Project Report

A Sleep Tracking App for a Better Night's Rest

1. Introduction

1.1 Overview

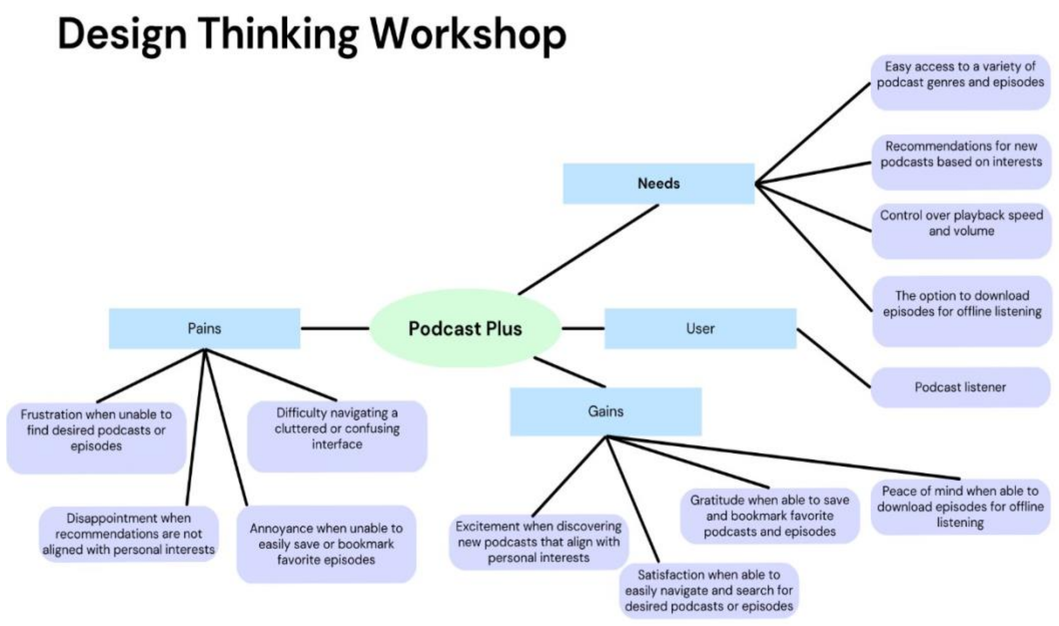
We have developed a sleep tracking app for a better night's sleep using Kotlin and Jetpack Compose. It is a compact app that you can use anywhere everywhere.

1.2 Purpose

The main purpose of our sleep tracking app is to develop an app that is comfortable to use and to pretty much have a very less learning curve when it comes to using the app for everyone

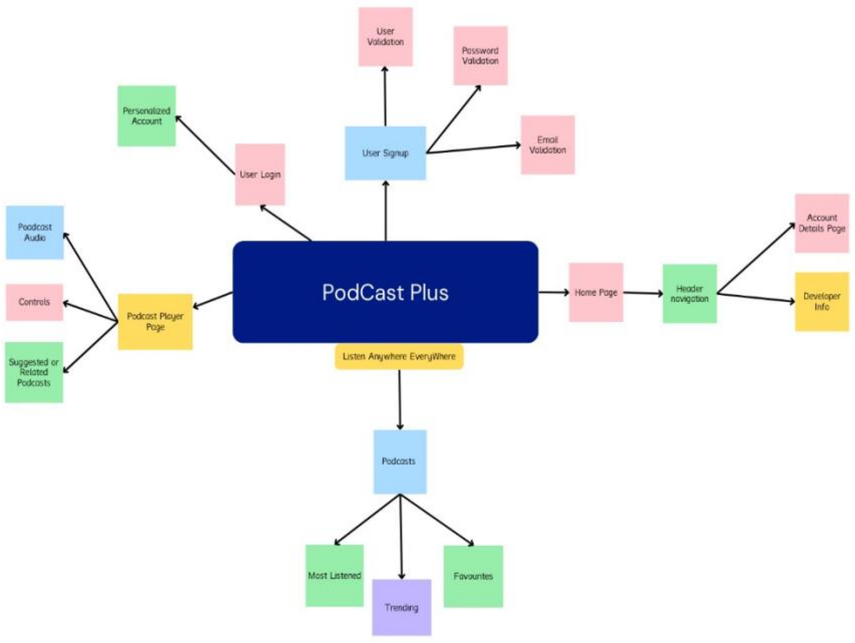
2. Problem Definition & Design Thinking

2.1 Empathy Map



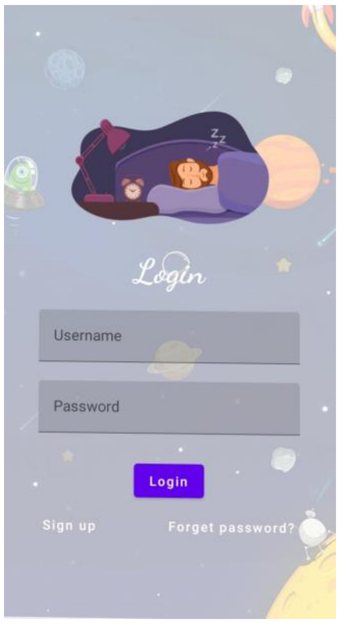
Design Thinking Workshop
Podcast Plus
Needs
Easy access to a variety of podcast genres and episodes
Recommendations for new podcasts based on interests
Control over playback speed and volume
The option to download episodes for offline listening
User
Podcast listener
Pains
Frustration when unable to find desired podcasts or episodes
Difficulty navigating a cluttered or confusing interface
Disappointment when recommendations are not aligned with personal interests
Annoyance when unable to easily save or bookmark favorite episodes
Gains
Excitement when discovering new podcasts that align with personal interests
Satisfaction when able to easily navigate and search for desired podcasts or episodes
Gratitude when able to save and bookmark favorite podcasts and episodes
Peace of mind when able to download episodes for offline listening

2.2 Ideation and Brainstorm Map

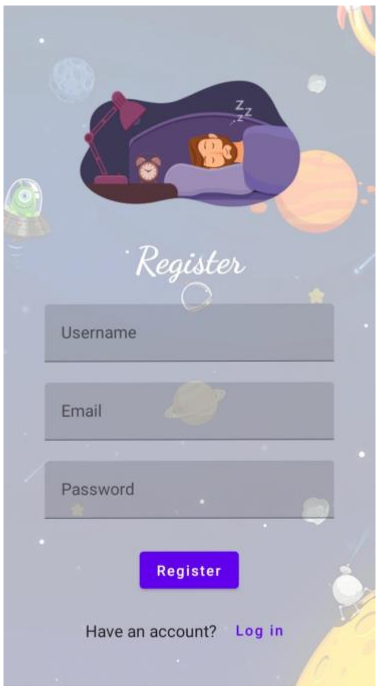


3. Result

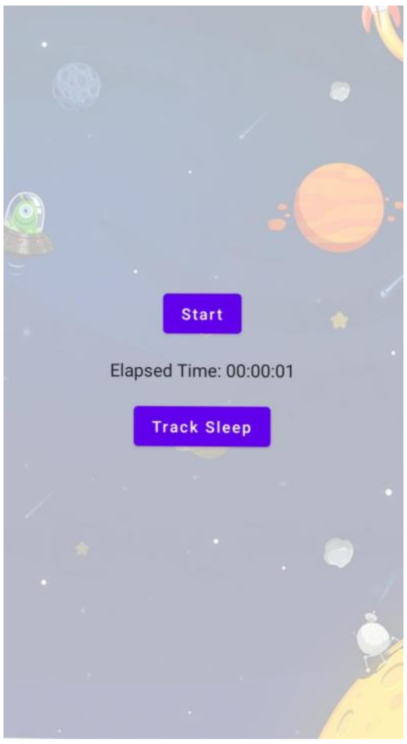
Login Page:



Registeration Page:

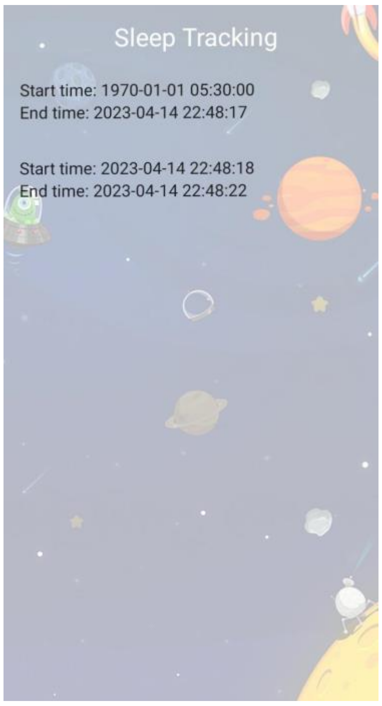


Main Page:



Elapsed Time: 00:00:01

Tracking Page:


Sleep Tracking

Start time: 1970-01-01 05:30:00
End time: 2023-04-14 22:48:17

Start time: 2023-04-14 22:48:18
End time: 2023-04-14 22:48:22

4. Advantages:

- Easy to use: A sleep tracking app is easy to use, navigate, and understand. Users can quickly track and check their sleeping schedule so far and access them without any confusion or frustration.

- Saves time: A simple sleep tracking app can save users a lot of time by providing them with a streamlined user experience. Users can quickly find and access the timer they want to choose , without having to spend a lot of time searching for them.

- Reduced complexity: A simple sleep tracking app reduces the complexity of the user experience, making it more accessible to a wider range of users. This can help attract and retain users who might otherwise be put off by a complex or confusing app.

- Better user engagement: A simple sleep tracking app can help increase user engagement by making it easy for users to find and access their needed icons and buttons .

- Enhanced user experience: A simple podcast app with a login page, sign up page, podcast page with some podcasts, and a player page for each podcast can provide users with a more engaging and personalized experience.

Disadvantages:

- Limited features: A simple sleep tracking app with only basic features may not meet the needs of all users, especially those who are looking for more advanced features such as playing background soothing music ,,calculating average sleep time, setting alarms while sleeping

- Lack of differentiation: A simple sleep tracking app with a basic design and limited features may not stand out in a crowded marketplace, making it harder to attract and retain users.

- Limited monetization opportunities: A simple sleep tracking app with limited features may have fewer opportunities to monetize through advertising, sponsorships, or premium subscriptions, potentially limiting revenue streams for the app.

- Limited scalability: A simple sleep tracking app with a limited set of features may not be able to handle a large user base or high levels of traffic, potentially limiting its scalability and ability to grow.

- Limited user data: A simple sleep tracking app with limited features may not collect enough user data to provide personalized recommendations or improve the user experience over time.

Application:

- User-friendly interface: The application should have a simple and intuitive interface that allows users to easily search for and listen to podcasts.

- Login and sign-up pages: Users should be able to create an account or log in to access personalized features such as bookmarking, creating playlists, and receiving personalized recommendations.

- Sleep tracking page: The application should have a page dedicated to displaying a variety of advanced options.

- Tracking page: The application should have a tracking page , which should include basic playback controls such as resume and pause after starting the tracking timer.

Conclusion:

Overall Sleep tracking is a simple to use sleep tracking app that provides a very easy to use interface for everybody. Further improvements can be added to this app which is being discussed in the next session.

Future Scope:

Personalized Suggestions:

The predominant thing that is lacking in sleep tracker is personalized suggestions due to lack of resources.

Personalized Pages:

Allowing the user to customize the user interface like adding night mode and etc.

Appendix:

Login Activity:

```
package com.example.sleeptracker

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.sleeptracker.ui.theme.SleepTrackerTheme


class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackerTheme() {
                // 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(
```

```kotlin
                    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)
}
```

Registeration Activity:

```
 package com.example.sleeptracker

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.sleeptracker.ui.theme.SleepTrackerT
heme



class MainActivity2 : ComponentActivity() {
    private lateinit var databaseHelper:
UserDatabaseHelper
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper =
UserDatabaseHelper(this)
        setContent {
            SleepTrackerTheme() {
                // 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)
```

[redacted]

Main Activity:

```
package com.example.sleeptracker

import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Button
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import com.example.sleeptracker.ui.theme.SleepTrackerTheme
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 {
            SleepTrackerTheme() {
                // 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) }
```

```kotlin
    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)
}
@RequiresApi(Build.VERSION_CODES.N)
fun getCurrentDateTime(): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
    val currentTime = System.currentTimeMillis()
    return dateFormat.format(Date(currentTime))
}

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

Sleep Tracker:

```
package com.example.sleeptracker

import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.sleeptracker.ui.theme.SleepTrackerTheme
import java.util.*

class TrackActivity : ComponentActivity() {

    private lateinit var databaseHelper: TimeLogDatabaseHelper

    @RequiresApi(Build.VERSION_CODES.N)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = TimeLogDatabaseHelper(this)
        setContent {
            SleepTrackerTheme() {
                // 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("Sleur" ,data.toString())
                    val timeLogs = databaseHelper.getTimeLogs()
                    ListListScopeSample(timeLogs)
                }
            }
        }
    }
}


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

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

        horizontalArrangement = Arrangement.SpaceBetween
    ){
        item {

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

```kotlin
                }
            }
        }

    }
}

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