A Sleep Tracking App for a Better Night’s Rest

V.Deepika

B.Vijay santhi

M.Anitha

K.Fathimakani

A.Hasmath Nasiha banu

1.Intoduction

1.1 overview

Sleep tracking can be helpful tool for improving the quality of your sleep .By monitoring your sleep patterns and habits, you can give insight into what factors may be contributing to poor sleep and make Changes to improve your overall sleep hygiene.

Project workflow

1.start by clearly defining the problem that the sleep tracking projects aims to slove.

2.Define the project objectives.

3.Choose a sleep tracking tool that suits Your needs.

4. Once you have selected a sleep Tracking tool, set it up according to the instructions Provided.

5.Start tracking your sleep data using the Sleep tracking tool.

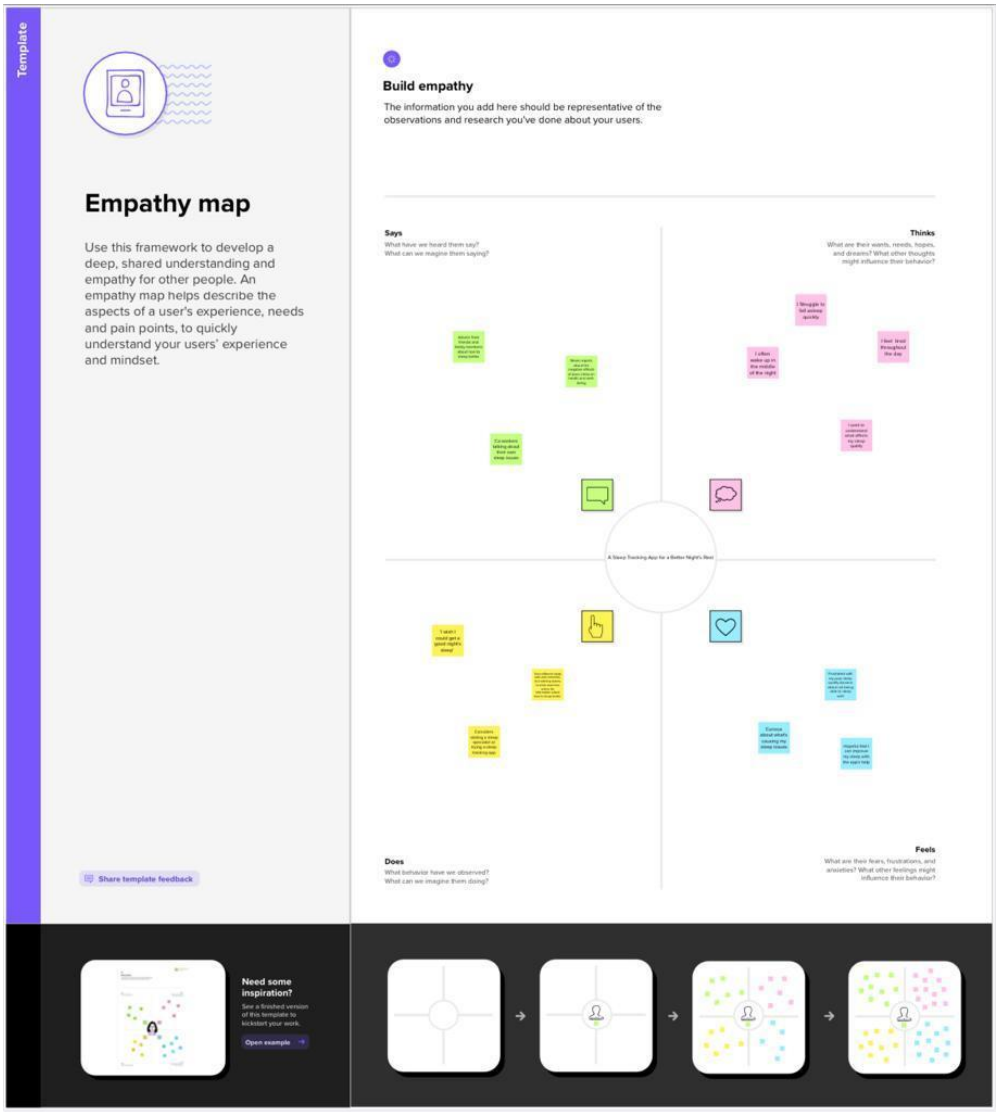
6.Analyze the data collected By the sleep Tracking tool.

1.2. Purpose

- User register, user login into the application.
- After registration,user login into the application.
- User enters into the main page.
- The app allows the user to choose,play and and pause the sleep tracking tool.

2.problem definition and design thinking

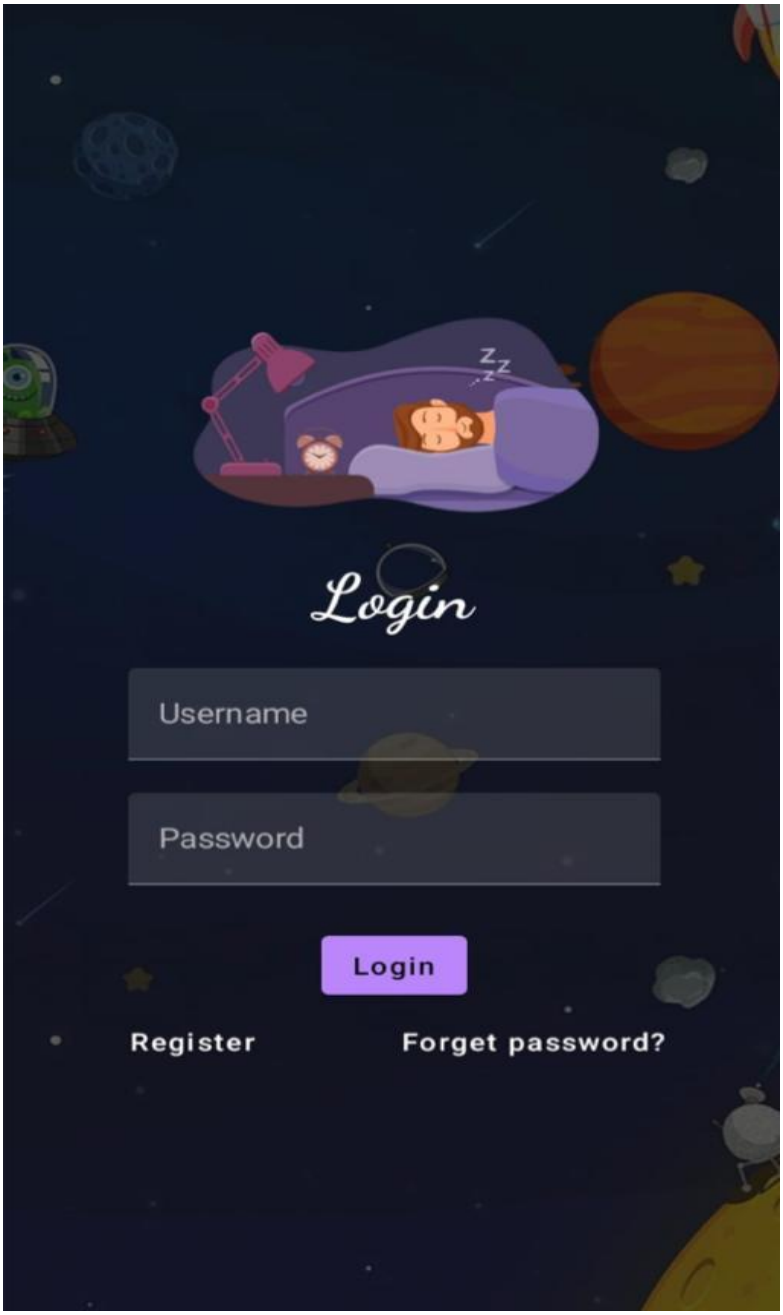
2.1Empathy map



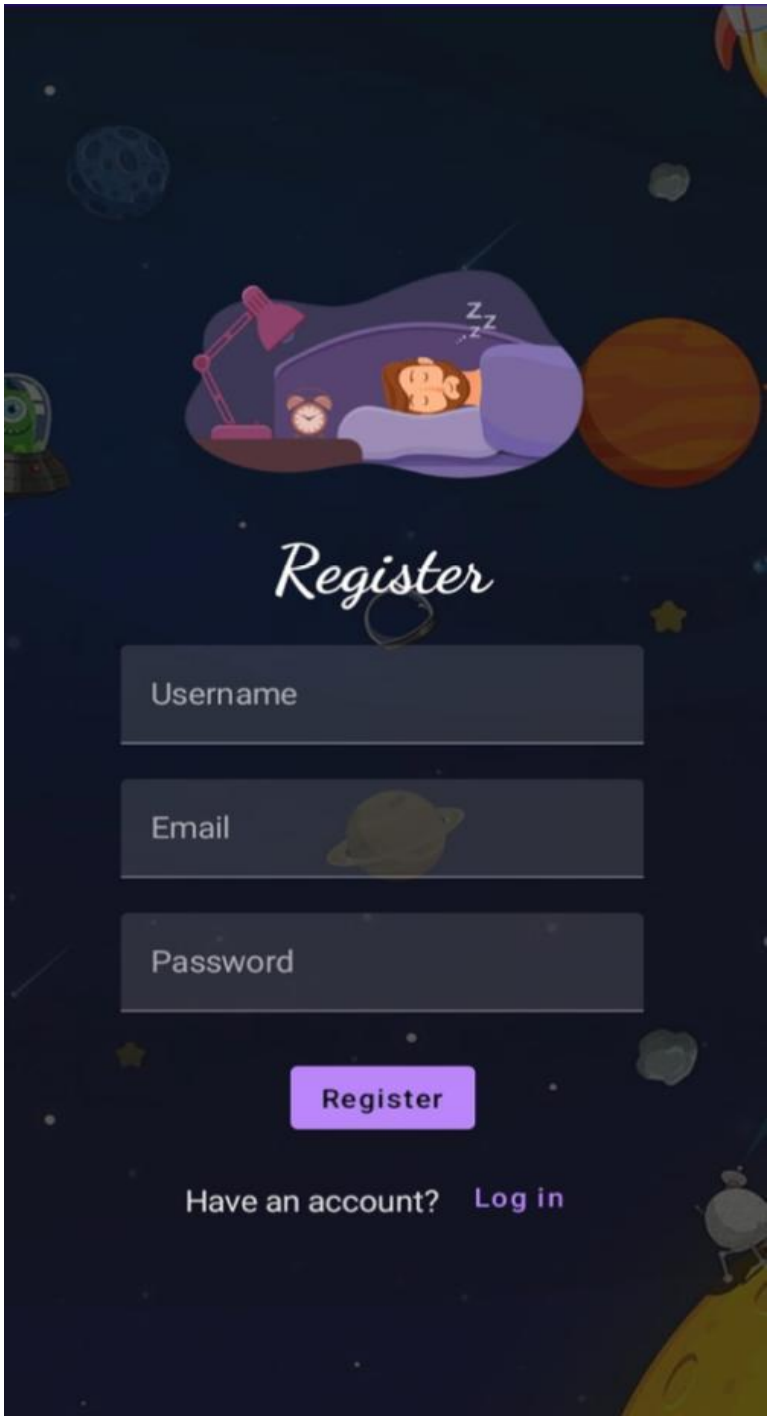
2.2 ideation & Brainstorming

3.Result

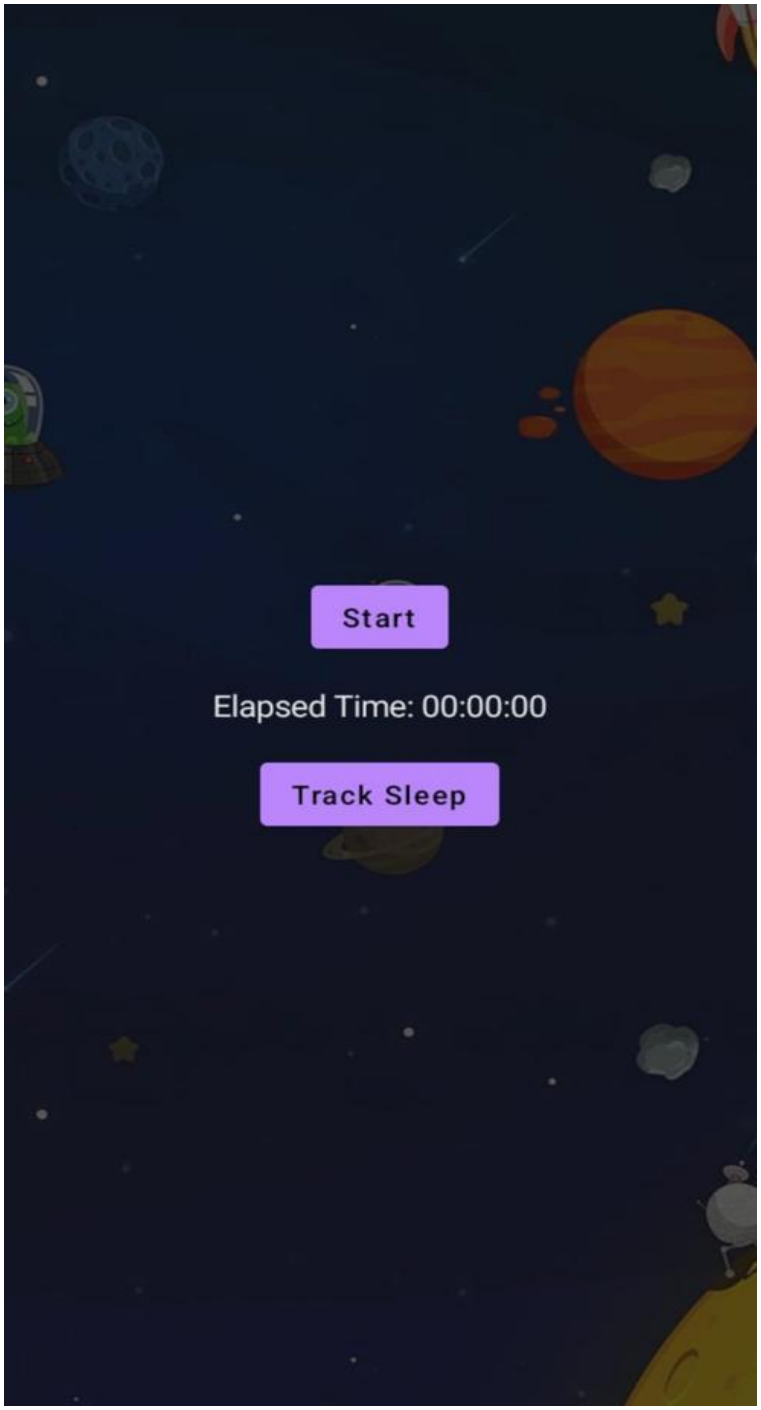
Login Page



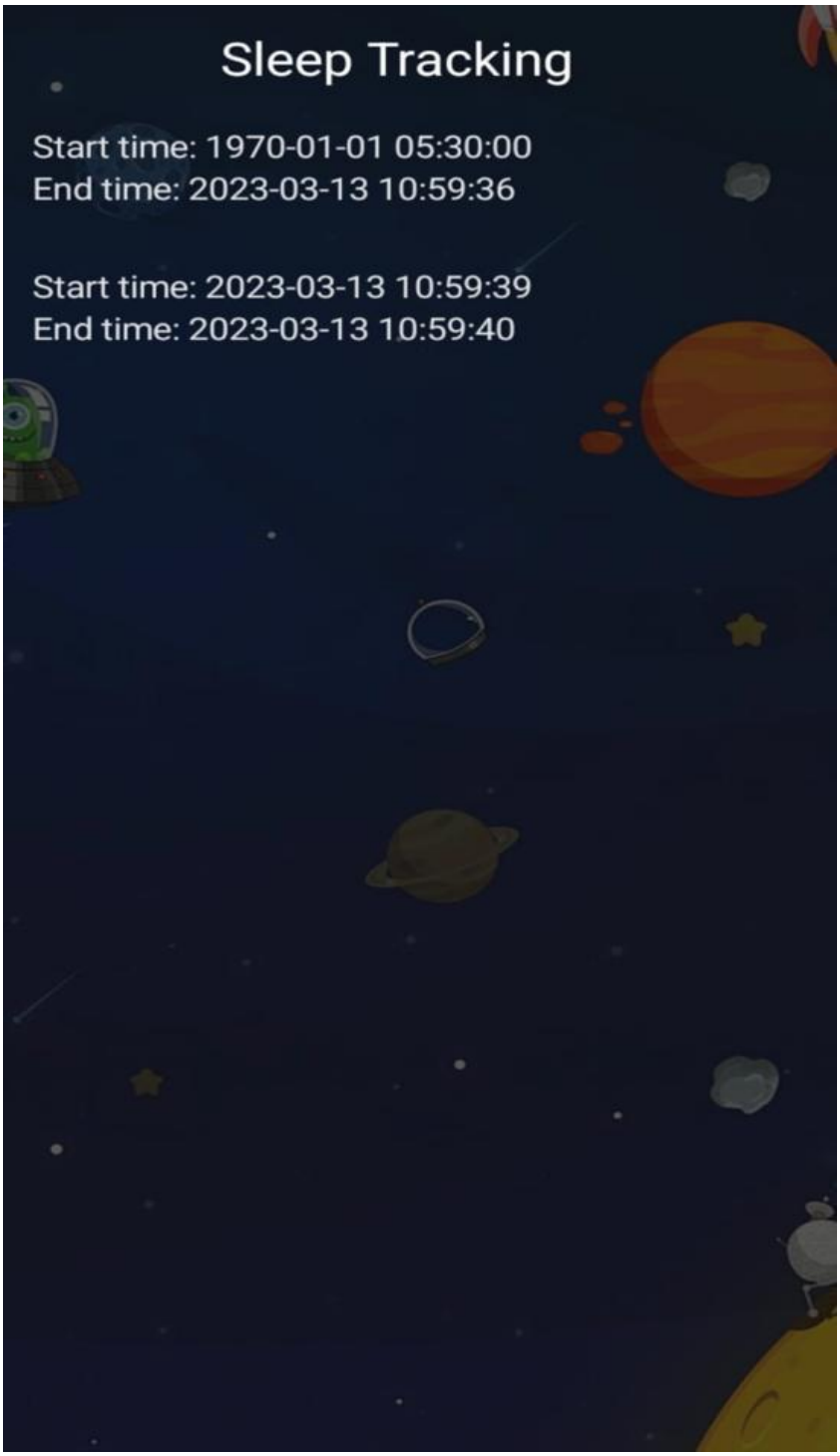
Registration page



Main page



Track sleep page



4.Advantage and Disadvantages

Advantages:

- Improved awareness: sleep tracking can

Provided individual with insights into their sleep

Patterns, helping them better understand their sleep

Needs and preferences.

- Identification of sleep issues: sleep tracking

Can help individuals identify factors that may

apnea or restless leg syndrome.

- Set up the tool: once you have selected a

Sleep tracking tool ,set it up according to the

instructions provided.

- Track sleep data: start tracking your sleep

Data using the sleep tracking tool.

Disadvantages:

- Accuracy: Sleep tracking tecnology can

Sometimes be inaccurate, leading to incorrect or

Unreliable data.

- Discomfort: some sleep tracking device

Can be uncomfortable to weer , which can interfere

With sleep and lead to inaccurate data.

- Dependency: Individuals may become overly

Reliant on sleep tracking data, leading to increased

Anxiety or obsession with sleep quality.

- Cost: High quality sleep tracking device and

App can be expensive and making them.

5.Application

- Fitbit: This wearable devices tracks sleep

duration and quality, as well as provide personalized

sleep insights and guidance.

- Sleep Cycle: This smartphone app track

Sleep quality and provide detailed analysis of sleep

Pattern, as well as personalized sleep recommend.

- Our Ring: This wearable devices tracks sleep

Duration and quality, as well as provide personalized

Sleep insights and guidance, and also monitor activities

and other health metrics.

- Apple watch: This wearable devices tracks

Sleep duration and quality, and provide personalized

Sleep insights and guidance as well as tracking others

Health metrics.

- Philips smart sleep: This wearable headband

Track brain waves and provide personalized feedback

To help individuals optimize their sleep health.

- Sleep score Max: This besides device tracks

Sleep quality and provide personalized sleep as

recommendations based on analysis of sleep data.

6.Conclusion

1. Sleep tracking can be valuable tool for improving

Sleep quality and optimizing overall health and well

Being. By tracking sleep duration, quality,and other

Metrics, individual can gain insight into their sleep

Habit and making adjustments to optimize their

Sleep health. Sleep tracking can also help identify

Potential sleep disorders and aid in their diagnosis

And treatment.

2. However, it is important to consider the potential Disadvantages of sleep tracking, such as accuracy, Discomfort, dependency, cost, privacy concern, Distraction ,and misinterpreted of data. Individuals Should make informed decisions about using sleep Tracking technology and consult with a healthcare Professional if they have concern about their sleep Health.

7.Future Scopes

1. Sleep duration tracking: This

Features tracks how long an individual sleep each

night.

2. Sleep quality tracking: This

Features measure the quality of sleep by tracking

Metrics such as cycle, heart rate, movement.

3. stage tracking: This features

Track the different stages of sleep, such as REM

(Rapid eye movement) and deep sleep.

4. Sleep environment tracking: This

Features measure factors such as temperature,

Noise,and light levels in the sleep environment.

5. Personalized recommendations:

Based on sleep tracking data, the system may be

Provide personalized recommendations to improve

Sleep quality, such as adjusting bedtime routine or

Sleep environment.

6. Smart alarm: This features wakes

Individuals up to during a lighter stage of sleep, making

It easier to wake up feeling refreshed.

7. Sleep coaching: Some system May

Offer coaching or educational resources to help

Individuals improve their sleep habits.

8. Integration with other devices:

Some sleep tracking system may integrate with other

Devices, such as smart home system or wearable

Fitness tracker.

9. Data visualization: The system may

Provide visualization or chart to help individuals easily

Understand and analysis their sleep data.

8.Appendix

A. Source code

Creating the database class

Step 1: create user data class

```
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?,

   )
```

Step 2: Create an Userdao interface

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)
}
```

Step 3: create an Userdatabase class

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
    }
  }
 }
}
```

Step 4: create an Userdatabasehelper class

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(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
    }

}
```

Database 2

Step 1: create timelog data class

```
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
)
```

Step 2: create an timelogdao interface

Timelogdao interface code

```
Package com.example.projectone
```

```
Import androidx.room.Dao
Import androidx.room.Insert

@Dao
Interface TimeLogDao {
   @Insert
   Suspend fun insert(timeLog: TimeLog)

}
```

Step 3: create an appdatabase class

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
            }
        }
    }
}
```

Step 4: create an timedatabase helper

Timedatabase helper 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(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"
```

```
    }
  }
```

```
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)
}
```

```
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)
}
```

```
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)
}

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))
}
```

```
<?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>
```