PROJECT REPORT

AN ANDOID APPLICATION TO TO TAKE THE SURVEY OF DIABETICS PATIENTS

1.INTRODUCTION

1.1 Overview

The app's main aim is to take survey of number of people affected in Diabetics. The user can enter the detail and organization can enroll the and Save the detail of people. This app use compose Input : Demonstration of Text input and validation with Android compose. This app is fetch the input the room database and demonstrate how to use the Jetpack compose UI Android Development.

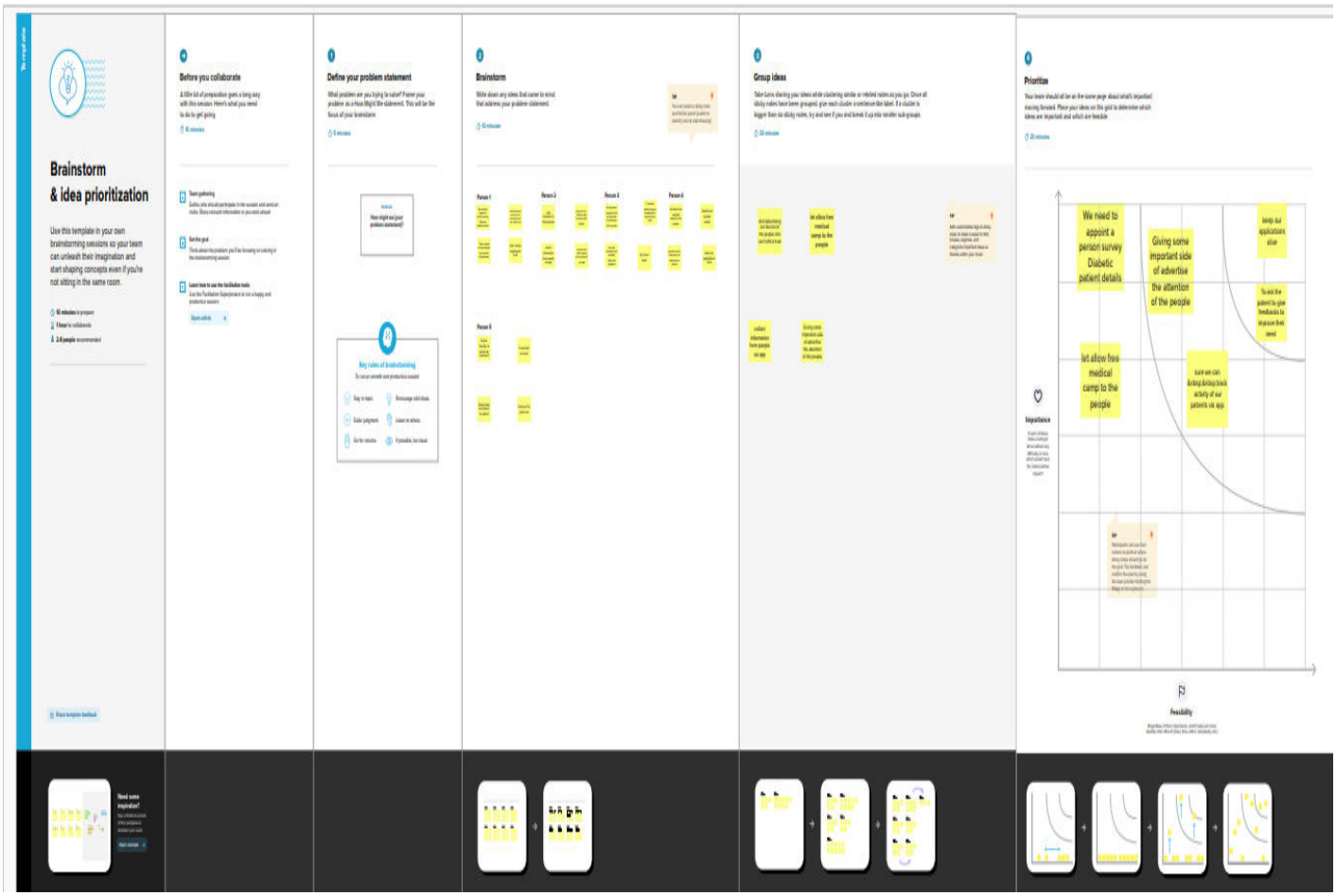
1.2 Purpose

This project is used to demonstrate to collect the information from people. Survey results provide insights on trends that health care providers can apply in their own practices and that the diabetes community can use to reach populations affected by diabetes. Data from the National Diabetes Survey may complement statistics on diabetes prevalence and cost collected by other organizations.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map

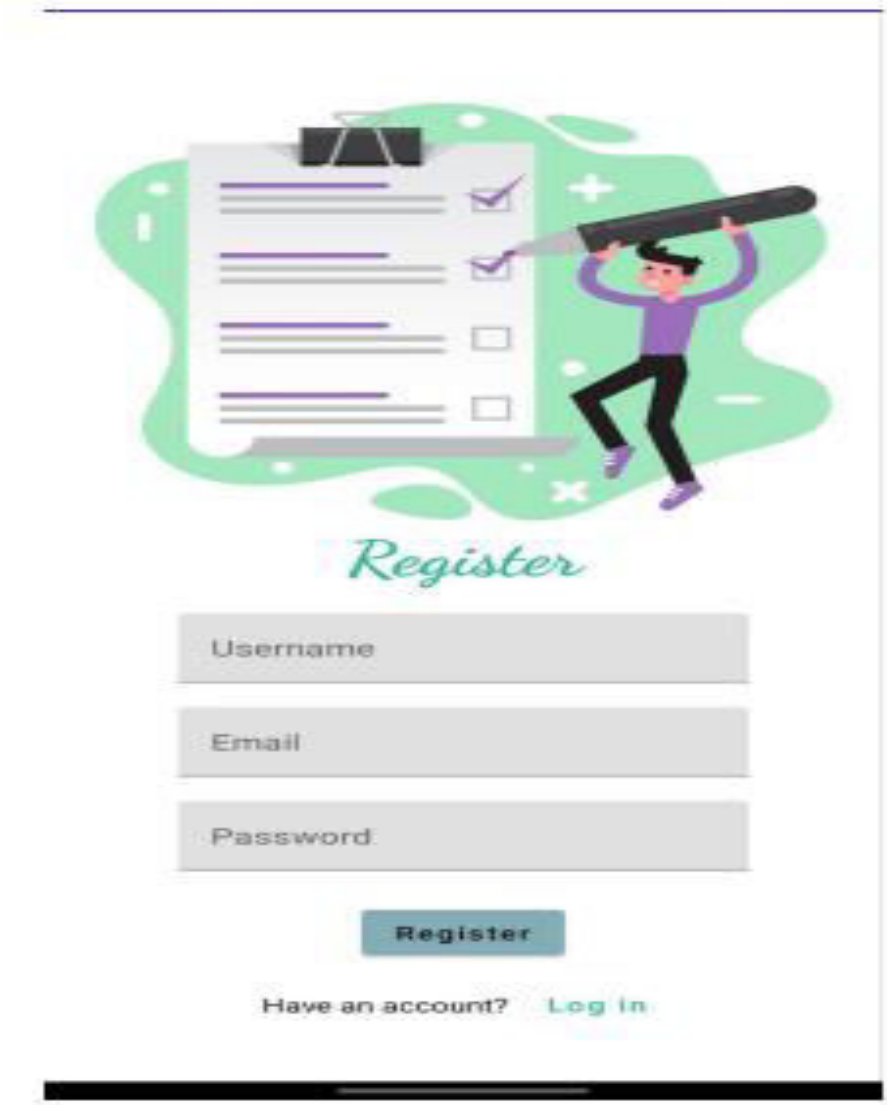
2.2 Ideation & Brainstorming Map



3.RESULT

LOGIN PAGE

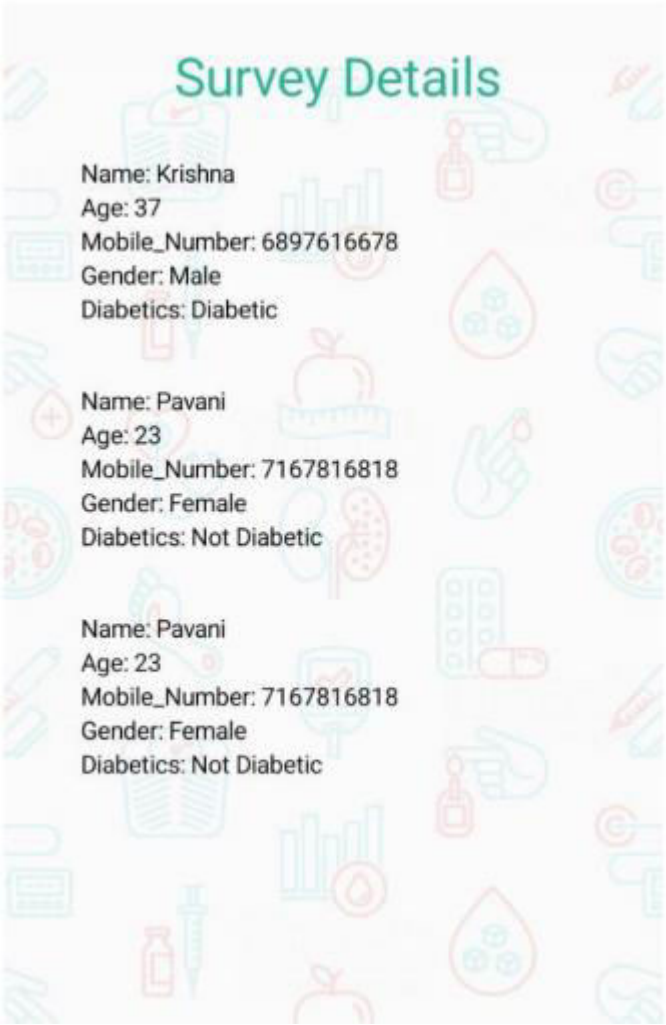
REGISTER PAGE



MAIN PAGE


Survey on Diabetics
Name :
Age :
Mobile Number :
Gender :
Male
Female
Other
Diabetics :
Diabetic
Not Diabetic
Submit

ADMIN PAGE


Survey Details
Name: Krishna
Age: 37
Mobile_Number: 6897616678
Gender: Male
Diabetics: Diabetic
Name: Pavani
Age: 23
Mobile_Number: 7167816818
Gender: Female
Diabetics: Not Diabetic
Name: Pavani
Age: 23
Mobile_Number: 7167816818
Gender: Female
Diabetics: Not Diabetic

4.ADVANTAGES & DISADVANTAGES

4.1 ADVANTAGES

1) Easy to use

2) Call appointment

3) patient can Track their Health

4.2 DISADVANTAGES

1) Not Flexible

2) Not good As Get diagosis

5.APPLICATION

User input validation: By demonstrating how to validate user input, developers can ensure that the app only accepts valid input, which improves the overall user experience. Form submission: With text input and validation, developers can easily create forms that allow users to submit data. This feature is useful in various applications, such as e-commerce, banking, or health care. User registration: User registration is a crucial feature in many applications. By demonstrating text input and validation with Android Compose, developers can ensure that the registration process is smooth and error-free. Messaging apps: Messaging apps rely heavily on text input and validation. By demonstrating how to handle user input, developers can ensure that messages are delivered correctly and without errors. Search functionality: Search functionality is an essential feature of many applications, such as e-commerce, social media, and news apps. By demonstrating text input and validation with Android Compose, developers can ensure that the search functionality works correctly and provides accurate results. User input validation: By demonstrating how to validate user input, developers can ensure that the app only

accepts valid input, which improves the overall user experience.

Form submission: With text input and validation, developers can easily create forms that allow users to submit

data. This feature is useful in various applications, such as e-commerce, banking, or health care.

User registration: User registration is a crucial feature in many applications. By demonstrating text input and

validation with Android Compose, developers can ensure that the registration process is smooth and error-free.

Messaging apps: Messaging apps rely heavily on text input and validation. By demonstrating how to handle user

input, developers can ensure that messages are delivered correctly and without errors.

Search functionality: Search functionality is an essential feature of many applications, such as e-commerce,

social media, and news apps. By demonstrating text input and validation with Android Compose, developers can

ensure that the search functionality works correctly and provides accurate results.

6.CONCLUSION

demonstrating text input and validation with Android Compose is an essential aspect of creating high-quality Android applications. By implementing robust text input and validation features, developers can ensure that user input is handled correctly, improving the overall user experience. This is especially important in applications that rely heavily on user input, such as messaging apps, search functionality, and forms. The use of Android Compose makes implementing these features easier and more efficient, allowing developers to focus on creating user-friendly applications. Overall, by demonstrating text input and validation with Android Compose, developers can ensure that their applications are of high quality, user-friendly, and error-free. demonstrating text input and validation with Android Compose is an essential aspect of creating high-quality

Android applications. By implementing robust text input and validation features, developers can ensure that user

input is handled correctly, improving the overall user experience. This is especially important in applications that

rely heavily on user input, such as messaging apps, search functionality, and forms. The use of Android

Compose makes implementing these features easier and more efficient, allowing developers to focus on

creating user-friendly applications. Overall, by demonstrating text input and validation with Android Compose,

developers can ensure that their applications are of high quality, user-friendly, and error-free.

7.FUTURE SCOPE

The future scope of demonstrating text input and validation with Android Compose is vast and promising. As technology evolves, so do the ways in which we interact with applications, and text input remains a fundamental aspect of user interaction. With the increasing use of mobile devices, the demand for high-quality mobile applications that offer robust text input and validation features is only set to grow. In the future, we can expect to see more advanced text input and validation features, such as voice recognition, machine learning-based input prediction, and natural language processing. Additionally, the use of augmented reality and virtual reality technologies in mobile applications can further enhance the text input experience. The future scope of demonstrating text input and validation with Android Compose is vast and promising. As

technology evolves, so do the ways in which we interact with applications, and text input remains a fundamental

aspect of user interaction. With the increasing use of mobile devices, the demand for high-quality mobile

applications that offer robust text input and validation features is only set to grow.

In the future, we can expect to see more advanced text input and validation features, such as voice recognition,

machine learning-based input prediction, and natural language processing. Additionally, the use of augmented

reality and virtual reality technologies in mobile applications can further enhance the text input experience.

8.APPENDIX

//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?,


    )
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)
```

}

Give feedback

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


}
```

//SurveyDao.kt

```
package com.example.surveyapplication


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)
}
```

//SurveyData.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.applicationContext,
                    SurveyDatabase::class.java,
                    "user_database"
```

```
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
//survey databasehelper .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 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.kt

```
package com.example.surveyapplication

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.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 : 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.surveyapplication

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.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(
    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.kt

```
package com.example.surveyapplication

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.surveyapplication.ui.theme.SurveyApplicationTheme

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"

        } 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.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 = "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}")
                    }
                }
            }
        }
    }
}
```