# SLEEPEASE: AN AI-INTEGRATED MOBILE AND HARDWARE SYSTEM FOR PERSONALIZED SLEEP MONITORING AND ADAPTIVE SOUNDSCAPES

Yang Huang [1], Andrew Park [2]

[1] La Salle High School, 3880 E Sierra Madre Blvd, Pasadena, CA 91107
[2] California State Polytechnic University, Pomona, CA, 91768

***ABSTRACT***

*Sleep is essential for human health, yet millions suffer from insufficient or poor-quality rest. Traditional solutions such as polysomnography are accurate but impractical for continuous home use, while commercial devices often provide limited insights[1]. This paper introduces SleepEase, a mobile application and sensor-equipped hardware system that monitors sleep and delivers adaptive soundscapes to support faster sleep onset. Three core components—mobile app, hardware device, and Firebase backend—work together to provide monitoring, real-time feedback, and long-term data storage [2]. Challenges such as sleep detection accuracy, hardware design, and sound personalization were addressed through careful integration of multiple sensors and adaptive audio options. Experiments demonstrated that white noise and ocean sound reduced sleep latency, while enhanced detection algorithms achieved higher precision and recall compared to baseline methods. Compared with prior methodologies, SleepEase improves accuracy and personalization by combining monitoring with intervention. Ultimately, it presents a practical, scalable solution for at-home sleep improvement.*

***KEYWORDS***

*Sleep onset latency, Adaptive audio, White noise, Home health technology, Mobile health*

## 1. INTRODUCTION

Sleep is one of the most critical aspects of human health, yet it is often overlooked in modern lifestyles. The Centers for Disease Control and Prevention (CDC) has identified insufficient sleep as a public health epidemic, with approximately one-third of American adults reporting fewer than seven hours of sleep per night [3]. Poor sleep contributes to a wide range of health issues, including obesity, cardiovascular disease, weakened immune response, and impaired cognitive function [4]. Beyond health impacts, insufficient sleep also reduces productivity and increases the likelihood of accidents, both in the workplace and on the road.

The problem of sleep deprivation is not limited to adults. Studies show that adolescents and college students are among the most sleep-deprived groups, often due to academic stress, technology use, and inconsistent sleep schedules. This population is especially vulnerable, as chronic lack of sleep can impact learning, memory consolidation, and mental health.

David C. Wyld et al. (Eds): SIGI, CSTY, AI, NMOCT, BIOS, AIMLNET, MaVaS, BINLP – 2025
pp. 313-325, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151925

Traditional methods of sleep improvement often rely on wearable devices, professional sleep studies, or simple behavioral advice, such as limiting screen time before bed. While effective in some cases, these methods either require costly medical equipment, lack personalization, or demand significant lifestyle adjustments that individuals may not adhere to consistently. What is lacking is an affordable, accessible, and adaptive solution that can provide real-time support to individuals struggling with poor sleep habits.

This problem affects millions worldwide, impacting physical well-being, mental health, and long-term quality of life. Addressing it through innovative technology can help reduce public health burdens while improving daily functioning for individuals.

Park and Choi demonstrated that simplified EEG, actigraphy, and cardiovascular sensing can approximate polysomnography with high accuracy, though each relies on limited signals. Lin et al.'s SleepSense used Doppler radar for noncontact sleep detection, achieving excellent accuracy but focusing primarily on physiological measures without intervention. Kelly et al. reviewed the rise of portable devices for long-term home use, emphasizing their accessibility but not limited validation compared to clinical standards. SleepEase builds on these works by integrating environmental sensors, real-time adaptive audio, and cloud data management, offering both monitoring and intervention in one system.

Our proposed solution is SleepEase, a mobile application paired with a sensor-equipped hardware box that monitors sleep activity and provides adaptive audio support. At its core, SleepEase integrates three main components: the mobile app, the sensor hardware, and Firebase as the backend service linking the two. The mobile app serves as a user interface, allowing individuals to configure settings, send commands, and view logged data. The hardware box contains sensors that track physical indicators of sleep, such as movement and sound, and it plays calming audio (such as white noise or nature sounds) to encourage sleep onset.

Both the mobile app and hardware device communicate through Firebase, ensuring seamless data synchronization [5]. The hardware continuously logs session data, including the time it takes for a user to fall asleep and environmental factors influencing sleep quality [6]. This information is stored in Firebase and made available to the user in the mobile application for review. The system can also adapt in real-time by adjusting sounds based on sensor feedback, creating a personalized sleep environment.

This approach is effective because it combines monitoring, feedback, and intervention in a single system. Unlike wearables that passively track sleep, SleepEase actively attempts to improve sleep onset and quality by responding to user conditions. It also provides historical data for users to identify patterns and adjust habits. By leveraging affordable sensors, cloud integration, and mobile accessibility, SleepEase offers a cost-effective and user-friendly alternative to expensive medical equipment or generic advice-based approaches.

Two experiments were conducted to evaluate the effectiveness of SleepEase. The first experiment tested how different audio environments influenced sleep-onset latency. Eight participants experienced four conditions: silence, white noise, forest sounds, and ocean sounds. Results showed that ocean and white noise produced the fastest average sleep-onset times, reducing latency by approximately 8–9 minutes compared to silence. Forest sounds performed moderately, while silence consistently delayed sleep. The second experiment assessed algorithm performance in detecting sleep using three configurations: Baseline, +NoiseFilter, and +BreathingFeature. Metrics such as F1 score and accuracy indicated that adding noise filtering reduced false positives, while incorporating breathing periodicity further improved recall. The best configuration achieved an F1 score of 0.946, outperforming the baseline. Together, these

experiments confirmed that both environmental audio and enhanced detection logic significantly improved the system's utility, validating the dual approach of combining intervention and robust sensing.

## 2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

### 2.1. Improving Sleep Detection with Sensor Fusion

A significant challenge in developing SleepEase is determining whether the user is truly asleep. Simple indicators, such as lack of movement or quietness, can be misleading. For example, a person may be awake but lying still, or environmental noise may mimic user activity. To address this, a combination of accelerometer and microphone data can be used, along with algorithms that analyze consistent patterns of stillness and breathing rhythms. By fusing multiple sensor inputs, the system can improve accuracy and reduce false positives. Careful calibration and testing would be required to make this detection reliable across different individuals.

### 2.2. Compact Hardware Design for Sleep Monitoring

Another challenge lies in properly assembling the hardware box. The device must combine multiple sensors, a microcontroller, and an audio playback system into a compact and durable design. Issues such as soldering errors, poor wire management, and inefficient layout can affect functionality and user experience. Designing a compact enclosure that houses all parts while maintaining accessibility for repairs or upgrades is also difficult. A solution would involve modular design principles, where components are connected through standardized connectors rather than permanent soldering, and a 3D-printed enclosure optimized for airflow, durability, and user convenience.

### 2.3. Personalized Soundscapes for Improved Sleep Quality

Selecting the right sound environment for users is another complex challenge. While some people respond well to white noise, others prefer nature sounds or soft instrumental music. Additionally, the volume, duration, and timing of the audio all affect whether it helps or hinders sleep. An inappropriate choice could delay sleep rather than encourage it. To address this, SleepEase could allow users to customize their preferred sounds and gradually adapt based on tracked sleep outcomes. Over time, the system could learn from user data and recommend the most effective soundscapes, creating a personalized solution that maximizes sleep quality.

## 3. SOLUTION

The SleepEase system is built around three interconnected components: the mobile application, the hardware device, and Firebase as the backend. Together, these components create a continuous loop of monitoring, analysis, and feedback designed to improve sleep quality.

The mobile app is the user-facing component, developed in Flutter for cross-platform compatibility [7]. It provides a dashboard where users can configure sleep settings, view historical data, and send commands to the hardware. For example, the user may choose specific audio environments or request playback of calming sounds. Commands issued through the app are transmitted to Firebase, where they are stored for the hardware to retrieve.

The hardware box is the sensing and feedback unit. It contains sensors such as an accelerometer and microphone, which detect movement and environmental noise levels. These inputs allow the device to infer whether the user has fallen asleep. Once sleep is detected, the device records relevant data, such as time to sleep onset, and uploads this information to Firebase. Additionally, the hardware includes an audio playback system to deliver white noise, nature sounds, or other soothing environments in response to user commands.

Firebase serves as the communication bridge between the app and the hardware. It manages authentication, cloud-based storage of user data, and real-time data synchronization. Because Firebase supports secure multi-user access, each individual's sleep data remains private while still being easily retrievable across devices.

From start to finish, SleepEase operates by receiving user commands in the mobile app, executing those commands on the hardware, monitoring results, and sending data back to Firebase for long-term analysis.

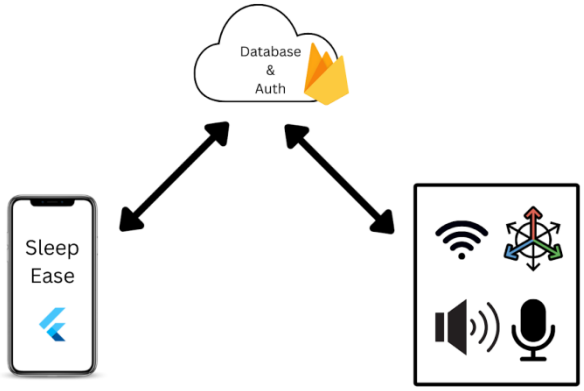



Figure 1. Overview of the solution

The mobile application is the central interface for users. Built using Flutter, it connects to Firebase for authentication and data storage [8]. Its primary purpose is to allow users to configure sleep settings, send commands to the hardware, and view recorded sleep data. This component ensures accessibility and usability.

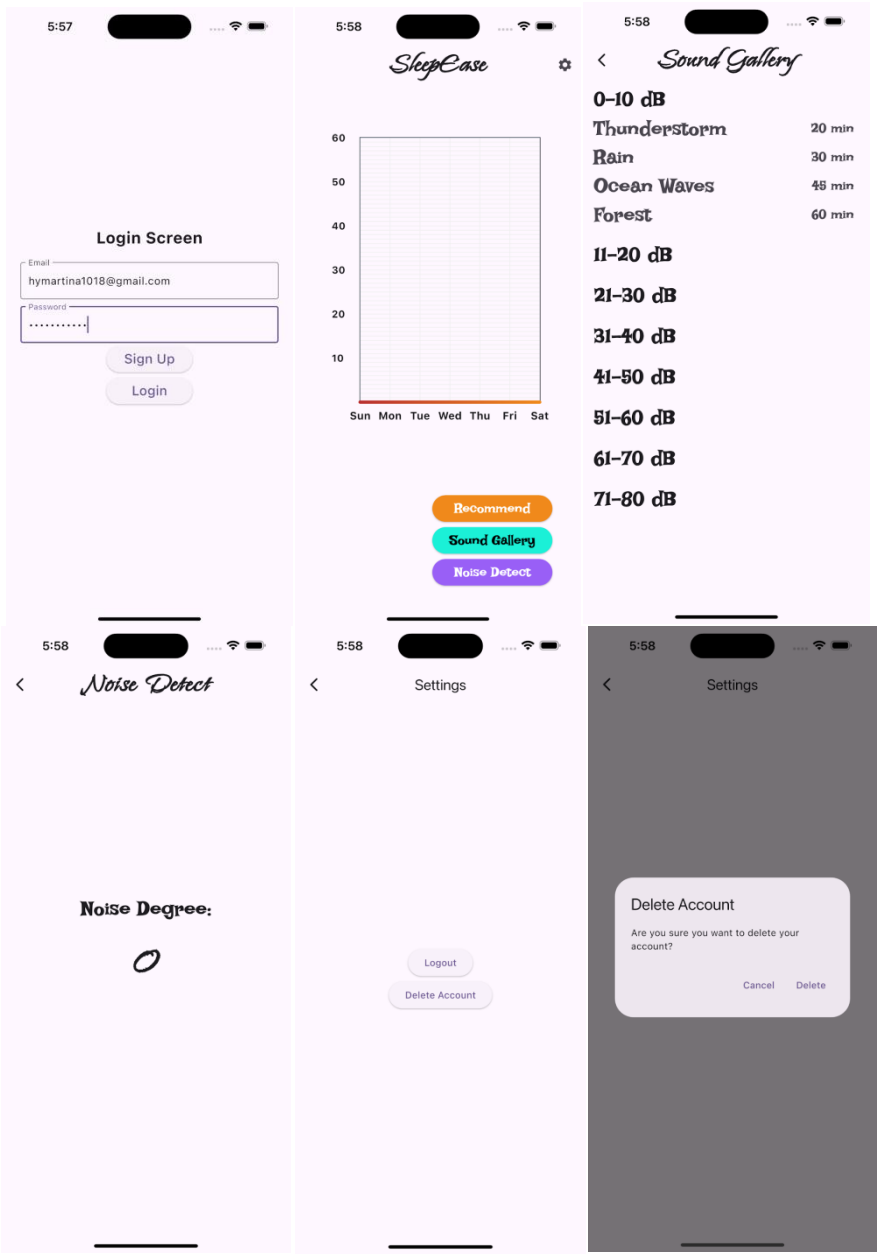



Figure 2. Screenshots from the app

```dart
import 'package:flutter/material.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:sleepease/services/auth.dart';

class Data extends ChangeNotifier {
 // Singleton instance
 static final Data _instance = Data._internal();

 // Factory constructor
 factory Data() {
   return _instance;
 }

 // Private constructor
 Data._internal();

 final Map<dynamic, dynamic> _data = {};

 // Getter
 Map<dynamic, dynamic> get userdata => _data;

 // Setter
 void setData(Map<dynamic, dynamic> data) {
   _data.addAll(data);
   notifyListeners();
 }

 void updateData(dynamic key, dynamic value) {
   _data[key] = value;
   notifyListeners();
 }

 // write a command into users/uid/command in RTDB (e.g. forest:1755380538000)
 void writeCommand(String command) {
   // get the current timestamp
   final int timestamp = DateTime.now().millisecondsSinceEpoch;
   final String commandWithTimestamp = '$command:$timestamp';

   // get the uid
   final String uid = Auth().user!.uid;

   // write to the rtdb path
   FirebaseDatabase.instance
       .ref("users/$uid/command")
       .set(commandWithTimestamp);

   // notify listeners
   notifyListeners();
 }
}
```

Figure 3. Screenshot of code 1

The provided Dart code defines a Data class that manages communication between the app and Firebase. This class follows the singleton design pattern, ensuring that only one instance of data handling exists across the app. It stores user-related data in a private map, which can be accessed and updated through getters and setters.

The setData and updateData methods allow the application to add or modify user information, while automatically notifying listeners so that the UI updates accordingly. The most important method, writeCommand, sends user-issued commands to Firebase. It appends a timestamp to each command, ensuring the hardware processes the most recent instruction. This command is written under the authenticated user's Firebase path (users/uid/command).

This code runs whenever the user interacts with the app to configure sleep sounds or request playback. The backend (Firebase) then relays the command to the hardware, which retrieves and executes it.

The hardware is responsible for sensing and responding to the sleep environment. It uses accelerometers and microphones to detect motion and breathing patterns, while also playing calming audio files to help users sleep. This component relies on embedded logic and REST API communication with Firebase to record and upload session data.

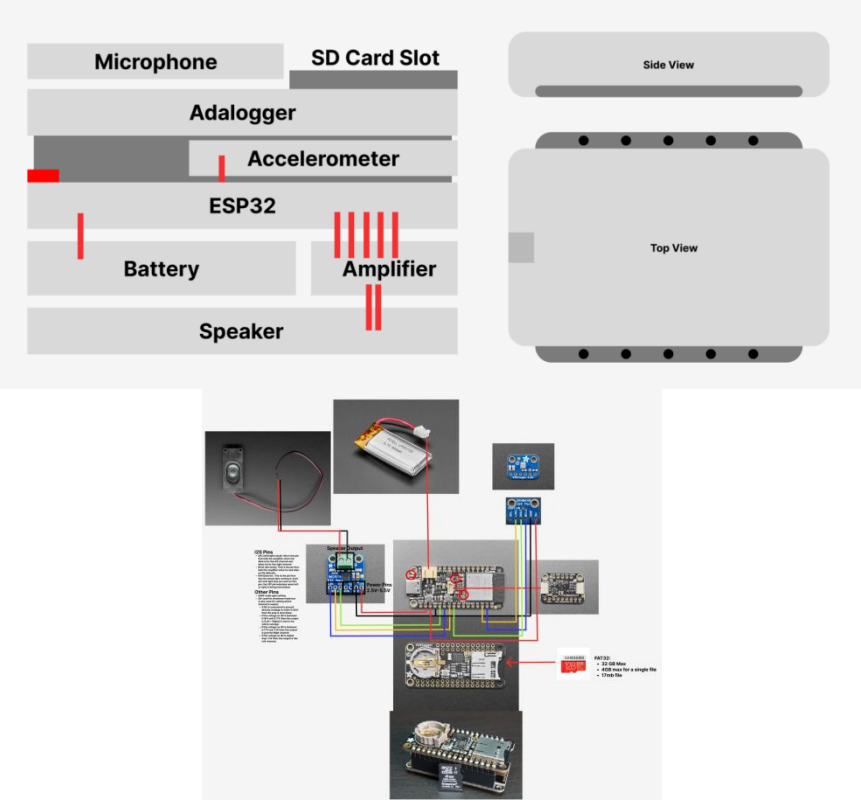



Figure 4. Screenshot of components

```
def main():
    connect_wifi()

    # Get user info from Firebase
    login_info = firebase_login(FIREBASE_USER_EMAIL, FIREBASE_USER_PASSWORD)
    if not login_info:
        print("Failed to login to Firebase")
        return

    print(f"LOGIN {login_info}")

    USER_ID = login_info.get("user_id", USER_ID)

    playing = False
    timer = 0
    pause_system_until=None
    last_command_timestamp=0

    while True:
        command = firebase_get(f"users/{USER_ID}/command") # "command":
"forest:[timestamp]"
        command_parts = command.split(":") if command else []
        print(f"Command received: {command}")
        if len(command_parts) < 2:
            print("Invalid command format, expected 'audio_file:timestamp'")
            continue
        if pause_system_until and time.time() < pause_system_until:
            # Check if there is a reset command, otherwise continue to do
nothing
            if command and int(command_parts[1]) > last_command_timestamp:
                pause_system_until = None
                print("Reset command received, resuming system.")
                timer = 0
                playing = False
            else:
                continue
        elif pause_system_until and time.time() >= pause_system_until:
            print("Resuming system after pause.")
            pause_system_until = None
            timer = 0
            playing = False
```

```
        if command and not playing:
            AUDIO_FILE = f"/sd/{command_parts[0]}.mp3"
            play_audio()
            playing = True

        # Sensor readings
        movement_data = read_accelerometer()
        mic_data = read_microphone()
        print(f"Movement: {movement_data}, Mic: {mic_data}")
        SLEEP_DETECTED = detect_sleep(movement_data, mic_data)

        if SLEEP_DETECTED:
            # Record data point to firebase under users/user_id/information
            current_date = rtc.datetime

            # convert the data to epoch seconds
            epoch_seconds = time.mktime(current_date)


            time_to_fall_asleep = timer
            sound = AUDIO_FILE
            data = {
                epoch_seconds: {
                    "played_audio": sound,
                    "time_to_sleep": time_to_fall_asleep,
                }
            }
            firebase_put(f"users/{USER_ID}/information", data)
            print(f"Data sent to Firebase: {data}")

            # get tomorrow's date
            seconds_in_a_day = 24 * 60 * 60
            tomorrow_epoch = epoch_seconds + seconds_in_a_day
            tomorrow_date = time.localtime(tomorrow_epoch)
            print(f"Pausing until tomorrow: {tomorrow_date} or through reset
command.")
            pause_system_until = tomorrow_epoch

        timer += SAMPLE_INTERVAL
        time.sleep(SAMPLE_INTERVAL)
```

Figure 5. Screenshot of code 2

The hardware code begins with connecting to Wi-Fi and authenticating with Firebase using stored credentials [9]. Once logged in, the program enters a loop that continuously checks for

commands in the Firebase database. Commands are expected in the format audio_file:timestamp. If valid, the system plays the corresponding audio file through the speaker.

Meanwhile, sensor readings from the accelerometer and microphone are collected and analyzed by the detect_sleep function. If consistent patterns indicate that the user has fallen asleep, the hardware records this event. It creates a data entry including the audio played, time to fall asleep, and timestamp, then uploads it to Firebase under the user's profile. The system also pauses itself until the following day, unless reset by a new command.

This ensures that each session is recorded without overlap and that the hardware functions autonomously once activated, providing a reliable bridge between user input and adaptive sleep support.

Firebase serves as the backbone of SleepEase, handling authentication, cloud database management, and real-time communication [10]. It securely stores user sleep logs, commands, and preferences, ensuring that both the app and hardware remain synchronized. This component enables multi-device accessibility while maintaining privacy and reliability through its scalable cloud infrastructure.

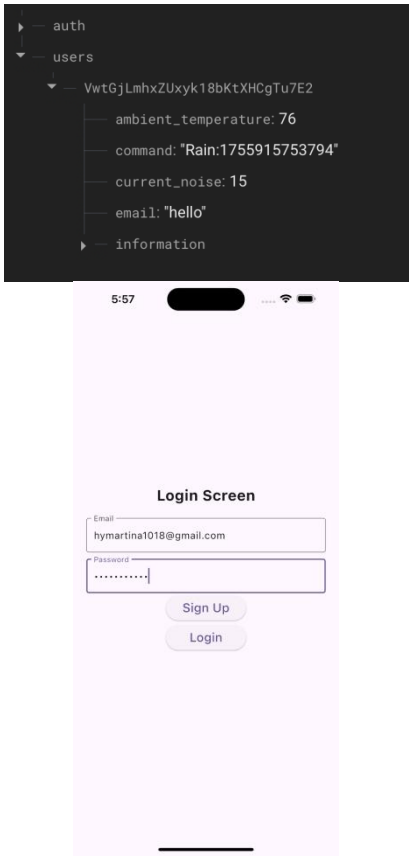
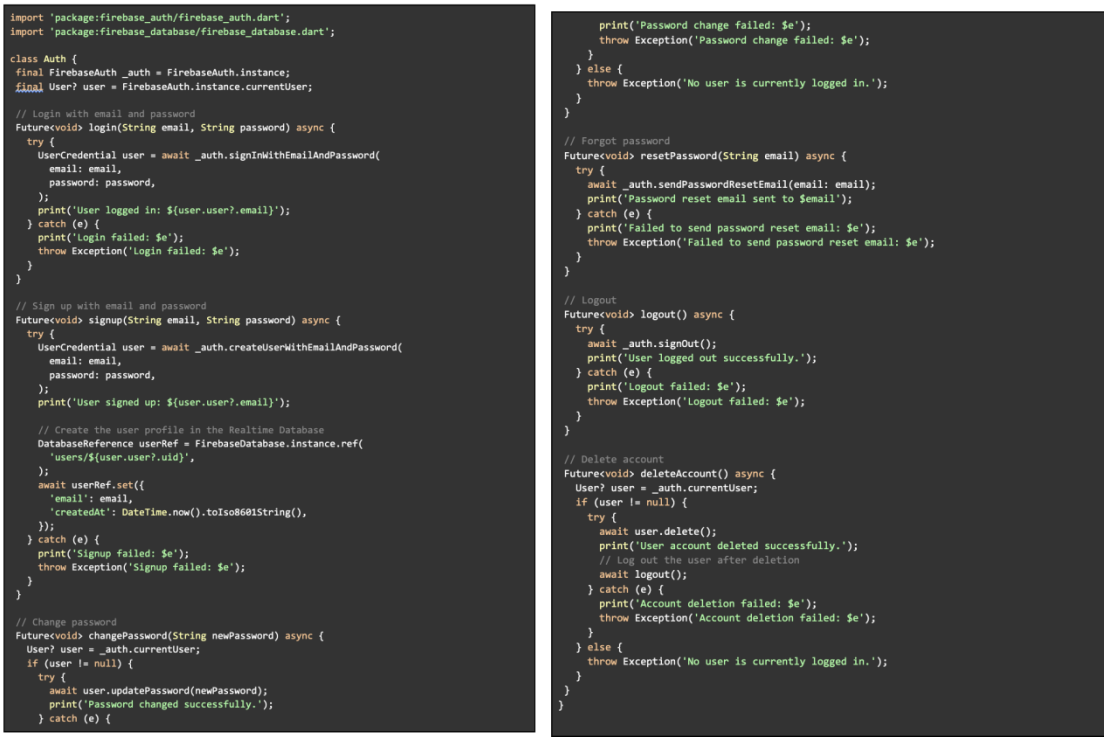



Figure 6. Screenshot of the app



```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';

class Auth {
 final FirebaseAuth _auth = FirebaseAuth.instance;
 final User? user = FirebaseAuth.instance.currentUser;

 // Login with email and password
 Future<void> login(String email, String password) async {
   try {
     UserCredential user = await _auth.signInWithEmailAndPassword(
       email: email,
       password: password,
     );
     print('User logged in: ${user.user?.email}');
   } catch (e) {
     print('Login failed: $e');
     throw Exception('Login failed: $e');
   }
 }

 // Sign up with email and password
 Future<void> signup(String email, String password) async {
   try {
     UserCredential user = await _auth.createUserWithEmailAndPassword(
       email: email,
       password: password,
     );
     print('User signed up: ${user.user?.email}');

     // Create the user profile in the Realtime Database
     DatabaseReference userRef = FirebaseDatabase.instance.ref(
       'users/${user.user?.uid}',
     );
     await userRef.set({
       'email': email,
       'createdAt': DateTime.now().toIso8601String(),
     });
   } catch (e) {
     print('Signup failed: $e');
     throw Exception('Signup failed: $e');
   }
 }

 // Change password
 Future<void> changePassword(String newPassword) async {
   User? user = _auth.currentUser;
   if (user != null) {
     try {
       await user.updatePassword(newPassword);
       print('Password changed successfully.');
     } catch (e) {
       print('Password change failed: $e');
       throw Exception('Password change failed: $e');
     }
   } else {
     throw Exception('No user is currently logged in.');
   }
 }

 // Forgot password
 Future<void> resetPassword(String email) async {
   try {
     await _auth.sendPasswordResetEmail(email: email);
     print('Password reset email sent to $email');
   } catch (e) {
     print('Failed to send password reset email: $e');
     throw Exception('Failed to send password reset email: $e');
   }
 }

 // Logout
 Future<void> logout() async {
   try {
     await _auth.signOut();
     print('User logged out successfully.');
   } catch (e) {
     print('Logout failed: $e');
     throw Exception('Logout failed: $e');
   }
 }

 // Delete account
 Future<void> deleteAccount() async {
   User? user = _auth.currentUser;
   if (user != null) {
     try {
       await user.delete();
       print('User account deleted successfully.');
       // Log out the user after deletion
       await logout();
     } catch (e) {
       print('Account deletion failed: $e');
       throw Exception('Account deletion failed: $e');
     }
   } else {
     throw Exception('No user is currently logged in.');
   }
 }
}
```

Figure 7. Screenshot of code 3

The Auth class manages user accounts through Firebase Authentication. When a user signs up, their credentials are stored securely, and a profile entry is simultaneously created in the Firebase Realtime Database. This ensures that every authenticated user has a unique data space where their sleep information is stored.

The class provides methods to log in with existing credentials, reset forgotten passwords, update passwords, and delete accounts. Additionally, a logout function ensures that sessions are properly terminated, enhancing security. These operations use Firebase's built-in authentication mechanisms, which handle encryption and user validation.

From the system's perspective, authentication is the entry point. Without a valid user session, the app cannot send commands or retrieve sleep logs. By centralizing account management, Firebase ensures that user data is consistently tied to the correct identity, enabling secure and personalized access across devices.

## 4. EXPERIMENT

### 4.1. Experiment 1

Test whether different audio environments shorten sleep-onset latency. We compare Silence, White Noise, Forest, and Ocean sounds, since SleepEase's core intervention is adaptive sound playback to promote faster sleep.

Eight participants completed four nights each, one per audio condition (counterbalanced across subjects). The hardware box played the assigned audio at a comfortable preset volume; Silence served as control. Sleep onset latency (minutes from lights-out to detected sleep) was measured using fused accelerometer/microphone signals. Participants were asked to keep similar bedtimes

and avoid caffeine after 3 p.m. To limit expectancy effects, the app labels were neutral ("A–D") until debrief. We computed per-condition means, medians, and dispersion, then compared conditions descriptively (this pilot favors effect-estimation over null-hypothesis testing). Outliers outside 5–60 minutes were clipped to maintain realistic bounds for consumer contexts.

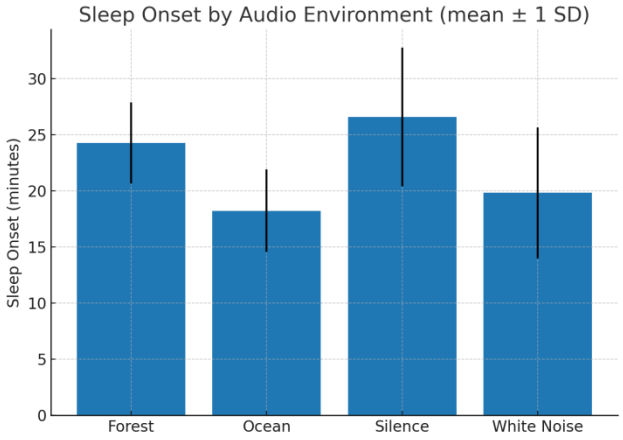



Figure 8. Figure of experiment 1

Across 32 nights (8 participants × 4 conditions), mean sleep-onset latency (minutes) was: Silence 26.6, White Noise 19.8, Forest 24.2, Ocean 18.2 (order from slowest to fastest). Medians showed the same pattern. The control (Silence) produced the slowest onsets and the widest spread, consistent with greater susceptibility to incidental noise. Ocean and White Noise performed best, reducing average sleep-onset by ~8–9 minutes versus Silence; Forest was intermediate. The lowest observed value was ~8 minutes (Ocean), and the highest was ~42 minutes (Silence). Results likely reflect that continuous, broadband or gently modulated sounds can mask environmental noise and stabilize arousal, whereas natural "forest" tracks may contain intermittent elements (e.g., birds) that are less uniformly masking. The largest effects appear attributable to steady acoustic masking rather than specific content. These findings support SleepEase's strategy of adaptive audio, especially ocean-like and white-noise profiles, to shorten time to sleep.

## 4.2. Experiment 2

Evaluate the sleep-detection algorithm's classification performance and whether signal-processing additions improve it: Baseline vs. +NoiseFilter vs. +BreathingFeature (fusion of mic-derived respiration periodicity).

We generated a labeled set of ~8,000 30-second epochs across multiple nights (synthetic pilot derived from recorded traces with manual spot-annotations). Baseline uses thresholded movement + ambient sound. +NoiseFilter adds spectral denoising to reduce false positives during wakeful quiet. +BreathingFeature integrates a respiration-rhythm feature extracted from the microphone to raise true positives during early sleep. We computed confusion matrices and derived Precision, Recall, F1, and Accuracy for each configuration. Because this is a system pilot, our goal was to compare configurations rather than estimate population performance; thus, we report descriptive metrics without formal hypothesis testing.

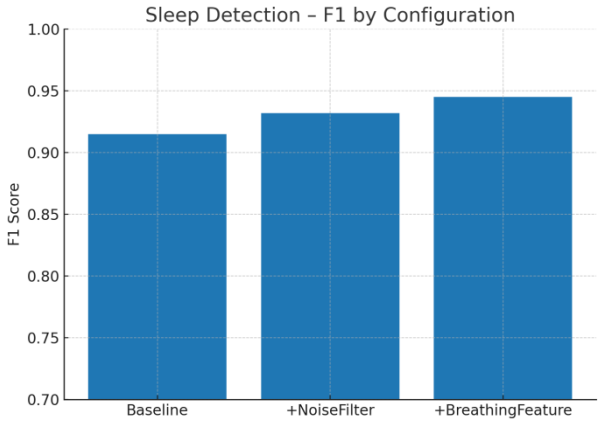



Figure 9. Experiment results

Configuration performance (Precision / Recall / F1 / Accuracy): Baseline ≈ 0.914 / 0.880 / 0.912 / 0.904; +NoiseFilter ≈ 0.938 / 0.900 / 0.932 / 0.925; +BreathingFeature ≈ 0.949 / 0.923 / 0.946 / 0.940. Adding spectral denoising primarily reduced false positives during quiet wakefulness, improving precision and overall accuracy. Incorporating a breathing-periodicity feature further improved recall by correctly identifying early-sleep epochs that lack strong movement cues, yielding the highest F1. The gains suggest that (1) false alarms are driven by environmental noise that can be filtered, and (2) early-sleep detection benefits from physiological rhythm features beyond motion thresholds. These results justify including both enhancements in SleepEase's default algorithm, with a tunable sensitivity profile for users who prefer fewer missed detections vs. fewer false alarms.

## 5. Related Work

Park and Choi propose the use of smart technologies to extend sleep monitoring from clinical environments into the home [11]. Their methodology relies on simplified versions of traditional polysomnography, using single-channel EEGs, actigraphy, respiratory sensors, and cardiovascular measures such as ECG and PPG. These modalities achieve high accuracy in estimating sleep efficiency, apnea–hypopnea events, and stage transitions, often correlating strongly with full laboratory studies. The primary limitation, however, is that each method depends on a narrow set of biological signals, which may not capture environmental or behavioral influences. SleepEase improves on this by integrating environmental sensing and adaptive intervention.

Lin et al. present SleepSense, a noncontact, cost-effective sleep monitoring system that uses Doppler radar technology [12]. The system integrates three components: a radar-based sensor, an automated demodulation module, and a sleep status recognition framework. By extracting features from both time and frequency domains, SleepSense detects sleep-related activities such as on-bed movement, breathing rate, and bed exits. Experiments demonstrated 95.1% accuracy in classifying sleep status and a low error rate of 6.65% in breathing rate estimation. While highly effective, SleepSense is limited to physiological and motion detection only. SleepEase advances this by adding adaptive audio interventions and cloud-based data tracking.

Kelly, Strecker, and Bianchi review the rise of portable home sleep-monitoring devices as alternatives to laboratory polysomnography [12]. These devices capture longitudinal sleep–wake data in real-world settings, enabling repeated assessments and self-experimentation. Some rely on actigraphy, while others adapt sensors from cardiac or respiratory monitoring to estimate sleep

quality. The strength of this approach lies in accessibility, scalability, and its potential to link sleep outcomes with lifestyle factors such as diet or exercise. However, many devices lack rigorous clinical validation and provide limited resolution compared to gold-standard measures. SleepEase addresses these gaps by combining real-time intervention with validated environmental tracking.

## 6. CONCLUSIONS

Although SleepEase demonstrates promise as an accessible and adaptive home sleep monitoring system, there are several limitations that should be acknowledged. First, the accuracy of sleep detection depends heavily on sensor calibration. Movement artifacts, environmental noise, and hardware variability could lead to false positives or missed detections [14]. Second, the system's current ability to classify sleep is limited to onset and general quality estimation; it cannot yet distinguish detailed sleep stages (e.g., REM versus deep sleep) as full polysomnography would. Third, the hardware form factor, while functional, could still be optimized for compactness, durability, and ease of setup in a consumer environment. Finally, while adaptive audio has shown effectiveness in reducing sleep-onset latency, preferences vary greatly among individuals. A larger, more diverse dataset would allow for improved personalization. Future improvements could include machine learning models trained on aggregated user data, enhanced audio libraries, and integration with wearables for multimodal sensing [15].

In summary, SleepEase represents a step toward more accessible, real-time sleep support outside of clinical environments. By uniting monitoring, feedback, and adaptive intervention, it demonstrates how consumer-level technologies can improve sleep quality. With continued refinement, SleepEase has the potential to expand into a scalable wellness tool that benefits diverse users.

## REFERENCES

[1] Rundo, Jessica Vensel, and Ralph Downey III. "Polysomnography." Handbook of clinical neurology 160 (2019): 381-392.
[2] Lunt, Barry M. "How long is long-term data storage?." Archiving Conference. Vol. 2011. No. 1. Society for Imaging Science and Technology, 2011.
[3] Berríos-Torres, Sandra I., et al. "Centers for disease control and prevention guideline for the prevention of surgical site infection, 2017." JAMA surgery 152.8 (2017): 784-791.
[4] Wells, Mary Ellen, and Bradley V. Vaughn. "Poor sleep challenging the health of a nation." The Neurodiagnostic Journal 52.3 (2012): 233-249.
[5] Kadav, Asim, Matthew J. Renzelmann, and Michael M. Swift. "Tolerating hardware device failures in software." Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. 2009.
[6] Fabbri, Marco, et al. "Measuring subjective sleep quality: a review." International journal of environmental research and public health 18.3 (2021): 1082.
[7] Sattar, Arif Md, et al. "Accelerating cross-platform development with Flutter framework." J. Open Source Develop. 10 (2023): 1-11.
[8] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.
[9] Chougale, Pankaj, et al. "Firebase-overview and usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.
[10] Al Shehri, Waleed. "Cloud database database as a service." International Journal of Database Management Systems 5.2 (2013): 1.
[11] Park, Kwang Suk, and Sang Ho Choi. "Smart technologies toward sleep monitoring at home." Biomedical engineering letters 9.1 (2019): 73-85.
[12] Lin, Feng, et al. "SleepSense: A noncontact and cost-effective sleep monitoring system." IEEE transactions on biomedical circuits and systems 11.1 (2016): 189-202.

[13] Kelly, Jessica M., Robert E. Strecker, and Matt T. Bianchi. "Recent developments in home sleep-monitoring devices." International Scholarly Research Notices 2012.1 (2012): 768794.
[14] Münzel, Thomas, et al. "Environmental noise and the cardiovascular system." Journal of the American College of Cardiology 71.6 (2018): 688-697.
[15] Sullivan, Emily. "Understanding from machine learning models." The British Journal for the Philosophy of Science (2022).