***PROJECT REPORT TEMPLATE***

1.INTRODUCTION

* 1. OVERVIEW

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

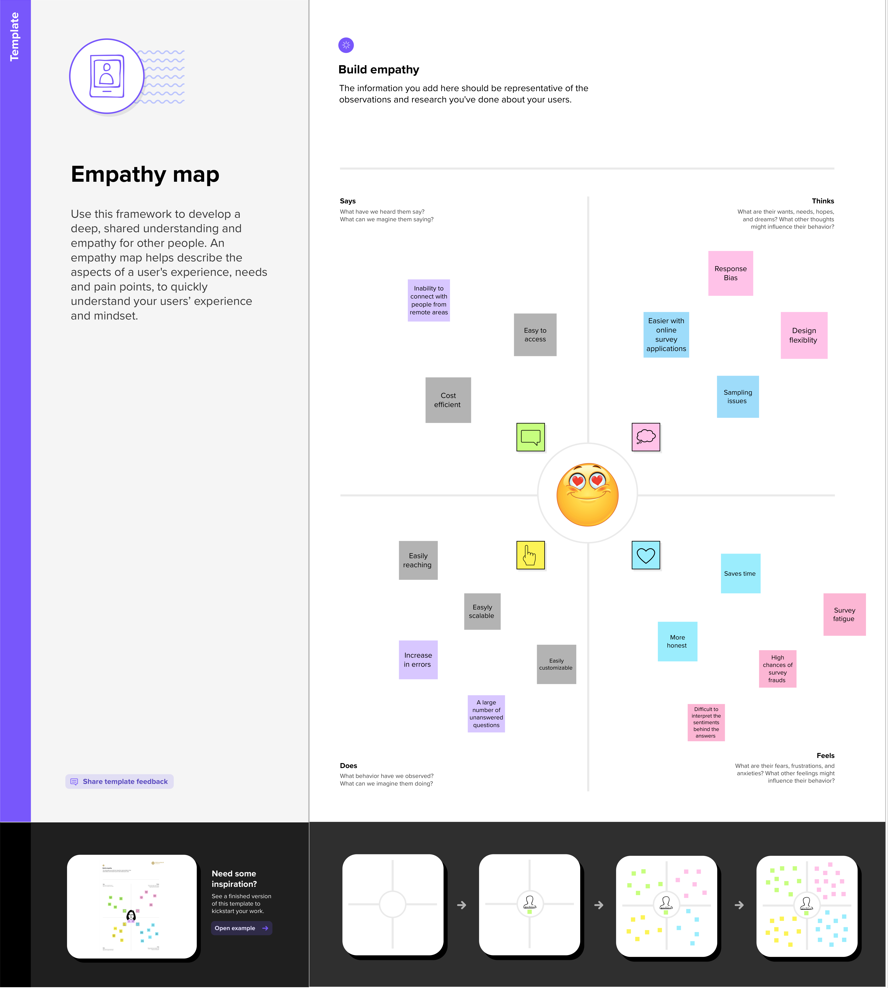
* 1. **PURPOSE**

Sleep tracker apps analyze your sounds,

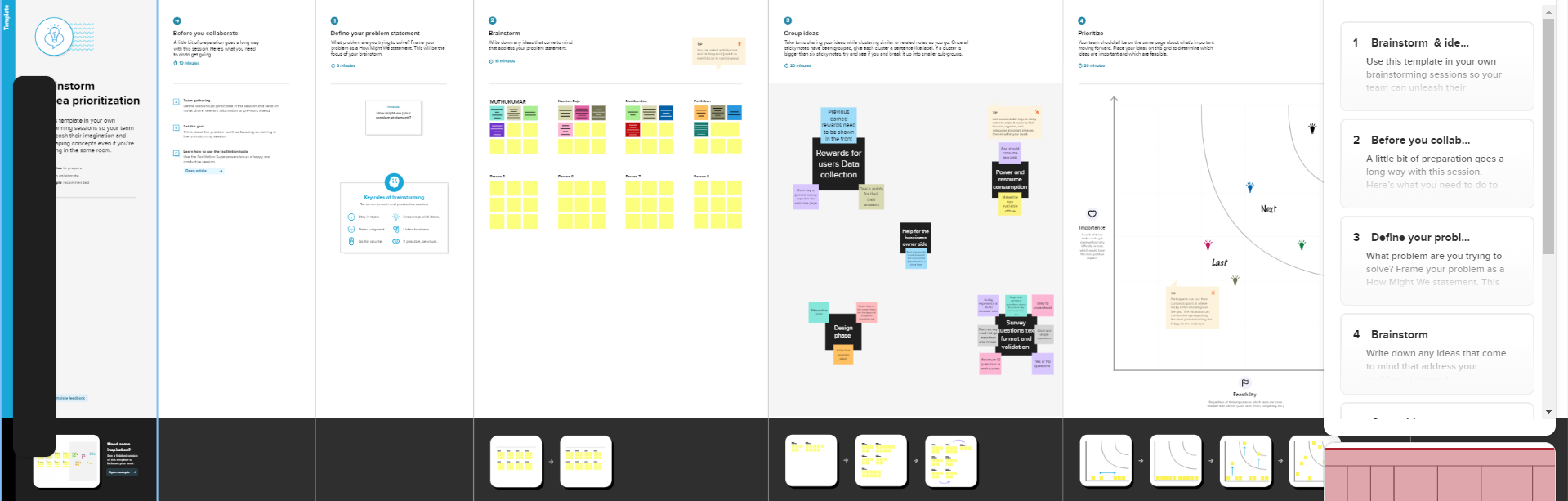
movement, and behaviors as you sleep to give you a clear snapshot of the duration and quality of your sleep. These apps can also help you determine how much time you spend in Rapid Eye Movement (REM) sleep and how many times you are disturbed throughout the night.

2.PROBLEM DEFINITION&DESIGN THINKING

2.1Empathy Map



2.2 Ideation & Brainstorming Map



3.RESULT

|  |  |
| --- | --- |
|  |  |

4. ADVANTAGE & DIS-ADVANTAGE SLEEP TRACKING OF SLEEP TRACKING APP:

ADVANTAGE :

### 1. Helps You Better Understand Your Sleep Patterns

### 2. Tracks Your Different Sleep Phases

### 3. Captures Possible Sleep Interruptions and Disruptions

### 4. Gently Wakes You Up at the Ideal Time

### 5. Convenient Compared to Wearing a Sleep Tracker Device

DIS-ADVANTAGE :

### 1. Hinders Sleep More Than It Helps

### 2. Promotes Nighttime Screen Use Before Bed

### 3. Incorrect or Unreliable Sleep Tracking Data

### 4. Creates an Unhealthy Sleep Obsession

### 5. Unable to Track Heart Rate and Movements

**5.APPLICATIONS**

## 1. Sleep Cycle

Sleep Cycle takes the scientific approach to waking you up after a good night's rest. Using your smartphone’s accelerometer and other sensors to record your sleeping habits, the free app uses sleep cycle theory to wake you at just the right time, ensuring you’re getting optimal rest.

## 2.SleepScore

## SleepScore takes a different approach to recording your sleep quality than some of the other best sleep apps we've included here. It ditches wearables or the need to place your phone on the mattress to record your movement, instead using your phone’s microphone and speakers like a sonar station to record your sleep movements. That helps the app generate a sleep score and record a variety of other metrics that show your sleep.

## 3. Headspace

Free Headspace users get a limited library guided meditations and Sleepcasts, with premium subscribers who pay $12.99 a month getting access to the full library with more than 40 different courses and shorter meditations.

## 4. Calm

Calm is another relaxation app that finds a place on our list of best sleep apps. This download for guided meditation and mindfulness features a variety of programs to reduce stress and anxiety, which includes helping you get better sleep.

## 5. Pillow

Pillow turns to the sensors on your iPhone or Apple Watch to track your sleep duration and quality. From the data it collects, Pillow can provide breakdowns showing different sleep stages, such as REM and deep sleep, along with heart rate recording and sleep quality

6.CONCLUSION

There are a variety of sleep analysis apps with a range of functionality. The apps with the most reviews from the each store are featured. Many apps provide data on sleep structure; however the algorithms are not validated by scientific literature or studies. Since patients may inquire about their sleep habits from these apps, it is necessary for physicians to be aware of the most common apps and the features offered and their limitations in order to properly counsel patients.

Sleep apps are becoming increasingly popular among those looking for a better night’s rest. The apps use various methods to track sleep, including motion sensing and sound analysis, to help users sleep better.

“All sleep apps focus on the timing of sleep,” says Dr.Santhi. “Ever wondered why? Well early in the night when sleep pressure is high, we tend to have a great amount of  ‘deep sleep’ or stage N3. This reduces as you sleep through the night.”

“On the other hand REM sleep is primarily governed by the circadian clock and REM sleep tends to occur later in the night towards the early morning hours. This is why the duration of deep sleep is greater in the first part of your sleep and there is more REM sleep towards the latter part.”

By providing users with access to personalized sleep data and tools, these apps can help improve sleep quality and duration. We do want to caution that sleep apps are not as precise as the equipment used in proper sleep studies. They can be relied on for a general idea of how well you sleep at night, but it’s important to avoid obsessing over the data.

“One of the best things to do for sleep is to not stress about it,” says Dr.Santhi.

Sleep and lifestyle apps are not necessary for health, but they do make it easier to monitor our diet, mental state, and sleep.

7.FEATURE SCOPE:

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

**Must-Have:**  
Users need to be able to set up their profile to include important information relevant to their goals  
Users need to be able to set goals and track their progress  
Users need to be able to share their stats with their wellness coaches

**Nice to Have:**  
Educational component: find a way for users to stay informed throughout the process and understand why they are doing these things, and how it will affect their well-being.

This work describes user expectations and experiences of using four commercially available sleep tracking devices. Further, we analyze the sleep data quality collected in home settings by these devices. We first discuss on the experiences collected *via* interviews, and then we highlight the takeaways and recommendations on research point of view.

We recruited participants with open calls at the authors' workplace and using social media to publish the calls to the general public (but limited to the city where the research was conducted). The participants were selected to include both young adults and older participants.

**8.Appendix**

**Source code :-**

**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

}

}

}

}

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)

}

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)

}

Regsiterionactivity.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

)

Timelogdeo.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))

}

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?,

)

Userdeo.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

}

}