

PROJECT REPORT TEMPLATE

1 INTRODUCTION

1.1 Overview

1.2 Purpose

2 PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map

2.2 Ideation and Brainstorming Map

3 RESULT

3.1 Signup Image

3.2 Login Image

3.3 Start time Image

3.4 Stop time Image

3.5 Sleep Tracking Records

4 ADVANTAGES AND DISADVANTAGES

4.1 Advantages

4.2 Disadvantages

5 APPLICATION

6 CONCLUSION

7 FUTURE SCOPE

8 APPENDIX

A. Source Code

1. INTRODUCTION

A sleep tracking app is a software application designed to monitor and analyse a user's sleep patterns. It typically utilizes sensors on the user's phone or wearable device to collect data such as the number of hours slept, time spent in different sleep stages (REM, light, and deep sleep), and movement during the night. This data is then analysed and presented to the user in an easy-to-understand format, allowing them to gain insights into their sleep quality and make adjustments to improve it.

The benefits of using a sleep tracking app include better understanding of one's sleep patterns, identifying potential sleep issues such as insomnia or sleep apnea, and making adjustments to sleep habits for improved health and well-being. Additionally, some sleep tracking apps offer features such as personalized sleep recommendations, meditation exercises, and bedtime reminders.

Sleep tracking apps are popular among people who prioritize their health and fitness, as well as those who suffer from sleep-related issues. With the increasing prevalence of wearable technology, sleep tracking apps are becoming more widely available and accessible to users.

1.1 OVERVIEW

Sleep tracking apps have become increasingly popular in recent years, as people become more aware of the importance of sleep for their overall health and well-being. These apps use various sensors and algorithms to monitor and analyse a user's sleep patterns, providing insights and recommendations to help them improve the quality and quantity of their sleep.

Here is an overview of some of the key features of sleep tracking apps:

- **Sleep monitoring:** Sleep tracking apps use sensors in smartphones or wearable devices to track the user's movements, heart rate, and other data to monitor their sleep patterns. They can detect when the user falls asleep, how long they sleep, how often they wake up, and other important sleep metrics.
- **Sleep analysis:** Once the sleep data is collected, the app uses algorithms to analyse the data and provide insights into the user's sleep patterns. For example, the app may show how much time the user spends in each stage of sleep (such

as deep sleep, light sleep, and REM sleep) and how many times they wake up during the night.

- **Sleep Environment:** The app can also monitor the user's sleep environment, including factors such as room temperature, humidity, and noise levels, and provide recommendations for creating a more conducive sleep environment.
- **Sleep Aids:** Some sleep tracking apps offer features such as guided meditations, white noise generators, or bedtime stories to help users relax and fall asleep more easily.
- **Reminders:** The app can provide reminders to the user to establish a consistent bedtime routine, to avoid caffeine or alcohol before bed, or to avoid using electronic devices before bedtime.

1.2 PURPOSE:

Improved sleep quality: By tracking and analysing sleep patterns, sleep tracking apps can provide valuable insights into sleep quality, including the duration and quality of different sleep stages. Users can use this information to identify areas for improvement and make changes to their sleep habits to promote better sleep quality.

Personalized sleep recommendations: Many sleep tracking apps offer personalized sleep recommendations based on the user's sleep patterns. These recommendations may include tips for improving sleep hygiene, establishing a consistent bedtime routine, and reducing stress and anxiety before bed.

Tracking progress: Sleep tracking apps allow users to track their progress over time, making it easier to see improvements in sleep quality and identify areas that still need work.

Better sleep environment: Some sleep tracking apps also monitor the user's sleep environment, including noise levels and room temperature. By providing insights into how these factors affect sleep quality, users can make changes to their sleep environment to create a more conducive sleeping environment.

Enhanced overall health: Getting enough quality sleep is essential for overall health and well-being. By improving sleep quality, sleep tracking apps can help users improve their

physical and mental health, including reducing the risk of chronic diseases and improving cognitive function.

Achieved using this application:

Improving sleep quality: By tracking sleep patterns and providing insights into factors that may be affecting sleep, such as noise levels or irregular sleep patterns, a sleep tracking app can help users identify areas for improvement and make changes to improve sleep quality.

Establishing healthy sleep habits: Sleep tracking apps can help users set personalized sleep goals and reminders, such as establishing a consistent bedtime routine or avoiding caffeine before bed, to establish healthy sleep habits and improve overall sleep quality.

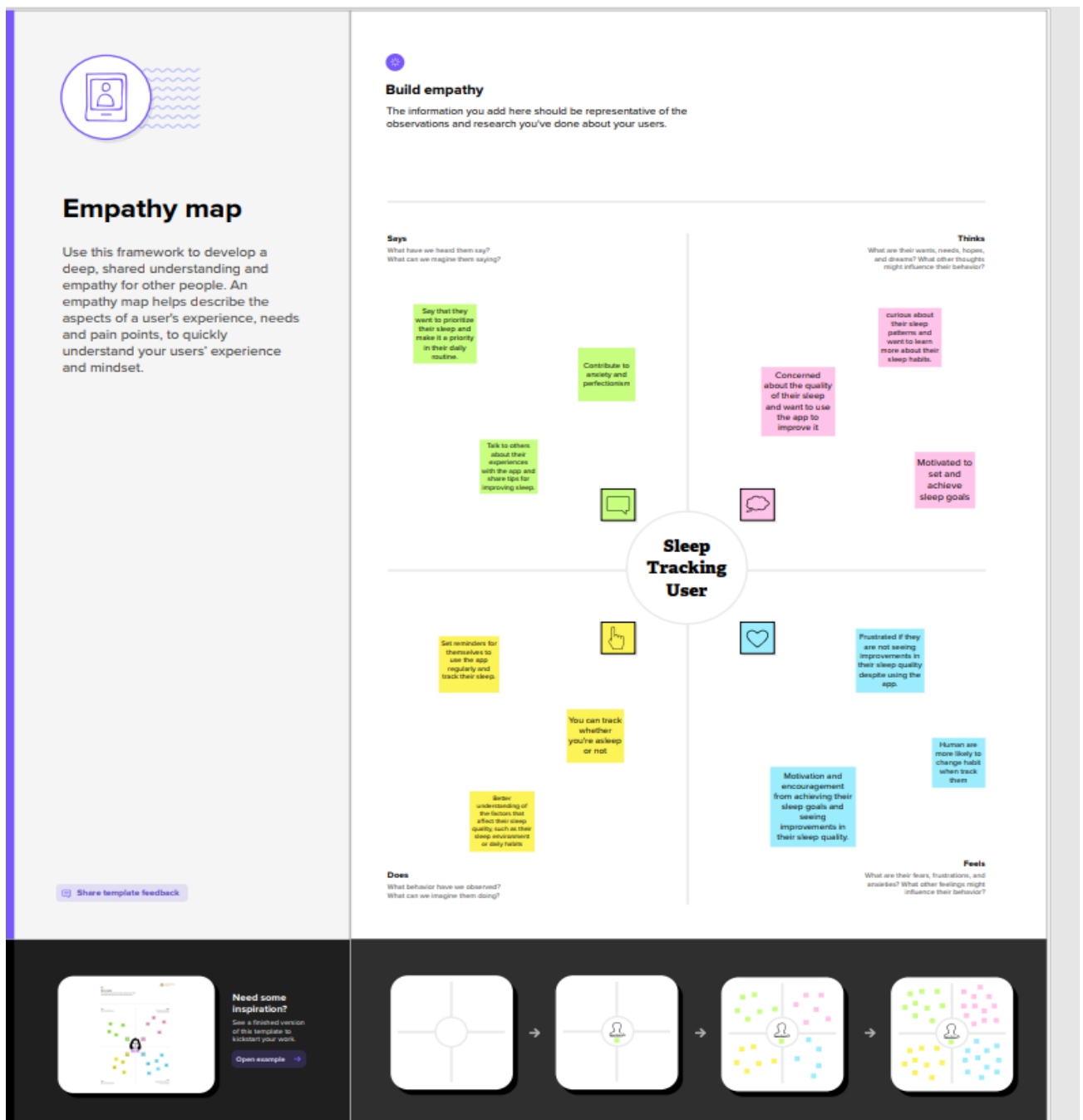
Increasing energy and productivity: Getting a good night's sleep is essential for maintaining energy and productivity throughout the day. By helping users improve their sleep quality, sleep tracking apps can help increase energy levels and improve overall productivity.

Reducing stress and anxiety: Poor sleep quality can contribute to feelings of stress and anxiety. By providing relaxation techniques and sleep aids, such as meditation exercises and white noise generators, sleep tracking apps can help users reduce stress and anxiety and improve overall mental health.

Monitoring health conditions: Sleep tracking apps can also be useful for monitoring health conditions, such as sleep apnea or insomnia, and working with a healthcare provider to develop a treatment plan.

2. PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 24 people recommended

[Share template feedback](#)

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

- 10 minutes

Define your problem statement

What problem are you trying to solve? Frame your problem as a **How Might We** statement. This will be the focus of your brainstorm.

- 5 minutes

Problem

How can we design a system that automatically sorts and organizes information based on user preferences and behavior? The system should be able to learn from user interactions and provide personalized recommendations and suggestions.

Key rules of brainstorming

Be as an unusual and productive session.

- Stay in focus.
- Encourage wild ideas.
- Defer judgment.
- Build on others.
- One idea per person.
- Be possible, but stretch.

Brainstorm

Write down any ideas that come to mind that address your problem statement.

- 10 minutes

TIP When collaborating in person, use sticky notes to capture ideas and then group them into themes.

K. BACHHETERA AFREEN	K. BACHHETERA	J. BACHHETERA	M. BACHHETERA
How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?
How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?
How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?
How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?	How can we design a system that automatically sorts and organizes information based on user preferences and behavior?

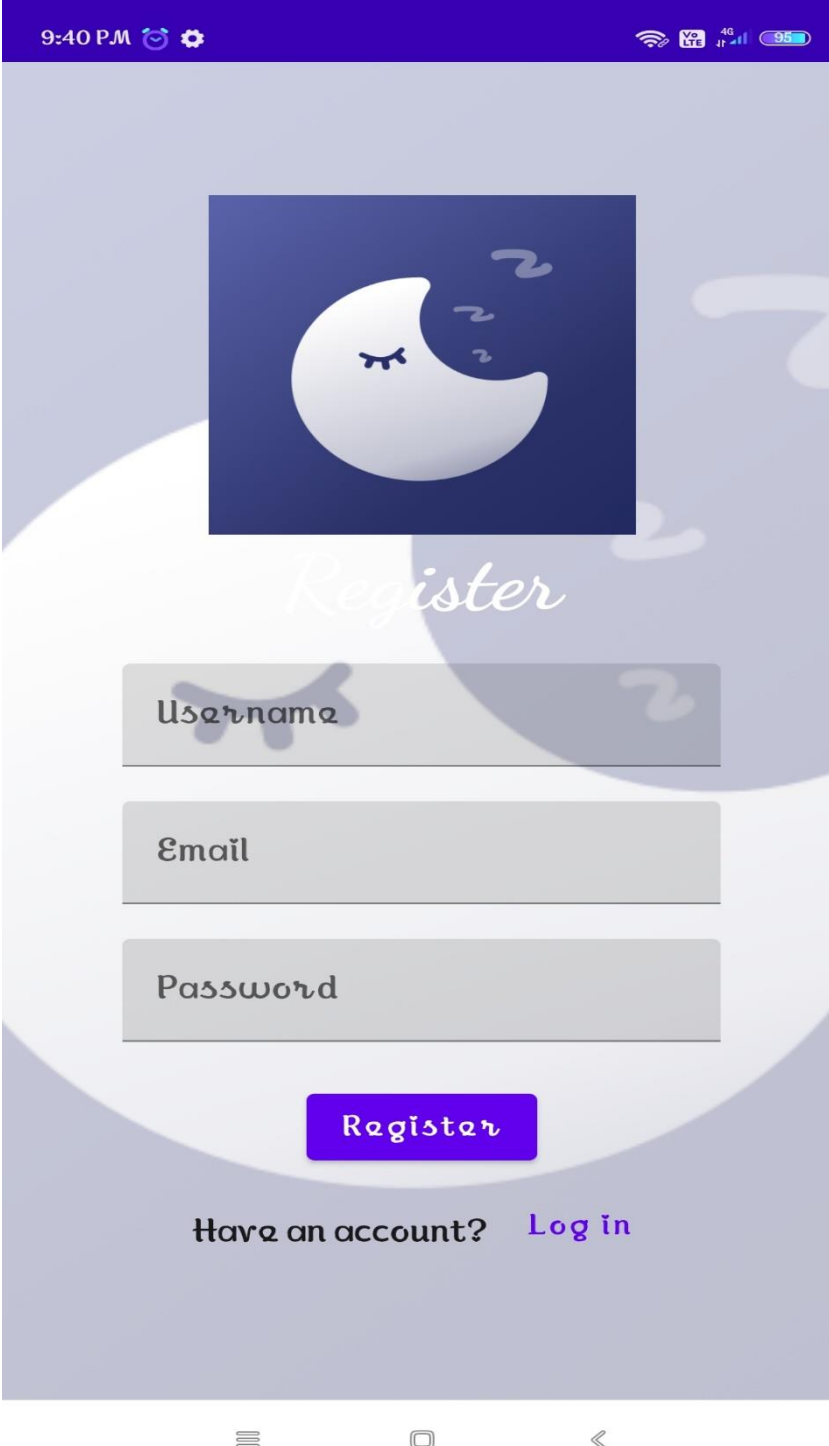
Need some inspiration?

Get a fresh perspective on your ideas by looking at other people's ideas.

[Brainstorming](#)


6

3. RESULT



A screenshot of a mobile application's registration screen. The status bar at the top shows the time as 9:40 P.M., along with icons for a clock, settings, Wi-Fi, LTE, 4G, and a battery level of 95%. The app's logo, featuring a stylized crescent moon with closed eyes and three small 'Z' marks above it, is centered in a dark blue square. Below the logo, the word "Register" is written in a white, cursive font. There are three light gray input fields for "Username", "Email", and "Password". A solid blue "Register" button is positioned below the fields. At the bottom, the text "Have an account?" is followed by a blue "Log in" link. The background is a light blue gradient with faint, stylized cloud patterns. The bottom of the screen shows three standard Android navigation icons: a hamburger menu, a home button, and a back arrow.

9:40 P.M. ⌚ ⚙️ 📶 LTE 4G 🔋 95%



Register

Username

Email

Password

Register

Have an account? [Log in](#)

Fig: 3.1 Signup Image

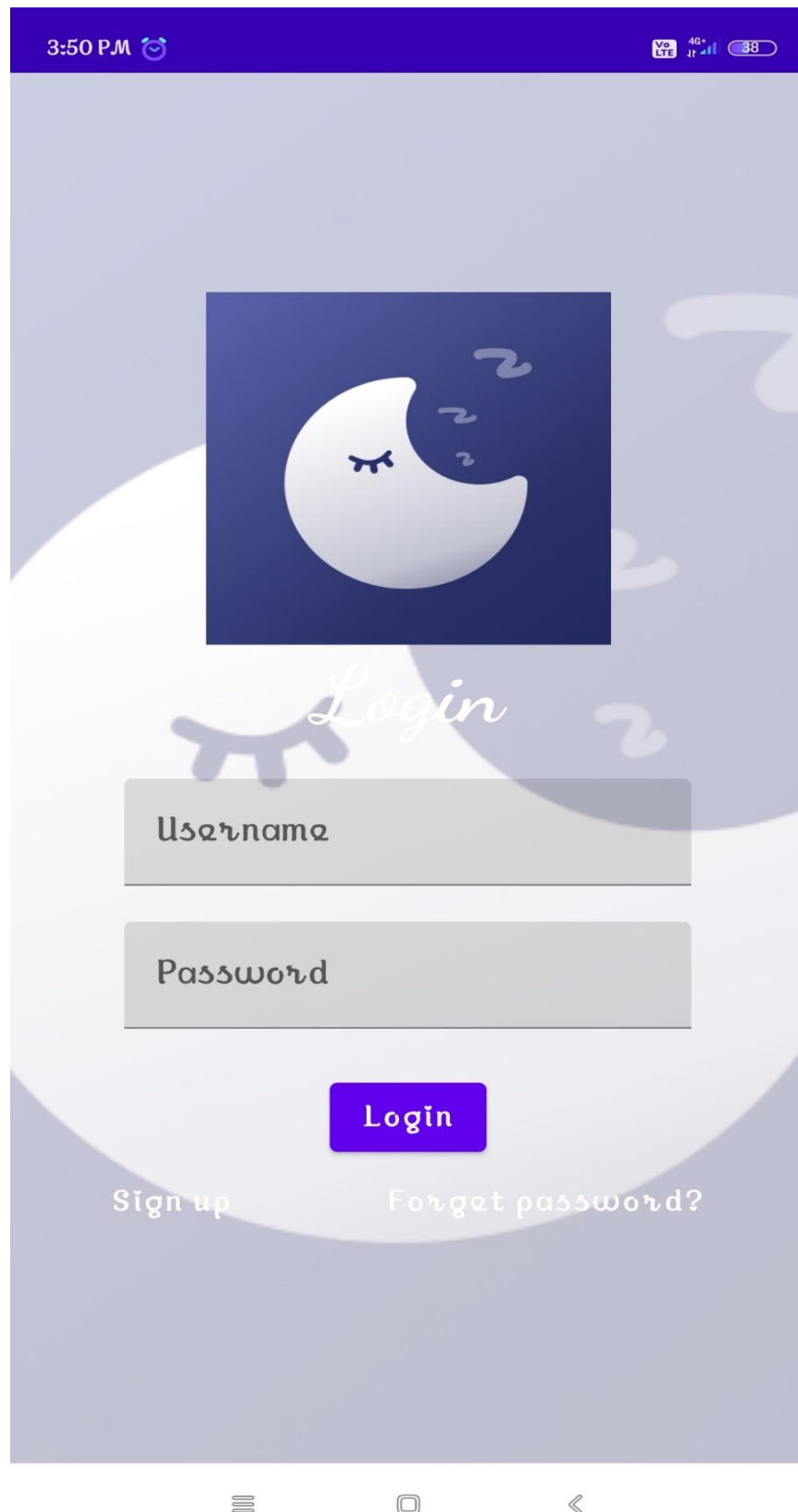


Fig:3.2 Login Image



Fig:3.3 Start time Image



Fig:3.4. Stop time Image



Fig:3.5 Sleep Tracking Records Image

4. ADVANTAGES AND DISADVANTAGES

4.1 Advantages:

- Provides insights into your sleep patterns: A sleep tracking app helps you understand your sleeping habits and patterns by monitoring your sleep stages and recording your sleep metrics.
- Helps you identify sleep disorders: The app can alert you if you experience any sleep disorders or disturbances like snoring, sleep apnea, and restless legs syndrome.
- Provides personalized recommendations: Based on your sleep data, the app can suggest personalized recommendations such as sleep hygiene tips, relaxation techniques, and sleep-inducing activities.
- Monitors your daily activities: Some sleep tracking apps can track your daily activities like exercise, caffeine consumption, and alcohol intake, which can impact your sleep quality.
- Tracks your sleep environment: The app can record your bedroom temperature, noise levels, and light exposure, which can impact your sleep quality and quantity.
- Provides comprehensive sleep analysis: A sleep tracking app can provide a comprehensive analysis of your sleep quality, including sleep duration, sleep stages, and sleep efficiency.
- Helps you set sleep goals: The app can help you set and achieve sleep goals by providing a sleep diary and setting reminders to ensure you get adequate sleep.
- Provides motivation and encouragement: Some sleep tracking apps provide motivation and encouragement by celebrating your sleep milestones and progress towards your sleep goals.
- Improves overall health and wellness: By improving your sleep quality, a sleep tracking app can contribute to your overall health and wellness, including reducing stress levels, improving cognitive function, and boosting your immune system.
- Easy to use: Most sleep tracking apps are easy to use and require minimal effort, allowing you to track your sleep data effortlessly.

4.2 Disadvantages:

- **Inaccuracy:** Sleep tracking apps may not always accurately measure your sleep patterns, especially if you move around a lot during the night or if the app is not properly calibrated.
- **Battery drain:** Sleep tracking apps often require a lot of battery power to operate, which can drain your phone battery quickly.
- **Dependency:** Using a sleep tracking app can lead to a dependence on technology to regulate your sleep, which can be harmful if the app fails or if you become too reliant on it.
- **Privacy concerns:** Sleep tracking apps often collect personal data such as your location and other sensitive information, which can be a privacy concern for some users.
- **Cost:** Some sleep tracking apps may require a subscription or purchase to access all features, which can be a financial burden for some users.
- **Sleep anxiety:** Using a sleep tracking app can also lead to sleep anxiety, as users may become overly concerned with their sleep patterns and quality, leading to sleep disturbances.
- **Invasive:** Some sleep tracking apps may require you to wear a wearable device or place your phone under your pillow, which can be invasive or uncomfortable for some users.
- **Limited data:** Sleep tracking apps may not provide enough data to make meaningful changes to your sleep habits, or the data may be difficult to interpret without medical training.
- **Misinterpretation:** Sleep tracking apps may misinterpret sleep data, leading to incorrect conclusions or recommendations about your sleep patterns.
- **Overreliance:** Using a sleep tracking app may lead to an overreliance on technology to manage your sleep, instead of addressing underlying health issues or lifestyle factors that may be affecting your sleep.

5. APPLICATIONS

- **Personal Use:** Sleep tracking apps can be used by individuals who want to monitor their sleep patterns and improve their sleep quality. These apps can help track the amount of time spent in different sleep stages, including deep sleep, light sleep, and REM sleep.
- **Clinical Use:** Sleep tracking apps can be used by healthcare professionals to diagnose and treat sleep disorders such as sleep apnea, insomnia, and restless leg syndrome. These apps can provide valuable insights into a patient's sleep patterns and help doctors tailor treatment plans accordingly.
- **Athletic Performance:** Sleep tracking apps can be used by athletes and coaches to monitor the impact of sleep on athletic performance. These apps can help athletes optimize their sleep patterns to improve recovery, reduce fatigue, and enhance their overall performance.
- **Workplace Wellness Programs:** Sleep tracking apps can be used as part of workplace wellness programs to help employees monitor their sleep patterns and improve their overall health and productivity. These apps can provide valuable insights into the impact of sleep on work performance and help employees develop healthy sleep habits.
- **Research:** Sleep tracking apps can be used by researchers to gather data on sleep patterns and behaviours. This data can be used to study the impact of sleep on various health outcomes and to develop new treatments and interventions for sleep disorders.
- **Medical settings:** Sleep tracking apps can be used by medical professionals to monitor the sleep patterns of patients with sleep disorders such as insomnia, sleep apnea, or restless leg syndrome. This data can help healthcare providers develop personalized treatment plans for their patients.

6. CONCLUSION

In conclusion, sleep tracking apps can be a valuable tool for monitoring and improving sleep quality, providing users with insights into their sleep patterns and personalized recommendations for improving sleep habits. By tracking factors such as movement, heart rate, and noise levels, these apps can provide detailed analysis of the different stages of sleep, including deep sleep, REM sleep, and light sleep, as well as information on how long it takes to fall asleep, how many times the user wakes up during the night, and the total amount of time spent asleep.

However, there are also some potential disadvantages to consider when using a sleep tracking app, including inaccurate data, battery drain, privacy concerns, obsessive tracking, and false reassurance. It's important for users to carefully review the app's privacy policy and settings before using it, and to use the app in moderation, as part of a broader effort to improve sleep habits and overall health and wellness.

Overall, sleep tracking apps can be a valuable tool for anyone looking to improve their sleep quality and overall health and wellness. By providing personalized insights and recommendations, these apps can help users achieve better sleep, reduce stress and anxiety, and improve energy and productivity throughout the day.

In Summary, a useful tool for anyone looking to improve their sleep quality and overall health and wellness. It can help users establish healthy sleep habits, improve energy levels and productivity, reduce stress and anxiety, monitor health conditions, and achieve better sleep. However, it's important to use the app in moderation, be aware of potential drawbacks, and seek professional advice if experiencing sleep-related health issues.

7. FUTURE ENHANCEMENT

- **More accurate data collection:** As sensor technology improves, sleep tracking apps will become more accurate, providing users with more detailed and reliable data on their sleep patterns.
- **Integration with other health data:** Sleep tracking apps could be integrated with other health data, such as fitness tracking data, to provide a more complete picture of overall health and wellness.
- **Artificial intelligence and machine learning:** By using artificial intelligence and machine learning algorithms, sleep tracking apps could become even more personalized, providing users with tailored recommendations based on their individual sleep patterns and health goals.
- **Wearable devices:** As wearable technology becomes more advanced, sleep tracking apps could be integrated into devices such as smartwatches and fitness trackers, making sleep tracking even more convenient and accessible.
- **Medical applications:** Sleep tracking apps could be used in medical applications, such as monitoring patients with sleep disorders or tracking the sleep patterns of individuals with chronic illnesses.
- **Augmented reality (AR) and virtual reality (VR):** AR and VR technology could be used to provide immersive sleep environments, such as simulated nature scenes or calming soundscapes, to help users fall asleep more quickly and stay asleep longer.
- **Biofeedback and neurofeedback:** Future sleep tracking apps may incorporate biofeedback and neurofeedback techniques to help users better understand their sleep patterns and learn to regulate their own sleep cycles.
- **Collaboration with healthcare providers:** As our understanding of sleep and its impact on overall health continues to improve, sleep tracking apps may be designed to collaborate with healthcare providers to monitor and treat sleep disorders such as sleep apnea, insomnia, and restless leg syndrome.

8. APPENDIX

A. Source Code

MainActivity

```
package com.example.sleeptracking

import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Button
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.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.colors.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) }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),

```

```

contentScale = ContentScale.FillHeight,
contentDescription = "",
modifier = imageModifier
.alpha(0.3F),
)

```

```

Column(
modifier = Modifier.fillMaxSize(),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
) {
if (!isRunning) {
Button(onClick = {
startTime = System.currentTimeMillis()
isRunning = true
}) {
Text("Start")
//databaseHelper.addTimeLog(startTime)
}
} else {
Button(onClick = {
elapsedTime = System.currentTimeMillis()
isRunning = false
}) {
Text("Stop")
databaseHelper.addTimeLog(elapsedTime,startTime)
}
}
Spacer(modifier = Modifier.height(16.dp))
Text(text = "Elapsed Time: ${formatTime(elapsedTime - startTime)}")

```

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

```
Button(onClick = { context.startActivity(
```

```
Intent(
```

```
context,
```

```
TrackActivity::class.java
```

```
)
```

```
) }) {
```

```
Text(text = "Track Sleep")
```

```
}
```

```
}
```

```
}
```

```
private fun startTrackActivity(context: Context) {
```

```
val intent = Intent(context, TrackActivity::class.java)
```

```
ContextCompat.startActivity(context, intent, null)
```

```
}
```

```
@RequiresApi(Build.VERSION_CODES.N)
```

```
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)
}
```

AppDatabase

```
package com.example.sleeptracking

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
}
}
}
}

```

TimeDatabaseHelper

```

package com.example.sleeptracking
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"
}
}
}
}

```

TimeLog

```

package com.example.sleeptracking
import androidx.room.Entity
import androidx.room.PrimaryKey
import java.sql.Date
@Entity(tableName = "TimeLog")

```



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

TimeLogDao

```
package com.example.sleeptracking  
  
import androidx.room.Dao  
import androidx.room.Insert  
  
@Dao  
interface TimeLogDao {  
  
    @Insert  
    suspend fun insert(timeLog: TimeLog)  
  
}
```

User

```
package com.example.sleeptracking  
  
import androidx.room.ColumnInfo  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@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?,
)
```

UserDao

```
package com.example.sleeptracking
import androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

UserDatabase

```
package com.example.sleeptracking
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao
}
```

```

companion object {
    @Volatile
    private var instance: UserDatabase? = null

    fun getDatabase(context: Context): UserDatabase {
        return instance ?: synchronized(this) {
            val newInstance = Room.databaseBuilder(
                context.applicationContext,
                UserDatabase::class.java,
                "user_database"
            ).build()
            instance = newInstance
            newInstance
        }
    }
}

```

UserDatabaseHelper

```

package com.example.sleeptracking

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 androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

```

```

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
}
}

```

Register

```

package com.example.chatapp.view.register
import androidx.compose.foundation.Image
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.layout.R
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.Edit
import androidx.compose.material.icons.filled.Email
import androidx.compose.material.icons.filled.Lock
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color

```

```

import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.example.chatapp.view.Appbar
import com.example.chatapp.view.Buttons
import com.example.chatapp.view.TextFormField

import com.example.chatapp.view.register.ui.theme.ChatAppTheme

```

```

@Composable
fun RegisterView(
    home: () -> Unit,
    back: () -> Unit,
    login: () -> Unit,
    registerViewModel: RegisterViewModel = viewModel()
) {
    val email: String by registerViewModel.email.observeAsState("")
    val loading: Boolean by registerViewModel.loading.observeAsState(initial = false)
    val password: String by registerViewModel.password.observeAsState("")
    val name: String by registerViewModel.name.observeAsState("")
    val text = "Login"

    Surface(
        color = Color.White

```



```

) {
Box{
Image(painter = painterResource(id = com.example.chatapp.R.drawable.bg_login),
contentDescription = "bg",
contentScale = ContentScale.FillBounds,
modifier = Modifier
.fillMaxWidth()
.size(310.dp))
Column(
modifier = Modifier
.padding(
horizontal = 32.dp, vertical = 40.dp,
)
.fillMaxSize()
) {
val Purple200 = Color(0xFFBB86FC)
Image(painter = painterResource(id = com.example.chatapp.R.drawable.app_logo_2),
contentDescription = "app_logo",
modifier = Modifier
.fillMaxWidth()
.size(180.dp))
Spacer(modifier = Modifier.padding(bottom = 70.dp))
Text(text = "Welcome to", color = Color.White,
fontWeight = FontWeight.Black,
fontSize = 36.sp
)
Text(text = "Chat Connect", color = Purple200 ,
fontWeight = FontWeight.Black,
fontSize = 36.sp
)

```

```

Text(text = "Enter the following details and get connected",
color = Color.DarkGray,
//          fontWeight = FontWeight.Black,
fontSize = 14.sp
)

Spacer(modifier = Modifier.padding(bottom = 50.dp))
//
Column(
horizontalAlignment = Alignment.CenterHorizontally,
modifier = Modifier
.padding(0.dp)
.fillMaxSize()
) {
val Purple200 = Color(0xFFBB86FC)

OutlinedTextField(
value = name,
onValueChange = { registerViewModel.updateName(it) },

placeholder = { Text(text = "Full Name") },
colors = TextFieldDefaults.textFieldColors(
backgroundColor = Color.White,
),
leadingIcon = { Icon(imageVector = Icons.Default.Edit, contentDescription = "emailIcon") },
modifier = Modifier
.padding(bottom = 10.dp)
.fillMaxWidth()
)
OutlinedTextField(

```

```

value = email,
onValueChange = { registerViewModel.updateEmail(it) },

placeholder = { Text(text = "Email") },
colors = TextFieldDefaults.textFieldColors(
backgroundColor = Color.White,
),
leadingIcon = { Icon(imageVector = Icons.Default.Email, contentDescription = "emailIcon")
},
modifier = Modifier
.padding(bottom = 10.dp)
.fillMaxWidth()
)
OutlinedTextField(
value = password,
onValueChange = { registerViewModel.updatePassword(it) },
placeholder = { Text(text = "Password") },
leadingIcon = { Icon(imageVector = Icons.Default.Lock, contentDescription = "emailIcon") },

colors = TextFieldDefaults.textFieldColors(
backgroundColor = Color.White,
),
modifier = Modifier
.padding(bottom = 20.dp)
.fillMaxWidth()
)
Button( onClick = { registerViewModel.registerUser(home = home) },

shape = RoundedCornerShape(percent = 50),
modifier = Modifier.border(width = 1.dp,

```

```

color = Color.White,
shape = RoundedCornerShape(percent = 50),
),
colors = ButtonDefaults.buttonColors(
backgroundColor = Purple200,
contentColor = Color.White)) {
Text(text = "Sign Up",
modifier = Modifier.padding(horizontal = 40.dp, vertical = 4.dp),
fontSize = 15.sp,
fontWeight = FontWeight.SemiBold
)
}
if (loading) { CircularProgressIndicator()
}
Row(
modifier = Modifier.padding(top = 30.dp)
) {
Text(text = "Already have an account? ", color = Color.Black)
Text(
modifier = Modifier
.clickable(enabled = true) {
login()
},
text = text,
color = Purple200,
)
}
}
//      }
}

```

```
}  
}  
}
```

RegistrationActivity

```
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.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 RegistrationActivity : 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.colors.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.sleeptracking),
        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.sleeptracking),
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.colors.error,
    modifier = Modifier.padding(vertical = 16.dp)
  )
}

```



```

Button(
onClick = {
if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {
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 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 = {
        startLoginActivity(context)
    })

    {
        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)
    }

```

LoginActivity

```

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.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 androidx.core.content.ContextCompat.startActivity
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.colors.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.sleeptracking),  
        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.sleeptracking),  
            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,
  onChange = { username = it },
  label = { Text("Username") },
  modifier = Modifier.padding(10.dp)
  .width(280.dp)
)

```

```

TextField(
  value = password,
  onChange = { password = it },
  label = { Text("Password") },
  modifier = Modifier.padding(10.dp)
  .width(280.dp)
)
if (error.isNotEmpty()) {
  Text(
    text = error,
    color = MaterialTheme.colors.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,
RegistrationActivity::class.java
)
)}
)
{ Text(color = Color.White,text = "Sign up") }
TextButton(onClick = {
//      startActivity(
//      Intent(
//      applicationContext,
//      MainActivity2::class.java
//      )
//      )
//      context.startActivity(
//      Intent(
//      context,
//      RegistrationActivity::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)
}
private fun startRegisterPage(context: Context) {
val intent = Intent(context, RegistrationActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}

```

TrackActivity

```

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.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 androidx.core.content.ContextCompat.startActivity
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.colors.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.sleeptracking),
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.sleeptracking),
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.colors.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,
RegistrationActivity::class.java
)
)})
)
{ Text(color = Color.White,text = "Sign up") }
TextButton(onClick = {
//      startActivity(
//      Intent(

```

```

//          applicationContext,
//          MainActivity2::class.java
//      )
//  )
//      context.startActivity(
//          Intent(
//              context,
//              RegistrationActivity::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)
}

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

```