

**A SLEEP TRACKING APP FOR A BETTER NIGHT REST**

# **1. INTRODUCTION**

## **1.1 OVERVIEW**

**A brief description about sleep tracking app:**

Sleep apps monitor your sleep and provide you with an easy-to-read analysis of how well you slept. Increasing efficiency and overall quality of life is a key factor in creating a successful app.

It's what users demand, especially in this COVID-19 era we're all living in.

Sleep, in particular, offers a huge opportunity in the mobile app industry.

Between work, family duties, and an ongoing pandemic, 1 in 3 American adults don't get the sleep they need, according to the CDC. As a result, sleep depth and tracking apps are becoming more innovative and technologically advanced than ever before.

## **1.2 PURPOSE**

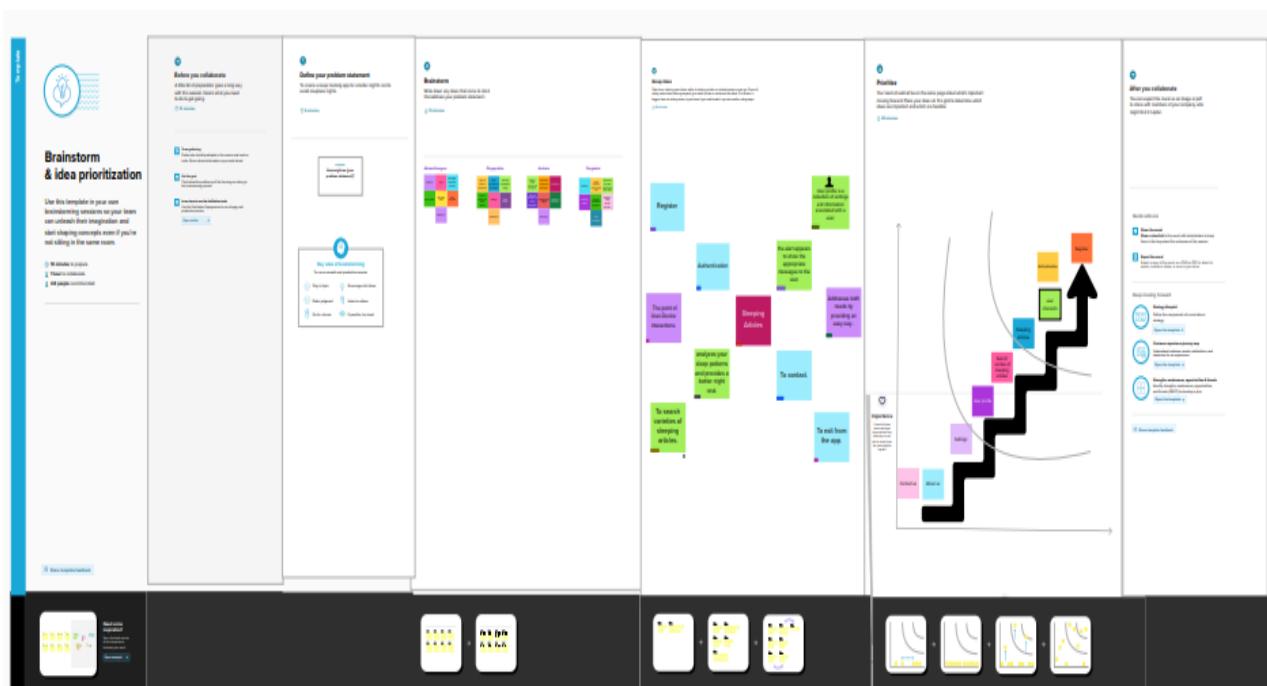
**The use of this App, what can be achieved using this:**

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.

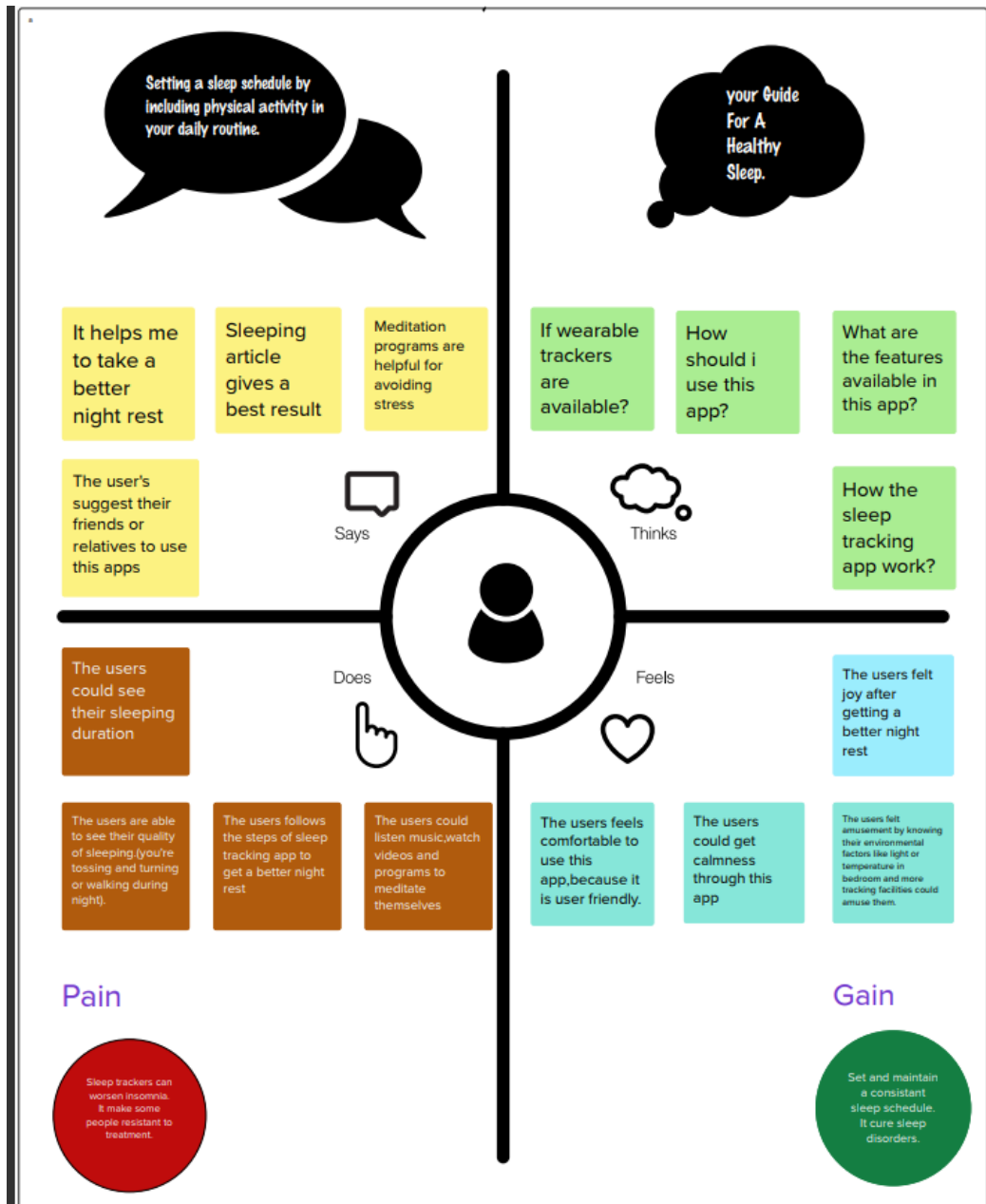
Sleep track is our best overall app as it is available for IOS and Android and offers many great features to track and improve your sleep. As the most advanced sleep tracker on this list, sleep tracking app uses sonar technology to measure sleep habits from your better night rest.

## 2. PROBLEM DEFINITION & DESIGN THINKING

### 2.1 EMPATHY MAP



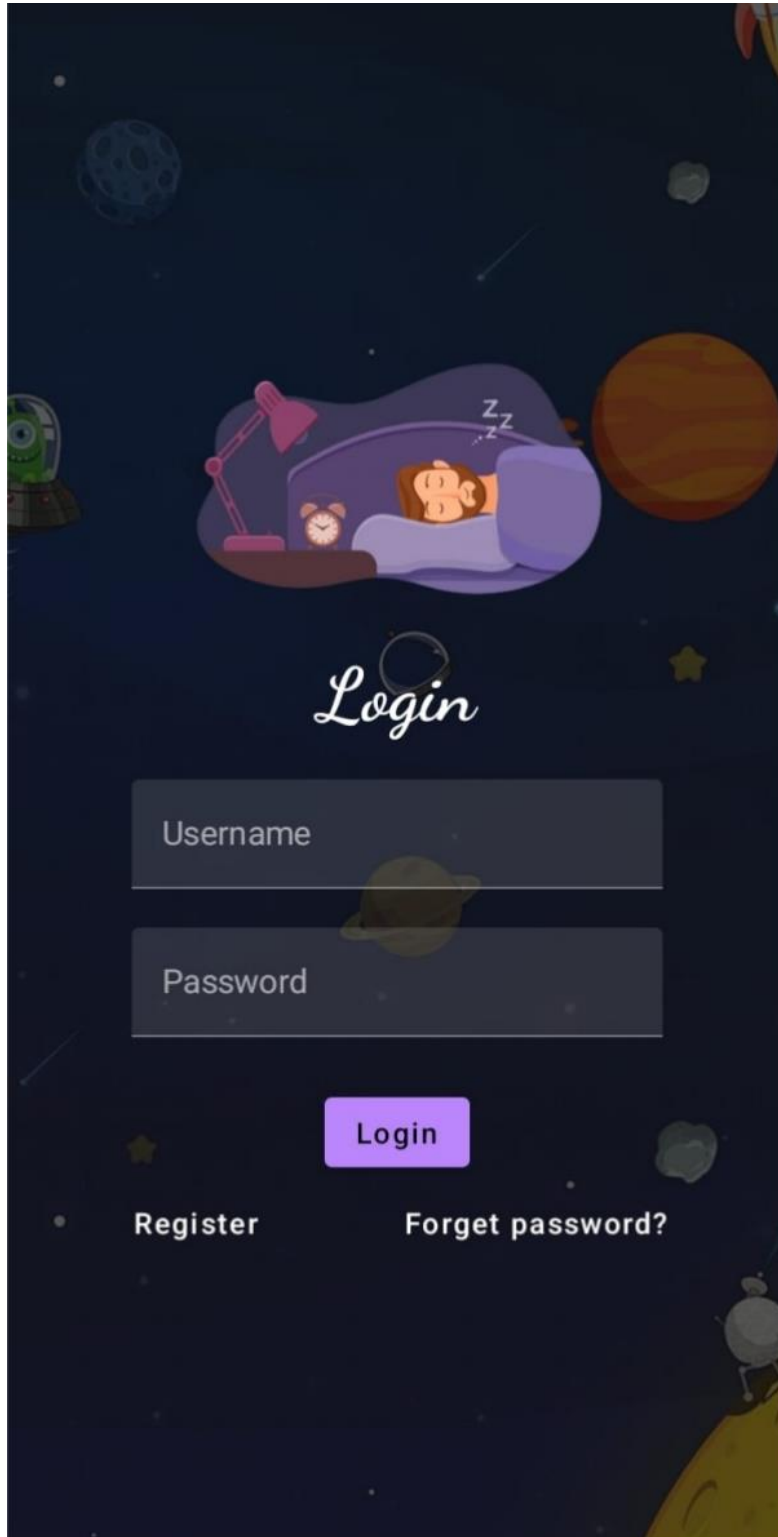
## 2.2 IDEATION & BRAINSTORMING MAP



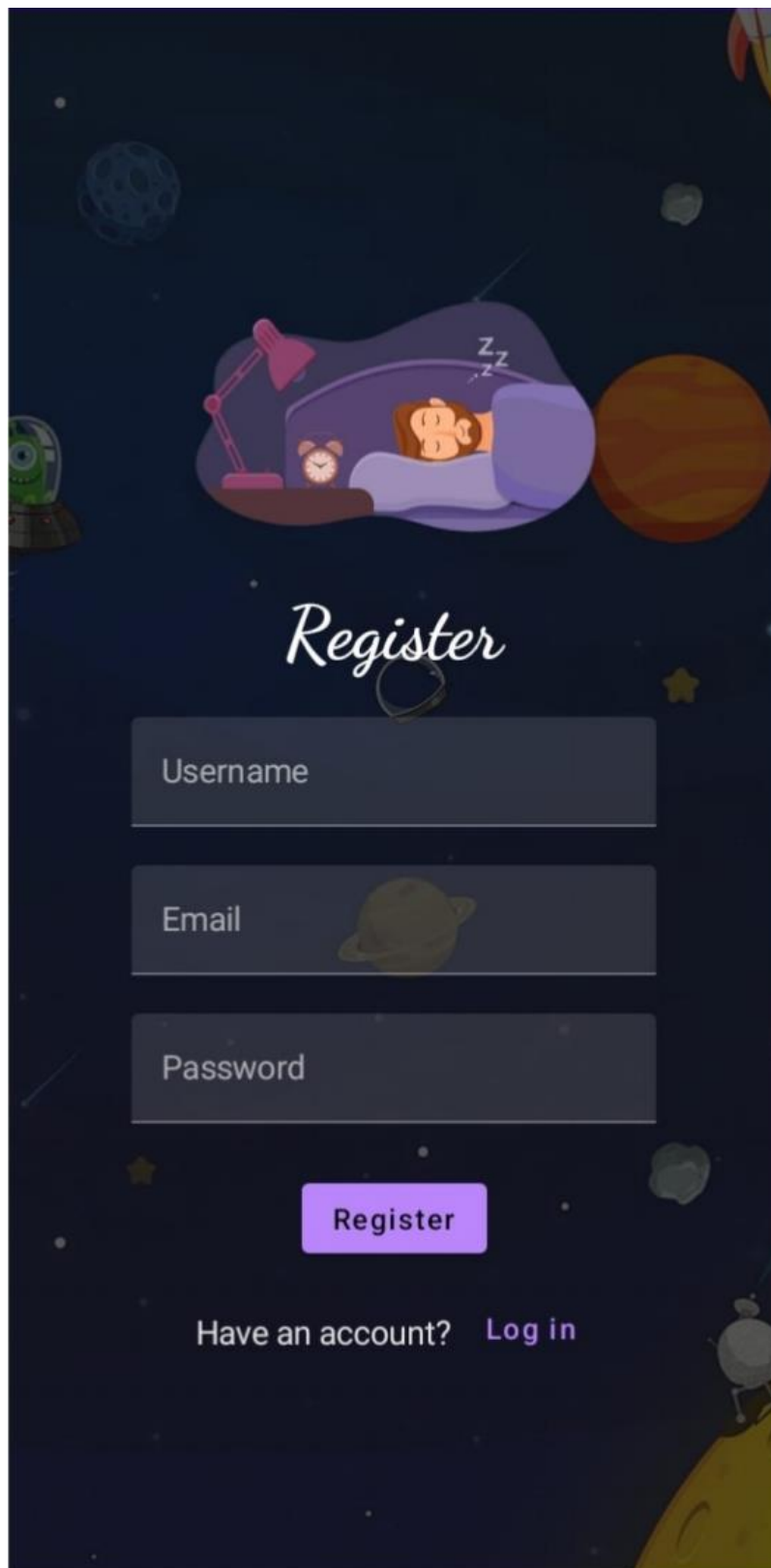
### 3. RESULT:

**Final findings (output) of the project:**

**Login Page:**



## Registration Page:

The background of the registration page is a dark blue space-themed illustration. It features a man with a beard sleeping in a bed with a pink lamp and a clock. There are also planets, stars, and a small robot in the background.

*Register*

Username

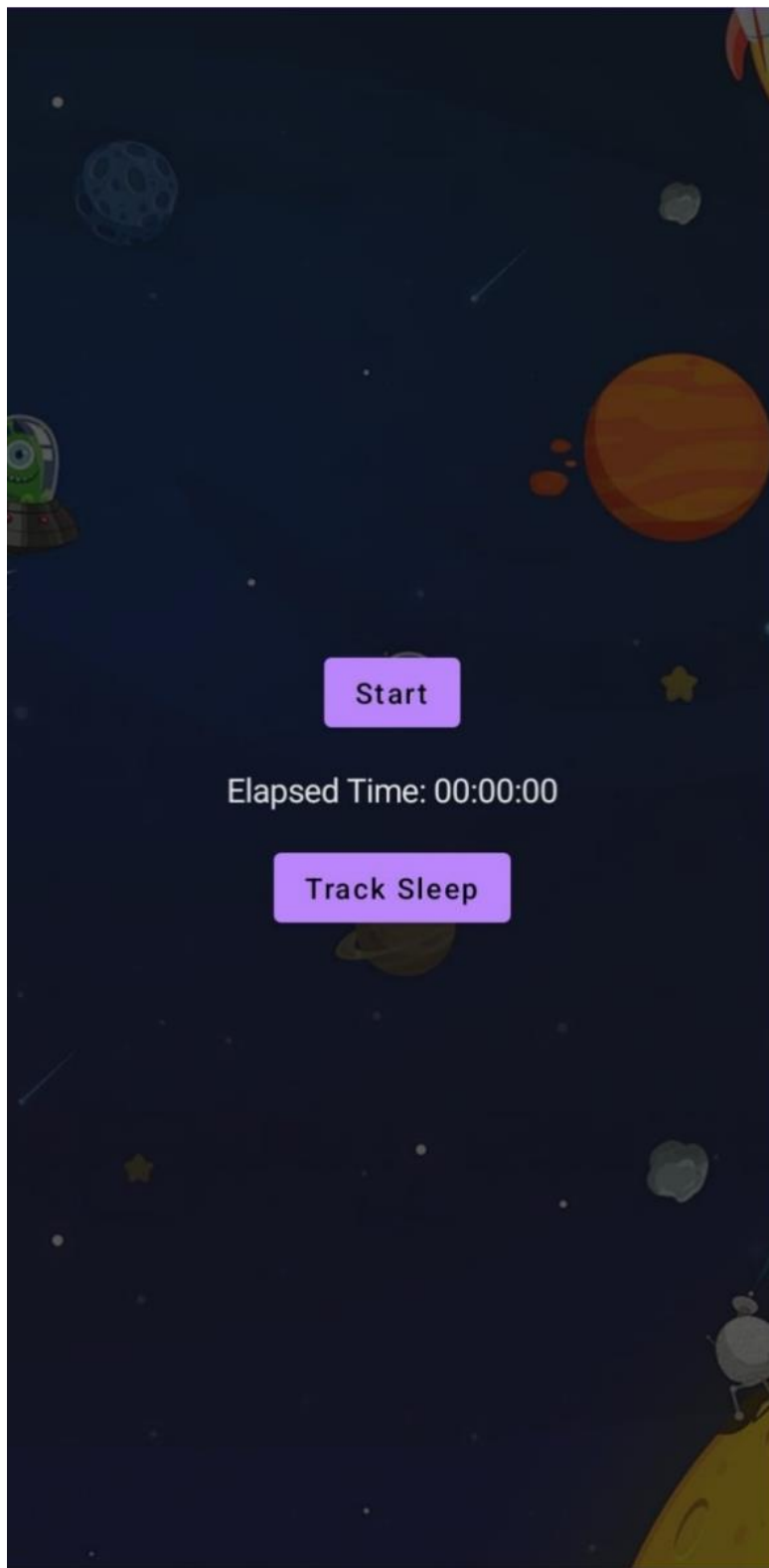
Email

Password

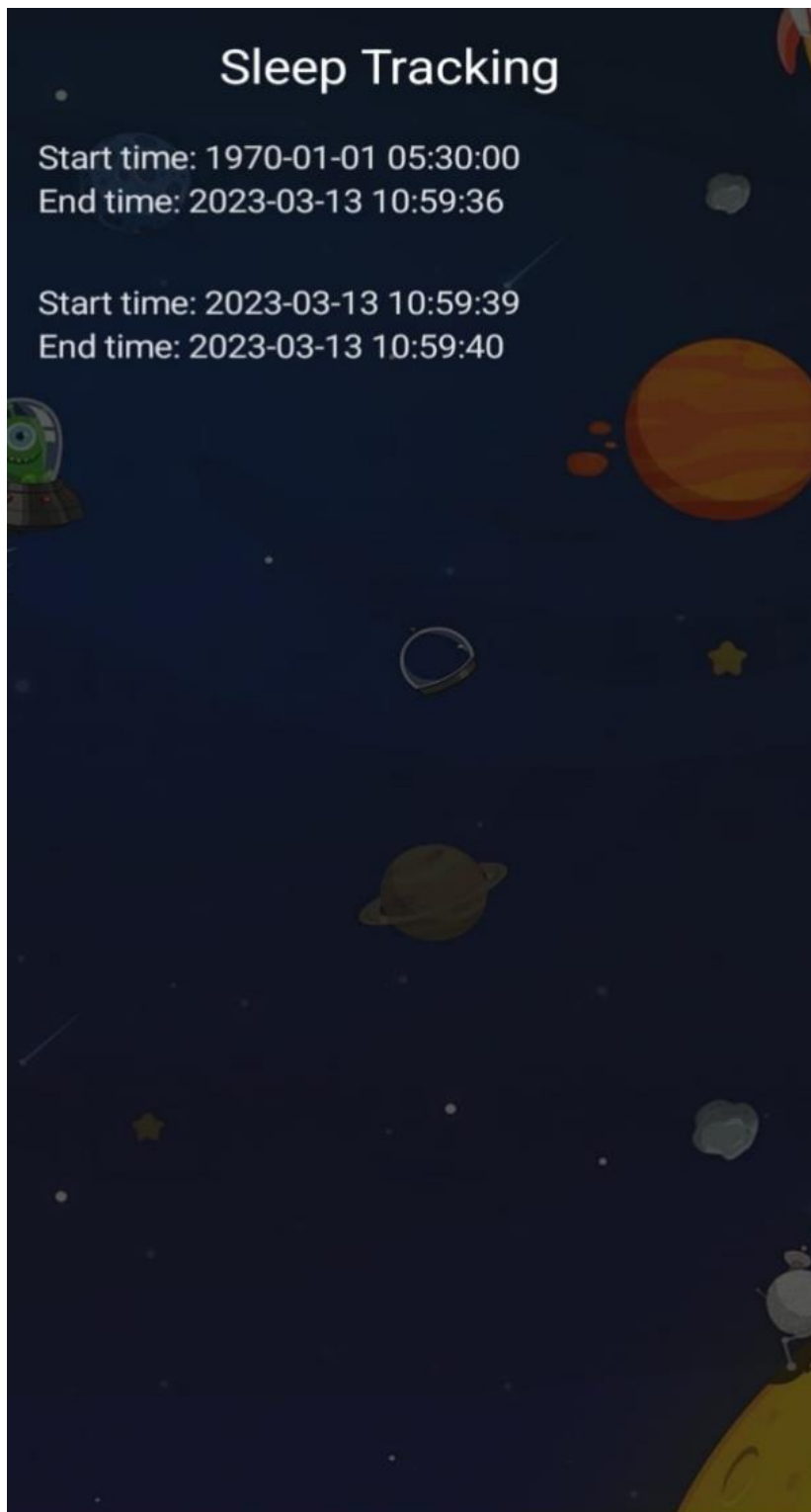
Register

Have an account? [Log in](#)

## Main Page:



## Track Sleep Page:





## **4. ADVANTAGES AND DISADVANTAGES:**

### **Advantages:**

- Get sick less often.
- Stay at a healthy weight.
- Lower your risk for serious health problems, like diabetes and heart disease.
- Reduce stress and improve your mood.
- Think more clearly and do better in school and at work.
- Get along better with people.

### **Disadvantages:**

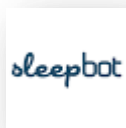
- Sleep trackers introduce poor sleep hygiene. ...
- Sleep trackers may be inaccurate. ...
- Sleep trackers can worsen insomnia. ...
- Sleep trackers make some people resistant to treatment. ...
- Sleep trackers are tied to a sleep disorder.

## 5. APPLICATIONS :

Prime Nap



Sleep Sounds - Relax & Sleep



Auto Sleep Track Sleep on Watch



Noisli



Sleep Reset



Sleep Talk Recorder



## Pillow - Auto Sleep Tracker



Snore Lab

## Sleep++



Sleep Monitor



Sleep meditation app



Digipill



Sleepzy



Yours App

## Relax Melodies & Sleep Sounds



Relax & Sleep Well Hypnosis

Sleep Monitor



White Noise Lite



Prime Sleep Recorder



Google Fit

Google Fit

Better Sleep



Samsung Health

Sleep by Slumber



Somryst



Shut Eye



Alarm Clock Xtreme

Twilight



Do I Snore or Grind

Breathwrk

Breathwrk



Insight Timer - Meditation App



Calm Sleep

Better Sleep



Snore App

White Noise Baby



Sleep Town



Alarm Clock App

## 6. CONCLUSION:

- Apps available for sleep self-management and tracking may be valuable tools for self-management of sleep disorder and/or improving sleep quality, yet they require improvement in terms of quality and content, highlighting the need for further validity studies.
- **Keywords:** Sleep Tracker, Sleep Health, Smartphone Applications, Behavioural Change, Consumer Sleep Technology, Mobile Devices
- Amid concerns regarding the limited validity of mobile sleep trackers and variation in results compared to golden standard PSG, other roles of such apps must be considered. We conclude that sleep trackers may be useful in improving user's self-management, and increasing sleep hygiene awareness, knowledge, and behaviours. Thus, apps may present valuable tools for improving sleep quality. However, continuous audits and validation trials for available apps are vital to improve their quality. It is recommended to assess behavioral changes associated with sleep trackers in different populations, such as elders, and people with sleep disorders and major illnesses.

## **7. FUTURE SCOPE**

In recent years, there has been a significant expansion in the development and use of multi-modal sensors and technologies to monitor physical activity, sleep and circadian rhythms. These developments make accurate sleep monitoring at scale a possibility for the first time. Vast amounts of multi-sensor data are being generated with potential applications ranging from large-scale epidemiological research linking sleep patterns to disease, to wellness applications, including the sleep coaching of individuals with chronic conditions. However, in order to realise the full potential of these technologies for individuals, medicine and research, several significant challenges must be overcome. There are important outstanding questions regarding performance evaluation, as well as data storage, curation, processing, integration, modelling and interpretation. Here, we leverage expertise across neuroscience, clinical medicine, bioengineering, electrical engineering, epidemiology, computer science, mHealth and human–computer interaction to discuss the digitisation of sleep from an inter-disciplinary perspective. We introduce the state-of-the-art in sleep-monitoring technologies, and discuss the opportunities and challenges from data acquisition to the eventual application of insights in clinical and consumer settings. Further, we explore the strengths and limitations of current and emerging sensing methods with a particular focus on novel data-driven technologies, such as Artificial Intelligence.

## 8. APPENDIX

```
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?,
)
```

```
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)
    }
}
```

```
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()
    }
    @SuppressWarnings("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_FIRST_NAME = ?",

```

```

        arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD))
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressWarnings("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
}
}

```

package com.example.projectone

```

import androidx.room.Entity
import
androidx.room.PrimaryKey

```

```

package com.example.projectone

import java.sql.Date
@Entity(tableName =
"TimeLog")
data class TimeLog(

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

import androidx.room.Dao
import androidx.room.Insert
@Dao
interface TimeLogDao {
    @Insert
    suspend fun
insert(timeLog: TimeLog)
}

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
            )
        )*/
    })
    {
        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)
}

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)
}

```

```

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)
    }
}

```

```

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 dbHelper: TimeLogDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        dbHelper = 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))
}

```

```

<?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:supportRtl="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>
```

