**Project Title**  
Compose Input: A Demonstration of Text Input and Validation with Android Compose

**1. Project Overview**

This project focuses on building a demonstration Android application using Jetpack Compose UI Toolkit. The core objective is to highlight text input handling and validation through a survey-based app interface. By utilizing Jetpack Compose, the project aims to showcase dynamic UI capabilities and enhance user experience, data handling, and form validation compared to traditional Android UI paradigms. This application serves to illustrate modern, scalable solutions for handling user input and ensuring data accuracy, thereby enhancing both user satisfaction and developer productivity.

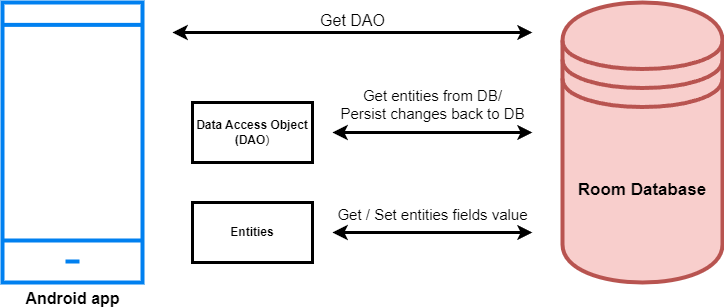
**2. Objectives**

**Business Goals:**

* Develop a fully functional, user-friendly Android application using Jetpack Compose.
* Demonstrate real-time text input validation to ensure accurate data capture.
* Improve user experience by leveraging the declarative UI features of Compose.

**Specific Outcomes:**

* Creation of an application with user registration, login, input forms, and admin data access features.
* Implementation of robust input validation mechanisms.
* Integration with a database for persistent data storage.
* Display of efficient input handling and UI updates via state management.



**3. Key Functionalities and Concepts Utilized**

**Jetpack Compose Features:**

* Declarative UI structure, creating dynamic input fields and forms.
* State management for real-time UI updates based on user input.
* Input validation using built-in TextField components and error messages.
* Navigation component integration for multi-page workflows.

**Database and Data Handling Concepts:**

* Integration with a local or cloud-based database for data persistence.
* CRUD (Create, Read, Update, Delete) operations demonstrated for user responses and admin data access.

**4. Detailed Steps to Solution Design**

1. **UI Design and User Flow**
   * Create registration, login, and survey pages using Compose Column, Row, TextField, and Button components.
   * Design user input forms with dynamic validation, error displays, and responsive layouts.
2. **ViewModel Setup**
   * Integrate ViewModel for state management and data validation.
   * Utilize LiveData or Compose's State objects to handle UI changes.
3. **Data Management and Persistence**
   * Connect with a Room Database or other data persistence solution.
   * Define data models and schemas for user registration data, survey responses, etc.
4. **Admin View Integration**
   * Develop admin views for accessing and managing survey data.
   * Implement security measures and access controls.

**PROGRAM**

**ADMIN ACTIVITY**

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}")

}

}

}

}

}

}

**LOGINACTIVITY**

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.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.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

)

)

}

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)

}

**MAINACTIVITY**

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)

)

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)

)

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

)

}

}

}

}

**REGISTER ACTIVITY**

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.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.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"

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)

}

**SURVEY KT:**

package com.example.surveyapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "survey\_table")

data class Survey(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "name") val name: String?,

@ColumnInfo(name = "age") val age: String?,

@ColumnInfo(name = "mobile\_number") val mobileNumber: String?,

@ColumnInfo(name = "gender") val gender: String?,

@ColumnInfo(name = "diabetics") val diabetics: String?,

)

**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?,

)

**5. Testing and Validation**

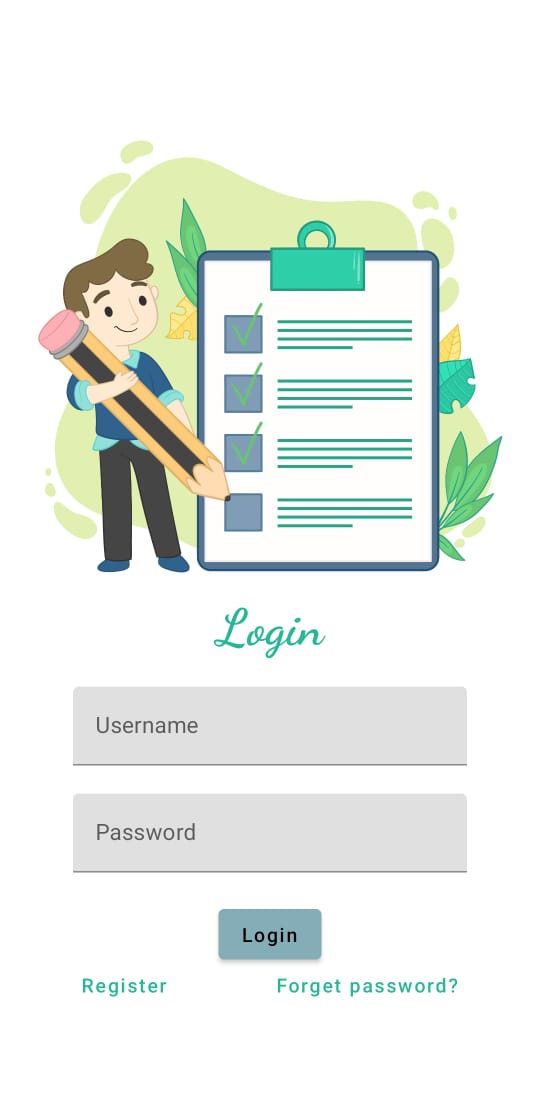
**Unit Testing**

* Implement unit tests for ViewModels, ensuring proper validation and state handling.
* Test individual Compose components for rendering and input handling.

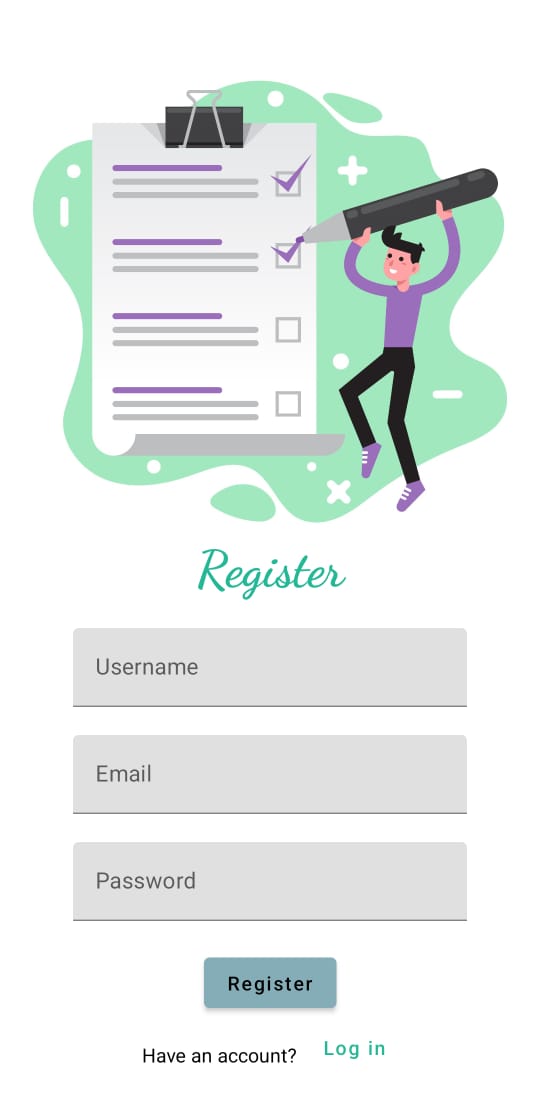
**UI Testing**

* Use Compose's testing library to simulate user interactions with the UI components.
* Validate input correctness, error handling, and navigation behavior.

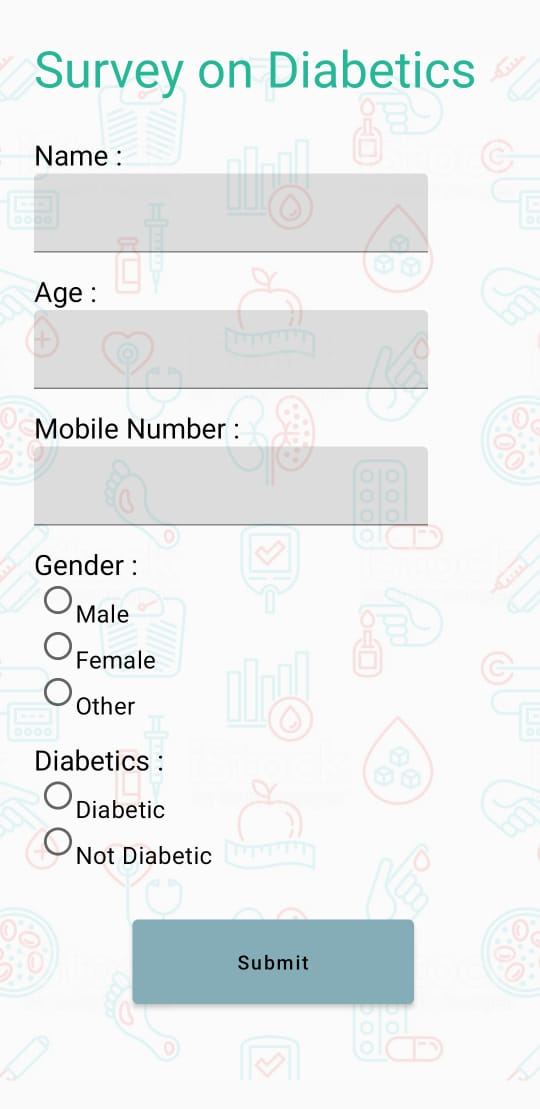
**LOGIN PAGE:**



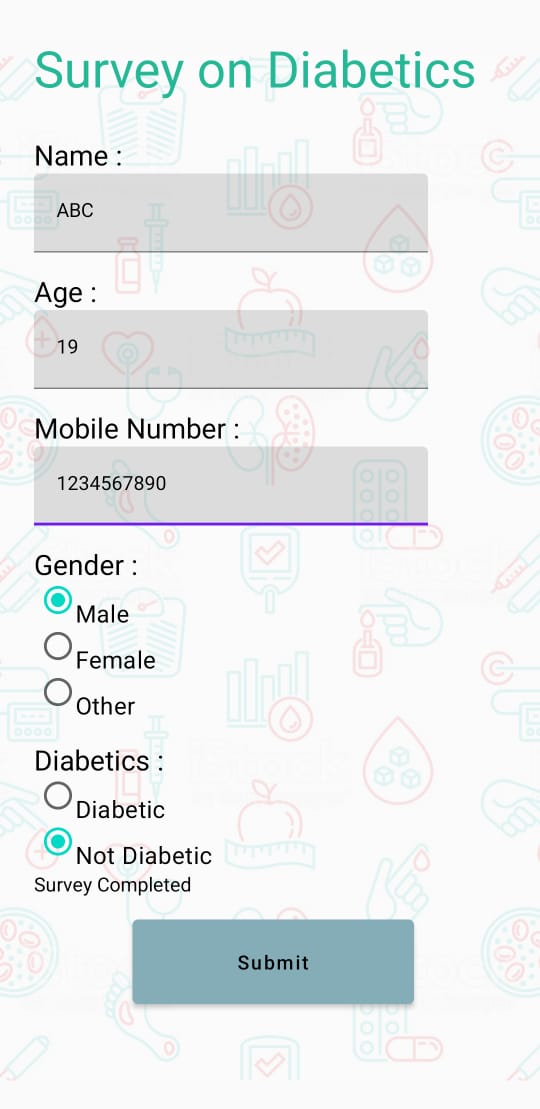
**REGISTER PAGE:**

****

**BEFORE SUBMIT:**

****

**AFTER SUBMIT:**

****

**6. Key Scenarios Addressed in Implementation**

* **User Registration and Login**: Ensure correct input and authentication for new users.
* **Real-Time Input Validation**: Prevent invalid data entries and guide user corrections.
* **Admin Data Access**: Showcase CRUD operations and data retrieval functionalities from the admin side.

**7. Conclusion**

**Summary of Achievements**  
The project demonstrates a complete workflow from user input handling to validation and database integration using Jetpack Compose. The use of declarative UI principles, efficient state management, and robust testing establishes a strong foundation for future Android development endeavors.