

Android Application Development

A Sleep Tracking App for a Better Night's Rest

Submitted By

Induri Sirisha (Team Leader)- 20KE1A0533

T.Sushma (Team Member) - 20KE1A05A2

R.Deekshitha(Team Member)- 20KE1A0589

Under the Guidance of mentor

CHARLES JAYARAO

ASSISTANT PROFESSOR

Department of Computer Science

Malineni Lakshmaiah Womens Engineering College, Pulladigunta

Guntur-



DEPARTMENT OF COMPUTER SCIENCE

**MALINENI LAKSHMAIAH WOMENS
ENGINEERING COLLEGE, PULLADIGUNTA
GUNTUR**

LIST OF CONTENT

CHAPTER NO	TITLES	PAGE NO
I	INTRODUCTION	3
	1.1. OVERVIEW	3
	1.2. PURPOSE	6
II	PROBLEM DEFINITION & DESIGN THINKING	8
	2.1. EMPATHY MAP	8
	2.2 IDEATION & BRAINSTORMING MAP	11
III	RESULT	13
	3.1. LOGIN PAGE	13
	3.2. REGISTRATION PAGE	15
	3.3. MAIN PAGE	17
	3.4. SLEEP TRACKING PAGE	19
IV	ADVANTAGES & DISADVANTAGES	21
	4.1. ADVANTAGES	21
	4.2. DISADVANTAGES	22
V	APPLICATIONS	24
VI	CONCLUSION	25
VII	FUTURE SCOPE	26
VIII	APPENDIX	27

Chapter – 1

Introduction

1.1 Overview:

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.

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.

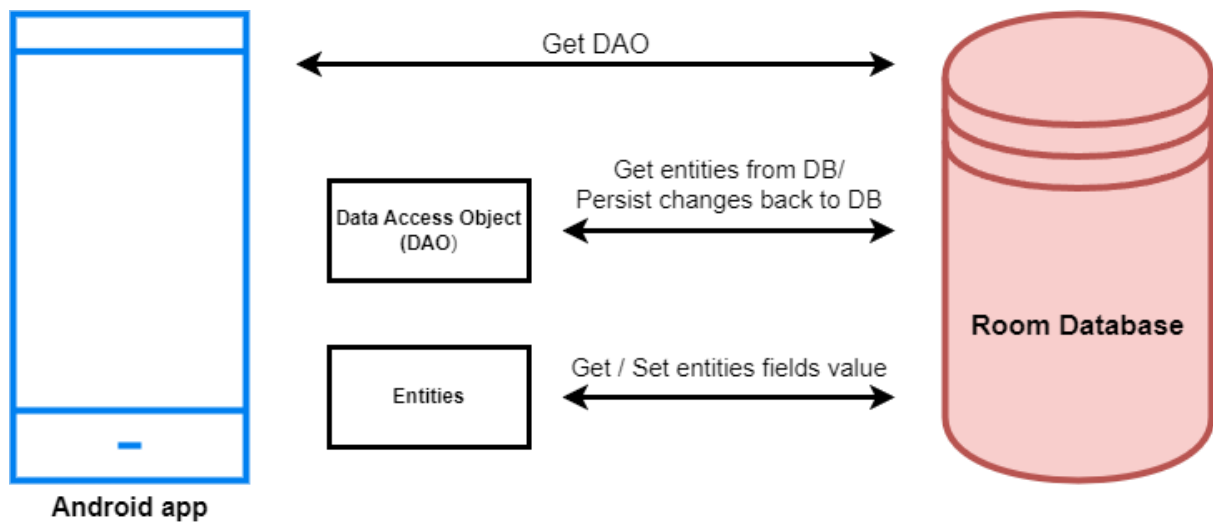
Our phones have become great tools for making more informed decisions about our sleep. And by being informed about sleep habits, people can make better decisions throughout the day about sleep, which affects things like concentration and mental health. The Sleep Application Programming Interface is an Android Activity Recognition API that surfaces information about the user's sleep. It can 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 'sleep confidence', which is reported at a regular interval.
- 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. The app still running smoothly and providing accurate sleep data. By running in the background, the app can also use a variety of sensors to track sleep without the need for the user to keep the app open and actively tracking. This can improve the user experience by reducing the need to constantly interact with the app and potentially drain the battery.

Architecture:



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.

1.2 Purpose:

Sleep is a fundamental aspect of human health and well-being. It is essential for physical and mental restoration and plays a critical role in maintaining optimal cognitive function, mood, and overall quality of life. However, many individuals struggle with getting enough high-quality sleep due to various factors, such as busy lifestyles, stress, and environmental factors.

The purpose of a sleep tracking application, like the one described in the project workflow, is to help individuals improve their sleep quality and overall health by providing personalized insights and recommendations based on their sleep data. Using a sleep tracking app can be a valuable tool for improving the quality and quantity of your sleep. By tracking your sleep patterns and providing insights into factors that may be affecting your sleep, such as bedtime routine, environmental conditions, and lifestyle choices, the app can help you make changes to improve your sleep habits.

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.

The app can provide you with data to share with your healthcare provider, enabling you to receive appropriate medical treatment or referral to a specialist. Overall, using a sleep tracking app can help you take a proactive approach to your sleep health, and can ultimately lead to better overall wellness.

A sleep tracking app for children can have several purposes, including:

- **Monitoring sleep patterns:**

By tracking a child's sleep patterns, parents can better understand their child's sleep habits and identify any potential sleep problems or disturbances.

- **Promoting healthy sleep habits:**

Sleep tracking apps can help parents establish a consistent sleep routine for their child, which can improve the quality and duration of their sleep.

- **Identifying sleep disorders:**

If a child consistently has trouble sleeping, a sleep tracking app can help parents identify potential sleep disorders and seek medical advice.

- **Providing insights:**

By analysing sleep data over time, sleep tracking apps can provide insights into how certain activities or behaviours affect a child's sleep, which can help parents make informed decisions about their child's sleep habits.

Chapter- 2

Problem Definitions & Design Thinking

2.1. Empathy Map:

An empathy map is a tool used to gain a deeper understanding of a particular target audience, such as customers or users. It is a visual representation of the thoughts, feelings, and behaviours of the target audience that helps to create a more comprehensive understanding of their needs and motivations. Empathy maps typically consist of a simple diagram with four quadrants: "Says," "Thinks," "Feels," and "Does." Each quadrant is filled with relevant information that is gathered through research or observation, and can include things like quotes, actions, and emotions. The purpose of an empathy map is to help teams or individuals gain a more empathetic perspective on their target audience, which can help them design products, services, or experiences that better meet their needs. By creating an empathy map, businesses can develop a more human-centered approach to design and innovation, which can ultimately lead to better outcomes for both the business and its customers.



Says

"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."

Thinks

"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."

Does

Sets bedtime routine

Adjusts sleep environment

Tries relaxation techniques

Tries different sleep aids

Feels

Frustrated

Anxious

Worried

Hopeful

Says:

- "I want to sleep better at night."
- "I feel tired in the morning, even after sleeping for 8 hours."
- "I have trouble falling asleep at night."
- "I wake up several times during the night."

Thinks:

- "I need to know if my sleep patterns are affecting my health."
- "I wonder if I'm getting enough quality sleep."
- "I'm curious about how my sleep habits compare to others."
- "I don't know to optimize my sleep."

Feels:

- Frustrated
- Anxious
- Hopeful
- Worried

Does:

- Tries to create a consistent sleep schedule.
- Tries relaxation techniques.
- Tries different sleep aids or medication.
- Adjust sleep environment.

By creating an empathy map, the developers of the sleep tracking app can better understand the needs and motivations of their target user.

2.2. Ideation & Brainstorming Map:

Identify the problem or opportunity: The first step is to identify the problem or opportunity that the sleep tracking app will address. In this case, the problem could be that many people struggle with getting enough quality sleep, and the opportunity could be to help people improve their sleep habits and feel more rested.

Brainstorm ideas: Next, brainstorm ideas for the sleep tracking app. The app could track sleep patterns and provide personalized recommendations for improving sleep, such as adjusting the user's sleep schedule or providing relaxation techniques before bedtime.

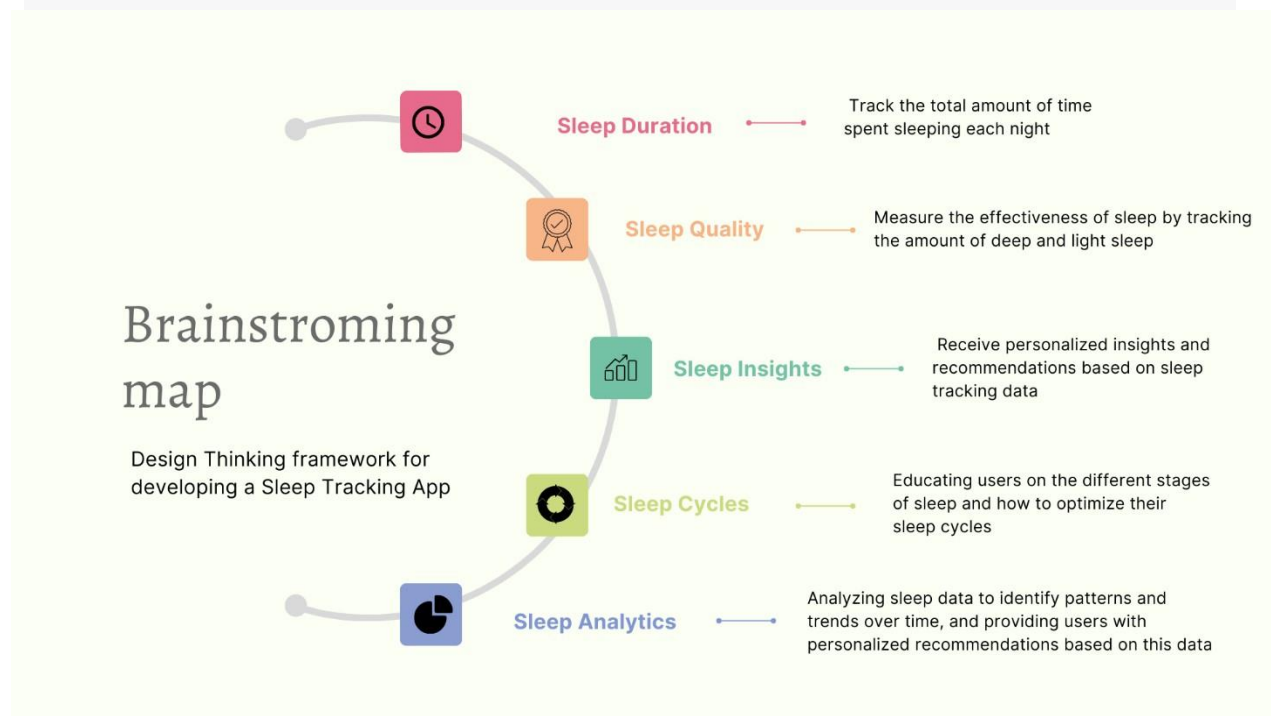
Evaluate ideas: Once ideas are generated, evaluate each one based on factors such as feasibility, impact, and alignment with project goals. Some features may be more technically challenging to implement than others, or may be more likely to help users improve their sleep habits.

Refine the ideas: This could involve conducting user research to better understand user needs and creating prototypes to test the app's functionality and usability.

Select the best idea: After refining the ideas, select the best idea and develop a plan for executing it. This could involve creating a development timeline, identifying resources and stakeholders, and creating a budget.

Brainstorming Map:

Brainstorming maps are visual tools that can be used for a variety of purposes, such as generating ideas, problem-solving, planning, communication, and evaluation. These maps are useful for organizing and connecting different ideas, which can lead to creative and innovative solutions. Brainstorming maps can help to do this by visually organizing ideas and connections, which can make it easier to see patterns and identify new opportunities.



Chapter – 3

Result






3.1. Login page:


The login page of a sleep tracking app is typically the first screen that a user sees when they open the app. The purpose of the login page is to allow users to access their account by entering their login credentials, such as their email address and password.

The login page typically consists of two main elements:

- The login form
- The sign up button

The login form typically includes fields for the user's email address and password. Once the user enters their login credentials, they can click the "login" button to access their account. If the user forgets their password, there is usually a "forgot password" link or button that they can click to reset their password. The sign up button is usually located below the login form and is used to allow new users to create an account. Once the user has entered all of their information, they can click the "sign up" button to create their account. In summary, the login page of a sleep tracking app typically includes a login form where users can enter their login credentials to access their account, as well as a registration button that allows new users to create an account by providing their personal information.

7:34 PM 8.5KB/s     



Register

Username

Email

Password

[Register](#)

Have an account? [Log in](#)

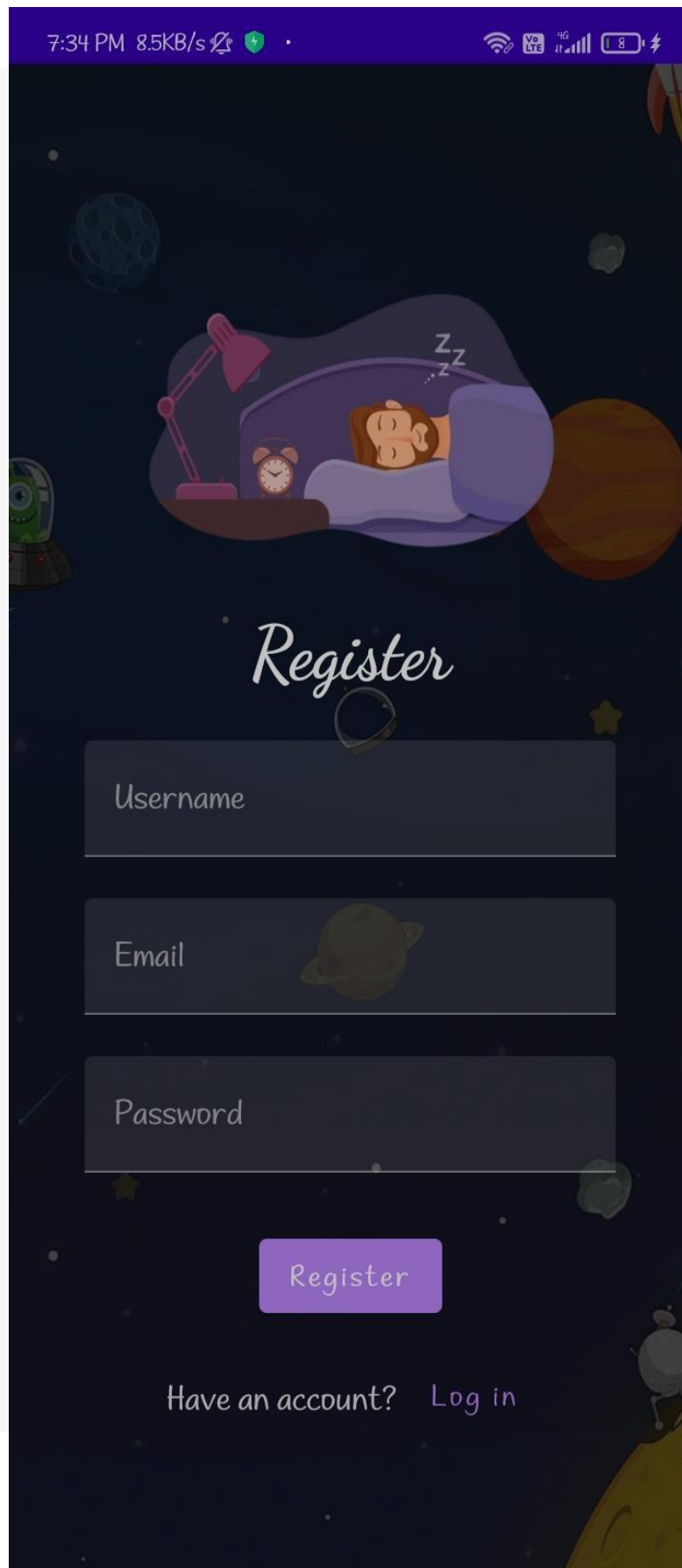
3.2. Registration page:

A registration page of a sleep tracking app typically includes a form where new users can create an account by providing their personal information such as their username, email address, and a password. The purpose of this page is to gather the necessary information to create a unique user profile that can be used to track and analyse sleep patterns.

Once the user fills out the required fields and clicks on the "Sign up" button, the app will create a new user account and redirect them to the login page. The login page typically includes a form where users can enter their username and password to access their account.

The login button is an essential part of the login page, as it allows users to securely access their account by verifying their credentials. When a user clicks on the login button, the app will compare the entered email and password with the stored user data and authenticate the user if the information matches.

If the user enters incorrect login credentials, the app will display an error message and prompt them to enter their credentials again. By collecting relevant user information and securely authenticating user accounts, the app can effectively track and analyse sleep patterns to provide valuable insights and recommendations to users.



3.3. Main page:

1. Start button/ Stop button:

This button is used to start tracking your sleep. When you are ready to go to bed, you would click on the "Start" button to begin tracking your sleep.

When you wake up in the morning, you would click on the "Stop" button to stop tracking your sleep. This will record the total amount of time you slept for.

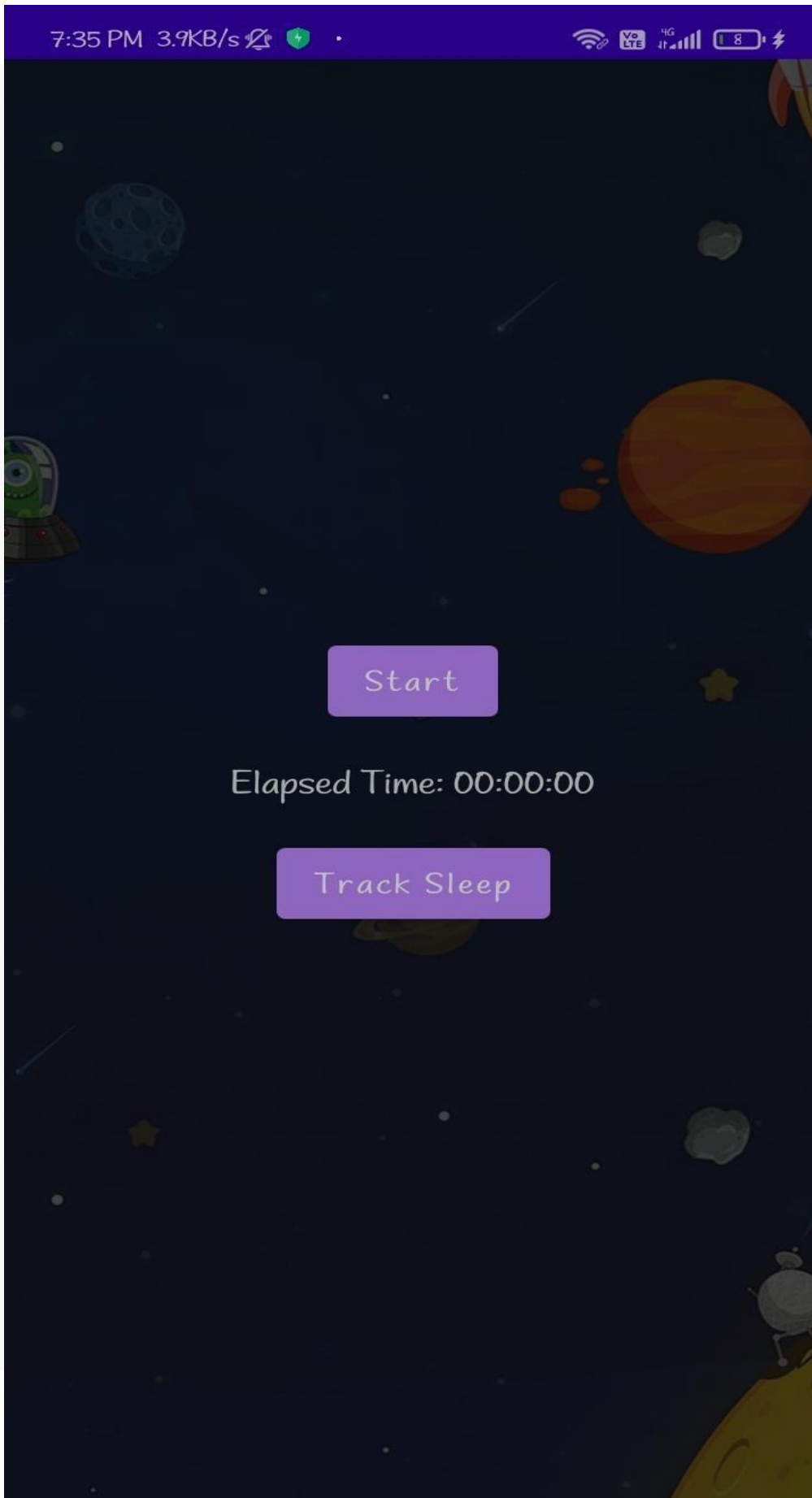
2. Elapsed time:

Once you click on the "Start" button, the elapsed time starts counting. This will help you keep track of how long you have been sleeping for.

3. Track button:

The track button is used to move to the next page of the application. This page will allow you to view the list of sleep tracking records that you have previously recorded.

Overall, this sleep tracking application seems to be designed to help you monitor and track your sleep patterns over time. By recording this information, you can better understand your sleep habits and make adjustments to improve the quality and duration of your sleep.



3.4. Sleep Tracking page:

A tracking page that displays a user's sleep start time to end time and shows the history of their sleep is typically a tool used to monitor and analyse a person's sleeping patterns over time. This type of tracking page can provide insights into the quality and duration of a user's sleep, allowing them to make adjustments to their sleep habits and improve their overall well-being.

One of the key benefits of a sleep tracking page is that it can help users identify patterns and trends in their sleep habits. For example, they may notice that they tend to sleep longer on weekends than weekdays, or that they have trouble falling asleep on nights when they exercise late in the day.

Armed with this information, users can make changes to their sleep routines and habits to optimize their sleep and improve their overall health and well-being.

Overall, a sleep tracking page that displays a user's sleep start time to end time and shows the history of their sleep can be a powerful tool for improving sleep habits and optimizing health. By providing detailed insights into sleeping patterns, these tracking pages can help users make informed decisions about their sleep habits and ultimately lead to better rest, more energy, and improved overall health.

7:36 PM 17.1KB/s



Sleep Tracking

Start time: 1970-01-01 05:30:00

End time: 2023-04-16 19:35:24

Start time: 2023-04-16 19:35:27

End time: 2023-04-16 19:35:28

Chapter – 4

Advantage and Disadvantage

4.1. Advantages:

- **Provides insight into sleeping patterns:** Sleep tracking apps can provide detailed information about a user's sleep patterns, including the duration of their sleep, the quality of their sleep, and how often they wake up during the night. This information can help users identify patterns and make changes to their sleep routines to improve the quality of their sleep.
- **Helps users set sleep goals:** Sleep tracking apps often allow users to set goals for their sleep, such as a target number of hours per night or a desired level of sleep quality. These goals can serve as motivation to improve sleep habits and achieve better sleep.
- **Monitors sleep-related health issues:** Sleep tracking apps can also help monitor and identify sleep-related health issues, such as restless leg syndrome. By tracking sleep patterns over time, users may notice changes that could indicate the presence of a health issue and seek medical attention if needed.
- **Provides personalized recommendations:** Some sleep tracking apps use the data they collect to provide personalized recommendations for improving sleep habits. These recommendations may include adjusting bedtime

routines, changing sleep environments, or making lifestyle changes to support better sleep.

- **Improves overall health and well-being:** By improving sleep quality and duration, sleep tracking apps can have a positive impact on overall health and well-being. Better sleep has been linked to improved cognitive function, reduced stress levels, and a lower risk of developing chronic health conditions such as obesity, diabetes, and heart disease.
- **Awareness:** Sleep tracking apps can provide insights into your sleep patterns and help you become more aware of your sleep habits. You can learn how much sleep you are getting, how restful your sleep is, and how long it takes you to fall asleep.
- **Motivation:** Seeing progress over time can be motivating. Make your sleep patterns as chart or graphs and progress towards your goals. This can help you stay motivated to continue improving your sleep.

4.2. Disadvantages:

- **Accuracy:** Sleep tracking apps are not always accurate. They may not be able to differentiate between light and deep sleep, which can affect the accuracy of the data. Additionally, sleep quality can be affected by various factors such as stress, diet, and physical activity, which cannot always be accurately measured by a sleep tracking app.

-
- **Battery drain:** To track your sleep throughout the night, sleep tracking apps require your phone to be running continuously. This can lead to a significant drain on your phone's battery, and you

may need to charge your phone more frequently or have a backup charger available. Furthermore, the battery drain caused by a sleep tracking app can potentially impact the performance of your phone during the day, making it less responsive or causing other apps to run slower.

- **Obsessive behaviour:** Sleep tracking apps can provide you with a lot of detailed information about your sleep patterns, including the amount of time spent in each sleep stage, how many times you woke up, and how long it took you to fall asleep. While this information can be useful for identifying trends and making adjustments to your sleep routine, it can also lead to obsessive behaviour and anxiety. Constantly checking your sleep data and worrying about improving your sleep quality can have a negative impact on your mental health and may even make it harder for you to fall asleep.
- **Privacy concerns:** Sleep tracking apps collect a lot of personal data about your sleep habits. There is a risk that this data could be used for other purposes or fall into the wrong hands. This information can be very valuable to marketers, researchers, and other third parties who may be interested in analysing sleep data

for their own purposes. There is a risk that this data could be sold or shared with other companies, or even used for targeted advertising.

Chapter- 5

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- 6

Conclusion:

Sleep tracking apps have the potential to be useful tools for improving sleep habits, managing sleep disorders, and optimizing performance in various domains. They can provide valuable data for sleep researchers and help individuals monitor their sleep quality and identify potential triggers for sleep disturbances. In covid pandemic, users prefer to have distant contact with their doctor for sleep issues. Sleep tracking app uses to monitor our sleep and that can helps to taking care of ourselves by using the sleep tracking app.

Chapter- 7

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.

Chapter- 8 Appendix:

A. Source code:

1. Adding Room dependencies (sync)

```
implementation 'androidx.room:room-common:2.5.0' implementation  
'androidx.room:room-ktx:2.5.0'
```

2. AppDatabase.kt

```
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
    }
    }
    }
    }

```

3. Time Log

```

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)

```

4. Time Log

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

5. 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?,
)
```

6. 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)
}
```

7. UserDatabase:

```

    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
        }
        }
    }
}

```

8. 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 class
UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME,
DATABASE_VERSION) {
    companion object { private const val
DATABASE_VERSION = 1
private const val DATABASE_NAME =
"UserDatabase.db" private const val TABLE_NAME = "user_ta
private const val COLUMN_ID = "id" private const
COLUMN_FIRST_NAME = "first_name" private const
COLUMN_LAST_NAME = "last_name" private const
COLUMN_EMAIL = "email" private const val COLUMN_PASSWO
= "password"
    }

    override fun onCreate(db: SQLiteDatabase?) { val createTabl
"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()
    }

    @SuppressWarnings("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
    }

```

```

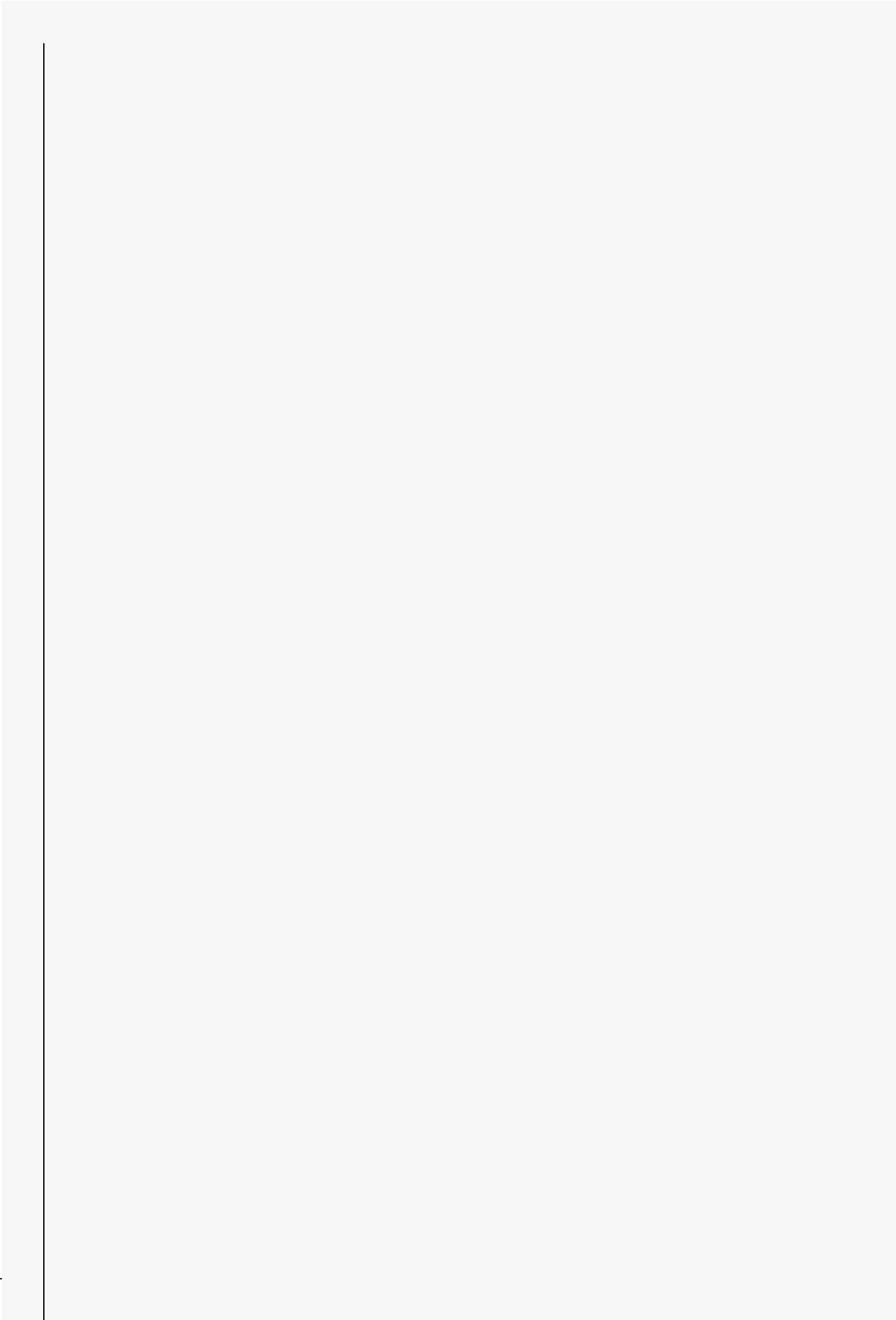
        @SuppressWarnings("Range")    fun getAllUsers(): List<User> {
val users = mutableListOf<User>()      val db = readableDatabase
val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME", null)
        if (cursor.moveToFirst()) {    do {
val user = User(
            id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            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
    }
}
```

9. 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.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
```



```

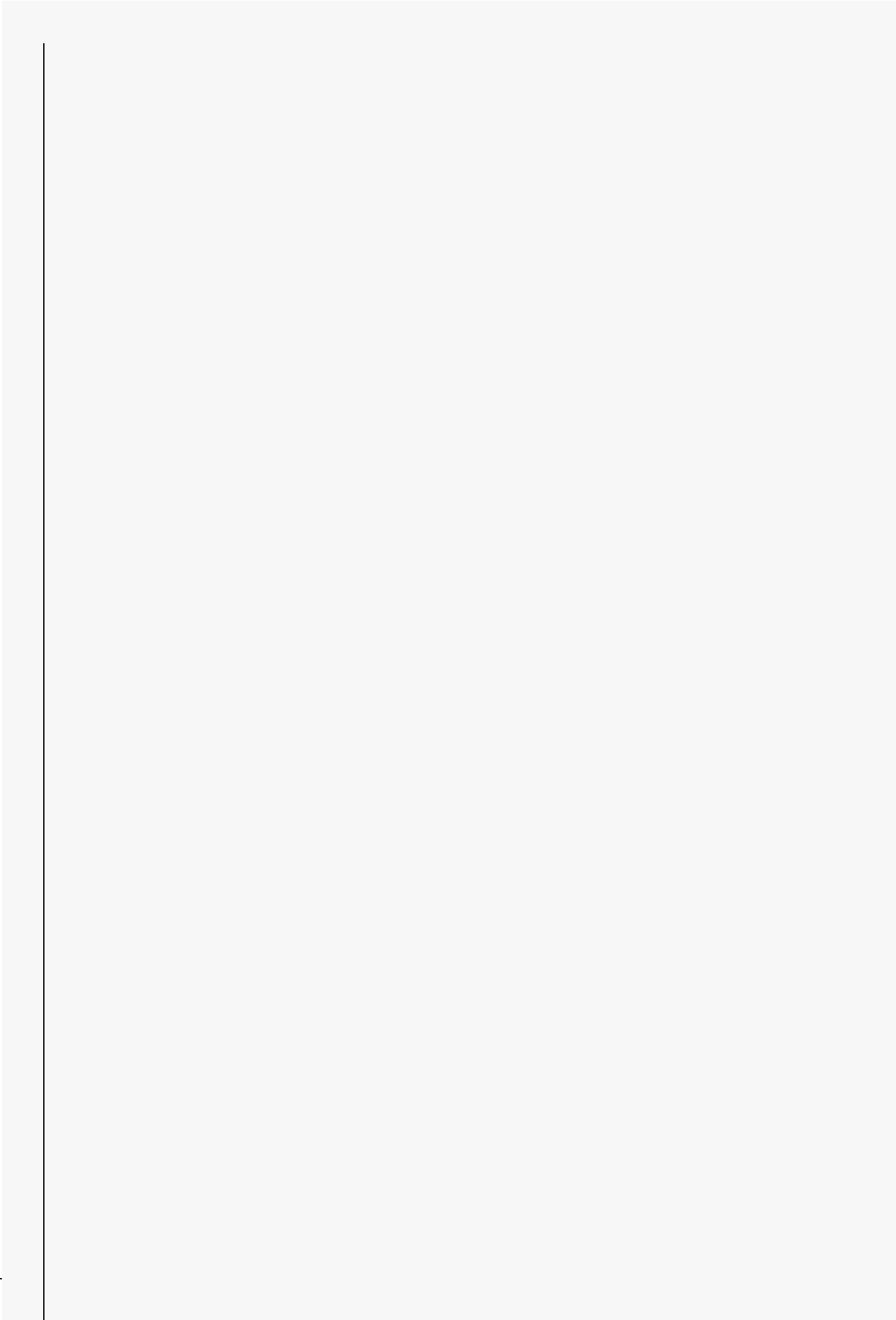
import com.example.sleeptracking.ui.theme.SleepTrackingTheme

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.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.sleep),    contentDescription = "",

        modifier = imageModifier
        .width(260.dp)

        .height(200.dp)
    )

```

```
        if (error.isEmpty()) {            Text(
text = error,                color =
MaterialTheme.colors.error,        modifier =
Modifier.padding(vertical = 16.dp)
        )
    }

    Button(        onClick = {            if (username.isEmpty()
&& password.isEmpty()) {                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

```

```
import androidx.compose.foundation.layout.*
import androidx.compose.material.* import
import androidx.compose.runtime.* import
import androidx.compose.ui.Alignment import
import androidx.compose.ui.Modifier import
import androidx.compose.ui.draw.alpha import
import androidx.compose.ui.graphics.Color import
import androidx.compose.ui.layout.ContentScale import
import androidx.compose.ui.res.painterResource import
import androidx.compose.ui.text.font.FontFamily import
import androidx.compose.ui.text.font.FontWeight import
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.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.sleep),        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
                    )
                    private fun startLoginActivity(context: Context) {
                        val intent = Intent(context, LoginActivity::class.java)
                        ContextCompat.startActivity(context, intent, null)
                    }
                }
            }
        )
    }
}

```

```

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

```

```
} )  
{  
Spacer(modifier = Modifier.width(10.dp))  
Text(text = "Log in")  
    }  
}  
}  
}
```

11.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.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)
}

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)
}
```

12 .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.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.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.colors.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.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )

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

```

```

Spacer(modifier = Modifier.height(30.dp))
LazyRow(
    modifier = Modifier
        .fillMaxSize()
        .padding(top = 56.dp),

    horizontalArrangement = Arrangement.SpaceBetween
){
    item {

        LazyColumn {
            items(timeLogs) { timeLog ->
                Column(modifier = Modifier.padding(16.dp)) {
                    //Text("ID: ${timeLog.id}")
                    Text("Start time:
${formatDateTime(timeLog.startTime)}")
                    Text("End time: ${timeLog.endTime?.let {
formatDateTime(it) }}")
                }
            }
        }
    }
}

```

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

13 .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/sleepicon"  
        android:label="@string/app_name"  
        android:supportRtl="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" />
            </category>
        </intent-filter>
    </activity>
</application></manifest>
```

