Android Application Development

A Sleep Tracking App for a Better Night's Rest

TEAM ID –NM2024TMID05856

Submitted By

Ajay kumar V(Team Leader)	-	**D37195C88825DEC9B034FF3665C5DAE0**
Suryaprakash J(Team Member)	-	**86B3AB99837206AAA7B4978404551360**
Gabilarraja R(Team Member)	-	**9BA1D7F777C48CE98FB44979DD0EA112**
Ramakrishnan G B(Team Member)	-	**CC865F9BE9D8866ACF542286A8444854**
Bharathi S(Team Member)	-	**6B13BEC0C7348AF80A1460B972C0E6CA**

SEMESTER – V

B.E COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR – 2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE

COIMBATORE-641046

NOVEMBER 2024

LIST OF CONTENT

CHAPTER – I
INTRODUCTION

1.1 OBJECTIVE

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 OVERVIEW

A sleep tracking app is a mobile application designed to monitor and analyse your sleep patterns and provide insights on how to improve your sleep quality and overall health by providing personalized recommendations based on their sleep data. A sleep tracking app can be a useful tool for anyone looking to optimize their sleep and improve their overall health and well-being. However, it's important to note that while these apps can provide helpful information.

"Sleep Tracker" application enables you to start the timer when they are in the bed and about to fall asleep. The timer will keep running in the background until it is stopped, whenever the user wakes up. Based on the sleep experience, you can rate your sleep quality. Finally, the app will display an analysis of the kind of sleep, you had the previous night. In an effort to help our users stay informed about their sleep, we are making Sleep API.

Our phones have become great tools for making more informed decisions about our sleep. And by being informed about sleep habits, people can make better decisions throughout the day about sleep, which affects things like concentration and mental health. The Sleep Application Programming Interface is an Android Activity be used to power features like the Bedtime mode in Clock. Elapsed Time return the time since the system was booted, and include deep sleep. This clock is guaranteed to be monotonic, and continues to tick even when the CPU is in power saving modes, so is the recommend basis for general purpose interval timing.

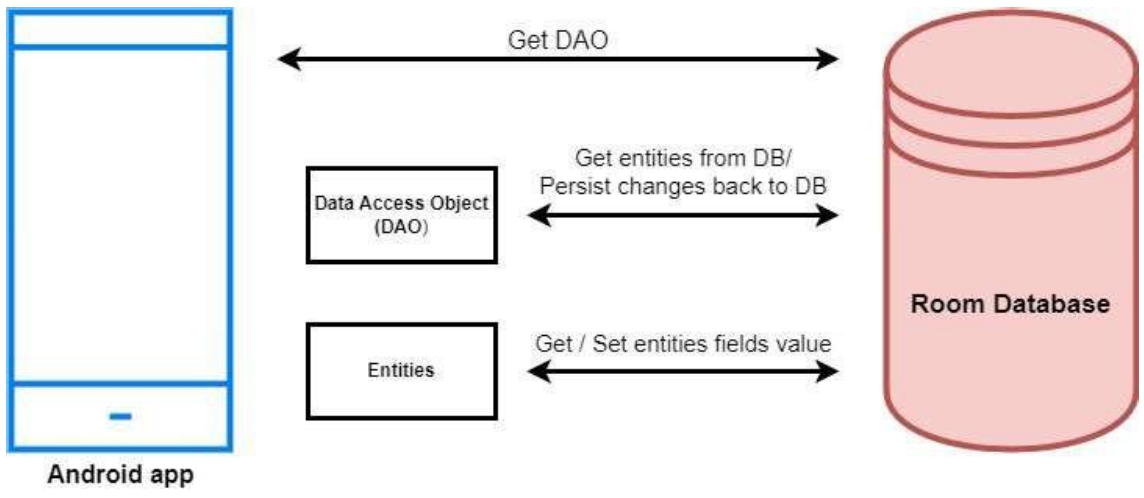
This sleeping information is reported in two ways .

- A daily sleep segment which is reported after a wakeup is detected.
- App can run in the background to optimize battery usage based on the user's habits and preferences.

CHAPTER – II

FUNCTIONALITIES

2.1 ARCHITECTURE


Get DAO
Data Access Object (DAO)
Get entities from DB/ Persist changes back to DB
Entities
Get / Set entities fields value
Room Database
Android app

2.2 PROJECT WORKFLOW

- **User registration:** Users provide their personal information and create an account to access the sleep tracking features of the application.
- **User login:** Once registered, users can log in to the application using their email and password to access the sleep tracking functionality.
- **Main page:** After logging in, the user is directed to the main page of the application, where they can access the sleep tracking features.
- **Sleep tracking:** The user can record the time they go to sleep and the time they wake up.

CHAPTER - III

3.1 PROBLEM STATEMENT

With growing awareness of the importance of quality sleep for both mental and physical health, there is a need for accessible tools that help individuals monitor and improve their sleep habits. Many existing solutions are either too complex or lack a user-friendly interface that promotes consistent usage. As a result, individuals may not have access to easy-to-understand insights on their sleep patterns, making it challenging to address sleep-related issues effectively. This project aims to address this gap by developing a straightforward and engaging sleep tracking app that provides actionable insights into sleep quality and duration.

3.2 SOLUTION

The **Sleep Tracker** app provides a minimalist, easy-to-use platform for users to monitor and improve their sleep habits. Using **Jetpack Compose** for an intuitive UI, the app enables users to start a sleep timer, record their sleep duration, rate their sleep quality, and view daily and historical analyses of their sleep patterns. The app's data visualization offers insights into trends, helping users recognize patterns that may impact their sleep. Future enhancements, such as smart alarms and health metric integration, will make the app a comprehensive solution for those looking to enhance their sleep quality through accessible technology.

Good quality sleep is essential for physical and mental health, and lack of sleep can lead to various health problems, such as obesity, diabetes, cardiovascular disease, and depression. By using a sleep tracking app, you can identify areas where you may be falling short in terms of sleep hygiene, and make adjustments accordingly. In addition to improving your sleep quality, a sleep tracking app can also help you identify underlying sleep disorders or health issues that may be affecting your sleep.

CHAPTER - IV

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

1. **Development Device (Computer/Laptop)**
   - **Processor**: Intel i5 (8th generation or higher) or AMD Ryzen 5 equivalent
   - **RAM**: 8 GB minimum (16 GB recommended for smoother performance)
   - **Storage**: 100 GB of available storage space
   - **Graphics**: Integrated graphics (discrete GPU optional for emulator acceleration)
2. **Mobile Device for Testing (Optional)**
   - **Operating System**: Android 8.0 (Oreo) or higher
   - **RAM**: 2 GB minimum
   - **Storage**: At least 50 MB of free storage space for the app

4.2 SOFTWARE REQUIREMENTS

1. **Operating System** (for development):
   - Windows 10 or 11 (64-bit), macOS (Big Sur or later), or Linux (Ubuntu 20.04 or later)
2. **Development Tools**
   - **Android Studio**: Version 4.2 or higher
   - **JDK**: Java Development Kit 11 or higher
   - **Android SDK**: Android SDK API Level 26 or higher
3. **Libraries and Frameworks**
   - **Android Jetpack Compose**: For building UI components
   - **Room Database**: For local data storage (optional but recommended for sleep data persistence)
   - **Kotlin Coroutines**: For managing asynchronous tasks smoothly
   - **Dagger Hilt**: For dependency injection (optional, for larger app projects)
4. **Database**
   - **SQLite or Room Database**: For offline data storage and sleep record persistence
   - **SharedPreferences**: For lightweight data (e.g., user settings, last session data)

5. **Additional Tools (Optional)**
    - **Firebase**: For remote data storage, analytics, and crash reporting (optional)
    - **Git**: For version control and collaboration, with a GitHub or GitLab repository for project storage
    - **Postman**: For testing any APIs (if integrated with other health services)

4.3 TESTING REQUIREMENTS

1. **Android Emulator**: Configured in Android Studio with API Level 26 or higher
2. **Physical Android Device** (optional): For real-world testing, ideally with Android 8.0 (Oreo) or higher

4.4 NETWORK REQUIREMENTS

- **Internet Connection**: Required only for features such as updates, analytics, or remote data storage if using Firebase or other cloud services.

Here’s an overview of these network needs:

1. Low-Latency Data Transfer

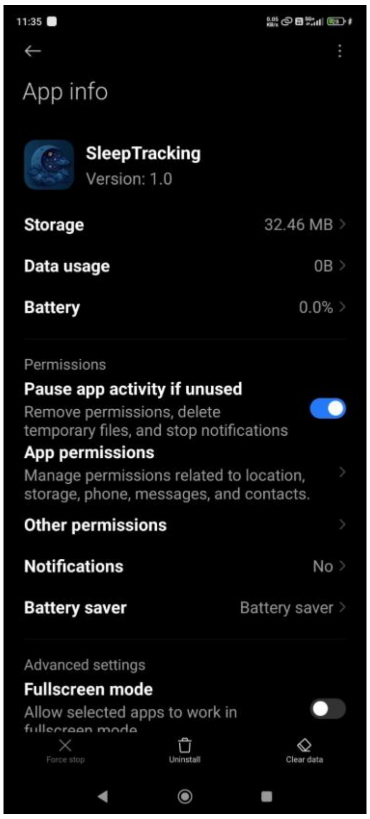
- **Requirement**: Since sleep tracking often involves real-time or near-real-time data collection from wearable devices and sensors, the network should support low-latency connections to process and relay data quickly and accurately.
- **Solution**: Implement edge computing and caching techniques closer to users to reduce latency.

2. Scalable Cloud Infrastructure

- **Requirement**: As the user base grows, the system will need a scalable cloud infrastructure to handle the large influx of data without performance degradation.
- **Solution**: Use cloud platforms like AWS, Azure, or Google Cloud, which offer scalable storage and computing resources and support for load balancing to manage traffic spikes.

CHAPTER - V

APPLICATION SCREENSHOTS



11:35

App info

SleepTracking
Version: 1.0

Storage 32.46 MB

Data usage 0B

Battery 0.0%

Permissions

Pause app activity if unused
Remove permissions, delete temporary files, and stop notifications

App permissions
Manage permissions related to location, storage, phone, messages, and contacts.

Other permissions

Notifications No

Battery saver Battery saver

Advanced settings

Fullscreen mode
Allow selected apps to work in fullscreen mode

FIG 1 – APP INFORMATION

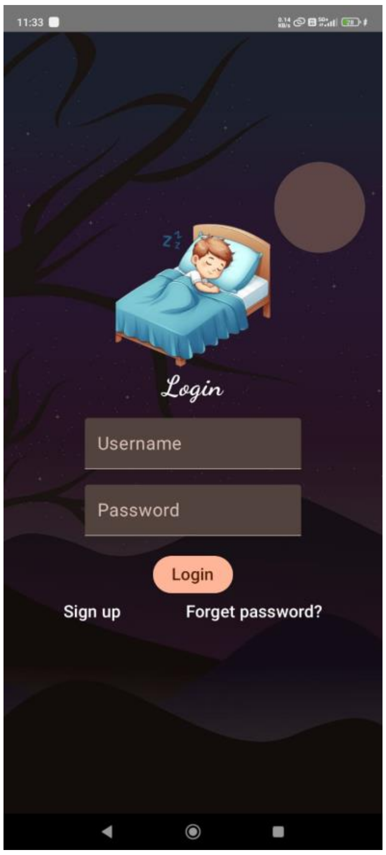


FIG 2 – APP LOGIN

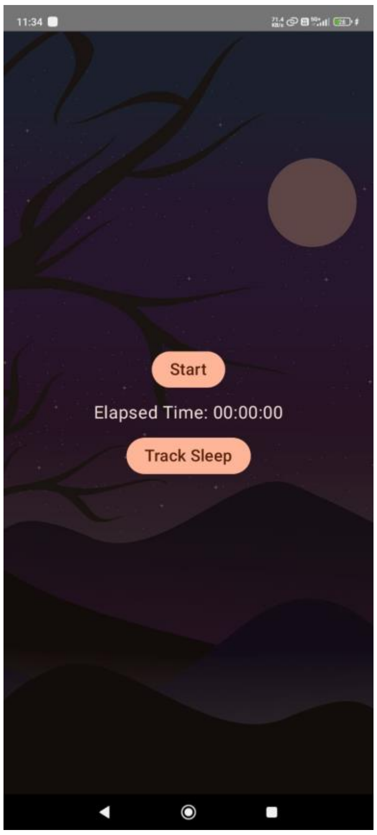


FIG 3 – SLEEP TRACKER

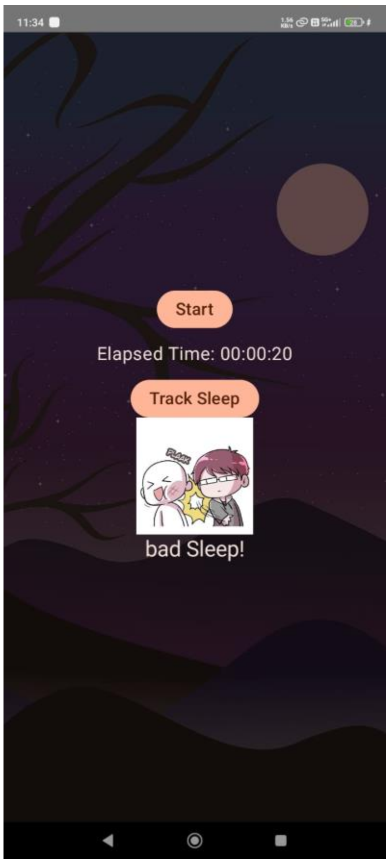
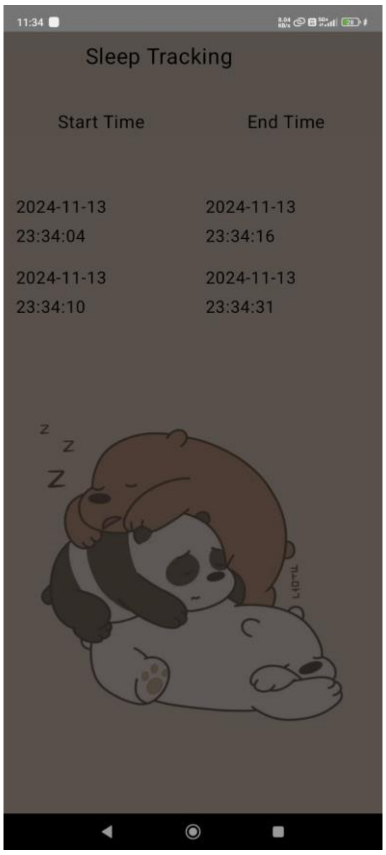


FIG 4– SLEEP TRACKER 2


Sleep Tracking

Start Time	End Time
2024-11-13 23:34:04	2024-11-13 23:34:16
2024-11-13 23:34:10	2024-11-13 23:34:31

FIG 5 – TRACKED RESULT

CHAPTER - VI

APPLICATIONS

- **Improving sleep habits:** By monitoring your sleep patterns, sleep tracking apps can help you identify habits that may be affecting your sleep quality, such as late-night screen time or caffeine intake. This can help you make changes to your routine and improve your sleep quality over time.

- **Managing sleep disorders:** Sleep tracking apps can help people with sleep disorders such as insomnia monitor their sleep patterns and identify potential triggers for their symptoms. This information can then be shared with a healthcare provider to help develop a treatment plan.

- **Sports performance:** Sleep tracking apps can also be used by athletes to monitor their sleep quality and quantity. Adequate sleep is essential for physical recovery, and monitoring sleep patterns can help athletes optimize their performance.

- **Research:** Sleep tracking apps can provide valuable data for sleep researchers studying sleep patterns and habits on a larger scale. By aggregating data from many users, researchers can gain insights into sleep patterns and trends across different populations.

CHAPTER – VII

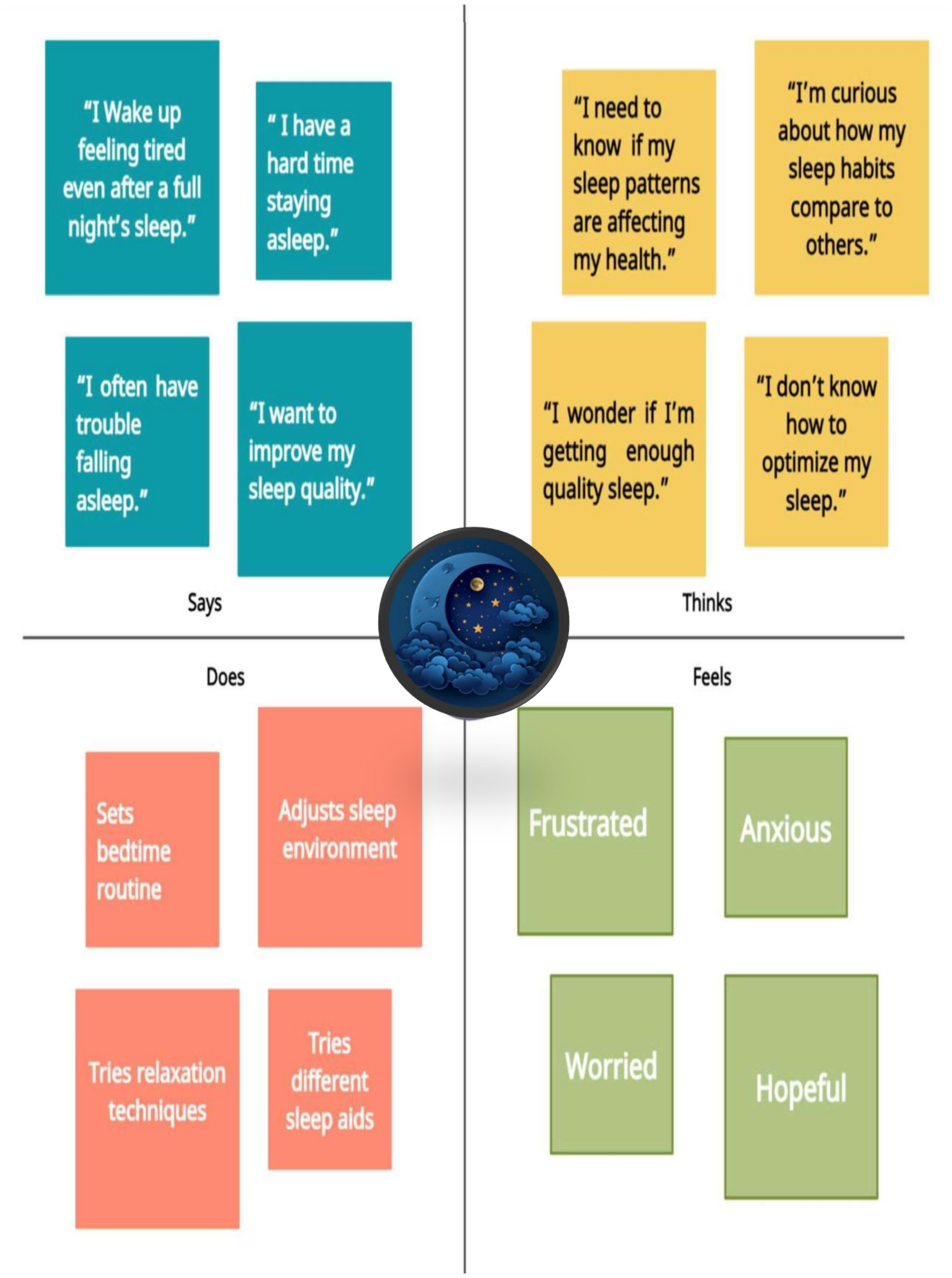
7.1 FUTURE SCOPE

The future of sleep tracking apps looks promising, with continued advancements in technology. Advance sensors can improve the accuracy of sleep tracking apps, capturing more detailed information about sleep patterns. Improved algorithms can better differentiate between light and deep sleep. Sleep tracking apps may provide personalized recommendations based on individual sleep patterns, such as adjusting bedtime, avoiding beverages, or incorporating specific relaxation techniques. Enhanced privacy and security features to protect user data. Gamification elements may motivate users to achieve sleep goals, making it a more fun and engaging experience.

Here are some additional possibilities for the future scope of sleep tracking apps:

1. **Integration with Wearable and Smart Home Devices**: Sleep tracking apps could integrate seamlessly with wearable devices (like smartwatches and rings) as well as smart home systems. Imagine your app dimming lights, adjusting room temperature, or playing specific sleep-inducing sounds based on your real-time sleep stage, all aimed at improving sleep quality.
2. **AI-Driven Insights and Predictions**: By utilizing AI and machine learning, sleep tracking apps can analyze large datasets to provide predictive insights, such as forecasting potential insomnia episodes or identifying early signs of sleep disorders. This proactive approach could help users make changes before issues arise.

7.2 EMPATHY MAP


"I Wake up feeling tired even after a full night's sleep."
" I have a hard time staying asleep."
"I often have trouble falling asleep."
"I want to improve my sleep quality."
Says
"I need to know if my sleep patterns are affecting my health."
"I'm curious about how my sleep habits compare to others."
"I wonder if I'm getting enough quality sleep."
"I don't know how to optimize my sleep."
Thinks
Does
Sets bedtime routine
Adjusts sleep environment
Tries relaxation techniques
Tries different sleep aids
Feels
Frustrated
Anxious
Worried
Hopeful

CHAPTER – VIII

CONCLUSION

In conclusion, the future of sleep tracking apps looks exceptionally promising, with advancements that go beyond merely tracking sleep patterns. As technology evolves, these apps will become more personalized, accurate, and holistic, addressing a wide range of factors that affect sleep quality and overall health. By integrating with wearables, smart home devices, and even mental health platforms, sleep tracking apps could offer users a comprehensive tool for improving their sleep and well-being. The addition of features like AI-driven insights, gamification, and enhanced privacy protections will likely increase engagement, making sleep tracking not only more effective but also more enjoyable and secure. Ultimately, these developments position sleep tracking apps as vital components of personal health management, empowering users to make informed, data-driven choices for a better, healthier life.

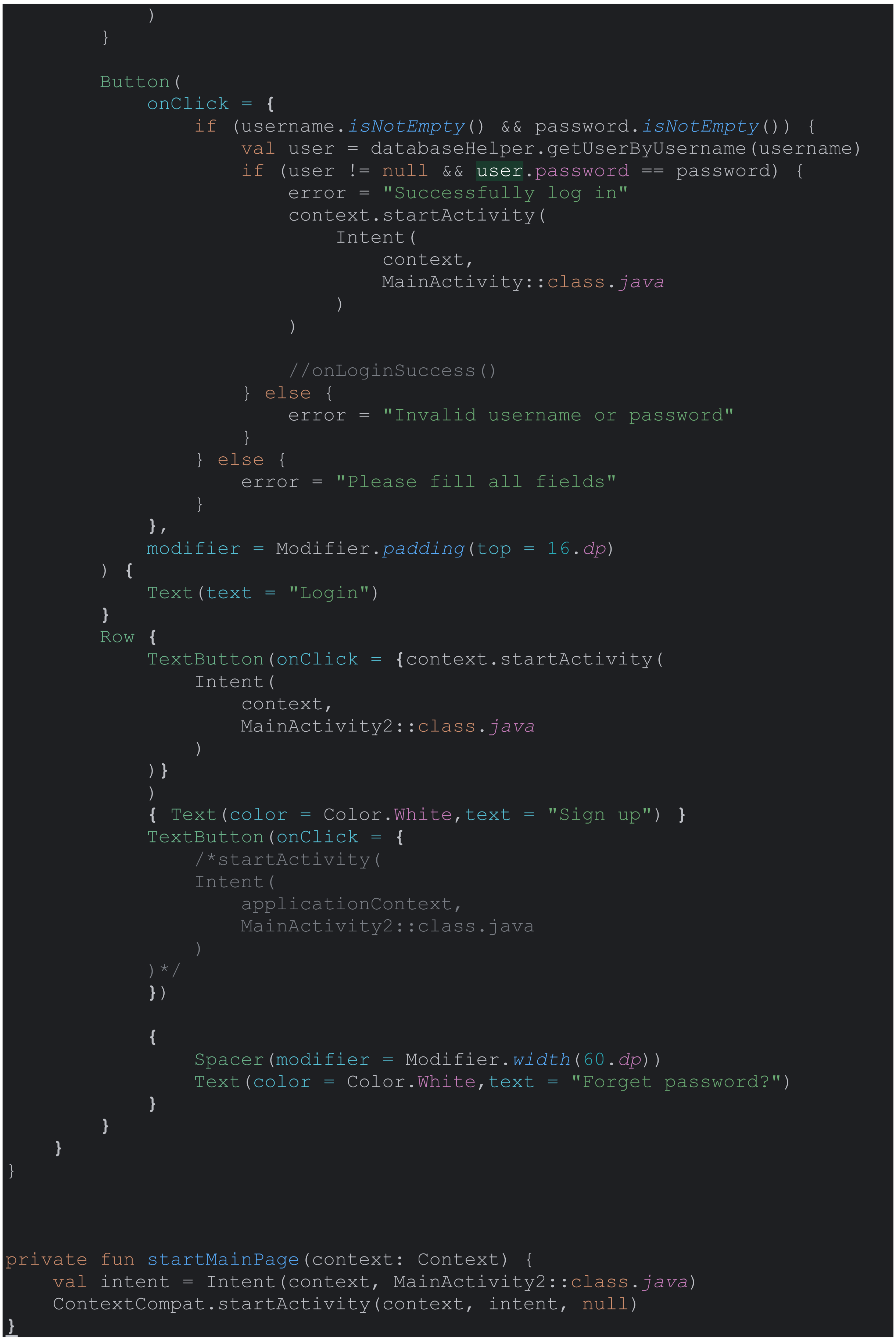
CHAPTER IX

APPENDIX

SOURCE CODE

LOGINACTIVITY.KT

```
package com.example.sleeptracking

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.material3.TextField
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.sleeptracking.ui.theme.SleepTrackingTheme



class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackingTheme {
                // A surface container using the 'background' color from
the theme


                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.summa),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(
            painter = painterResource(id = R.drawable.sum3),
            contentDescription = "",

            modifier = imageModifier
                .width(260.dp)
                .height(200.dp)
        )
        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.White,
            text = "Login"
        )
        Spacer(modifier = Modifier.height(10.dp))

        TextField(
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )

        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )

        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colorScheme.error,
                modifier = Modifier.padding(vertical = 16.dp)
```



```
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user = databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainActivity::class.java
                            )
                        )

                        //onLoginSuccess()
                    } else {
                        error = "Invalid username or password"
                    }
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    MainActivity2::class.java
                )
            )}
            )
            { Text(color = Color.White,text = "Sign up") }
            TextButton(onClick = {
                /*startActivity(
                Intent(
                    applicationContext,
                    MainActivity2::class.java
                )
            )*/
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color.White,text = "Forget password?")
            }
        }
    }
}



private fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity2::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

MAINACTIVITY.KT

```
package com.example.sleeptracking

import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*

import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import com.example.sleeptracking.ui.theme.SleepTrackingTheme
import java.util.*

class MainActivity : ComponentActivity() {

    private lateinit var databaseHelper: TimeLogDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = TimeLogDatabaseHelper(this)
        databaseHelper.deleteAllData()
        setContent {
            SleepTrackingTheme{
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MyScreen(this,databaseHelper)
                }
            }
        }
    }
}
@Composable
fun MyScreen(context: Context, databaseHelper: TimeLogDatabaseHelper) {
    var startTime by remember { mutableStateOf(0L) }
    var elapsedTime by remember { mutableStateOf(0L) }
    var isRunning by remember { mutableStateOf(false) }
    var showGoodSleepMessage1 by remember { mutableStateOf(false) }
    var showGoodSleepMessage2 by remember { mutableStateOf(false) }
    var showGoodSleepMessage3 by remember { mutableStateOf(false) }


    // LaunchedEffect to keep updating the elapsed time while the timer is
```

```
running
    LaunchedEffect(isRunning) {
        if (isRunning) {
            startTime = System.currentTimeMillis() - elapsedTime //
Preserve elapsed time when resuming
            while (isRunning) {
                elapsedTime = System.currentTimeMillis() - startTime
                kotlinx.coroutines.delay(1000) // Update every second
            }
        }
    }
    LaunchedEffect(showGoodSleepMessage1) {
        if (showGoodSleepMessage1) {
            kotlinx.coroutines.delay(2000) // Show for 2 seconds
            showGoodSleepMessage1 = false // Hide after delay
        }
    }
    LaunchedEffect(showGoodSleepMessage2) {
        if (showGoodSleepMessage2) {
            kotlinx.coroutines.delay(2000) // Show for 2 seconds
            showGoodSleepMessage2 = false // Hide after delay
        }
    }
    LaunchedEffect(showGoodSleepMessage3) {
        if (showGoodSleepMessage3) {
            kotlinx.coroutines.delay(2000) // Show for 2 seconds
            showGoodSleepMessage3 = false // Hide after delay
        }
    }
    Image(
        painterResource(id = R.drawable.summa),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = Modifier.alpha(0.3F),
    )

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        if (!isRunning) {
            Button(onClick = {
                startTime = System.currentTimeMillis() + elapsedTime //
Reset or resume from last elapsed time
                isRunning = true
            }) {
                Text("Start")
            }
        }  else {
            Button(onClick = {
                val endTime = System.currentTimeMillis()
                isRunning = false
                databaseHelper.addTimeLog(startTime, endTime) // Save the
time log

                // Check if the sleep duration is between 7 and 8 hours
                val duration = endTime - startTime


                if (duration >= (7 * 60 * 60 * 1000) && duration <= (8 * 60
* 60 * 1000)) {
                    // Show "Good Sleep" message and image for 7-8 hours
                    showGoodSleepMessage2 = true
                } else if (duration < (7 * 60 * 60 * 1000)) {
                    // Show "Less than 7 hours" message and image
                    showGoodSleepMessage1 = true
                } else {
                    // Code for other cases (e.g., more than 8 hours)
                    showGoodSleepMessage3 = true
                }

            }) {
                Text("Stop")
            }
        }
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Elapsed Time: ${formatTime(elapsedTime)}")

        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = {
            context.startActivity(
                Intent(context, TrackActivity::class.java)
            )
        }) {
            Text(text = "Track Sleep")
        }

        if (showGoodSleepMessage1) {
            Image(
                painter = painterResource(id = R.drawable.lesssleep),
                contentDescription = "bad Sleep",
                modifier = Modifier
                    .height(150.dp)
                    .width(150.dp)
            )
            Text(text = "bad Sleep!", style =
MaterialTheme.typography.headlineSmall)
        }
        if (showGoodSleepMessage2) {
            Image(
                painter = painterResource(id = R.drawable.goodsleep),
                contentDescription = "Good Sleep",
                modifier = Modifier
                    .height(150.dp)
                    .width(150.dp)
            )
            Text(text = "Good Sleep!", style =
MaterialTheme.typography.headlineSmall)
        }
        if (showGoodSleepMessage3) {
            Image(
                painter = painterResource(id = R.drawable.oversleep),
                contentDescription = "Over Sleep",
                modifier = Modifier
                    .height(150.dp)
                    .width(150.dp)
            )
            Text(text = "Over Sleep!", style =
MaterialTheme.typography.headlineSmall)
        }
```

```
    }

}

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 /n 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)
}
```

REGISTRATIONACTIVITY.KT

```
package com.example.sleeptracking

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.material3.TextField
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.sleeptracking.ui.theme.SleepTrackingTheme
```

```
class MainActivity2 : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackingTheme{
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {

                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
}


@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.summa),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(
            painter = painterResource(id = R.drawable.sum3),
            contentDescription = "",

            modifier = imageModifier
                .width(260.dp)
                .height(200.dp)
        )
        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color.White,
            text = "Register"
        )

        Spacer(modifier = Modifier.height(10.dp))
        TextField(
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)

        )

        TextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )

        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )


        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colorScheme.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    if (email.contains('@')) {
                        val user = User(
                            id = null,
                            firstName = username,
                            lastName = null,
                            email = email,
                            password = password
                        )
                        databaseHelper.insertUser(user)
                        error = "User registered successfully"
                        // Start LoginActivity using the current context
                        context.startActivity(
                            Intent(
                                context,
                                LoginActivity::class.java
                            )
                        )
                    } else {
                        error = "Please enter a valid email with '@'
symbol"
```

```
                    }
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register")
        }

        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))

        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {

context.startActivity(Intent(context,LoginActivity::class.java
                ))

            })

            {
                Spacer(modifier = Modifier.width(10.dp))
                Text(text = "Log in")
            }
        }
    }
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

TIMEDATABASEHELPER.KT

```
package com.example.sleeptracking

//class TimeDatabaseHelper {
// package com.example.

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import java.util.*

class TimeLogDatabaseHelper(context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_NAME = "timelog.db"
        private const val DATABASE_VERSION = 1
        const val TABLE_NAME = "time_logs"
        private const val COLUMN_ID = "id"
        const val COLUMN_START_TIME = "start_time"
        const val COLUMN_END_TIME = "end_time"

        // Database creation SQL statement
        private const val DATABASE_CREATE =
            "create table $TABLE_NAME ($COLUMN_ID integer primary key
autoincrement, " +
                    "$COLUMN_START_TIME integer not null, $COLUMN_END_TIME
integer);"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        db?.execSQL(DATABASE_CREATE)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    // function to add a new time log to the database
    fun addTimeLog(startTime: Long, endTime: Long) {
        val values = ContentValues()
        values.put(COLUMN_START_TIME, startTime)
        values.put(COLUMN_END_TIME, endTime)
        writableDatabase.insert(TABLE_NAME, null, values)
    }

    // function to get all time logs from the database
    @SuppressLint("Range")
    fun getTimeLogs(): List<TimeLog> {
        val timeLogs = mutableListOf<TimeLog>()
        val cursor = readableDatabase.rawQuery("select * from $TABLE_NAME",
null)
        cursor.moveToFirst()
        while (!cursor.isAfterLast) {
            val id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_START_TIME))
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_END_TIME))
            timeLogs.add(TimeLog(id, startTime, endTime))
            cursor.moveToNext()
        }
        cursor.close()
        return timeLogs
    }

    fun deleteAllData() {
        writableDatabase.execSQL("DELETE FROM $TABLE_NAME")
    }

    fun getAllData(): Cursor? {
        val db = this.writableDatabase
        return db.rawQuery("select * from $TABLE_NAME", null)
    }
    data class TimeLog(val id: Int, val startTime: Long, val endTime:
Long?) {
        fun getFormattedStartTime(): String {
            return Date(startTime).toString()
        }

        fun getFormattedEndTime(): String {
            return endTime?.let { Date(it).toString() } ?: "not ended"
        }
    }
}

//}
```

TRACKACTIVITY.KT

```
package com.example.sleeptracking

import android.icu.text.SimpleDateFormat
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.sleeptracking.ui.theme.SleepTrackingTheme
import java.util.*

class TrackActivity : ComponentActivity() {

    private lateinit var databaseHelper: TimeLogDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        databaseHelper = TimeLogDatabaseHelper(this)
        setContent {
            SleepTrackingTheme{
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    //ListListScopeSample(timeLogs)

                    val data=databaseHelper.getTimeLogs();
                    Log.d("Sandeep" ,data.toString())
                    val timeLogs = databaseHelper.getTimeLogs()
                    ListListScopeSample(timeLogs)
                }
            }
        }
    }
}


@Composable
fun ListListScopeSample(timeLogs: List<TimeLogDatabaseHelper.TimeLog>) {
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.last),
        contentScale = ContentScale.FillBounds,
        contentDescription = "",
        modifier = imageModifier.alpha(0.3F)
    )

    Text(
        text = "Sleep Tracking",
        modifier = Modifier
            .padding(top = 16.dp, start = 106.dp),
        color = Color.Black,
        fontSize = 24.sp
    )

    // Spacer to separate the "Sleep Tracking" text and the table header
    Spacer(modifier = Modifier.height(40.dp))

    // Table Header
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .padding(bottom = 100.dp),
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        Text("Start Time", modifier = Modifier
            .weight(1f)
            .padding(top = 100.dp, start = 70.dp), color = Color.Black,
fontSize = 16.sp)
        Text("End Time", modifier = Modifier
            .weight(1f)
            .padding(top = 100.dp, start = 70.dp), color = Color.Black,
fontSize = 16.sp)
    }

    // Spacer between the header and rows for additional clarity
    Row {
        Spacer(modifier = Modifier.height(90.dp))

        // Table Rows
        LazyColumn(modifier = Modifier
            .fillMaxSize()
            .padding(bottom = 10.dp , top = 200.dp) ) {
```

```
            items(timeLogs) { timeLog ->
                Row(
                    modifier = Modifier
                        .fillMaxWidth()
                        .padding(vertical = 8.dp),
                    horizontalArrangement = Arrangement.SpaceBetween
                ) {
                    Text(
                        text = formatDateTime(timeLog.startTime),
                        modifier = Modifier.weight(1f) .padding(horizontal
= 16.dp),
                        color = Color.Black,
                        fontSize = 14.sp
                    )
                    Text(
                        text = timeLog.endTime?.let { formatDateTime(it) }
?: "N/A",
                        modifier = Modifier.weight(1f) .padding(horizontal
= 16.dp),
                        color = Color.Black,
                        fontSize = 14.sp
                    )
                }
            }
        }
    }
}

private fun formatDateTime(timestamp: Long): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
    return dateFormat.format(Date(timestamp))
}
```

ANDROIDMANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/summa2"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.SleepTracking"
        tools:targetApi="31">
        <activity
            android:name=".TrackActivity"
            android:exported="false"
            android:label="@string/title_activity_track"
            android:theme="@style/Theme.SleepTracking" />
        <activity
```

```
            android:name=".MainActivity"
            android:exported="false"
            android:label="SleepTracking"
            android:theme="@style/Theme.SleepTracking" />
        <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:label="RegisterActivity"
            android:theme="@style/Theme.SleepTracking" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="SleepTracking"
            android:theme="@style/Theme.SleepTracking">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

STRING.XML

```
<resources>
    <string name="app_name">SleepTracking</string>
    <string name="title_activity_login">LoginActivity</string>
    <string
name="title_activity_registration">RegistrationActivity</string>
    <string name="title_activity_track">TrackActivity</string>
</resources>
```

APP DATABASE

```
package com.example.sleeptracking

//class AppDatabase {
//  package com.example.

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [TimeLog::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {

    abstract fun timeLogDao(): TimeLogDao
    companion object {
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            val tempInstance = INSTANCE
            if (tempInstance != null) {
                return tempInstance
            }
            synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                ).build()
                INSTANCE = instance
                return instance
            }
        }
    }
}

//}
```

USER_DATABASE_HELPER

```
package com.example.sleeptracking

//class UserDatabaseHelper {
//    package com.example.

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }

    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }

}
//}
```