A SLEEP TRACKING APP FOR A BETTER NIGHT'S REST

Submitted by:

VEL MURUGAN P

SURYA VARMA N

MAHESH RAJA M

NITHISH KUMAR KS

PROJECT REPORT

SLEEP TRACKING APP

1.INTRODUCTION

1.1.1 Overview :

Sleep tracking apps are becoming increasingly popular as people are recognizing the importance of getting good quality sleep for their physical and mental health. These apps use sensors on wearable devices or smartphones to track and monitor the user's sleep patterns, including the duration, quality, and consistency of their sleep.

The apps typically provide users with detailed information about their sleep, such as the amount of time they spend in different stages of sleep, how long it takes them to fall asleep, and how often they wake up during the night. Some apps may also offer personalized recommendations for improving sleep based on the user's sleep data.

Sleep tracking apps can be a valuable tool for people who struggle with sleep issues or simply want to optimize their sleep habits. By providing insights into sleep patterns and offering recommendations for improvement, these apps can help users make informed decisions about their sleep habits and ultimately improve their overall health and well-being.

1.2 Purpose

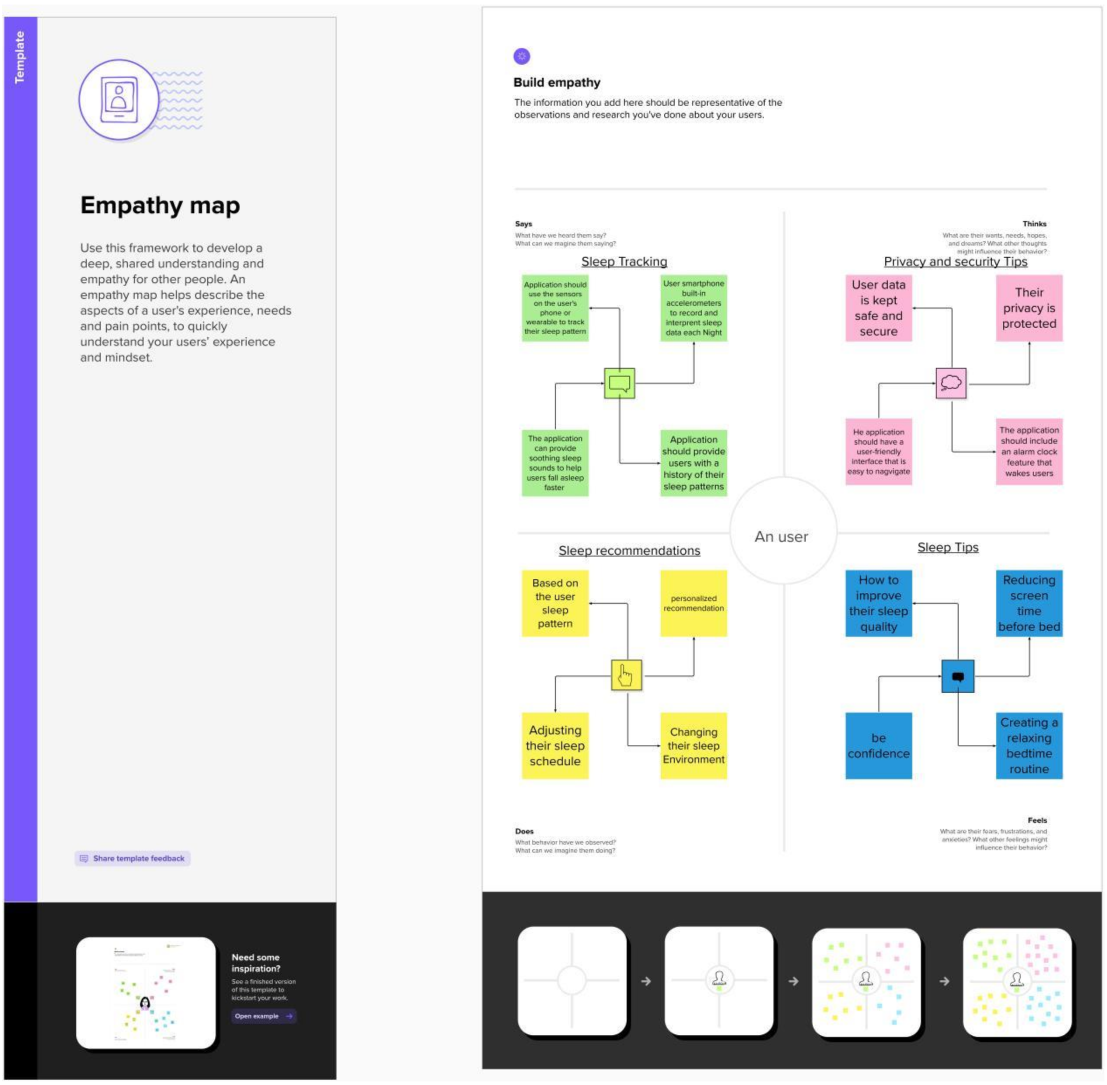
The purpose of a sleep tracking app is to help users monitor and improve the quality and duration of their sleep. Sleep is an essential part of our overall health and well-being, and getting sufficient and restful sleep can have a significant impact on our physical and mental health.

Sleep tracking apps use sensors to monitor the user's sleep patterns and provide data on factors such as the duration and quality of sleep, the number of times the user wakes up during the night, and the time it takes to fall asleep. This information can help users identify patterns and areas for improvement in their sleep habits.

By providing insights into sleep patterns and offering personalized recommendations, sleep tracking apps can help users optimize their sleep habits, which can lead to improved mood, increased productivity, and better physical health. Additionally, sleep tracking apps can help users identify and address potential sleep disorders, such as sleep apnea, that can have significant impacts on health and quality of life.

2.Problem Definition & Design Thinking

2.1 Empathy Map



Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

What have we heard them say? What can we magine them saying?

Sleep Tracking

Application should use the sensors on the user's phone or wearable to track their sleep pattern

User smartphone built-in accelerometers to record and interprent sleep data each Night

The application can provide soothing sleep sounds to help users fall asleep faster

Application should provide users with a history of their sleep patterns

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

Privacy and security Tips

User data is kept safe and secure

Their privacy is protected

He application should have a user-friendly interface that is easy to nagvigate

The application should include an alarm clock feature that wakes users

An user

Sleep recommendations

Based on the user sleep pattern

personalized recommendation

Adjusting their sleep schedule

Changing their sleep Environment

Sleep Tips

How to improve their sleep quality

Reducing screen time before bed

be confidence

Creating a relaxing bedtime routine

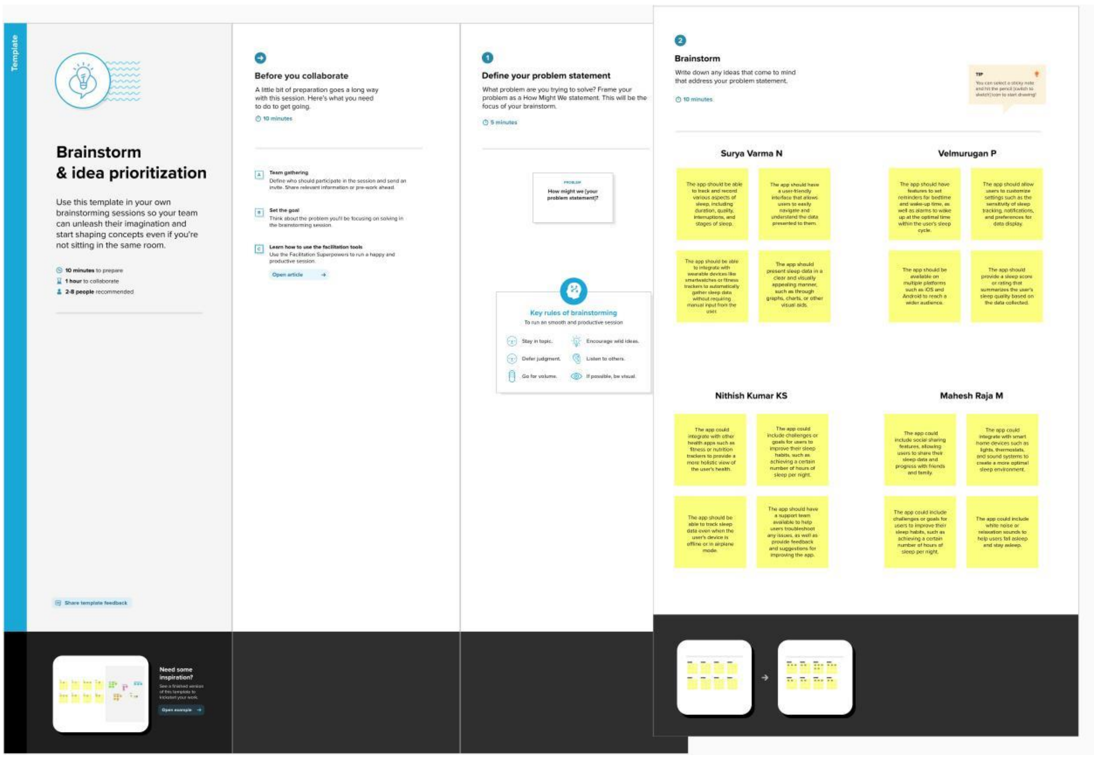
Does

What behavior have we observed? What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

2.2 Ideation & Brainstorming Map



Template

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.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.
Open article

1 **Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

2 **Brainstorm**

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Surya Varma N

- The app should be able to track and record various aspects of sleep, including duration, quality, interruptions, and stages of sleep.
- The app should have a user-friendly interface that allows users to easily navigate and understand the data presented to them.
- The app should be able to integrate with wearable devices like smartwatches or fitness trackers to automatically gather sleep data without requiring manual input from the user.
- The app should present sleep data in a clear and visually appealing manner, such as through graphs, charts, or other visual aids.

Velmurugan P

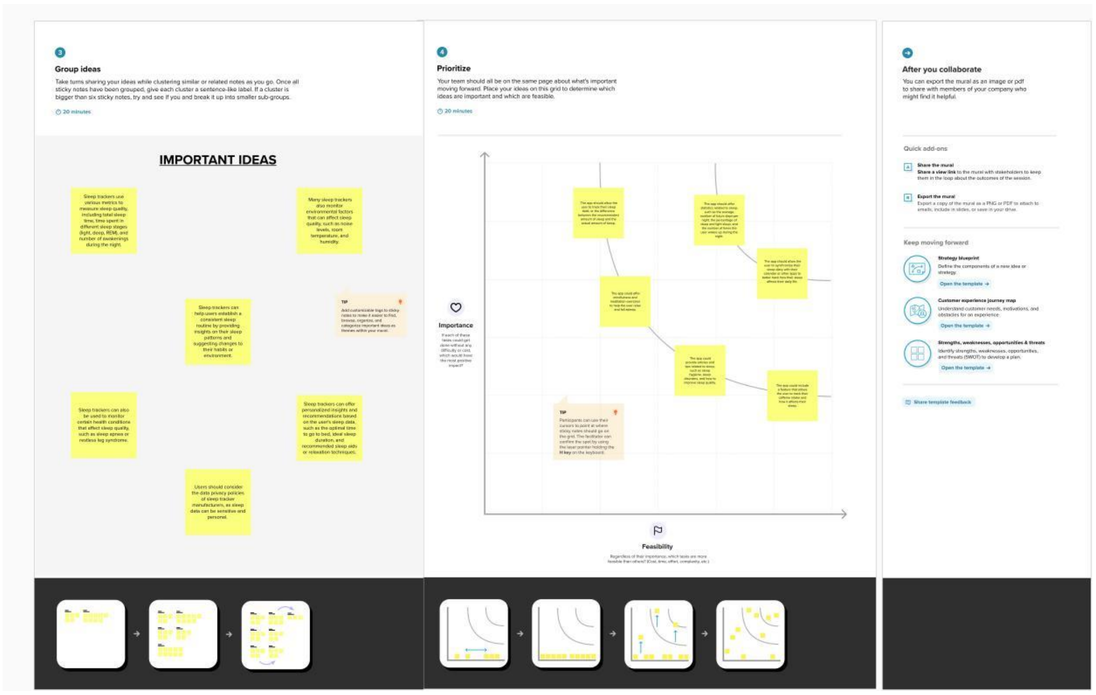
- The app should have features to set reminders for bedtime and wake-up time, as well as alarms to wake up at the optimal time within the user's sleep cycle.
- The app should allow users to customize settings such as the sensitivity of sleep tracking, notifications, and preferences for data display.
- The app should be available on multiple platforms such as iOS and Android to reach a wider audience.
- The app should provide a sleep score or rating that summarizes the user's sleep quality based on the data collected.

Nithish Kumar KS

- The app could integrate with other health apps such as fitness or nutrition trackers to provide a more holistic view of the user's health.
- The app could include challenges or goals for users to improve their sleep habits, such as achieving a certain number of hours of sleep per night.
- The app should be able to track sleep data even when the user's device is offline or in airplane mode.
- The app should have a support team available to help users troubleshoot any issues, as well as provide feedback and suggestions for improving the app.

Mahesh Raja M

- The app could include social sharing features, allowing users to share their sleep data and progress with friends and family.
- The app could integrate with smart home devices such as lights, thermostats, and sound systems to create a more optimal sleep environment.
- The app could include challenges or goals for users to improve their sleep habits, such as achieving a certain number of hours of sleep per night.
- The app could include white noise or relaxation sounds to help users fall asleep and stay asleep.



3 **Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

IMPORTANT IDEAS

- Sleep trackers use various metrics to measure sleep quality, including total sleep time, time spent in different sleep stages (light, deep, REM), and number of awakenings during the night.
- Many sleep trackers also monitor environmental factors that can affect sleep quality, such as noise levels, room temperature, and humidity.
- Sleep trackers can help users establish a consistent sleep routine by providing insights on their sleep patterns and suggesting changes to their habits or environment.
- Sleep trackers can also be used to monitor certain health conditions that affect sleep quality, such as sleep apnea or restless leg syndrome.
- Sleep trackers can offer personalized insights and recommendations based on the user's sleep data, such as the optimal time to go to bed, ideal sleep duration, and recommended sleep aids or relaxation techniques.
- Users should consider the data privacy policies of sleep tracker manufacturers, as sleep data can be sensitive and personal.

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4 **Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

- The app should allow the user to track their sleep debt, or the difference between the recommended amount of sleep and the actual amount of sleep.
- The app should offer statistics related to sleep, such as the average number of hours slept per night, the percentage of deep and light sleep, and the number of times the user wakes up during the night.
- The app should allow the user to synchronize their sleep diary with their calendar or other apps to better track how their sleep affects their daily life.
- The app could offer mindfulness and meditation exercises to help the user relax and fall asleep.
- The app could provide articles and tips related to sleep, such as sleep hygiene, sleep disorders, and how to improve sleep quality.
- The app could include a feature that allows the user to track their caffeine intake and how it affects their sleep.

TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template

Share template feedback

3.RESULT:

The result of using a sleep tracking app can be improved sleep quality and duration, as well as a better understanding of one's sleep habits. By monitoring and analyzing sleep data, users can identify patterns and areas for improvement in their sleep habits, such as adjusting their bedtime routine or sleep environment.

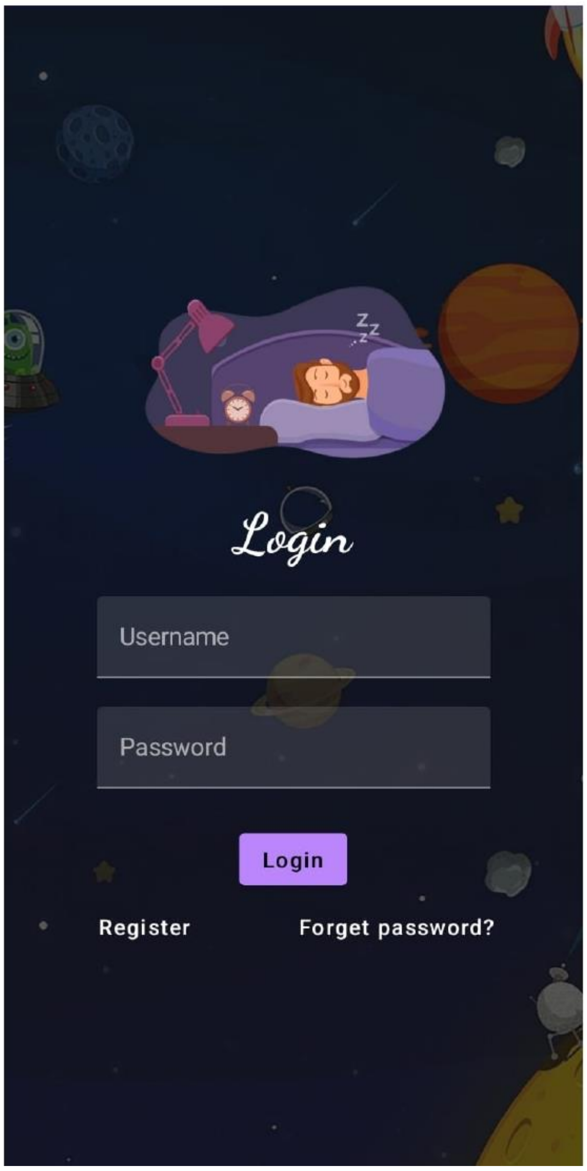
In addition to providing insights into sleep patterns, some sleep tracking apps offer personalized recommendations for improving sleep, such as adjusting the user's bedtime routine, optimizing their sleep environment, or practicing relaxation techniques before bed.

Over time, using a sleep tracking app can lead to improved overall health and well-being. Studies have shown that getting sufficient, restful sleep can have numerous health benefits, such as reducing the risk of chronic diseases, improving cognitive function, and promoting mental health.

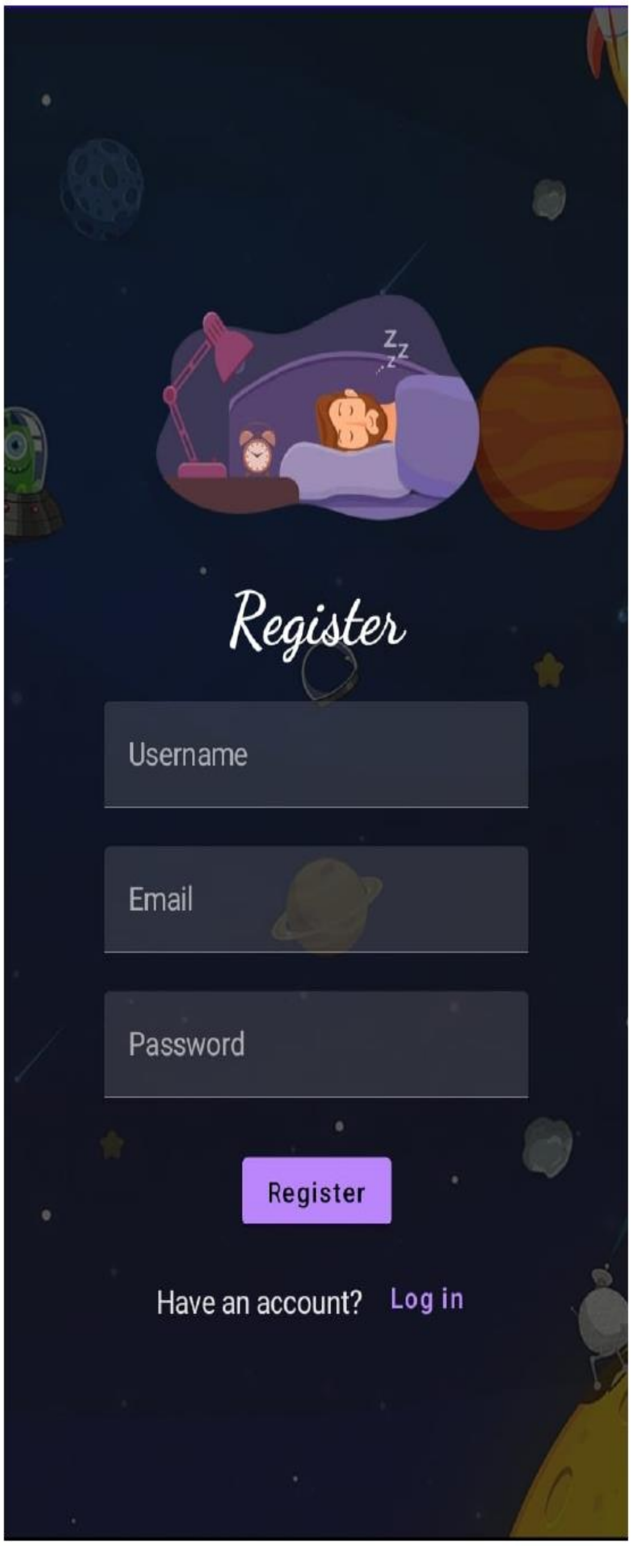
By providing users with insights into their sleep habits and personalized recommendations for improvement, sleep tracking apps can help users achieve better quality sleep, leading to improved physical and mental health outcomes.

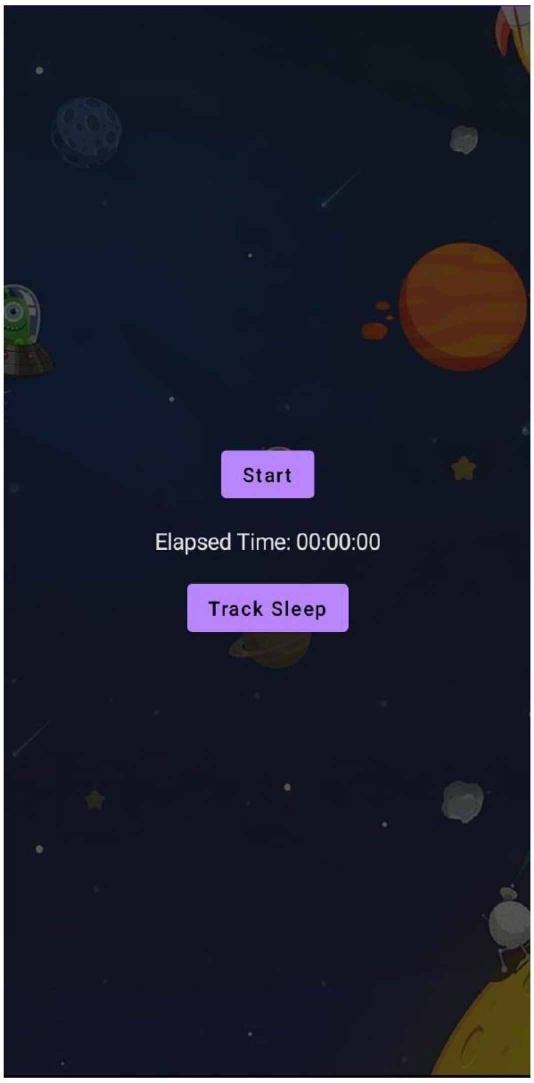
Final Output of the Application :

Login Page :



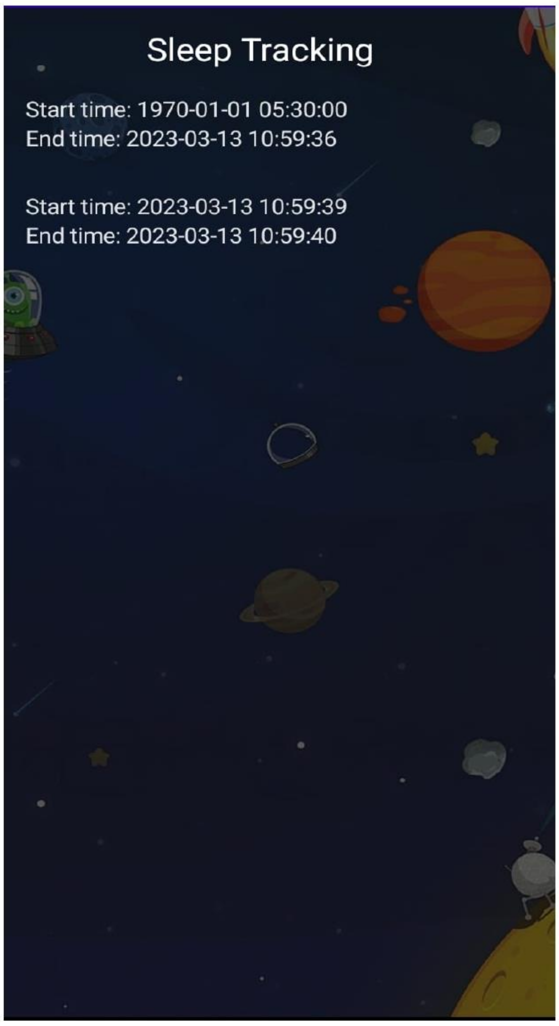
Registration Page:





MAIN PAGE

Track Sleep Page:



Sleep Tracking

Start time: 1970-01-01 05:30:00
End time: 2023-03-13 10:59:36

Start time: 2023-03-13 10:59:39
End time: 2023-03-13 10:59:40

4.ADVANTAGES & DISADVANTAGES :

There are several advantages to using a sleep tracking app:

Increased awareness of sleep habits: By using a sleep tracking app, users can gain a better understanding of their sleep habits and identify patterns that may be affecting the quality and duration of their sleep.

Personalized recommendations: Many sleep tracking apps offer personalized recommendations based on the user's sleep data, which can help them improve their sleep habits and optimize their sleep environment.

Improved sleep quality: By monitoring and analyzing sleep data, users can identify areas for improvement in their sleep habits and make changes that can lead to better quality sleep.

Improved health outcomes: Getting sufficient, restful sleep has numerous health benefits, including reducing the risk of chronic diseases, improving cognitive function, and promoting mental health. By using a sleep tracking app to improve their sleep habits, users can potentially improve their overall health and well-being.

Convenience: Sleep tracking apps are often easy to use and can be accessed from a smartphone or wearable device, making it convenient for users to monitor their sleep habits and make changes as needed.

Overall, using a sleep tracking app can be a valuable tool for improving sleep habits and achieving better overall health outcomes.

DISADVANTAGES:

While sleep tracking apps can offer many advantages, there are also some potential disadvantages to consider:

Inaccuracy: Sleep tracking apps rely on sensors to monitor sleep patterns, which may not always be accurate. For example, movements during sleep can be misinterpreted as being awake, leading to inaccurate data.

Reliance on technology: Using a sleep tracking app may encourage users to rely too heavily on technology to improve their sleep habits, rather than addressing underlying issues that may be contributing to poor sleep.

Disruption of sleep: Some users may become overly focused on their sleep data, leading to anxiety or even disrupted sleep as they try to improve their sleep habits.

Privacy concerns: Sleep tracking apps collect sensitive data about users' sleep habits, which can raise privacy concerns if the data is shared or accessed without the user's consent.

Cost: Some sleep tracking apps may require a subscription or in-app purchases to access certain features, which can be a disadvantage for users who cannot afford to pay for these services.

It is important for users to weigh the potential advantages and disadvantages of using a sleep tracking app and decide whether it is a useful tool for improving their sleep habits. It is also important for users to be aware of any privacy concerns and to protect their personal data when using a sleep tracking app.

5.APPLICATIONS :

Sleep tracking apps can be applied in a variety of settings and industries, including:

Healthcare: Sleep tracking apps can be used in clinical settings to help diagnose and monitor sleep disorders, such as sleep apnea and insomnia.

Wellness: Sleep tracking apps can be used by individuals who are interested in optimizing their sleep habits and improving their overall health and well-being.

Fitness: Sleep is an important aspect of physical fitness and recovery, and sleep tracking apps can help athletes and fitness enthusiasts monitor their sleep patterns and make adjustments to their training and recovery plans as needed.

Workplaces: Sleep tracking apps can be used by employers to promote better sleep habits among employees and improve productivity and performance in the workplace.

Education: Sleep is essential for learning and academic performance, and sleep tracking apps can be used by students and educators to monitor and optimize their sleep habits.

Overall, sleep tracking apps can be applied in any setting where individuals or organizations are interested in monitoring and improving sleep habits to promote better health and well-being.

6.CONCLUSION :

In conclusion, sleep tracking apps can be a valuable tool for individuals and organizations looking to monitor and improve sleep habits. These apps offer several advantages, including increased awareness of sleep habits, personalized recommendations for improvement, and improved sleep quality, which can lead to better overall health outcomes. However, there are also potential disadvantages to consider, such as inaccuracy and privacy concerns. It is important for users to weigh the potential advantages and disadvantages and decide whether a sleep tracking app is a useful tool for improving their sleep habits. Sleep tracking apps can be applied in a variety of settings and industries, including healthcare, wellness, fitness, workplaces, and education.

7.FUTURE SCOPE :

The future scope of sleep tracking apps is quite promising. Here are some potential areas where sleep tracking apps may continue to evolve and make an impact:

Advanced sensors: As sensor technology continues to improve, sleep tracking apps may become even more accurate and reliable, allowing users to gain even more insights into their sleep habits.

Machine learning: Machine learning algorithms can help to identify patterns and provide personalized recommendations for improving sleep habits. This technology may become more prevalent in sleep tracking apps, leading to even more tailored and effective solutions.

Integration with other health technologies: Sleep tracking apps may become more integrated with other health technologies, such as fitness trackers and smart home devices, to provide a more comprehensive picture of an individual's health and well-being.

Telemedicine: Sleep tracking apps may become a key tool in telemedicine, allowing healthcare providers to remotely monitor and diagnose sleep disorders and provide personalized treatment plans.

Wearable technology: Sleep tracking apps may become even more convenient and accessible as they continue to be integrated with wearable technology, such as smartwatches and fitness bands.

Overall, sleep tracking apps have the potential to continue to evolve and make a significant impact in the future of healthcare, wellness, and personal health monitoring.

8.APPENDIX

A Source Code:

User class code:

```
package com.example.projectone

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

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

  )
```

UserDao interface code:

```
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 class code :

```
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 class code :

```
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(COLUM
N_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(CO
LUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(CO
LUMN_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(COLUM
N_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(CO
LUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(CO
LUMN_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(COLUM
N_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(CO
LUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(CO
LUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(CO
LUMN_PASSWORD)),
                )
                users. add(user)
            } while (cursor. moveToNext())
        }
        cursor. close()
        db. close()
        return users
    }

}
```

TimeLog data class code:

```
package com. example. projectone

import androidx. room. Entity
import androidx. room. PrimaryKey
import java. sql. Date

@Entity(tableName = "TimeLog")
data class TimeLog(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val startTime: Date,
    val stopTime: Date
)
```

TimeLogDao interface code:

```
package com.example.projectone

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

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

}
```

AppDatabase class code:

```
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
            }
        }
    }
}
```

TimeDatabaseHelper class code:

```
 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(COLUM
N_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLU
MN_START_TIME))
```

```
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLU
MN_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"
        }
    }
}
```

Database connection in LoginActivity.k

```
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.ProjectOneT
heme


  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)
  }
```

Database connection in RegistrationActivity.kt

```
  package com.example.projectone
```

```
 import android.content.Context
 import android.content.Intent
 import android.os.Bundle
 import
androidx.activity.ComponentActiv
ity
 import
androidx.activity.compose.setCon
tent
 import
androidx.compose.foundation.Imag
e
 import
androidx.compose.foundation.layo
ut.*
 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.Col
or
 import
androidx.compose.ui.layout.Conte
ntScale
```

```
 import
androidx.compose.ui.res.painterR
esource
 import
androidx.compose.ui.text.font.Fo
ntFamily
 import
androidx.compose.ui.text.font.Fo
ntWeight
 import
androidx.compose.ui.unit.dp
 import
androidx.compose.ui.unit.sp
 import
androidx.core.content.ContextCom
pat
 import
com.example.projectone.ui.theme.
ProjectOneTheme


 class MainActivity2 :
ComponentActivity() {
   private lateinit var
databaseHelper:
UserDatabaseHelper
   override fun
onCreate(savedInstanceState:
Bundle?) {

super.onCreate(savedInstanceStat
e)
      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, database
Helper)
            }
          }
        }
      }
    }


    @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 = "Login")
        }
      }
    }
 }
 private fun 
startLoginActivity(context: 
Context) {
    val intent = Intent(context, 
LoginActivity::class.java)

ContextCompat.startActivity(cont
ext, intent, null)
 }
```

MainActivity.kt :

```
 package com.example.projectone

 import android.content.Context
 import android.content.Intent
 import 
android.icu.text.SimpleDateForma
t
 import android.os.Bundle
 import 
androidx.activity.ComponentActiv
ity
```

```
 import
androidx.activity.compose.setCon
tent
 import
androidx.compose.foundation.Imag
e
 import
androidx.compose.foundation.layo
ut.*
 import
androidx.compose.material.Button
 import
androidx.compose.material.Materi
alTheme
 import
androidx.compose.material.Surfac
e
 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.Conte
ntScale
 import
androidx.compose.ui.res.painterR
esource
```

```
 import
androidx.compose.ui.unit.dp
 import
androidx.core.content.ContextCom
pat
 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(savedInstanceStat
e)
        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(star
tTime)
       }
     } else {
       Button(onClick = {
         elapsedTime =
System.currentTimeMillis()
         isRunning = false
       }) {
         Text("Stop")

databaseHelper.addTimeLog(elapse
dTime, 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(cont
ext, intent, null)
 }
 fun getCurrentDateTime(): String
{
    val dateFormat =
SimpleDateFormat("yyyy-MM-dd
HH:mm:ss", Locale.getDefault())
    val currentTime =
System.currentTimeMillis()
    return
dateFormat.format(Date(currentTi
me))
 }

 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)
 }
```

Database connection and fetching in TrackActivity.k:

```
 package com.example.projectone

 import
android.icu.text.SimpleDateForma
t
```

```
 import android.os.Bundle
 import android.util.Log
 import
androidx.activity.ComponentActiv
ity
 import
androidx.activity.compose.setCon
tent
 import
androidx.compose.foundation.Imag
e
 import
androidx.compose.foundation.layo
ut.*
 import
androidx.compose.foundation.lazy
.LazyColumn
 import
androidx.compose.foundation.lazy
.LazyRow
 import
androidx.compose.foundation.lazy
.items
 import
androidx.compose.material.Materi
alTheme
 import
androidx.compose.material.Surfac
e
 import
androidx.compose.material.Text
 import
androidx.compose.runtime.Composa
ble
```

```
 import
androidx.compose.ui.Modifier
 import
androidx.compose.ui.draw.alpha
 import
androidx.compose.ui.graphics.Col
or
 import
androidx.compose.ui.layout.Conte
ntScale
 import
androidx.compose.ui.res.painterR
esource
 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(savedInstanceStat
e)

      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.TimeL
og>) {
   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.startTi
me)}")
                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
))
 }
```

Complete AndroidManifest.xml
code:

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

xmlns:tools="http://schemas.andr
oid.com/tools">

    <application
       android:allowBackup="true"

android:dataExtractionRules="@xm
l/data_extraction_rules"

android:fullBackupContent="@xml/
backup_rules"

android:icon="@mipmap/ic_launche
r"

android:label="@string/app_name"
       android:supportsRtl="true"

android:theme="@style/Theme.Proj
ectOne"
       tools:targetApi="31">
       <activity

android:name=".TrackActivity"
          android:exported="false"
```

```
android:label="@string/title_act
ivity_track"

android:theme="@style/Theme.Proj
ectOne" />
      <activity

android:name=".MainActivity"
        android:exported="false"

android:label="@string/app_name"

android:theme="@style/Theme.Proj
ectOne" />
      <activity

android:name=".MainActivity2"
        android:exported="false"

android:label="RegisterActivity"

android:theme="@style/Theme.Proj
ectOne" />
      <activity

android:name=".LoginActivity"
        android:exported="true"

android:label="@string/app_name"

android:theme="@style/Theme.Proj
ectOne">
        <intent-filter>
```

```
            <action
android:name="android.intent.act
ion.MAIN" />

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

 </manifest>
```