DOCUMENTATION OF

COMPOSE INPUT:

A DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE

Done By

Aathesh.v

Balaji.S

Muthu Aravind.B

Sandhiya Devi.B

Thangasri.P

``

INTRODUCTION

Overview:

1. **Text Input**: Allows users to input text data through a variety of input methods, such as voice, keyboard, or stylus. The input can be restricted to specific characters, such as numbers or letters, and can be formatted according to certain rules.

2. **Text Validation**: Ensures that the text data input by the user is valid, based on predefined rules. This can include checking for null or empty values, validating input against a database, or verifying that the input matches a specific format or pattern.

3. **UI components**: Displays the user interface components that allow users to input and validate text data. These components may include text fields, buttons, and other UI elements that help guide the user through the input and validation process.

4. **Error messages**: Displays error messages when the user enters invalid data or when validation fails. These messages can be displayed in real-time as the user types or can be displayed when the user submits the form

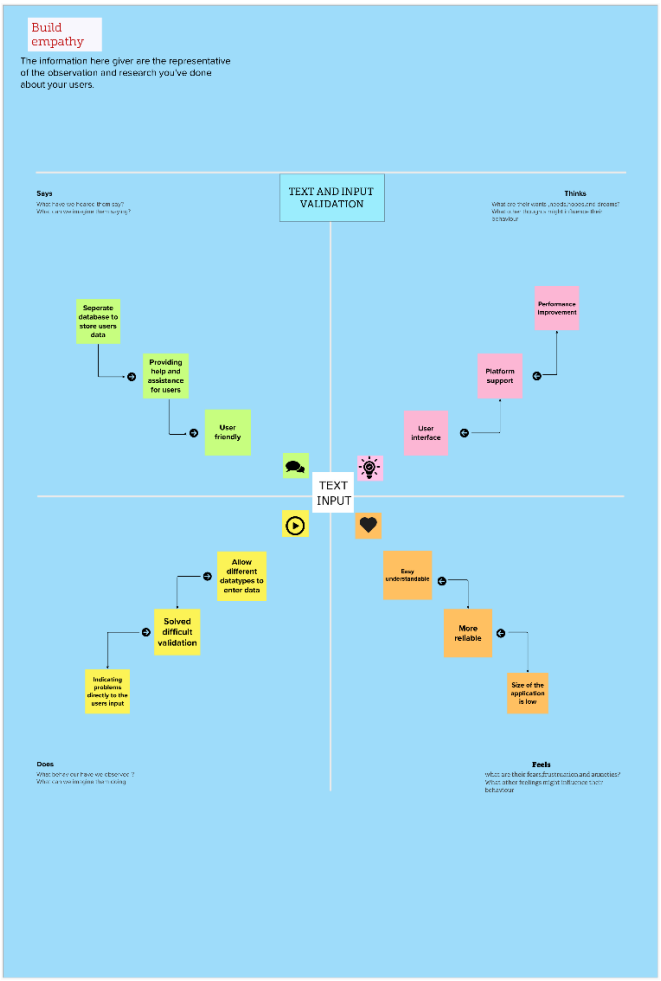
``

Purpose:

The purpose of demonstrating how to create a text input field with validation using Android Compose is to show developers how to implement similar features in their own Android applications. Text input fields are a common component in many mobile apps, and adding validation rules can help to ensure data accuracy and prevent errors. This tutorial can help developers learn how to use Android Compose to create user interfaces and handle user input in a more efficient and streamlined way compared to traditional Android views.

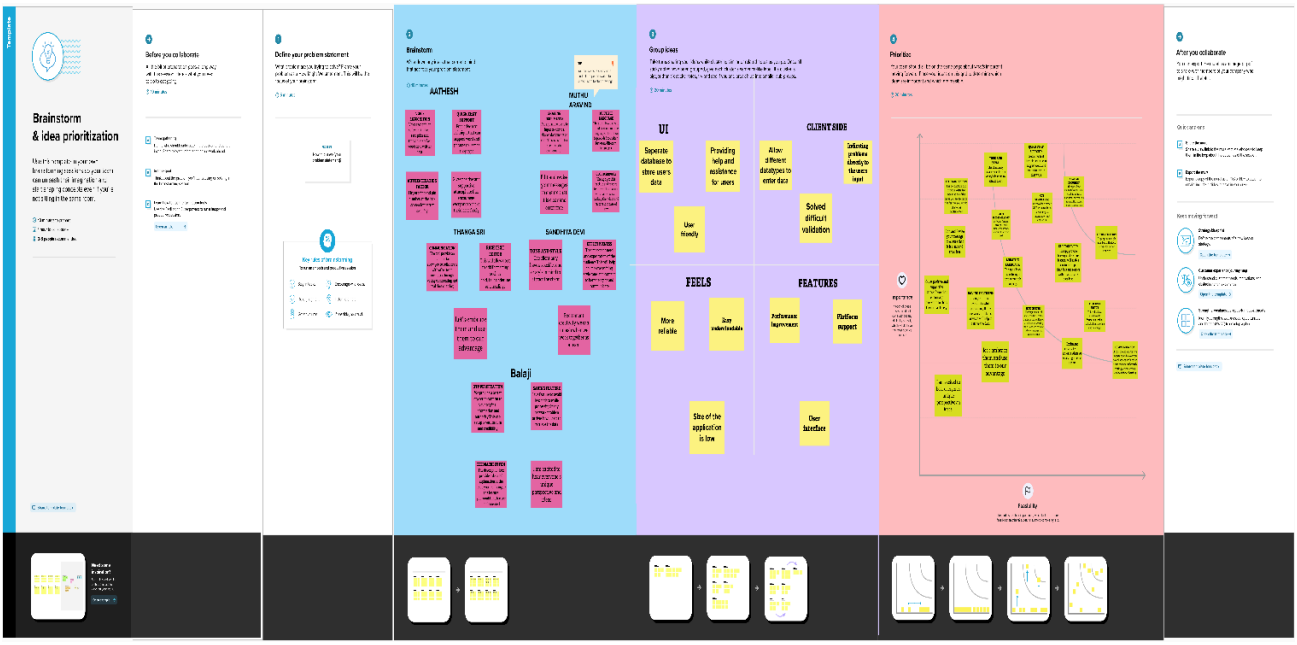
PROBLEM DEFINITIONS AND DESIGN THINKING

Empathy map



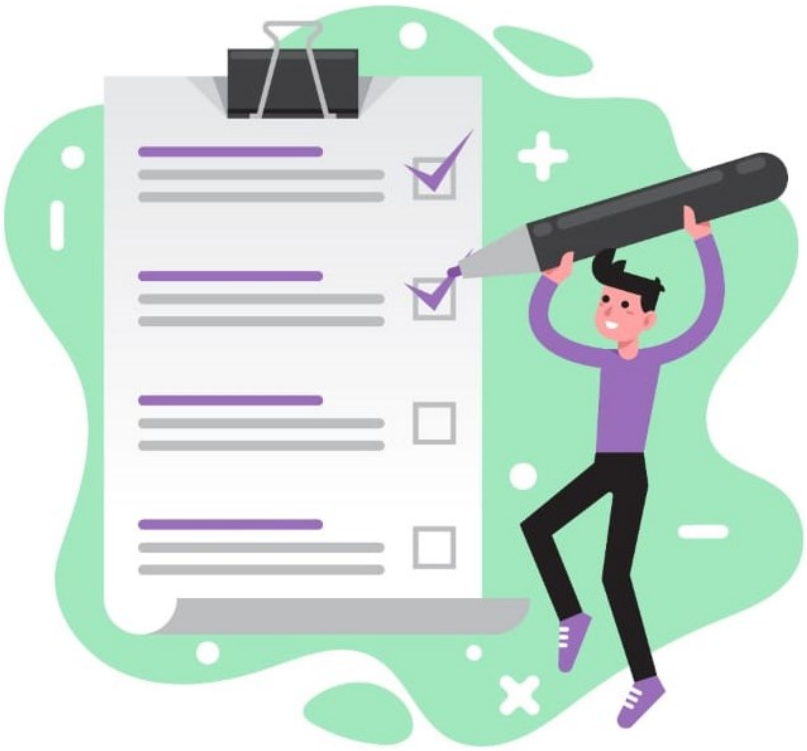
``

BRAINSTORMING MAP



``

Result:



Register

Username

Email

Password

Register

Have an account? Log in

``

Login

Username

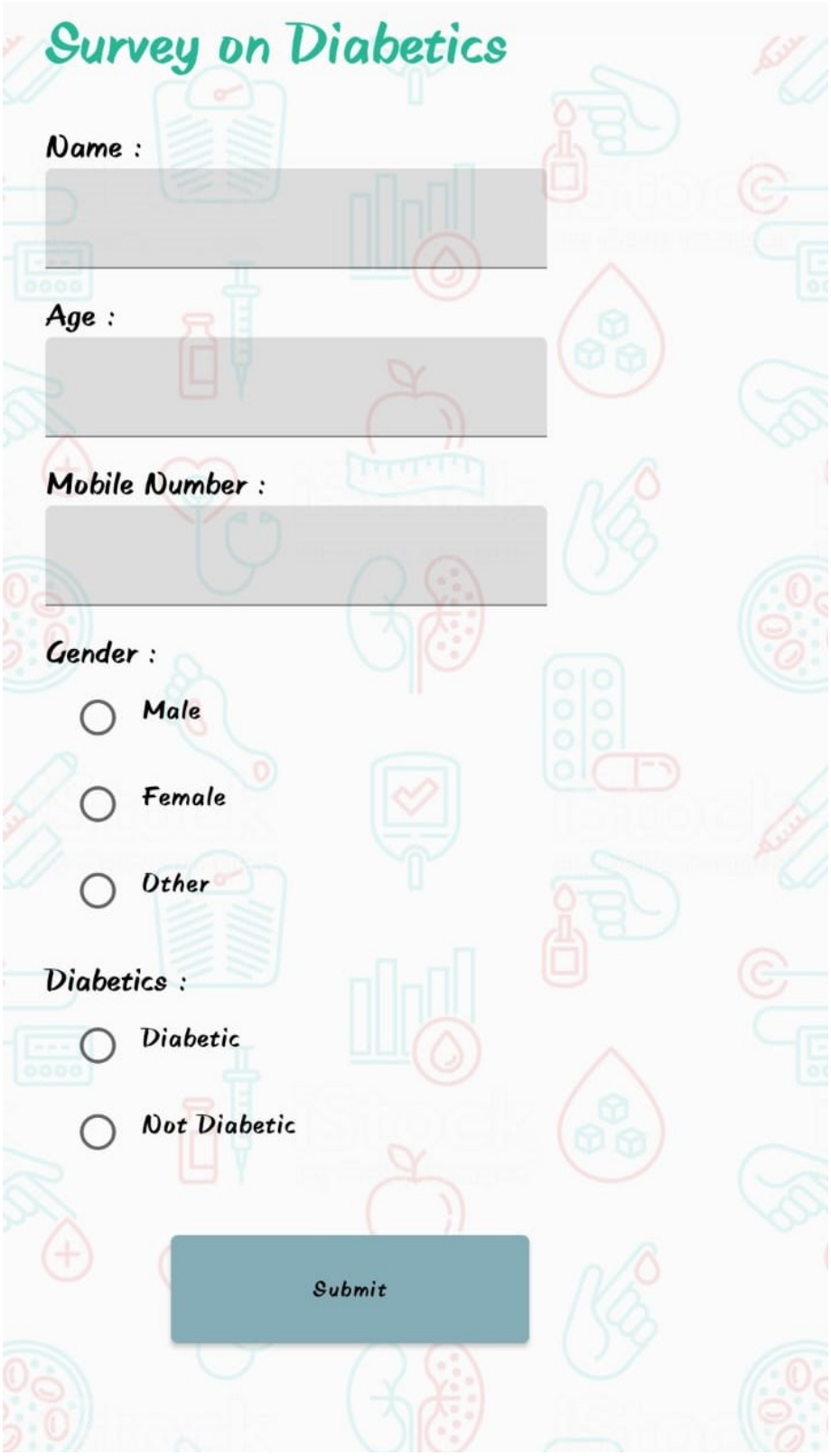
Password

Login

Register

Forget password?

``


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

``

ADVANTAGES & DISADVANTAGES

Advantages

1. **Easy to use**: Android Compose makes it easy to create and customize UI components for text input and validation. It provides a simplified, declarative approach that allows you to quickly build and iterate on your designs.

2. **Consistency**: With Android Compose, you can ensure consistency across your app's UI by using a single set of UI components that are easy to reuse and maintain.

3. **Immediate feedback**: Android Compose provides immediate feedback to users when they enter invalid data. This helps prevent errors and reduces frustration for users.

4. **Flexibility**: Android Compose provides a high degree of flexibility when it comes to text input and validation. You can easily customize the UI components to meet your specific needs, and you can choose from a variety of input types, including numbers, dates, and text.

5. **Increased productivity**: Android Compose provides a streamlined development process that can help developers be more productive. This allows you to spend more time creating features and less time dealing with UI issues.

``

Disadvantages:

One potential disadvantage of demonstrating text input and validation with Android Compose is that it is a relatively new technology, which means that there may be a learning curve for developers who are not familiar with it. This could potentially slow down development time, especially for teams that are not already proficient with Compose.

Additionally, as a new technology, there may be some bugs or compatibility issues that have not yet been discovered or resolved. These issues could potentially impact the functionality of text input and validation in your app.

Lastly, another potential disadvantage is that Android Compose is not yet fully matured, and the API may change over time. This means that there may be additional work needed to maintain the app and keep up with changes to the API and framework.

APPLICATIONS

1. In a messaging app, text input can be used to compose and send messages. With validation, the user can be prompted if they try to send an empty message or exceed the character limit.

``

2. In a food delivery app, text input can be used for the user to input their address or special dietary requirements. With validation, the user can be alerted if their address is invalid, ensuring that the order is delivered to the correct location.

3. In a healthcare app, text input can be used to collect medical information from the patient. With validation, the app can ensure that the user enters the right format for health data, such as specific range of blood sugar levels or BMI.

CONCLUSION

In conclusion, demonstrating how to create a text input field with validation using Android Compose is an essential aspect of developing Android applications. By following the steps outlined in such a demonstration, developers can implement similar functionality within their own apps. Using Android Compose to create user interfaces and handle user input can provide numerous advantages, such as simplifying the UI code, improving performance, and enhancing the overall user experience. Furthermore, text input validation helps to ensure data accuracy and prevents errors, making the app more reliable and user-friendly.

``

FUTURE SCOPE

1. The future scope of a demonstration of text input and validation with Android Compose is quite promising. Android Compose is a new UI toolkit that provides developers with many powerful features for building complex and responsive user interfaces.

2. One of the key features of Android Compose is its support for declarative UI programming. This means that developers can write code that describes the UI they want to build, rather than writing imperative code that describes how to build it.

3. In the case of text input and validation, this means that developers can write code that describes what a text input field should look like, how it should behave, and how it should be validated. This can be done in a simpler, more intuitive way than with traditional Android UI development.

4. As Android Compose continues to evolve and gain in popularity, it is likely that more and more developers will adopt it for their UI needs. This will provide a rich ecosystem of tools and resources for building robust and efficient UIs, including text input and validation components.

5. So, the future scope of a demonstration of text input and validation with Android Compose is bright, and there are many opportunities for developers to create innovative and compelling user interfaces with this powerful toolkit.

``

APPENDIX

1. **Sample code snippets**: Include code examples that demonstrate how to implement text input and validation features with Android Compose. This can help other developers to learn how to use Android Compose effectively.

2. **User guide**: Provide a user guide that explains how to use the text input and validation features in an Android Compose app. This can help users to understand how to input and validate text data in the app.

3. **Screenshots**: Include screenshots of the app in action to show the text input and validation features. This can help users to understand how the app works and what they can expect as they input and validate text.

4. **Resources**: Include links to relevant resources that developers can use to learn more about Android Compose and how to build effective UIs. This can include documentation, tutorials, and other helpful tools and resources.

``

Source code:

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
                            )
                        )
                        //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)
}
```

``

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)
    )




    // 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
                )
            }
        }
    }
}
```

``

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.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
                )
            }
        }
    }
}
```

``

THANK YOU

``