INDEX

1. INTRODUCTION

Sleep tracking is the process of monitoring and analyzing one's sleep patterns and behaviors to better understand the quality and quantity of sleep. This can be done through various methods, including wearable technology, mobile apps, and specialized devices. Sleep tracking can provide insights into factors that may be affecting sleep, such as stress, diet, and exercise habits. By tracking sleep over time, individuals can identify patterns and make adjustments to improve their sleep health. Sleep tracking can also be useful for individuals with sleep disorders, as it can provide information to healthcare professionals to aid in diagnosis and treatment. Overall, sleep tracking can be a valuable tool for promoting better sleep and overall health and well-being. one's sleep patterns and behaviors to better understand the quality and quantity of sleep. This can be done through various methods, including wearable technology, mobile apps, and specialized devices. Sleep tracking can provide insights into factors that may be affecting sleep, such as stress, diet, and exercise habits. By tracking sleep over time, individuals can identify patterns and make adjustments to improve their sleep health. Sleep tracking can also be useful for individuals with sleep disorders, as it can provide information to healthcare professionals to aid in diagnosis and treatment. Overall, sleep tracking can be a valuable tool for promoting better sleep and overall health and well-being. Sleep tracking app can provide valuable insights into your sleep patterns, such as how long you spend in different sleep stages, how many times you wake up during the night, and how long it takes you to fall asleep. Armed with this information, you can make adjustments to your lifestyle, such as improving your sleep hygiene or adjusting your bedtime routine to get better quality sleep. While sleep tracking can be a useful tool, it's important to remember that it's just one aspect of maintaining good sleep health. It's always best to consult with a healthcare professional if you're experiencing sleep-related issues or have concerns about your sleep patterns.

1.1. Overview

Sleep tracking app is the practice of monitoring and analyzing an individual's sleep patterns and quality. This is typically done using wearable devices such as smartwatches, fitness trackers, or specialized sleep tracking devices. Sleep tracking technology can provide insight into how long and how well an individual sleeps, the different stages of sleep, and the frequency and duration of interruptions during sleep. Sleep tracking technology uses a variety of sensors, including accelerometers, gyroscopes, and heart rate monitors, to collect data on an individual's sleep. This data is then analyzed by algorithms to provide insights into the individual's sleep patterns, including the duration of each stage of sleep, the number of times they wake up during the night, and the total amount of time spent in each stage of sleep. Sleep tracking app can provide a range of benefits, including helping individuals identify patterns or behaviors that may be negatively affecting their sleep, such as caffeine or alcohol consumption, stress, or an uncomfortable sleep environment. Sleep tracking can also help individuals develop better sleep habits by providing personalized recommendations for improving sleep quality, such as adjusting bedtime routines or sleep environment factors. However, it's important to note that sleep tracking apps are not medical devices and should not be used to diagnose or treat sleep disorders. If an individual is experiencing ongoing sleep difficulties, they should consult with a healthcare professional for a proper diagnosis and treatment plan

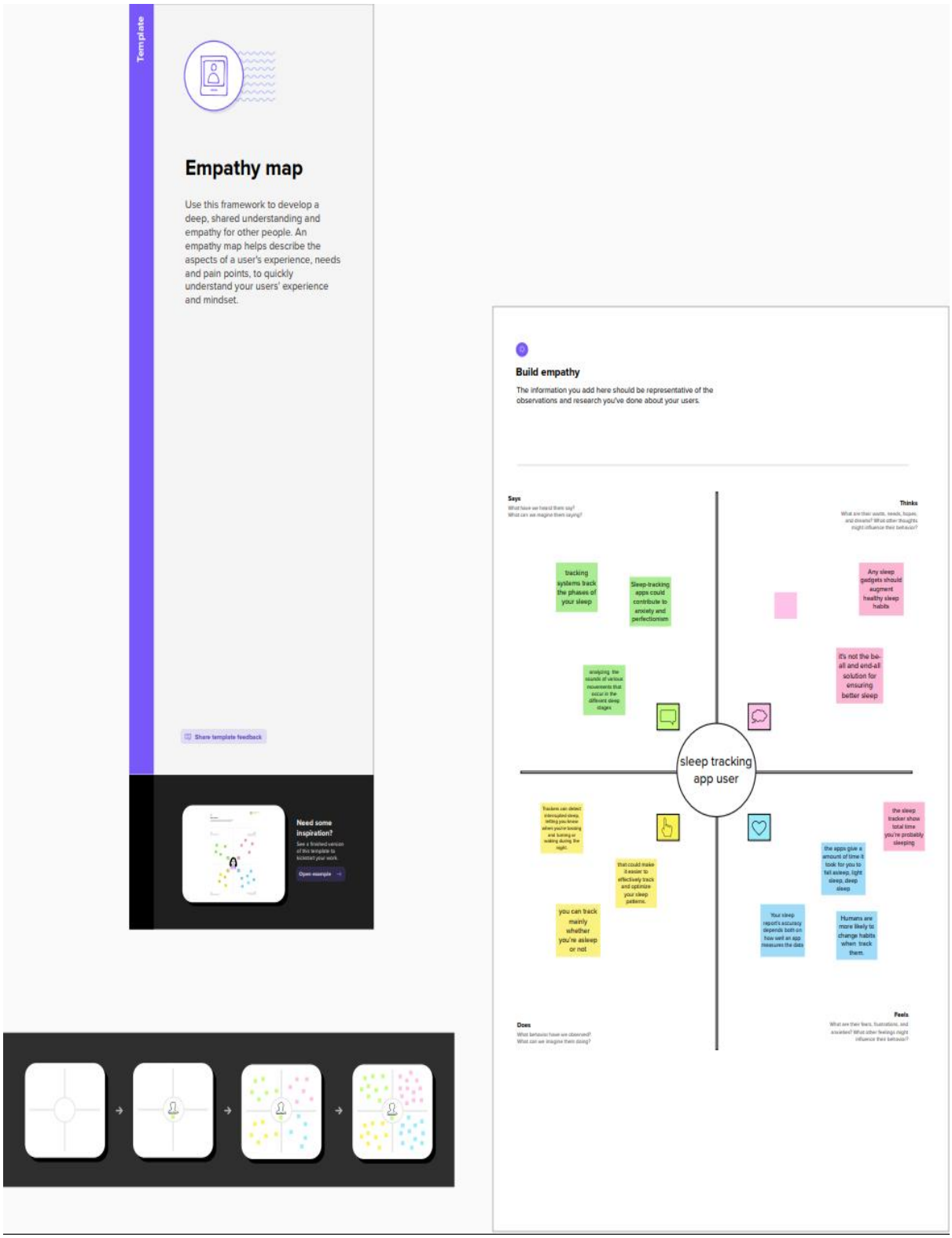
1.2. Purpose

Understanding sleep patterns: Sleep tracking app can help individuals understand their sleep patterns, including the amount of time spent in each stage of sleep, the number of times they wake up during the night, and the overall quality of their sleep. This information can help individuals identify potential sleep disorders or other issues that may be impacting their sleep. Improving sleep habits: By tracking their sleep, individuals can identify behaviors or habits that may be negatively impacting their sleep and make changes to improve their sleep quality. For example, they may find that they sleep better when they avoid caffeine or electronic devices before bed. Monitoring sleep for medical reasons: Sleep tracking app can be used to monitor sleep patterns in individuals with sleep disorders, such as sleep apnea or insomnia, to help diagnose and manage these conditions.

Enhancing athletic performance: Athletes may use sleep tracking to optimize their sleep for improved physical performance and recovery. Overall, sleep tracking app can provide valuable insights into an individual's sleep habits and patterns, which can help improve overall health and well-being. Sleep tracking app is the process of monitoring and measuring the quality and quantity of your sleep. The main purpose of sleep tracking is to help you understand and improve your sleep habits and overall well-being. By tracking your sleep, you can gain insights into your sleep patterns and identify any issues that may be affecting the quality of your sleep, such as snoring, sleep apnea, or insomnia. Sleep tracking can also help you identify lifestyle factors that may be affecting your sleep, such as caffeine intake, alcohol consumption, or exercise habits. With this information, you can make changes to your lifestyle to improve your sleep and overall health. Additionally, some people use sleep tracking app to monitor the effectiveness of sleep aids or treatments for sleep disorders.

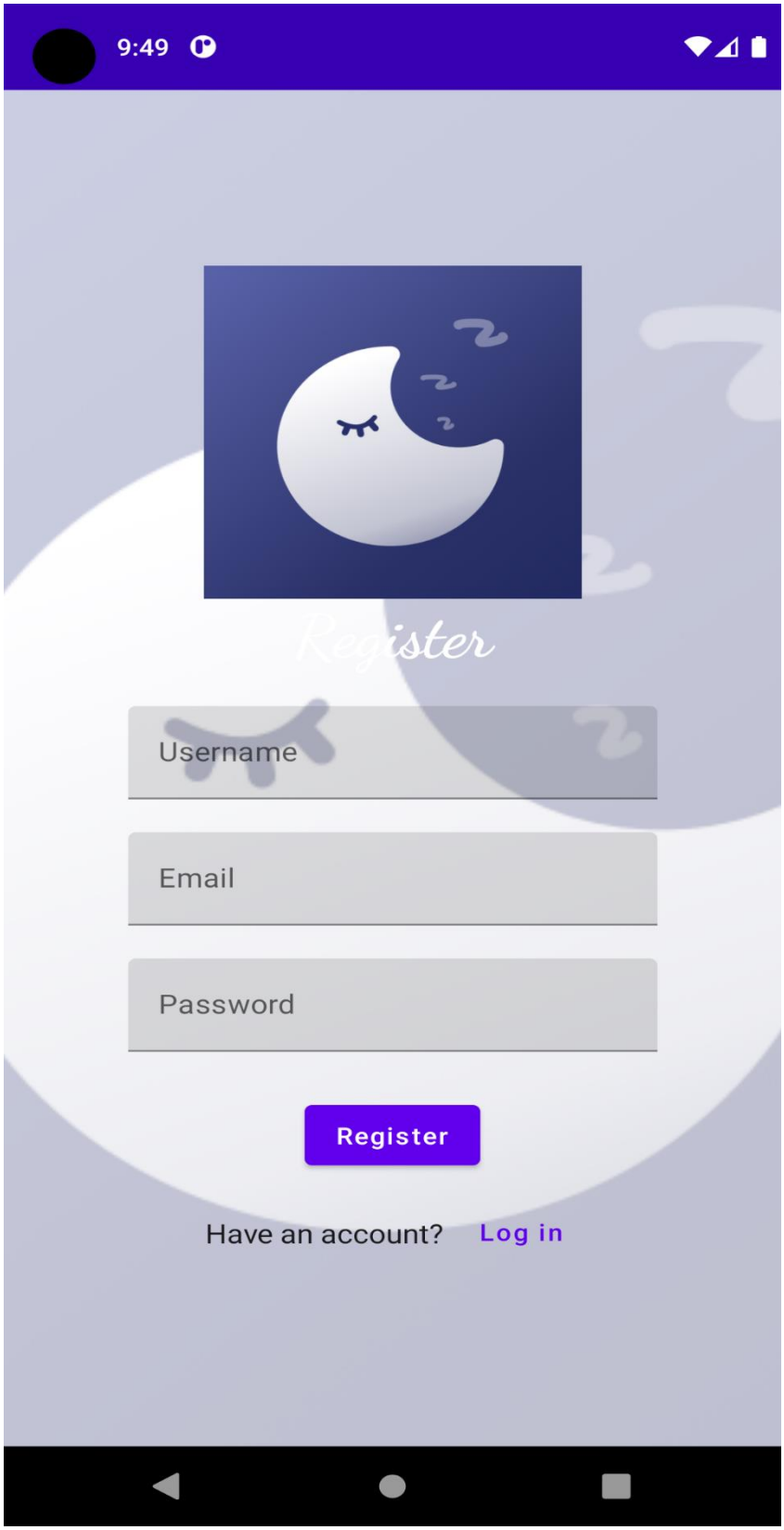
2. PROJECT DEFINITION AND DESIGN THINKING

2.1. Empathy Map

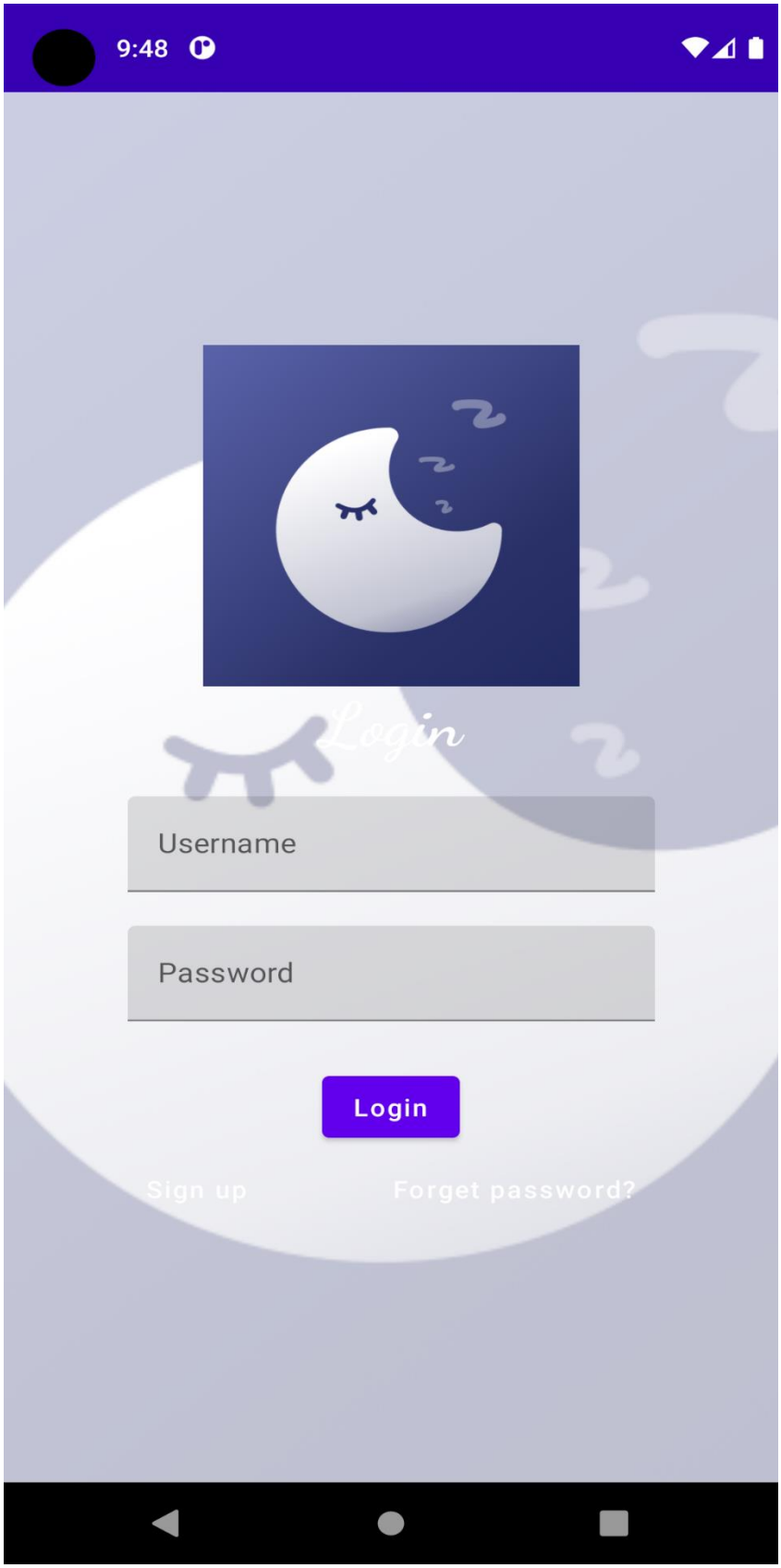


2.2. Ideation & Brainstorming Map

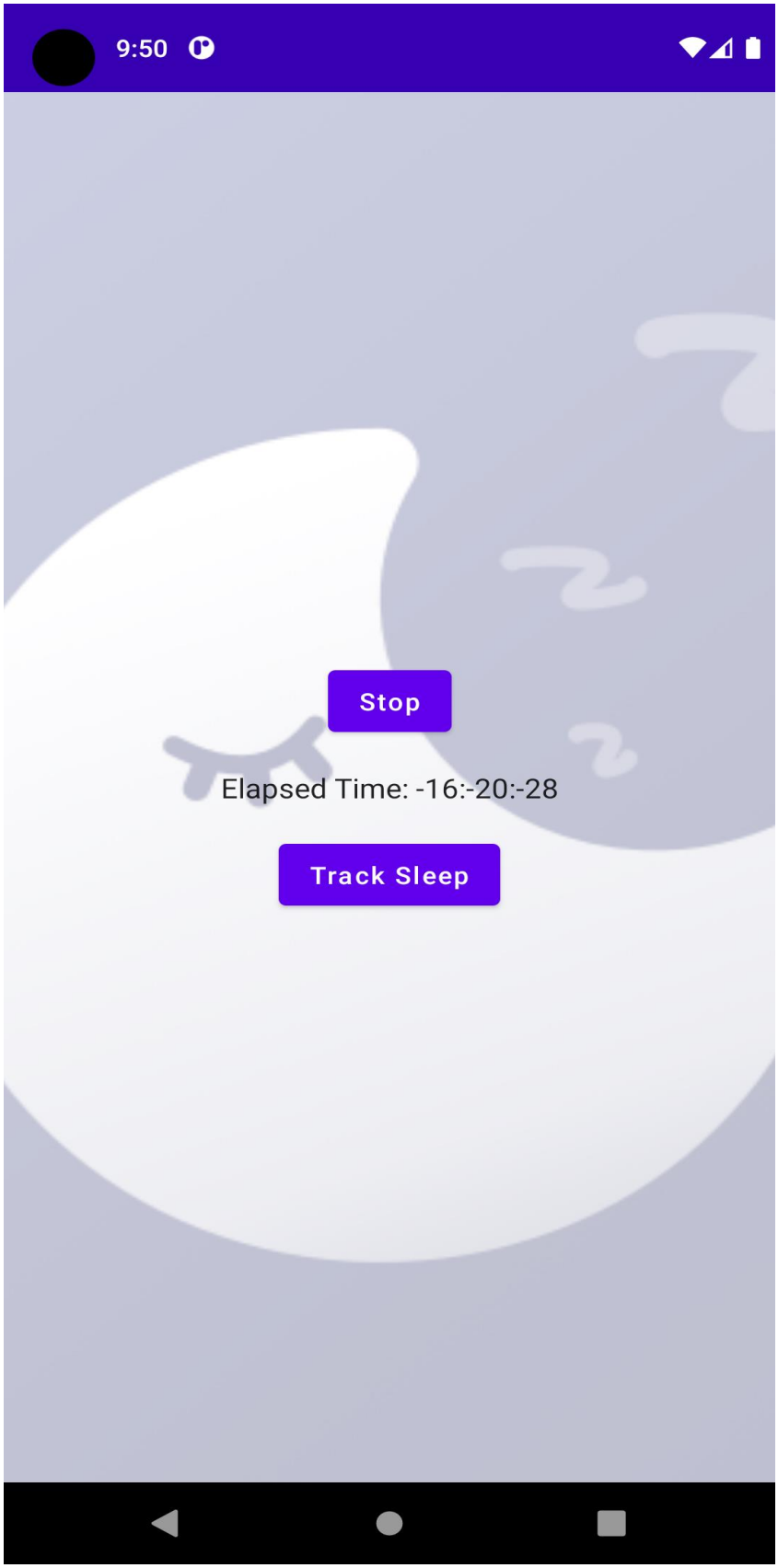
3. RESULT



3.1. Registration Activity



3.2. Login Activity



3.3. MainActivity

3.4. TrackActivity

4. ADVANTAGES & DISADVANTAGES

ADVANTAGES

Awareness of sleep patterns: Sleep tracking app can help you become more aware of your sleep patterns, including the duration and quality of your sleep. This can help you identify any issues that may be affecting your sleep and take steps to improve your sleep hygiene.

Improved sleep quality: Sleep tracking app can help you identify factors that may be affecting your sleep quality, such as caffeine consumption or exercise habits. By making adjustments to these factors, you can improve the quality of your sleep and wake up feeling more refreshed.

Better overall health: Consistent, quality sleep is important for overall health and well-being. Sleep tracking can help you monitor your sleep patterns and make adjustments to ensure you are getting the recommended amount of sleep for your age and lifestyle.

Enhanced performance: Good sleep is essential for cognitive and physical performance. By tracking your sleep, you can identify factors that may be affecting your performance and take steps to improve your sleep, leading to enhanced performance in your daily life.

Personalized insights: Sleep tracking app can provide personalized insights based on your unique sleep patterns and habits, allowing you to make adjustments specific to your needs for better sleep and overall health.

DISADVANTAGES

Inaccuracy: Sleep tracking technology can sometimes provide inaccurate readings, leading to false conclusions about sleep quality or duration. For example, movement during the night can be misinterpreted as wakefulness, or the device may not be able to distinguish between light and deep sleep stages. Obsession: Constantly tracking sleep can lead to an unhealthy obsession with sleep metrics, causing unnecessary stress and anxiety. Privacy concerns: Sleep tracking app often requires personal data to be stored in the cloud or on the device, raising privacy concerns for some users. False sense of security: Relying too heavily on sleep tracking app can create a false sense of security, leading users to overlook other factors that can affect sleep quality, such as diet, exercise, and stress levels.

5. APPLICATIONS

Sleep tracking apps are used to monitor and analyze a person's sleep patterns and behaviors during the night. These apps are usually installed on smartphones, fitness trackers, or smartwatches and use sensors like accelerometers or heart rate monitors to track various aspects of sleep, such as sleep duration, quality, and interruptions.

Some common uses of sleep tracking apps include:

Monitoring sleep patterns: Sleep tracking apps can help people understand their sleep patterns and identify any irregularities, such as sleep disruptions or sleep disorders.

Improving sleep quality: By analyzing sleep data, these apps can provide personalized recommendations to improve sleep quality, such as adjusting sleep schedules or sleep environments.

Identifying health issues: Sleep tracking apps can help identify health issues related to sleep, such as sleep apnea, restless leg syndrome, or insomnia.

Tracking sleep-related activities: Some apps allow users to track activities related to sleep, such as caffeine intake or exercise, to understand how they affect sleep.

Overall, sleep tracking apps can be useful for people who want to improve their sleep habits, identify potential health issues related to sleep, or simply monitor their sleep patterns for personal reasons.

6. CONCLUSION

Sleep tracking app can be a valuable tool for individuals who are looking to improve the quality and quantity of their sleep. By monitoring sleep patterns and identifying potential issues, such as sleep apnea or insomnia, people can take steps to address these issues and improve their overall health and well-being. While sleep tracking app can provide valuable insights into your sleep habits, it's important to keep in mind that no single metric can tell the whole story. Factors such as stress, diet, and exercise can also impact your sleep, and it's important to take a holistic approach to improving your sleep hygiene. Overall, sleep tracking can be a helpful tool for anyone looking to optimize their sleep habits and improve their overall health and well-being. While sleep tracking app can be beneficial, it is important to remember that it is not a substitute for professional medical advice or treatment. If you have concerns about your sleep or suspect you may have a sleep disorder, it is important to consult with a healthcare provider. Overall, incorporating sleep tracking app into a comprehensive approach to sleep health can be a helpful tool for promoting better sleep and overall well-being.

7. FUTURE SCOPE

Sleep tracking app is already advancing at a rapid pace, and it is likely to continue evolving in the future. Here are some potential future advancements and applications of sleep tracking technology:

Personalized sleep recommendations: As sleep tracking app becomes more advanced, it will be able to provide personalized recommendations to users based on their sleep patterns, genetics, and lifestyle factors. This could include suggestions for optimal sleep duration, ideal sleep environment, and even personalized sleep aids.

Real-time tracking: Future sleep tracking technology may be able to track sleep in real-time, providing users with instant feedback on the quality of their sleep. This could allow for immediate adjustments to improve sleep quality.

Integration with wearable app: As wearable technology becomes more advanced, sleep tracking could be seamlessly integrated into devices like smartwatches and fitness trackers. This could provide users with even more detailed information about their sleep patterns.

Overall, the future of sleep tracking app looks promising, with numerous potential applications and advancements on the horizon.

8. APPENDIX

A. Source Code

MainActivity.kt

```
package com.example.sleeptracking
import android.content.Context
import android.content.Intent
import android.icu.text.SimpleDateFormat
import android.os.Build
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.Button
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import com.example.sleeptracking.ui.theme.SleepTrackingTheme
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 {
            SleepTrackingTheme {
                // 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)
}
@RequiresApi(Build.VERSION_CODES.N)
fun getCurrentDateTime(): String {
    val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())
    val currentTime = System.currentTimeMillis()
    return dateFormat.format(Date(currentTime))
}
fun formatTime(timeInMillis: Long): String {
    val hours = (timeInMillis / (1000 * 60 * 60)) % 24
    val minutes = (timeInMillis / (1000 * 60)) % 60
    val seconds = (timeInMillis / 1000) % 60
    return String.format("%02d:%02d:%02d", hours, minutes, seconds)
}
```

AppDatabase.kt

```
package com.example.sleeptracking
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.kt

```
package com.example.sleeptracking
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_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.sleeptracking

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.sleeptracking

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

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

}
```

User.kt

```
package com.example.sleeptracking

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.sleeptracking
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.sleeptracking

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.sleeptracking

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 androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase


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
    }

}
```

RegistrationActivity.kt

```
package com.example.sleeptracking

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.sleeptracking.ui.theme.SleepTrackingTheme

class RegistrationActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackingTheme {
                // 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.sleeptracking),
        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 = {
                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)
}
```

LoginActivity.kt

```
package com.example.sleeptracking



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 androidx.core.content.ContextCompat.startActivity
import com.example.sleeptracking.ui.theme.SleepTrackingTheme

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

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackingTheme {
                // 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.sleeptracking),
        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,
            RegistrationActivity::class.java
        )
    )}
    )
    { Text(color = Color.White,text = "Sign up") }
    TextButton(onClick = {
//          startActivity(
//          Intent(
//              applicationContext,
//              MainActivity2::class.java
//          )
//      )
//          context.startActivity(
//              Intent(
```

```
//                    context,
//                    RegistrationActivity::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)
}
private fun startRegisterPage(context: Context) {
  val intent = Intent(context, RegistrationActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

TrackActivity.kt

```
package com.example.sleeptracking



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 androidx.core.content.ContextCompat.startActivity
import com.example.sleeptracking.ui.theme.SleepTrackingTheme

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

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            SleepTrackingTheme {
                // 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.sleeptracking),
        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,
            RegistrationActivity::class.java
        )
    )}
    )
    { Text(color = Color.White,text = "Sign up") }
    TextButton(onClick = {
//          startActivity(
//          Intent(
//              applicationContext,
//              MainActivity2::class.java
//          )
//      )
//          context.startActivity(
//              Intent(
```

```
//                    context,
//                    RegistrationActivity::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)
}
private fun startRegisterPage(context: Context) {
  val intent = Intent(context, RegistrationActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```