A Sleep Tracking App for a Better Night's Rest

Project Report

1 Introduction:

1.1) Overview:

A sleep tracking app provides information on the user's sleep time that helps them find the quality time for sleep and these apps may track total sleep time, suggestions based on the feedback of the every night sleep quality.

A sleep tracking apps can be a useful tool for individual who want to monitor their sleep patterns and gain insights into their sleep quality. However, it's important to keep in mind that these apps are not a substitute for professional medical advice and should not be relied on to diagnose or treat sleep disorders.

Users should also be aware that the every human has different sleep patterns, thus consulting the doctor is the advisable to find the better sleep pattern of yourself.

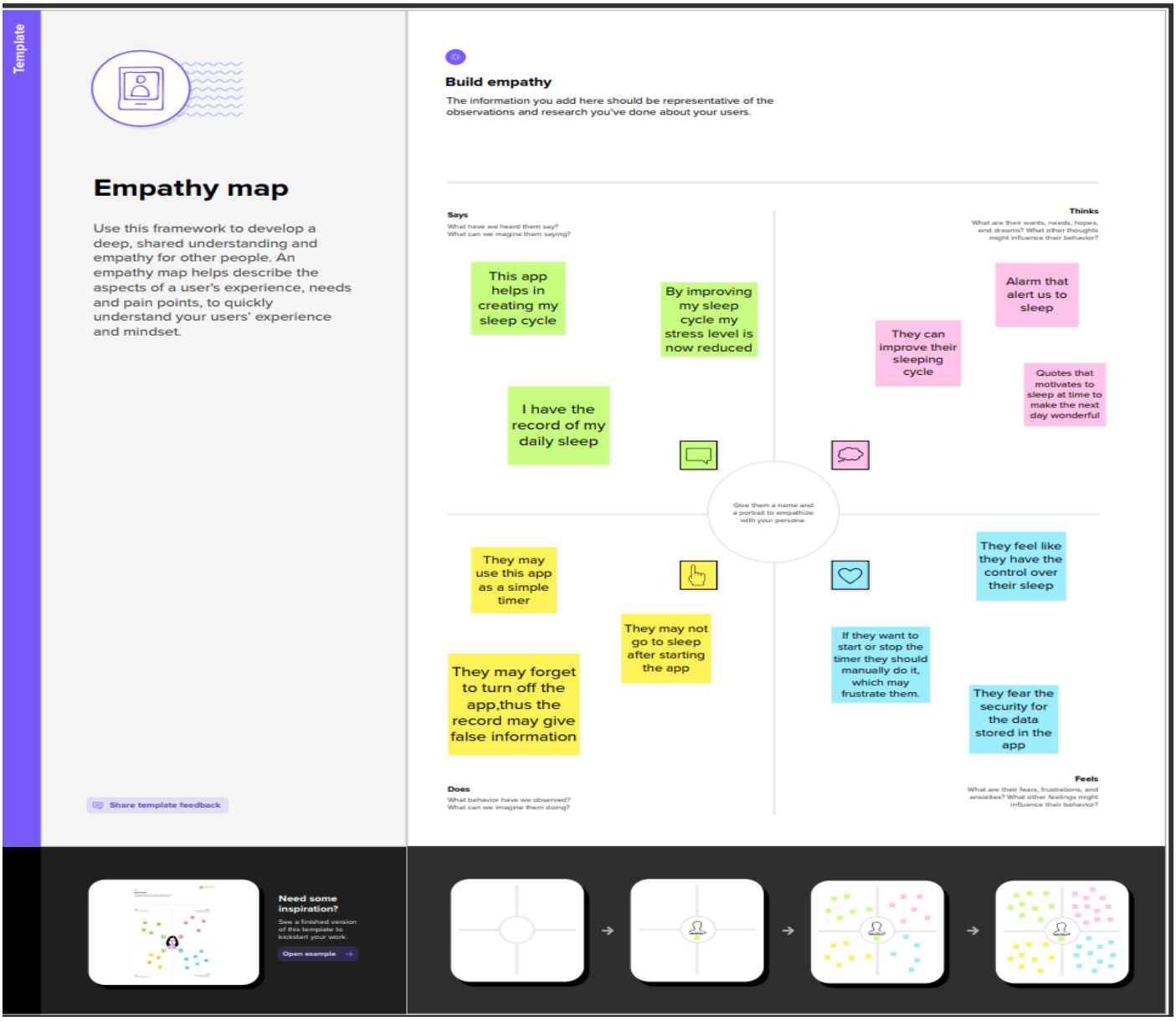
1.2) Purpose:

The purpose of this sleep tracking app is to monitor and track an individual's sleep patterns, including the duration and quality of their sleep.

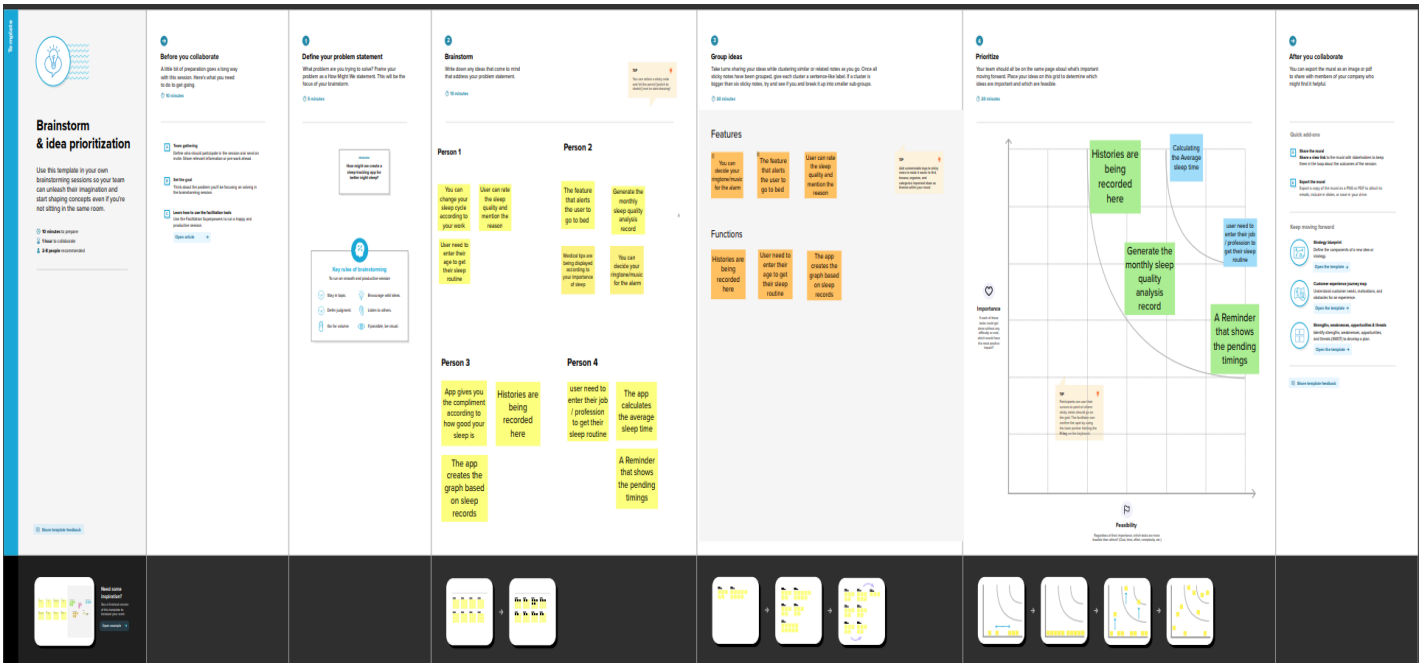
The user can find their sleep pattern after using this app for tracking their sleep.

2 Problem Definition & Design Thinking

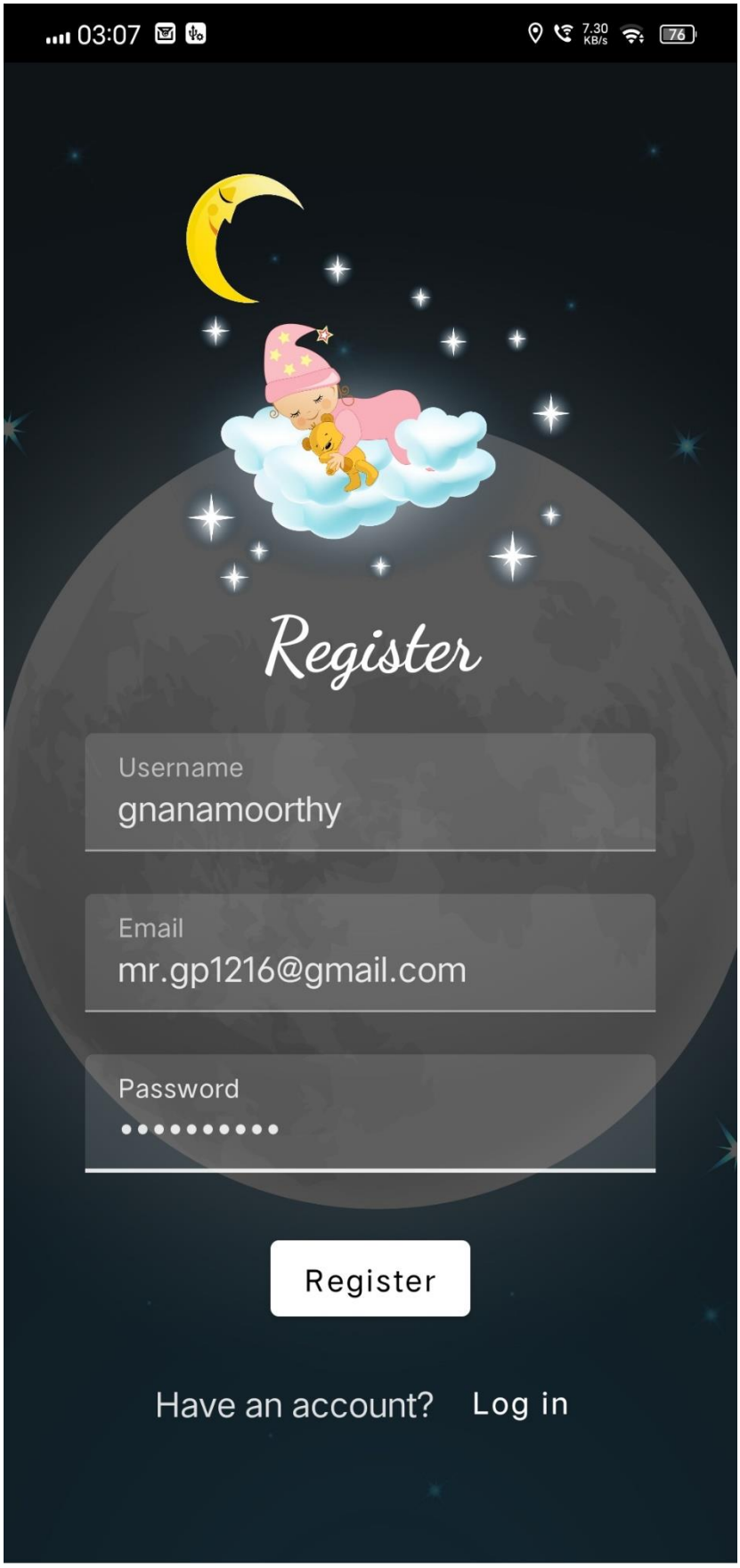
2.1 Empathy Map:

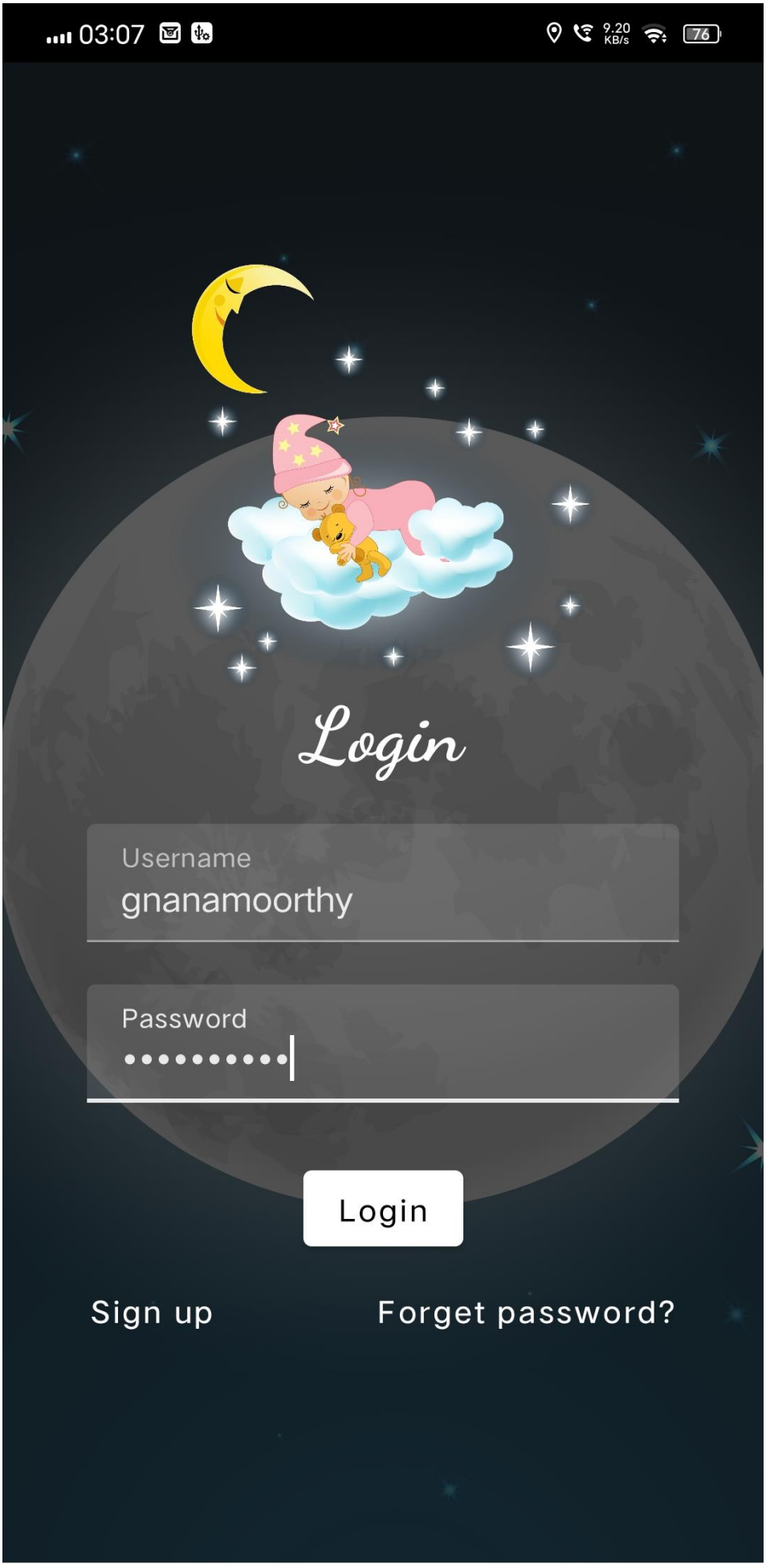


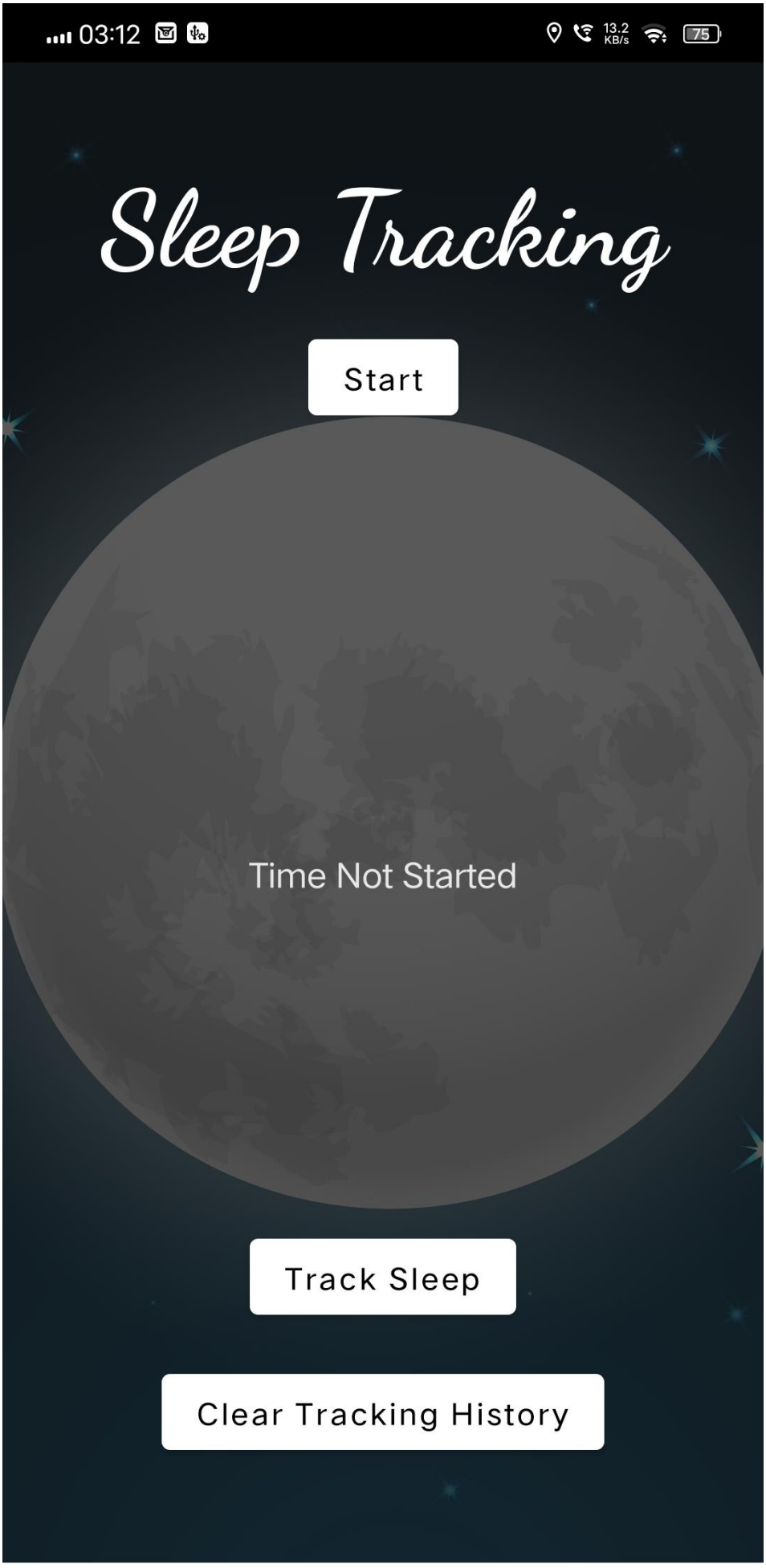
2.2 Ideation & Brainstorming Map:


Brainstorm
& idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Person 1
Person 2
Person 3
Person 4
Group ideas
Features
Functions
Prioritize
Histories are being recorded here
Calculating the Average sleep time
Generate the monthly sleep quality analysis record
A Reminder that shows the pending timings
Importance
Feasibility
After you collaborate

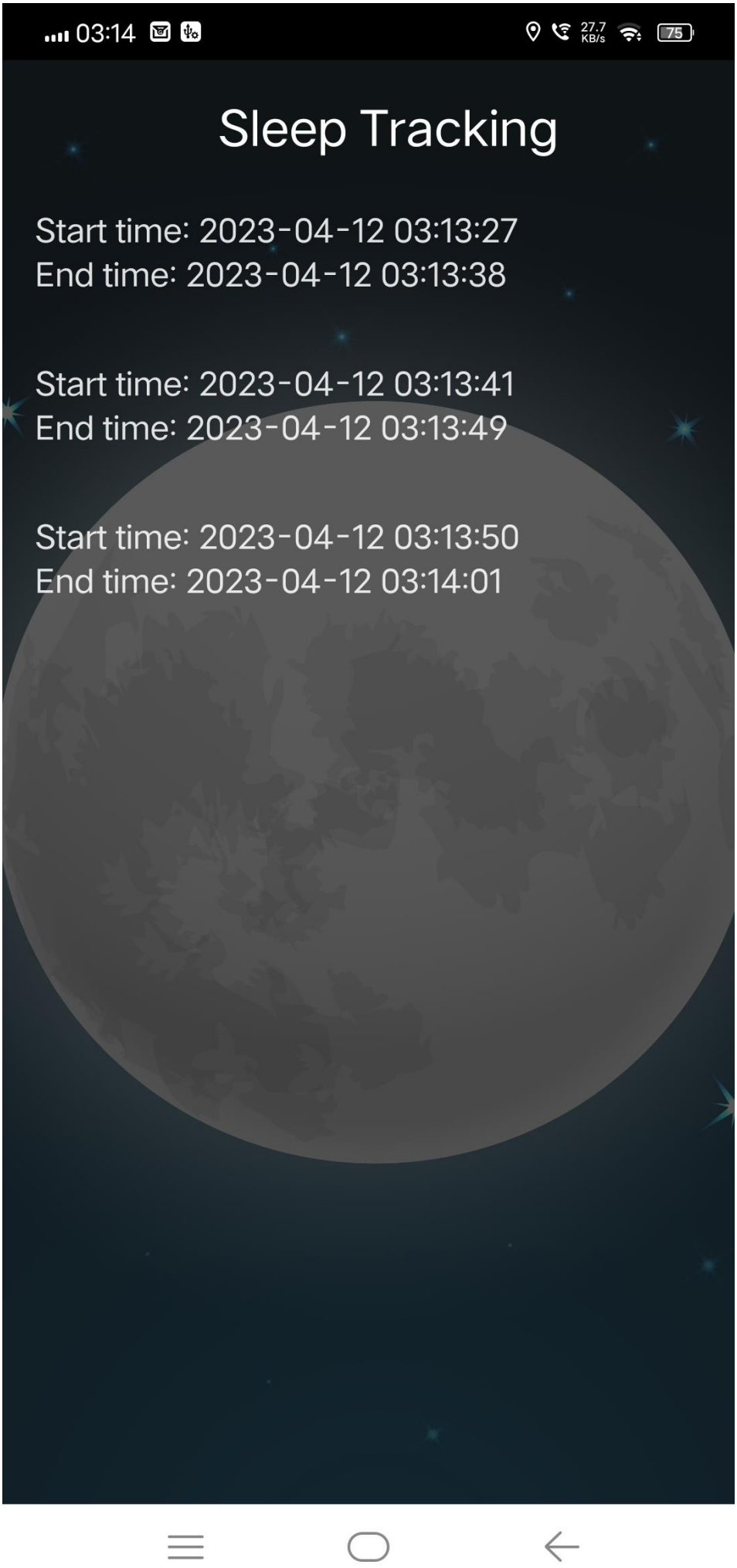
3 Result:


03:07
7.30 KB/s
76
Register
Username
gnanamoorthy
Email
mr.gp1216@gmail.com
Password
••••••••••
Register
Have an account?
Log in


03:07
9.20 KB/s
76
Login
Username
gnanamoorthy
Password
Login
Sign up
Forget password?


03:12
13.2 KB/s
75
Sleep Tracking
Start
Time Not Started
Track Sleep
Clear Tracking History


03:13
9.20 KB/s
75
Sleep Tracking
Stop
Sleep Time: 00:00:18
Track Sleep
Clear Tracking History


03:14
27.7 KB/s
75
Sleep Tracking
Start time: 2023-04-12 03:13:27
End time: 2023-04-12 03:13:38
Start time: 2023-04-12 03:13:41
End time: 2023-04-12 03:13:49
Start time: 2023-04-12 03:13:50
End time: 2023-04-12 03:14:01

4 Advantages & Disadvantages

Advantages:

- Improved sleep quality
- Increased awareness about sleep and factors that affect sleep
- Improved Energy level
- Improvement in Overall health
- Reduce Stress

Disadvantages:

- The user have to spend more time to get the exact result
- As the app only contains the start time and end time of the sleep
- Thus the user should calculate the difference between those two time which are stored previously

5 Applications:

- This app help in finding the sleep pattern thus any age group who were intrested in finding sleep pattern can use it.
- It can be used to monitor the sleep disorders such as sleep apnea, imsomnia, and restless leg syndrome.
- Stress reduction can be done by tracking sleep and make correct treatment by doctor

6 Conclusion:

Thus this sleep tracking app can be useful for who want to monitor and optimize their sleep for better night rest and better overall health and well-being.

7 Future Scope:

This app is currently in developing stage and it has many updates in future some of them are,

- Review after every sleep to give suggestions
- Improvement in User Interface
- Adding alarm that indicates it’s the time to sleep

8 APPENDIX

A) Source Code

https://github.com/TGP1216/A-Sleep-Tracking-App-for-a-Better-Night-s-Rest/tree/main/A-Sleep-Tracking-App-for-a-Better-Night-s-Rest

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>
```