

# **A SLEEP TRACKING APP FOR BETTER NIGHT'S REST**

**PROJECT PRESENTED BY:**

**CATEGORY: ANDROID APPLICATION  
DEVELOPMENT**

**TEAM ID: NM2023TMID09272**

**TEAM LEADER: AKASH M**

**TEAM MEMBERS:**

- **ABINASH B**
- **ARUN R**
- **ARUNPRAKASH K**
- **SUBASH S**

# 1)INTRODUCTION

## 1.1Over view

Our sleep tracking app is designed to help you monitor and improve your sleep quality by providing you with insights and recommendations based on your sleep data. Here are some key features of our app:

1. Sleep tracking: Our app uses your smartphone's sensors to track your sleep duration, quality, and consistency. It provides you with a detailed analysis of your sleep patterns, including the time it takes for you to fall asleep, the number of times you wake up during the night, and the duration of each sleep stage.
2. Sleep score: Our app calculates a sleep score based on your sleep data, which provides an overall assessment of your sleep quality. The sleep score takes into account various factors such as the duration of each sleep stage, sleep disturbances, and your sleep goals.
3. Personalized insights and recommendations: Our app provides personalized insights and recommendations based on your sleep data. It helps you identify patterns and make adjustments to improve your sleep habits, such as going to bed earlier, reducing caffeine intake, or creating a more comfortable sleep environment.
4. Sleep goal setting: Our app allows you to set sleep goals based on your personal preferences and lifestyle. You can choose to focus on improving your sleep duration, quality, or consistency, and track your progress over time.
5. Sleep diary: Our app includes a sleep diary where you can record any factors that may affect your sleep, such as stress, diet, or exercise. This helps you identify patterns and make adjustments to improve your sleep habits.

## 1.2 Purpose

Some specific purposes of a sleep tracking app are:

1. To help users identify sleep patterns: By monitoring sleep data, the app can help users identify patterns in their sleep habits. For example, if the app shows that the user has trouble falling asleep, they may need to adjust their bedtime routine or environment to promote relaxation.
2. To promote healthy sleep habits: The app can provide users with recommendations to promote healthy sleep habits, such as setting a regular sleep schedule, avoiding caffeine and alcohol before bedtime, or creating a comfortable sleep environment.
3. To improve overall health and well-being: Sleep is essential for physical and mental health, and a sleep tracking app can help users optimize their sleep habits to improve their overall health and well-being.
4. To track progress: The app can track progress over time, allowing users to see how their sleep habits are improving and where they need to make further adjustments.

Overall, the purpose of a sleep tracking app is to help users achieve better sleep quality and improve their overall health and well-being.

## 2) PROBLEM DEFINITION & DESIGN THINKING

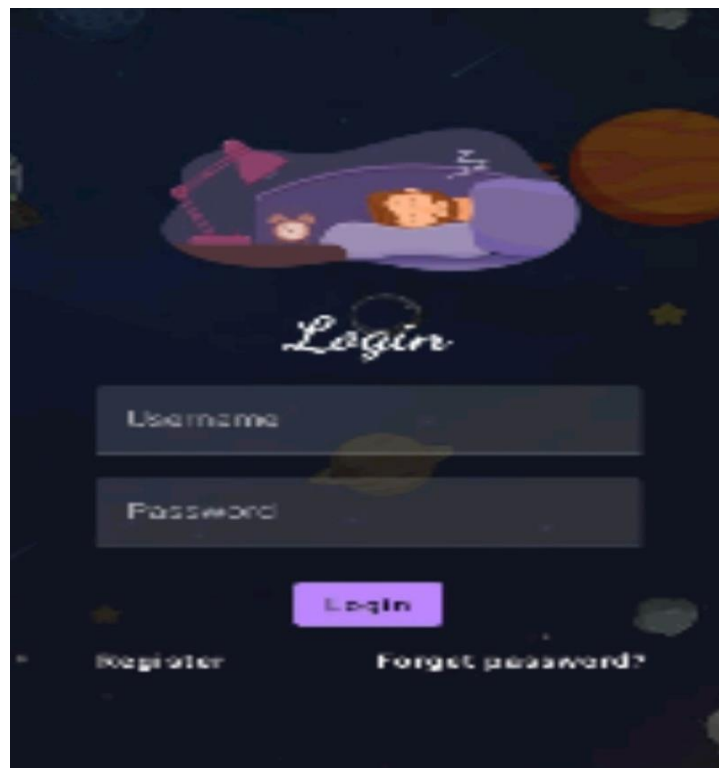
### 2.1 Emphathy Map



## 2.2 Ideation & Brainstorming map



### 3) RESULT



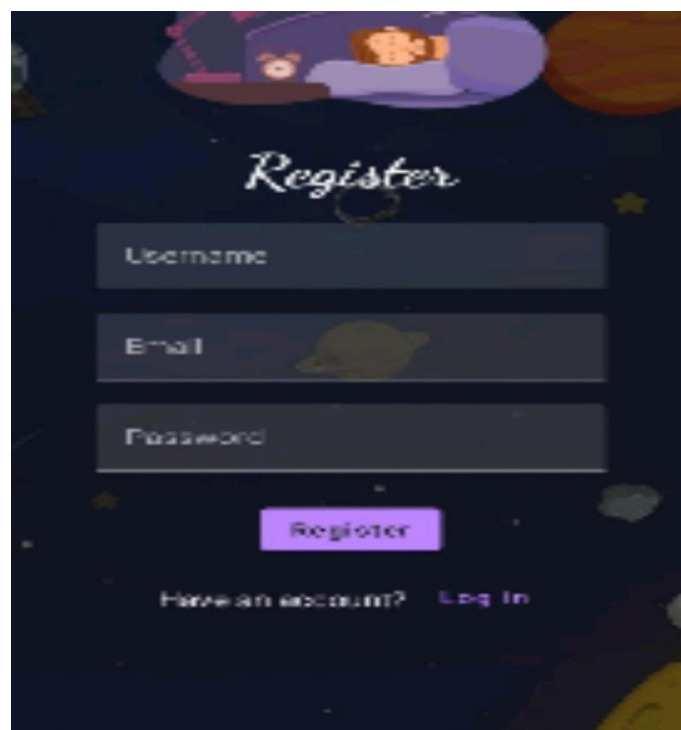
*Login*

Username

Password

[Login](#)

[Register](#) [Forget password?](#)



*Register*

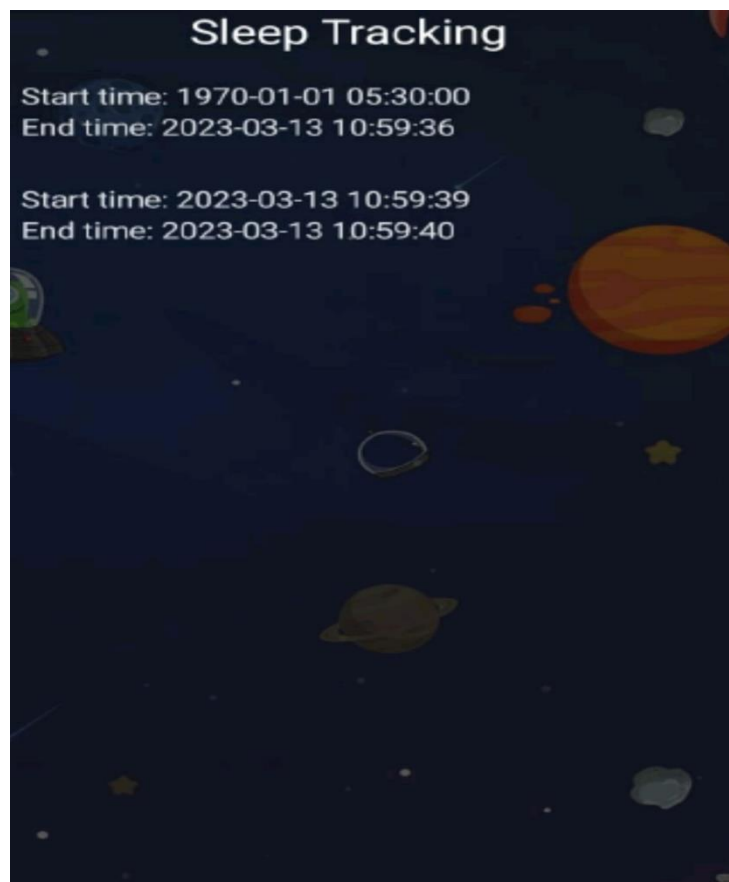
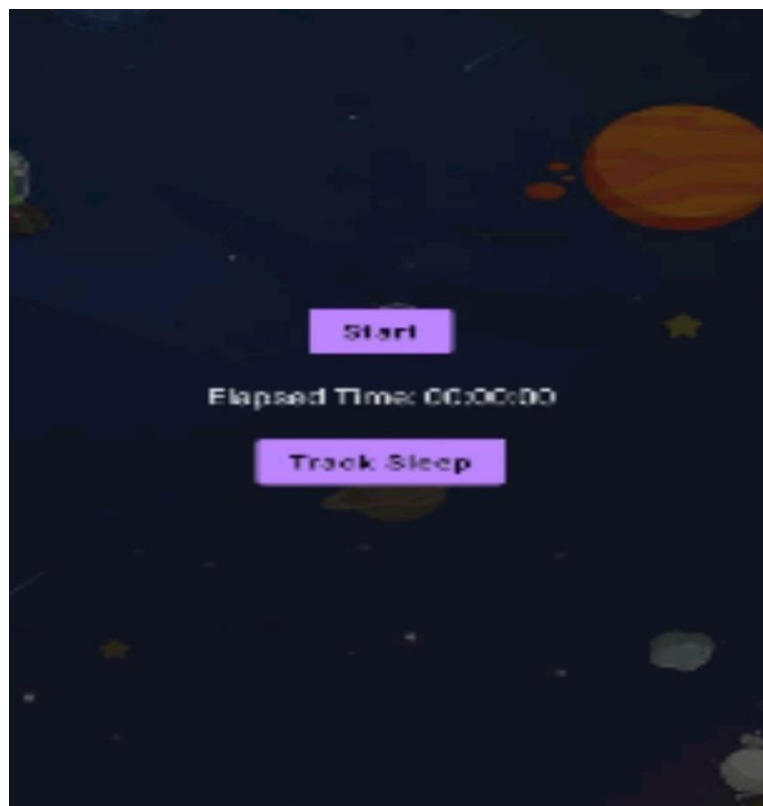
Username

Email

Password

[Register](#)

[Have an account? Log In](#)



#### 4) ADVANTAGES :

There are several advantages of using a sleep tracking app, including:

1. **Better understanding of sleep patterns:** A sleep tracking app provides users with insights into their sleep patterns, including sleep duration, quality, and consistency. By tracking these metrics over time, users can gain a better understanding of their sleep habits and make adjustments to improve their sleep quality.
2. **Improved sleep quality:** By monitoring sleep patterns and providing recommendations, a sleep tracking app can help users improve their sleep quality. Users can identify factors that are affecting their sleep and make adjustments to improve their sleep environment and habits.
3. **Increased awareness of sleep hygiene:** A sleep tracking app can help users become more aware of the importance of sleep hygiene, such as setting a regular sleep schedule, avoiding caffeine and alcohol before bedtime, and creating a comfortable sleep environment.

#### DISADVANTAGES:

While sleep tracking apps offer several benefits, there are also some potential disadvantages to using them. These include:

1. **Inaccuracy:** Sleep tracking apps rely on sensors in smartphones or wearables to track sleep data. While these sensors have improved in recent years, they may not always provide accurate data. For example, a sleep tracking app may not be able to distinguish between light and deep sleep stages accurately.
2. **User error:** Sleep tracking apps require users to wear a device or keep their phone close to them while sleeping. If the user forgets to wear the device or keep the phone close to them, the app may not be able to track their sleep data accurately.
3. **Privacy concerns:** Sleep tracking apps collect sensitive data, such as sleep patterns, and may share this data with third-party services. Users may be uncomfortable with sharing this data and may have concerns about their privacy.



## 5) APPLICATIONS:

A sleep tracking app can be useful for several purposes, including:

1. **Monitoring sleep patterns:** A sleep tracking app can monitor your sleep patterns and provide you with information on how long you slept, how many times you woke up, and the quality of your sleep. This information can help you identify patterns and make changes to improve the quality of your sleep.
2. **Identifying sleep disorders:** A sleep tracking app can help you identify sleep disorders such as sleep apnea, insomnia, and restless leg syndrome. By monitoring your sleep patterns, the app can detect irregularities that may indicate a sleep disorder, allowing you to seek medical attention.
3. **Improving sleep hygiene:** A sleep tracking app can help you develop good sleep habits by providing information on factors that affect sleep quality, such as caffeine intake, exercise, and screen time before bed. The app can also suggest strategies to improve your sleep hygiene, such as establishing a regular sleep schedule and creating a relaxing bedtime routine.
4. **Increasing productivity:** A good night's sleep is essential for productivity, and a sleep tracking app can help you optimize your sleep to improve your work performance. By monitoring your sleep patterns, the app can provide suggestions on how to get better sleep and help you wake up feeling refreshed and ready to take on the day.
5. **Managing stress:** Stress can interfere with sleep, and a sleep tracking app can help you manage stress by providing relaxation techniques and other strategies to help you unwind before bed. By tracking your sleep patterns, the app can also help you identify how stress affects your sleep and suggest ways to manage it.

Overall, a sleep tracking app can be a useful tool for anyone looking to improve their sleep quality, manage sleep-related health conditions, and enhance their overall well-being.

## 6) CONCLUSION

- In conclusion, sleep tracking apps can be a valuable tool for individuals looking to improve their sleep habits and overall health and well-being. By providing insights into

sleep patterns and recommendations for improving sleep quality, these apps can help users make adjustments to their sleep environment and habits. However, it is important to weigh the potential disadvantages, such as inaccuracy and privacy concerns, before deciding to use a sleep tracking app. Additionally, sleep tracking apps can be applied in various areas, including personal health and wellness, healthcare, sports performance, workplace productivity, and education.

## **7) FUTURE SCOPE:**

The future scope of sleep tracking apps is vast, and we can expect to see several advancements and improvements in the coming years. Some of the potential future developments include:

1. **Enhanced accuracy:** Sleep tracking apps may use more advanced sensors or algorithms to provide more accurate data on sleep patterns. This may involve integrating with other devices such as smart mattresses or pillows that can monitor sleep data more precisely.
2. **Artificial intelligence:** AI-powered sleep tracking apps could use machine learning algorithms to provide more personalized recommendations for improving sleep quality. These apps could learn from user data over time to provide more accurate and relevant insights.
3. **Wearable technology:** Sleep tracking apps may continue to evolve to incorporate wearable technology such as smartwatches or smart bands that can monitor sleep data more accurately and provide real-time feedback on sleep quality.

## 8) APPENDIX

### A.Source Code

#### 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="@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>
```

#### UI Theme

##### Type.kt

```
package com.example.projectone.ui.theme
```

```

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)

```

## Color.kt

```

package com.example.projectone.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)

```

## Shape.kt

```

package com.example.projectone.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(4.dp),
    large = RoundedCornerShape(0.dp)
)

```

## Theme.kt

```
package com.example.projectone.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200

    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
)

@Composable
fun ProjectOneTheme(darkTheme: Boolean = isSystemInDarkTheme(), content:
@Composable () -> Unit) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

## 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
            }
        }
    }
}

```

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

```

```

)

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

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

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

                //onLoginSuccess()
            } else {
                error = "Invalid username or password"
            }
        } else {
            error = "Please fill all fields"
        }
    },
    modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Login")
}

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

```





```

        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)
}

```

## 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,
            onChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier
```

```

        .padding(10.dp)
        .width(280.dp)

    )

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

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

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

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

```

## 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
    @SuppressWarnings("Range")
    fun getTimeLogs(): List<TimeLog> {
        val timeLogs = mutableListOf<TimeLog>()
        val cursor = readableDatabase.rawQuery("select * from $TABLE_NAME",
null)
        cursor.moveToFirst()
        while (!cursor.isAfterLast) {
            val id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_START_TIME))
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_END_TIME))
            timeLogs.add(TimeLog(id, startTime, endTime))
            cursor.moveToNext()
        }
        cursor.close()
        return timeLogs
    }

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

    fun getAllData(): Cursor? {
        val db = this.writableDatabase
        return db.rawQuery("select * from $TABLE_NAME", null)
    }

    data class TimeLog(val id: Int, val startTime: Long, val endTime: Long?)
{
    fun getFormattedStartTime(): String {
        return Date(startTime).toString()
    }

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

```

## TimeLog.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)
}
```

## 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 ->

```



```

        @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
            }
        }
    }
}

```

## UserDatabaseHelper

```

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
}
}

```

## ExampleInstrumentedTest.kt

```

package com.example.projectone

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation] (http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext =
            InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.projectone", appContext.packageName)
    }
}

```

## ExampleUnitTest.kt

```

package com.example.projectone

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine
 * (host).
 *
 * See [testing documentation] (http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}

```