A Sleep Tracking App for a Better Night's Rest

SRINIVAS P.S

HARISHBABU K

ABIMANYU S

SUSEENDIRAN G.S.

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.

Project workflow

- Users register into the application.

- After registration , user logins into the application.
- User enters into the main page
- User can track the sleep timing and he record the time

1.2. Purpose

- Sleep tracking apps use smartphones' built-in accelerometers to record and interpret sleep data each night.
- These apps commonly track movements during sleep, record sound, wake sleepers up during light stages of their sleep cycle, and provide insights to help you interpret the data

- Sleep apps monitor your sleep and provide you with an easy-to-read analysis of how well you slept. They are useful for people who want to identify how many times they wake up during the night and why, and want to set goals to improve their sleep.

2. Problem definition & design thinking

2.1 Empathy Map

Template

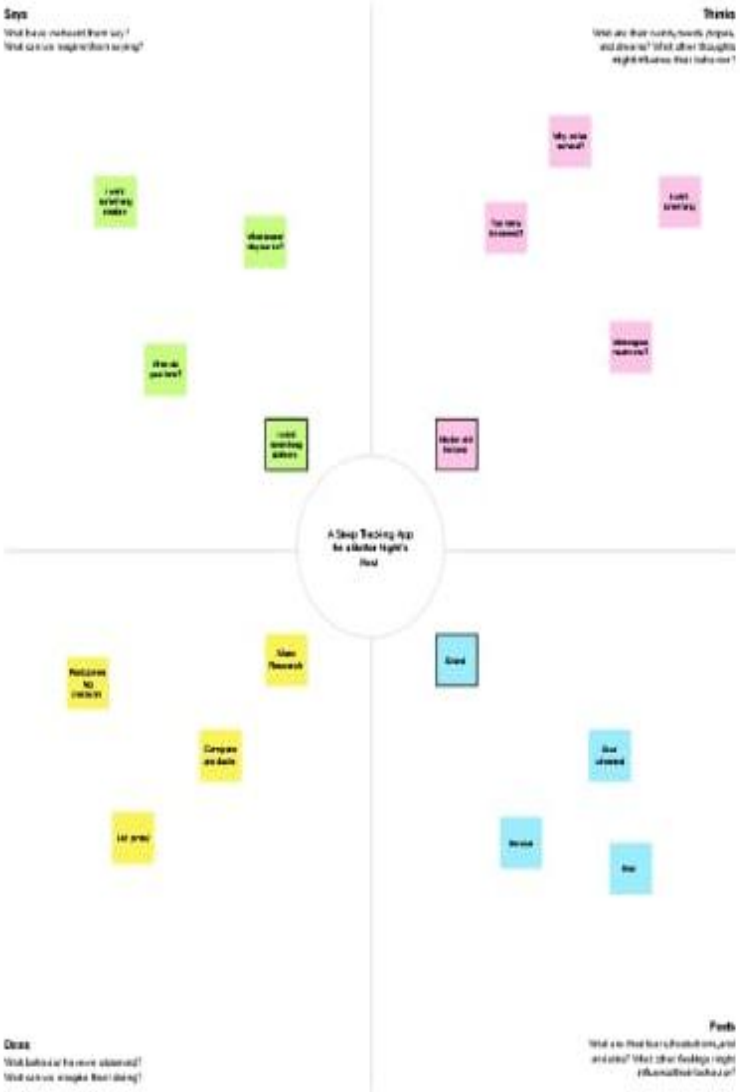
Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share to replete to setback

Build empathy

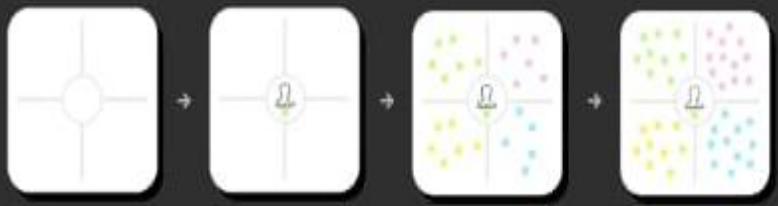
The information you add here should be representative of the observations and research you've done about your users.



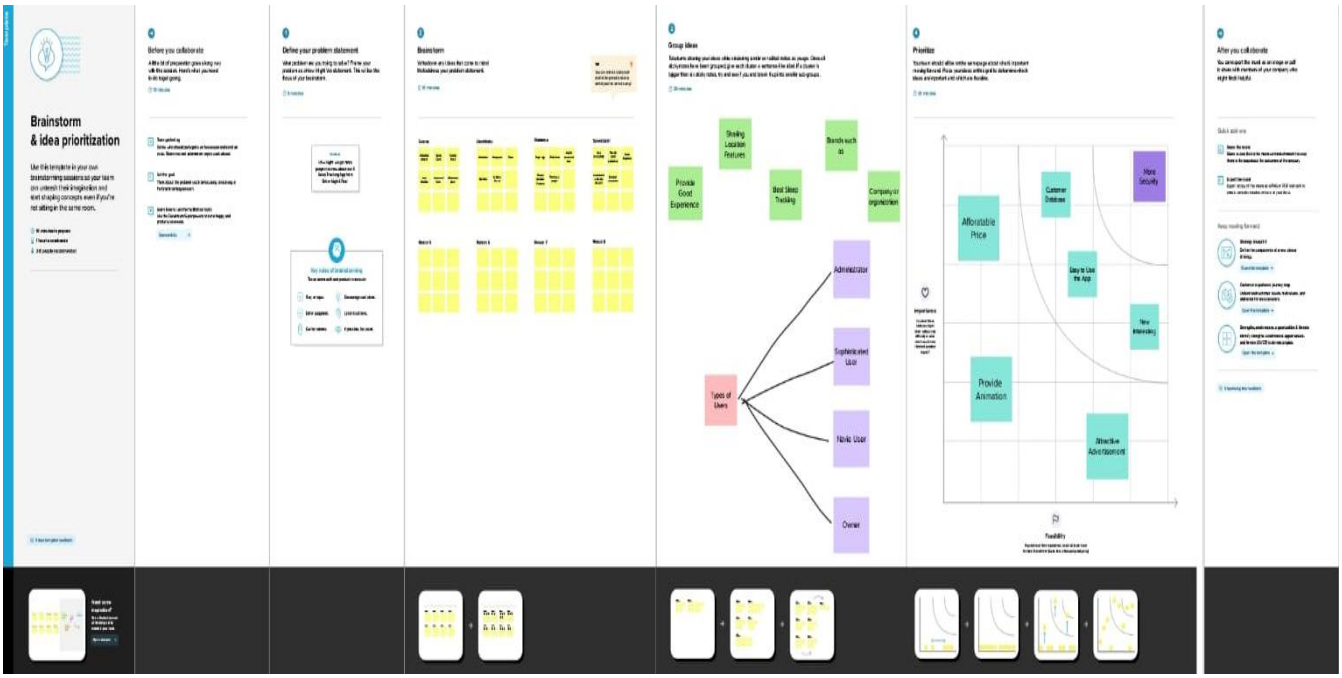
Need some inspiration?

See a finished version of this template to kickstart your work.

Open example

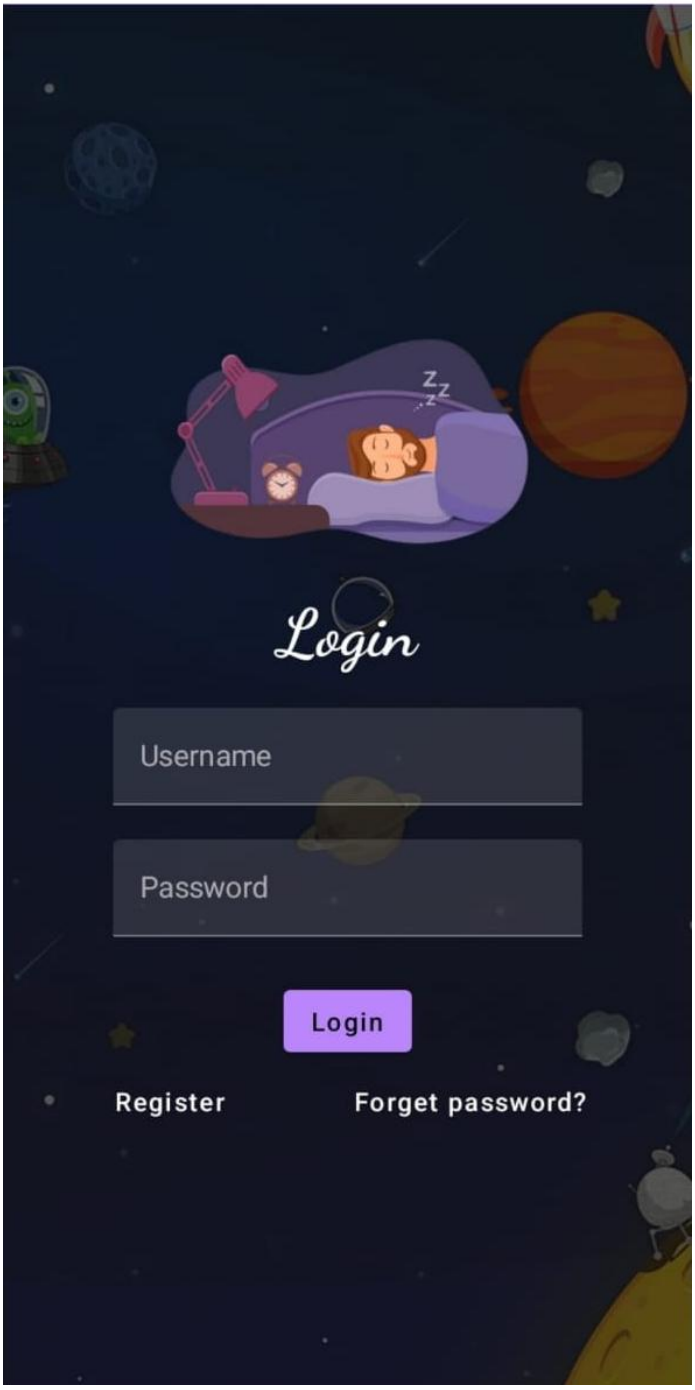


2.2 ideation & Brainstroming

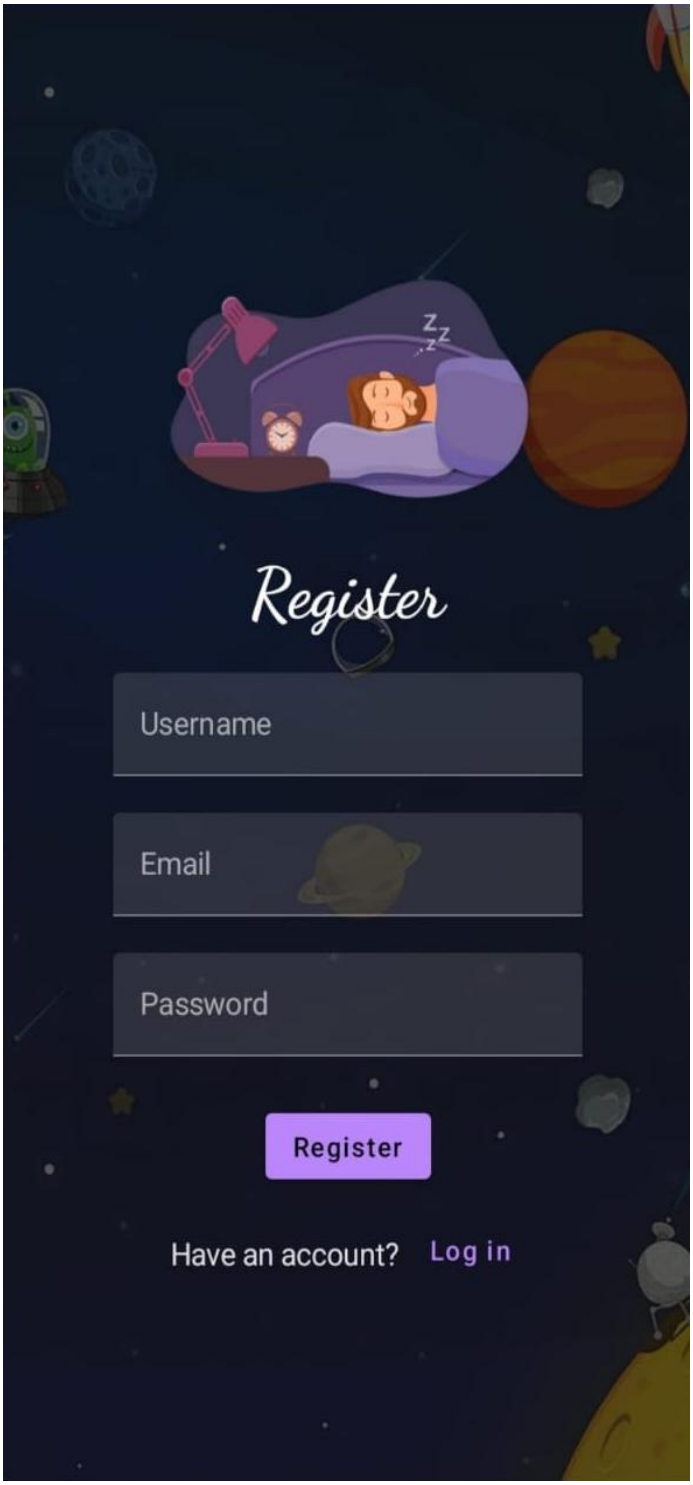


1. RESULT

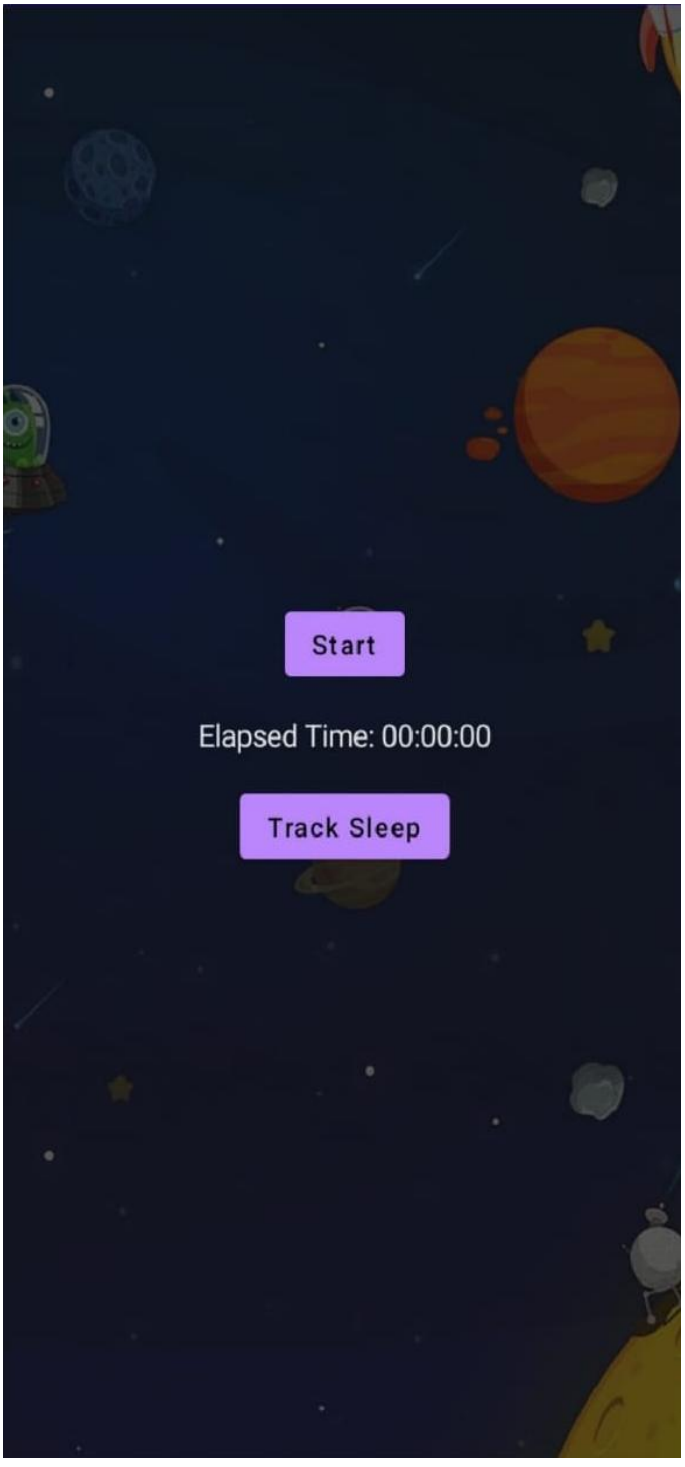
Login Page:



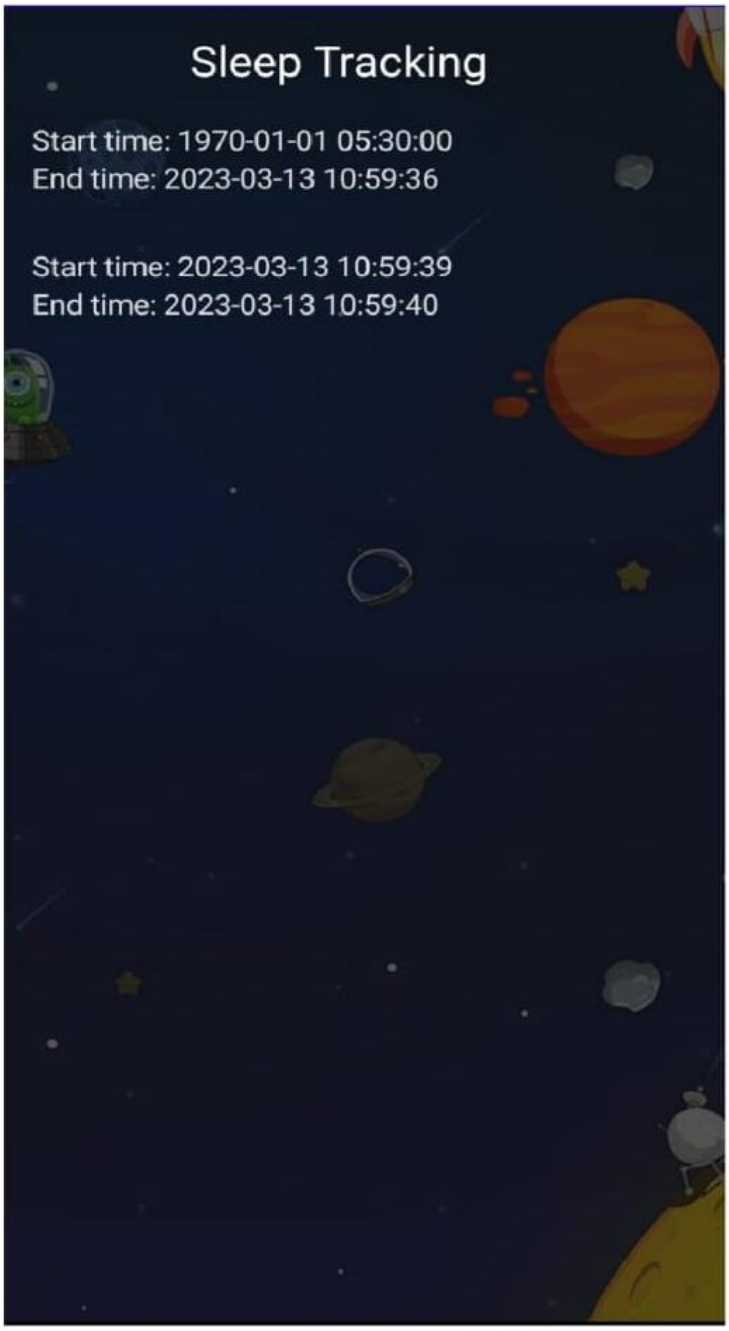
RegisterPage:


Register
Username
Email
Password
Register
Have an account?
Log in

Main Page


Start
Elapsed Time: 00:00:00
Track Sleep

Track Sleep Page:



2. ADVANTAGES & DISADVANTAGES

Advantages:

- Sleep is mysterious. But sleep-tracking apps promise to help you understand when you cross the threshold between waking and sleeping and what happens in between.
- For people who want an uncomplicated interface with an intuitive design, we recommend sleep score (which works a lot better with iPhones than with Android phones) as well as sleep cycle (which is as compatible with Android models as it is with iPhones).
- After more than a hundred hours of research, including interviews with eight sleep scientists, and more than a month spent testing four popular sleep-tracking apps .
- we found that no app offers objectively accurate sleep analysis, and they can’t replicate the experience of a traditional sleep lab. But you can use them to glean trends and patterns, which may help you improve your sleep over time.

Disadvantage:

- The Sleep Cycle app does have some drawbacks. Battery drain: In order to function, the program must run overnight and this will slowly drain your battery. Although it will no doubt vary with your battery's age, it drained mine by 20% over 8 hours.
- According to a recent scientific study, tracking sleep with apps, wearables or in-bed sleep monitors can be directly linked to causing bedtime anxiety and stress... which are hardly the ideal recipe for achieving a sound night's sleep.
- Whether you get your evening Netflix fix on your phone, tablet or TV, all of these devices have the same inherent problem: blue light exposure. All of these devices emit blue light that your body perceives as daylight.
- Repeated use of white noise could be harmful to the body, especially for those who suffer from tinnitus, or ringing in the ears.

5. APPLICATION:

Application of sleep tracking for a better night rest

Sleep tracking can be a useful tool for improving the quality of your sleep. Here are some applications of sleep tracking that can help you get a better night's rest:

Identifying sleep patterns: Sleep tracking devices can monitor your sleep patterns, including how long it takes you to fall asleep, how long you stay asleep, and how much time you spend in each stage of sleep. This information can help you identify any patterns or habits that may be affecting your sleep quality.

Monitoring sleep quality: Some sleep tracking devices can also monitor your heart rate, breathing rate, and other physiological factors that can affect your sleep quality. By tracking these factors, you can identify any changes in your sleep quality over time and take steps to improve it.

Setting sleep goals: Many sleep tracking devices allow you to set goals for the amount and quality of sleep you want to achieve each night. By setting specific goals, you can track your progress and make adjustments to your sleep habits as needed.

Tracking sleep hygiene: Sleep tracking devices can also monitor your sleep hygiene habits, such as your caffeine and alcohol intake, exercise routine, and bedtime routine. By tracking these habits, you can identify any changes that may be affecting your sleep quality and make adjustments as needed.

Providing personalized recommendations: Some sleep tracking devices can provide personalized recommendations based on your sleep data. For example, if you're not getting enough deep sleep, the device may recommend changing your bedtime routine or reducing your caffeine intake to improve your sleep quality.

Overall, sleep tracking can be a valuable tool for improving your sleep quality and helping you achieve a better night's rest. However, it's important to remember that sleep tracking devices are not a substitute for medical advice, and if you're experiencing persistent sleep problems, it's important to talk to your doctor.

About Android Studio Application:

- Android Studio is the officialintegrated development environment (IDE)for building Android apps. It was first releasedby Google in

2013 and has since become themost popular development environment forAndroid app developers.

- Android Studio is based on theINTELIJ IDEA community edition, and itincludes many tools and features designedspecifically for developing Android apps

Code Editor:

- Android Studi includes a powerful code editor that provides syntax highlighting, code completion, and other features to help developers write clean, efficient code. The code editor alsosupports debugging and refactoring tools.

Emulator:

- Android Studio includes a built-in emulator that allows developers to test their apps on different Android devices without needing to own the actual devices. Theemulator supports a wide range of Android versions and device configurations.

Gradle Build System:

- Android Studio uses the Gradle build system, which makes it easy to manage dependencies and build complex apps with multiple modules

Version Control:

- Android Studio supports version control systems like Git, allowing developers to easily manage their code changes and collaborate with other team members.

Performance Profiling:

- Android Studio includes performance profiling tools that allow developers to identify performance bottlenecks in their apps and optimize their code for better performance. Overall, Android Studio is a powerful tool for developing high-quality Android apps. It provides a range of features and tools that make it easy for developers to build, test, and deploy their apps.

Conclusion:

- In conclusion, sleep tracking can be a useful tool for improving your sleep quality and helping you achieve a better night's rest. By monitoring your sleep patterns, tracking your sleep hygiene habits, and setting sleep goals, you can identify any factors that may be affecting your sleep quality and make adjustments as needed. Some sleep tracking

devices can also provide personalized recommendations to help you improve your sleep quality. However, it's important to remember that sleep tracking devices are not a substitute for medical advice, and if you're experiencing persistent sleep problems, it's important to talk to your doctor. Overall, sleep tracking can be a valuable tool for optimizing your sleep and improving your overall health and wellbeing.

Future scope:

- The future of sleep tracking for a better night's rest is promising, with continued advancements in technology and research. Here are some potential areas of future development:

Wearable technology:

- Sleep tracking devices are becoming increasingly integrated into wearable technology, such as smartwatches and fitness trackers. As these devices become more sophisticated, they may be able to provide even more detailed information about your

sleep quality and identify new ways to improve it.

Artificial intelligence:

- The use of artificial intelligence (AI) in sleep tracking may enable devices to provide more personalized recommendations based on individual sleep data. AI could help identify specific factors that are affecting sleep quality and suggest customized solutions based on an individual's habits and preferences.

Non-invasive tracking:

- While current sleep tracking devices often require sensors or electrodes to be attached to the body, future developments in non-invasive tracking technology may enable sleep tracking without physical contact. This couldlead to more comfortable and convenient sleep tracking for users.

Integration with other health data:

- As sleep is closely linked to overall health and wellbeing, future sleep tracking devices may integrate with other health data, such as activity levels, nutrition, and stress levels. This could provide a more holistic view of an

individual's health and help identify the root causes of sleep problems.

Treatment options:

- With a better understanding of individual sleep patterns and habits, future sleep tracking devices may be able to recommend specific treatment options, such as light therapy or cognitive behavioral therapy, to help improve sleep quality and overall health.

  In summary, the future of sleep tracking for a better night's rest is likely to be focused on providing more personalized and integrated solutions for individuals. By leveraging new technologies and data sources, sleep tracking devices may be able to provide even more accurate and actionable information to help users achieve optimal sleep quality and overall health.

APPENDIX

Creating the database classes:

User class code:

```
package com.example.projectone
```

```
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?,

    )
```

Create an user dao interface:

User Dao interface:

```
package com.example.projectone

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 class:

```
package com.example.projectone

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
            }
        }
    }
}
```

Userdatabase Helper:

```
package com.example.projectone
```

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

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

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

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

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
```

```
            "$COLUMN_ID INTEGER PRIMARY KEY
AUTOINCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$COLUMN_LAST_NAME TEXT, " +
            "$COLUMN_EMAIL TEXT, " +
            "$COLUMN_PASSWORD TEXT" +
            ")"

        db?.execSQL(createTable)
    }

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

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

```
    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_FIRST_NAME = ?",
arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NA
ME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWO
RD)),
            )
        }
        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_NA
ME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWO
RD)),
            )
        }
        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_NA
ME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWO
RD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
```

```
        db.close()
        return users
    }

}
```

Database 2:

TimeLog data class:

```
package com.example.projectone

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
)
```

TimeLog Dao interface:

```
package com.example.projectone

import androidx.room.Dao
import androidx.room.Insert

@Dao
interface TimeLogDao {
    @Insert
    suspend fun insert(timeLog: TimeLog)

}
```

AppDatabase class:

```
package com.example.projectone

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
            }
        }
    }
}
```

TimeDatabase class:

```
package com.example.projectone
```

```
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_TI
ME))
            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"
        }
    }
}
```

LoginActivity.kt

```
package com.example.projectone

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.projectone.ui.theme.ProjectOneTheme


class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ProjectOneTheme {
                // 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)
        )
```

```
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,
                MainActivity2::class.java
            )
        )}
        )
        { Text(color = Color.White,text = "Sign up") }
        TextButton(onClick = {
            /*startActivity(
            Intent(
                applicationContext,
                MainActivity2::class.java
            )
        )*/
        })

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

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

RegistrationActivity.kt

```
package com.example.projectone

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.projectone.ui.theme.ProjectOneTheme


class MainActivity2 : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            ProjectOneTheme {
                // 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
                )
                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 = {

            })

            {
                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)
}
```

MainActivity.kt

```
package com.example.projectone

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.projectone.ui.theme.ProjectOneTheme
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 {
            ProjectOneTheme {
```

```
            // 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)
}
```

TrackActivity.kt

```
package com.example.projectone

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.projectone.ui.theme.ProjectOneTheme
import java.util.*

class TrackActivity : ComponentActivity() {

    private lateinit var databaseHelper:
TimeLogDatabaseHelper

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

        databaseHelper = TimeLogDatabaseHelper(this)
        setContent {
            ProjectOneTheme {
                // 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))
}
```

Androidmanifest.xml

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

   <application
      android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
      android:fullBackupContent="@xml/backup_rules"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/Theme.ProjectOne"
      tools:targetApi="31">
      <activity
         android:name=".TrackActivity"
         android:exported="false"
         android:label="@string/title_activity_track"
         android:theme="@style/Theme.ProjectOne" />
      <activity
         android:name=".MainActivity"
         android:exported="false"
         android:label="@string/app_name"
         android:theme="@style/Theme.ProjectOne" />
```

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

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

</manifest>
```