



A sleep tracking app for a better night's rest

RUFFIN EUNOGIA D

710422243043

DESCRIPTION

- 1. Sleep Tracking: Monitor sleep duration, stages (light, deep, REM), and sleep cycles.
- 2. Sleep Score: Get a daily sleep score based on sleep quality, duration, and consistency.
- 3. Sleep Stage Tracking: Visualize sleep stages in real-time, identifying light, deep, and REM sleep.
- 4. Smart Alarms: Wake up during light sleep phases, feeling refreshed and energized.
- 5. Sleep Diary: Log sleep-related events, such as coffee consumption, exercise, or stress levels.
- 6. Personalized Recommendations: Receive tailored advice on sleep schedule, relaxation techniques, and sleep environment optimization.
- 7. Sleep Goals: Set and track sleep goals, monitoring progress over time.
- 8. Mood Tracking: Monitor emotions and energy levels, correlating them with sleep quality.
- 9. Relaxation Techniques: Access guided meditations, breathing exercises, and soothing sounds.
- 10. Integrations: Connect with popular health and fitness apps (e.g., Fitbit, Apple Health).

Main Activity:

```
package com.example.projectoneimport androidx.test.platform.app.  
InstrumentationRegistryimport androidx.test.ext.junit.runners.  
AndroidJUnit4import org.junit.Testimport org.  
junit.runner.RunWithimport org.  
junit.Assert.  
*/** * Instrumented test, which will execute on an Android device.  
* * See [testing documentation](http://d.android.com/tools/testing).  
*/@RunWith(AndroidJUnit4::class)  
class ExampleInstrumentedTest  
{  
    @Test fun useApplicationContext() {  
        // Context of the app under test.  
        val applicationContext = InstrumentationRegistry.getInstrumentation().  
            targetContext  
        assertEquals("com.example.projectone",  
            applicationContext.packageName)  {}
```

```
<?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>
```

```
<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" />
```

```
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,  
    MainPage::class.java  
)  
)  
//onLoginSuccess()  
}  
  
if (user != null && user.password == "admin") {  
    error = "Successfully log in"  
    context.startActivity(  
        Intent(  
            context,  
            AdminActivity::class.java  
        )  
    )  
}  
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,
            MainActivity::class.java
        )
    })
}
{
    Text(color = Color.White, text = "Sign up")
}
TextButton(onClick = {
})
{
    Spacer(modifier = Modifier.width(60.dp))
```

Text(color = Color.White,text = "Forget password?")

```
    }
}

}

private fun startMainPage(context: Context) {
    val intent = Intent(context, MainPage::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

MAIN PAGE.KT

```
package com.example.snackordering

import android.annotation.SuppressLint
import android.content.Context
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.DrawableRes
```

```
import androidx.annotation.StringRes
```

```
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons(Icons)
import androidx.compose.material.icons.filled.-
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Text
import androidx.compose.ui.unit.dp
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
compose.
```

```
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat.startActivity
import com.example.snackordering.ui.theme.SnackOrderingTheme

import android.content.Intent as Intent1

class MainPage : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SnackOrderingTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                )
            }
        }
    }
}
```

```
FinalView(this)
    val context = LocalContext.current
    //PopularFoodColumn(context)
}
}
}
}

@Composable
fun TopPart() {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color(0xffcecef0)), Arrangement.SpaceBetween
    ) {
        Icon(
            imageVector = Icons.Default.Add, contentDescription = "Menu Icon",

```

Modifier

```
.clip(CircleShape)
.size(40.dp),
tint = Color.Black,
)
Column(horizontalAlignment = Alignment.CenterHorizontally) {
    Text(text = "Location", style = MaterialTheme.typography.subtitle1, color =
Color.Black)
    Row {
        Icon(
            imageVector = Icons.Default.LocationOn,
            contentDescription = "Location",
            tint = Color.Red,
        )
        Text(text = "Accra" , color = Color.Black)
    }
}
Icon(
    imageVector = Icons.Default.Notifications, contentDescription = "Notification
Icon",
```

Modifier

```
.size(45.dp),  
tint = Color.Black,
```

```
)
```

```
}
```

```
}
```

```
@Composable
```

```
fun CardPart() {
```

```
    Card(modifier = Modifier.size(width = 310.dp, height = 150.dp),
```

```
        RoundedCornerShape(20.dp)) {
```

```
        Row(modifier = Modifier.padding(10.dp), Arrangement.SpaceBetween) {
```

```
            Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {
```

```
                Text(text = "Get Special Discounts")
```

```
                Text(text = "up to 85%", style = MaterialTheme.typography.h5)
```

```
                Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) {
```

```
                    Text(text = "Claim voucher", color = MaterialTheme.colors.surface)
```

```
}
```

```
}
```

```
    Image(
```

```
        painter = painterResource(id = R.drawable.food_tip_im),
```

```
        contentDescription = "Food Image", Modifier.size(width = 100.dp, height =
```

200.dp)

)

```
)  
}  
}  
}  
  
@Composable  
fun PopularFood(  
    @DrawableRes drawable: Int,  
    @StringRes text1: Int,  
    context: Context  
) {  
    Card(  
        modifier = Modifier  
            .padding(top=20.dp, bottom = 20.dp, start = 65.dp)  
            .width(250.dp)  
    ) {
```

```
Column(  
    verticalArrangement = Arrangement.Top,  
    horizontalAlignment = Alignment.CenterHorizontally  
) {  
    Spacer(modifier = Modifier.padding(vertical = 5.dp))  
    Row(  
        modifier = Modifier  
            .fillMaxWidth(0.7f), Arrangement.End  
) {  
        Icon(  
            imageVector = Icons.Default.Star,  
            contentDescription = "Star Icon",  
            tint = Color.Yellow  
        )  
        Text(text = "4.3", fontWeight = FontWeight.Black)  
    }  
    Image(  
        painter = painterResource(id = drawable),  
        contentDescription = "Food Image",  
        contentScale = ContentScale.Crop,
```

```
modifier = Modifier
    .size(100.dp)
    .clip(CircleShape)
)

Text(text = stringResource(id = text1), fontWeight = FontWeight.Bold)
Row(modifier = Modifier.fillMaxWidth(0.7f), Arrangement.SpaceBetween) {
    /*TODO Implement Prices for each card*/
    Text(
        text = "$50",
        style = MaterialTheme.typography.h6,
        fontWeight = FontWeight.Bold,
        fontSize = 18.sp
    )

    IconButton(onClick = {
        //var no=FoodList.lastIndex;
        //Toast.
        val intent = Intent1(context, TargetActivity::class.java)
        context.startActivity(intent)
    })
}
```

```
} {  
    Icon(  
        imageVector = Icons.Default.ShoppingCart,  
        contentDescription = "shopping cart",  
    )  
}  
}  
}  
}  
}
```

```
private val FoodList = listOf(  
    R.drawable.sandwich to R.string.sandwich,  
    R.drawable.sandwich to R.string.burgers,  
    R.drawable.pack to R.string.pack,  
    R.drawable.pasta to R.string.pasta,
```

```
R.drawable.tequila to R.string.tequila,  
R.drawable.wine to R.string.wine,  
R.drawable.salad to R.string.salad,  
R.drawable.pop to R.string.popcorn  
).map { DrawableStringPair(it.first, it.second) }
```

```
private data class DrawableStringPair(  
    @DrawableRes val drawable: Int,  
    @StringRes val text1: Int  
)
```

```
@Composable  
fun App(context: Context) {  
  
    Column(  
        modifier = Modifier  
            .fillMaxSize()  
            .background(Color(0xffecef0))  
            .padding(10.dp),
```

```
verticalArrangement = Arrangement.Top,  
    horizontalAlignment = Alignment.CenterHorizontally  
) {  
    Surface(modifier = Modifier, elevation = 5.dp) {  
        TopPart()  
    }  
    Spacer(modifier = Modifier.padding(10.dp))  
    CardPart()  
  
    Spacer(modifier = Modifier.padding(10.dp))  
    Row(modifier = Modifier.fillMaxWidth(), Arrangement.SpaceBetween) {  
        Text(text = "Popular Food", style = MaterialTheme.typography.h5, color =  
Color.Black)  
        Text(text = "view all", style = MaterialTheme.typography.subtitle1, color =  
Color.Black)  
    }  
    Spacer(modifier = Modifier.padding(10.dp))  
    PopularFoodColumn(context) // <- call the function with parentheses  
}  
}
```

```
@Composable
fun PopularFoodColumn(context: Context) {

    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        content = {
            items(FoodList) { item ->
                PopularFood(context = context,drawable =
item.drawable, text1 = item.text1)
            }
        },
        verticalArrangement = Arrangement.spacedBy(16.dp))
}

@SuppressLint("UnusedMaterialScaffoldPaddingParameter")
```

```
@Composable
fun FinalView(mainPage: MainPage) {
    SnackOrderingTheme {
        Scaffold() {
            val context = LocalContext.current
            App(context)
        }
    }
}
```

ORDER.KT

```
package com.example.snackordering

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

@Entity(tableName = "order_table")
data class Order(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "address") val
```

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

```
String?,  
    @ColumnInfo(name = "address") val address:  
String?,  
)  
ORDERDAO.KT
```

```
package com.example.snackordering  
import androidx.room.*  
@Dao  
interface OrderDao {  
    @Query("SELECT * FROM order_table WHERE  
address= :address")  
    suspend fun getOrderByAddress(address: String): Order?  
  
    @Insert(onConflict = OnConflictStrategy.REPLACE)  
    suspend fun insertOrder(order: Order)  
  
    @Update  
    suspend fun updateOrder(order: Order)  
  
    @Delete
```

```
suspend fun deleteOrder(order: Order)  
{
```

ORDERDATABASE.KT

```
package com.example.snackordering

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

@Database(entities = [Order::class], version = 1)
abstract class OrderDatabase : RoomDatabase() {

    abstract fun orderDao(): OrderDao

    companion object {

        @Volatile
        private var instance: OrderDatabase? = null

        fun getDatabase(context: Context): OrderDatabase {
            return instance ?: synchronized(this) {
```

```
val newInstance = Room.databaseBuilder(
```

```
    context.applicationContext,  
        OrderDatabase::class.java,  
        "order_database"  
    ).build()  
    instance = newInstance  
    newInstance  
}  
}  
}  
}
```

ORDER DATABASE HELPER.KT

```
package com.example.snackordering  
  
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 OrderDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION){

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "OrderDatabase.db"

        private const val TABLE_NAME = "order_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_QUANTITY = "quantity"
        private const val COLUMN_ADDRESS = "address"
    }
}
```

```
override fun onCreate(db: SQLiteDatabase?) {  
    val createTable = "CREATE TABLE $TABLE_NAME (" +  
        "${COLUMN_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " +  
        "${COLUMN_QUANTITY} Text, " +  
        "${COLUMN_ADDRESS} TEXT " +  
        ")"  
  
    db?.execSQL(createTable)  
}
```

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

```
fun insertOrder(order: Order) {  
    val db = writableDatabase  
    val values = ContentValues()
```

```
values.put(COLUMN_QUANTITY, order.quantity)
values.put(COLUMN_ADDRESS, order.address)
db.insert(TABLE_NAME, null, values)
db.close()
}
```

```
@SuppressLint("Range")
fun getOrderByQuantity(quantity: String): Order? {
    val db = readableDatabase
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_QUANTITY = ?",
arrayOf(quantity))
    var order: Order? = null
    if (cursor.moveToFirst()) {
        order = Order(
            id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            quantity = cursor.getString(cursor.getColumnIndex(COLUMN_QUANTITY)),
            address = cursor.getString(cursor.getColumnIndex(COLUMN_ADDRESS)),
        )
    }
}
```

```
plugins { id 'com.android.application'  
    id 'org.jetbrains.kotlin.android'  
android { namespace 'com.example.projectone'  
    compileSdk 33 defaultConfig  
{  
    applicationId "com.example.projectone"  
    minSdk 24  
    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.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
"androidx.compose.ui:ui-tooling:$compose_ui_version"
debugImplementation "androidx.compose.ui:ui-test-manifest:
$compose_ui_version"}
```

OUTPUT:







