A SLEEP TRACKING APP FOR A BETTER NIGHT'S REST

Introduction

1.1 Overview

Project Description

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.

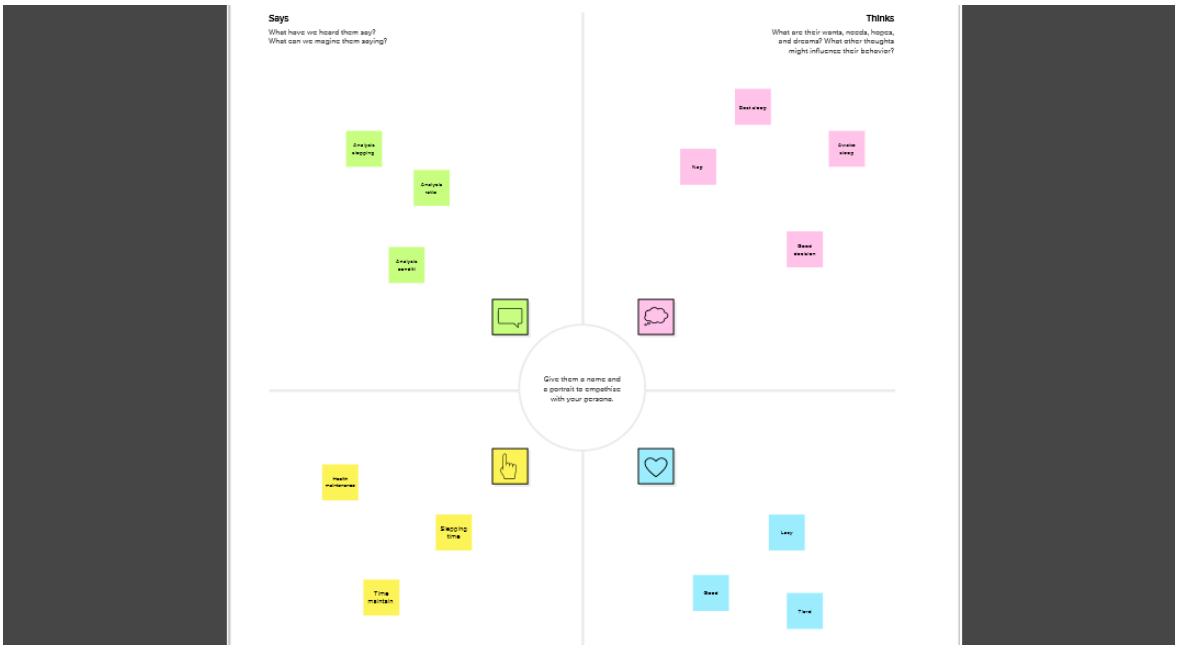
"Sleep Tracker" application enables you to start the timer when they are in the bed and about to fall asleep. The timer will keep running in the background until it is stopped, whenever the user wakes up. Based on the sleep experience, you can rate your sleep quality. Finally , the app will display an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

1.2 Purpose

Self-management applications on a smart phone have been advocated as a means to help people with sleep disturbance to achieve better lavels of sleep control and beater seep quality out comes. With increased mobile phone ownership on a global scale, a rapid devolepment in health technologies allows users to self-monitor and visualize their sleep patterns, symptoms, and behavioural data and aid them in taking appropriate actions on a potentially daily basis 22. Most of the available sleep tracking apps are accelerometer-based and can employ built-in mobile sensors such as microphone and light detectors to obtain sleep data 1. Howerer, and unlike accelerometry used in standard sleep assessment tools, mobile accelerometer operates via undefined and means a concern due to the lack of validation studies1,22.

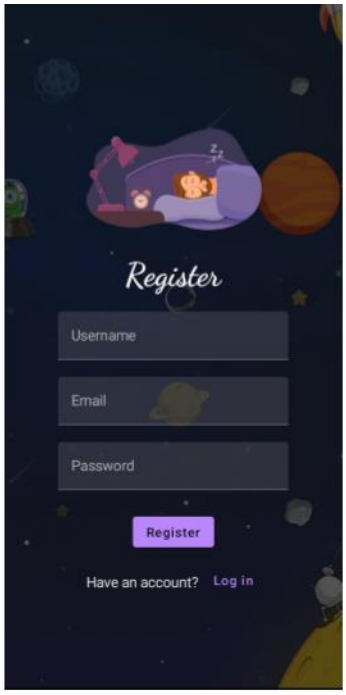
Problem Definition & Design thinking

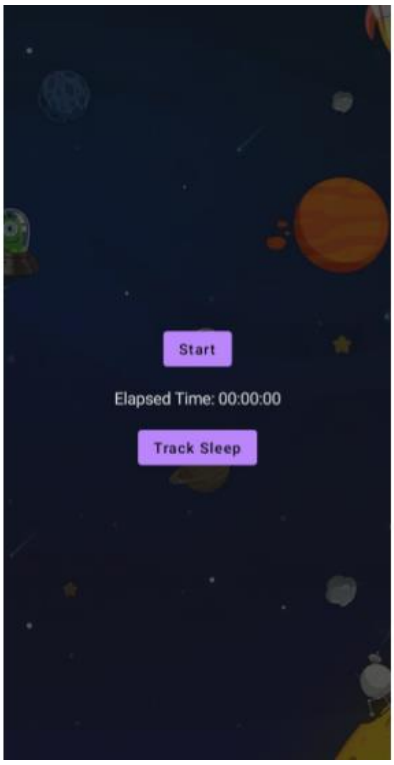
2.1 Empathy Map

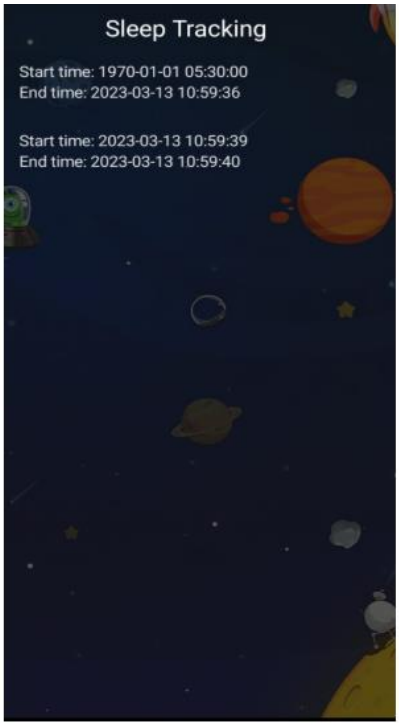


RESULT


Login
Username
Password
Login
Register
Forget password?


Register
Username
Email
Password
Register
Have an account? Log in


Start
Elapsed Time: 00:00:00
Track Sleep


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

ADVANTAGES

- They are great tools to measure the quality of your sleep.

- It can especially help if you have issues waking up often during the night or feel sleepy throughout the day.

- If you are a more anxious person, be cautious about tracking your sleep.

DISADVANTAGES

- The stigma arount employee monitoring. The srigma with employee monitoring is that it negatively impacts employee morale.

- Feelings of distrust, some employees may feel like their employers dou't trust them.

- Employee privacy concerns.

- Lega issues in monitoring.

APPLICATION

These applications provide a wide range of functions, incuding smart alarms, sleep aids, sound recording during sleep, and analysis of sleep. Other applications were designed to assist healthcare professionas in monitoring and screening their patients for habitual snoring and obstrucrive sleep apnea. A few health trackers were compared to standard sleep assessment tests including polysomnography(PSG), wrist actigraphy, and the Pittsburgh sleep quality index (PSQI).parameters assessed in validation studies included sleep onset latency, total sleep time, snoring events, sleep stages, and sleep efficacy. Most application have shown good correlation to wrist actigraphy but not PSG. Moreover, a drop in reliability was commonly seen in clinical populations compared to heathly users, a trend also seen with conventional actigraphy24,25,26,27,28,29.

CONCLUSION

App 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 furthee validity studies.

APPENDIX

User.kt

```
Import andoridx.room.Columninfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,
    )
```

UserDao.kt

```
import android.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

```
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
            }
        }
    }
}
```

UserDatabaseHelpter.kt

```
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
            }
        }
```

Timelog.kt

```
        import android.room.Entity
        import androidx.room.PrimaryKey
        import java.sql.Date
        @Entity(tableName = "TimeLog")
        data class TimeLog(
            @PrimaryKey(autoGenerate = true)
            val id: Int = 0,
            val startTime: Date,
            val stopTime: Date
        )
```

TimeLog.kt

```
        Import android.room.Dao
        import androidx.room.Insert
        @Dao
        interface TimeLogDao {
```

```
    @Insert
    suspend fun insert(timeLog: TimeLog)
}
```

TimeDatabaseHelper.kt

```
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"

        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)
    }
    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)
    }

    @SuppressLint("Range")
    fun getTimeLogs(): List<TimeLog> {
        val timeLogs = mutableListOf<TimeLog>()
        val cursor = readableDatabase.rawQuery("select * from
$TABLE_NAME", null)
        cursor.moveToFirst()
        while (!cursor.isAfterLast) {
            val id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_START_TIME))
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_END_TIME))
            timeLogs.add(TimeLog(id, startTime, endTime))
            cursor.moveToNext()
        }
        cursor.close()
        return timeLogs
    }
    fun deleteAllData() {
        writableDatabase.execSQL("DELETE FROM $TABLE_NAME")
    }
    fun getAllData(): Cursor? {
        val db = this.writableDatabase
```

```
        return db.rawQuery("select * from $TABLE_NAME", null)
    }
    data class TimeLog(val id: Int, val startTime: Long, val endTime:
Long?) {
        fun getFormattedStartTime(): String {
            return Date(startTime).toString()
        }
        fun getFormattedEndTime(): String {
            return endTime?.let { Date(it).toString() } ?: "not
ended"
        }
    }
}
```

AppDatabase.kt

```
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

```
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 {

                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    LoginScreen(this, databaseHelper)
                }
            }
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper)
{
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )
    Column(
```

```
    modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    Image(
        painter = painterResource(id = R.drawable.sleep),
        contentDescription = "",
        modifier = imageModifier
            .width(260.dp)
            .height(200.dp)
    )
    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,
        color = Color.White,
        text = "Login"
    )
    Spacer(modifier = Modifier.height(10.dp))
    TextField(
        value = username,
        onValueChange = { username = it },
        label = { Text("Username") },
        modifier = Modifier.padding(10.dp)
            .width(280.dp)
    )
    TextField(
        value = password,
        onValueChange = { password = it },
        label = { Text("Password") },
        modifier = Modifier.padding(10.dp)
            .width(280.dp)
    )
    if (error.isNotEmpty()) {
```

```
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }
        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user =
databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainActivity::class.java
                            )
                        )
                        //onLoginSuccess()
                    } else {
                        error = "Invalid username or password"
                    }
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
```

```
                        context,
                        MainActivity2::class.java
                    )
                )}
                )
                { Text(color = Color.White,text = "Sign up") }
                TextButton(onClick = {
                    /*startActivity(
                    Intent(
                        applicationContext,
                        MainActivity2::class.java
                    )
                )*/
                })
                {
                    Spacer(modifier = Modifier.width(60.dp))
                    Text(color = Color.White,text = "Forget password?")
                }
            }
        }
    }
    private fun startMainPage(context: Context) {
        val intent = Intent(context, MainActivity2::class.java)
        ContextCompat.startActivity(context, intent, null)
    }
```

RegisterActivity.kt

```
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"

                    context.startActivity(
                        Intent(
                            context,
                            LoginActivity::class.java
                        )
                    )
                } else {
                    error = "Please fill all fields"
                }
            },
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))
        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text =
"Have an account?"
```

```
                )
                TextButton(onClick = {
                })
                {
                    Spacer(modifier = Modifier.width(10.dp))
                    Text(text = "Log in")
                }
            }
        }
    }
    private fun startLoginActivity(context: Context) {
        val intent = Intent(context, LoginActivity::class.java)
        ContextCompat.startActivity(context, intent, null)
    }
```

MainActivity.kt

```
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 {
                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")
            }
        } else {
            Button(onClick = {
                elapsedTime =
System.currentTimeMillis()
```

```
                isRunning = false
            }) {
                Text("Stop")

databaseHelper.addTimeLog(elapsedTime,startTime)
            }
        }
        Spacer(modifier =
Modifier.height(16.dp))
        Text(text = "Elapsed Time:
${formatTime(elapsedTime - startTime)}")
        Spacer(modifier =
Modifier.height(16.dp))
        Button(onClick = {
context.startActivity(
            Intent(
                context,
                TrackActivity::class.java
            )
        ) }) {
            Text(text = "Track Sleep")
        }
    }
}
private fun startTrackActivity(context: Context)
{
    val intent = Intent(context,
TrackActivity::class.java)
    ContextCompat.startActivity(context, intent,
null)
}
fun getCurrentDateTime(): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-
dd HH:mm:ss", Locale.getDefault())
```

```
        val currentTime = System.currentTimeMillis()
        return dateFormat.format(Date(currentTime))
    }
    fun formatTime(timeInMillis: Long): String {
        val hours = (timeInMillis / (1000 * 60 *
    60)) % 24
        val minutes = (timeInMillis / (1000 * 60)) %
    60
        val seconds = (timeInMillis / 1000) % 60
        return String.format("%02d:%02d:%02d",
    hours, minutes, seconds)
    }
```

TrackActivity.kt

```
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 {

                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {

                    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("Start time:
${formatDateTime(timeLog.startTime)}")
                        Text("End time: ${timeLog.endTime?.let {
formatDateTime(it) }}")
                    }
                }
            }
        }
        }
    }
private fun formatDateTime(timestamp: Long): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
    return dateFormat.format(Date(timestamp))
}
```

AndroidManifest.Xml

```
<xml version=”1.0” encoding=”jtf-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>
```