

Project Report

Introduction:

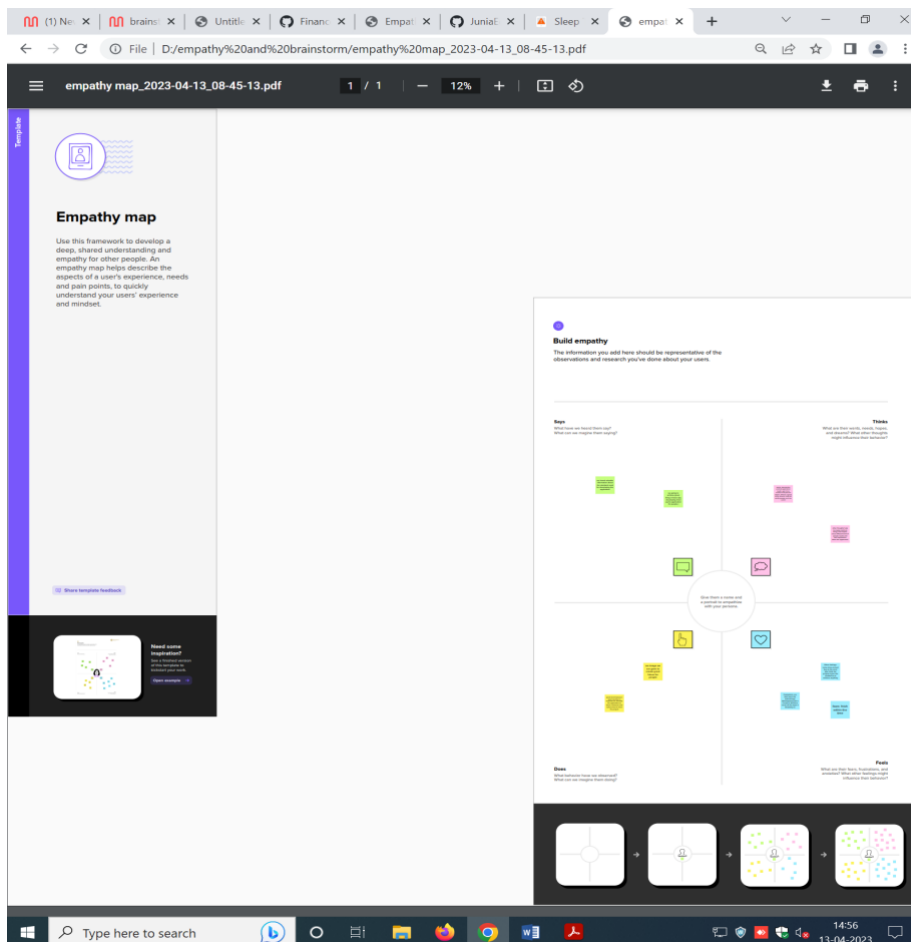
Irregular sleeping patterns are a common problem. This web app will fulfil the user's needs in tracking their sleeping patterns, including duration and timings. You will create a web app to track three parameters: sleep time, wake up time, and sleep duration. Users can add, edit, or remove any sleep entries. Irregular sleeping patterns are a common problem. This application will fulfil the user's needs in tracking their sleeping patterns, including duration and timings. You will create a web app to track three parameters: sleep time, wake up time, and sleep duration. Users can add, edit, or remove any sleep entries.

Purpose:

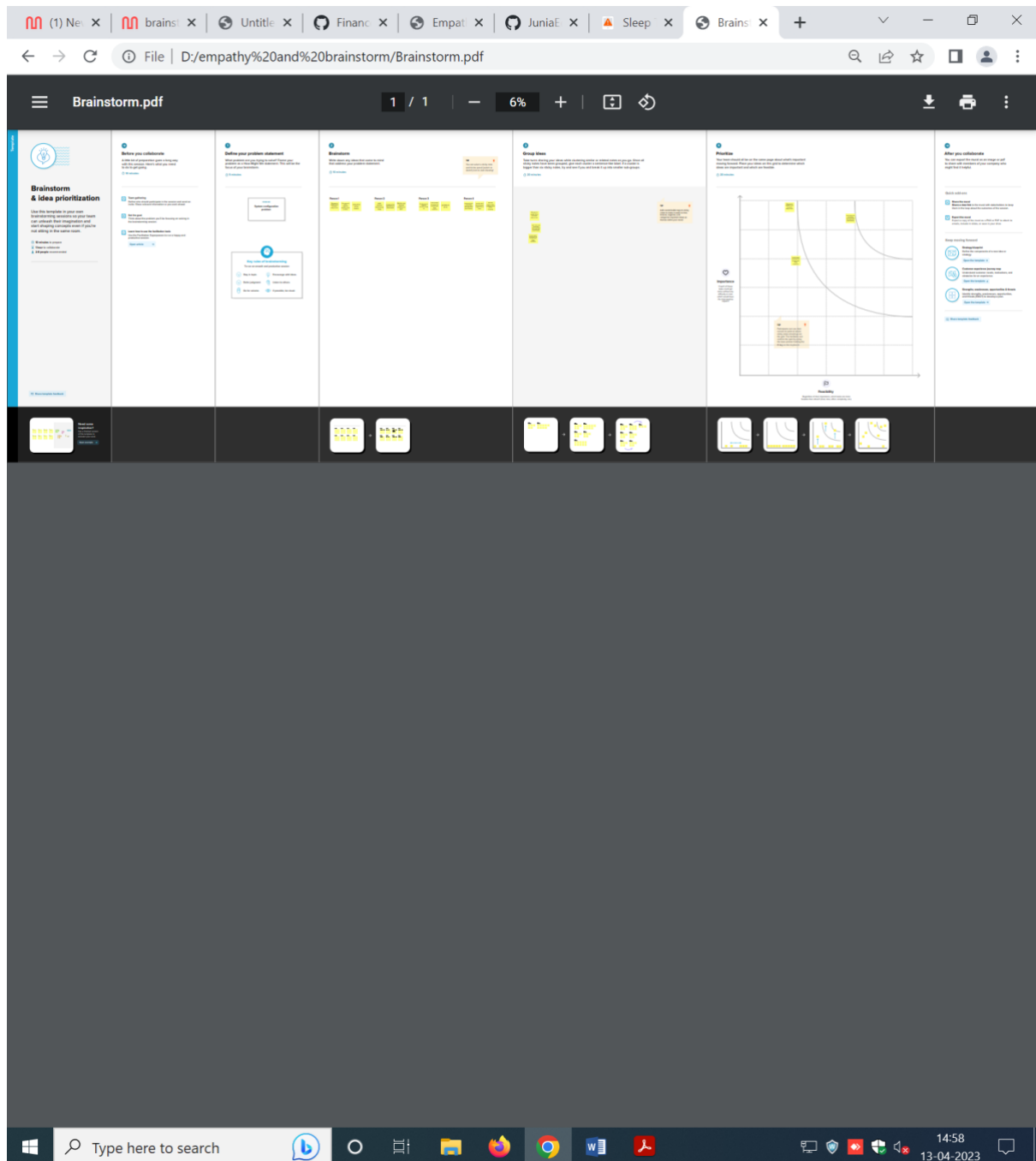
find whatever you wish to know about your sleep patterns, check out your snoring and dream talks, and customize the smart alarm to relieve sleep issues and aid your sleep.

Problem Definition & Design Thinking:

Empathy Map:

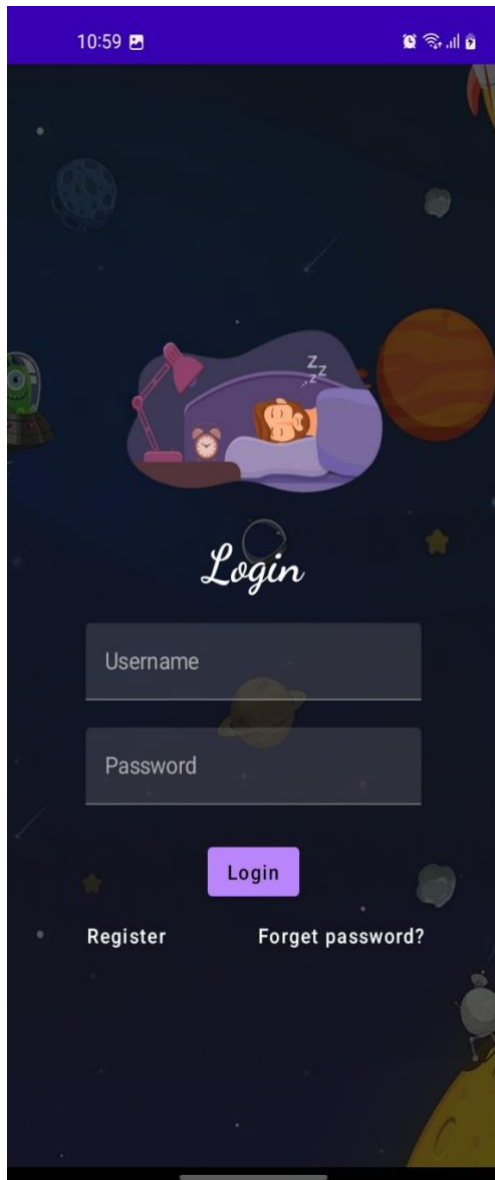


Ideation & Brainstorming map:

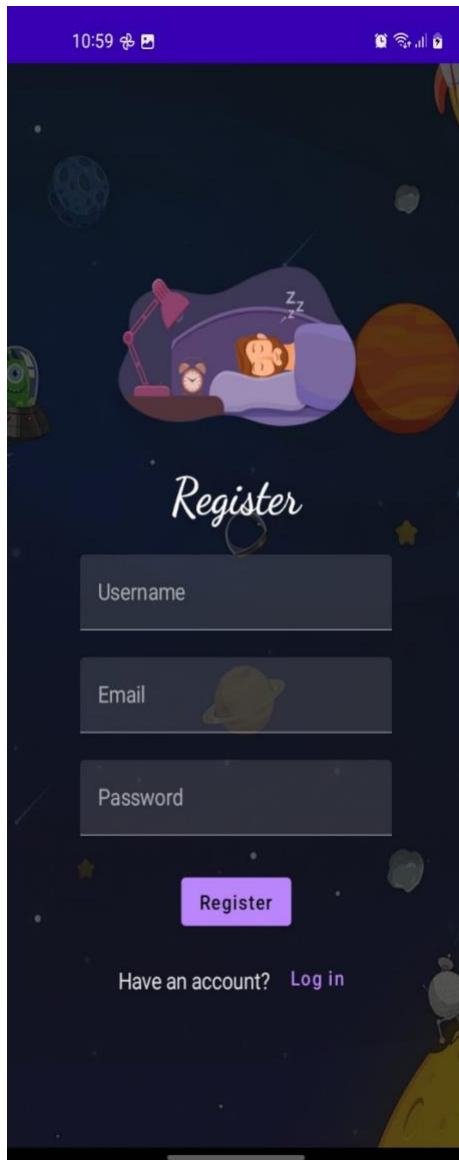


Result:

Login page:



Register page:



10:59

Register

Username

Email

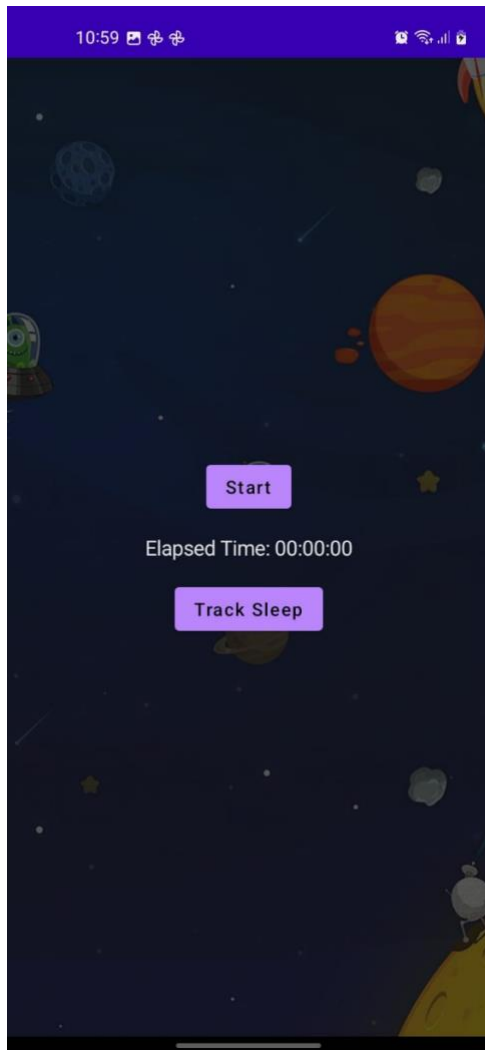
Password

Register

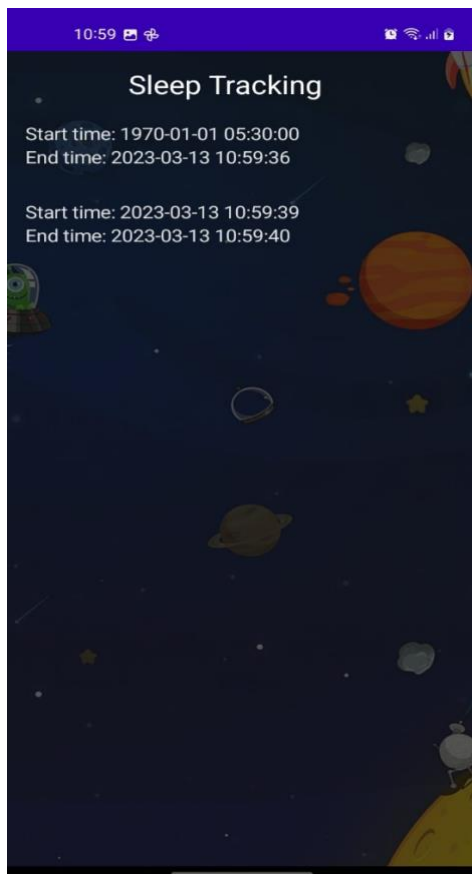
Have an account? [Log in](#)

The image shows a mobile app registration screen with a dark blue space-themed background. At the top, there's a status bar with the time 10:59 and various icons. Below the status bar is a large illustration of a person sleeping in a space pod with a lamp and a planet in the background. The word 'Register' is written in a cursive font. Below it are three input fields for 'Username', 'Email', and 'Password'. A purple 'Register' button is positioned below the fields. At the bottom, there's a link 'Log in' next to the text 'Have an account?'.

Main page:



Track Sleep page:



Advantage:

1. **You can manage what you measure.** [Dr. Kelly Baron](#), director of the University of Utah's behavioural 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 night time behaviours impact your sleep quality and duration, you can make changes to help you sleep better.
2. **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.

3. **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.
4. **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:

1. **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.
2. **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.
3. **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.
4. **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.
5. **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 night-time anxiety, which can make it harder for you to sleep.

Application:

It is used in medical field and it improve the patient health.

All using these to self-aware about their health.

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.

FUTURE SCOPE:

- We goes to update UI for all people easily to use.
Give personal doctor suggestion based on mind condition.

APPENDIX:

Adding dependency in build.gradle (Module: app)file in project


```
plugins
{
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.sleeptracking'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.sleeptracking"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = '1.8'
    }

    buildFeatures {
        compose true
    }
}
```

```

    }

    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }

    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"


    // Adding Room dependencies
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
}

```

Adding App Database file:

```
package com.example.sleeptracking
```

```

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

```

```

        @Database(entities = [TimeLogDao::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
                    }
                }
            }
        }
    }
}

```

Adding Expense Login Activity file:

packagcom.example.sleeptracking

```

import android.content.Context9
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.sleeptracking.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,
            LoginActivity::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,
LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}

```

Adding Expense Main Activity file:

Packagecom.example.sleeptracking

```

import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
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.TimeLogDatabaseHelper
import
com.example.sleeptracking.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,

```

```

        MainActivity::class.java
    )
    ) }) {
        Text(text = "Track Sleep")
    }

}

}

private fun startTrackActivity(context: Context) {
    val intent = Intent(context,
MainActivity::class.java)
    ContextCompat.startActivity(context, intent,
null)
}
@RequiresApi(Build.VERSION_CODES.N)
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)
}

```

Adding Expense Registration Activity file:

```

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 RegistrationActivity2 : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        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)
}

```

Adding Expense Time Database Helper file:

```

package
com.example.projectone

```

```

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import java.util.*

class TimeLogDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
    companion object {
        private const val DATABASE_NAME = "timelog.db"
        private const val DATABASE_VERSION = 1
        const val TABLE_NAME = "time_logs"
        private const val COLUMN_ID = "id"
    }
}

```

```

        const val COLUMN_START_TIME = "start_time"
        const val COLUMN_END_TIME = "end_time"

        // Database creation SQL statement
        private const val DATABASE_CREATE =
            "create table $TABLE_NAME ($COLUMN_ID integer
primary key autoincrement, " +
                "$COLUMN_START_TIME integer not null,
$COLUMN_END_TIME integer);"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        db?.execSQL(DATABASE_CREATE)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion:
Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    // function to add a new time log to the database
    fun addTimeLog(startTime: Long, endTime: Long) {
        val values = ContentValues()
        values.put(COLUMN_START_TIME, startTime)
        values.put(COLUMN_END_TIME, endTime)
        writableDatabase.insert(TABLE_NAME, null, values)
    }

    // function to get all time logs from the database
    @SuppressWarnings("Range")
    fun getTimeLogs(): List<TimeLog> {
        val timeLogs = mutableListOf<TimeLog>()
        val cursor = readableDatabase.rawQuery("select *
from $TABLE_NAME", null)
        cursor.moveToFirst()
        while (!cursor.isAfterLast) {
            val id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_START_TIME))
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_END_TIME))
            timeLogs.add(TimeLog(id, startTime, endTime))
            cursor.moveToNext()
        }
    }

```

```

        cursor.close()
        return timeLogs
    }

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

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

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

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

```

Adding Expense Time Log file:

Packagecom.example.projectone

```

import androidx.room.Entity
import androidx.room.PrimaryKey
import java.sql.Date

@Entity(tableName = "TimeLog")
data class TimeLog(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val startTime: Date,
    val stopTime: Date
)

```

Adding Expense Time Log Dao file:

packagecom.example.sleeptracking

```
import androidx.room.Dao
import androidx.room.Insert
import com.example.projectone.TimeLog
```

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

Adding Expense Track Activity file:

Packagecom.example.sleeptracking

```
import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
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.TimeLogDatabaseHelper
import com.example.sleeptracking.ui.theme.ProjectOneTheme
import java.util.*
```

```
class TrackActivity : ComponentActivity() {
```

```

private lateinit var databaseHelper: TimeLogDatabaseHelper

@RequiresApi(Build.VERSION_CODES.N)
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    databaseHelper = TimeLogDatabaseHelper(this)
    setContentView {
        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)
            }
        }
    }
}

```

```

@RequiresApi(Build.VERSION_CODES.N)
@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) }}" )
                    }
                }
            }
        }
    }

    @RequiresApi(Build.VERSION_CODES.N)
    private fun formatDateTime(timestamp: Long): String {
        val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
        return dateFormat.format(Date(timestamp))
    }
}

```

Adding Expense User Data class file:

```

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?,

)

```

Adding Expense User Database file:

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

```
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 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
}

```

```

    }

```

Adding Expense User Database Helper file:

```

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 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
}
}

```

Adding Expense User Deo file:

```
package
com.example.sleeptracking

import androidx.room.*

interface UserDeo {

    @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)
    }
}
```