoring Samples: 100
* Number of Non-Snoring Samples: 100

1. Data Preprocessing Constraints:

* During preprocessing, noise such as background sounds, speech, and non-relevant audio segments must be filtered out to isolate snoring sounds effectively.
* Techniques such as noise reduction, signal segmentation, and normalization may be applied to enhance the quality of audio recordings.
* Standardization of audio formats and sampling rates is necessary for consistent processing across different recording devices.
* Preprocessing steps should be optimized to minimize information loss while reducing computational complexity.

1. Feature Extraction and Model Training:

* Features are extracted from audio recordings using techniques such as Short-Time Fourier Transform (STFT) or Mel-Frequency Cepstral Coefficients (MFCCs) to capture relevant characteristics of snoring sounds.
* Models, including Convolutional Neural Networks (CNNs) and recurrent neural networks (RNNs), are trained to classify snoring and non-snoring sounds.
* Hyperparameters such as filter sizes, kernel lengths, and model architectures must be tuned to achieve optimal performance.
* Evaluation metrics such as accuracy, sensitivity, specificity, and F1-score are used to assess model performance.

1. Model Evaluation and Interpretation:

* Performance metrics, including receiver operating characteristic (ROC) curves and confusion matrices, are utilized for model evaluation and visualization.
* Misclassifications are analyzed to identify patterns and potential areas for improvement in model training and feature extraction.
* Interpretation of results should consider the implications for practical use cases, such as the effectiveness of snoring detection in real-world scenarios.

1. Resource Constraints:

* Training deep learning models for audio classification requires significant computational resources, including GPU accelerators for accelerated training.
* Availability of suitable hardware and software environments, such as TensorFlow or PyTorch frameworks, may impact project scalability and execution.
* Storage capacity for storing large audio datasets and model checkpoints should be considered to accommodate project requirements.
* Time constraints associated with data collection, preprocessing, model training, and evaluation may necessitate efficient workflows and task prioritization.

1. Mobile App Development:

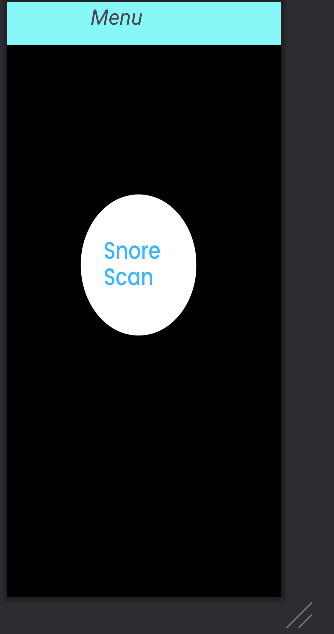
* Compatibility: The mobile app must be compatible with various Android and iOS devices, necessitating thorough testing across different screen sizes, resolutions, and operating system versions.
* User Interface: The app's interface should be intuitive and user-friendly, considering factors such as navigation flow, button placement, and accessibility features for users with disabilities.
* Performance: Optimizing the app's performance is crucial to ensure smooth operation and responsiveness, requiring efficient resource management and minimal battery consumption.
* Security: Implementation of robust security measures, including data encryption, secure authentication mechanisms, and protection against common mobile app vulnerabilities, is essential to safeguard user data and privacy.

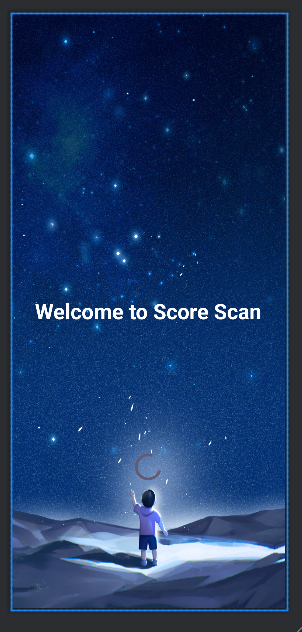
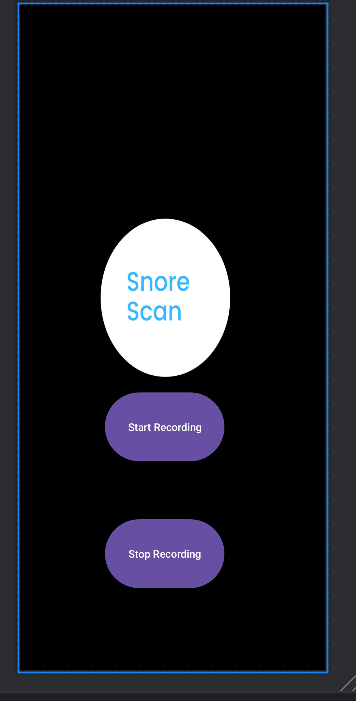
1. Python Script for Audio Processing:

* Compatibility: The Python script should be compatible with the Flask framework for web server deployment, ensuring seamless integration with the mobile app's backend.
* Audio Input Handling: The script must handle audio input from the mobile app, including audio file uploads or real-time audio streaming, while ensuring data integrity and security.
* Error Handling: Robust error handling mechanisms should be implemented to handle exceptions gracefully and prevent application crashes or data loss.
* Scalability: The script's architecture should be designed to accommodate potential scalability requirements, such as handling multiple concurrent audio streams or processing large volumes of data efficiently.
* Resource Management: Efficient utilization of computational resources, including memory and CPU usage, is critical to ensure optimal performance and scalability of the Python script.

### 3.4. Interface and Procedural Design

**3.4.1. User Interface Design:**



  
**Fig 3 App Screenshots**

**4.Implementation**

**4.1. Implementation Approaches:**

The implementation approach for the snoring and apnea detection system follows a structured and iterative process aimed at delivering a reliable and user-friendly solution. Here's an outline of the implementation approach:

* Environment Setup: We will begin by setting up the necessary development environment, including tools such as Android Studio for mobile app development and Python for server-side scripting. Additionally, we will configure a local server using Flask to receive audio data from the mobile app.
* Agile Development Methodology: We will adopt an Agile development methodology to iteratively build and refine the system components. This approach allows for flexibility and adaptability, enabling us to respond to changes and feedback efficiently.
* Phase-wise Development:
* Phase 1: Mobile App Development: The initial phase will focus on developing the mobile app using Android Studio. This includes implementing features for recording snoring sounds and transmitting the audio data to the server.
* Phase 2: Server-side Scripting: In parallel, we will develop the server-side scripts using Python and Flask. These scripts will handle audio data reception, preprocessing, and passing it to the machine learning models for classification.
* Phase 3: Machine Learning Model Integration: Next, we will integrate the machine learning models for snoring and apnea classification into the server-side scripts. This involves training the CNN models for snoring and apnea detection and deploying them to classify the audio data received from the mobile app.
* Phase 4: Testing and Validation: Throughout the development process, rigorous testing and validation will be conducted to ensure the accuracy and reliability of the system. This includes unit testing, integration testing, and real-world validation using diverse datasets.
* Phase 5: User Feedback and Iteration: Following the initial deployment, we will gather feedback from users and stakeholders to identify areas for improvement. Based on this feedback, iterative updates and enhancements will be made to enhance the system's performance and usability.
* User Experience and Integration: Our implementation approach prioritizes user experience, with a focus on creating an intuitive and seamless interaction between the mobile app and server-side components. Integration of technologies such as Flask for server-side scripting and Android Studio for mobile app development ensures a cohesive and efficient system architecture.
* Future Development Roadmap: The implementation approach also includes planning for future development, such as incorporating advanced machine learning techniques, expanding the feature set, and optimizing performance based on evolving user needs and technological advancements. This iterative approach ensures that the system remains relevant and effective in addressing snoring and apnea detection challenges.

**4.2. Coding Standard**

1. Naming Conventions:

* Use descriptive and meaningful variable, function, and class names.
* Follow camelCase convention for variable and function names (e.g., recordAudio, classifySnoring).
* Class names should be in PascalCase (e.g., AudioRecorder, SnoringClassifier).

1. Indentation and Formatting:

* Use consistent indentation with either spaces or tabs (typically 4 spaces).
* Maintain consistent line lengths (recommended maximum of 80-120 characters per line).
* Follow a consistent coding style for braces placement (e.g., Allman style or K&R style).

1. Comments and Documentation:

* Include descriptive comments to explain complex logic, algorithms, and function behavior.
* Use docstrings to provide documentation for classes, functions, and modules.
* Document input parameters, return values, and any exceptions raised by functions.

1. Error Handling:

* Implement robust error handling to handle exceptions gracefully.
* Use meaningful error messages to aid in debugging and troubleshooting.
* Log errors and exceptions to facilitate error diagnosis and resolution.

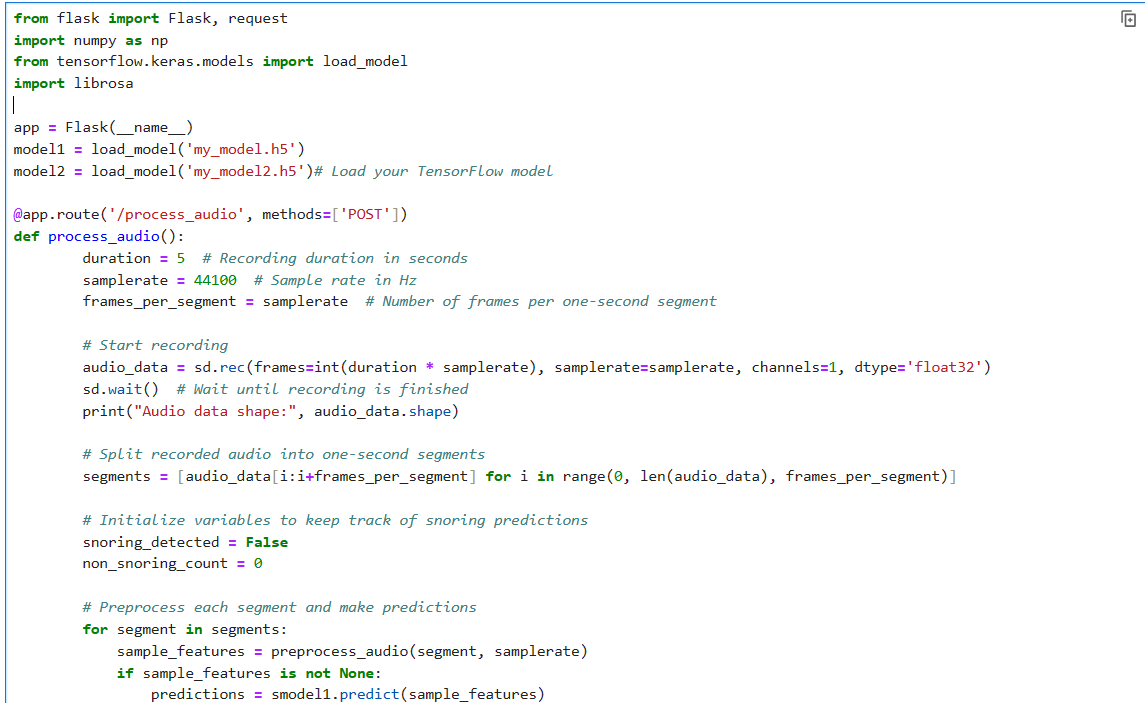
**4.3.Coding Details**

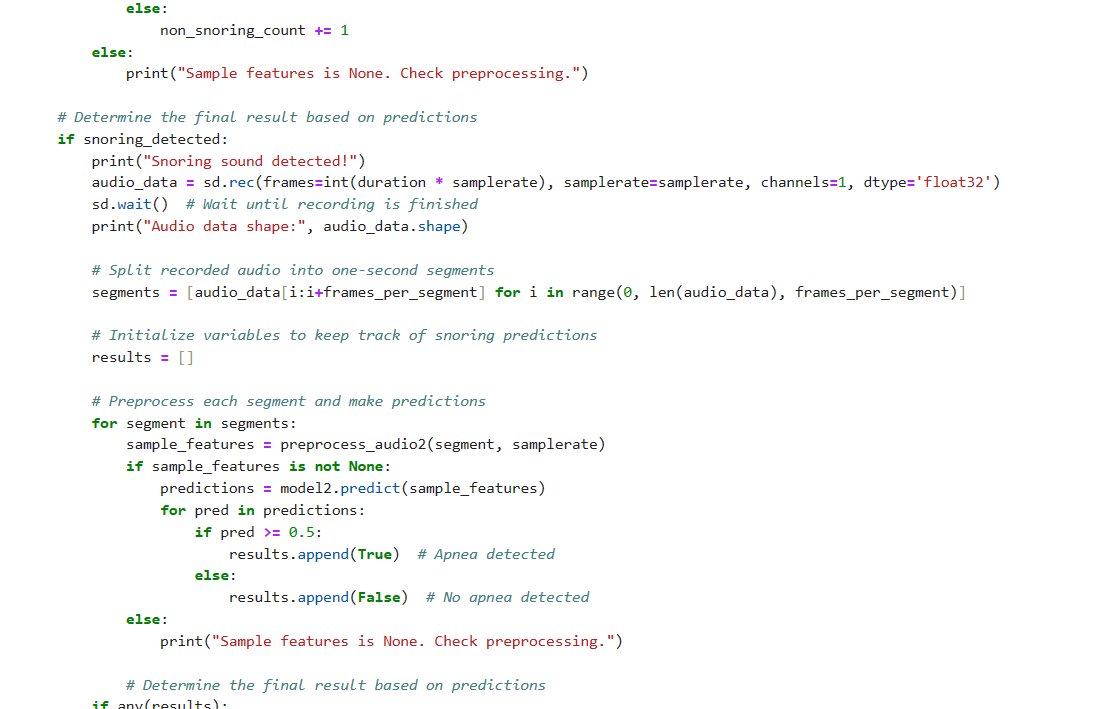
**4.3.1. Screenshots**

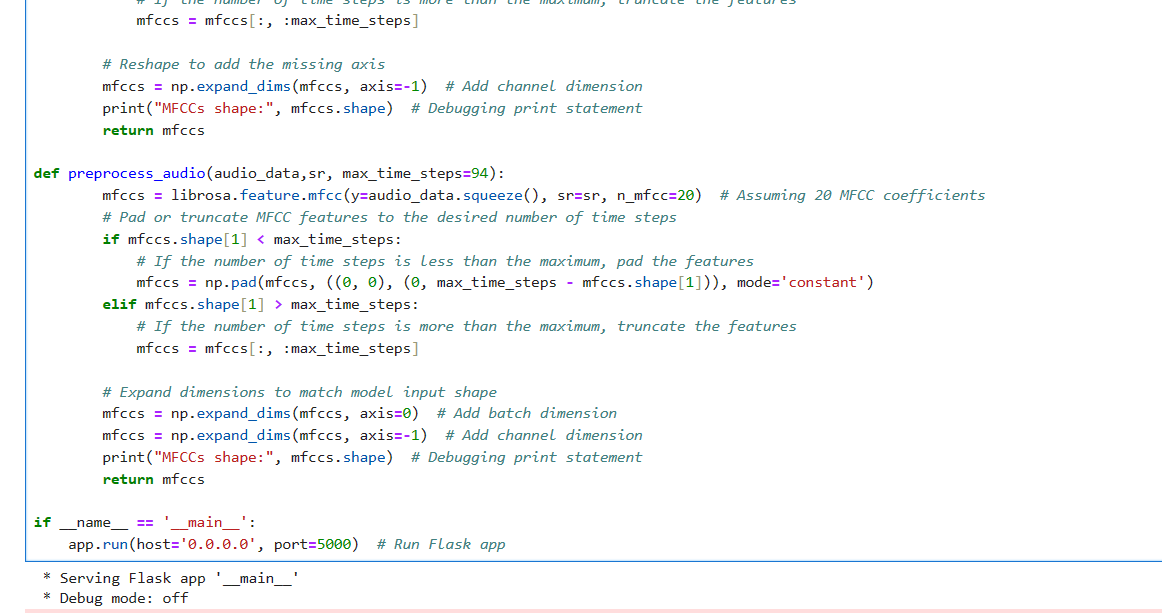
**Model Training**

****

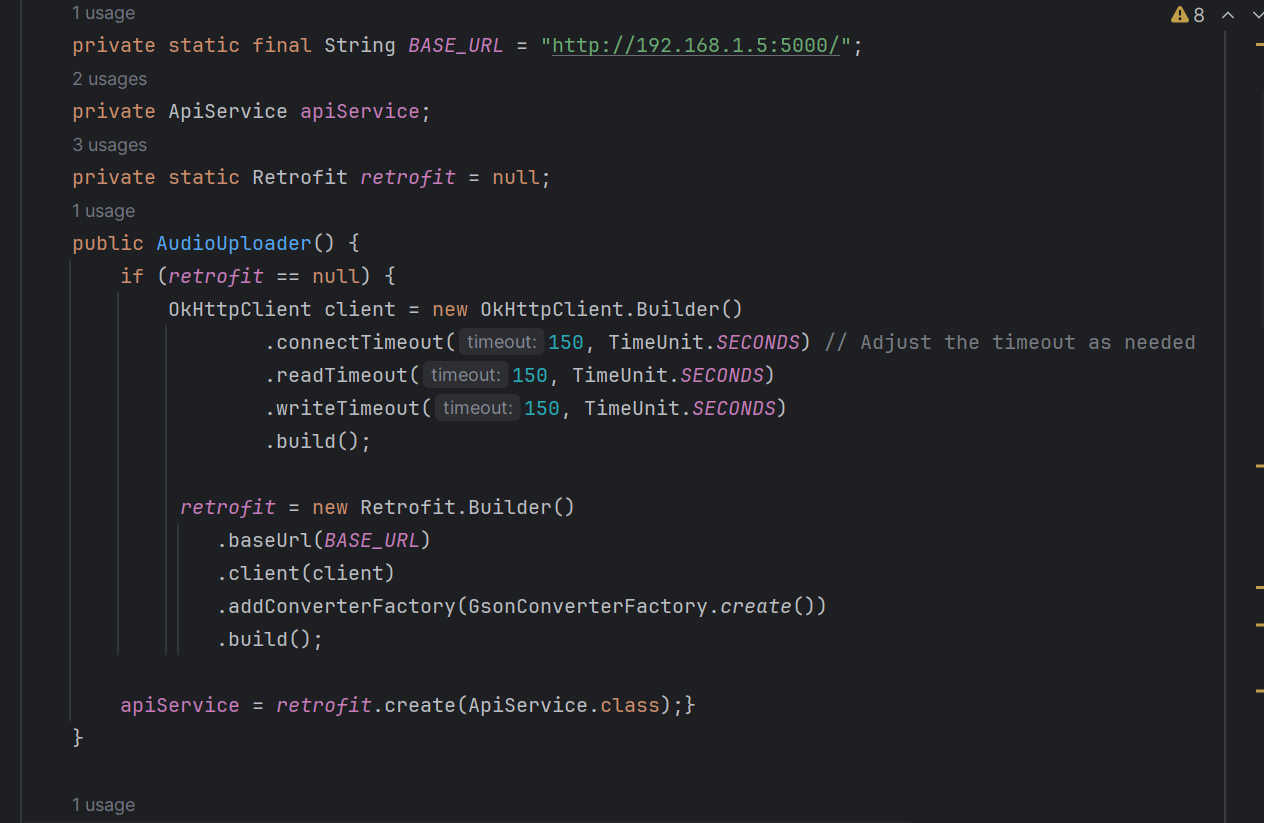
**Creating local server using Flask**

****

****

****

**Creating REST API using Retrofit**

****

**Fig 4 Code Screenshots**

**5.Testing**

**5.1.Testing Cases**

Data Preprocessing:

* Test Case 1: Verify that the audio data is correctly sampled and formatted before preprocessing.
* Test Case 2: Ensure that noise reduction techniques effectively remove background noise from the audio signals.
* Test Case 3: Validate that the audio signals are segmented into appropriate timeframes for analysis.
* Test Case 4: Check that features such as spectrograms or MFCCs are extracted accurately from the preprocessed audio data.
* Test Case 5: Verify that labels (snoring, non-snoring, apnea, non-apnea) are assigned correctly to the preprocessed audio samples.

Model Training (CNN for Snoring Detection):

* Test Case 6: Train the CNN model using the preprocessed audio dataset and monitor loss and accuracy metrics during training.
* Test Case 7: Validate that the trained CNN model achieves convergence and satisfactory performance on a validation set.
* Test Case 8: Assess the CNN model's ability to generalize by evaluating its performance on a separate test set.
* Test Case 9: Analyze the confusion matrix and ROC curve of the CNN model to understand its classification performance.

Model Training (CNN for Apnea Detection):

* Test Case 10: Train the CNN model for apnea detection using the preprocessed audio dataset and monitor training metrics.
* Test Case 11: Ensure that the trained CNN model converges and achieves satisfactory performance on a validation set.
* Test Case 12: Validate the generalization ability of the CNN model by evaluating its performance on a separate test set.
* Test Case 13: Analyze the confusion matrix and ROC curve of the CNN model to assess its effectiveness in detecting apnea events.

Integration Testing:

* Test Case 14: Verify that the trained CNN models are integrated correctly into the mobile app for real-time audio analysis.
* Test Case 15: Ensure that the mobile app can capture audio input from the device's microphone and transmit it to the local server.
* Test Case 16: Validate that the local server receives audio data from the mobile app, processes it using the CNN models, and returns the classification results.
* Test Case 17: Test the end-to-end functionality of the system, including audio capture, transmission, analysis, and result display on the mobile app interface.

**6. Conclusion**

**6.1. Design and Implementation Issues**

Design Issues:

* Selection of appropriate machine learning models for snoring and apnea detection is crucial to ensure accuracy and reliability.
* Integration of the Android app with the local Flask server for audio data transmission requires careful consideration of communication protocols and security measures.
* Designing the user interface of the Android app to provide seamless audio recording and transmission functionalities poses usability challenges.
* Ensuring compatibility and scalability of the system architecture to accommodate potential future enhancements and increased user demand is essential.

Implementation Issues:

* Optimizing the CNN models for snoring and apnea classification to balance between model complexity and performance is challenging.
* Developing the Flask Python script to receive audio data from the Android app and process it in real-time necessitates robust error handling and data validation mechanisms.
* Integrating the CNN models with the Android app to enable on-device audio classification requires efficient memory management and processing optimizations.
* Conducting comprehensive testing and validation of the entire system, including the Android app, local server, and CNN models, to ensure seamless functionality and accurate classification results.

**6.2. Advantages and Limitations**

Advantages:

* Provides a user-friendly and accessible solution for snoring and apnea detection using widely available Android smartphones.
* Offers real-time audio classification capabilities, enabling timely identification of potential sleep-related breathing disorders.

Limitations:

* Accuracy of snoring and apnea classification may vary depending on the quality of audio recordings and environmental noise levels.
* Reliability of the system may be impacted by network connectivity issues or limitations in computational resources on mobile devices.

**6.3. Future Scope of the Project**

* Expanding the system's capabilities to include additional sleep-related metrics, such as sleep stages or respiratory rates, for more comprehensive sleep monitoring.
* Implementing machine learning techniques for personalized sleep disorder risk assessment based on individual user data and sleep patterns.
* Enhancing the user experience of the Android app through advanced audio visualization features and integration with wearable devices for continuous sleep monitoring.
* Collaborating with healthcare professionals and sleep specialists to validate the system's accuracy and reliability for clinical use in diagnosing sleep disorders.

**References**

1.Berry, R. B. et al. The AASM manual for the scoring of sleep and associated events. Am. Acad. Sleep Med. 53, 1689–1699 (2013).

2. Tsara, V., Amflochiou, A., Papagrigorakis, M. J., Georgopoulos, D. & Liolios, E. Guidelines for diagnosis and treatment of sleep related breathing disorders in adults and children:

Definition and classifcation of sleep related breathing disorders in adults.

Different types and indications for sleep studies (Part 1). Hippokratia 13, 187–191 (2009).

3. Guilleminault, C. Obstructive sleep apnea. The clinical syndrome and historical perspective. Med. Clin. North Am. 69, 1187–1203,

https://doi.org/10.1016/S0025-7125(16)30982-8 (1985).

4. Ruixue Li, Wenjun Li, Keqiang Yue, Yilin Li. Convolutional neural network for screening of obstructive sleep apnea using snoring sounds

<https://doi.org/10.1016/j.bspc.2023.104966> (2023).

5.  Maimon N, Hanly PJ. Does snoring intensity correlate with the severity of obstructive sleep apnea? *J Clin Sleep Med.*2010 Oct;6(5):475–478.