Project Report Format

1. **INTRODUCTION**

   1.1 Project Overview

   1.2 Purpose

2. **LITERATURE SURVEY**

   2.1 Existing problem

   2.2 References

   2.3 Problem Statement Definition

3. **IDEATION & PROPOSED SOLUTION**

   3.1 Empathy Map Canvas

   3.2 Ideation & Brainstorming

4. **REQUIREMENT ANALYSIS**

   4.1 Functional requirement

   4.2 Non-Functional requirements

5. **PROJECT DESIGN**

   5.1 Data Flow Diagrams & User Stories

   5.2 Solution Architecture

6. **PROJECT PLANNING & SCHEDULING**

   6.1 Technical Architecture

   6.2 Sprint Planning & Estimation

   6.3 Sprint Delivery Schedule

7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**

   7.1 Feature 1

   7.2 Feature 2

   7.3 Database Schema (if Applicable)

8. **PERFORMANCE TESTING**

   8.1 Performace Metrics

9. **RESULTS**

   9.1 Output Screenshots

10. **ADVANTAGES & DISADVANTAGES**

11. **CONCLUSION**

12. **FUTURE SCOPE**

13. **APPENDIX**

    Source Code

    GitHub & Project Demo Link

1. INTRODUCTION

1.1. Project Overview

A Sleep Tracking App For A Better Night's Rest

A project that demonstrates the use of Android Jetpack Compose to build a UI for a sleep tracking app. The app allows users to track their sleep. With the "Sleep Tracker" app, you can assess the quality of sleep they have had in a day. It has been time and again proven that a good quality sleep is pretty essential for effective functioning of both mind and body.

1.2. Purpose

The purpose of your app is to serve as a unique and engaging combination of a music player and a smart alarm clock. Users can set an alarm for a specified time, typically 8:00 AM, and when the alarm triggers, they are presented with a math problem to solve before they can turn it off. This feature encourages users to wake up more alert and engage their cognitive functions early in the morning. Additionally, the app includes an 8-hour timer that counts down before the alarm sounds, providing users with a visual indicator of the remaining time. It also features a music player that plays an MP3 file, which can be activated by the user. In case of an incorrect answer to the math problem, the app displays a toast message, saying "Wrong Answer," to prompt users to retry.

2. **LITERATURE SURVEY**

2.1. Existing problem

To create a working sleep tracking app with the ability to play soothing music and set a smart alarm.

2.2. References

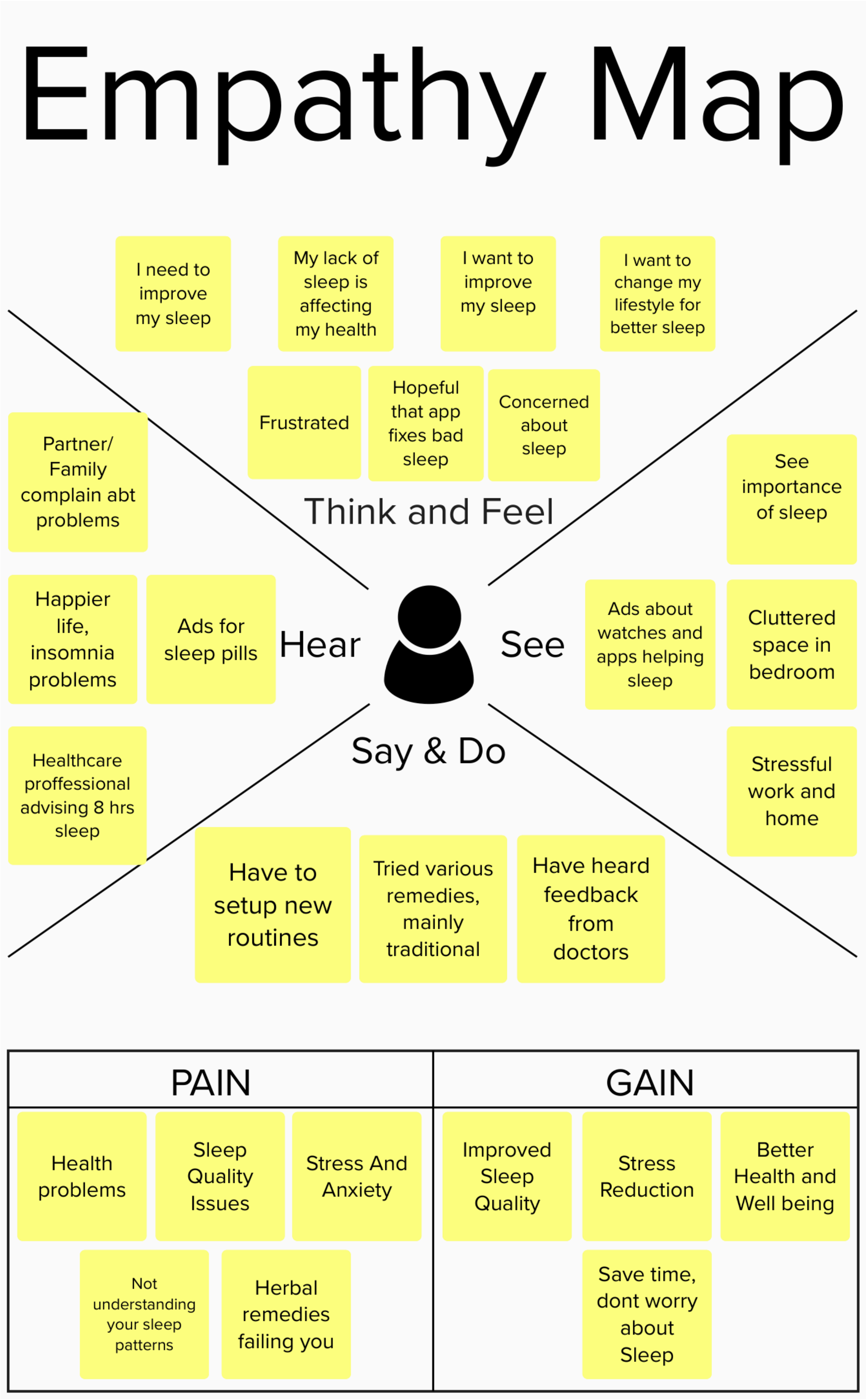
https://smartinternz.com/Student/guided_project_workspace/587559
https://developer.android.com/jetpack/compose/tutorial
https://www.youtube.com/watch?v=pXeWBPXxkK8

2.3. Problem Statement Definition

“How might we create a working sleep tracking app?"

3. **IDEATION & PROPOSED SOLUTION**

3.1. Empathy Map Canvas



Empathy Map

Think and Feel

I need to improve my sleep

My lack of sleep is affecting my health

I want to improve my sleep

I want to change my lifestyle for better sleep

Frustrated

Hopeful that app fixes bad sleep

Concerned about sleep

Hear

Partner/ Family complain abt problems

Happier life, insomnia problems

Ads for sleep pills

Healthcare proffessional advising 8 hrs sleep

See

See importance of sleep

Ads about watches and apps helping sleep

Cluttered space in bedroom

Stressful work and home

Say & Do

Have to setup new routines

Tried various remedies, mainly traditional

Have heard feedback from doctors

PAIN

Health problems

Sleep Quality Issues

Stress And Anxiety

Not understanding your sleep patterns

Herbal remedies failing you

GAIN

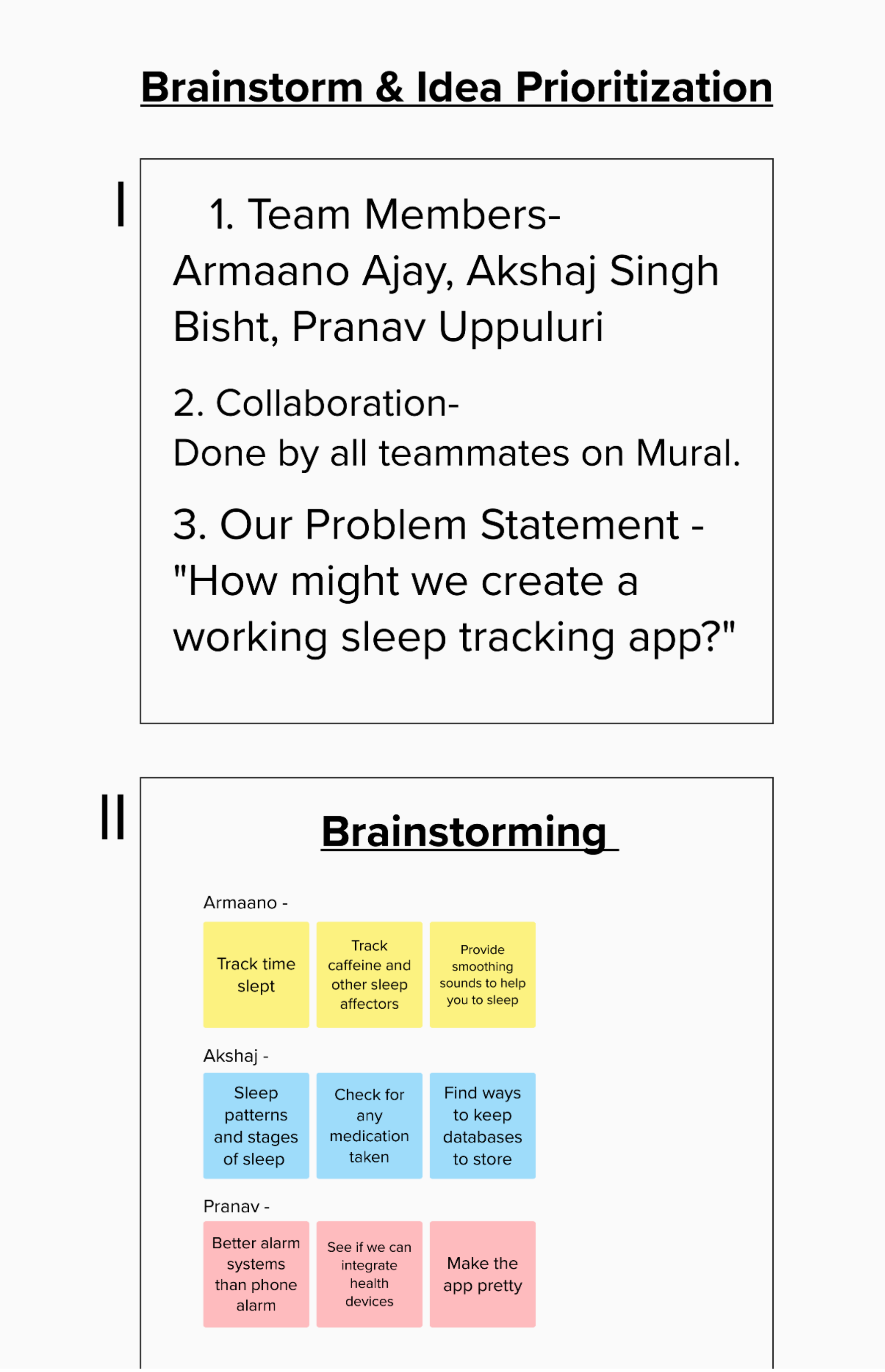
Improved Sleep Quality

Stress Reduction

Better Health and Well being

Save time, dont worry about Sleep

3.2. Ideation & Brainstorming



Brainstorm & Idea Prioritization

I

1. Team Members-
Armaano Ajay, Akshaj Singh Bisht, Pranav Uppuluri

2. Collaboration-
Done by all teammates on Mural.

3. Our Problem Statement -
"How might we create a working sleep tracking app?"

II

Brainstorming

Armaano -

- Track time slept
- Track caffeine and other sleep affectors
- Provide smoothing sounds to help you to sleep

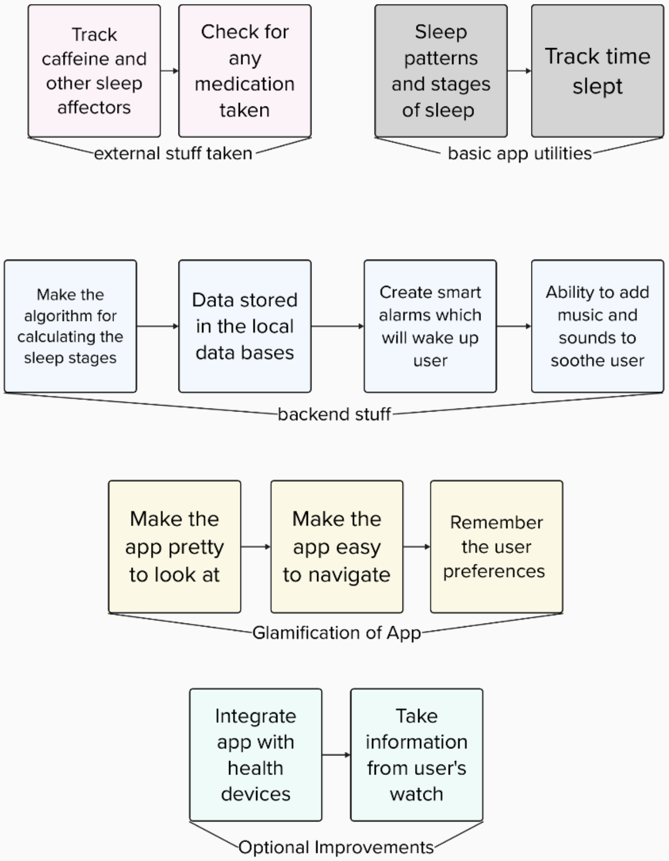
Akshaj -

- Sleep patterns and stages of sleep
- Check for any medication taken
- Find ways to keep databases to store

Pranav -

- Better alarm systems than phone alarm
- See if we can integrate health devices
- Make the app pretty

Grouping



Track caffeine and other sleep affectors

Check for any medication taken

external stuff taken

Sleep patterns and stages of sleep

Track time slept

basic app utilities

Make the algorithm for calculating the sleep stages

Data stored in the local data bases

Create smart alarms which will wake up user

Ability to add music and sounds to soothe user

backend stuff

Make the app pretty to look at

Make the app easy to navigate

Remember the user preferences

Glamification of App

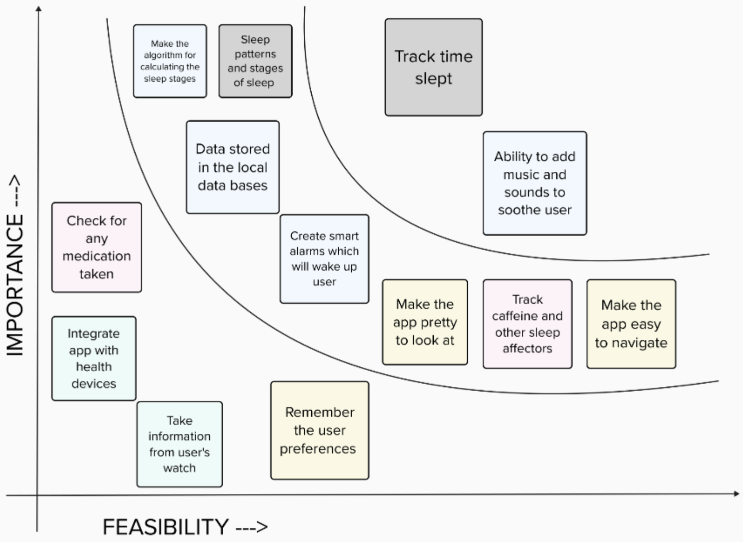
Integrate app with health devices

Take information from user's watch

Optional Improvements

III

Idea Prioritization



4. **REQUIREMENT ANALYSIS**

4.1. Functional requirement

The primary function of our app is to track sleep time. Users can initiate sleep tracking when they are ready to go to bed, and the app will monitor their sleep duration and quality throughout the night. Sleep data, including the time of falling asleep and waking up, as well as any disturbances during the night, are recorded and displayed to the user in an easily comprehensible format. To enhance the user experience, we have incorporated a smart alarm feature that allows users to set an alarm for the morning. What makes our app particularly intelligent is its ability to wake the user up at the optimal point in their sleep cycle, ensuring a more refreshing wake-up experience. The alarm uses data collected during sleep tracking to wake the user at the most suitable time within a set alarm window. With a user-friendly and intuitive interface, our app provides valuable insights into sleep patterns and aims to improve overall sleep quality.

4.2. Non-Functional requirements

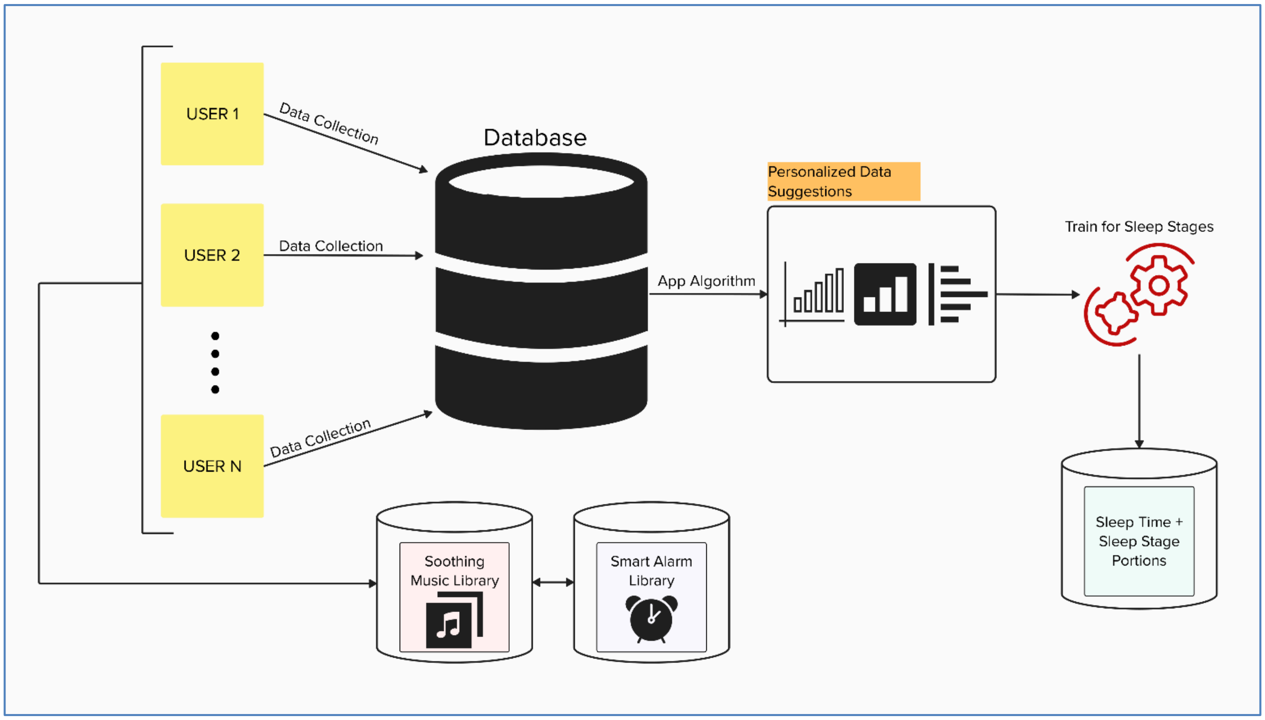
Our non-functional blocks include making a wearable device functionality to the app. Including sleep stage algorithms which change the sleep tracking time.

5. PROJECT DESIGN

5.1. Data Flow Diagrams & User Stories

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application byentering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	I can receive confirmationemail & click confirm	High	Sprint-1
	Login	USN-3	As a user, I can log into the application byentering email & password		High	Sprint-1
	Sleep Tracking and	USN-4	As a user, I want to track my sleep patterns by inputting my bedtime and wake-up time.	The app successfully records my sleep	High	Sprint-1

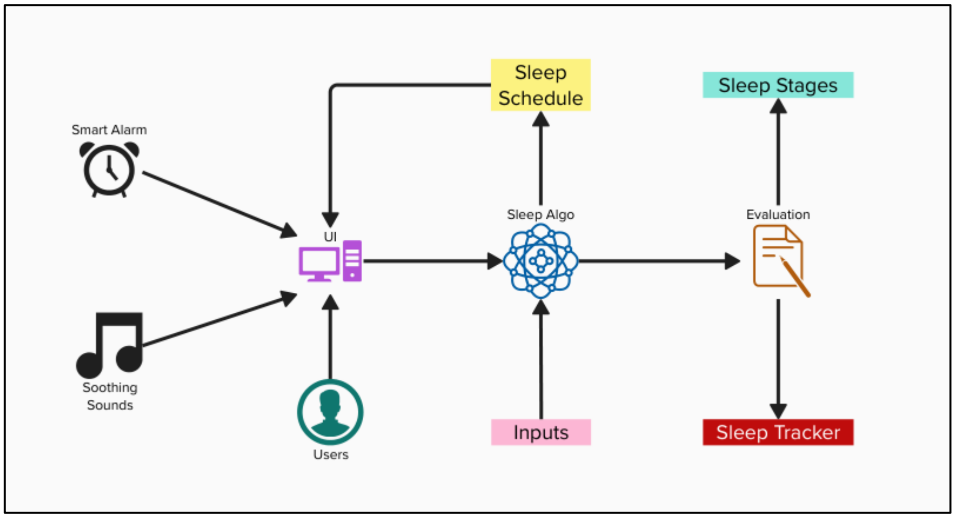
	Improvement			patterns.		
		USN-5	As a user, I want to receive personalized recommendations for improving my sleep quality based on the analysis of my sleep data.	The app provides me with tailored sleep improvement suggestions.	Medium	Sprint-2
	Relaxation and Sound Library	USN-6	As a user, I want access to a library of calming sounds and white noise to create a soothing sleep environment.	The app allows me to select and customize calming sounds for a peaceful sleep environment.	Medium	Sprint-1
	Smart Alarm and Wake-Up	USN-7	As a user, I want to set a smart alarm that wakes me up at the optimal point in my sleep cycle for a more refreshing wake-up.	The smart alarm effectively wakes me up at the optimal time.	Medium	Sprint-1
Data Analyst	Requires Data	USN-8	As a data analyst, I want to get my sleep data so that I can check my tendencies.	The app gives accurate data to the analyst.	Low	Sprint-3

5.2. Solution Architecture

Solution Architecture:

Our sleep tracking app's solution architecture is a user-centric system with a well-designed user interface for intuitive interaction. It features a robust back end hosted on cloud infrastructure to handle data storage and processing, ensuring scalability as our user base grows. The app integrates sleep tracking data from various sources, utilizes AI for personalized insights, offers a content library for relaxation, and prioritizes data security and user privacy. With support for multiple platforms, external APIs, and user feedback mechanisms, our architecture is agile, adaptable, and poised for ongoing enhancements, enabling a seamless and effective approach to sleep tracking and improvement.

Solution Architecture Diagram:



6. PROJECT PLANNING & SCHEDULING

6.1. Technical Architecture

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

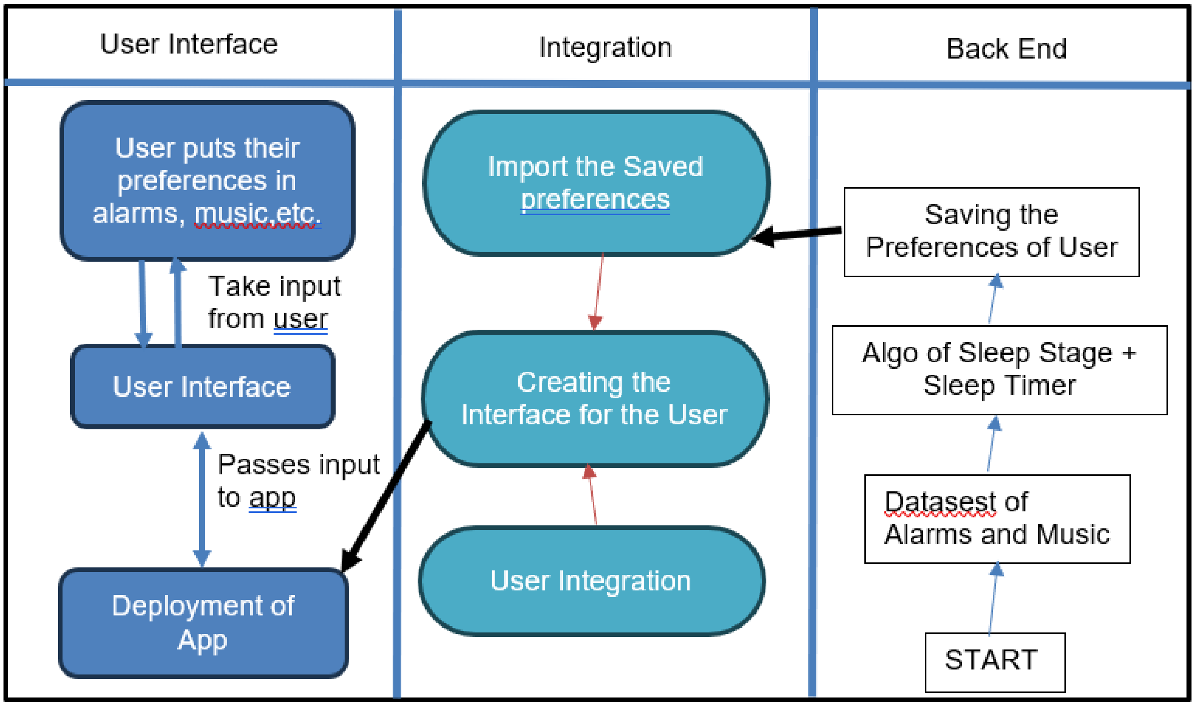

User Interface
Integration
Back End
User puts their preferences in alarms, music,etc.
Take input from user
User Interface
Passes input to app
Deployment of App
Import the Saved preferences
Creating the Interface for the User
User Integration
Saving the Preferences of User
Algo of Sleep Stage + Sleep Timer
Datasest of Alarms and Music
START

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web UI, Mobile App, Chatbot etc.	(Mobile) Android App
2.	Application Logic-1	Logic for a process in the application (Sleep Tracking)	Kotlin / JetpackCompose
3.	Application Logic-2	Logic for a process in the application (Sleep Stage Tracker)	JavaScript
4.	Application Logic-3	Logic for a process in the application (Smart Alarm)	JavaScript
5.	Database	Data Type, Configurations etc. (For Alarms and Music)	NoSQL
6.	Cloud Database	Database Service on Cloud	For future iterations

7.	File Storage	File storage requirements	Local Filesystem

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented,use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier,Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g. use ofload balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of theapplication (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

6.2. Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the sleep tracking app.	2	High	Akshaj
Sprint-1	Development environment	USN-2	Gather necessary resources and libraries for sleep data collection and storage.	3	High	Armaano
Sprint-2	Data collection	USN-3	Implement data collection methods, either through user input or integration with wearable devices.	5	High	Akshaj
Sprint-2	Data preprocessing	USN-4	Ensure the integration with wearable devices for automatic sleep tracking is functional.	4	High	Pranav
Sprint-3	User Data Analysis	USN-5	Store user sleep data in a secure manner, ensuring data privacy. Begin implementing basic sleep data analysis features for insights into sleep patterns.	3	High	Armaano

Sprint-3	User Experience Enhancement	USN-6	Develop the user interface for the app, focusing on intuitive design and data presentation. Implement a smart alarm feature for optimized wake-up times based on sleep data.	6	medium	Armaano
Sprint-4	Content Library	USN-7	Create a library of calming sounds, white noise, and educational content related to sleep.	4	medium	Pranav
Sprint-5	Testing & quality assurance	USN-8	Conduct thorough testing of the app to identify and report any issues or bugs. Fine-tune the app's algorithms and features based on user feedback and testing results.	1	medium	Akshaj

6.3. Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

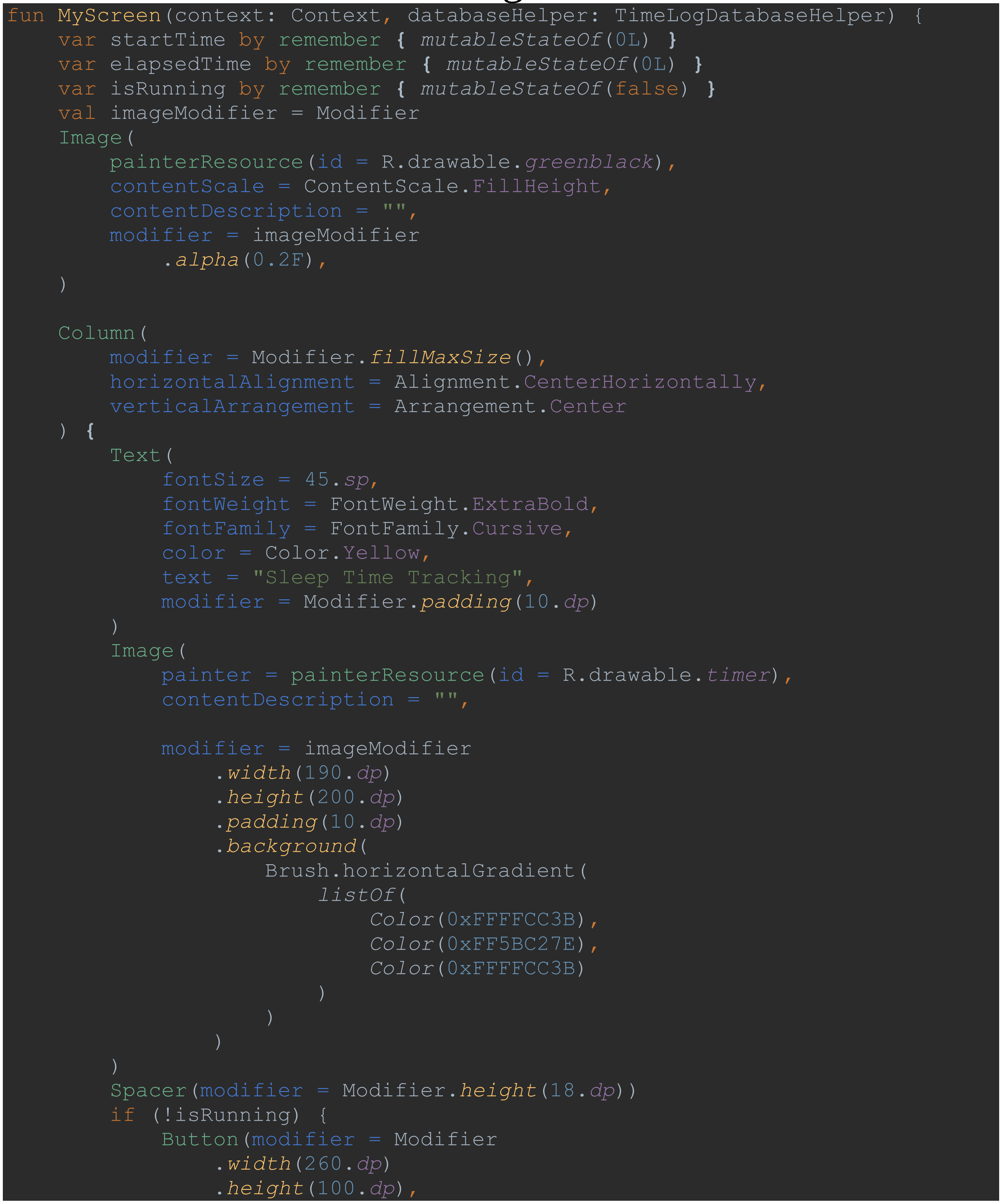
Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	5	1 Day	28 Oct 2023	29 Oct 2023	27	9 Nov 2023
Sprint-2	9	3 Days	29 Oct 2023	1 Nov 2023		
Sprint-3	9	2 Days	1 Nov 2023	3 Nov 2023		
Sprint-4	4	2 Days	3 Nov 2023	5 Nov 2023		
Sprint-5	1	1 Days	5 Nov 2023	6 Nov 2023		

AV = 27/10 = 2.7

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1. Feature 1

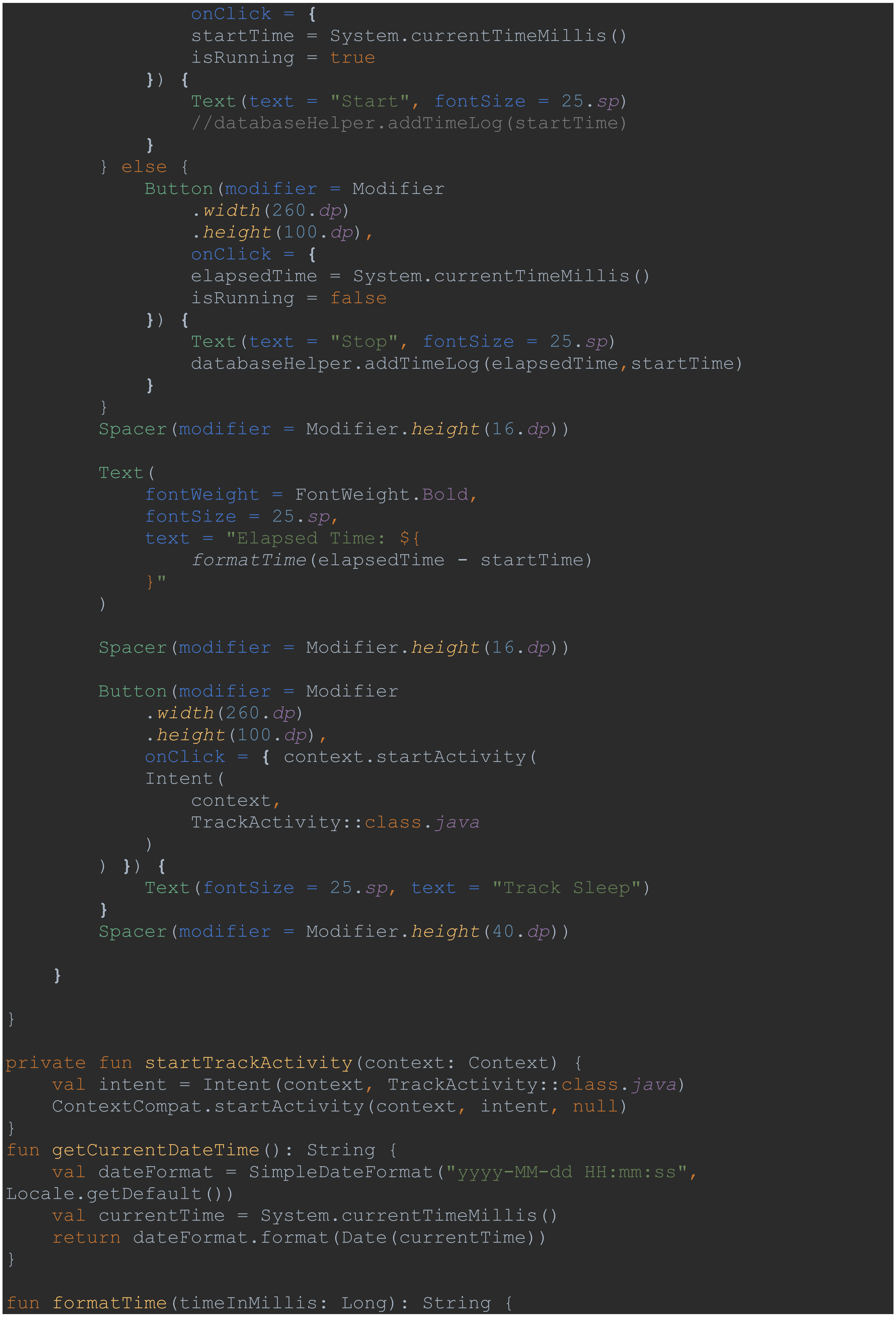
The main feature of tracking sleep time.
This feature has the following code-



```
fun MyScreen(context: Context, databaseHelper: TimeLogDatabaseHelper) {
    var startTime by remember { mutableStateOf(0L) }
    var elapsedTime by remember { mutableStateOf(0L) }
    var isRunning by remember { mutableStateOf(false) }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.greenblack),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.2F),
    )

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            fontSize = 45.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.Yellow,
            text = "Sleep Time Tracking",
            modifier = Modifier.padding(10.dp)
        )
        Image(
            painter = painterResource(id = R.drawable.timer),
            contentDescription = "",

            modifier = imageModifier
                .width(190.dp)
                .height(200.dp)
                .padding(10.dp)
                .background(
                    Brush.horizontalGradient(
                        listOf(
                            Color(0xFFFFCC3B),
                            Color(0xFF5BC27E),
                            Color(0xFFFFCC3B)
                        )
                    )
                )
        )
        Spacer(modifier = Modifier.height(18.dp))
        if (!isRunning) {
            Button(modifier = Modifier
                .width(260.dp)
                .height(100.dp),
```



```
                onClick = {
                startTime = System.currentTimeMillis()
                isRunning = true
            }) {
                Text(text = "Start", fontSize = 25.sp)
                //databaseHelper.addTimeLog(startTime)
            }
        } else {
            Button(modifier = Modifier
                .width(260.dp)
                .height(100.dp),
                onClick = {
                elapsedTime = System.currentTimeMillis()
                isRunning = false
            }) {
                Text(text = "Stop", fontSize = 25.sp)
                databaseHelper.addTimeLog(elapsedTime,startTime)
            }
        }
        Spacer(modifier = Modifier.height(16.dp))

        Text(
            fontWeight = FontWeight.Bold,
            fontSize = 25.sp,
            text = "Elapsed Time: ${
                formatTime(elapsedTime - startTime)
            }"
        )

        Spacer(modifier = Modifier.height(16.dp))

        Button(modifier = Modifier
            .width(260.dp)
            .height(100.dp),
            onClick = { context.startActivity(
            Intent(
                context,
                TrackActivity::class.java
            )
        ) }) {
            Text(fontSize = 25.sp, text = "Track Sleep")
        }
        Spacer(modifier = Modifier.height(40.dp))

    }

}

private fun startTrackActivity(context: Context) {
    val intent = Intent(context, TrackActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
fun getCurrentDateTime(): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
    val currentTime = System.currentTimeMillis()
    return dateFormat.format(Date(currentTime))
}

fun formatTime(timeInMillis: Long): String {
    val hours = (timeInMillis / (1000 * 60 * 60)) % 24
    val minutes = (timeInMillis / (1000 * 60)) % 60
    val seconds = (timeInMillis / 1000) % 60
    return String.format("%02d:%02d:%02d", hours, minutes, seconds)
}
```

7.2. Feature 2

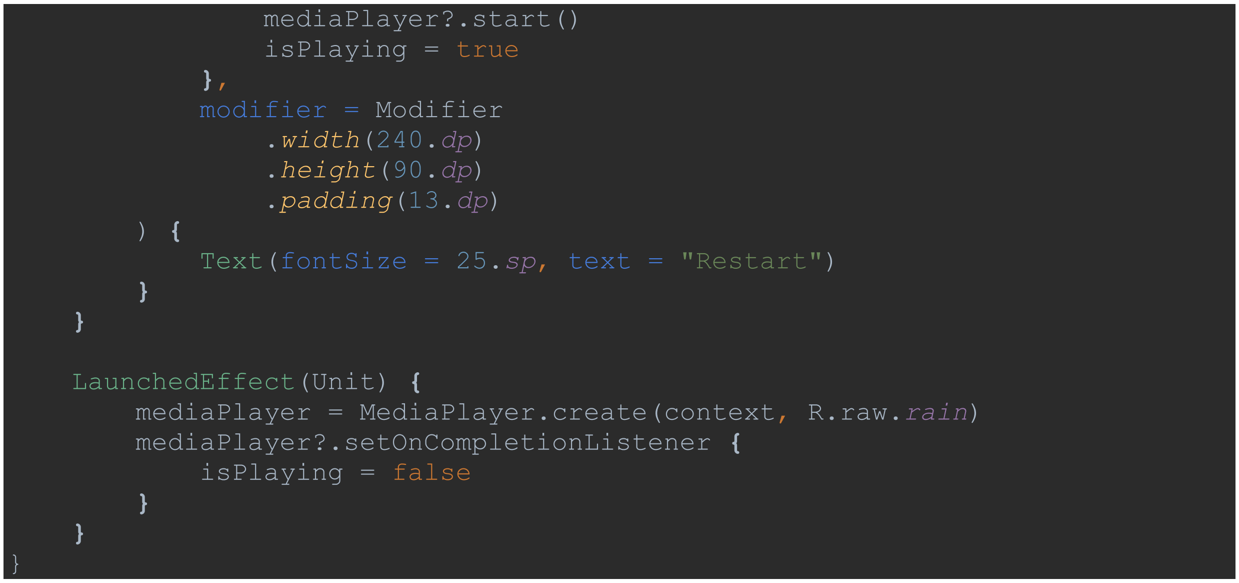
The second feature is Soothing Music.

The code –

```
fun MusicPlayerApp() {
    val context = LocalContext.current

    var mediaPlayer by remember { mutableStateOf<MediaPlayer?>(null) }
    var isPlaying by remember { mutableStateOf(false) }
    Image(
        painterResource(id = R.drawable.tiles),
        contentScale = ContentScale.Crop,
        contentDescription = "",
        modifier = Modifier
            .alpha(0.2F),
    )
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = R.drawable.msuic),
            contentDescription = "",
            modifier = Modifier
                .width(260.dp)
                .height(180.dp)
                .padding(10.dp)
        )
        Button(
            onClick = {
                if (isPlaying) {
                    mediaPlayer?.pause()
                } else {
                    mediaPlayer?.start()
                }
                isPlaying = !isPlaying
            },
            modifier = Modifier
                .width(260.dp)
                .height(100.dp)
                .padding(bottom = 16.dp)
        ) {
            Text(fontSize = 26.sp, text = if (isPlaying) "Pause" else
"Play")
        }

        Button(
            onClick = {
                mediaPlayer?.seekTo(0)
```



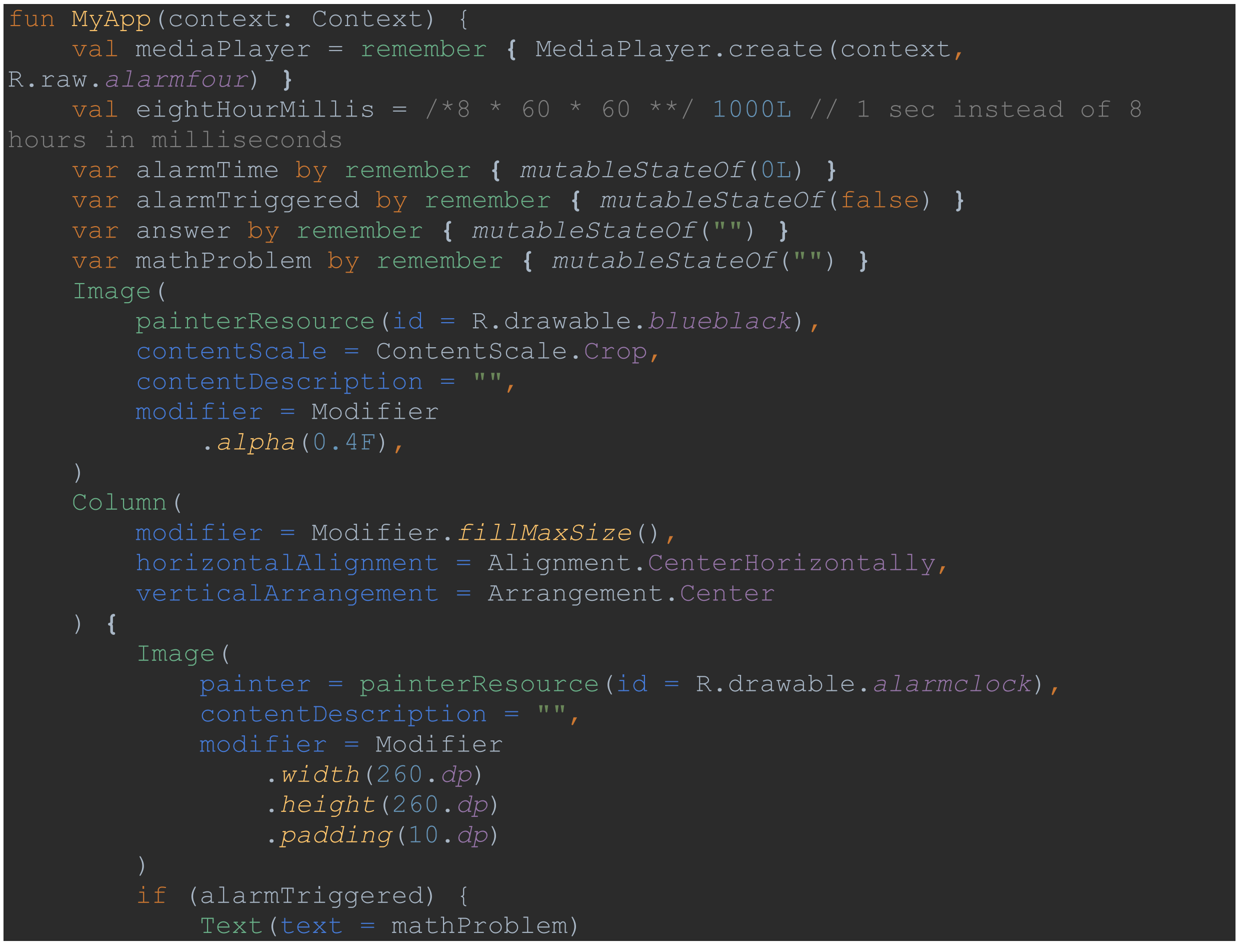
```
                mediaPlayer?.start()
                isPlaying = true
            },
            modifier = Modifier
                .width(240.dp)
                .height(90.dp)
                .padding(13.dp)
        ) {
            Text(fontSize = 25.sp, text = "Restart")
        }
    }

    LaunchedEffect(Unit) {
        mediaPlayer = MediaPlayer.create(context, R.raw.rain)
        mediaPlayer?.setOnCompletionListener {
            isPlaying = false
        }
    }
}
```

7.3. Feature 3

The third feature is the smart alarm.

The code –



```
fun MyApp(context: Context) {
    val mediaPlayer = remember { MediaPlayer.create(context,
R.raw.alarmfour) }
    val eightHourMillis = /*8 * 60 * 60 **/ 1000L // 1 sec instead of 8
hours in milliseconds
    var alarmTime by remember { mutableStateOf(0L) }
    var alarmTriggered by remember { mutableStateOf(false) }
    var answer by remember { mutableStateOf("") }
    var mathProblem by remember { mutableStateOf("") }
    Image(
        painterResource(id = R.drawable.blueblack),
        contentScale = ContentScale.Crop,
        contentDescription = "",
        modifier = Modifier
            .alpha(0.4F),
    )
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Image(
            painter = painterResource(id = R.drawable.alarmclock),
            contentDescription = "",
            modifier = Modifier
                .width(260.dp)
                .height(260.dp)
                .padding(10.dp)
        )
        if (alarmTriggered) {
            Text(text = mathProblem)
            TextField(value = answer, onValueChange = { answer = it })
            Button(
                onClick = {
                    if (answer == getMathProblemAnswer(mathProblem)) {
                        alarmTime = 0L
                        answer = ""
                        alarmTriggered = false
                        stopAlarm(mediaPlayer)
                    } else {
                        answer = ""
                        Toast.makeText(context, "Wrong Answer",
Toast.LENGTH_LONG).show()
                    }
                },
                modifier = Modifier.padding(top = 16.dp)
            ) {
                Text(text = "Submit")
            }
        } else {
            val alarmTimeString = SimpleDateFormat("HH:mm:ss").format(
                Date(System.currentTimeMillis() + eightHourMillis)
            )
            Text(fontSize = 30.sp, text = "Alarm will ring at
$alarmTimeString")
            Spacer(modifier = Modifier.height(10.dp))
            Button(
                modifier = Modifier
                    .width(260.dp)
                    .height(100.dp),
                onClick = {
                    mathProblem = generateMathProblem()
                    alarmTriggered = true
                    alarmTime = System.currentTimeMillis() + eightHourMillis
                    startCountdownTimer(alarmTime -
System.currentTimeMillis(), mediaPlayer)
                }
            ) {
                Text(fontSize = 25.sp, text = "Set Alarm")
            }
        }
    }
}

private fun generateMathProblem(): String {
    val num1 = Random.nextInt(1, 11)
    val num2 = Random.nextInt(1, 11)
    return "$num1 + $num2 = ?"
}

private fun getMathProblemAnswer(mathProblem: String): String {
    val equationParts = mathProblem.split(" + ")
    val num1 = equationParts[0].toInt()
    val num2 = equationParts[1].split(" = ")[0].toInt()
    return (num1 + num2).toString()
}

private fun startAlarm(mediaPlayer: MediaPlayer) {
    // Start playing the alarm sound
    mediaPlayer.start()
}
```

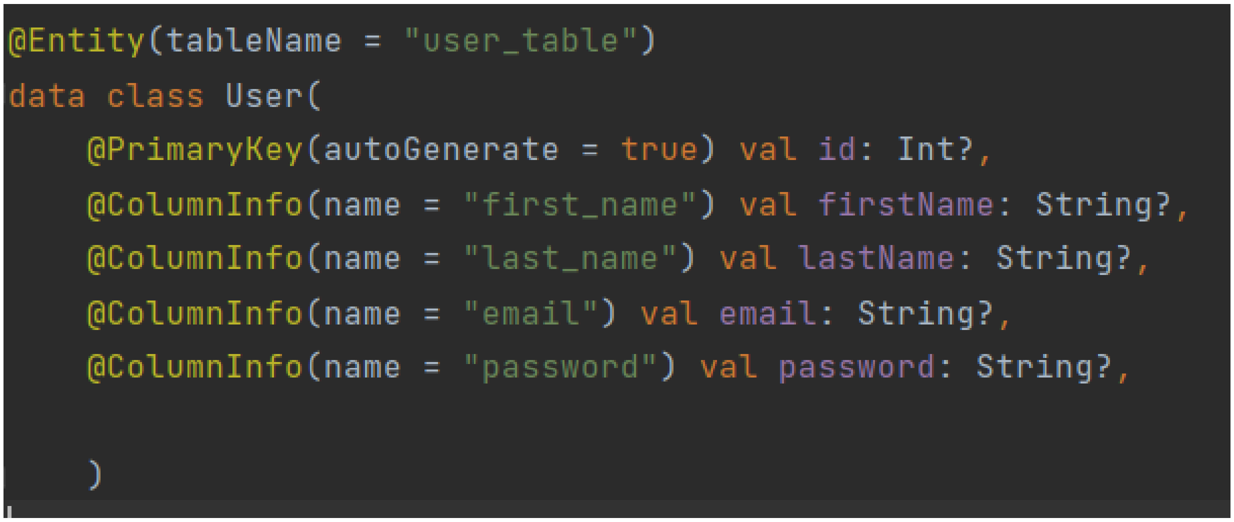
```
private fun stopAlarm(mediaPlayer: MediaPlayer) {
    // Stop playing the alarm sound
    mediaPlayer.stop()
    mediaPlayer.prepare()
}

private fun startCountdownTimer(duration: Long, mediaPlayer: MediaPlayer) {
    object : CountDownTimer(duration, 1000) {
        override fun onTick(millisUntilFinished: Long) {}

        override fun onFinish() {
            startAlarm(mediaPlayer)
        }
    }.start()
}
```

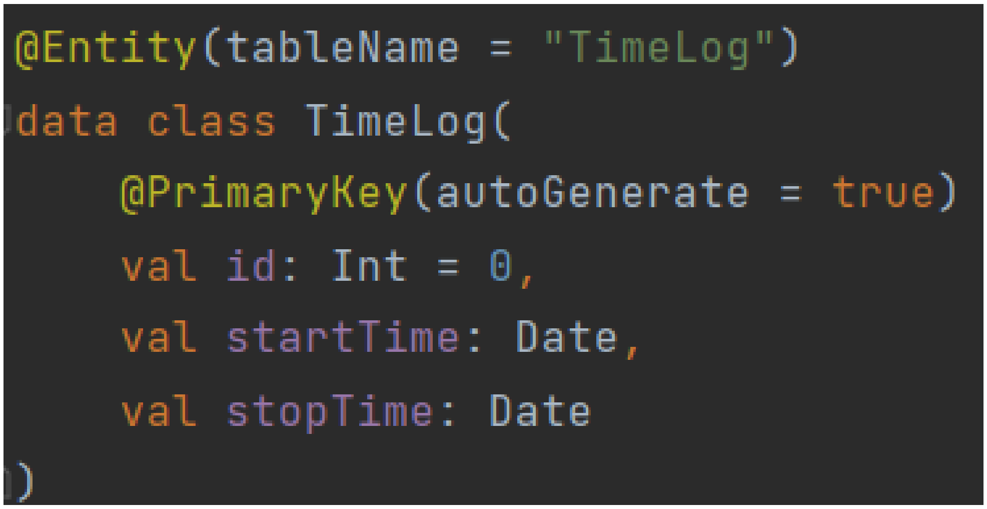
7.4. Database Schema (if Applicable)

The different databases we used-



```
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,

    )
```



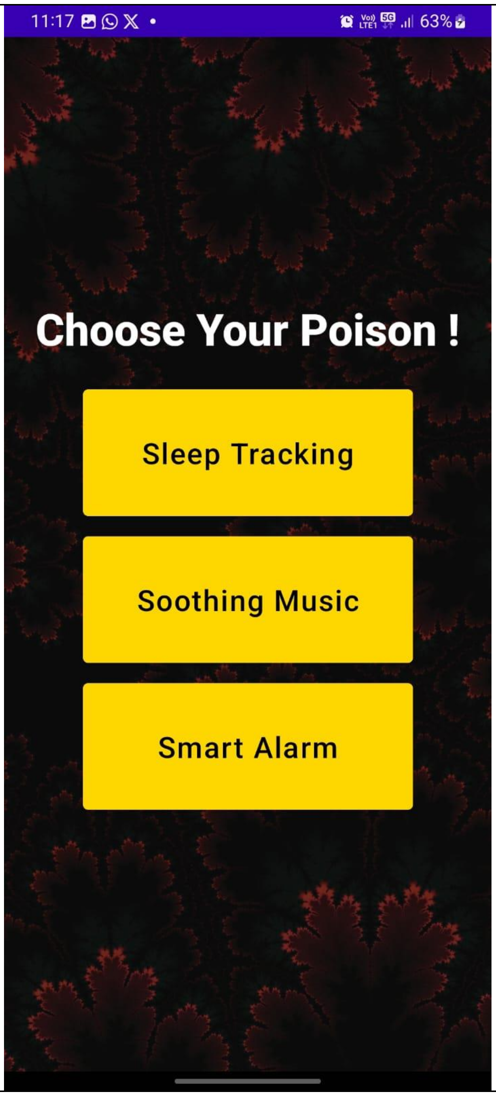
```
@Entity(tableName = "TimeLog")
data class TimeLog(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val startTime: Date,
    val stopTime: Date
)
```

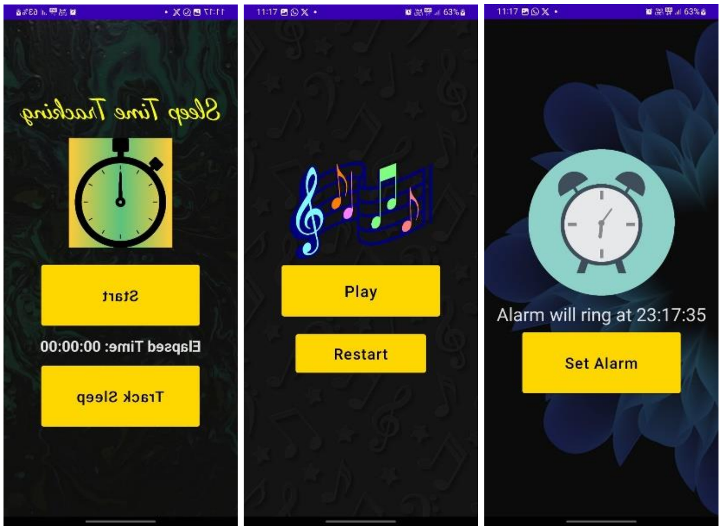
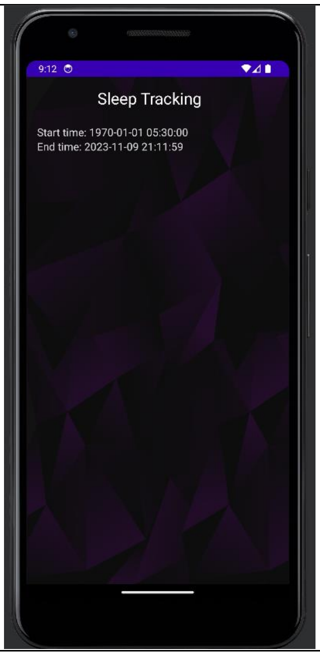
8. PERFORMANCE TESTING

8.1. Performace Metrics

Model Performance Testing:

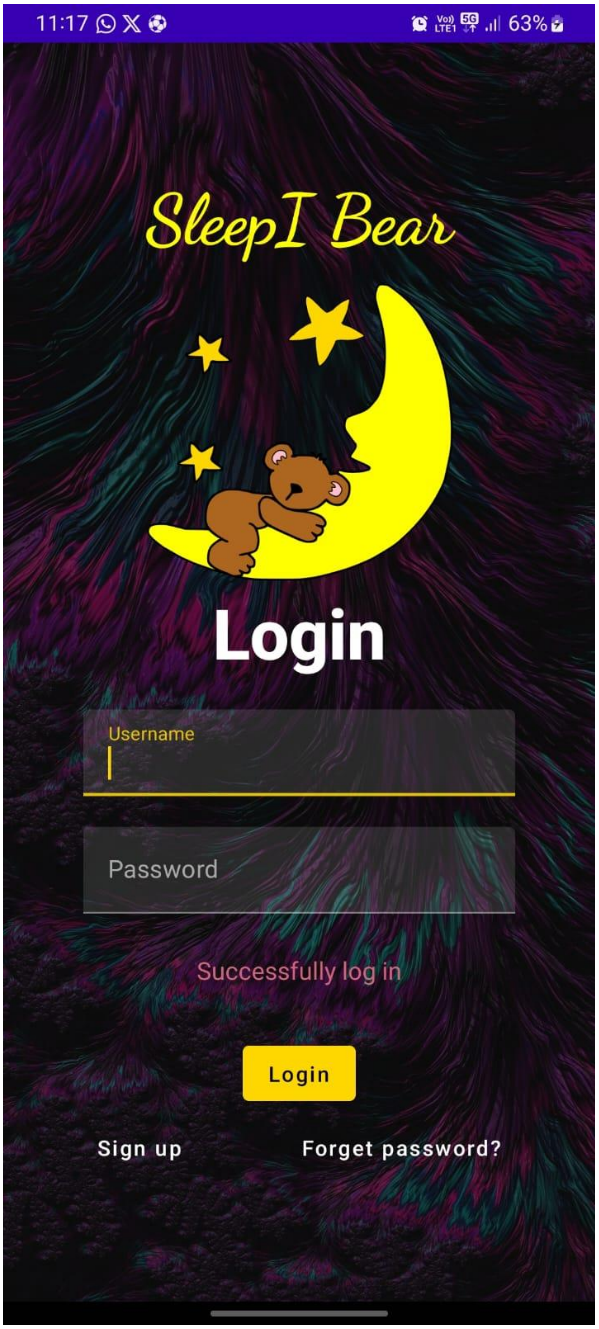
Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Screenshot / Values
1.	Dashboard design	 11:17 63% Choose Your Poison ! Sleep Tracking Soothing Music Smart Alarm
2.	Data Responsiveness	Time tracking is done instantly. The math question is selected from a database of addition of random numbers.
3.	Amount Data to Rendered (DB2 Metrics)	Userbase dataset, Time tracking Dataset, User Dataset
4.	Utilization of Data Filters	Data filters include similar username.

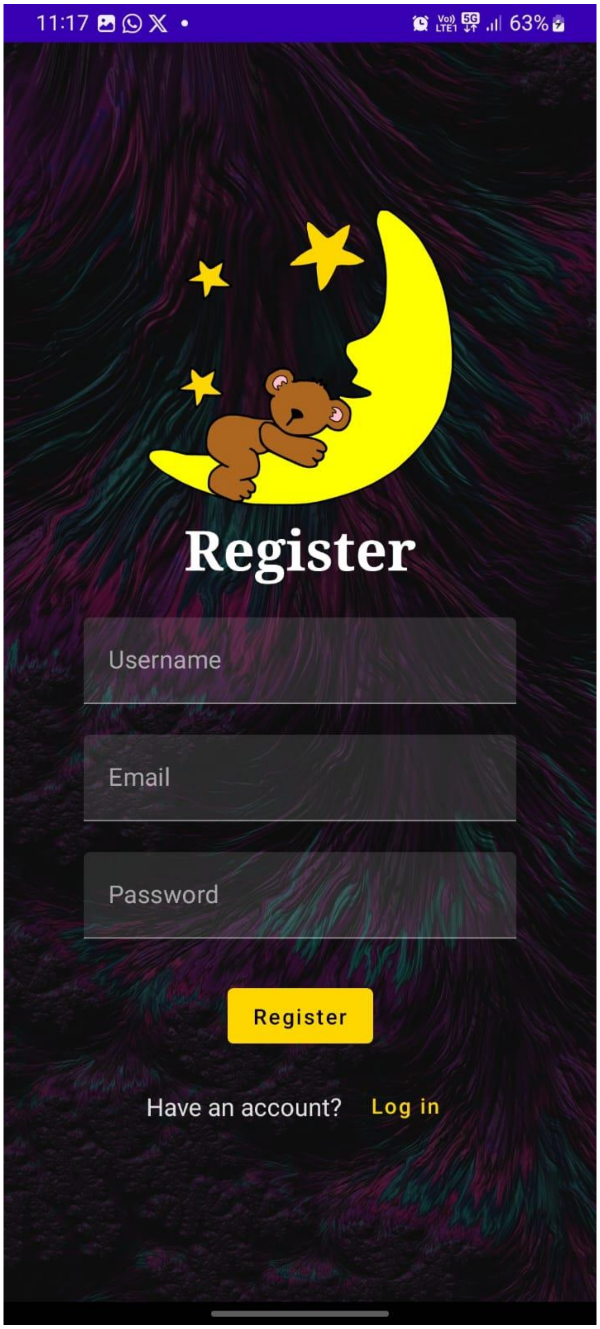
5.	Effective User Story	3 Scenes Added  Sleep Time Tracking Start Track Sleep Play Restart Set Alarm Elapsed Time: 00:00:00 Alarm will ring at 23:17:35
6.	Descriptive Reports	 Sleep Tracking Start time: 1970-01-01 05:30:00 End time: 2023-11-09 21:11:59

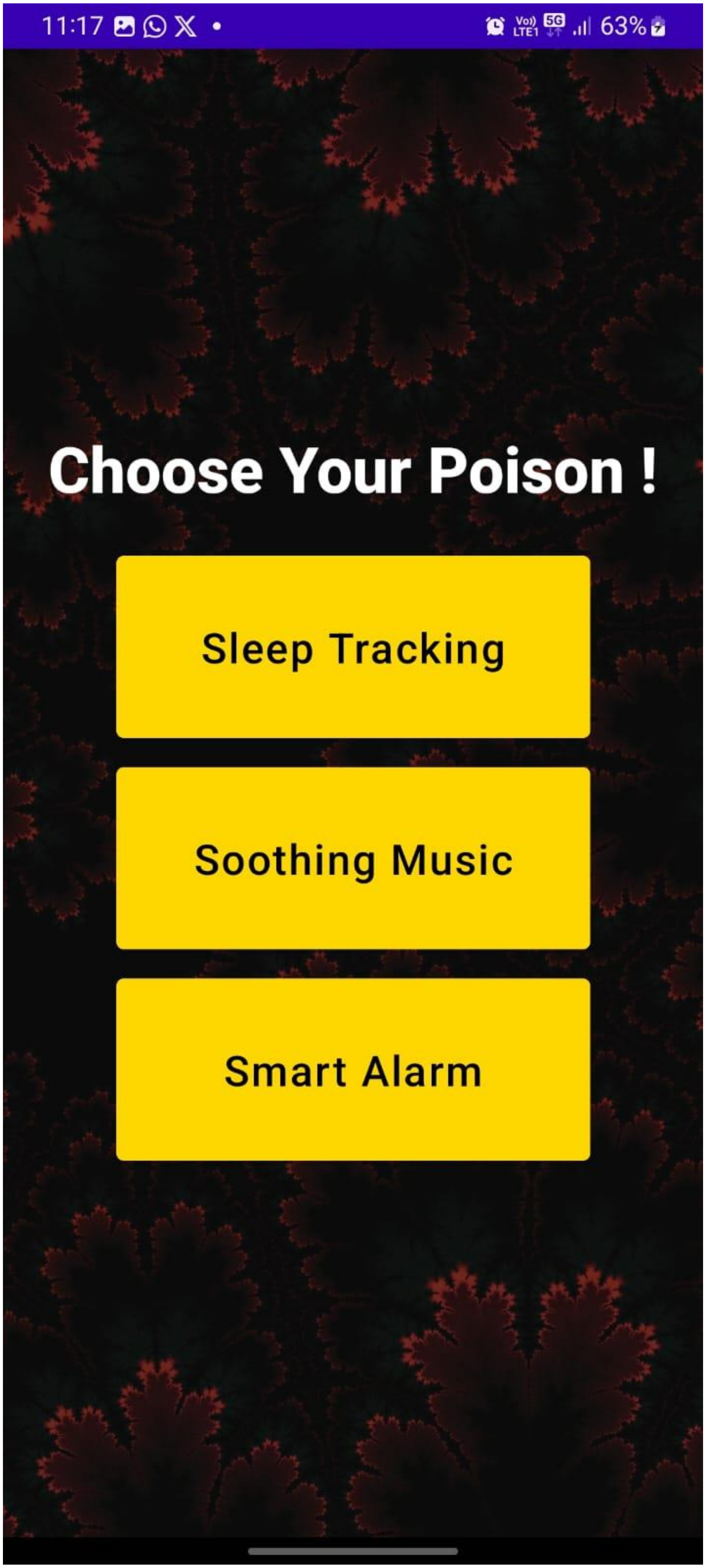
9. RESULTS

9.1. Output Screenshots

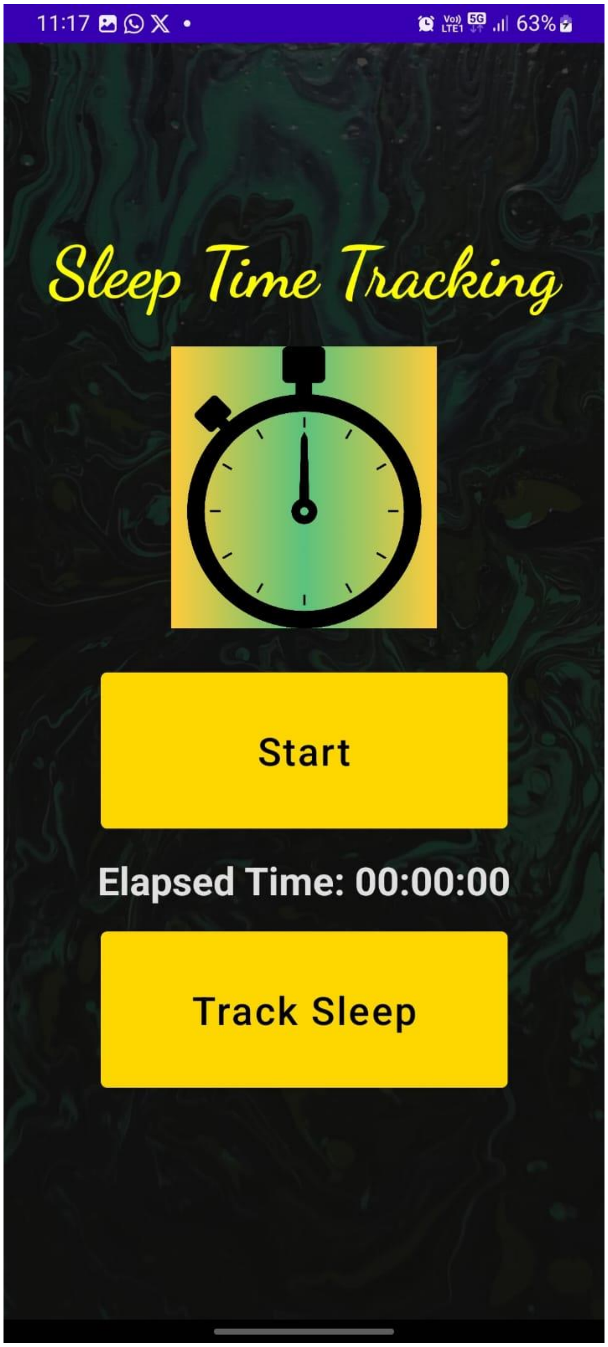


Successfully log in

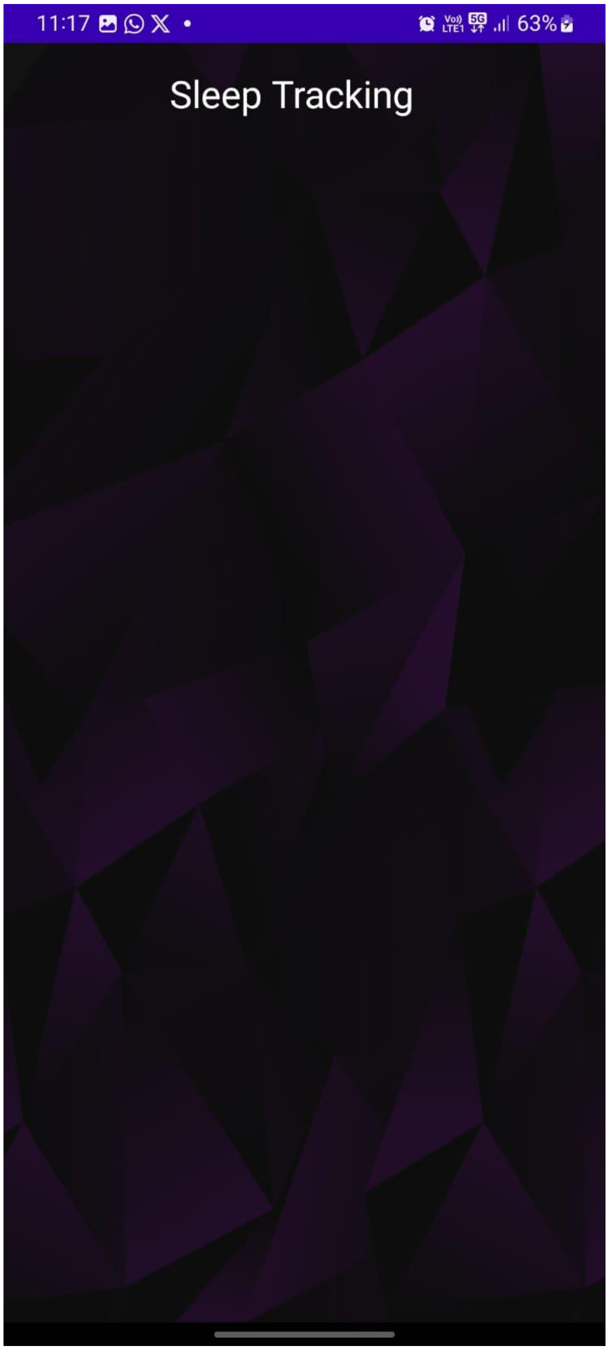


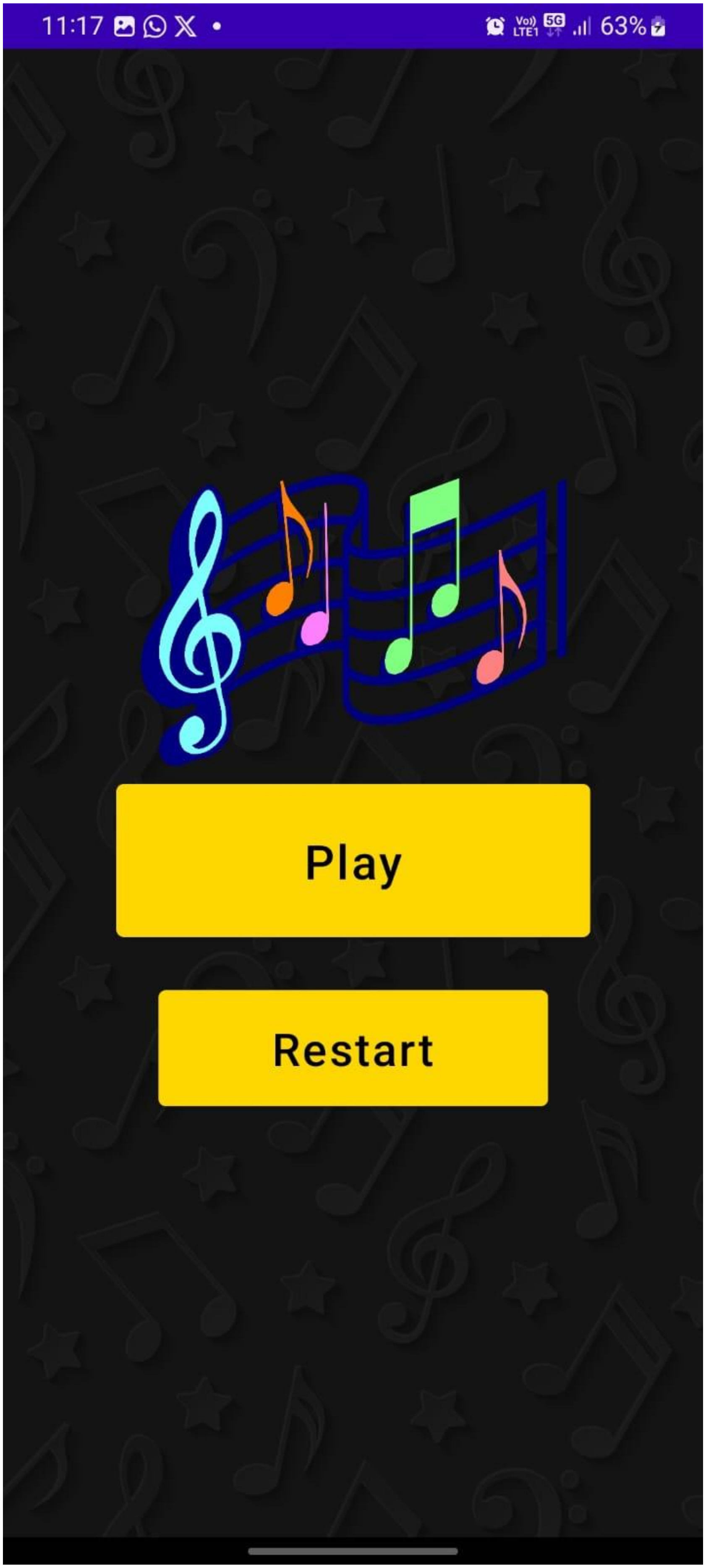


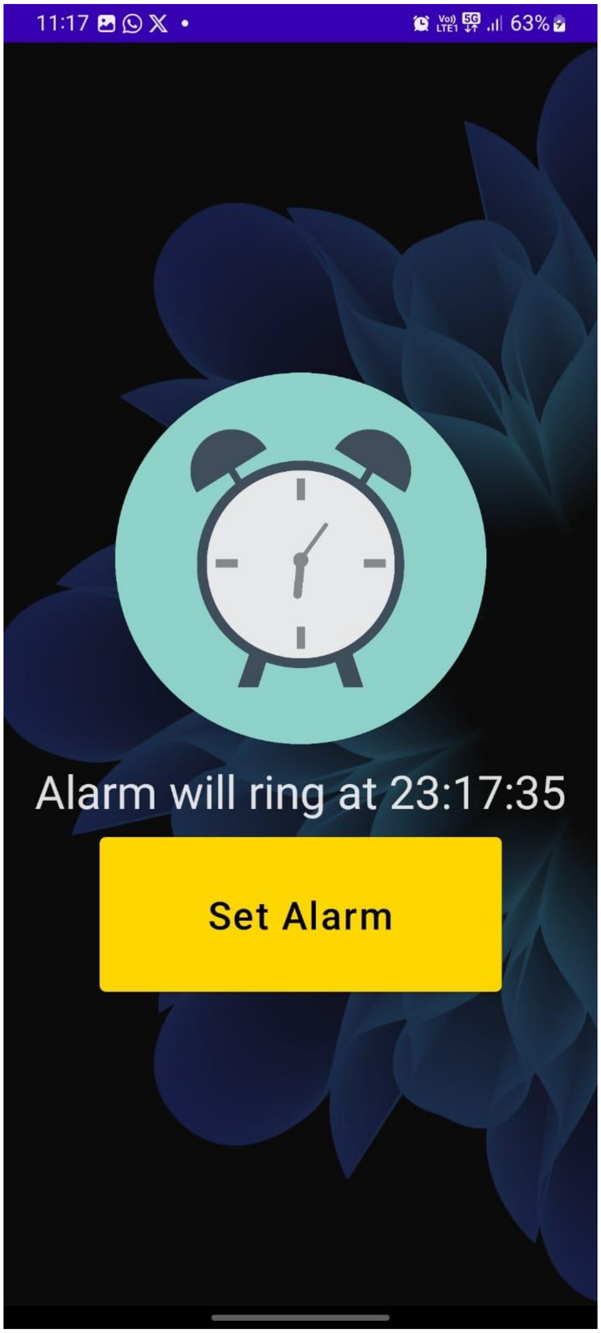
Choose Your Poison !



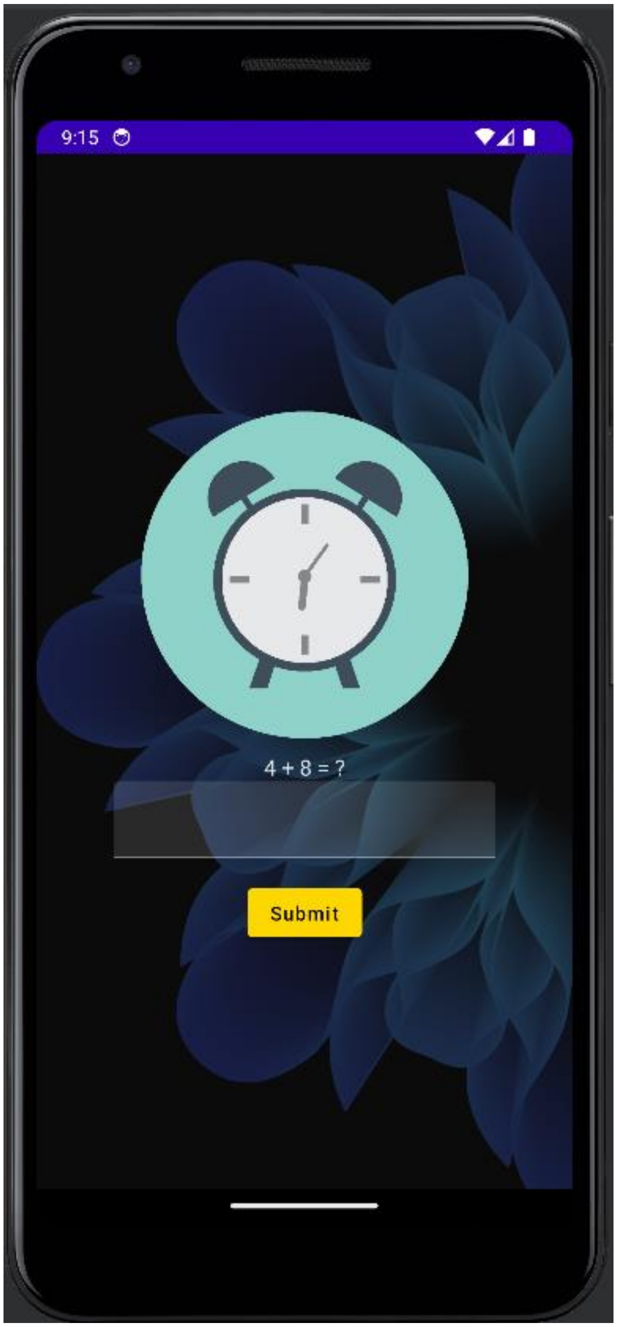
Elapsed Time: 00:00:00




11:17
63%
Play
Restart



Alarm will ring at 23:17:35



4 + 8 = ?

10. ADVANTAGES & DISADVANTAGES

Advantages of our app include:

- Improved Sleep Quality: Our app helps users monitor and analyze their sleep patterns, allowing them to make informed decisions to improve the quality of their sleep.
- Smart Alarm Feature: The smart alarm feature ensures that users wake up at the right point in their sleep cycle, leading to more refreshing mornings.
- User-Friendly Interface: The app boasts an intuitive and easy-to-navigate interface, making it accessible to a wide range of users.
- Customizable Alarms: Users can set alarms tailored to their preferred wake-up times.
- Data-Driven Insights: Sleep data collected by the app provides valuable insights into sleep patterns and areas for improvement.

However, there are some disadvantages to consider:

- Device Dependence: The app relies on a smartphone or tablet, which may not be ideal for users who prefer standalone sleep tracking devices.
- Battery Consumption: Continuous use of the app overnight may lead to increased battery consumption.
- Privacy Concerns: Sleep tracking involves the collection of personal data, raising privacy concerns that need to be addressed.
- Sensor Accuracy: The app's accuracy may be influenced by the device's sensors, which can vary in quality.
- Potential for Sleep Anxiety: Constant sleep tracking and analysis might lead to sleep anxiety for some users who become overly concerned about their sleep data.

It's essential to continually update and refine the app to address these concerns and provide the best possible user experience.

11. CONCLUSION

In conclusion, our app offers a valuable solution for individuals seeking to enhance their sleep quality and overall well-being. With its sleep tracking capabilities and smart alarm feature, it empowers users to make informed decisions about their sleep routines. While there are certain challenges to overcome, such as device dependence and privacy concerns, our commitment to improving and refining the app will address these issues over time. By prioritizing user-friendly design and data-driven insights, our app has the potential to positively impact the lives of many, promoting healthier sleep habits and better mornings. We remain dedicated to further enhancing the app's features and addressing user feedback, ensuring its continued growth and effectiveness.

12. FUTURE SCOPE

The future scope of our app is promising and opens up several exciting possibilities. First and foremost, we aim to expand the app's compatibility with a wider range of devices and operating systems, making it accessible to a broader audience. Additionally, we plan to integrate more advanced sleep tracking technologies, such as wearables and sensors, to provide even more accurate data and insights.

One important aspect of the app's future development is personalization. We intend to implement machine learning and AI algorithms to analyze users' sleep patterns and provide tailored recommendations for improving their sleep quality. This could include customized sleep schedules, relaxation techniques, and environmental adjustments.

Furthermore, we see potential in collaborating with healthcare professionals and sleep experts to offer in-app consultations and advice for individuals with specific sleep disorders or concerns. This can enhance the app's credibility as a sleep management tool.

Overall, the future of our app involves continuous improvement, user-centric features, and a commitment to helping people achieve better sleep and, by extension, a better quality of life.

13. APPENDIX

13.1.1. Source Code

The GitHub Link for my project-

https://github.com/smartinternz02/SI-GuidedProject-587558-1696963149

13.1.2. GitHub & Project Demo Link-

https://github.com/444aki/ProjectOne