**PROJECT REPORT TEMPLATE**

**1 INTRODUCTION**

* 1. **Overview**

Surveys are **a method of gathering information from a group of individuals by asking them questions**. Surveys can be conducted through various mediums such as paper and pencil, online forms, telephone, or face-to-face interviews.

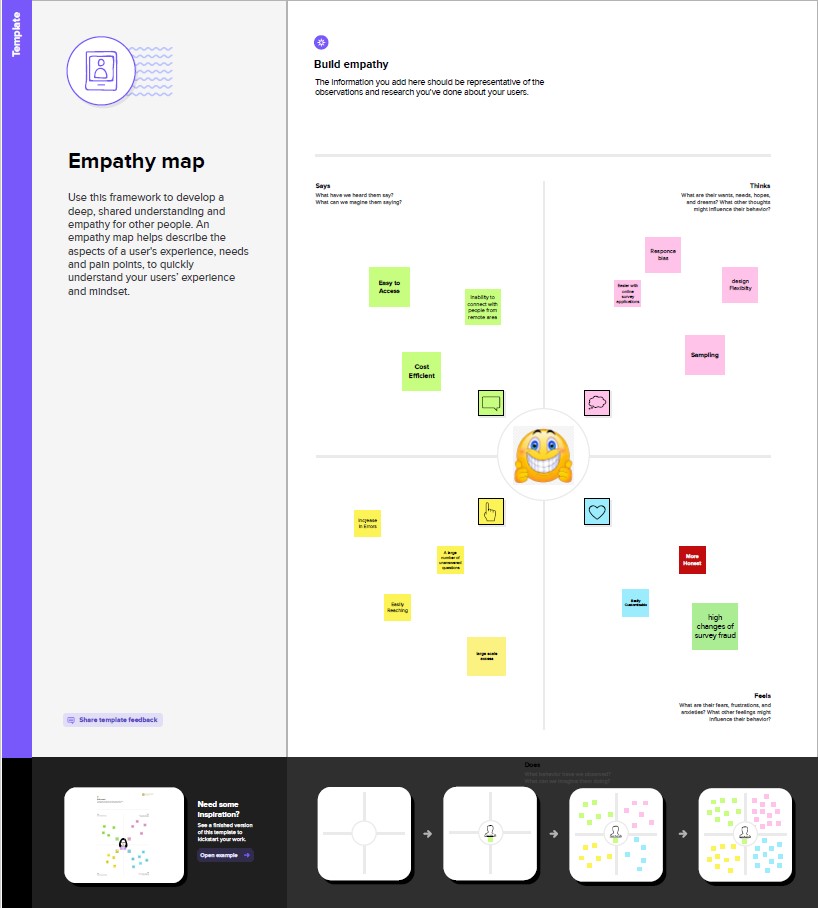
Surveys are a powerful tool for data collection, enabling us to collect valuable data quickly and efficiently. By understanding the purpose and process of surveys, we can gather accurate and meaningful data that can inform our decisions and lead to successful outcomes.

* 1. **Purpose**

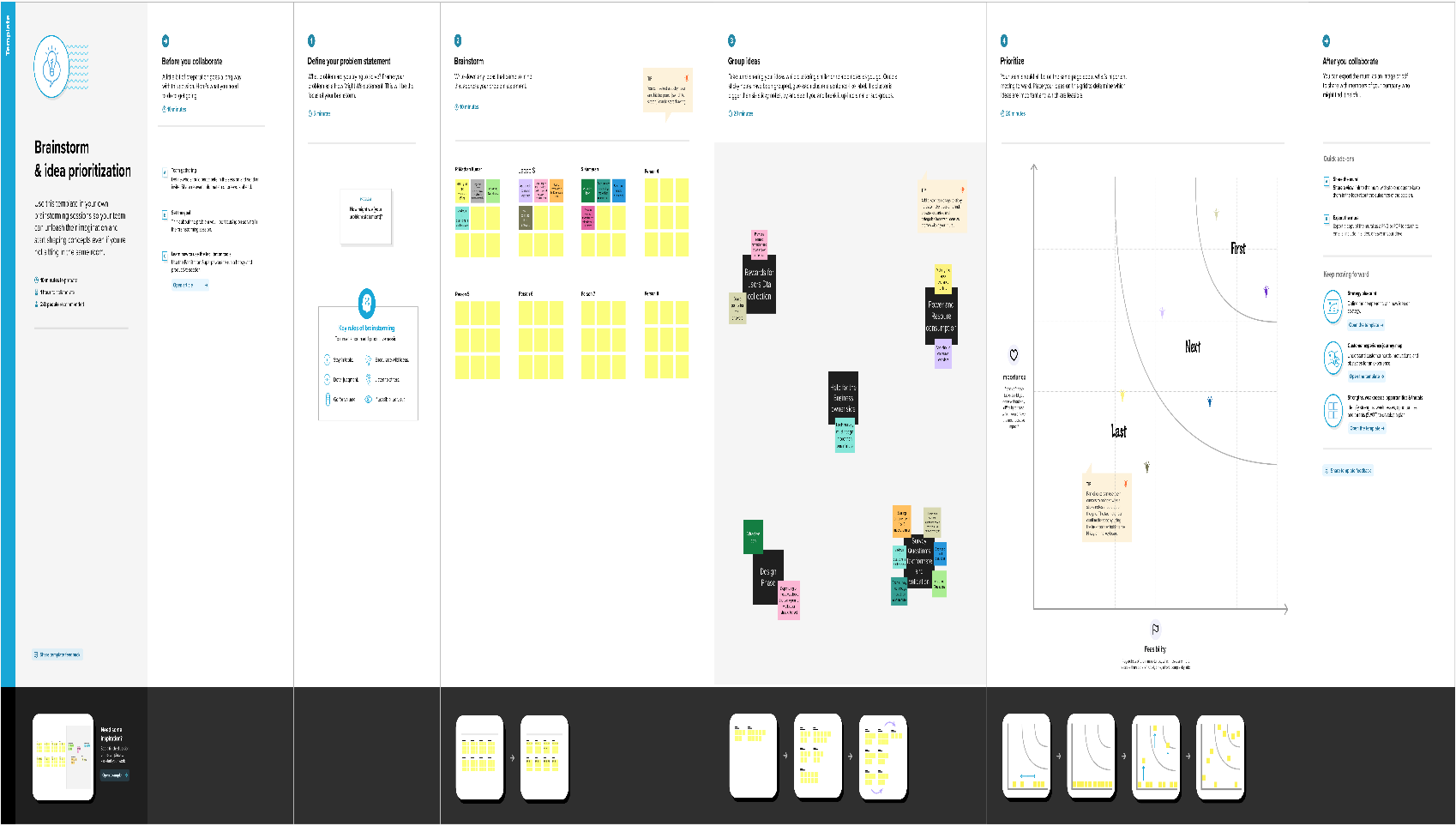
Survey apps are applications that enable people to take surveys on their smartphone or tablet, even when the device is not connected to the internet. They are used **to collect feedback, design, send and analyse surveys**.

The main goal of a survey is **to collect data that is representative of the group being surveyed, allowing researchers to make informed decisions or draw conclusions**.

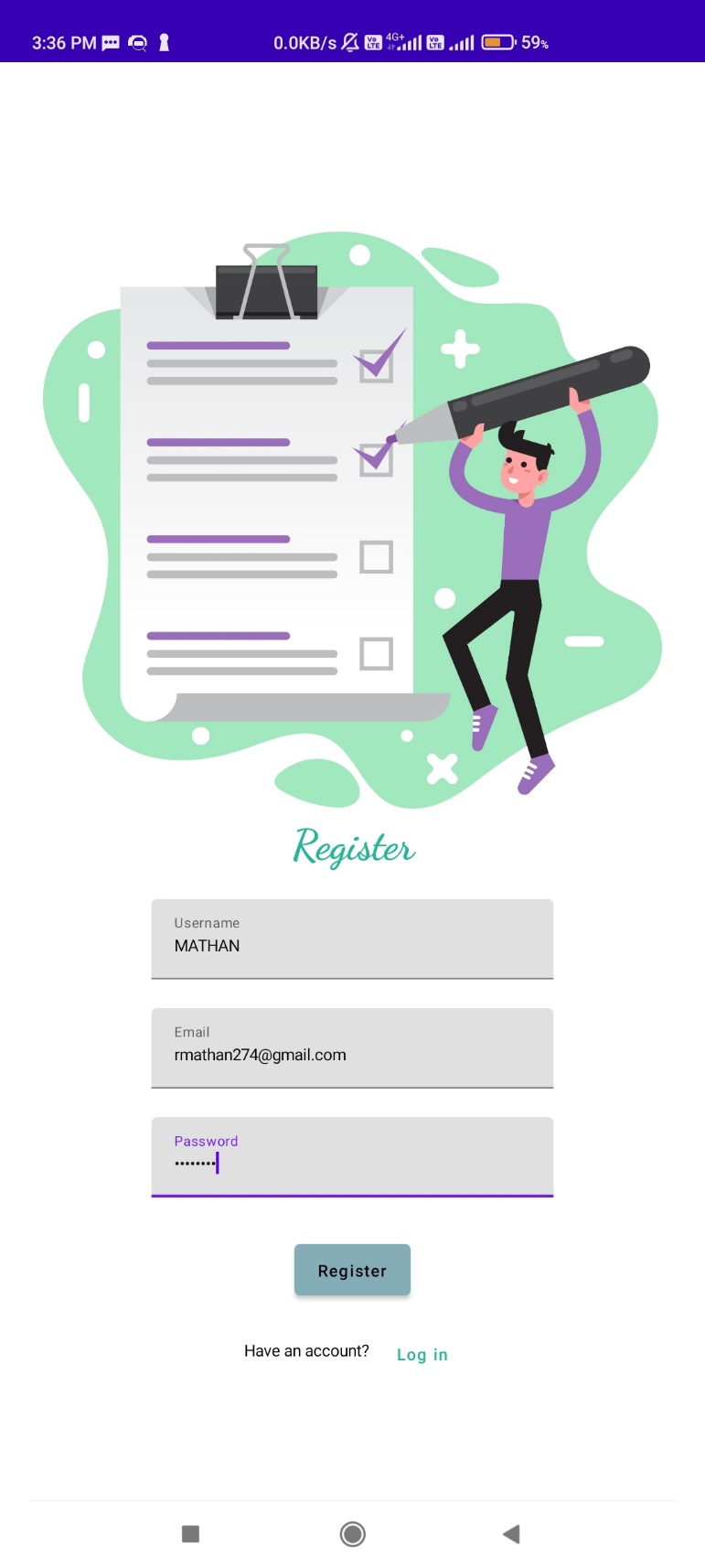
1. **Problem Definition & Design Thinking**
   1. **Empathy Map**

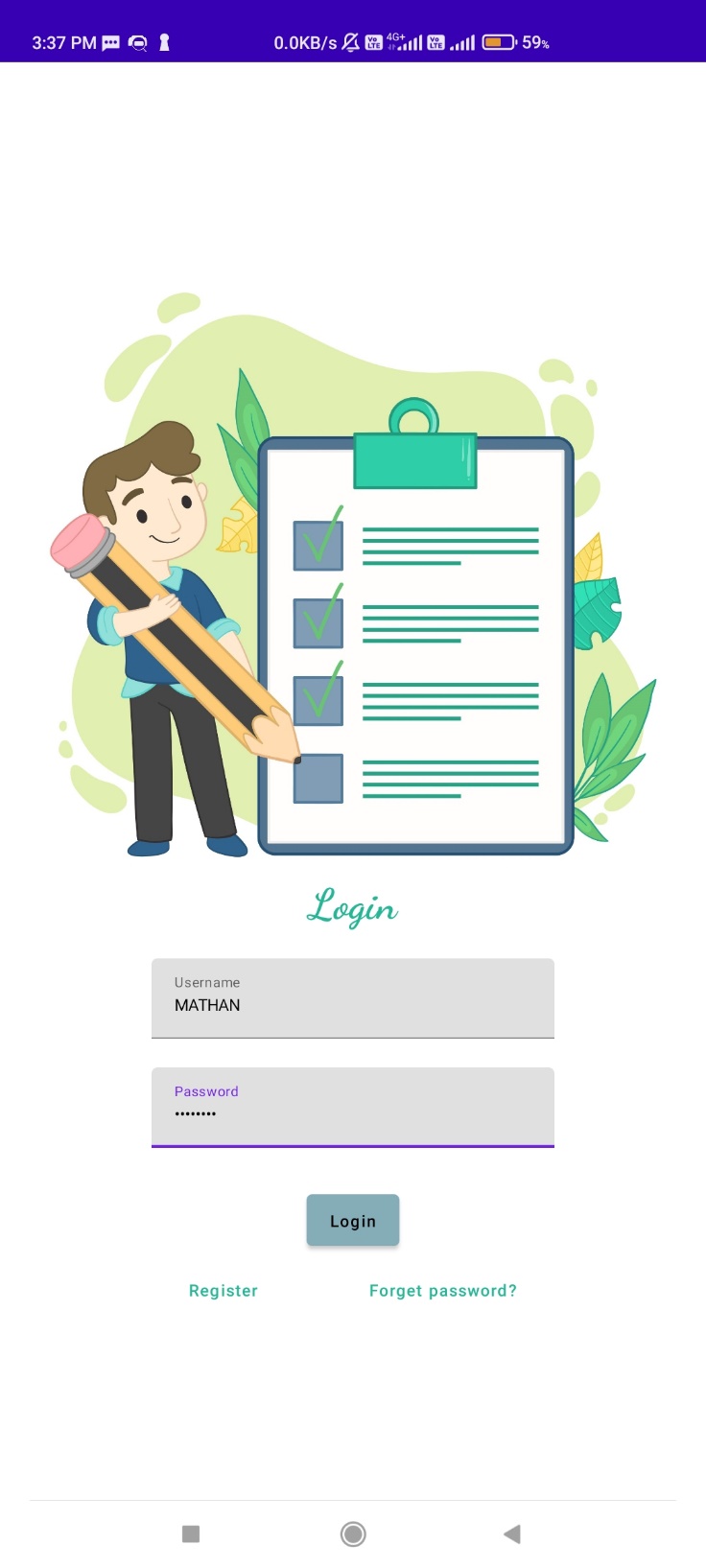


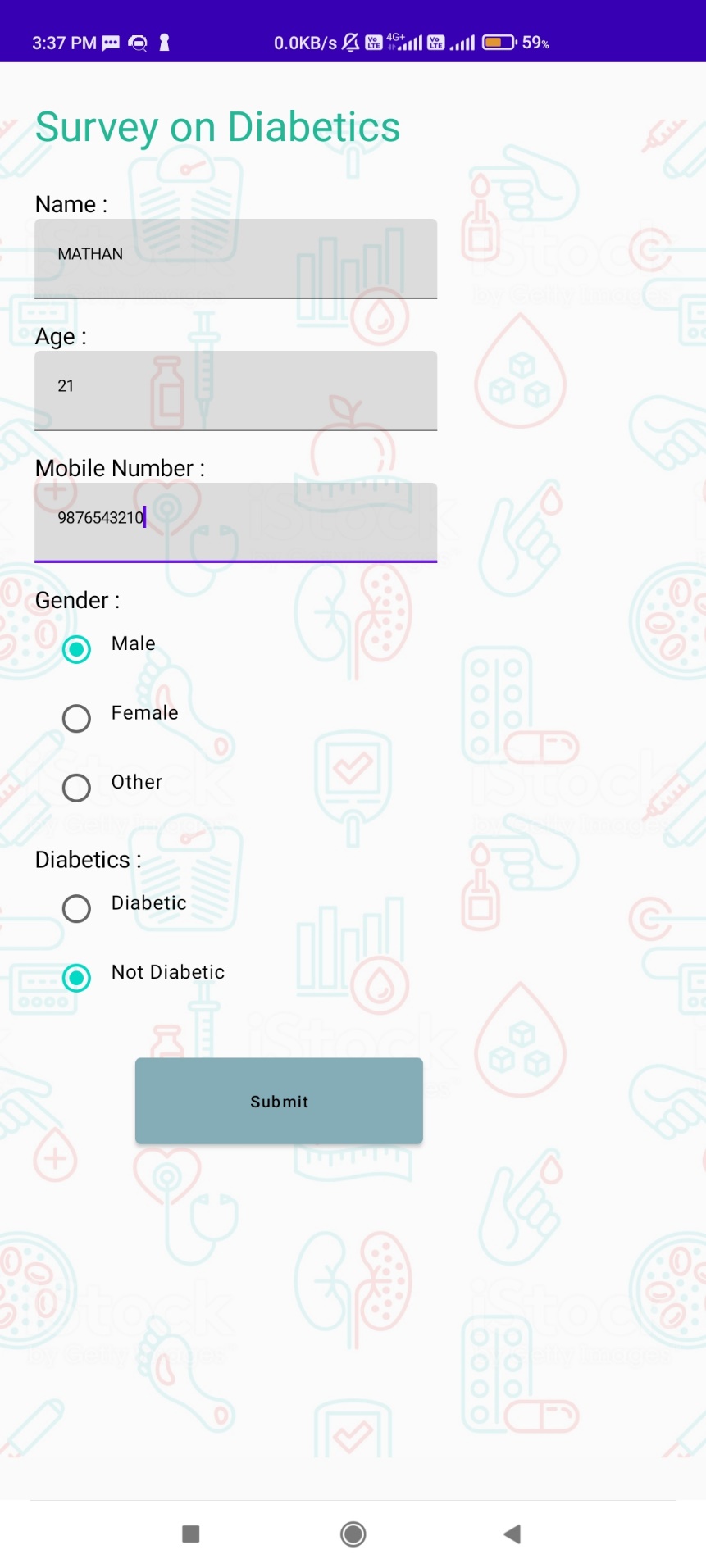
* 1. **Ideation & Brainstorming Map**



1. **RESULT**

****

****

****

****

**Advantage :-**

Advantages of surveys include **having a large population and therefore a greater statistical power, the ability to gather large amounts of information and having the availability of validated models**.

**Disadvantage :-**

* **Respondents may not feel encouraged to provide accurate, honest answers.**
* **Respondents may not be fully aware of their reasons for any given answer because of lack of memory on the subject, or even boredom.**

1. **Application :-**

The materials and methods section should include **a clear and brief description of your research procedures**. One important purpose of this section is to convince the readers that your work is valid. Another purpose is for researchers to use your methodology to guide his or her own experiments.

Survey application show the list entire data of user.

1. **Conclusion :-**

If an app is being built primarily for the Android platform, then using Material Design is an easy choice. Because of Google’s widespread adoption, any app based on Material Design principles is going to feel like a native app. Everyone can easily access this application in every where

1. **Future Scope:-**

In this way, Android app developers will provide a more enhanced personalized experience to users. Future applications may integrate different AI features such as text, image classification, voice recognition, predictive maintenance, face detection, etc. The activities and items covered by the survey or inspection was the major role in survey.

**8.Appendix**

**Source code :-**

**Paste the git hub link here**

**code :-**

User

package com.example.composeinput  
  
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

package com.example.composeinput  
  
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

package com.example.composeinput  
  
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

package com.example.composeinput  
  
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  
 }  
  
}

Survey:

package com.example.composeinput  
  
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

package com.example.composeinput  
  
import androidx.room.\*  
  
@Dao  
interface SurveyDao {  
  
 @Query("SELECT \* FROM survey\_table WHERE age = :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

package com.example.composeinput  
  
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.*applicationContext*,  
 SurveyDatabase::class.*java*,  
 "user\_database"  
 ).build()  
 instance = newInstance  
 newInstance  
 **}** }  
 }  
}

SurveyDatabaseHelper

package com.example.composeinput  
  
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 AUTOINCREMENT, " +  
 "$COLUMN\_NAME TEXT, " +  
 "$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 = *writableDatabase* 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 = *readableDatabase* val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_AGE = ?", *arrayOf*(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 = *mutableListOf*<Survey>()  
 val db = *readableDatabase* 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  
 }  
  
}

LoginActivity

package com.example.composeinput  
  
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.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.text.input.PasswordVisualTransformation  
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.composeinput.ui.theme.ComposeinputTheme  
  
class LoginActivity : ComponentActivity() {  
 private lateinit var databaseHelper: UserDatabaseHelper  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 databaseHelper = UserDatabaseHelper(this)  
 *setContent* **{** *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 log in"  
 context.startActivity(  
 Intent(  
 context,  
 MainActivity::class.*java* )  
 )  
 //onLoginSuccess()  
 }  
 if (user != null && user.password == "admin") {  
 error = "Successfully log in"  
 context.startActivity(  
 Intent(  
 context,  
 AdminActivity::class.*java* )  
 )  
 }  
 else {  
 error = "Invalid username or password"  
 }  
  
 } else {  
 error = "Please fill all fields"  
 }  
 **}**,  
 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 = "Forget password?")  
 **}  
 }  
 }**}  
private fun startMainPage(context: Context) {  
 val intent = Intent(context, MainActivity::class.*java*)  
 ContextCompat.startActivity(context, intent, null)  
}

RegisterActivity

package com.example.composeinput  
  
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.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.text.input.PasswordVisualTransformation  
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.composeinput.ui.theme.ComposeinputTheme  
  
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*(  
 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") **}**,  
 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"  
 // Start LoginActivity using the current context  
 context.startActivity(  
 Intent(  
 context,  
 LoginActivity::class.*java* )  
 )  
  
 } else {  
 error = "Please fill all fields"  
 }  
 **}**,  
 colors = ButtonDefaults.buttonColors(backgroundColor = *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 = "Have an account?"  
 )  
 *TextButton*(onClick = **{** context.startActivity(  
 Intent(  
 context,  
 LoginActivity::class.*java* )  
 )  
 **}**)  
  
 **{** *Spacer*(modifier = Modifier.*width*(10.*dp*))  
 *Text*( color = *Color*(0xFF25b897),text = "Log in")  
 **}  
 }  
 }**}  
private fun startLoginActivity(context: Context) {  
 val intent = Intent(context, LoginActivity::class.*java*)  
 ContextCompat.startActivity(context, intent, null)  
}

MainActivity

package com.example.composeinput  
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.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.composeinput.ui.theme.ComposeinputTheme  
  
class MainActivity : ComponentActivity() {  
 private lateinit var databaseHelper: SurveyDatabaseHelper  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 databaseHelper = SurveyDatabaseHelper(this)  
 *setContent* **{** *FormScreen*(this, databaseHelper)  
 **}** }  
}  
  
@Composable  
fun FormScreen(context: Context, databaseHelper: 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*("") **}** 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,  
 onValueChange = **{** name = **it }**,  
 )  
  
 *Spacer*(modifier = Modifier.*height*(14.*dp*))  
  
 *Text*(text = "Age :", fontSize = 20.*sp*)  
 *TextField*(  
 value = age,  
 onValueChange = **{** age = **it }**,  
 )  
  
 *Spacer*(modifier = Modifier.*height*(14.*dp*))  
  
 *Text*(text = "Mobile Number :", fontSize = 20.*sp*)  
 *TextField*(  
 value = mobileNumber,  
 onValueChange = **{** mobileNumber = **it }**,  
 )  
  
 *Spacer*(modifier = Modifier.*height*(14.*dp*))  
  
 *Text*(text = "Gender :", fontSize = 20.*sp*)  
 *RadioGroup*(  
 options = genderOptions,  
 selectedOption = selectedGender,  
 onSelectedChange = **{** selectedGender = **it }** )  
  
 *Spacer*(modifier = Modifier.*height*(14.*dp*))  
  
 *Text*(text = "Diabetics :", fontSize = 20.*sp*)  
 *RadioGroup*(  
 options = diabeticsOptions,  
 selectedOption = selectedDiabetics,  
 onSelectedChange = **{** selectedDiabetics = **it }** )  
  
 *Text*(  
 text = error,  
 textAlign = TextAlign.Center,  
 modifier = Modifier.*padding*(bottom = 16.*dp*)  
 )  
 // Display Submit button  
 *Button*(  
 onClick = **{** if (name.*isNotEmpty*() && age.*isNotEmpty*() && mobileNumber.*isNotEmpty*() && genderOptions.*isNotEmpty*() && diabeticsOptions.*isNotEmpty*()) {  
 val survey = Survey(  
 id = null,  
 name = name,  
 age = age,  
 mobileNumber = mobileNumber,  
 gender = selectedGender,  
 diabetics = selectedDiabetics  
 )  
 databaseHelper.insertSurvey(survey)  
 error = "Survey Completed"  
 context.startActivity(  
 Intent(  
 context,  
 AdminActivity::class.*java* )  
 )  
  
 } else {  
 error = "Please fill all fields"  
 }  
 **}**,  
 colors = ButtonDefaults.buttonColors(backgroundColor = *Color*(0xFF84adb8)),  
 modifier = Modifier.*padding*(start = 70.*dp*).*size*(height = 60.*dp*, width = 200.*dp*)  
 ) **{** *Text*(text = "Submit")  
 **}  
 }**}  
@Composable  
fun RadioGroup(  
 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* )  
 **}  
 }  
 }**}

AdminActivity

package com.example.composeinput  
  
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.composeinput.ui.theme.ComposeinputTheme  
  
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 = "Survey Details",  
 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}")  
 **}  
 }  
 }  
 }  
 }**}

AndroidManiFest

<?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.Composeinput"  
 tools:targetApi="31">  
 <activity  
 android:name=".RegisterActivity"  
 android:exported="false"  
 android:label="@string/title\_activity\_register"  
 android:theme="@style/Theme.Composeinput" />  
 <activity  
 android:name=".MainActivity"  
 android:exported="false"  
 android:label="MainActivity"  
 android:theme="@style/Theme.Composeinput" />  
 <activity  
 android:name=".AdminActivity"  
 android:exported="false"  
 android:label="@string/title\_activity\_admin"  
 android:theme="@style/Theme.Composeinput" />  
 <activity  
 android:name=".LoginActivity"  
 android:exported="true"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.Composeinput">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>