A SLEEP TRACKING APP FOR A BETTER NIGHT’S SLEEP

1.INTRODUCTION

1.1 OVERVIEW

A project that demonstrates the use of Android Jetpack Compose to build a UI for a sleep tracking app. The app allows users to track their sleep. With the “Sleep Tracker” app, you can assess the quality of sleep they have had in a day. It has been time and again proven that a good quality sleep is pretty essential for effective functioning of both mind and body.

“Sleep Tracker” application enables you to start the timer when they are in the bed and about to fall asleep. The timer will keep running in the background until it is stopped, whenever the user wakes up. Based on the sleep experience, you can rate your sleep quality. Finally, the app will display an analysis of the kind of sleep, you had the previous night.

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

When activated for the night, our sleep recorder is trained to recognize the kind of sounds that normally occur when we're asleep and our detection algorithm categorizes each type of noise to easily listen back to them the next morning. It will also show you your sleep data for the day/week.

1.2 PURPOSE

Helps You Better Understand Your Sleep Patterns woman lying in bed and sleeping next to smartphone. In order to improve your sleep patterns, you need to understand them. Most sleep tracker apps record an extensive amount of data about your sleep, such as the time you fell asleep, the time you woke up, your sleep regularity, and sleep duration.

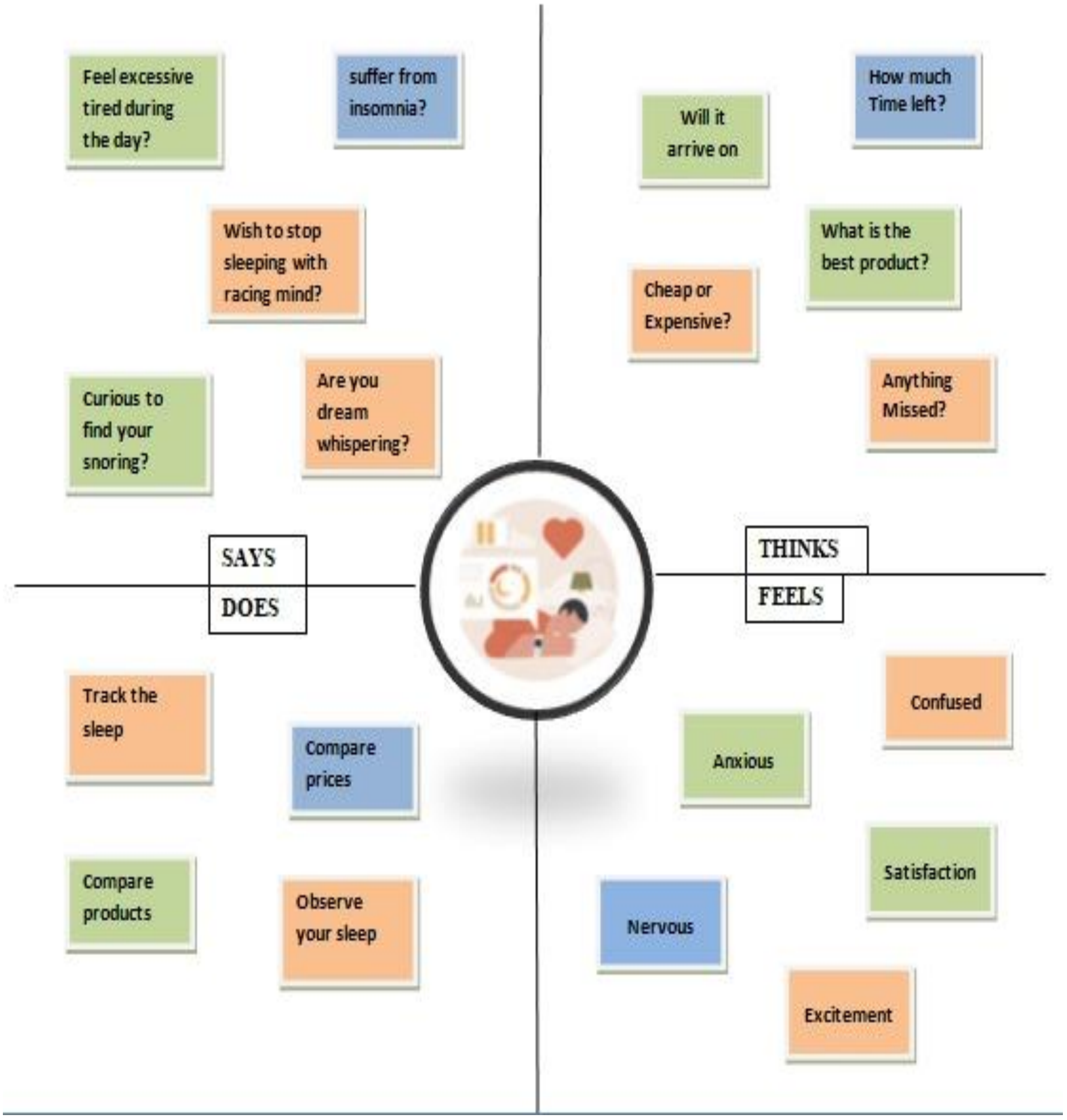
This important sleep data can reveal in detail how your sleep patterns affect you throughout the day, including your mood, productivity, et cetera.

Sleep is probably the most important part of your health and wellness, yet so many people struggle to get a good night's rest. If you want to reap the benefits of sleep, which include improved concentration, a boosted mood, and reduced stress levels, technology may be the solution.

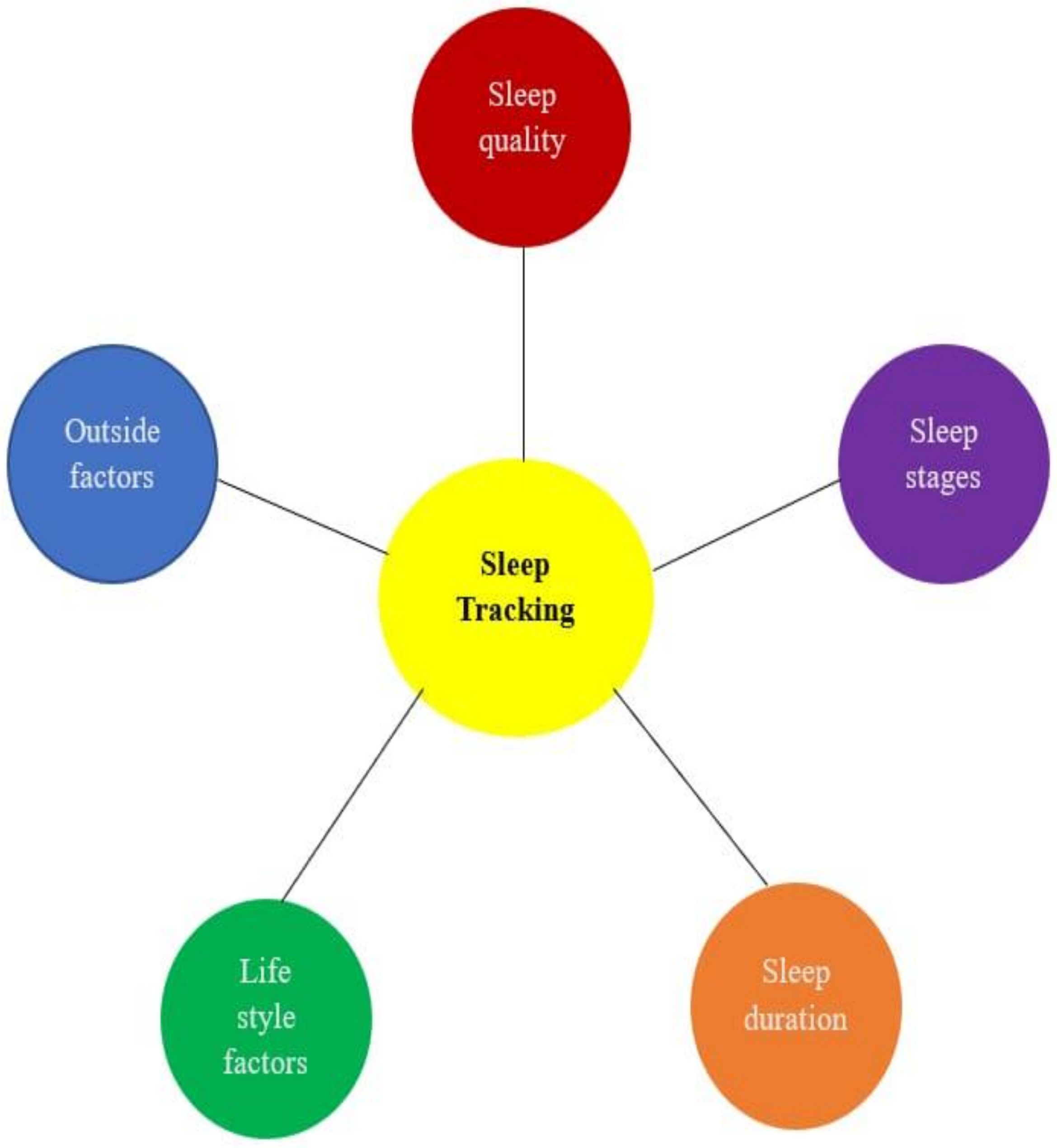
Sleep tracking apps can help you better understand your sleep and get some shut-eye, but along with the pros come some cons.

2.PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map



2.2 Ideation and brainstorming map



3.RESULT

Login page

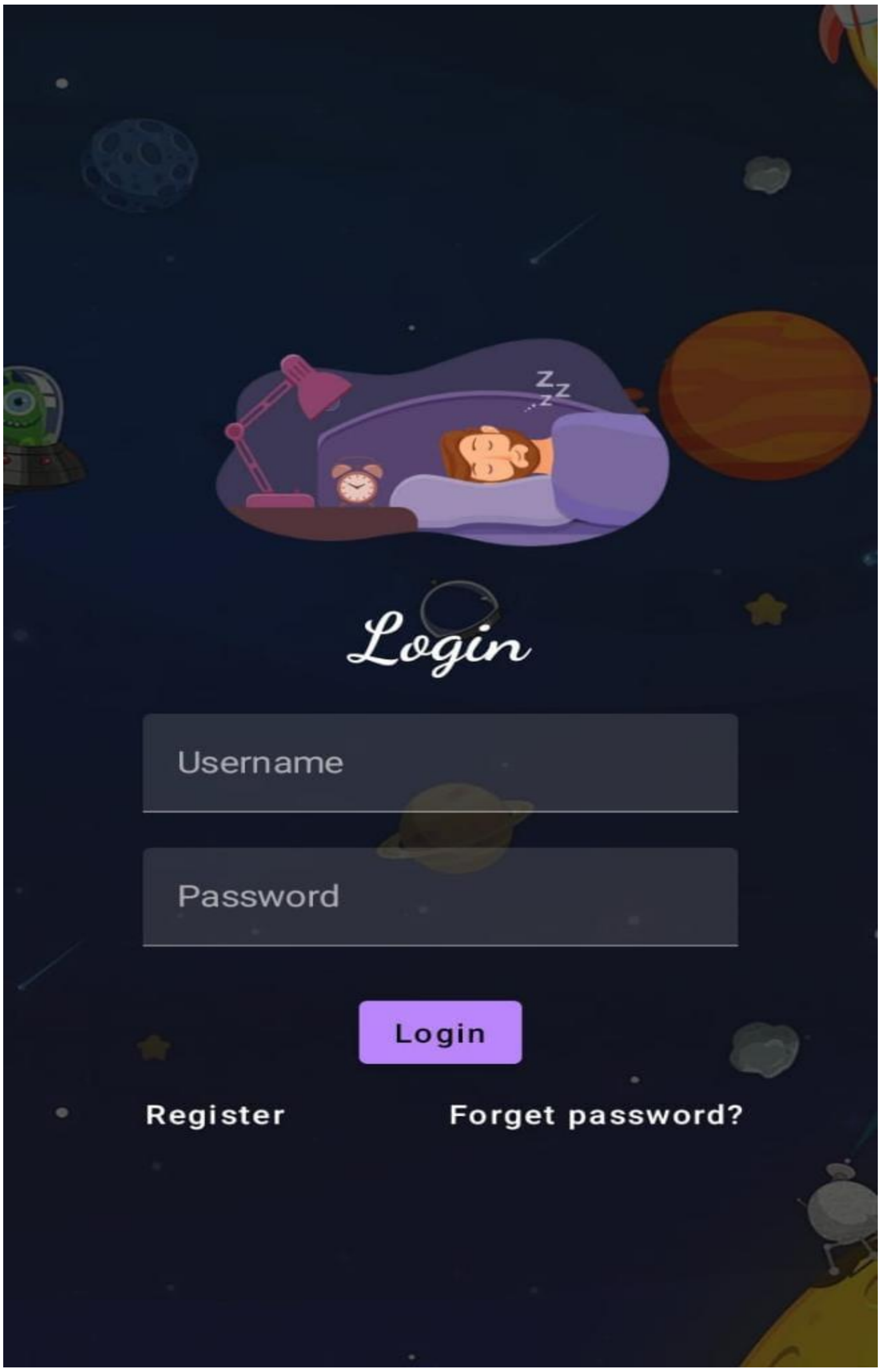


Figure 3.1

Register page

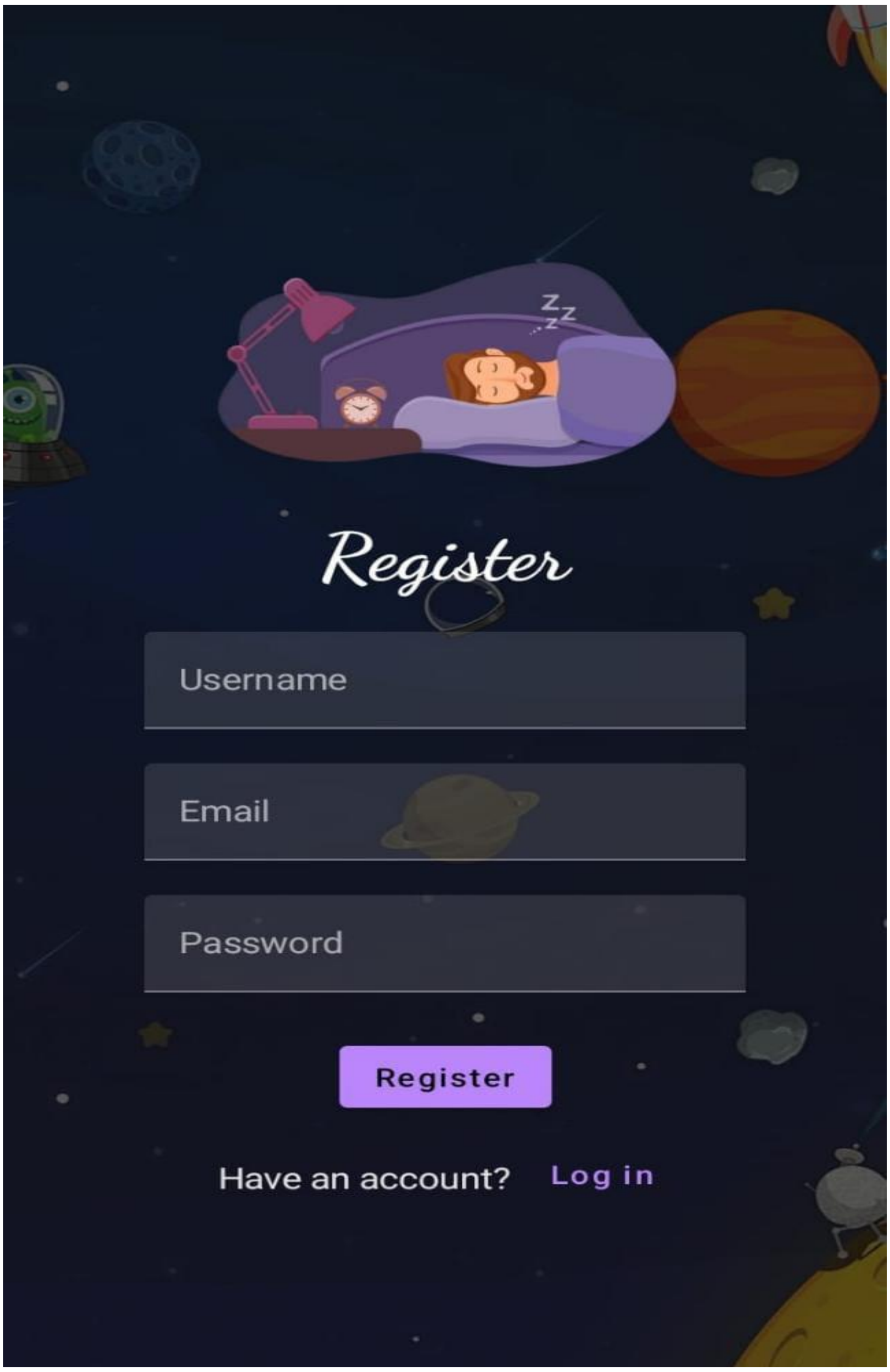


Figure 3.2

Tracking page

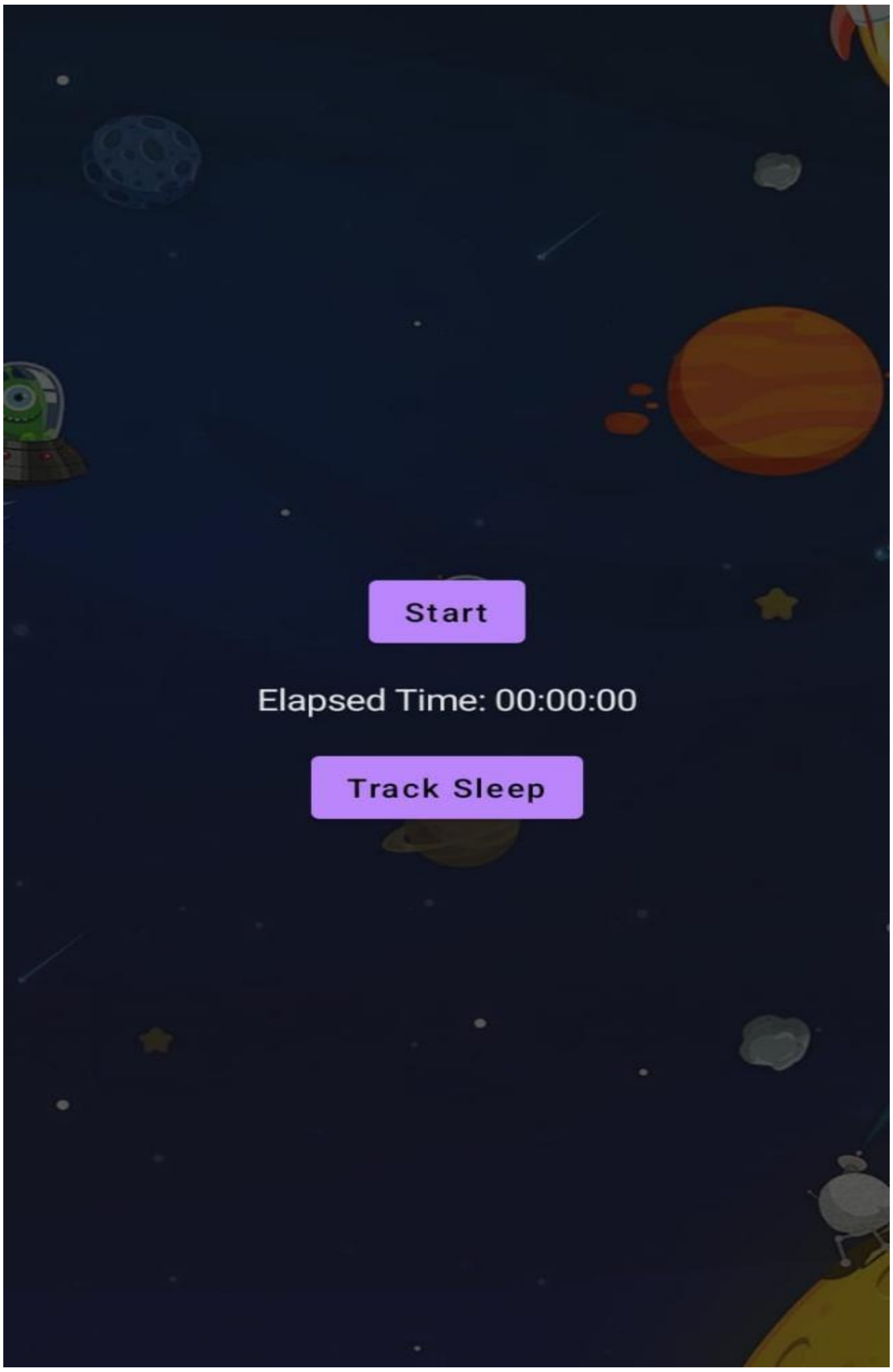
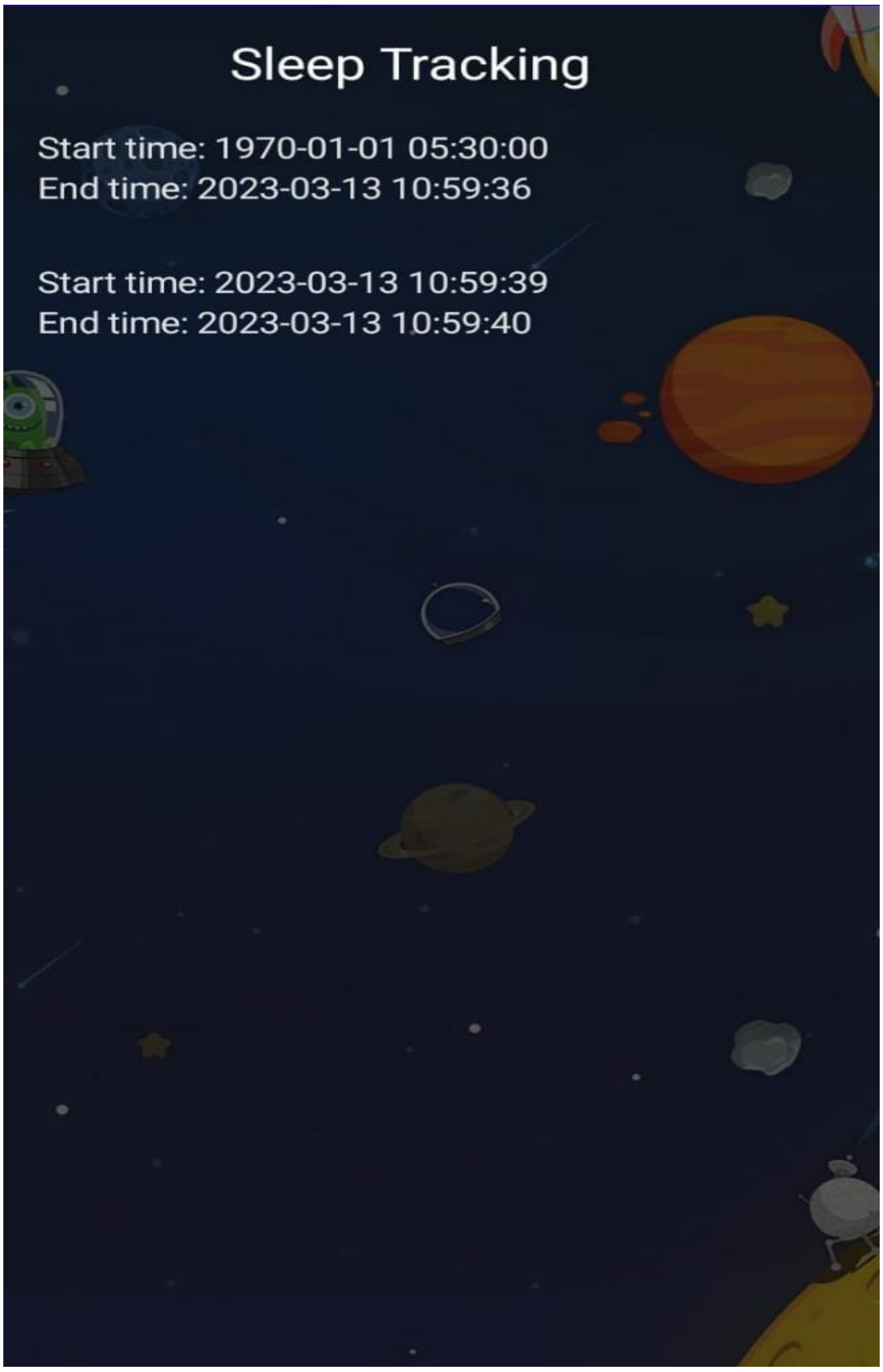


Figure 3.3

Tracked data


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

Figure 3.4

4.ADVANTAGE AND DISADVANTAGES

Advantage

You can manage what you measure. Dr. Kelly Baron, director of the University of Utah's behavioral sleep medicine program, claims this is the biggest benefit to monitoring your sleep. The idea is simple: If you can detect patterns in how your nighttime behaviors impact your sleep quality and duration, you can make changes to help you sleep better.

Sleep trackers may help with bedtime routines. Dr. Conor Heneghan, lead research scientist at Fitbit, argues that tracking your sleep makes you more conscious of when you go to bed and wake up each day. As a result, it may help you get the seven to nine hours of sleep you need.

Sleep trackers may help detect sleep disorders. Advanced sensors that measure your blood oxygen can flag the possibility that you have sleep apnea. However, a sleep study prescribed by your doctor will be required for an official diagnosis.

Sleep trackers may help you wake up. Some sleep trackers, which claim to differentiate between light and deep sleep stages, have special alarms that wake you up when you're closest to being awake. The theory is that you'll find it easier to wake up and be in a better mood as a result

Disadvantage

Sleep trackers introduce poor sleep hygiene. When it comes to proper sleep hygiene, viewing devices immediately before or after sleep is a no-no. A sleep tracker may incentivize people to break this rule, argues Dr. Baron and colleagues.

Sleep trackers may be inaccurate. Sleep trackers may suggest you get more sleep than you do in reality. This is especially true for trackers that rely on movement sensors, as you could simply be lying awake at night. They're also unable to accurately distinguish between light and deep sleep. “Even medically validated diagnostic technologies are at risk for false positive and false negative results," points out Massachusetts General Hospital's Kathryn Russo and colleagues.

Sleep trackers can worsen insomnia. Across three case studies, patients spent an excessive amount of time in bed trying to maximize their sleep duration. Unfortunately, that's known to exacerbate insomnia.

Sleep trackers make some people resistant to treatment. In the same study, patients trusted data from their wearable devices over results from official sleep studies. They also neglected evidence-based treatments for insomnia, instead preferring to go it alone.

Sleep trackers are tied to a sleep disorder. Some people who wear sleep trackers become preoccupied with optimizing their sleep data. This condition, known as orthosomnia, enhances your nighttime anxiety, which can make it harder for you to sleep.

5.APPLICATION

Sleep tracking application may be valuable for user self-management and improvement of sleep hygiene. In addition, they may help increase awareness and promote help seeking regarding sleep-related issues.

However, the severe lack of validation studies raises concerns around their use and limits their function as alternatives to standard clinical tools. Moreover, assessment of available applications is necessary to guide physician recommendations of sleep tracker apps for patient use. To our knowledge, this is the first study to assess the perceived behavioural changes associated with sleep health applications.

6.CONCLUTION

It was very interesting to discover that when users are using a health app they want to feel that it is more personal and not like they are using a cold automated device. For them, it was crucial to feel that their sleep is automatically tracked but that they also have the opportunity to provide their own input. Only then they would feel that their data has value for their health.

As designers, it was a great reminder to discover that users with sleep issues need clean and organized info as this would cause them less stress. The fast pace routines usually affect them not only during the day but mainly right before their night sleep so it was crucial for them to have an app that would ease this disturbance.

The covid pandemic and the cultural differences were two factors that determined the type of communication the users have with their health providers. Users prefer to have asynchronous/distant contact with their doctor for sleep issues. “Zzzloth” sleep tracking app seemed to the users a convenient way to communicate with their doctor efficiently and the feeling that they have someone taking care of them when needed.

This was a very insightful project, during which I feel I grew a lot as a designer. It was obvious that having a thorough and solid User Research process, helped us to understand the user from all perspectives. This knowledge gave us the power to compose realistic assumptions and to visually design the app in the most efficient way, which our tester users also confirmed in the end.

7.FUTURE SCOPE

The design should be focused on an MVP, as the app was not supposed to solve all of the wellness issues people have.

Must-Have:

- Users need to be able to set up their profile to include important information relevant to their goals
- Users need to be able to set goals and track their progress
- Users need to be able to share their status with their wellness coaches

Currently AI Application services are limited to Android Platform only, but in future it will be extrapolated to all part of the Mobile Application Platforms.

In future, this app is published in Play store to access easily by user.

8.APPENDIX

A. Source code

Creating Database

User.kt

```
package com.example.sleeptracking

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "user_table")

data class User(

    @PrimaryKey(autoGenerate = true) val id: Int?,

    @ColumnInfo(name = "first_name") valfirstName: String?,

    @ColumnInfo(name = "last_name") vallastName: String?,

    @ColumnInfo(name = "email") val email: String?,

    @ColumnInfo(name = "password") val password: String?,

    )
```

Userdao.kt

```
package com.example.sleeptracking

import androidx.room.*

@Dao

interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)

    suspend fun insertUser(user: User)

    @Update

    suspend fun updateUser(user: User)

    @Delete

    suspend fun deleteUser(user: User)

}
```

UserDatabase.kt

```
package com.example.sleeptracking

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) {

valnewInstance = Room.databaseBuilder(
context.applicationContext,
UserDatabase::class.java,
                    "user_database"
).build()
                instance = newInstance
newInstance
            }}}}
```

UserDatabaseHelper.kt

```
package com.example.sleeptracking
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
    companion object {
        private constval DATABASE_VERSION = 1
        private constval DATABASE_NAME = "UserDatabase.db"
        private constval TABLE_NAME = "user_table"
        private constval COLUMN_ID = "id"
        private constval COLUMN_FIRST_NAME = "first_name"
        private constval COLUMN_LAST_NAME = "last_name"

        private constval COLUMN_EMAIL = "email"

        private constval COLUMN_PASSWORD = "password"

    } override fun onCreate(db: SQLiteDatabase?) {

valcreateTable = "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)

}funinsertUser(user: User) {

valdb = 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? {
valdb = 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? {
valdb = 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>()
valdb = 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}}
```

Timelog.kt

```
package com.example.sleeptracking
import androidx.room.Entity
import androidx.room.PrimaryKey
import java.sql.Date
@Entity(tableName = "TimeLog")
data class TimeLog(
@PrimaryKey(autoGenerate = true)
val id: Int = 0,
valstartTime: Date,
valstopTime: Date
)
```

Timelogdao

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

AppDatabaseClass:

```
package com.example.sleeptracking
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [TimeLog::class], version = 1, exportSchema = false)
abstract class AppDatabase :RoomDatabase() {
    abstract fun timeLogDao(): TimeLogDao
    companion object {
        private var INSTANCE: AppDatabase? = null
        fun getDatabase(context: Context): AppDatabase {
valtempInstance = INSTANCE
            if (tempInstance != null) {
                return tempInstance
}synchronized(this) {
val instance = Room.databaseBuilder(
context.applicationContext,
AppDatabase::class.java,
                    "app_database"
).build()
```

```
            INSTANCE = instance

            return instance

        } }}}
```

Login Activity.kt

```
package com.example.sleeptracking

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.*

import androidx.compose.material.*

import androidx.compose.runtime.*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.draw.alpha

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.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("") }

valimageModifier = 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)
}
```

Registration Activity.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("") }

valimageModifier = 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)

}
```

Main Activity.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) }

valimageModifier = 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 {
valdateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())
valcurrentTime = 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)
}
```

Track Activity.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())
valtimeLogs = databaseHelper.getTimeLogs()
ListListScopeSample(timeLogs)
            }}}}}
   @Composable
fun ListListScopeSample(timeLogs: List<TimeLogDatabaseHelper.TimeLog>) {
valimageModifier = 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 {

valdateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())

    return dateFormat.format(Date(timestamp))

}
```

Android Manifest 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>
```