A Sleep Tracking App for a Better Night's Rest

1.INTRODUCTION

Overview

Here's a simple sleeping track that you can follow to help you ease into a better night's sleep:

Set a consistent bedtime and wake-up time: Try to go to bed and wake up at the same time every day, even on weekends.

Create a relaxing pre-sleep routine: Spend the last hour or so before bed doing relaxing activities such as taking a warm bath, reading a book, or listening to calming music.

Avoid stimulating activities before bed: Avoid using electronic devices such as smartphones, tablets, or laptops in the hour before bed, as the blue light can interfere with your sleep.

Keep your bedroom cool, dark, and quiet: Make sure your bedroom is cool and comfortable, with minimal noise and light to create a relaxing sleep environment.

Avoid caffeine, alcohol, and large meals before bedtime: Try to avoid consuming caffeine or alcohol in the evening, as well as large meals, as they can interfere with your sleep.

Use relaxation techniques: Practice relaxation techniques such as deep breathing, meditation, or progressive muscle relaxation to help you unwind and calm your mind before bed.

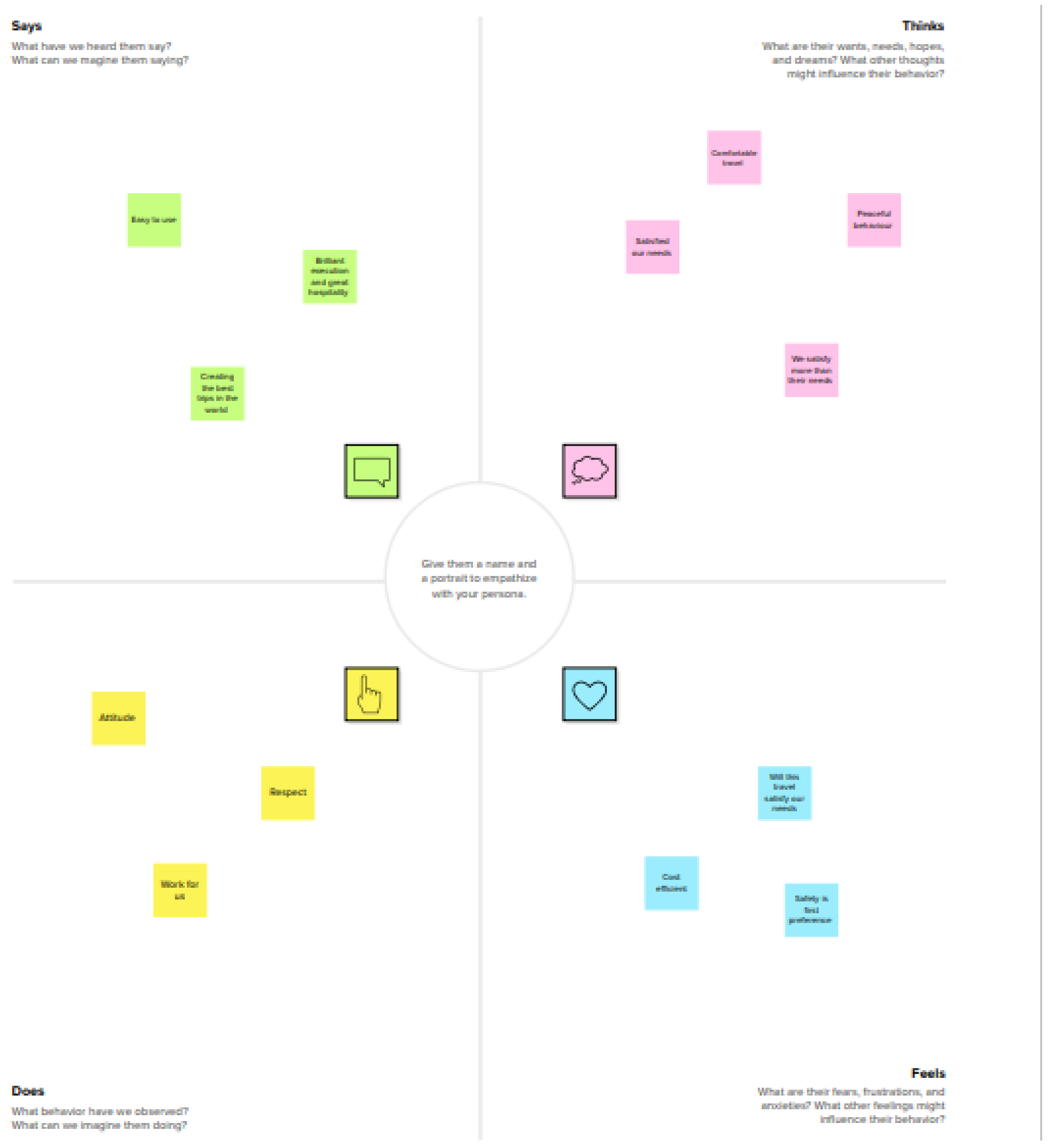
Consider using white noise or a sleep-inducing soundtrack: If you find it difficult to fall asleep, try using white noise or a sleep-inducing soundtrack to help you relax and drift off to sleep.

Purpose

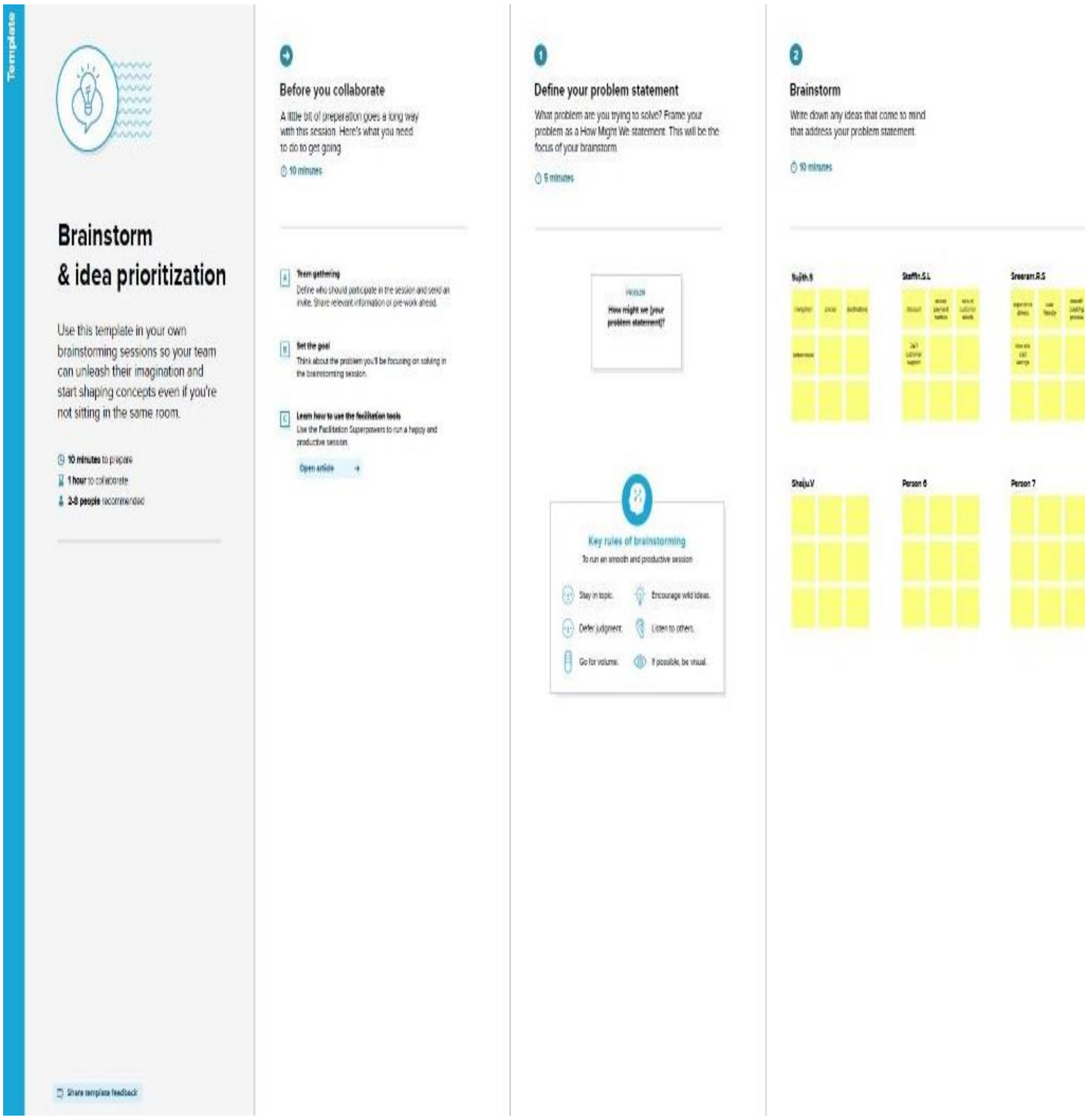
The purpose of the sleeping track I provided is to help individuals establish healthy habits and behaviors that promote better sleep. Getting a good night's sleep is essential for overall health and wellbeing, and following a consistent sleeping routine can improve sleep quality, increase daytime alertness, and enhance mental and physical performance. By incorporating these tips into your daily routine, you can create a sleep-conducive environment and improve the quantity and quality of your sleep, leading to a happier, healthier, and more productive life.

2.Problem Definition & Design Thinking

2.1 Empathy Map


Says
What have we heard them say?
What can we magine them saying?
Easy to use
Brilliant execution and great hospitality
Creating the best trips in the world
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
Comfortable travel
Peaceful behaviour
Satisfied our needs
We satisfy more than their needs
Give them a name and a portrait to empathize with your persona.
Attitude
Respect
Work for us
Does
What behavior have we observed?
What can we imagine them doing?
Will this travel satisfy our needs
Cost efficient
Safety is first preference
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

2.2 Ideation and 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

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]?

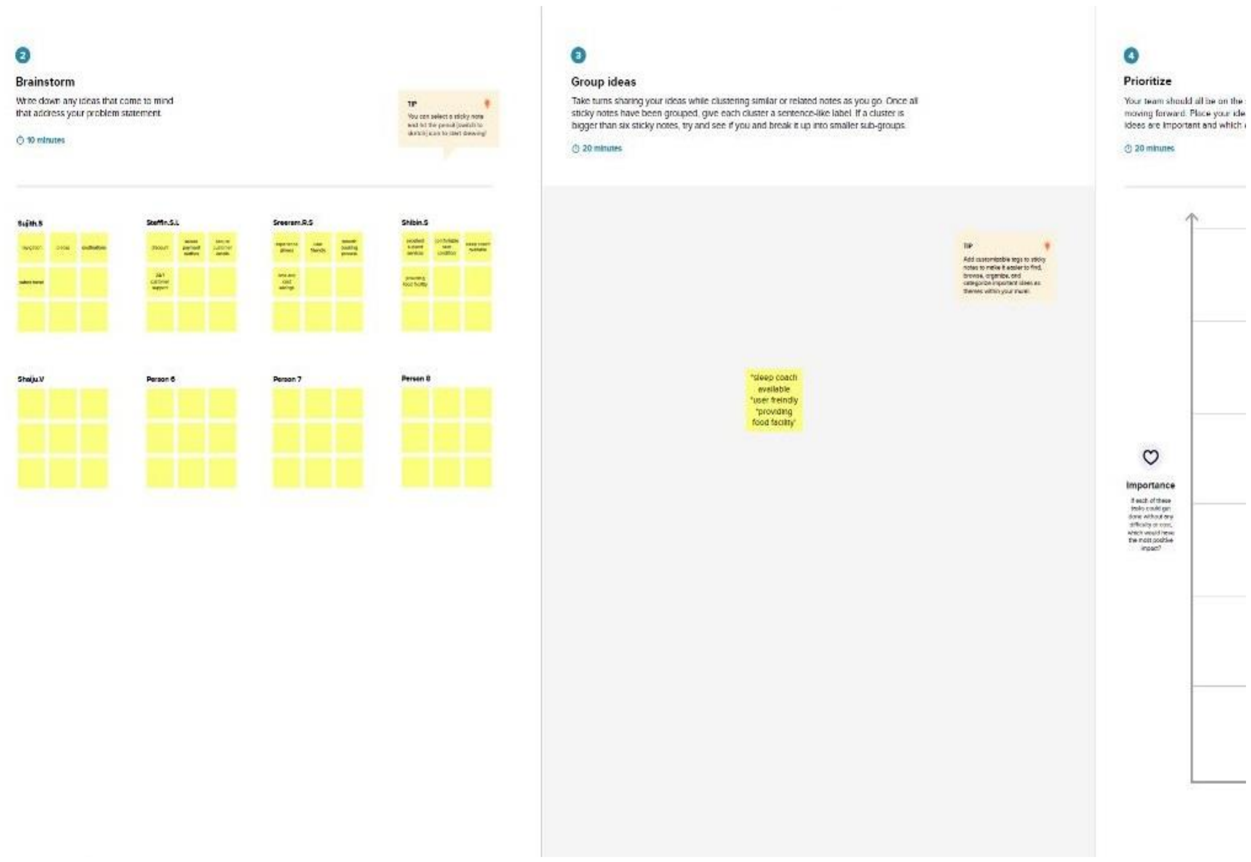
Key rules of brainstorming
To run an smooth and productive session

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



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!

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

TIP

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

*sleep coach available
*user freindly
*providing food facility'

4

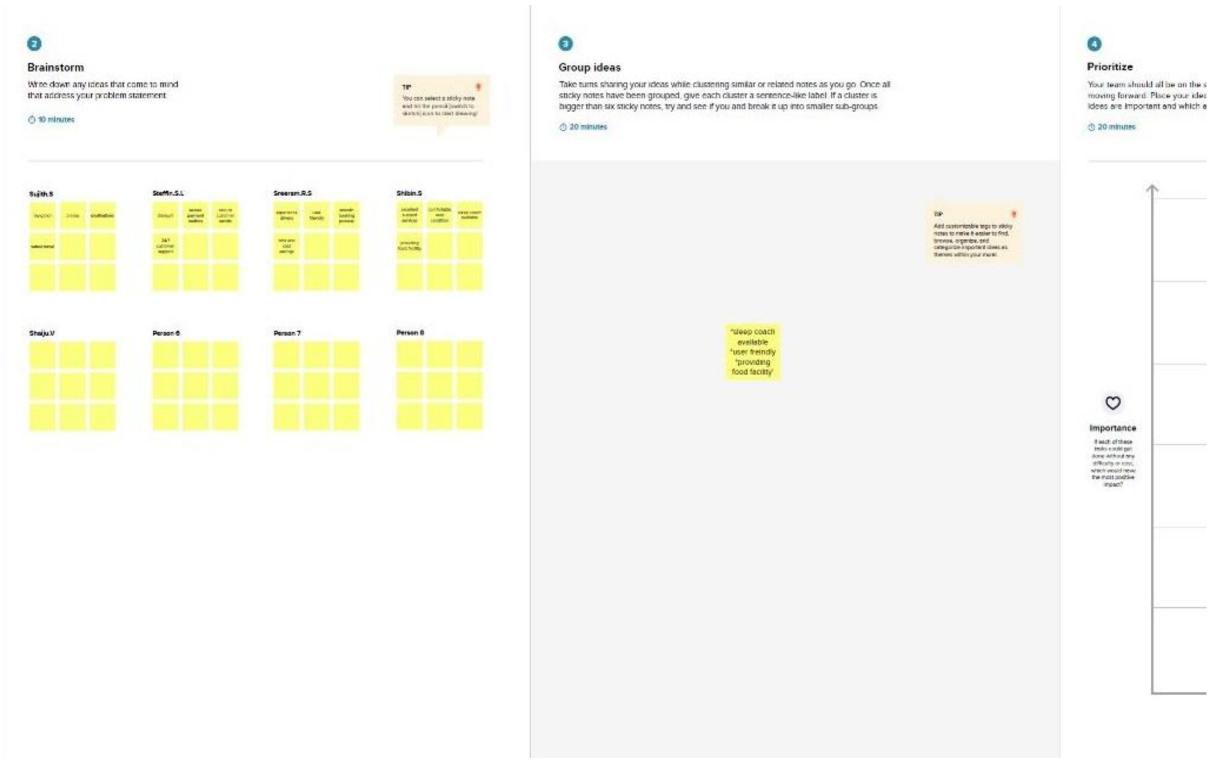
Prioritize

Your team should all be on the
moving forward. Place your ide
ideas are important and which

20 minutes

Importance

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



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!

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

TIP

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

*sleep coach available
*user freindly
*providing food facility'

4

Prioritize

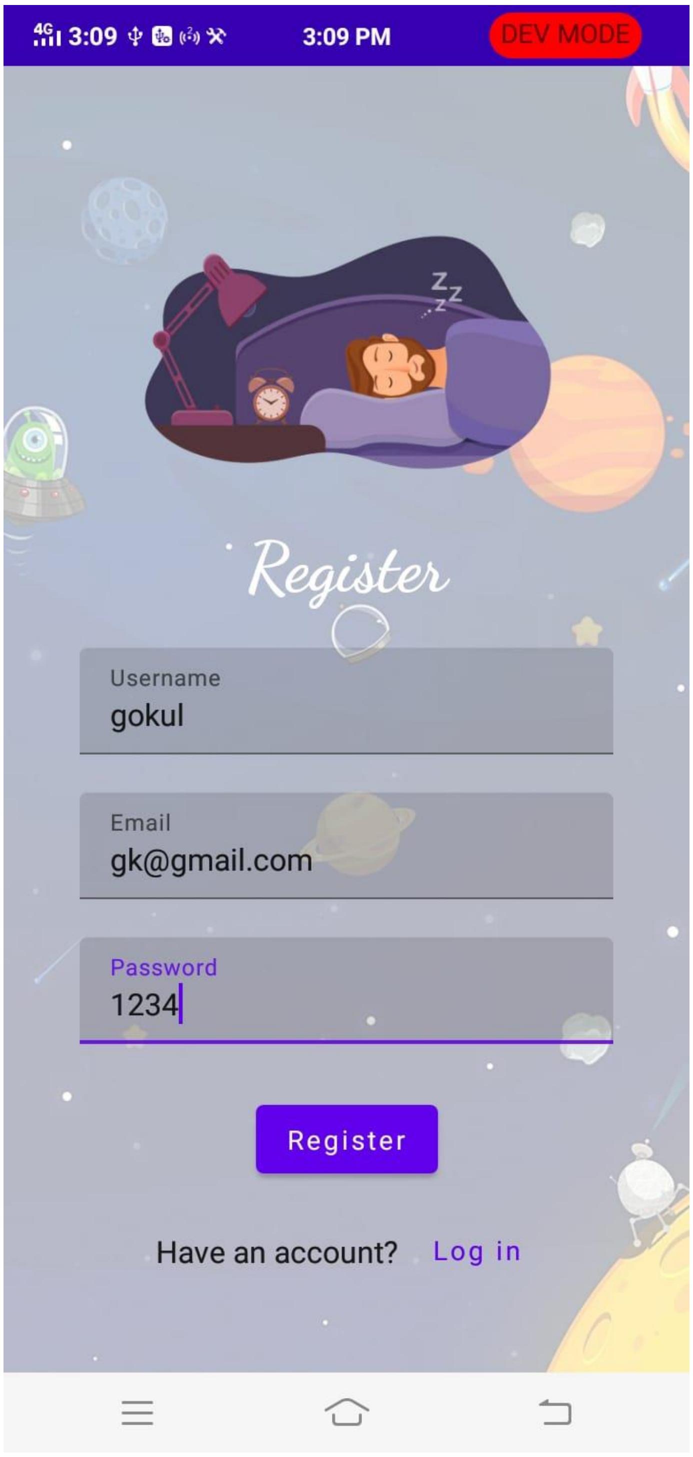
Your team should all be on the
moving forward. Place your ide
ideas are important and which

20 minutes

Importance

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

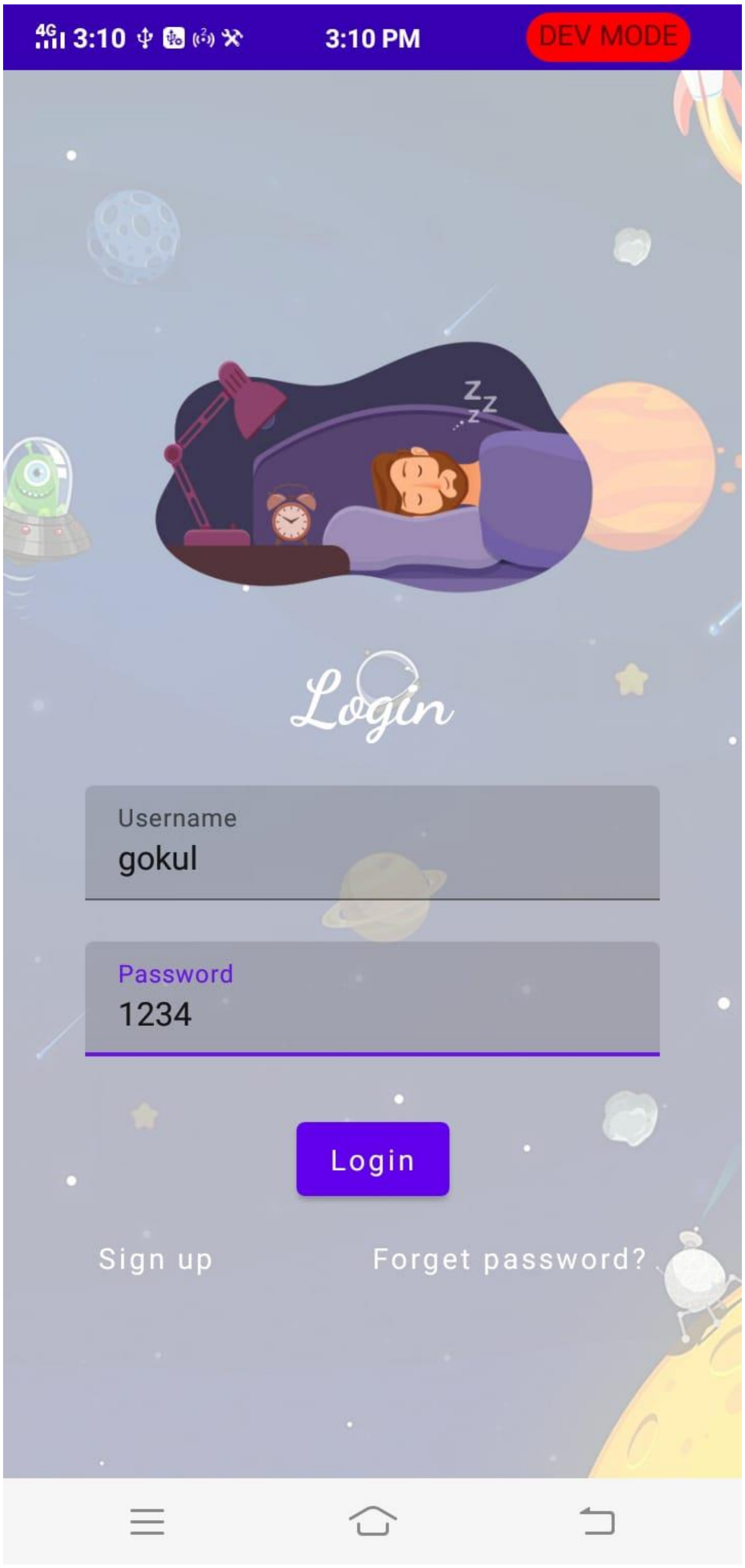
RESULT



Username
gokul

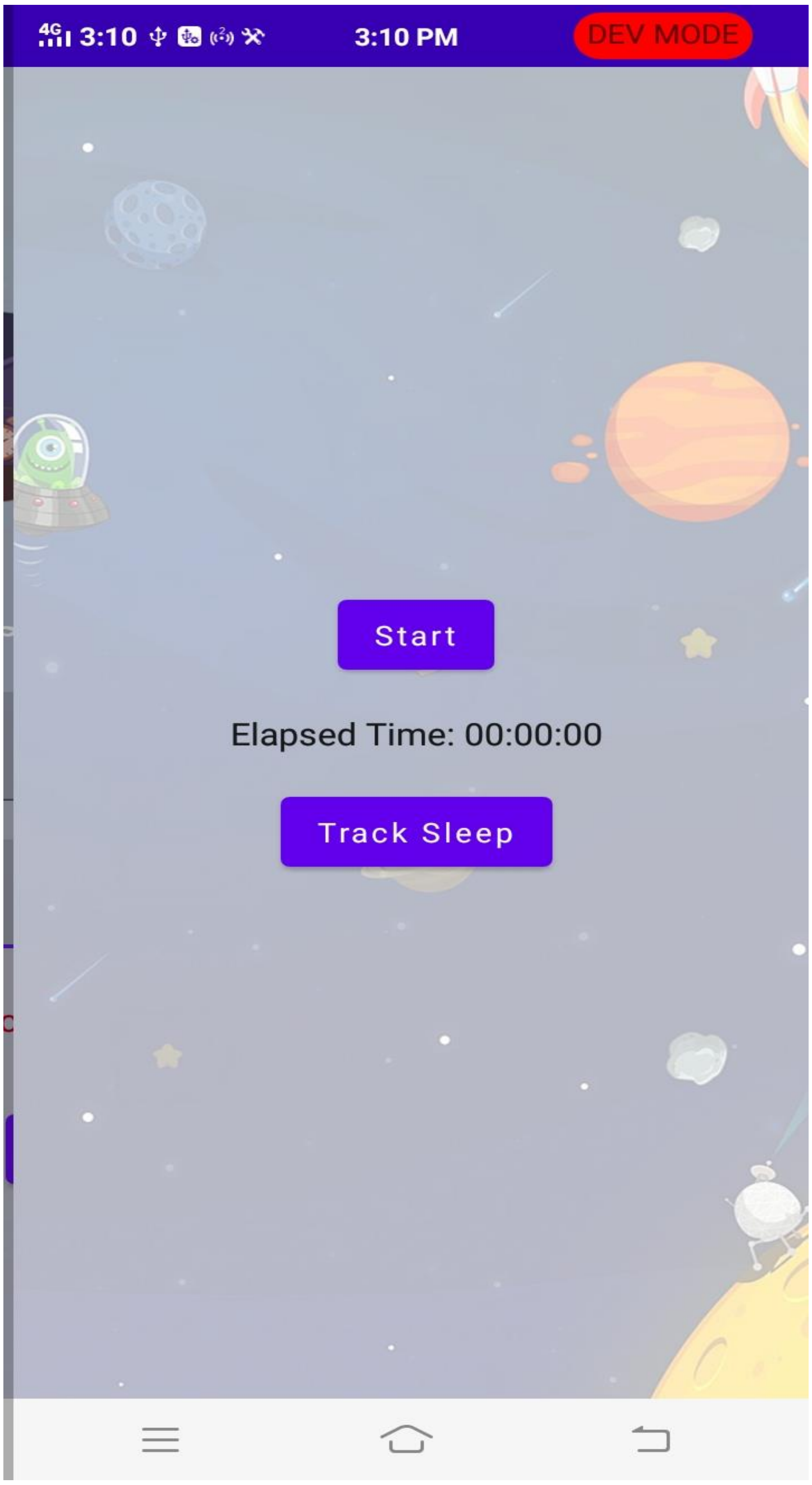
Email
gk@gmail.com

Password
1234

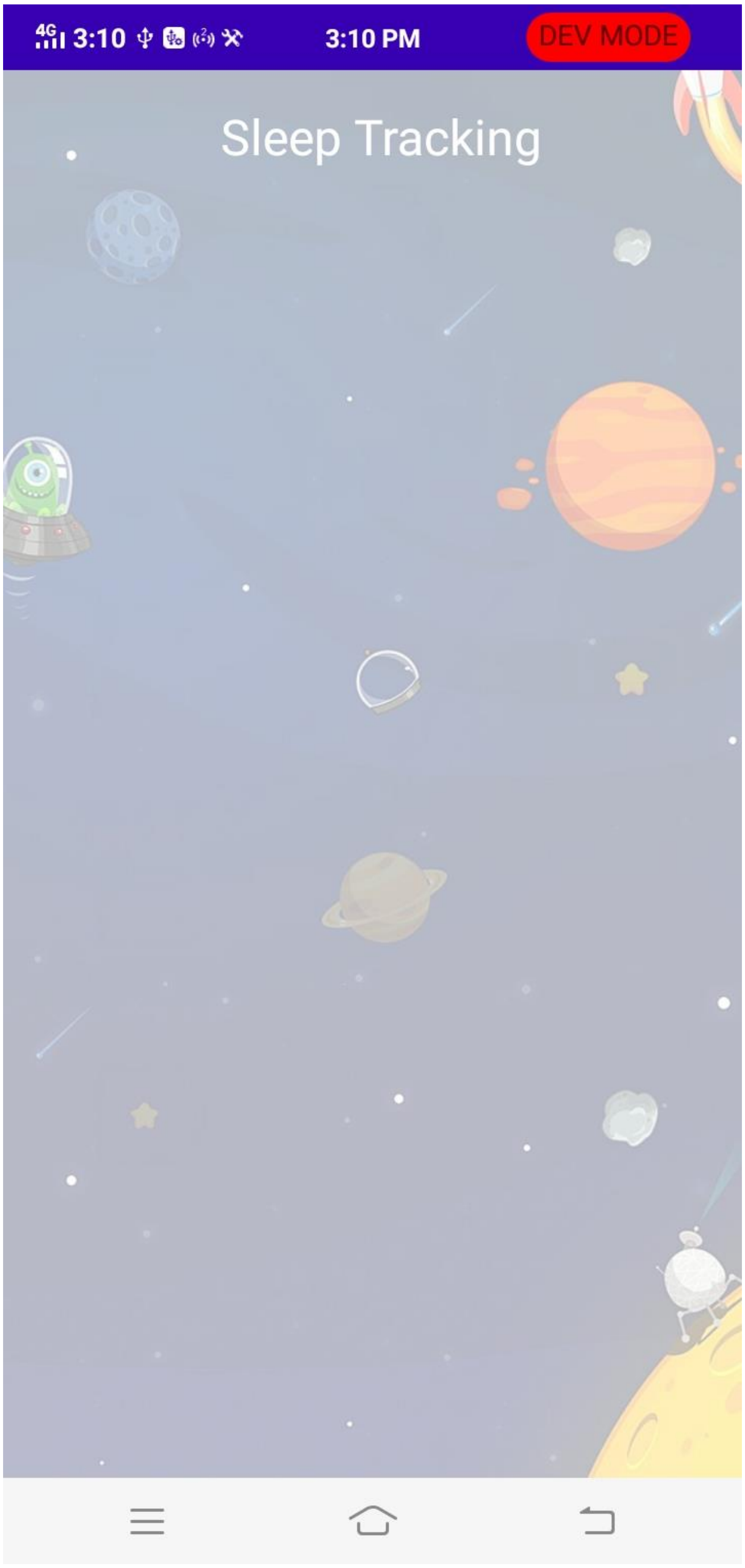


Username
gokul

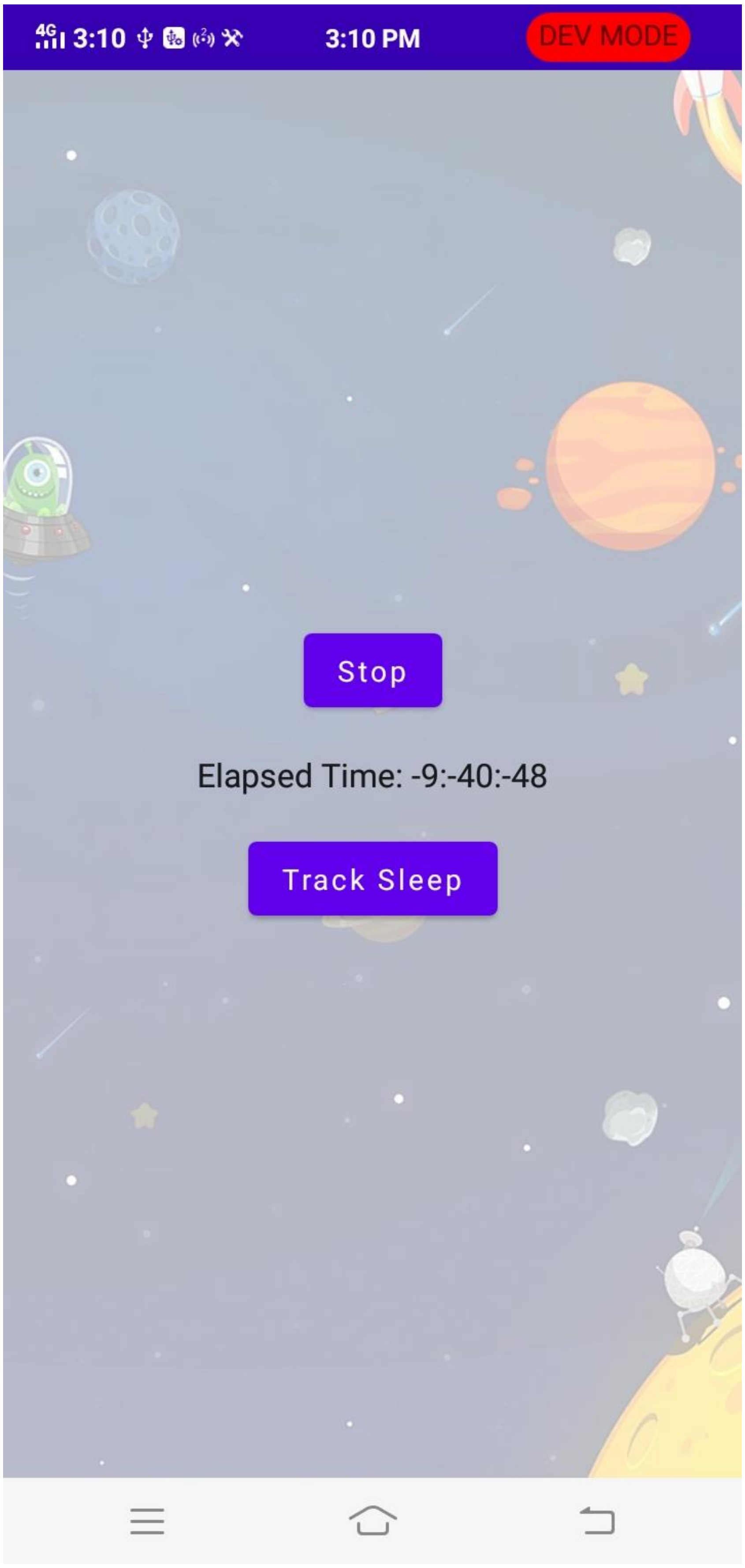
Password
1234



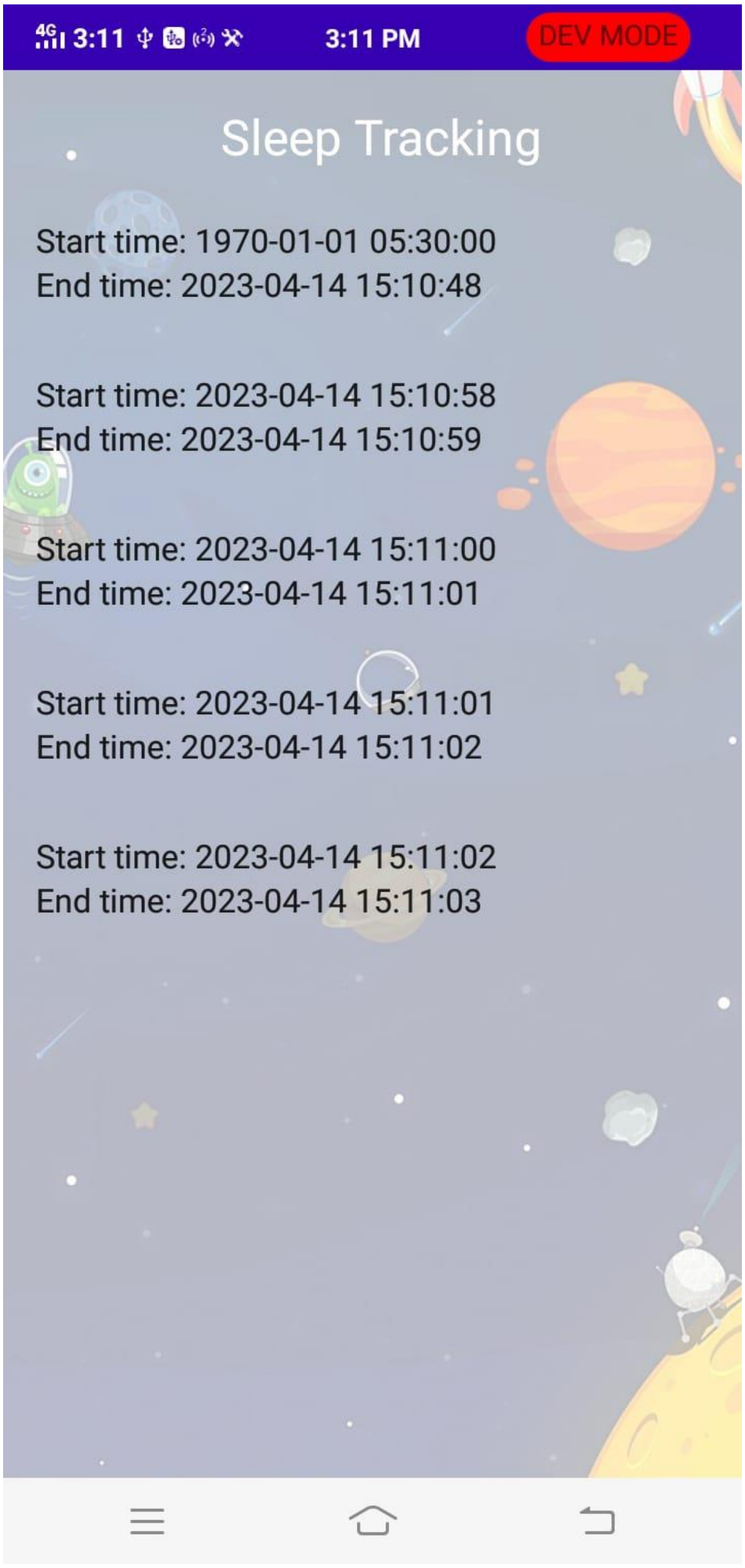
Elapsed Time: 00:00:00



Sleep Tracking



Elapsed Time: -9:-40:-48



Sleep Tracking

Start time: 1970-01-01 05:30:00
End time: 2023-04-14 15:10:48

Start time: 2023-04-14 15:10:58
End time: 2023-04-14 15:10:59

Start time: 2023-04-14 15:11:00
End time: 2023-04-14 15:11:01

Start time: 2023-04-14 15:11:01
End time: 2023-04-14 15:11:02

Start time: 2023-04-14 15:11:02
End time: 2023-04-14 15:11:03

ADVANTAGES & DISADVANTAGES

Advantages

- Advantages of technology: Technology has made our lives easier and more convenient by providing us with faster and more efficient ways of doing things. It has improved communication, transportation, healthcare, and many other areas of life.

- Advantages of education: Education is essential for personal growth and development. It provides individuals with knowledge and skills that are necessary for success in life. Education also opens up opportunities for better jobs and higher salaries.

- Advantages of teamwork: Working in teams can be beneficial for achieving common goals. It allows individuals to share ideas, skills, and resources, and can lead to more creative solutions. Teamwork also promotes better communication and collaboration.

- Advantages of diversity: Diversity in a workplace or community can lead to a range of benefits, including increased creativity, improved problem-solving, and better decision-making. It can also promote a more inclusive and tolerant society.

- Advantages of exercise: Regular exercise has numerous benefits, including improved physical health, mental health, and overall well-being. It can reduce the risk of chronic diseases such as diabetes, heart disease, and obesity, and can also improve mood and cognitive function.

Disadvantages

- Disadvantages of technology: Technology can be addictive and can lead to a sedentary lifestyle. It can also be expensive and may contribute to social isolation. Additionally, technology can create security and privacy concerns.

- Disadvantages of education: Education can be expensive and not accessible to everyone. It can also be stressful and may lead to academic pressure and anxiety. Additionally, education can sometimes create a sense of elitism and social inequality.

- Disadvantages of teamwork: Working in teams can sometimes lead to conflict and disagreements. It can also be difficult to manage team dynamics and ensure equal participation. Additionally, team members may have different priorities and goals.

- Disadvantages of diversity: Diversity can sometimes lead to misunderstandings and cultural clashes. It can also create challenges in communication and decision-making.

Additionally, diversity may sometimes lead to discrimination and prejudice.

- Disadvantages of exercise: Exercise can sometimes lead to injuries or health issues if done improperly. It can also be time-consuming and difficult to maintain a regular routine. Additionally, exercise may not be accessible to everyone due to physical limitations or financial constraints.

APPLICATIONS

- Customized Travel Planning: With a personalized travel planning app, users can create custom travel itineraries that cater to their preferences and interests. Users can input details about their budget, preferred activities, and travel dates, and the app will provide recommendations for accommodations, attractions, restaurants, and other activities that suit their interests.

- Real-time Tracking: Travelers can use the app to track their itinerary, monitor flight or train schedules, and receive real-time updates about changes or delays. This can help them stay on schedule and make necessary adjustments on the go.

- Location-based Recommendations: The app can use location data to provide personalized recommendations for nearby attractions, restaurants, and events. This feature can help travellers discover new places and experiences they may not have otherwise known about.

- Social Sharing: Users can share their travel plans, experiences, and recommendations with friends and family through social media or within the app. This can help build a community of travellers and provide useful insights and tips for others planning similar trips.

CONCLUSION

In conclusion, a Personalized Travel Planning and Tracking App developed in android studio can offer a wide range of benefits for both travellers and tourism industry professionals. With features such as customized travel planning, real-time tracking, location-based recommendations, social sharing, and analytics, the app can help users create personalized itineraries that cater to their interests and preferences, while also providing valuable insights for tourism industry professionals. As more people turn to technology for travel planning and organization, the development of such apps is likely to become increasingly important in the tourism industry.

FUTURE SCOPE

- Artificial Intelligence and Machine Learning: Integrating AI and machine learning capabilities into the app could allow it to learn from user behaviour and provide even more personalized recommendations and travel plans.

- Virtual Reality and Augmented Reality: Incorporating VR and AR technologies into the app could allow users to experience destinations and attractions virtually before they arrive, helping them make more informed travel decisions.

- Blockchain Technology: Implementing blockchain technology could help enhance the security and privacy of user data and transactions, which is especially important in the travel industry.

- Smart City Integration: Integrating with smart city technologies could allow the app to provide even more detailed and accurate recommendations based on real-time data about traffic, weather, and other factors.

- Sustainability and Responsible Tourism: As more travellers become conscious of the impact of their travel on the environment and local communities, the app could integrate features that promote sustainable and responsible tourism practices, such as eco-friendly accommodations and tours.

Overall, the future scope of a Personalized Travel Planning and Tracking App developed in android studio is vast, and there are many opportunities to continue enhancing and improving the app to meet the evolving needs of travellers and the tourism industry.

Appendix

A. Source code

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.ProjectOne"
            tools:targetApi="31">
            <activity
            android:name=".TrackActivity"
            android:exported="false"
            android:label="@string/title_activity_track"
            android:theme="@style/Theme.ProjectOne" />
            <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="@string/app_name"
            android:theme="@style/Theme.ProjectOne" />
            <activity
            android:name=".MainActivity2"
            android:exported="false"
            android:label="RegisterActivity"
            android:theme="@style/Theme.ProjectOne" />
            <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.ProjectOne">
            <intent-filter>
            <action android:name="android.intent.action.MAIN"
/>


            <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
        </application>


        </manifest>
```

AppDatabase.kt

```
        package com.example.projectone


        import android.content.Context
        import androidx.room.Database
        import androidx.room.Room
        import androidx.room.RoomDatabase


        @Database(entities = [TimeLog::class], version = 1,
exportSchema = false)
        abstract class AppDatabase : RoomDatabase() {


        abstract fun timeLogDao(): TimeLogDao
        companion object {
        private var INSTANCE: AppDatabase? = null


        fun getDatabase(context: Context): AppDatabase {
        val tempInstance = INSTANCE
        if (tempInstance != null) {
        return tempInstance
        }
        synchronized(this) {
        val instance = Room.databaseBuilder(
        context.applicationContext,
        AppDatabase::class.java,
        "app_database"
        ).build()
        INSTANCE = instance
        return instance
        }
        }
        }
        }
```

LoginActivity.kt

```
package com.example.projectone


import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import
com.example.projectone.ui.theme.ProjectOneTheme




class LoginActivity : ComponentActivity() {
private lateinit var databaseHelper:
UserDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = UserDatabaseHelper(this)
setContent {
ProjectOneTheme {
// A surface container using the 'background' color from
the theme
Surface(
modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colors.background
) {
LoginScreen(this, databaseHelper)
}
}
}
```

```
}
}
@Composable
fun LoginScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
var username by remember { mutableStateOf("") }
var password by remember { mutableStateOf("") }
var error by remember { mutableStateOf("") }
val imageModifier = Modifier
Image(
painterResource(id = R.drawable.sleeptracking),
contentScale = ContentScale.FillHeight,
contentDescription = "",
modifier = imageModifier
.alpha(0.3F),
)
Column(
modifier = Modifier.fillMaxSize(),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
) {


Image(
painter = painterResource(id = R.drawable.sleep),
contentDescription = "",


modifier = imageModifier
.width(260.dp)
.height(200.dp)
)
Text(
fontSize = 36.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.White,
text = "Login"
)
Spacer(modifier = Modifier.height(10.dp))


TextField(
value = username,
onValueChange = { username = it },
label = { Text("Username") },
modifier = Modifier.padding(10.dp)
.width(280.dp)
```

```
)


TextField(
value = password,
onValueChange = { password = it },
label = { Text("Password") },
modifier = Modifier.padding(10.dp)
.width(280.dp)
)


if (error.isNotEmpty()) {
Text(
text = error,
color = MaterialTheme.colors.error,
modifier = Modifier.padding(vertical = 16.dp)
)
}


Button(
onClick = {
```

```
            if (username.isNotEmpty() &&
password.isNotEmpty()) {
            val user =
databaseHelper.getUserByUsername(username)
            if (user != null && user.password == password) {
            error = "Successfully log in"
            context.startActivity(
            Intent(
            context,
            MainActivity::class.java
            )
            )


            //onLoginSuccess()
            } else {
            error = "Invalid username or password"
            }
            } else {
            error = "Please fill all fields"
            }
            },
            modifier = Modifier.padding(top = 16.dp)
            ) {
```

```
Text(text = "Login")
}
Row {
TextButton(onClick = {context.startActivity(
Intent(
context,
MainActivity2::class.java
)
)}
)
{ Text(color = Color.White,text = "Sign up") }
TextButton(onClick = {
/*startActivity(
Intent(
applicationContext,
MainActivity2::class.java
)
)*/
})


{
Spacer(modifier = Modifier.width(60.dp))
Text(color = Color.White,text = "Forget password?")
}
}
}
}
private fun startMainPage(context: Context) {
val intent = Intent(context, MainActivity2::class.java)
ContextCompat.startActivity(context, intent, null)
}
```

MainActivity.kt

```
package com.example.projectone



import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Button
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
```

```
            import androidx.compose.material.Text
            import androidx.compose.runtime.*
            import androidx.compose.ui.Alignment
            import androidx.compose.ui.Modifier
            import androidx.compose.ui.draw.alpha
            import androidx.compose.ui.layout.ContentScale
            import androidx.compose.ui.res.painterResource
            import androidx.compose.ui.unit.dp
            import androidx.core.content.ContextCompat
            import
com.example.projectone.ui.theme.ProjectOneTheme
            import java.util.*


            class MainActivity : ComponentActivity() {


            private lateinit var databaseHelper:
TimeLogDatabaseHelper


            override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            databaseHelper = TimeLogDatabaseHelper(this)
```

```
databaseHelper.deleteAllData()
setContent {
ProjectOneTheme {
// A surface container using the 'background' color from
the theme
Surface(
modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colors.background
) {
MyScreen(this,databaseHelper)
}
}
}
}
}
@Composable
fun MyScreen(context: Context, databaseHelper:
TimeLogDatabaseHelper) {
var startTime by remember { mutableStateOf(0L) }
var elapsedTime by remember { mutableStateOf(0L) }
var isRunning by remember { mutableStateOf(false) }
val imageModifier = Modifier
Image(
painterResource(id = R.drawable.sleeptracking),
contentScale = ContentScale.FillHeight,
contentDescription = "",
modifier = imageModifier
.alpha(0.3F),
)


Column(
modifier = Modifier.fillMaxSize(),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
) {
if (!isRunning) {
Button(onClick = {
startTime = System.currentTimeMillis()
isRunning = true
}) {
Text("Start")
//databaseHelper.addTimeLog(startTime)
}
} else {
Button(onClick = {
elapsedTime = System.currentTimeMillis()
isRunning = false
```

```
            }) {
            Text("Stop")
            databaseHelper.addTimeLog(elapsedTime,startTime)
            }
            }
            Spacer(modifier = Modifier.height(16.dp))
            Text(text = "Elapsed Time: ${formatTime(elapsedTime
- startTime)}")




            Spacer(modifier = Modifier.height(16.dp))
            Button(onClick = { context.startActivity(
            Intent(
            context,
            TrackActivity::class.java
            )
            ) }) {
            Text(text = "Track Sleep")
            }


            }
```

```
}



private fun startTrackActivity(context: Context) {
val intent = Intent(context, TrackActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}
fun getCurrentDateTime(): String {
val dateFormat = SimpleDateFormat("yyyy-MM-dd
HH:mm:ss", Locale.getDefault())
val currentTime = System.currentTimeMillis()
return dateFormat.format(Date(currentTime))
}


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

RegisterActivity.kt

```
package com.example.projectone


import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import
com.example.projectone.ui.theme.ProjectOneTheme




class MainActivity2 : ComponentActivity() {
private lateinit var databaseHelper:
UserDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = UserDatabaseHelper(this)
setContent {
ProjectOneTheme {
// A surface container using the 'background' color from
the theme
Surface(
modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colors.background
) {
```

```
RegistrationScreen(this,databaseHelper)
}
}
}
}
}



@Composable
fun RegistrationScreen(context: Context,
databaseHelper: UserDatabaseHelper) {
var username by remember { mutableStateOf("") }
var password by remember { mutableStateOf("") }
var email by remember { mutableStateOf("") }
var error by remember { mutableStateOf("") }


val imageModifier = Modifier
Image(
painterResource(id = R.drawable.sleeptracking),
contentScale = ContentScale.FillHeight,
contentDescription = "",
modifier = imageModifier
.alpha(0.3F),
)
Column(
modifier = Modifier.fillMaxSize(),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
) {


Image(
painter = painterResource(id = R.drawable.sleep),
contentDescription = "",


modifier = imageModifier
.width(260.dp)
.height(200.dp)
)
Text(
fontSize = 36.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
color = Color.White,
text = "Register"
)


Spacer(modifier = Modifier.height(10.dp))
TextField(
value = username,
onValueChange = { username = it },
label = { Text("Username") },
modifier = Modifier
.padding(10.dp)
.width(280.dp)


)


TextField(
value = email,
onValueChange = { email = it },
label = { Text("Email") },
modifier = Modifier
```

```
.padding(10.dp)
.width(280.dp)
)


TextField(
value = password,
onValueChange = { password = it },
label = { Text("Password") },
modifier = Modifier
.padding(10.dp)
.width(280.dp)
)




if (error.isNotEmpty()) {
Text(
text = error,
color = MaterialTheme.colors.error,
modifier = Modifier.padding(vertical = 16.dp)
)
}
```

```
Button(
onClick = {
if (username.isNotEmpty() &&
password.isNotEmpty() && email.isNotEmpty()) {
val user = User(
id = null,
firstName = username,
lastName = null,
email = email,
password = password
)
databaseHelper.insertUser(user)
error = "User registered successfully"
// Start LoginActivity using the current context
context.startActivity(
Intent(
context,
LoginActivity::class.java
)
)
} else {
error = "Please fill all fields"
}
},
modifier = Modifier.padding(top = 16.dp)
) {
Text(text = "Register")
}
Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))


Row() {
Text(
modifier = Modifier.padding(top = 14.dp), text =
"Have an account?"
)
TextButton(onClick = {


})


{
```

```
Spacer(modifier = Modifier.width(10.dp))
Text(text = "Log in")
}
}
}
}
private fun startLoginActivity(context: Context) {
val intent = Intent(context, LoginActivity::class.java)
ContextCompat.startActivity(context, intent, null)
}
```

TimeDatabaseHelper.kt

```
package com.example.projectone


import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import java.util.*



class TimeLogDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, null,
DATABASE_VERSION) {
companion object {
private const val DATABASE_NAME = "timelog.db"
private const val DATABASE_VERSION = 1
const val TABLE_NAME = "time_logs"
private const val COLUMN_ID = "id"
const val COLUMN_START_TIME = "start_time"
const val COLUMN_END_TIME = "end_time"



// Database creation SQL statement
private const val DATABASE_CREATE =
"create table $TABLE_NAME ($COLUMN_ID
integer primary key autoincrement, " +
"$COLUMN_START_TIME integer not null,
$COLUMN_END_TIME integer);"
}
```

```
            override fun onCreate(db: SQLiteDatabase?) {
            db?.execSQL(DATABASE_CREATE)
            }


            override fun onUpgrade(db: SQLiteDatabase?,
oldVersion: Int, newVersion: Int) {
            db?.execSQL("DROP TABLE IF EXISTS
$TABLE_NAME")
            onCreate(db)
            }


            // function to add a new time log to the database
            fun addTimeLog(startTime: Long, endTime: Long) {
            val values = ContentValues()
            values.put(COLUMN_START_TIME, startTime)
            values.put(COLUMN_END_TIME, endTime)
            writableDatabase.insert(TABLE_NAME, null, values)
            }


            // function to get all time logs from the database
            @SuppressLint("Range")
            fun getTimeLogs(): List<TimeLog> {
            val timeLogs = mutableListOf<TimeLog>()
            val cursor = readableDatabase.rawQuery("select * from
$TABLE_NAME", null)
            cursor.moveToFirst()
            while (!cursor.isAfterLast) {
            val id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID))
            val startTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_START_TIM
E))
            val endTime =
cursor.getLong(cursor.getColumnIndex(COLUMN_END_TIME)
)
            timeLogs.add(TimeLog(id, startTime, endTime))
            cursor.moveToNext()
            }
            cursor.close()
            return timeLogs
            }


            fun deleteAllData() {
            writableDatabase.execSQL("DELETE FROM
$TABLE_NAME")
            }
            fun getAllData(): Cursor? {
            val db = this.writableDatabase
            return db.rawQuery("select * from $TABLE_NAME",
null)
            }


            data class TimeLog(val id: Int, val startTime: Long, val
endTime: Long?) {
            fun getFormattedStartTime(): String {
            return Date(startTime).toString()
            }


            fun getFormattedEndTime(): String {
            return endTime?.let { Date(it).toString() } ?: "not
ended"
            }
            }
            }
```

TimeLog.kt

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

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


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

TimeLogDao.kt

```
package com.example.projectone


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

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


}
```

TrackActivity.kt

```
package com.example.projectone


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


class TrackActivity : ComponentActivity() {


private lateinit var databaseHelper:
TimeLogDatabaseHelper


override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = TimeLogDatabaseHelper(this)
setContent {
ProjectOneTheme {
// A surface container using the 'background' color from
the theme
Surface(
modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colors.background
) {
//ListListScopeSample(timeLogs)


val data=databaseHelper.getTimeLogs();
Log.d("Sandeep" ,data.toString())
val timeLogs = databaseHelper.getTimeLogs()
ListListScopeSample(timeLogs)
}
}
}
}
}
```

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


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



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



}
}
```

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

User.kt

```
package com.example.projectone


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


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


)
UserDao.kt

package com.example.projectone


import androidx.room.*


@Dao
interface UserDao {


@Query("SELECT * FROM user_table WHERE email
= :email")
suspend fun getUserByEmail(email: String): User?


@Insert(onConflict = OnConflictStrategy.REPLACE)
suspend fun insertUser(user: User)
```

```
@Update
suspend fun updateUser(user: User)


@Delete
suspend fun deleteUser(user: User)
}
```

UserDatabase.kt

```
package com.example.projectone


import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase


@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
```

```
abstract fun userDao(): UserDao



companion object {



@Volatile
private var instance: UserDatabase? = null



fun getDatabase(context: Context): UserDatabase {
return instance ?: synchronized(this) {
val newInstance = Room.databaseBuilder(
context.applicationContext,
UserDatabase::class.java,
"user_database"
).build()
instance = newInstance
newInstance
}
}
}
}
```

UserDatabaseHelper.kt

```
package com.example.projectone

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME,
null, DATABASE_VERSION) {

companion object {
private const val DATABASE_VERSION = 1
private const val DATABASE_NAME =
"UserDatabase.db"

private const val TABLE_NAME = "user_table"
```

```
            private const val COLUMN_ID = "id"
            private const val COLUMN_FIRST_NAME =
"first_name"
            private const val COLUMN_LAST_NAME =
"last_name"
            private const val COLUMN_EMAIL = "email"
            private const val COLUMN_PASSWORD =
"password"
            }



            override fun onCreate(db: SQLiteDatabase?) {
            val createTable = "CREATE TABLE $TABLE_NAME
(" +
            "$COLUMN_ID INTEGER PRIMARY KEY
AUTOINCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$COLUMN_LAST_NAME TEXT, " +
            "$COLUMN_EMAIL TEXT, " +
            "$COLUMN_PASSWORD TEXT" +
            ")"



            db?.execSQL(createTable)
            }
```

```
        override fun onUpgrade(db: SQLiteDatabase?,
oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS
$TABLE_NAME")
        onCreate(db)
        }


        fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
        }


        @SuppressLint("Range")
        fun getUserByUsername(username: String): User? {
            val db = readableDatabase
            val cursor: Cursor = db.rawQuery("SELECT * FROM
$TABLE_NAME WHERE $COLUMN_FIRST_NAME = ?",
arrayOf(username))
            var user: User? = null
            if (cursor.moveToFirst()) {
            user = User(
            id =
cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NA
ME)),
            lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
            )
            }
            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_NA
ME)),
lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
)
}
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_NA
ME)),
            lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
            )
            users.add(user)
            } while (cursor.moveToNext())
            }
            cursor.close()
            db.close()
            return users
            }


            }
```