Android Application Development

A Sleep Tracking App for a Better Night's Rest

TEAM ID- NM2024TMID03586

Submitted by

Deepa C(Team Leader) -F2C7FBCAA941A5CA64B4547D86C1621A

Sowntharya S(Team Member) - 0B4618865A21A70ABCD4685B0F5876AB

Haridharshini S(Team Member) - 8C8C15B5D33FC12B6AA4E184F60A2104

Ranjana Devi E(Team Member)- C29D7F06C08ACF793364739564DF7E47

SEMESTER - V

B.E COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR-2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE

COIMBATORE-641046

NOVEMBER 2024

CONTENT

A sleep tracking app for a better night's rest

1 INTRODUCTION

1.1 OBJECTIVE

The objective of the Sleep Tracker project is to design and develop an intuitive mobile application that enables users to monitor, assess, and improve their sleep quality. The app aims to provide a straightforward interface for tracking sleep duration, rating sleep quality, and generating meaningful insights into the user's sleep patterns. By offering detailed, data-driven feedback and analysis, the Sleep Tracker app empowers users to establish healthier sleep habits, recognize factors impacting their rest, and ultimately achieve a more consistent and restful sleep. The project seeks to enable users to start a sleep timer at bedtime and stop it upon waking, capturing precise sleep duration and to allow users to rate their sleep quality each morning to reflect on their sleep experience and to store and display historical sleep data, providing insights into trends over days and weeks and to utilize a modern, responsive UI built with Jetpack Compose to enhance usability and accessibility and to provide users with sleep pattern analysis to support informed decisions about improving sleep hygiene and overall well-being. Ultimately, the Sleep Tracker app aims to promote better sleep habits, leading to enhanced mental and physical health for its users.

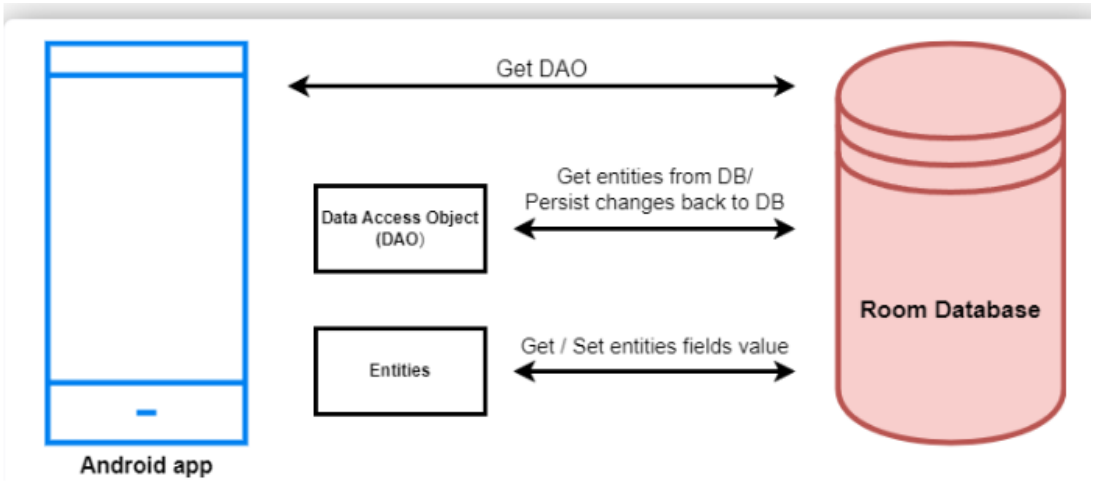
1.2 OVERVIEW

The Sleep Tracker app is a mobile application designed to help users monitor and improve their sleep quality. Built with Android Jetpack Compose, the app features a user-friendly interface for tracking sleep duration, rating sleep quality, and analyzing sleep patterns over time. By storing data locally with Room Database, the app provides secure access to users' sleep history, allowing them to identify trends and make informed decisions to enhance their rest.

Through simple tracking and valuable insights, the Sleep Tracker app promotes healthier sleep habits, contributing to improved well-being and productivity.

2 FUNCTIONALITIES

2.1 ARCHITECTURE



2.2 PROJECT WORKFLOW

The Sleep Tracker app workflow begins with the user starting a sleep timer before bed, which runs in the background. Upon waking, the user stops the timer and rates their sleep quality. This data (sleep duration and quality) is stored in Room Database. The View Model processes and updates the data, which is then displayed on the app's analysis screen, allowing users to view trends in their sleep patterns. The UI, built with Jetpack Compose, presents an intuitive interface for tracking, reviewing, and understanding sleep habits, enabling users to make informed adjustments for better rest.

3 PROJECT DESCRIPTION

3.1 PROBLEM STATEMENT

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.

3.2 SOLUTIONS

The Sleep Tracker app offers solutions to help users improve sleep quality. It provides simple sleep tracking by allowing users to start a timer before bed and rate their sleep each morning. The app securely stores data using Room Database, enabling insights into sleep patterns, such as average duration and quality trends. Built with Jetpack Compose, the app features a user-friendly, responsive interface that makes tracking and reviewing sleep habits easy and engaging. By promoting better sleep awareness and privacy-focused data storage, the app helps users establish healthier sleep routines for improved well-being.

4. SYSTEM REQUIREMETS:

4.1 SOFTWARE REQUIREMENTS

- **Operating System:** Windows 10, macOS, or Linux
- **Android Studio:** Latest stable version with Jetpack Compose support
- **Programming Language:** Kotlin
- **Database:** Room Database (for storing sleep data locally)
- **Development Kit:** Android SDK 23 (Marshmallow) or higher
- **Build Tool:** Gradle 7.0 or higher

4.2 HARDWARE REQUIREMENTS

- **Processor:** Minimum Intel i3 or AMD equivalent (i5/i7 recommended for smoother performance)
- **RAM:** 8 GB minimum (16 GB recommended)
- **Storage:** At least 4 GB of free space for Android Studio and Android SDK
- **Screen Resolution:** 1280 x 800 or higher (for better layout in Android Studio)

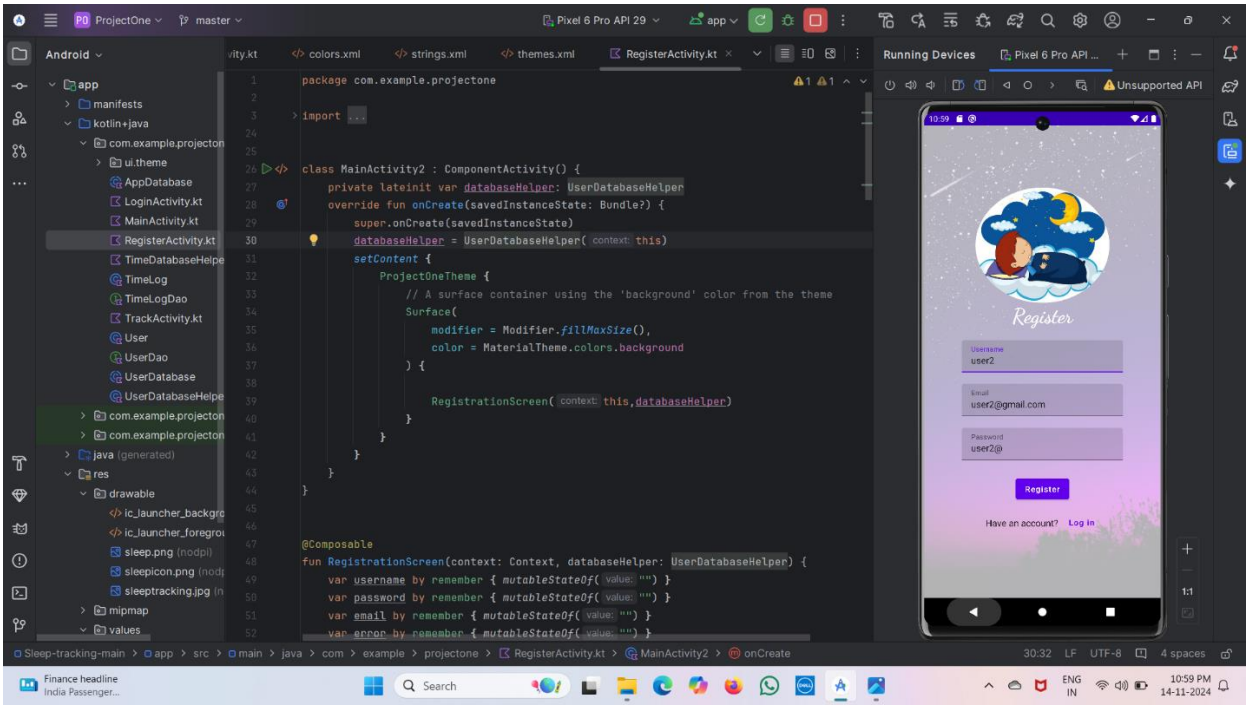
Android Device or Emulator: Android 6.0 (Marshmallow) or higher for testing

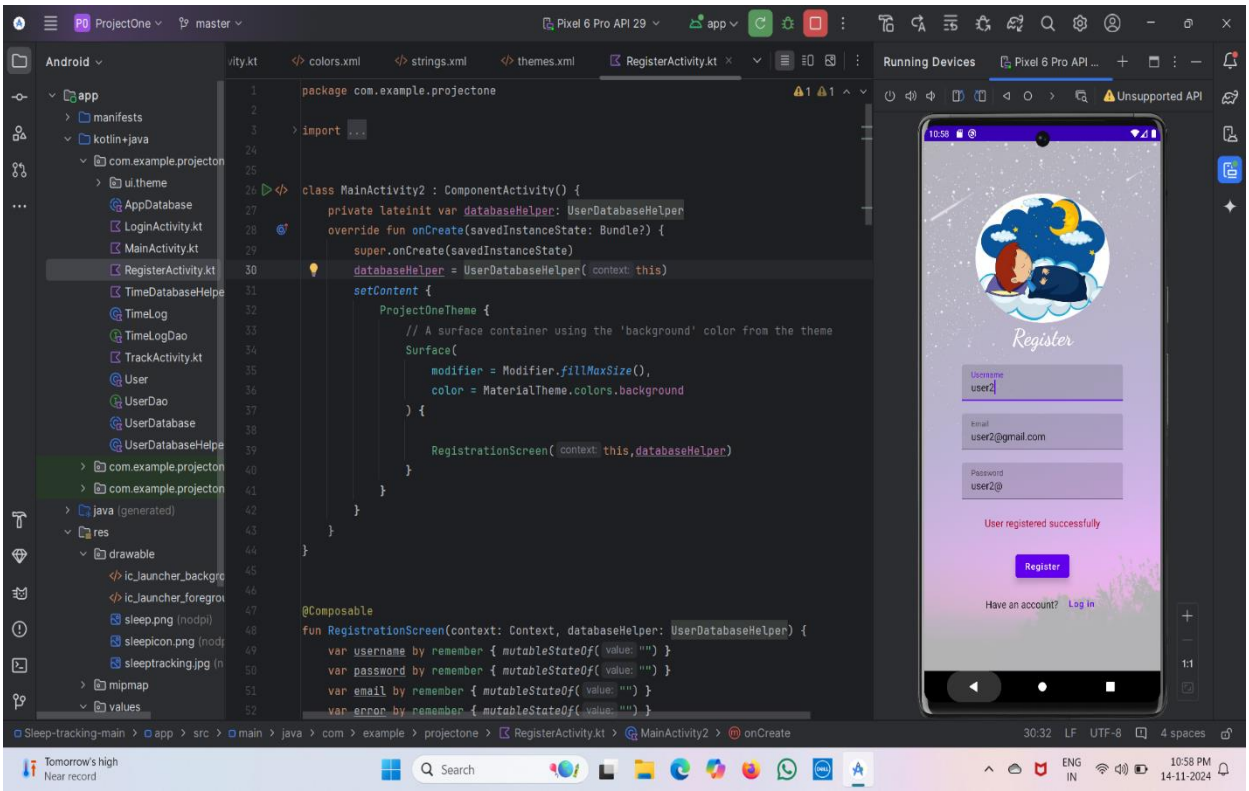
5 TOOLS AND VERSION

1. **Android Studio:** Arctic Fox (2020.3.1) or higher, with support for Jetpack Composes
2. **Kotlin Version:** 1.5.0 or higher
3. **Jetpack Compose:** 1.0.0 or higher (recommended to use the latest stable release)
4. **Gradle Version:** 7.0 or higher
5. **Room Database Library:** Version 2.4.0 or higher
6. **Coroutines:** Version 1.5.0 or higher for asynchronous tasks
7. **Navigation Component (Compose):** Version 2.4.0-alpha or higher for navigating between screens

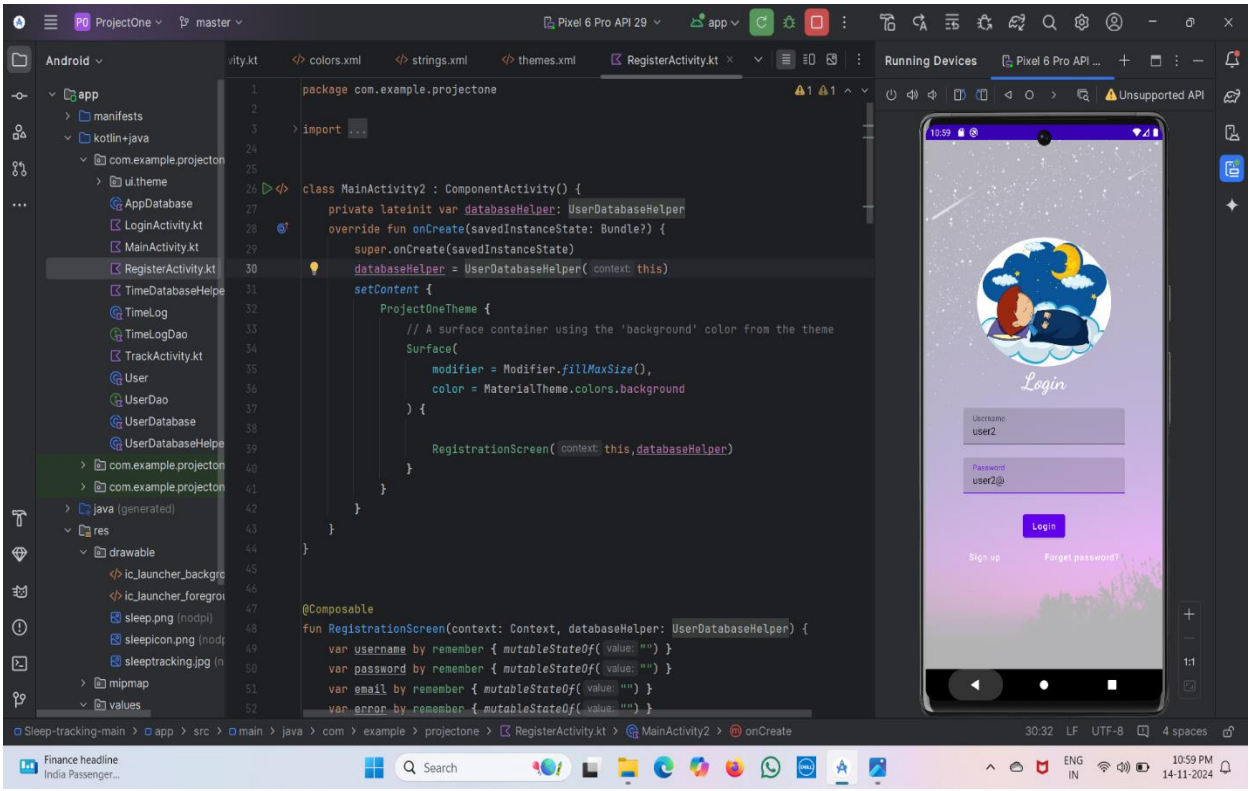
6 OUTPUT SCREENSHOTS

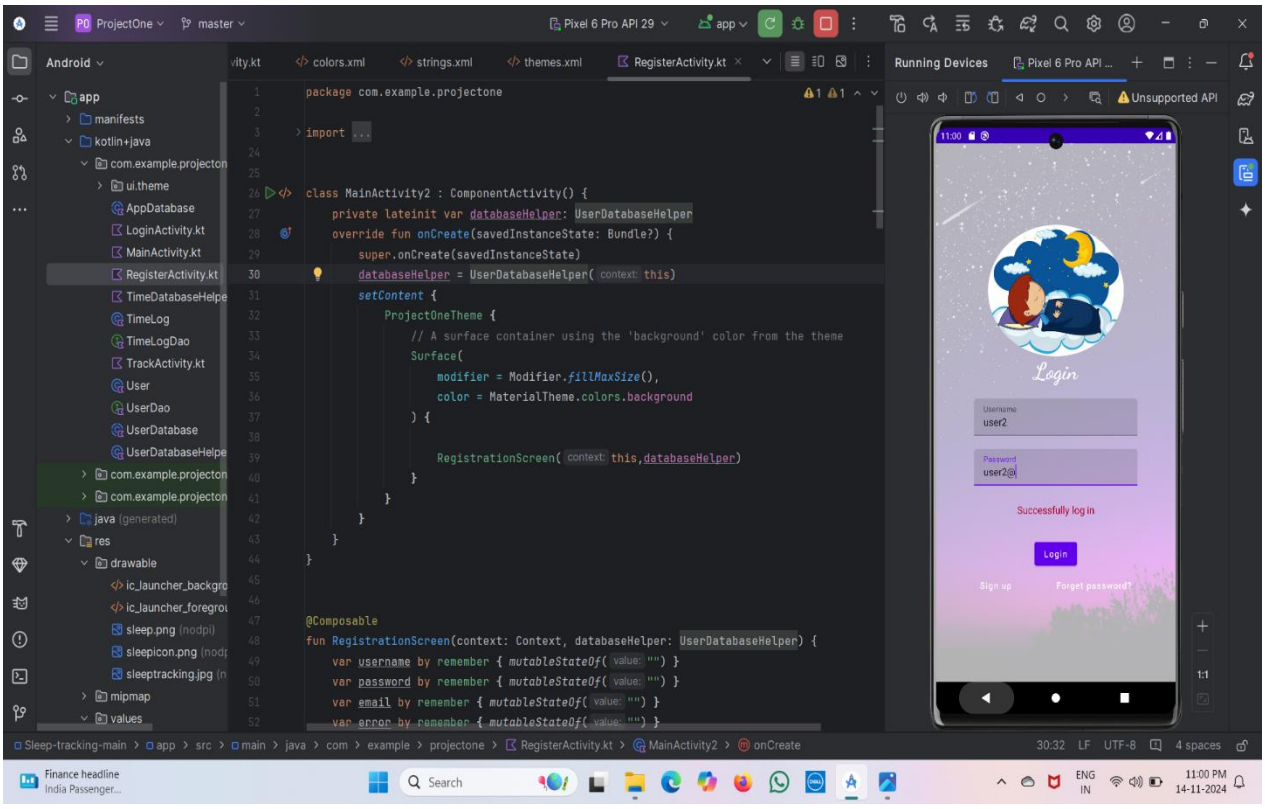
REGISTER PAGE



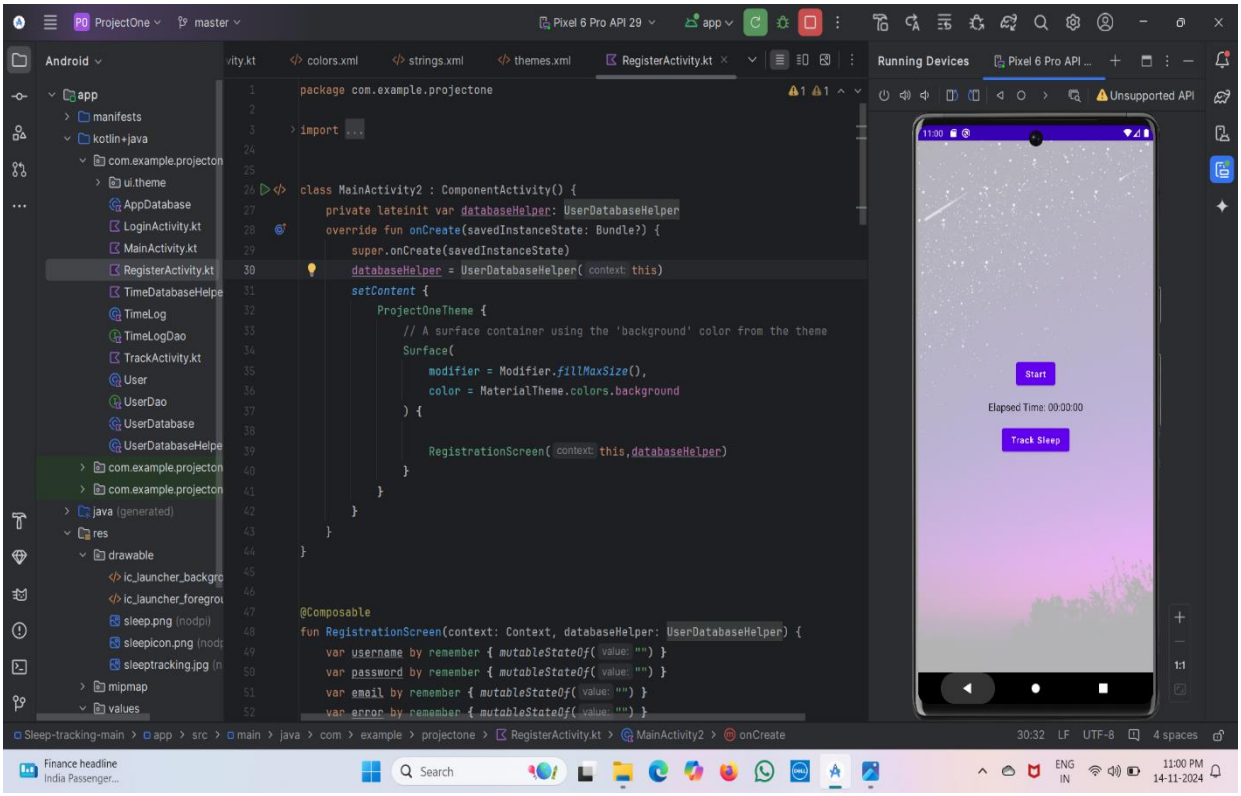


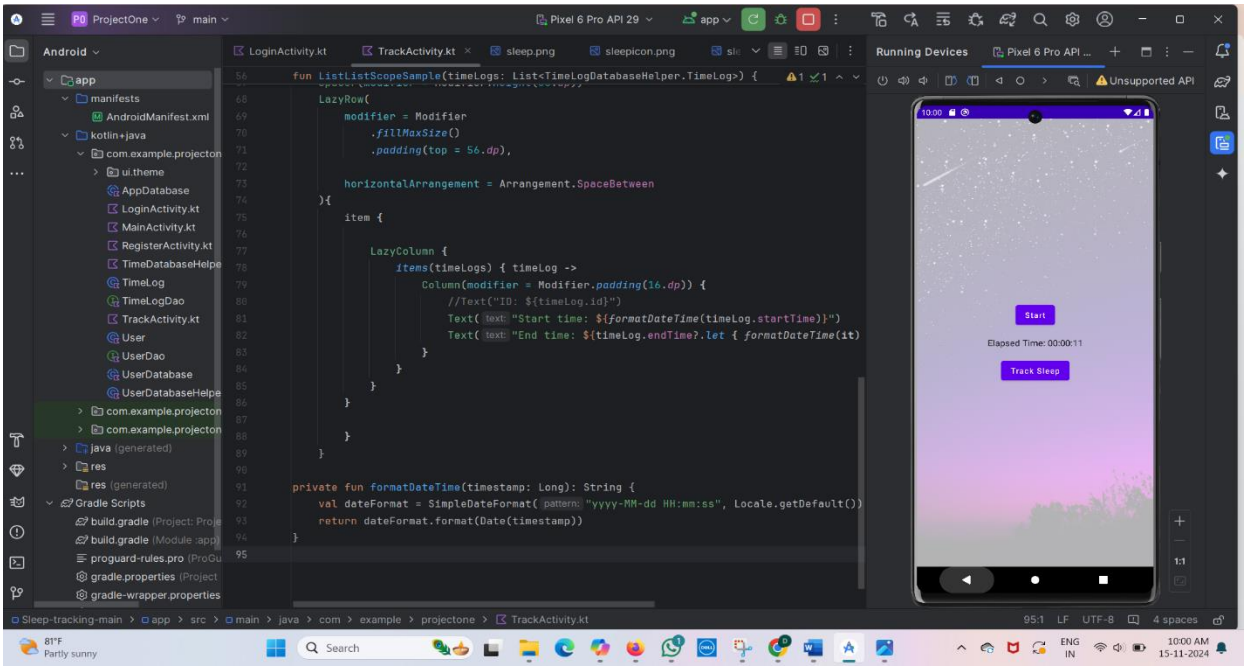
LOGIN PAGE


ProjectOne
master
Pixel 6 Pro API 29
app
Android
RegisterActivity.kt
Running Devices
Pixel 6 Pro API ...
Unsupported API
Login
user2
user2@
Sign up
Forget password?
Sleep-tracking-main
app
src
main
java
com
example
projectone
RegisterActivity.kt
MainActivity2
onCreate
30:32
LF
UTF-8
4 spaces
Finance headline
India Passenger...
Search
ENG
IN
10:59 PM
14-11-2024

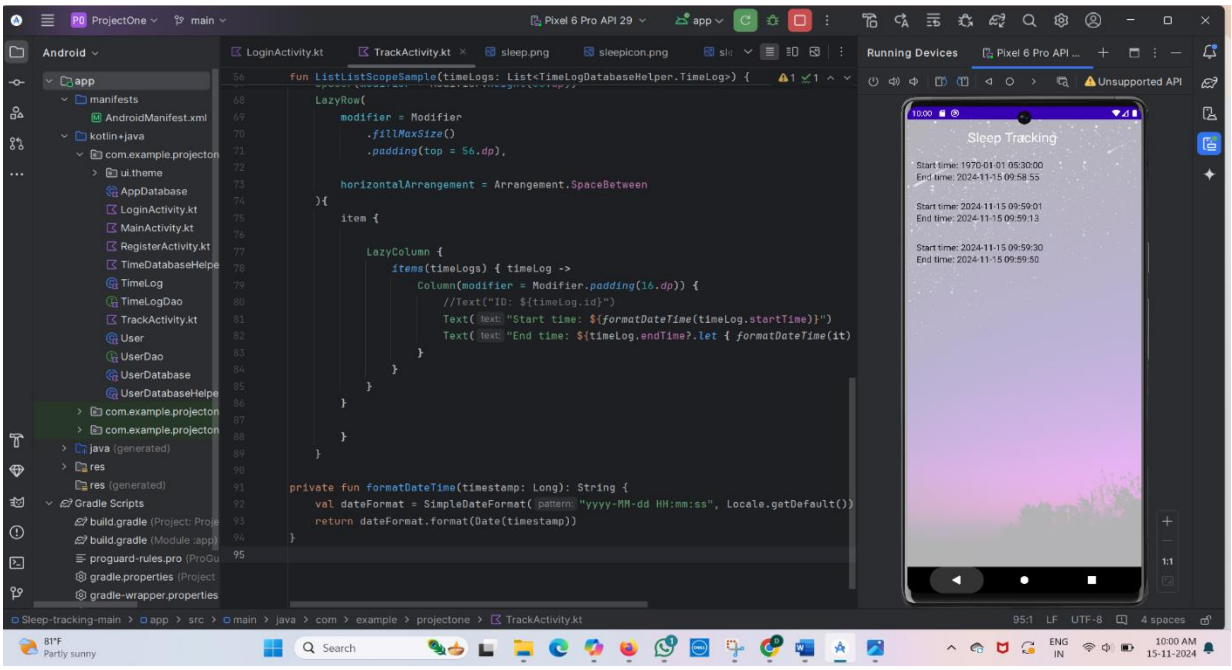

ProjectOne
master
Pixel 6 Pro API 29
app
Android
RegisterActivity.kt
Running Devices
Pixel 6 Pro API ...
Unsupported API
Login
user2
user2@
Successfully log in
Sign up
Forget password?
Sleep-tracking-main
app
src
main
java
com
example
projectone
RegisterActivity.kt
MainActivity2
onCreate
30:32
LF
UTF-8
4 spaces
Finance headline
India Passenger...
Search
ENG
IN
11:00 PM
14-11-2024

START TIME AND STOP TIME





TRACKED TIME



7 FUTURE SCOPE

The Sleep Tracker app has several potential areas for future enhancement. Future scope includes integrating with wearable devices (e.g., smartwatches) to provide more accurate sleep data, such as heart rate and sleep cycle analysis. Additional features could include personalized sleep improvement recommendations based on collected data, machine learning algorithms for predicting sleep patterns, and integration with voice assistants for hands-free operation. Furthermore, adding a smart alarm that wakes users during light sleep could improve the user experience, helping them wake up feeling more refreshed.

8 DEMO VIDEO LINK

https://drive.google.com/file/d/12N4PWiF-LsA1NPPkNNJs3jHBb6Z_-vGQ/view?usp=sharing

9 CONCLUSION

The **Sleep Tracker** app successfully fulfills its objective of helping users monitor and improve their sleep quality through a user-friendly interface and data-driven insights. By combining real-time sleep tracking, quality ratings, and historical analysis, the app provides users with a comprehensive view of their sleep patterns, enabling them to make informed decisions to enhance their rest. Leveraging Jetpack Compose for a modern and responsive UI, the app ensures an intuitive experience, making it accessible to a wide range of users.

This project not only highlights the effective use of Android development tools like Room Database and MVVM architecture but also emphasizes the importance of healthy sleep habits. The Sleep Tracker app offers a meaningful tool to promote better sleep hygiene, contributing to users' overall well-being and productivity. Moving forward, the app can be enhanced with additional features such as sleep cycle tracking, personalized recommendations, and integration with wearable devices, further increasing its value as a health-focused application.

10 APPENDIX

SOURCE CODE

LoginActivity.kt

```
package com.example.projectone

import android.content.Context
```

```
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.projectone.ui.theme.ProjectOneTheme
class 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,
        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

```
package com.example.projectone
import android.content.Context
import android.content.Intent
```

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.projectone.ui.theme.ProjectOneTheme
class MainActivity2 : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
```

```
            ProjectOneTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    RegistrationScreen(this,databaseHelper)
                }
            }
        }
    }
}
@Composable
fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
```

```
)
Column(
    modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    Image(
        painter = painterResource(id = R.drawable.sleep),
        contentDescription = "",

        modifier = imageModifier
            .width(260.dp)
            .height(200.dp)
    )
    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,
        color = Color.White,
        text = "Register"
    )

    Spacer(modifier = Modifier.height(10.dp))
    TextField(
        value = username,
```

```
    onValueChange = { username = it },
    label = { Text("Username") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)

)

TextField(
    value = email,
    onValueChange = { email = it },
    label = { Text("Email") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)

TextField(
    value = password,
    onValueChange = { password = it },
    label = { Text("Password") },
    modifier = Modifier
        .padding(10.dp)
        .width(280.dp)
)
```

```
if (error.isNotEmpty()) {
    Text(
        text = error,
        color = MaterialTheme.colors.error,
        modifier = Modifier.padding(vertical = 16.dp)
    )
}

Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {
            val user = User(
                id = null,
                firstName = username,
                lastName = null,
                email = email,
                password = password
            )
            databaseHelper.insertUser(user)
            error = "User registered successfully"
            // Start LoginActivity using the current context
            context.startActivity(
                Intent(
```

```
                context,
                LoginActivity::class.java
            )
        )

    } else {
        error = "Please fill all fields"
    }
},
modifier = Modifier.padding(top = 16.dp)
) {
Text(text = "Register")
}
Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))

Row() {
Text(
    modifier = Modifier.padding(top = 14.dp), text = "Have an account?"
)
TextButton(onClick = {

})

{
```

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

TimeDatabaseHelper.kt

```
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
@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"
        }
    }
}
```

TimeLog.kt

```
package com.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
)
```

TimeLogDao.kt

```
package com.example.projectone

import androidx.room.Dao
import androidx.room.Insert

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

}
```

TrackActivity.kt

```
package com.example.projectone

import android.icu.text.SimpleDateFormat
```

```
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.projectone.ui.theme.ProjectOneTheme
import java.util.*

class TrackActivity : ComponentActivity() {
```

```
    private lateinit var databaseHelper: TimeLogDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        databaseHelper = TimeLogDatabaseHelper(this)
        setContent {
            ProjectOneTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    //ListListScopeSample(timeLogs)

                    val data=databaseHelper.getTimeLogs();
                    Log.d("Sandeep" ,data.toString())
                    val timeLogs = databaseHelper.getTimeLogs()
                    ListListScopeSample(timeLogs)
                }
            }
        }
    }
}
```

```
@Composable
fun ListListScopeSample(timeLogs: List<TimeLogDatabaseHelper.TimeLog>) {
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
            .alpha(0.3F),
    )

    Text(text = "Sleep Tracking", modifier = Modifier.padding(top = 16.dp, start = 106.dp ), color = Color.White, fontSize = 24.sp)
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 56.dp),

        horizontalArrangement = Arrangement.SpaceBetween
    ){
        item {

            LazyColumn {
                items(timeLogs) { timeLog ->
```

```
                Column(modifier = Modifier.padding(16.dp)) {
                    //Text("ID: ${timeLog.id}")
                    Text("Start time: ${formatDateTime(timeLog.startTime)}")
                    Text("End time: ${timeLog.endTime?.let { formatDateTime(it) }}")
                }
            }
        }
    }

    }
}

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

MainActivity.kt

```
package com.example.projectone

import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Bundle
import androidx.activity.ComponentActivity
```

```
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Button
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import com.example.projectone.ui.theme.ProjectOneTheme
import java.util.*

class MainActivity : ComponentActivity() {

    private lateinit var databaseHelper: TimeLogDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = TimeLogDatabaseHelper(this)
```

```
        databaseHelper.deleteAllData()
        setContent {
            ProjectOneTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    MyScreen(this,databaseHelper)
                }
            }
        }
    }
}
@Composable
fun MyScreen(context: Context, databaseHelper: TimeLogDatabaseHelper) {
    var startTime by remember { mutableStateOf(0L) }
    var elapsedTime by remember { mutableStateOf(0L) }
    var isRunning by remember { mutableStateOf(false) }
    val imageModifier = Modifier
    Image(
        painterResource(id = R.drawable.sleeptracking),
        contentScale = ContentScale.FillHeight,
        contentDescription = "",
        modifier = imageModifier
```

```
        .alpha(0.3F),
)

Column(
    modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    if (!isRunning) {
        Button(onClick = {
            startTime = System.currentTimeMillis()
            isRunning = true
        }) {
            Text("Start")
            //databaseHelper.addTimeLog(startTime)
        }
    } else {
        Button(onClick = {
            elapsedTime = System.currentTimeMillis()
            isRunning = false
        }) {
            Text("Stop")
            databaseHelper.addTimeLog(elapsedTime,startTime)
        }
    }
```

```
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Elapsed Time: ${formatTime(elapsedTime - startTime)}")



        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = { context.startActivity(
            Intent(
                context,
                TrackActivity::class.java
            )
        ) }) {
            Text(text = "Track Sleep")
        }

    }

}

private fun startTrackActivity(context: Context) {
    val intent = Intent(context, TrackActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
fun getCurrentDateTime(): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())
    val currentTime = System.currentTimeMillis()
```

```
    return dateFormat.format(Date(currentTime))
}

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

AppDatabase.kt

```
package com.example.projectone

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
            }
        }
    }
}
```

User.kt

```
package com.example.projectone

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

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

    )
```

UserDao.kt

```
package com.example.projectone

import androidx.room.*

@Dao
interface UserDao {

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

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)
```

```
    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

UserDatabase.kt

```
package com.example.projectone

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
            }
        }
    }
}
```

UserDatabaseHelper.kt

```
package com.example.projectone

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

class UserDatabaseHelper(context: Context) :
```

```
SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

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

    private const val TABLE_NAME = "user_table"
    private const val COLUMN_ID = "id"
    private const val COLUMN_FIRST_NAME = "first_name"
    private const val COLUMN_LAST_NAME = "last_name"
    private const val COLUMN_EMAIL = "email"
    private const val COLUMN_PASSWORD = "password"
}

override fun onCreate(db: SQLiteDatabase?) {
    val createTable = "CREATE TABLE $TABLE_NAME (" +
            "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$COLUMN_LAST_NAME TEXT, " +
            "$COLUMN_EMAIL TEXT, " +
            "$COLUMN_PASSWORD TEXT" +
            ")"

    db?.execSQL(createTable)
}
```

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

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @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
}
```

```
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.ProjectOne"
        tools:targetApi="31">
        <activity
            android:name=".TrackActivity"
            android:exported="false"
            android:label="@string/title_activity_track"
            android:theme="@style/Theme.ProjectOne" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
```

```
            android:label="@string/app_name"
            android:theme="@style/Theme.ProjectOne" />
        <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:label="RegisterActivity"
            android:theme="@style/Theme.ProjectOne" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ProjectOne">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>


</manifest>
```

build.gradle

```
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.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    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"
}
```