A SLEEP TRACKING APP FOR A BETTER NIGHT'S REST

SUBMITTED BY

TEAM LEADER:

UMMAREDDY SAI RAGHAVENDRA-20X21A0424

TEAM MEMBERS:

UMMAREDDY SAI RAGHAVENDRA-20X21A0424
POTHANA KOWSHIK-20X21A0418
KONDAPATURI MAHESH BABU-21X25A0404

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PRIYADARSHINI INSTITUTE OF TECHNOLOGY AND SCIENCE

CHINTALAPUDI-522306

1.INTRODUCTION

1.1. Overview

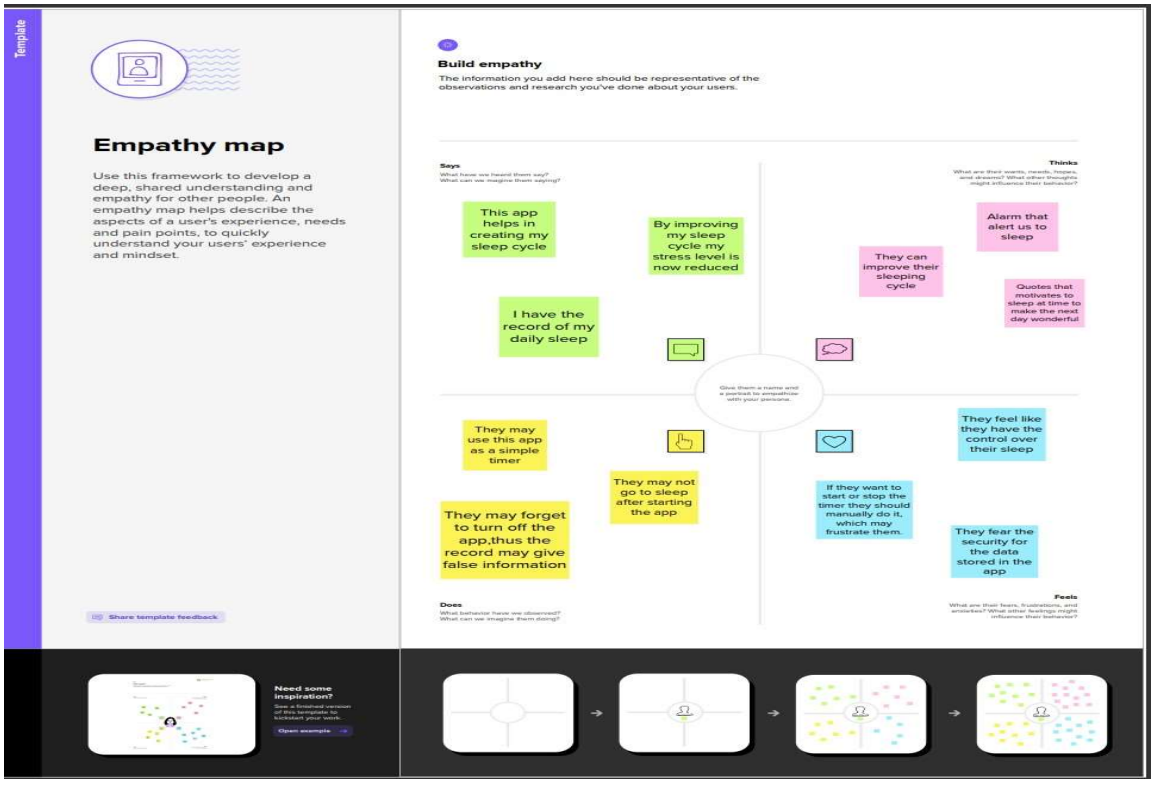
This project involves the development of a sleep tracking app that aims to help users achieve better quality sleep. The app includes features such as sleep tracking, analysis of sleep patterns, personalized recommendations, and alarm/wake-up features.

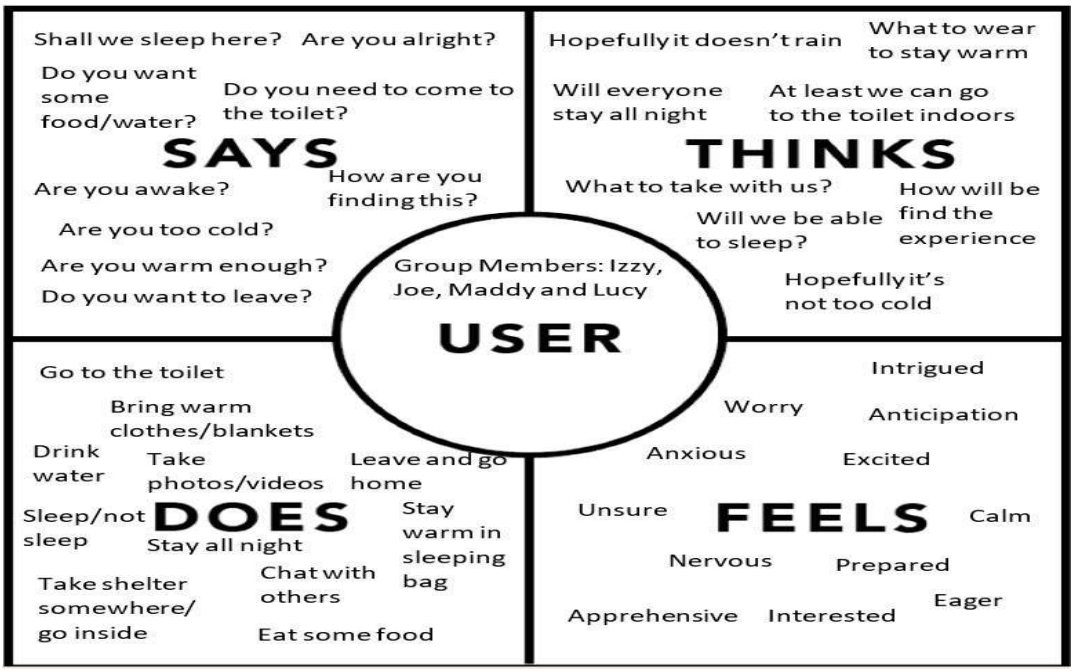
1.2. Purpose

The purpose of the sleep tracking app project is to create a digital tool that helps users improve their sleep quality by tracking and analyzing their sleep patterns. The app aims to provide personalized insights and recommendations based on users' data, ultimately leading to better sleep hygiene and overall health.

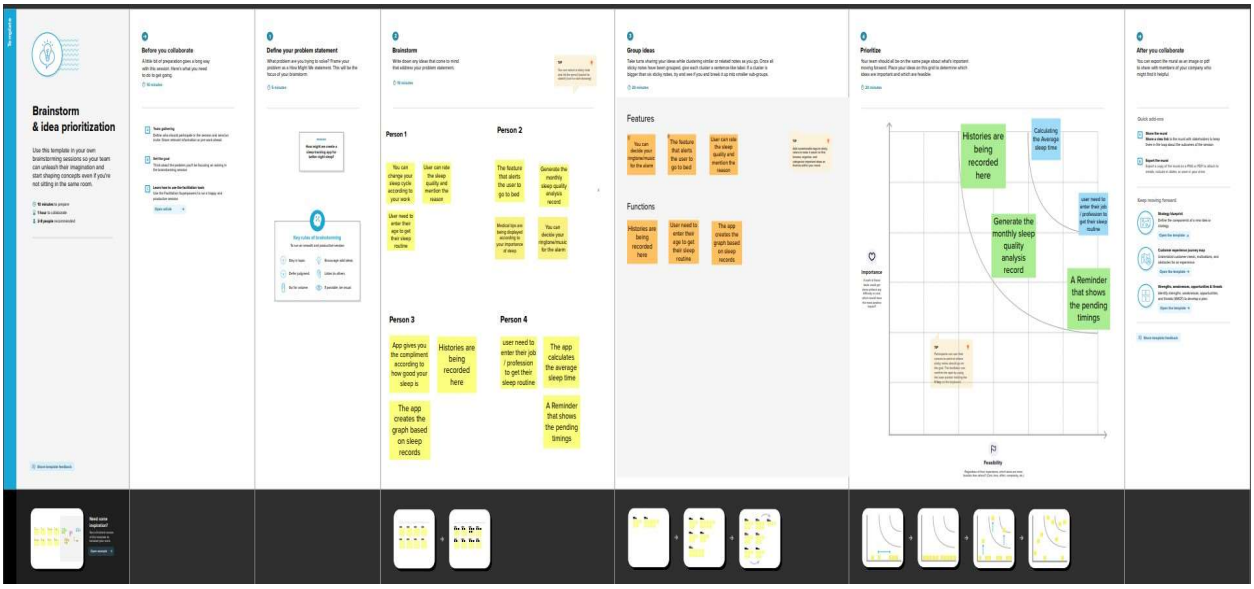
2. PROBLEM DEFINITION & DESIGN THINKING

2.1. Empathy Map





2.2 Ideation & Brainstorming Map

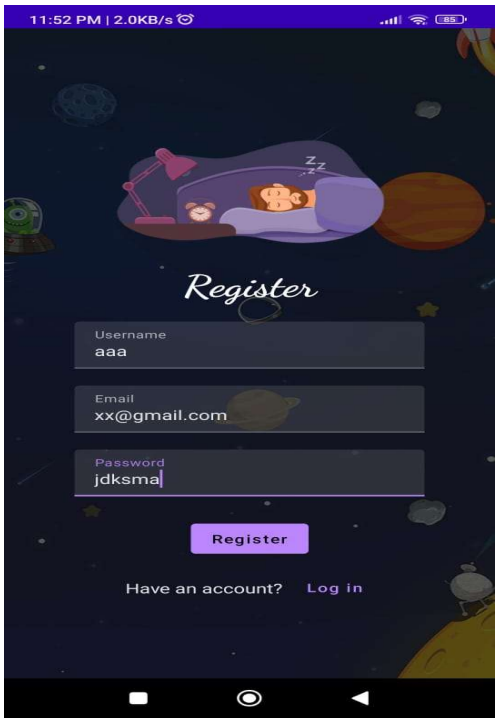


3. RESULT

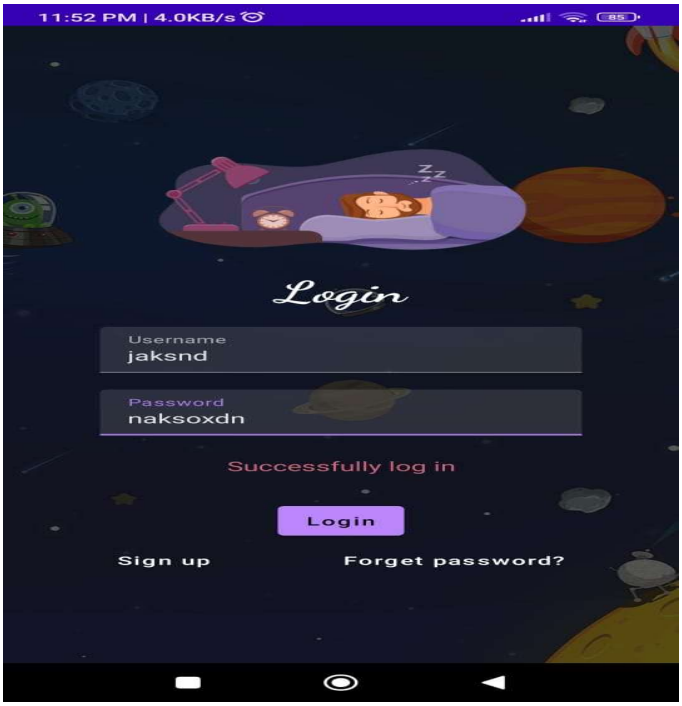
3.1. Data Model

Object name	Fields in the Object	
Sign up	Field label	Data type
	Username	String
	Password	String
	Email	String
Log in	Field label	Data type
	Username	String
	Password	String

3.2. The screenshots of your project activity along with the description.

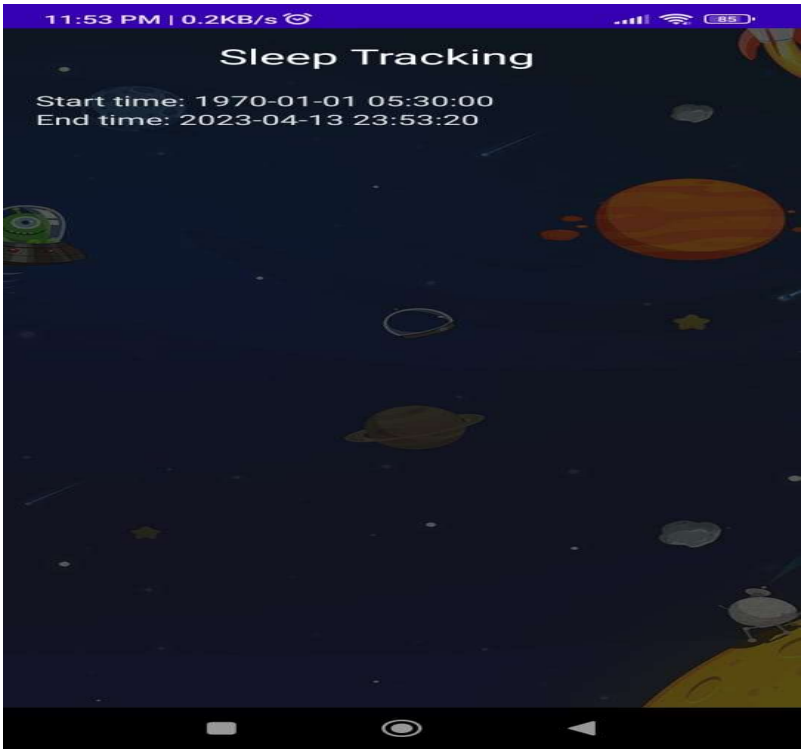


Sign Up :
It is a page where user have to register for creating new account for them , then only they can log in with that details



Login :
The user should enter a correct details to login

Elapsed time: sleep tracking is the duration of sleep, which can help users understand their sleep patterns and make improvements



4. ADVANTAGES & DISADVANTAGE

Advantages :

1. Helps users to understand their sleep patterns and identify areas for improvement
2. Provides personalized recommendations for better sleep habits
3. Can help users to feel more rested and refreshed during the day
4. May assist in identifying potential sleep disorders or issues that require medical attention
5. Can be a useful tool for tracking progress and achieving sleep-related goals

Disadvantages :

1. May not be entirely accurate in tracking sleep patterns or quality
2. Requires consistent use and may be inconvenient for some users to wear or use regularly
3. May create unnecessary anxiety or stress for users who become overly focused on sleep tracking data
4. Can be expensive to purchase and maintain a sleep tracking device or app subscription
5. May not be suitable for all individuals, such as those with certain medical conditions or disabilities.

5. APPLICATIONS

This would involve continuously monitoring a user's sleep patterns, such as sleep duration, quality, and interruptions, and providing real-time feedback to the user.

6. CONCLUSION

The sleep tracking app developed provides insights into sleep patterns, helping users to make adjustments to improve their sleep quality. The app's features make it a useful tool for tracking progress and achieving sleep-related goals.

7. FUTURE SCOPE

Future development could integrate wearable devices, AI, and community features. Partnerships with healthcare providers could allow for more comprehensive sleep analysis and treatment recommendations. These could help make a positive impact on individuals seeking to improve their sleep quality.

8. APPENDIX

CODE:

LoginActivity.kt

```
package com.example.sleeptrackingapp
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.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.sleeptrackingapp.ui.theme.SleepTrackingAppTheme

class LoginActivity : ComponentActivity() {
private lateinit var databaseHelper: UserDatabaseHelper
```

```
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = UserDatabaseHelper(this)
setContent {
SleepTrackingAppTheme {
// 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") },
visualTransformation = PasswordVisualTransformation(),
```

```
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
// )
// )
startMainPage(context)
}
)
{ 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)
}
```

RegistrationActivity.kt

```
package com.example.sleeptrackingapp
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.util.Patterns
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.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
```

```
import com.example.sleeptrackingapp.ui.theme.SleepTrackingAppTheme

class MainActivity2 : ComponentActivity() {
private lateinit var databaseHelper: UserDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = UserDatabaseHelper(this)
setContent {
SleepTrackingAppTheme {
// 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") },
visualTransformation = PasswordVisualTransformation(),
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()) {
if(email.isValidEmail()) {
```

```
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 check the email"
}
} 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 = {
// context.startActivity(
// Intent(
// context,
// LoginActivity::class.java
// )
// )
startLoginActivity(context)
})
{
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)
}
fun CharSequence?.isValidEmail() = !isNullOrEmpty() &&
Patterns.EMAIL_ADDRESS.matcher(this).matches()
```

User.kt

```
package com.example.sleeptrackingapp
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.sleeptrackingapp
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.sleeptrackingapp
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.sleeptrackingapp
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
}
}
```

MainActivity.kt

```
package com.example.sleeptrackingapp
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.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.sleeptrackingapp.ui.theme.SleepTrackingAppTheme
import java.util.*

import kotlin.concurrent.scheduleAtFixedRate
class MainActivity : ComponentActivity() {
private lateinit var databaseHelper: TimeLogDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = TimeLogDatabaseHelper(this)
databaseHelper.getAllData()
setContent {
SleepTrackingAppTheme {
// 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 endTime by remember { mutableStateOf(0L) }
var isRunning by remember { mutableStateOf(false) }
```

```
// var firstAttempt by remember { mutableStateOf(true) }
var currentTime by remember { mutableStateOf(System.currentTimeMillis()) }
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
) {

Text(
fontSize = 50.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.White,
text = "Sleep Tracking"
)
Spacer(modifier = Modifier.height(16.dp))
if (isRunning) {
Button(onClick = {

endTime = System.currentTimeMillis()
isRunning = false
}) {
```

```
Text("Stop")
//databaseHelper.addTimeLog(startTime)
}
} else {
Button(onClick = {
startTime = System.currentTimeMillis()
isRunning = true
}) {
Text("Start")
databaseHelper.addTimeLog(startTime, endTime)
}
}
Spacer(modifier = Modifier.height(200.dp))
if(isRunning)
{
Timer().scheduleAtFixedRate(0, 1000) {
currentTime = returnCurrentTime()
}
Text(text = "Sleep Time: ${formatTime(currentTime - startTime)}")
}
else
{
Text(text = "Time Not Started")
}

Spacer(modifier = Modifier.height(156.dp))
Button(onClick = {
// context.startActivity(
// Intent(
// context,
// TrackActivity::class.java
```

```
// )
// )
startTrackActivity(context)
}) {
Text(text = "Track Sleep")
}
Spacer(modifier = Modifier.height((16.dp)))
Button(onClick = {
databaseHelper.deleteAllData()
}){
Text(text = "Clear Tracking History")
}
}
}
private fun startTrackActivity(context: Context) {
val intent = Intent(context, TrackActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}
fun getCurrentDateTime(): String {
val dateFormat = SimpleDateFormat("dd-MM-yyyy 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)
}
fun returnCurrentTime() : Long {
```

```
return System.currentTimeMillis()
}
```

TimeDatabaseHelper.kt

```
package com.example.sleeptrackingapp
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 db = writableDatabase
val values = ContentValues()
values.put(COLUMN_START_TIME, startTime)
values.put(COLUMN_END_TIME, endTime)
db.insert(TABLE_NAME, null, values)
db.close()
}

// 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.sleeptrackingapp
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.sleeptrackingapp
import androidx.room.Dao
import androidx.room.Insert
@Dao
interface TimeLogDao {
@Insert
suspend fun insert(timeLog: TimeLog)
}
```

TrackActivity.kt

```
package com.example.sleeptrackingapp
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.sleeptrackingapp.ui.theme.SleepTrackingAppTheme
import java.util.*
class TrackActivity : ComponentActivity() {
private lateinit var databaseHelper: TimeLogDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = TimeLogDatabaseHelper(this)
setContent {
SleepTrackingAppTheme {
// 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))
```

```
}
```

AppDatabase.kt

```
package com.example.sleeptrackingapp
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
}
}
}
```

```
}
```

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.SleepTrackingApp"
tools:targetApi="31">
<activity
android:name=".TrackActivity"
android:exported="false"
android:label="@string/title_activity_track"
android:theme="@style/Theme.SleepTrackingApp" />
<activity
android:name=".MainActivity"
android:exported="false"
android:label="@string/app_name"
android:theme="@style/Theme.SleepTrackingApp" />
<activity
android:name=".MainActivity2"
android:exported="false"
android:label="RegisterActivity"
android:theme="@style/Theme.SleepTrackingApp" />
```

```
<activity
android:name=".LoginActivity"
android:exported="true"

android:label="@string/app_name"
android:theme="@style/Theme.SleepTrackingApp">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

THANK YOU