

# COMPOSE INPUT: DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE

PROJECT PRESENTATION BY  
TEAM ID: NM2023TMID14977  
TEAM LEADER: SUVEDHA P

TEAM MEMBERS:

SUMITHRA D

SWETHA E

MONISHA T

## 1.INTRODUCTION

### 1.1 Overview:

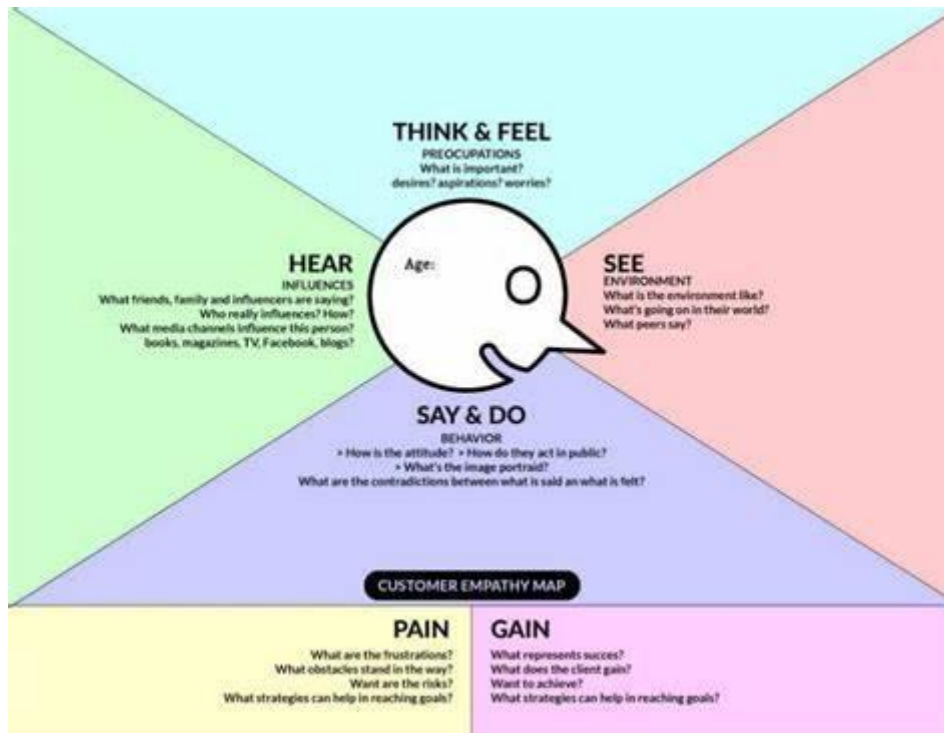
*Input compose in text is a central piece of any UI, and Jetpack Compose makes it easier to display or write text. Compose leverages composition of its building blocks, meaning you don't need to overwrite properties and methods or extend big classes to have a specific composable design and logic working the way you want.*

### 1.2. Purpose:

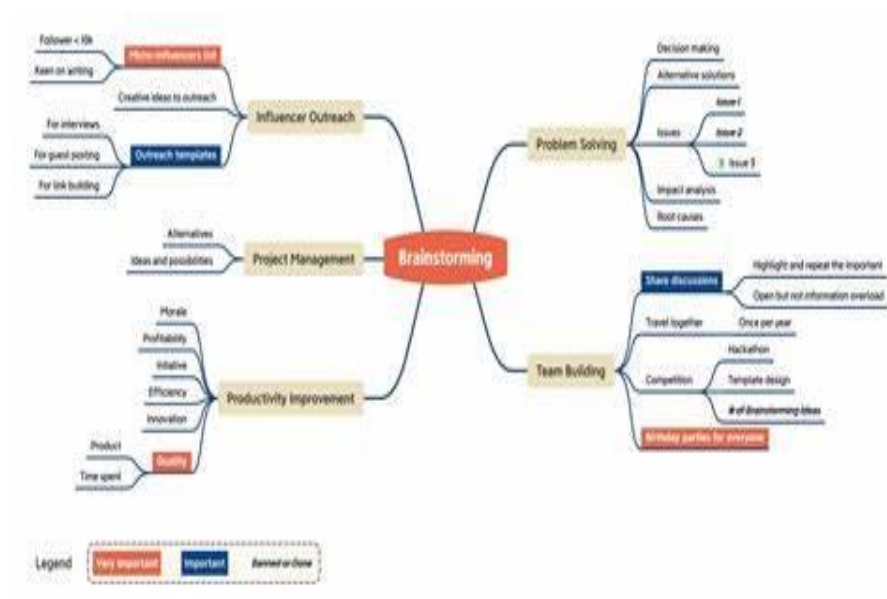
*In Android Jetpack Compose, a Text Field is used to take input from the user in the form of text. TextField when in focus or when clicked invokes a keypad which is technically termed a Keyboard or a Soft-Keyboard.*

## 2. PROBLEM DEFINITION AND DESIGN THINKING

### 2.1. Empathy map:



### 2.2. Ideation and Brainstorming map:



## Survey Details

Name: Raja

Age: 34

Mobile\_number: 9486096902

Gender: Male

Diabetic: Not Diabetic

Name: Priya

Age: 45

Mobile\_number: 9685268249

Gender:Female

Diabetic:Diabetic

## 4. ADVANTAGES AND DISADVANTAGES:

### Advantages:

*It simplifies and accelerates UI development on Android bringing your apps to life with less code, powerful tools, and intuitive Kotlin APIs. It makes building Android UI faster and easier. While creating Compose we worked with different partners who experienced all of these benefits first hand and shared some of their takeaways with us.*

### Disadvantages:

*Input Devices provide input signals to Computing devices. There can be disadvantages for a particular type of Input Device such as Mouse, Keyboard, Pointers etc. Nonetheless without Input devices Computer might not be able to receive information & remain useless. The only disadvantage of input devices is that they can't be used as output device.*

## 5. APPLICATION:

*Tap on a button with their finger. Navigate through a screen using their physical keyboard. Enter their email address using the on-screen keyboard. Compose has a lot of built-in support for these use cases, but in some scenarios you need to customize or extend the default behaviours.*

## 6. CONCLUTION:

*An effective conclusion is created by following these steps: Restate the thesis: An effective conclusion brings the reader back to the main point, reminding the reader of the purpose of the essay. However, avoid repeating the thesis verbatim. Paraphrase your argument slightly while still preserving the primary point.*

## 7. FUTURE SCOPE:

*Recompose scopes are an important piece of the Compose puzzle. They do some bookkeeping and help reduce the amount of work Compose has to do to prepare a frame. They are the smallest unit of a composition that can be re-executed (recomposed) to update the underlying tree.*

# ComposeInput:ADemonstrationofTextInputandValidation withAndroidCompose

## AndroidManifest.xml

```
<?xmlversion='1.0'encoding='utf-8' ?>
<manifestxmlns: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:supportRtl='true'
        android:theme='@style/Theme.SurveyApplication'
        tools:targetApi='31'>
        <activity
            android:name='.RegisterActivity'
            android:exported='false'
            android:label='@string/title_activity_register'
            android:theme='@style/Theme.SurveyApplication'/>
        <activity
            android:name='.MainActivity'
            android:exported='false'
            android:label='MainActivity'
            android:theme='@style/Theme.SurveyApplication'/>
        <activity
            android:name='.AdminActivity'
            android:exported='false'
            android:label='@string/title_activity_admin'
            android:theme='@style/Theme.SurveyApplication'/>
        <activity
            android:name='.LoginActivity'
            android:exported='true'
            android:label='@string/app_name'
            android:theme='@style/Theme.SurveyApplication'>
            <intent-filter>
                <actionandroid:name='android.intent.action.MAIN'/>

                <categoryandroid:name='android.intent.category.LAUNCHER'
            />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

## Color.kt

```
packagecom.example.surveyapplication.ui.theme
```

```
import androidx.compose.ui.graphics.Color

val colorPrimary = Color(0xFFBB86FC)
val colorPrimaryVariant = Color(0xFF6200EE)
val colorSecondary = Color(0xFF3700B3)
val colorTertiary = Color(0xFF03DAC5)
```

## Shape.kt

```
package com.example.surveyapplication.ui.theme
```

```
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp
```

```
val Shapes = Shapes(
    all = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(0.dp)
)
```

## Theme.kt

```
package com.example.surveyapplication.ui.theme
```

```
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable
```

```
private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)
```

```
private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200
)
```

```
/* Other default colors to override
background = Color.White,
surface = Color.White,
onPrimary = Color.White,
onSecondary = Color.Black,
onBackground = Color.Black,
onSurface = Color.Black,
*/
```

```
@Composable
fun SurveyApplicationTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable ()->Unit
){
```



```

        val colors = if (darkTheme) {
            DarkColorPalette
        } else {
            LightColorPalette
        }

        MaterialTheme(
            colors = colors,
            typography = Typography,
            shapes = Shapes
        )
    }
}

```

## Type.kt

```
package com.example.surveyapplication.ui.theme
```

```

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

```

```

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)

```

## AdminActivity.kt

```
package com.example.surveyapplication
```

```

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.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

class AdminActivity : ComponentActivity() {
    private lateinit var databaseHelper : SurveyDatabaseHelper
    override fun onCreate(savedInstanceState : Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = SurveyDatabaseHelper(this)
        setContent {
            val data = databaseHelper.getAllSurveys()
            Log.d("swathi", data.toString())
            val survey = databaseHelper.getAllSurveys()
            ListListScopeSample(survey)
        }
    }
}

@Composable
fun ListListScopeSample(survey : List<Survey>){

    Image(
        painterResource(id = R.drawable.background), contentDescription =
        "",
        alpha = 0.1f,
        contentScale = ContentScale.FillHeight,
        modifier = Modifier.padding(top = 40.dp)
    )

    Text(
        text = "SurveyDetails",
        modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom =
        24.dp),
        fontSize = 30.sp,
        color = Color(0xFF25b897)
    )
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .padding(top = 80.dp),
        horizontalArrangement = Arrangement.SpaceBetween
    ){
        item{

            LazyColumn{
                items(survey){ survey->
                    Column(
                        modifier = Modifier.padding(
                            top = 16.dp,
                            start = 48.dp,

```



```
        bottom=20.dp  
    )  
){  
    Text("Name:${survey.name}")  
    Text("Age:${survey.age}")  
    Text("Mobile_Number:${survey.mobileNumber}")  
    Text("Gender:${survey.gender}")  
    Text("Diabetics:${survey.diabetics}")  
}  
  
}  
  
}  
  
}
```

## LoginActivity.kt

**packagecom.example.surveyapplication**

```
import android.content.Context  
import android.content.Intent  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.compose.setContent  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.background  
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.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.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import androidx.compose.ui.unit.sp  
import androidx.core.content.ContextCompat  
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme
```

```
class LoginActivity: AppCompatActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContentView(
            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("") }

Column(
    modifier = Modifier.fillMaxSize().background(Color.White),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
){

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

    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,
        color = Color(0xFF25b897),
        text = "Login"
    )

    Spacer(modifier = Modifier.height(10.dp))

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

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

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

Button(
    onClick = {
        if (username.isNotEmpty() && password.isNotEmpty()) {
            val user = databaseHelper.getUserByUsername(username)
            if (user != null && user.password == password) {
                error = "Successfully login"
                context.startActivity(
                    Intent(
                        context, MainActivity::class.java
                    )
                )
            }
        }
    }
)

//onLoginSuccess()

```

```

    }
    if(user!=null&&user.password=="admin"){
        error="Successfullylogin" context.startActivity(
            Intent(
                context,AdminActivity::class.java
            )
        )
    }
    else{
        error=  "Invalidusernameorpassword"
    }

    }else{
        error="Pleasefillallfields"
    }
},
colors=ButtonDefaults.buttonColors(backgroundColor=
Color(0xFF84adb8)),
modifier=Modifier.padding(top=16.dp)
){
    Text(text="Login")
}
Row{
    TextButton(onClick={context.startActivity( Intent(
        context,RegisterActivity::class
        .java
    )
    )})
    {Text(color=Color(0xFF25b897),text="Register")}
    TextButton(onClick={
    })

    {
        Spacer(modifier=Modifier.width(60.dp))
        Text(color=Color(0xFF25b897),text="Forgetpassword?")
    }
}
}
}
privatefunstartMainPage(context:Context){
    valintent=Intent(context,MainActivity::class.java)
    ContextCompat.startActivity(context,intent,null)
}

```

## MainActivity.kt

```
packagecom.example.surveyapplication
```

```

importandroid.content.Context
importandroid.content.Intent
importandroid.os.Bundle
importandroidx.activity.ComponentActivity
importandroidx.activity.compose.setContent
importandroidx.compose.foundation.Image
importandroidx.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.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

```

```

class MainActivity : ComponentActivity() {
    private lateinit var dbHelper: SurveyDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        dbHelper = SurveyDatabaseHelper(this)

        setContent {
            FormScreen(this, dbHelper)
        }
    }
}

```

@Composable

```

fun FormScreen(context: Context, dbHelper: SurveyDatabaseHelper) {

```

```

    Image(
        painterResource(id = R.drawable.background), contentDescription =
        ...,
        alpha = 0.1f,
        contentScale = ContentScale.FillHeight,
        modifier = Modifier.padding(top = 40.dp)
    )

```

```

// Define state for form fields
var name by remember { mutableStateOf("") }
var age by remember { mutableStateOf(0) }
var mobileNumber by remember { mutableStateOf("") }
var genderOptions = listOf("Male", "Female", "Other")
var selectedGender by remember { mutableStateOf("") }
var error by remember { mutableStateOf("") }
var diabeticsOptions = listOf("Diabetic", "Not Diabetic")
var selectedDiabetics by remember { mutableStateOf("") }

```

```

Column(
    modifier = Modifier.padding(24.dp),
    horizontalAlignment = Alignment.Start,
    verticalArrangement = Arrangement.SpaceEvenly
){

```

```

    Text(
        fontSize = 36.sp,
        textAlign = TextAlign.Center,
        text = "Survey on Diabetics",
        color = Color(0xFF25b897)
    )

```



```

Spacer(modifier=Modifier. height(24. dp))

Text(text='Name:',fontSize=20. sp)

TextField(
    value=name,
    onChange={name=it},
)

Spacer(modifier=Modifier. height(14. dp))

Text(text='Age:',fontSize=20. sp)

TextField(
    value=age,
    onChange={age=it},
)

Spacer(modifier=Modifier. height(14. dp))

Text(
    (text='MobileNumber:',fontSize=20. sp)

TextField(
    value=mobileNumber,
    onChange={mobileNumber=it},
    height dp
)

Text
Spacer(modifier=Modifier. height(14. sp)
RadioGroup
    )(text=

    'Gender:',fontSize=20. )
    (
        options=genderOptions,
        selectedOption=selectedGender, height dp
        onChange={selectedGender=it}
    )
Text
RadioGroup sp

Spacer(modifier=Modifier. height(14. dp))

    (text='Diabetics:',fontSize=20. )
    (
        options=diabeticsOptions,
        selectedOption=selectedDiabetics,
        onChange={selectedDiabetics=it}
    )
Text
padding dp

    (
        // Display Submit button
        text=error,
        textAlign=TextAlign.Center,
        modifier=Modifier. isNotEmpty, isNotEmpty, isNotEmpty, isNotEmpty
    )
    isNotEmpty

Button(
    onClick={ if(name. ()&&age. ()&&mobileNumber.
                ()&&genderOptions. ()&&diabeticsOptions.
    )}{
        val survey=Survey(
            id=null,
            name=name,
            age=age,
            mobileNumber=mobileNumber,
            gender=selectedGender,
            diabetics=selectedDiabetics
        )
        databaseHelper.insertSurvey(survey)
    }
}

```

```

        error="SurveyCompleted"

    }else{
        error="Pleasefillallfields"
    }
},
colors=ButtonDefaults.buttonColors(backgroundColor=
Color(0xFF84adb8)),
modifier=Modifier. padding(start=70. dp).size(height=60. dp,
width=200. dp)
){
    Text(text="Submit")
}
}
}
@Composable
funRadioGroup(
    options:List<String>,
    selectedOption:String?,
    onSelectedChange:(String)->Unit
){
    Column{
        options. forEach {option->
            Row(
                Modifier
                    .fillMaxWidth()
                    .padding(horizontal=5. dp)
            ){
                RadioButton(
                    selected=option==selectedOption,
                    onClick={onSelectedChange(option)}
                )
                Text(
                    text=option,
                    style=MaterialTheme.typography.body1.merge(),
                    modifier=Modifier. padding(top=10. dp),
                    fontSize=17. sp
                )
            }
        }
    }
}
}

```

## RegisterActivity.kt

```
packagecom.example.surveyapplication
```

```

importandroid.content.Context
importandroid.content.Intent
importandroid.os.Bundle
importandroidx.activity.ComponentActivity
importandroidx.activity.compose.setContent
importandroidx.compose.foundation.Image
importandroidx.compose.foundation.background
importandroidx.compose.foundation.layout.*
importandroidx.compose.material.*
importandroidx.compose.runtime.*
importandroidx.compose.ui.Alignment
importandroidx.compose.ui.Modifier

```



```

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.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

```

```

class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper : UserDatabaseHelper
    override fun onCreate(savedInstanceState : Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            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("") }

```

```

Column(
    modifier = Modifier.fillMaxSize().background(Color.White),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
){

```

```

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

```

```

    Text(
        fontSize = 36.sp,
        fontWeight = FontWeight.ExtraBold,
        fontFamily = FontFamily.Cursive,
        color = Color(0xFF25b897),
        text = "Register"
    )

```

```

    Spacer(modifier = Modifier.height(10.dp))
    TextField

```

```

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

```

```

)

```

```

        TextField(
            value=email,
            onChange={email=it},
            label={Text('Email')},
            modifier=Modifier
                .padding(10.dp)
                .width(280.dp)
        )

        TextField(
            value=password,
            onChange={password=it},
            label={Text('Password')},
            visualTransformation=PasswordVisualTransformation(),
            modifier=Modifier
                .padding(10.dp)
                .width(280.dp)
        )

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

    Button(
        onClick={
            if(username.isNotEmpty() && password.isNotEmpty() && email.
isNotEmpty()){
                val user=User(
                    id=null,
                    firstName=username,
                    lastName=null,
                    email=email,
                    password=password
                )
                databaseHelper.insertUser(user)
                error='User registered successfully'

                context.startActivity(
                    // Start LoginActivity using the current context
                    Intent(context, LoginActivity::class.java)
                )
            }
            else{
                error='Please fill all fields'
            }
        },
        colors=ButtonDefaults.buttonColors(background-color=
Color(0xFF84adb8)),
        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='Havean
account?'
            )
            TextButton(onClick={ context.startActivity(
                Intent(
                    context,LoginActivity::class.java
                )
            )
        })

        {
            Spacer(modifier=Modifier.width(10.dp))
            Text(color=Color(0xFF25b897),text='Login')
        }
    }
}

private fun startLoginActivity(context: Context){
    val intent=Intent(context,LoginActivity::class.java)
    ContextCompat.startActivity(context,intent,null)
}

```

## Survey.kt

```
package com.example.surveyapplication
```

```

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

```

```

@Entity(tableName='survey_table')
data class Survey(
    @PrimaryKey(autoGenerate=true) val id: Int?,
    @ColumnInfo(name='name') val name: String?,
    @ColumnInfo(name='age') val age: String?,
    @ColumnInfo(name='mobile_number') val mobileNumber: String?,
    @ColumnInfo(name='gender') val gender: String?,
    @ColumnInfo(name='diabetics') val diabetics: String?,

)

```

## SurveyDao.kt

```
package com.example.surveyapplication
```

```
import androidx.room.*
```

```
@Dao
```

```
interface SurveyDao{
```

```
    @Query('SELECT*FROMsurvey_tableWHEREage=:age')
```

```

suspend fun getUserByAge(age: String): Survey?

@Insert(onConflict=OnConflictStrategy.REPLACE)
suspend fun insertSurvey(survey: Survey)

@Update
suspend fun updateSurvey(survey: Survey)

@Delete
suspend fun deleteSurvey(survey: Survey)
}

```

## SurveyDatabase.kt

```

package com.example.surveyapplication

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

@Database(entities=[Survey::class], version=1)
abstract class SurveyDatabase: RoomDatabase(){

    abstract fun surveyDao(): SurveyDao

    companion object{

        @Volatile
        private var instance: SurveyDatabase? = null

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

```

## SurveyDatabaseHelper.kt

```

package com.example.surveyapplication

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 SurveyDatabaseHelper(context: Context):
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION){

```



```

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

    private const val TABLE_NAME = "survey_table"
    private const val COLUMN_ID = "id"
    private const val COLUMN_NAME = "name"
    private const val COLUMN_AGE = "age"
    private const val COLUMN_MOBILE_NUMBER = "mobile_number"
    private const val COLUMN_GENDER = "gender"
    private const val COLUMN_DIABETICS = "diabetics"
}

override fun onCreate(db: SQLiteDatabase?) {
    val createTable = "CREATE TABLE $TABLE_NAME (" +
        "$COLUMN_ID INTEGER PRIMARY KEY AUTO INCREMENT, " +
        "$COLUMN_NAME TEXT, " +
        "$COLUMN_AGE TEXT, " +
        "$COLUMN_MOBILE_NUMBER TEXT, " +
        "$COLUMN_GENDER TEXT, " +
        "$COLUMN_DIABETICS TEXT" +
        ")"

    db?.execSQL(createTable)
}

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

fun insertSurvey(survey: Survey) {
    val db =
    val values = ContentValues()
    values.put(COLUMN_NAME, survey.name)
    values.put(COLUMN_AGE, survey.age)
    values.put(COLUMN_MOBILE_NUMBER, survey.mobileNumber)
    values.put(COLUMN_GENDER, survey.gender)
    values.put(COLUMN_DIABETICS, survey.diabetics)
    db.insert(TABLE_NAME, null, values)
    db.close()
}

@SuppressLint("Range")
fun getSurveyByAge(age: String): Survey? {
    val db =
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_AGE=?", readableDatabase(age))
    var survey: Survey? = null
    if (cursor.moveToFirst()) {
        survey = Survey(
            id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            name = cursor.getString(cursor.getColumnIndex(COLUMN_NAME)),

            age = cursor.getString(cursor.getColumnIndex(COLUMN_AGE)),
            mobileNumber =
            cursor.getString(cursor.getColumnIndex(COLUMN_MOBILE_NUMBER)),
            gender = cursor.getString(cursor.getColumnIndex(COLUMN_GENDER))
        ),
        diabetics =

```

```

        cursor.getString(cursor.getColumnIndex(COLUMN_DIABETICS)),
    )
    }
    cursor.close()
    db.close()
    return survey
}
@SuppressLint('Range')
fun getSurveyById(id: Int): Survey? {
    val db = readableDatabase
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_ID=?", arrayOf(id.toString()))
    var survey: Survey? = null
    if (cursor.moveToFirst()) {
        survey = Survey(
            id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
            name = cursor.getString(cursor.getColumnIndex(COLUMN_NAME)),

            age = cursor.getString(cursor.getColumnIndex(COLUMN_AGE)),
            mobileNumber =
            cursor.getString(cursor.getColumnIndex(COLUMN_MOBILE_NUMBER)),
            gender = cursor.getString(cursor.getColumnIndex(COLUMN_GENDER))
        ),
            diabetics =
            cursor.getString(cursor.getColumnIndex(COLUMN_DIABETICS)),
        )
    }
    cursor.close()
    db.close()
    return survey
}

@SuppressLint('Range')
fun getAllSurveys(): List<Survey> {
    val surveys = <Survey>()
    val db = mutableList<Survey>()
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
    if (cursor.moveToFirst()) {
        do {
            val survey = Survey(
                cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                cursor.getString(cursor.getColumnIndex(COLUMN_NAME)),
                cursor.getString(cursor.getColumnIndex(COLUMN_AGE)),

                cursor.getString(cursor.getColumnIndex(COLUMN_MOBILE_NUMBER)),
                cursor.getString(cursor.getColumnIndex(COLUMN_GENDER)),

                cursor.getString(cursor.getColumnIndex(COLUMN_DIABETICS))
            )
            surveys.add(survey)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return surveys
}
}

```



## User.kt

```
package com.example.surveyapplication

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.surveyapplication

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.surveyapplication

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, UserDatabase::class,
                "user_database"
            ).build()
            instance = newInstance
        }
    }
}

```

## UserDatabaseHelper.kt

```
package com.example.surveyapplication
```

```

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 AUTO INCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$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 =
    val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
    $COLUMN_FIRST_NAME = ?", (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 =
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
        $COLUMN_ID = ?", (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

```

```

        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
    }
}

```

## ExampleInstrumentedTest.kt

```

package com.example.surveyapplication

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext =
            InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.surveyapplication",
            appContext.packageName)
    }
}

```

## ExampleUnitTest.kt

```

package com.example.surveyapplication

import org.junit.Test

```

```
import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine
 * (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest { @Test
    fun addition_isCorrect() {
        assertEquals(4, 2+2)
    }
}
```

---