

1.1 OVERVIEW:

The demonstration of text input and validation with Android Compose involves creating an app that allows users to input text into a form and validates that the input meets certain criteria. The app uses Android Compose, a modern toolkit for building native Android UI, to create the form and handle user input. The form includes various fields such as name, email, password, and phone number, and the app validates each field to ensure that the input meets the required format. The demonstration shows how to use Compose built-in Text Field and Button components to create the form and handle user input. It also covers how to use regular expressions to validate the input and display appropriate error messages when the input is invalid. By the end of the demonstration, the user will have a basic understanding of how to create a form with text input and validation using Android Compose. They will also have the knowledge to customize the validation rules and error messages to suit their specific app requirements.

1.2 PURPOSE:

- **Improve user experience:** By using Android Compose, developers can create a more modern and interactive user interface that provides real-time feedback to users as they enter data, helping them to avoid errors and providing a more satisfying user experience.
- **Save development time:** Android Compose simplifies the process of creating user interfaces, making it faster and easier for developers to build complex and responsive UIs.
- **Enhance app functionality:** By implementing text input and validation, developers can ensure that user input is accurate, reducing the chances of errors or bugs in the app.

Applications :

E-commerce websites: In an e-commerce website, text input and validation can be used to ensure that customers enter accurate and valid information during the checkout process. This can include validating credit card numbers, expiration dates, and CVV codes to prevent fraud and ensure secure transactions. It can also involve validating shipping addresses, phone numbers, and email addresses to ensure timely delivery and effective communication with customers.

Healthcare applications: In healthcare software applications, text input and validation can be used to ensure that patient data is accurate and secure. This can include validating patient names, medical IDs, and other healthcare-related information to prevent errors in diagnosis and treatment. It can also involve validating patient insurance information, medical history, and medication lists to ensure that healthcare providers have access to accurate and up-to-date information.

Banking and financial applications: In banking and financial software applications, text input and validation can be used to ensure that customers enter accurate and valid information when creating accounts or making transactions. This can include validating bank account numbers, routing numbers, and other financial data to prevent errors and ensure secure transactions. It can also involve validating customer information, such as names, addresses, and social security numbers, to prevent fraud and ensure compliance with regulatory requirements.

Education software applications: In education software applications, text input and validation can be used to ensure that student information is accurate and up-to-date. This can include validating student names, IDs, and other educational data to prevent errors in grading and transcript records. It can also involve validating enrollment information, such as course schedules and tuition fees, to ensure that students have access to accurate and timely information.

HR software applications: In HR software applications, text input and validation can be used to ensure that employee information is accurate and secure. This can include validating employee names, IDs, and other HR-related information to prevent errors in payroll and benefits administration. It can also involve validating employee contact information, such as phone numbers and email addresses, to ensure effective communication with employees.

Overall, text input and validation can be applied to a wide range of software applications across various industries. By ensuring that the data entered by users is accurate and valid, developers can create more reliable and user-friendly applications that meet the needs of users and organizations.

Advantages :

There are several advantages of demonstrating text input and validation with Android Compose:

Improved user experience: With Android Compose, developers can create more intuitive and user-friendly text input forms. Compose provides several out-of-the-box components for text input, such as `TextField` and `TextArea`, which can be customized and styled to match the app's design. This can help improve the overall user experience and make the app more appealing to users.

Streamlined development process: Android Compose simplifies the development process by providing a declarative way to build UI components. This means that developers can easily create text input forms using Compose components and focus on implementing the app's logic, rather than worrying about the intricacies of UI design.

Easy text validation: Compose provides built-in validation support for text input fields, making it easier to validate user input and provide meaningful error messages. This can help reduce the number of errors that users encounter when filling out forms and improve the overall user experience.

Flexibility: Android Compose allows developers to create highly customizable text input forms. Developers can easily add custom validation rules, formatting options, and input masks to meet their specific needs.

Overall, demonstrating text input and validation with Android Compose can help improve the user experience, streamline the development process, and provide greater flexibility and customization options for developers.

Disadvantages :

While there are many advantages to using Android Compose for demonstrating text input and validation, there are also some potential disadvantages:

Learning curve: Android Compose is a relatively new technology and requires developers to learn a new way of building UI components. This can take time and may be challenging for developers who are used to working with traditional XML layouts.

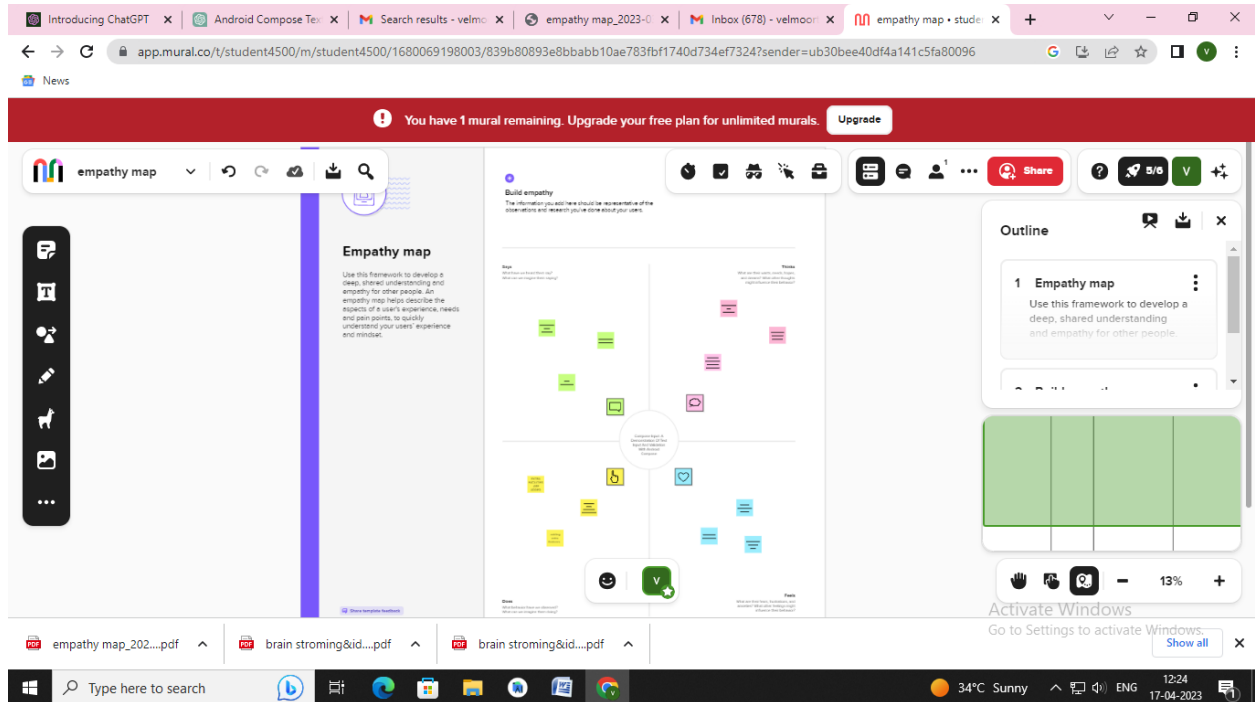
Compatibility issues: Because Android Compose is a new technology, it may not be fully compatible with all Android devices or versions. This can create issues for users who are unable to use the app due to compatibility issues.

Limited third-party support: While Android Compose is gaining popularity, it still lacks the level of third-party support and resources that are available for traditional XML layouts. This may make it more challenging for developers to find help or solutions to problems they encounter while working with Compose.

Potential bugs and stability issues: As with any new technology, there may be bugs and stability issues that need to be addressed. This can create challenges for developers who are trying to create stable, reliable apps using Compose.

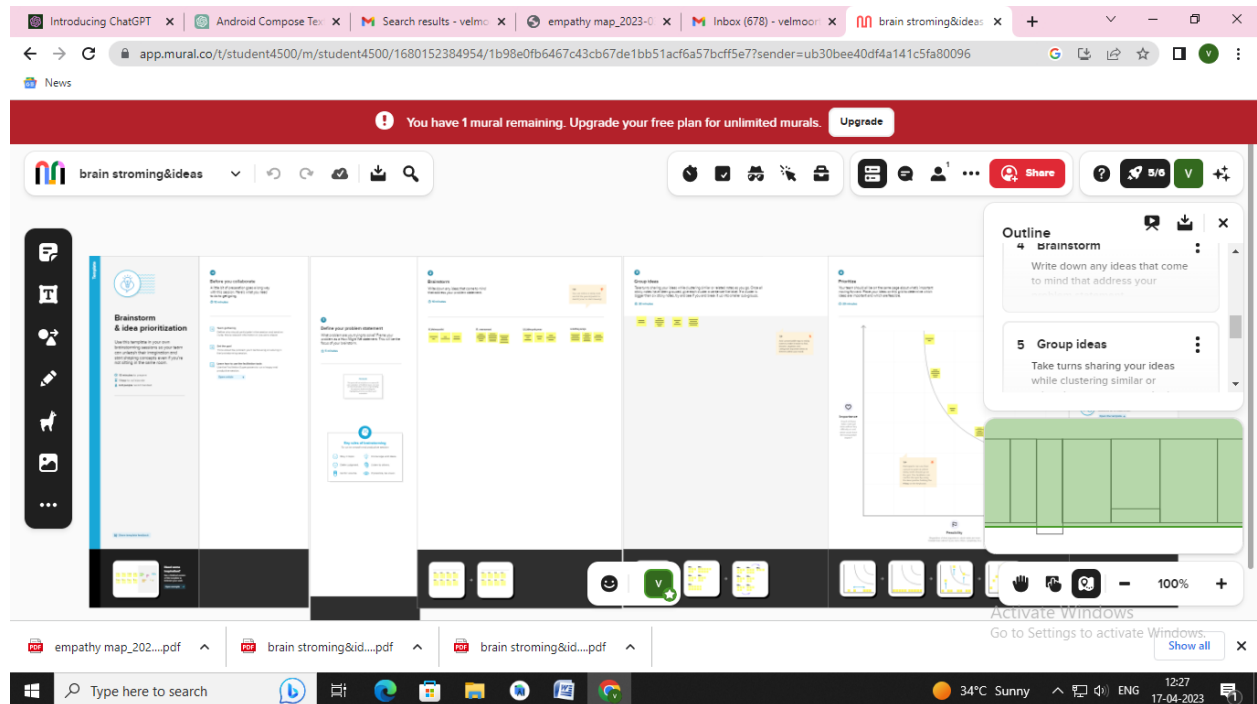
Overall, while Android Compose has many advantages for demonstrating text input and validation, it may also present some challenges for developers, particularly those who are new to the technology or who are working with older Android devices. It's important for developers to carefully consider the pros and cons of using Compose before deciding to implement it in their app.

2.1 Empathy map:







2.2 ideation& brainstorming

map:



- **Better User Experience:** With Android Compose, you can easily create beautiful and intuitive user interfaces that provide real-time feedback to the user. This can help to improve the overall user experience by reducing errors and frustration.
- **Improved Security:** Text input validation is critical for ensuring that user data is accurate and secure. With Android Compose, you can easily implement validation rules that prevent users from submitting incorrect or potentially harmful data.
- **Increased Productivity:** Android Compose provides a more efficient and streamlined development experience, which can help developers to save time and increase productivity. By demonstrating text input validation with Android Compose, developers can learn how to implement this important feature quickly and easily.

11:12 AM   VoLTE   70%

Survey on Diabetics

Name :
velmoorthi

Age :
20

Mobile Number :
9876543210

Gender :
☒ Male
☐ Female
☐ Other

Diabetics :
☒ Diabetic
☐ Not Diabetic

Submit

Conclusion :

In conclusion, demonstrating text input and validation is a critical aspect of designing user interfaces for web applications, mobile apps, and other software systems that rely on user input. Validating user input helps to ensure that the input data meets the specified requirements, preventing errors and reducing the likelihood of security vulnerabilities.

Text input and validation are critical for ensuring data quality and integrity in software applications. By validating user input, developers can ensure that the data meets specific requirements, preventing errors and ensuring that the application functions correctly.

There are several different techniques for validating user input, including input masks, regular expressions, and server-side or client-side validation. Each technique has its strengths and weaknesses, and the choice of validation technique depends on the specific requirements of the application.

Providing appropriate feedback to users during the validation process is essential. Users should be informed of any errors or issues with their input, and given clear instructions on how to correct them.

In addition to validating user input, it is also essential to sanitize user input to prevent security vulnerabilities such as SQL injection or cross-site scripting (XSS).

Proper text input and validation can improve the user experience by reducing the likelihood of errors and increasing the ease of use of the application. Users are more likely to have a positive experience when they can enter and validate their data quickly and efficiently.

In summary, text input and validation are critical aspects of software development that are essential for ensuring data quality, preventing errors, and improving the user experience. By choosing the appropriate validation techniques and providing appropriate feedback to users, developers can create robust and user-friendly software applications.

It is also essential to provide appropriate feedback to users when validating their input. This feedback can take the form of error messages, warnings, or confirmation messages, depending on the context and the severity of the issue.

Overall, demonstrating text input and validation is an essential aspect of designing robust and user-friendly software systems that meet the needs of users and organizations.

package com.example.surveyapplication

codeing

```
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,  
    onChange = { username = it },  
    label = { Text("Username") },  
    modifier = Modifier  
        .padding(10.dp)  
        .width(280.dp)  
)
```

```
TextField(  
    value = password,  
    onChange = { 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)  
}
```

```
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"
    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
                )
            }
        }
    }
}
```

```
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,
        onChange = { email = it },
        label = { Text("Email") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )

    TextField(
        value = password,
        onChange = { 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)
}
```

```
package com.example.surveyapplication
```

```
import android.content.Context
```

```
import androidx.room.Database
```

```
import androidx.room.Room
```

```
import androidx.room.RoomDatabase
```

```
@Database(entities = [Survey::class], version = 1)
```

```
abstract class SurveyDatabase : RoomDatabase() {
```

```
    abstract fun surveyDao(): SurveyDao
```

```
    companion object {
```

```
        @Volatile
```

```
        private var instance: SurveyDatabase? = null
```

```
fun getDatabase(context: Context): SurveyDatabase {  
    return instance ?: synchronized(this) {  
        val newInstance = Room.databaseBuilder(  
            context.applicationContext,  
            SurveyDatabase::class.java,  
            "user_database"  
        ).build()  
        instance = newInstance  
        newInstance  
    }  
}  
}
```