NAANMUDHALVAN SCHEME – 2023

COMPOSE A DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE

A major project report submitted to **Naan Mudhalvan Scheme – 2023**

in partial fulfilment of the requirement for the award of Degree Bachelor of Science in Computer Science.

SUBMITTED BY

TEAM LEADER:

M. BALA SATHISH – ASMSU2022010803(NANMUDHALVAN ID)

TEAM MEMBERS:

P. ANTONY PRAVIN – ASMSU2022010802(NANMUDHALVAN ID)

M. GANGATHARAN – ASMSU2022010805(NANMUDHALVAN ID)

K. ABISHEK – ASMSU2022010801(NANMUDHALVAN ID)

DEPARTMENT OF COMPUTER SCIENCE

ADITANAR COLLEGE OF ARTS AND SCIENCE

VIRAPANDIANPATNAM - 628216

PROJECT REPORT TEMPLATE

1.INTRODUCTION

1.1) Overview

A survey is a research technique used to collect data from a predetermined sample of respondents in order to gain information and insights on a variety of topics of interest.

They can serve a variety of objectives, and researchers can conduct them in a variety of ways based on the methodology used and the objective of the study.

Mobile application Survey have become an integrated part of every smartphone user across the globe.

Also people appreciate their role and participation in survey and giving in application feedback to people and services they interact with.

The Project is developing to manage the **Compose-Input of the user's information.** The details of users are stored into the database.

System that keeps track of all patients' details and enables easy access, retrieval and storage of the patient's information.

1.2) Purpose

Survey application are used **to collect feedback, design, send and analyze 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.**

It is an information system where data and information is handled in a hospital environment.

Information System helps in streamlining the patient information flow and its accessibility for doctors and other health care providers.

REGISTRATION ACTIVITY: The User's includes registration of patients, storing their details (User-name, E-mail, Password) into the application.

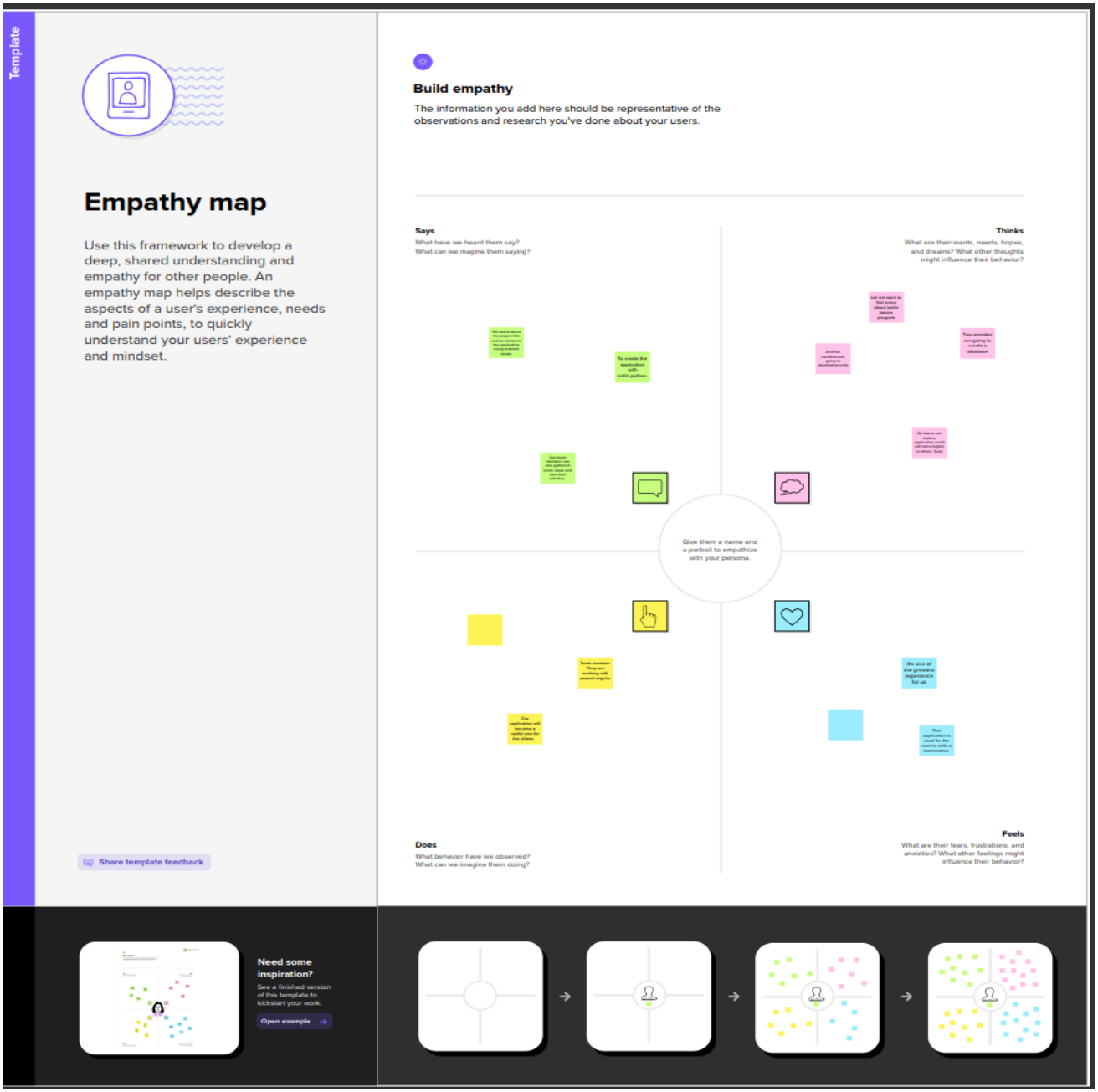
LOGIN ACTIVITY: The Users can be entered using a username and password.100%.

MAIN ACTIVITY: Ask the users to fill the valid fields and submit the application forms.

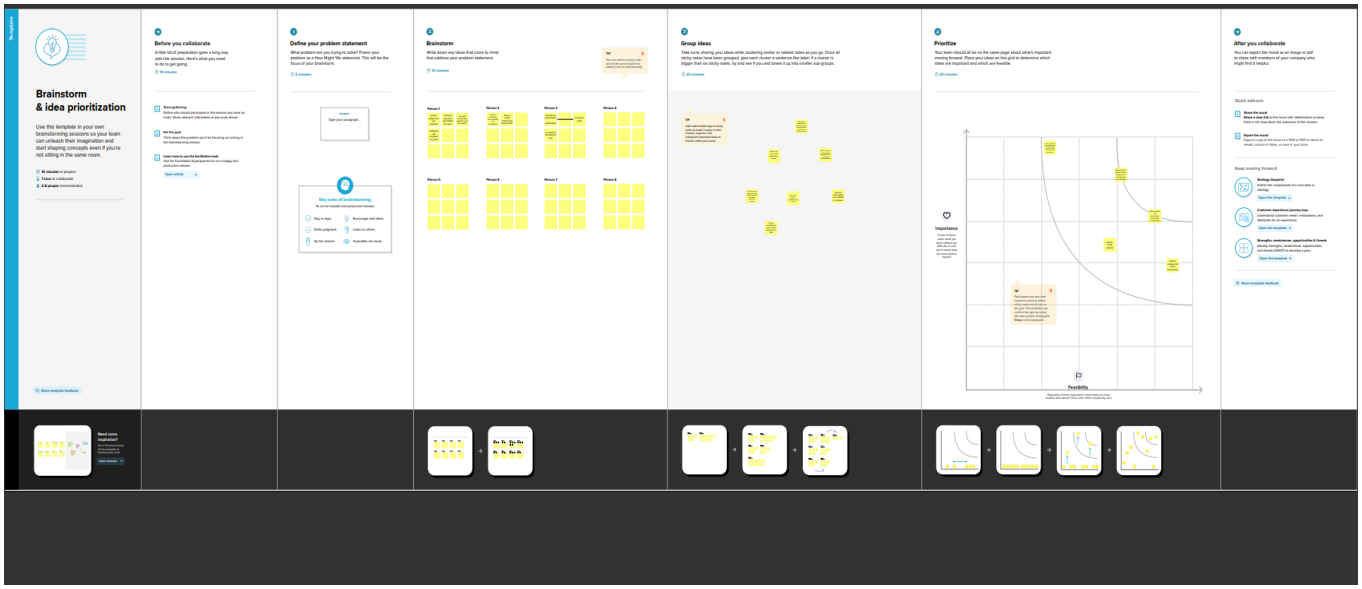
ADMIN ACTIVITY: Show the list of details to the given validation fields in the main activity application form.

2.Problem Definition & Design Thinking

2.1) Empathy Map

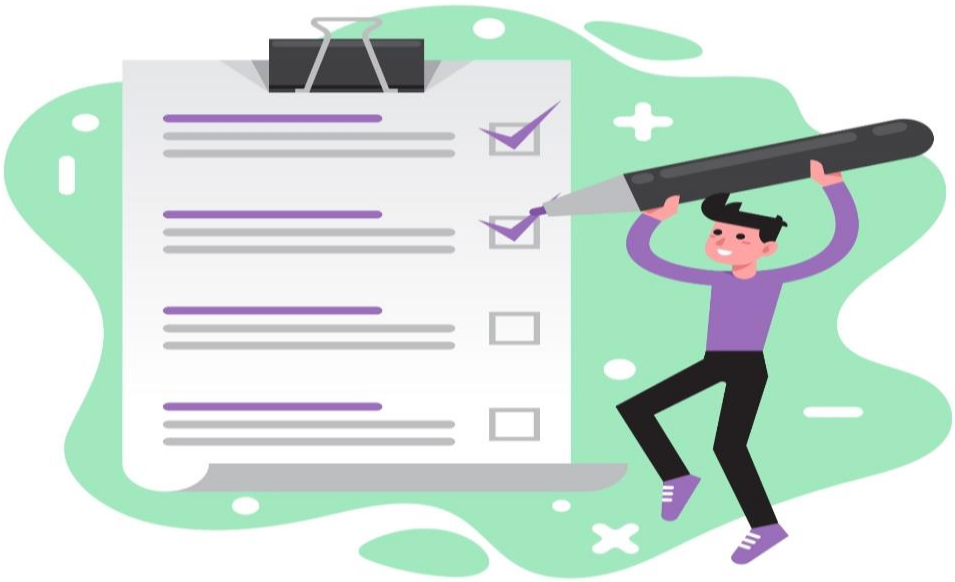


2.2) Ideation & Brainstorming Map



3.RESULT

OUTPUT 1



Register

Username
Balasathish

Email
balasathish301002@gmail.com

Password
••••

User registered successfully

Register

Have an account? Log in

OUTPUT 2

Login

Username
Balasathish

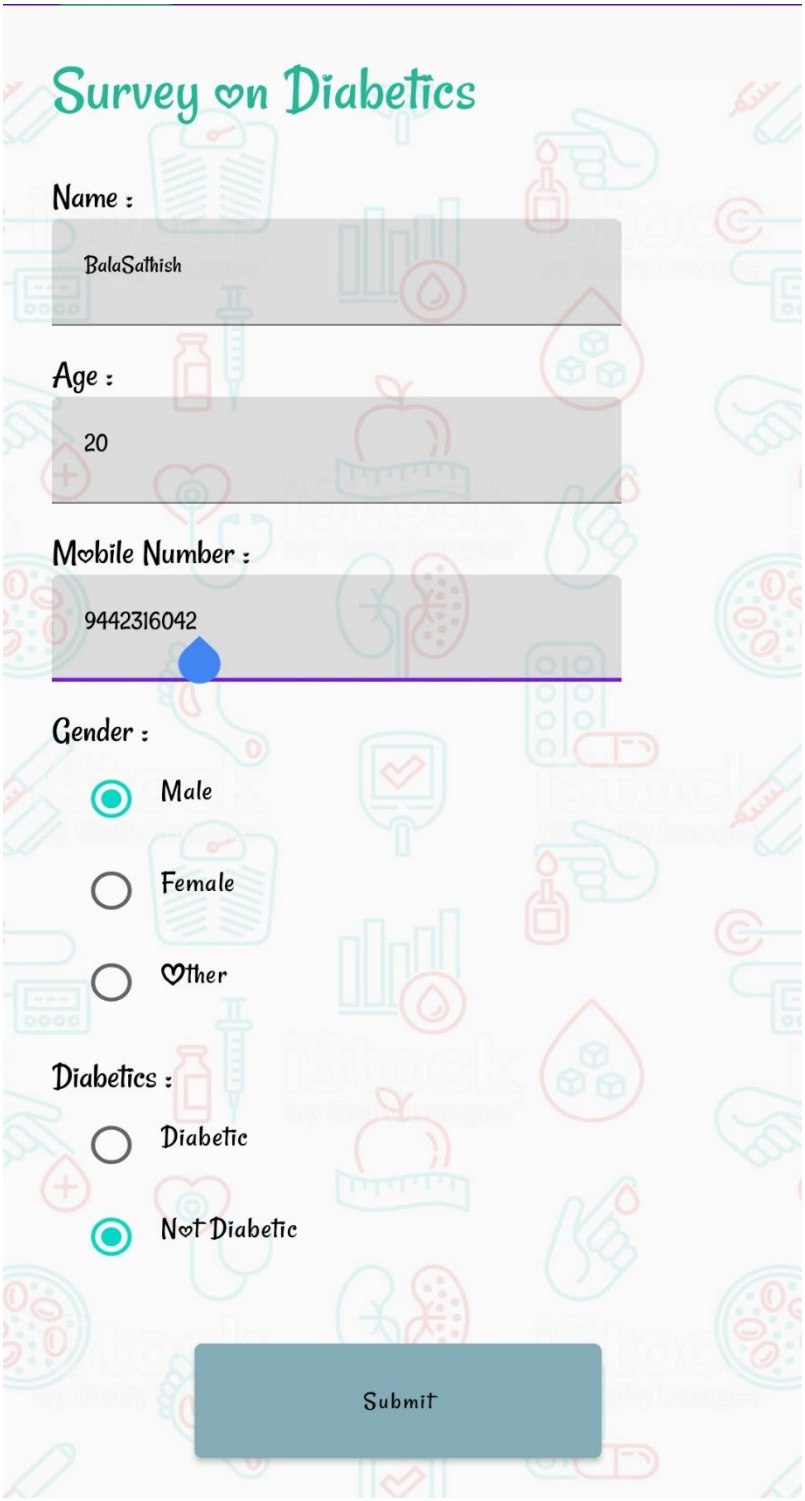
Password
••••

Login

Register

Forget password?

OUTPUT 3


Survey on Diabetics
Name :
BalaSathish
Age :
20
Mobile Number :
9442316042
Gender :
Male
Female
Other
Diabetics :
Diabetic
Not Diabetic
Submit

OUTPUT 4


Survey Details
Name: Balasathish
Age: 20
Mobile_Number: 9042316042
Gender: Male
Diabetics: Not Diabetic
Name: Abhishek
Age: 20
Mobile_Number: 9780833711
Gender: Male
Diabetics: Diabetic
Name: AntonyPravin
Age: 22
Mobile_Number: 6380833722
Gender: Male
Diabetics: Not Diabetic
Name: Gangatharan
Age: 21
Mobile_Number: 6383371120
Gender: Male
Diabetics: Diabetic

4.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Relatively easy to administer
- Can be developed in less time (compared to other data-collection methods)
- Cost-effective, but cost depends on survey mode
- Can be administered remotely via online, mobile devices or telephone.
- Advanced statistical techniques can be utilized to analyze survey data to determine validity, reliability,

DISADVANTAGES

- Respondents may not feel encouraged to provide accurate, honest answers
- Respondents may not feel comfortable providing answers that present themselves in the unfavorable manner.
- Respondents may not be fully aware of their reasons for any given answer because of lack of memory on the subject, or even boredom.
- Survey with closed-ended questions may have a lower validity rate than other question types.

5.Application

The materials and methods section are very essential to the **research and brief description of our project 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.

6.Conclusion

If an app is being built primarily for the Android-Studio platform, then using Material Design is very helpful of us.

Information System helps in streamlining the patient information flow and its accessibility for doctors and other health care providers. Everyone can easily access this application in everywhere. User can have stored their details in the application very easily and understandability. It's gives the attention to the user for their health issues.

Survey methodology is an important medical education research tool but should mainly be used to characterize unobservable, human phenomena such as emotions and opinions. Researchers should use methods other than surveys to gather observable data whenever possible.

7.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.

while AI-based technologies can analyze more information than humans can and automatically survey big databases. AI-based surveys simplify the surveying process, reduce surveyor workload and costs and increase its quality due to more data processing and evidence generation.

In future, Survey Application there were many features are available. User can interact with others.

And stored their medical repository, Reports, testing etc.

In Application, there were many options not only diabetes and Not diabetes.

Example: The user can store their previous reports, rating their heart beat levels, rating their sugar levels and so on.

8.APPENDIX

SOURCE CODE:

https://github.com/Balasathish301/Naanmuthalvan

CODING:

User.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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.kt

```
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

```
<?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>
```

THANK YOU