PROJECT REPORT

A Sleep Tracking App for A Better Night's Rest

1. INTRODUCTION

1.1 Overview

The project 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 Tracking” app, we 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.

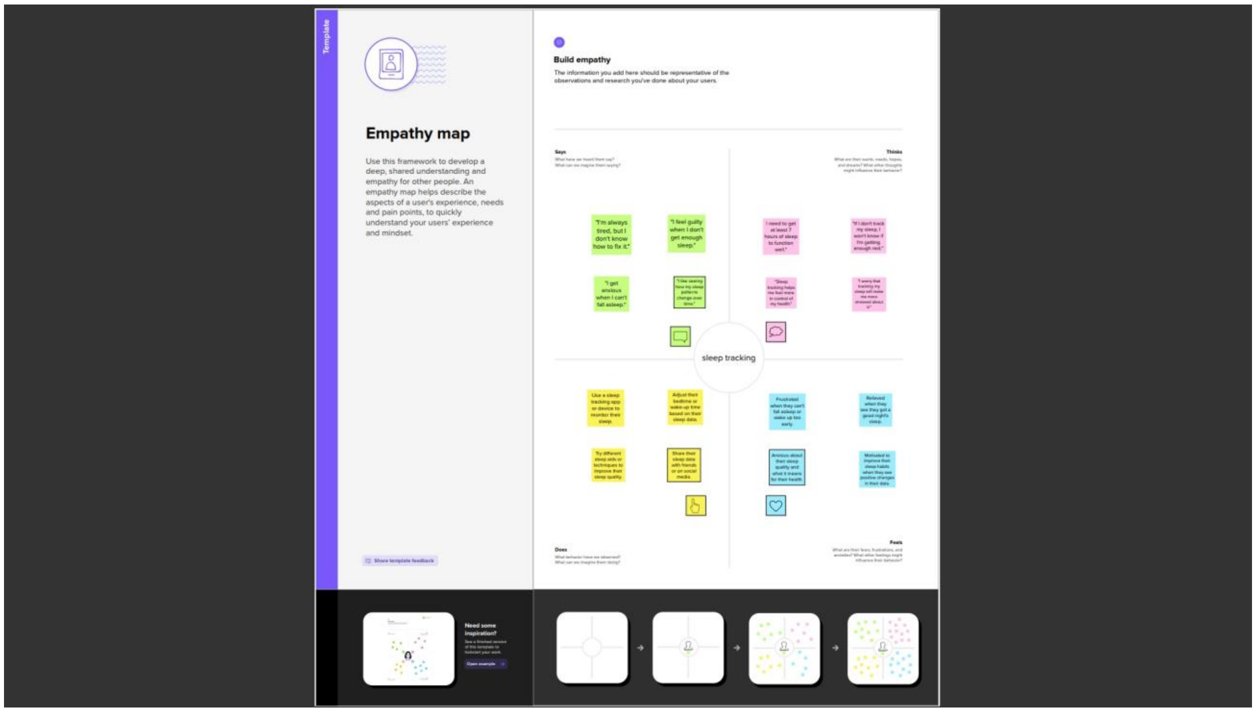
“Sleep Tracking” App enables us to start the timer when they are in the bed and about to fall asleep. The timer will keep running in the background until it is stopped, whenever the user wakes up.

1.2 Purpose

The main purpose of the project to track the uses’ sleep and assess the quality of the sleep of that night. It uses the timer for sleep tracking. The quality of the sleep is rated based on the sleep experience. The App will provide a sleep analysis of previous night.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map



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.

Build empathy

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

Says

What have we heard them say? What can we imagine them saying?

"I'm always tired, but I don't know how to fix it."

"I feel guilty when I don't get enough sleep."

"I get anxious when I can't fall asleep."

"I like seeing how my sleep patterns change over time."

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

I need to get at least 7 hours of sleep to function well."

"If I don't track my sleep, I won't know if I'm getting enough rest."

"Sleep tracking helps me feel more in control of my health."

"I worry that tracking my sleep will make me more stressed about it."

sleep tracking

Does

What behavior have we observed? What can we imagine them doing?

Use a sleep tracking app or device to monitor their sleep.

Adjust their bedtime or wake-up time based on their sleep data.

Try different sleep aids or techniques to improve their sleep quality.

Share their sleep data with friends or on social media.

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

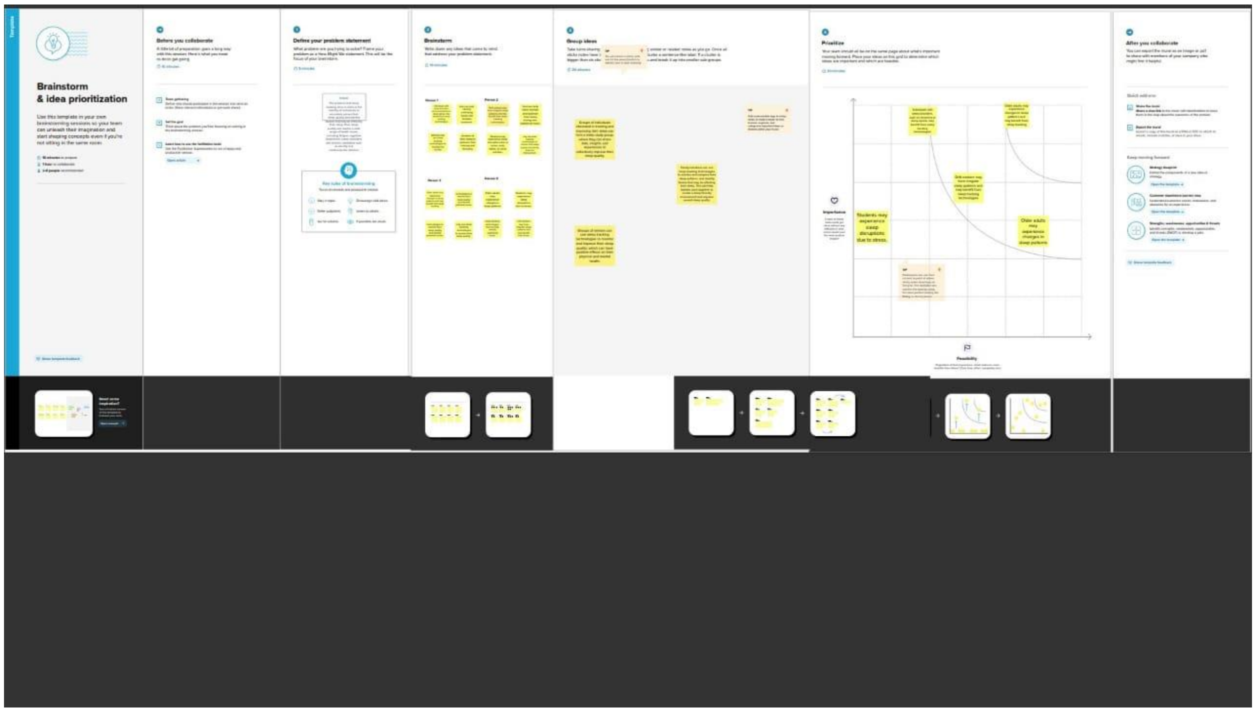
Frustrated when they can't fall asleep or wake up too early.

Relieved when they see they got a good night's sleep.

Anxious about their sleep quality and what it means for their health.

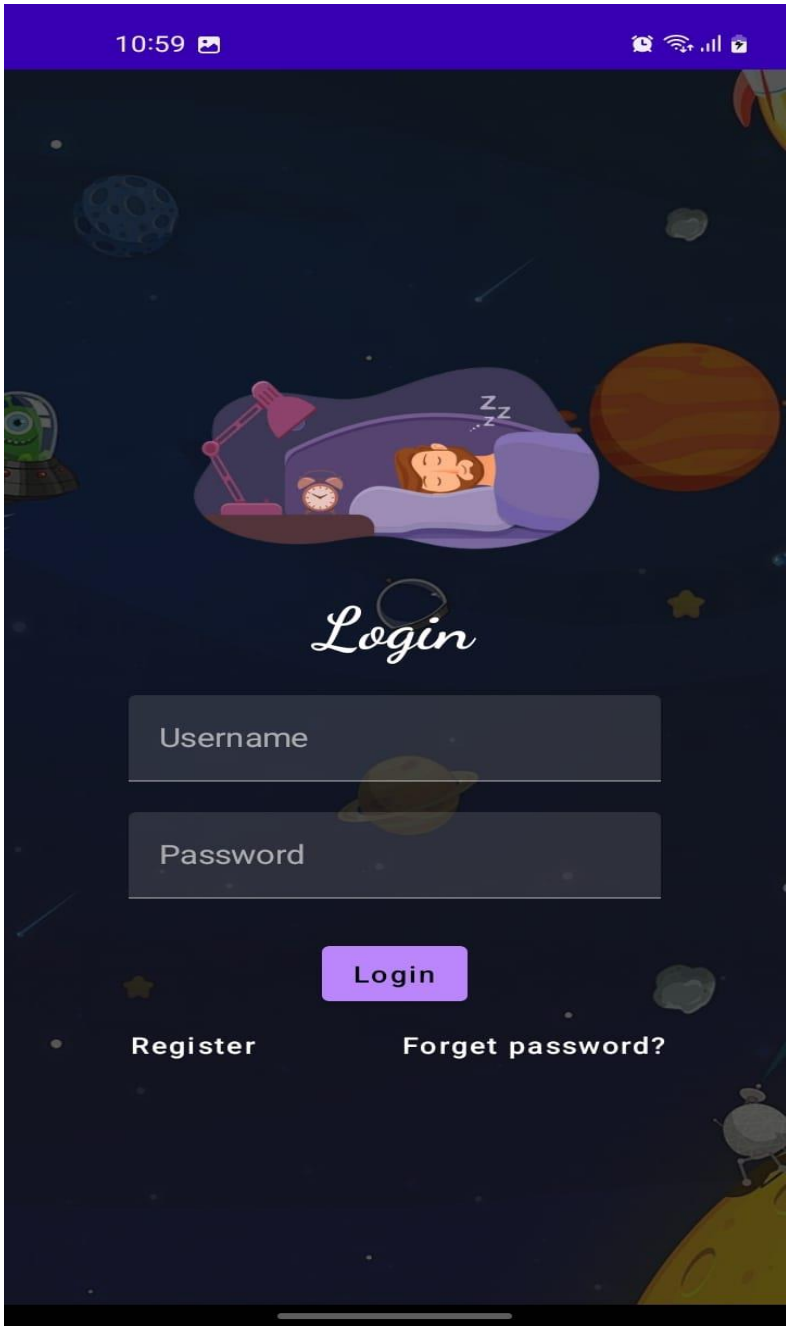
Motivated to improve their sleep habits when they see positive changes in their data.

2.2 Ideation & Brainstorming Map

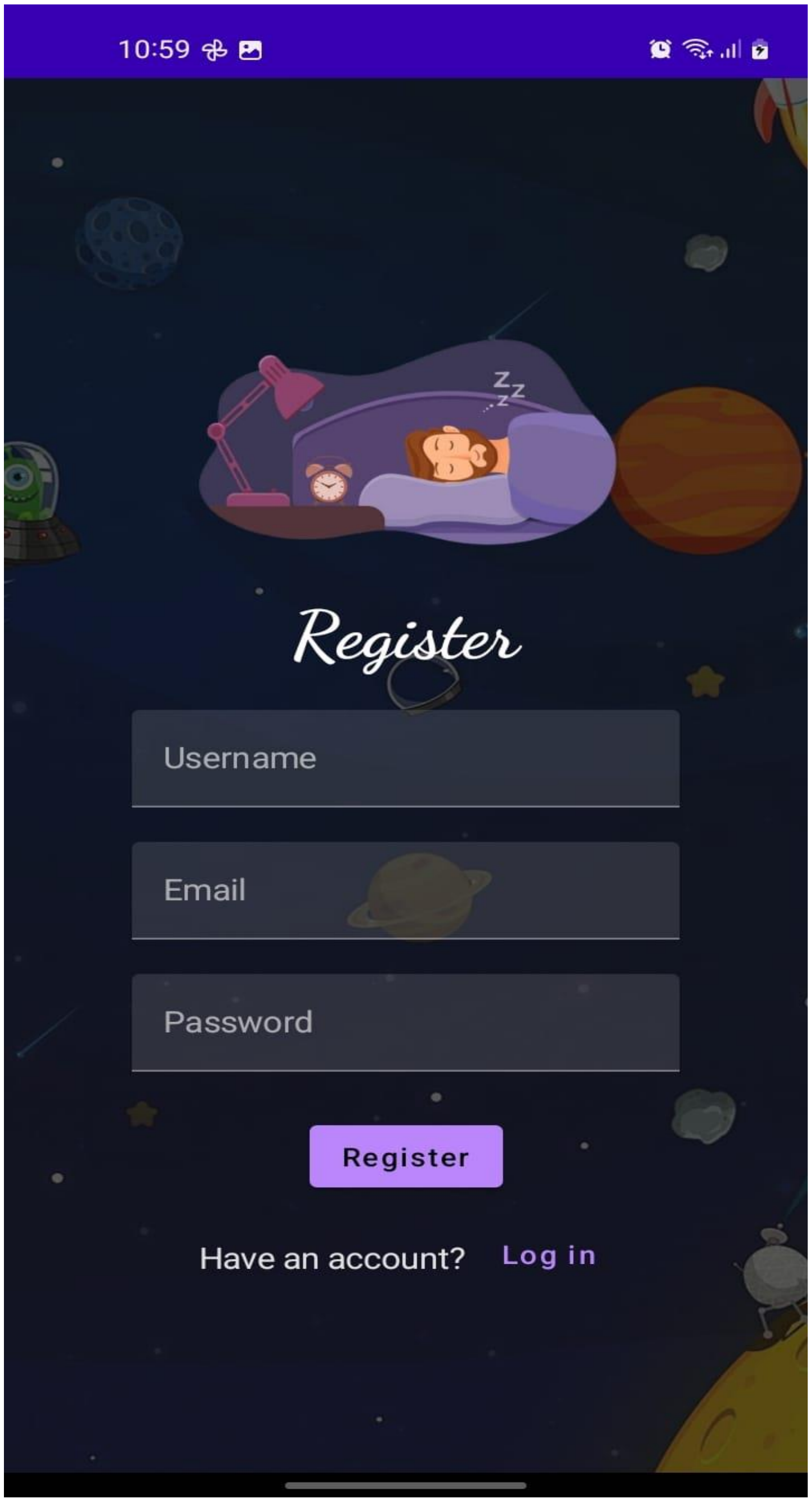

Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
Importance
Feasibility
Students may experience sleep disruption due to stress.
Older adults may experience changes in sleep patterns
After you collaborate

3. RESULT

Login Page

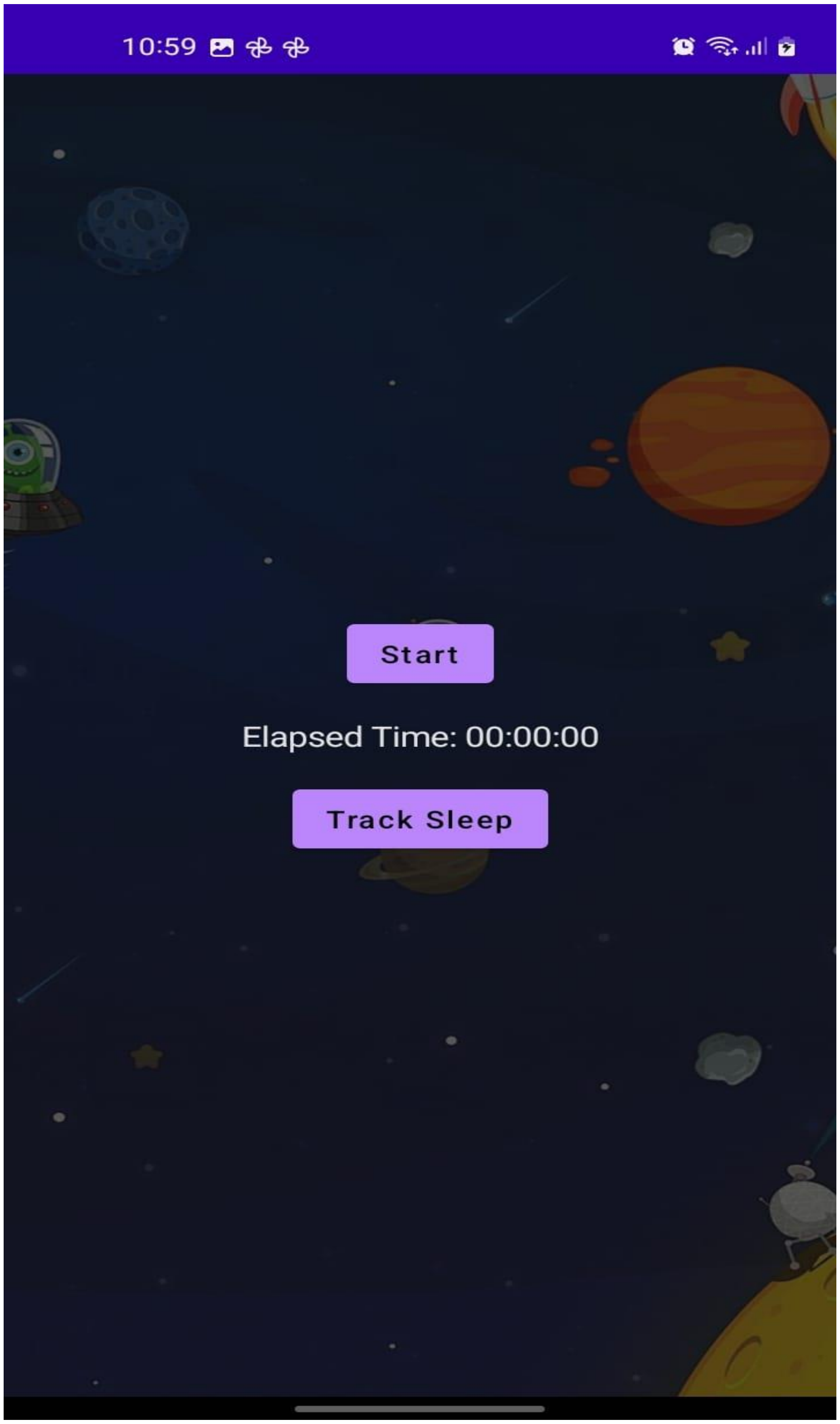


Register Page

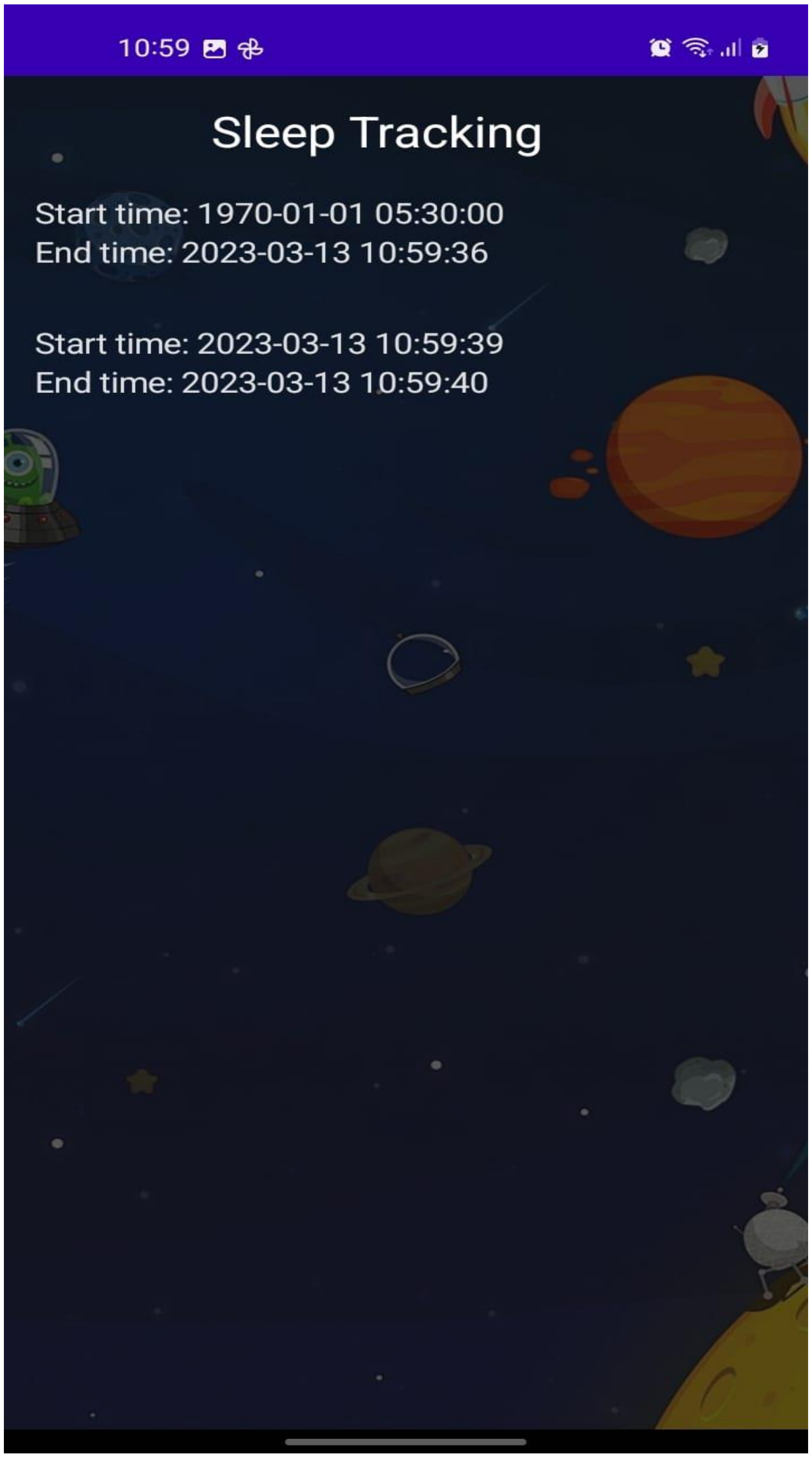

10:59
Register
Username
Email
Password
Register
Have an account?
Log in

Main Page

Track Sleep Page



Track Sleep Page



Sleep Tracking

Start time: 1970-01-01 05:30:00
End time: 2023-03-13 10:59:36

Start time: 2023-03-13 10:59:39
End time: 2023-03-13 10:59:40

4. ADVANTAGES & DISADVANTAGES

4.1 Advantages

1) Easy to use
2) Secured
3) Tracking of sleep
4) Providing analysis report

4.2 Disadvantages

1) Manual start
2) Sleep during only displayed
3) No detail about quality of sleep

5. APPLICATIONS

“Sleep Tracking” App is used to track the sleep time of the user and provide an analysis report. The App uses the timer control that is running in the background until it is stopped when the user wakes up. The app will help rate sleep quality. This application is useful for each person to the sleep of previous night.

6. CONCLUSION

We can assess the quality of sleep they have had in a day using the App. A good quality sleep is a must for effective functioning of both mind and body. The “Sleep Tracking” App uses a timer to track the sleep of the individual. The timer will continue running in the background. The app also produces the analysis of the kind of sleep during previous night.

7. FUTURE SCOPE

The present App tracks only the sleeping time. It can be enhanced by adding more features. New features include the finding the quality of sleep. The App may be altered to start automatically when the use fall asleep and stops when he wakes up.

8. APPENDIX

```
    // User.kt

    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?,

        )

// UserDao.kt

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.kt

```
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
            }
        }
    }
}
```

// UserDatabaseHelpper.kt

```
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_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
```

```kotlin
    }
}
// TimeLog.kt
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
)
// TimeLogDao.kt
package com.example.projectone

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

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

}
// AppDatabase.kt
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
            }
        }
    }
}
```

// TimeDatabaseHelper.kt

```
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_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"
        }
    }
}
```

// 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)
}
```

// RegisterActivity.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))
}
```