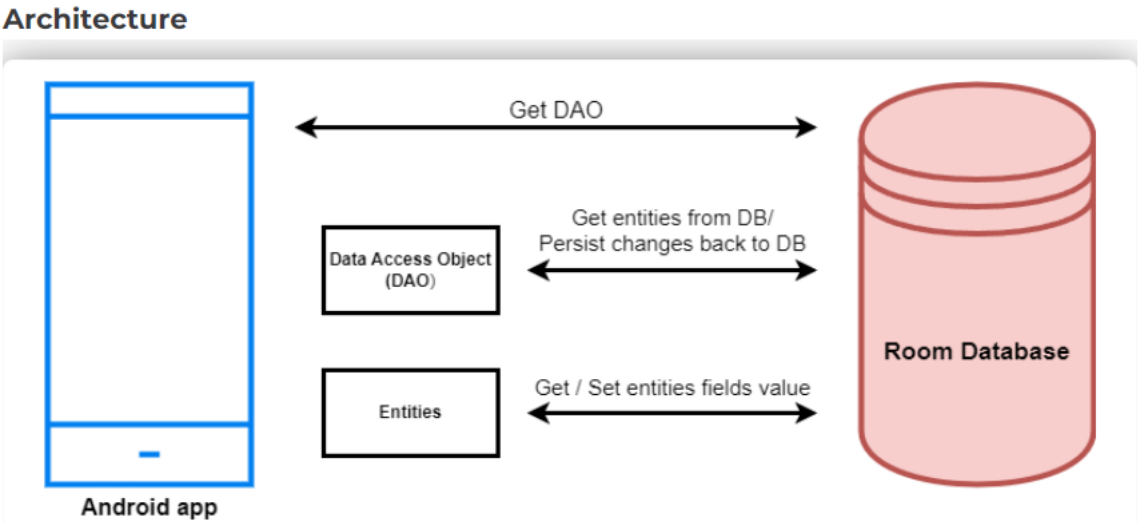
1-INTRODUCTION

1.1 OVERVIEW:

Compose Input: A Demonstration Of Text Input And Validation With Android Compose

Project Description:

The app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a survey app. The app allows the user to answer a series of questions. It showcases some of the key features of the Compose UI toolkit, data management, and user interactions.



Learning Outcomes:

By end of this project:

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.

Project Workflow:

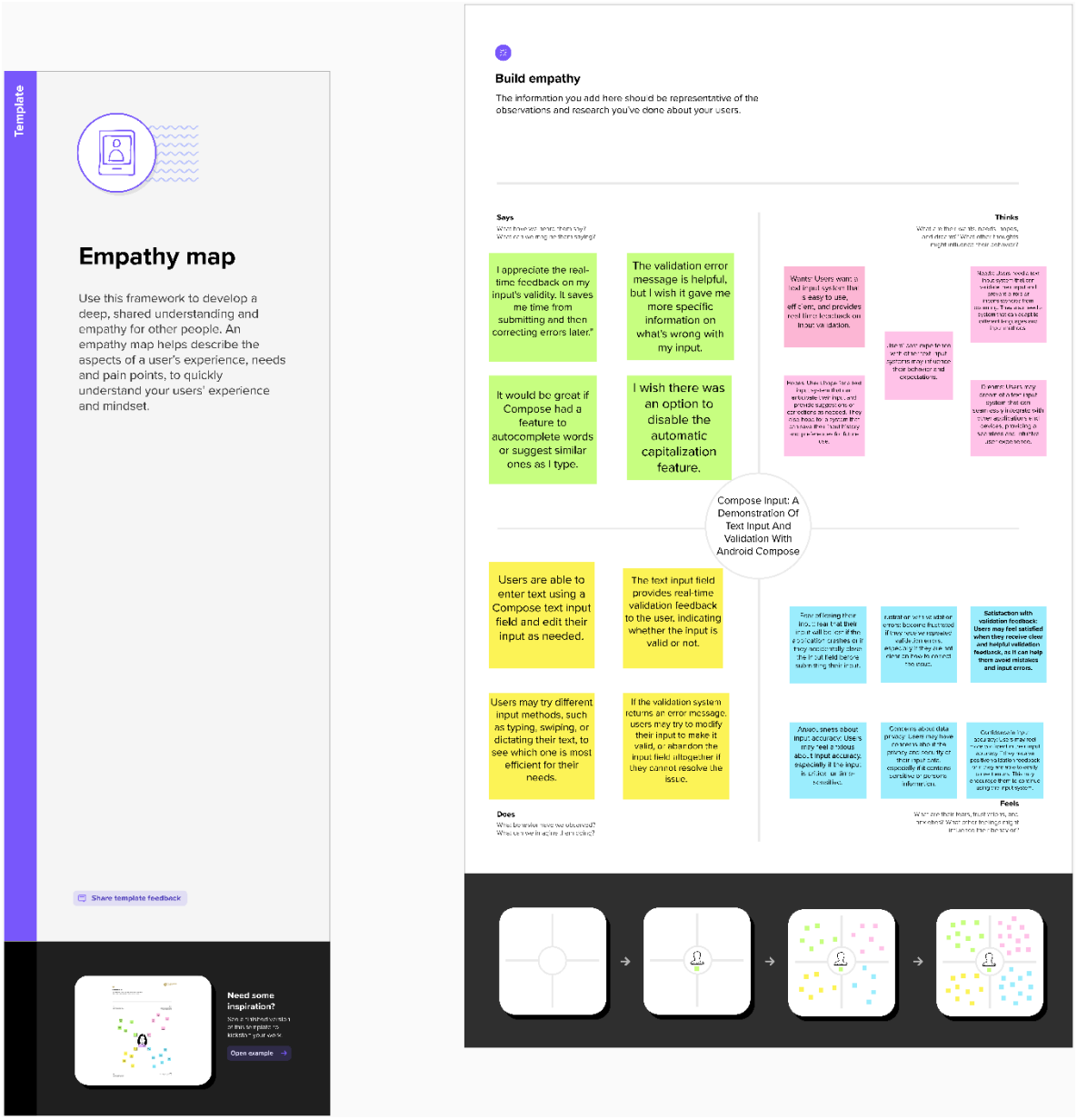
- Users register into the application.
- After registration user logins into the application.
- User enters into the main page
- From Admin Side he can login to the app and can view all the data.

1.2 PURPOSE:

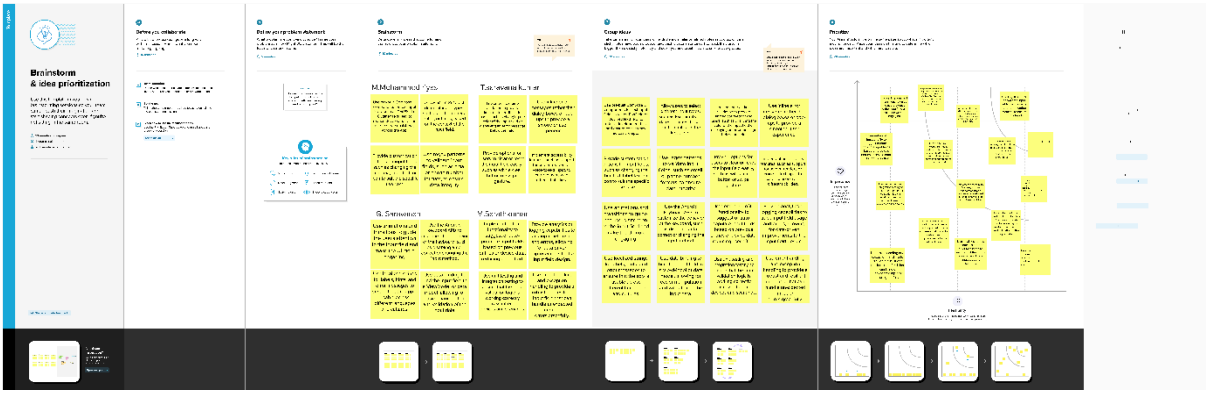
Collect data from a specific target audience for a particular purpose. Surveys are commonly used by businesses, organizations, and researchers to gather information about their customers, users, or target audience.

2-PROBLEM DEFINATION AND DESIGN THINKING:

2.1 EMPATHY MAP:



2.2 IDEATION AND BRAINSTORM:



3-RESULT:

Login

Username

Password

Login

Register Forget password?

Register

Username

Email

Password

Register

Have an account? Log in

Survey on Diabetics

Name :

Age :

Mobile Number :

Gender :

- Male
- Female
- Other

Diabetics :

- Diabetic
- Not Diabetic

Submit

Survey on Diabetics

Name : riyas

Age : 20

Mobile Number : 12345678

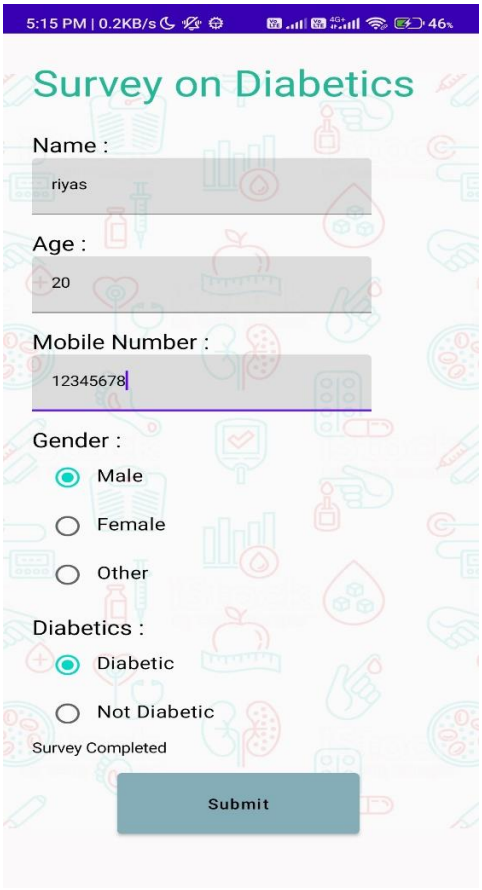
Gender :

- (•) Male
- ( ) Female
- ( ) Other

Diabetics :

- (•) Diabetic
- ( ) Not Diabetic

Submit



4-ADVANTAGE AND DISADVANTAGE:

Advantages of demonstrating text input and validation with Android Compose:

- Android Compose is a modern toolkit for building user interfaces that offers a declarative approach to UI design, making it easier to create and maintain UI elements.
- Android Compose provides easy-to-use functions and utilities for handling user input and validation, which can save developers a lot of time and effort.
- Demonstrating text input and validation with Android Compose can help developers better understand the capabilities of the toolkit and how to use it effectively.

Disadvantages of demonstrating text input and validation with Android Compose:

- Android Compose is a relatively new technology, and its documentation and community support may not be as comprehensive as other established UI frameworks.
- Since Android Compose is still in beta, it may not be stable enough for production applications.
- Demonstrating text input and validation with Android Compose may not be as relevant for developers who are not working on Android apps or who are not interested in using this particular toolkit.

5-APPLICATION:

The application of a demonstration of text input and validation with Android Compose is to showcase how to implement user input and validation in an Android app using the latest UI toolkit provided by Google.

This demonstration can be particularly useful for Android developers who are new to Android Compose and want to learn how to create user interfaces using this modern toolkit. By providing a practical example of how to implement text input and validation, developers can gain a better understanding of the capabilities of Android Compose and how to use it effectively in their own projects.

Additionally, this demonstration can also be used as a reference for developers who are already familiar with Android Compose but want to refresh their knowledge on how to handle user input and validation in their apps. By showcasing best practices and tips for implementing text input and validation, developers can ensure that their apps are user-friendly and provide a smooth user experience.

Overall, a demonstration of text input and validation with Android Compose is a valuable resource for developers who want to learn how to create modern and engaging user interfaces for their Android apps.

6-CONCLUTION:

In conclusion, a demonstration of text input and validation is a valuable tool for developers who want to learn how to create user-friendly interfaces and ensure that users can input valid data. Whether in web development or mobile app development, text input and validation are critical components of the user experience, and developers must know how to implement them effectively.

In the context of Android app development, using Android Compose to demonstrate text input and validation can be particularly beneficial. Android Compose is a modern UI toolkit that offers a declarative approach to UI design and

provides easy-to-use functions for handling user input and validation. By demonstrating how to use Android Compose for text input and validation, developers can better understand the capabilities of this toolkit and how to use it effectively in their projects.

Ultimately, a demonstration of text input and validation can help developers ensure that their apps are user-friendly, functional, and meet the needs of their users. By following best practices and implementing effective text input and validation, developers can create interfaces that are intuitive and engaging, leading to a better overall user experience.

7-FUTURE SCOPE:

The future scope of a demonstration of text input and validation with Android Compose is significant, given the growth and popularity of this modern UI toolkit for Android app development.

APPENDIX:

A.SOURCE CODE

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
                )
            }
        }
    }
}
```

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") },

            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)
}
```