

# **COMPOSE INPUT: A DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE**

## **ABSTRACT:**

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. The information maintained about products, services and customers is a most valuable organisational asset. Therefore, it is important for successful electronic business to have high quality websites. A website must however, do more than just look attractive it must be usable and present useful, usable information. Usability essentially means that the website is intuitive and allows visitors to find what they are looking for quickly and without effort. This means careful consideration of the structure of information and navigational design. According to the Open Web Applications Security Project, invalidated input is one of the top ten critical web-application security vulnerabilities. We empirically tested 21 Irish corporate websites. The findings suggested that one of the biggest problems is that many failed to use mechanisms to validate even the basic user data input at the source of collection which could potentially result in a database full of useless information.

## **INTRODUCTION**

Apps often require users to enter information into a text field. For example, you might require users to log in with an email address and password combination.

To make apps secure and easy to use, check whether the information the user has provided is valid. If the user has correctly filled out the form, process the information. If the user submits incorrect information, display a friendly error message letting them know what went wrong.

In this example, learn how to add validation to a form that has a single text field using the following steps:

1. Create a Form with a GlobalKey.
2. Add a TextFormField with validation logic.
3. Create a button to validate and submit the form.

The World Wide Web (WWW) is the largest available distributed dynamic repository of information, and has undergone massive and rapid growth since its inception. There are over 2,060,000 users in Ireland alone. Over the last seven years (2000 - 2007), Internet usage in Ireland has grown by 162.8%; in United Kingdom by 144.2%; in Europe by 221.5% and Worldwide by 244.7%. Based on these facts, the Internet has assumed a central role in many

aspects of our lives and therefore creates a greater need for businesses to design better websites in order to stay competitive and increase revenue. Interactivity is essential to engage visitors and lead them to the desired action and customers are more likely to return to a website that has useful interactivity.

The website's homepage should be a marketing tool designed as a 'billboard' for the organization. The design is critical in capturing the viewer's attention and interest and should represent the company in a meaningful and positive light. Therefore, there are many web design concerns for commercial organizations when designing their website. The most basic are as follows: content that should be included, selecting relevant and essential information, designing a secure, usable, user friendly web interface that is relatively easy to navigate, and ensuring the site is easy to find using any of the major search engines. In the drive to make the website look appealing from a visual perspective other factors are often ignored, such as validation and security, which leads to poor user experience and data quality problems.

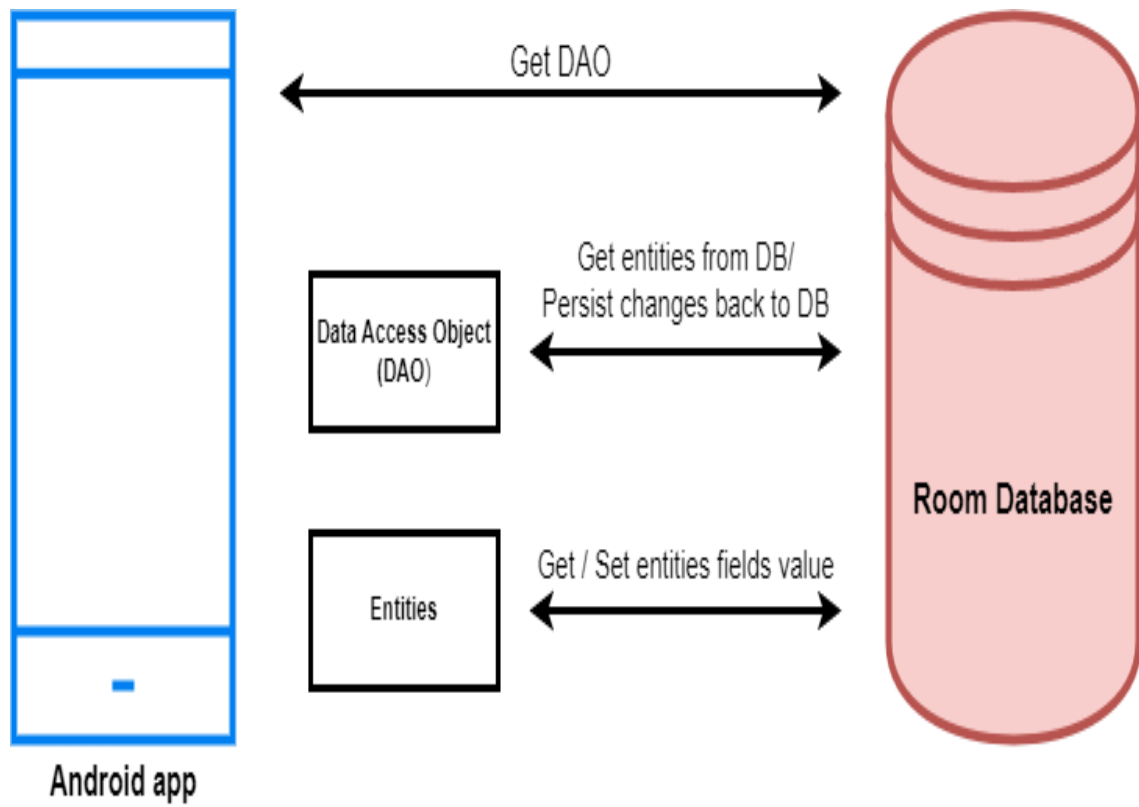
Data in the real world is constantly changing therefore feedback is necessary in order to ensure that quality is maintained. Data is deemed of high quality if it 'correctly represents the real-world construct to which it refers so that products or decisions can be made'. One can probably find as many definitions for quality on the web as there are papers on quality. There are however, a number of theoretical frameworks for understanding data quality.

**Project Workflow:**

- Users register into the application.
- After registration , user logs into the application.
- User enters into the main page
- The app allows users to choose , play and pause podcasts.

## SYSTEM MODEL

### Architecture:



CODE:

```
package  
com.example.podcastplay  
er
```

```
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.BorderStroke  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.background  
import androidx.compose.foundation.layout.*  
import androidx.compose.material.*  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.Email  
import androidx.compose.material.icons.filled.Lock  
import androidx.compose.material.icons.filled.Person  
import androidx.compose.runtime.*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.draw.alpha  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.layout.ContentScale  
import androidx.compose.ui.res.painterResource  
import androidx.compose.ui.text.font.FontWeight  
import  
androidx.compose.ui.text.input.PasswordVisualTransformati  
on  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp
```

```

import androidx.compose.ui.unit.em
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import
com.example.podcastplayer.ui.theme.PodcastPlayerTheme

class RegistrationActivity : ComponentActivity() { private
lateinit var databaseHelper: UserDatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            PodcastPlayerTheme {
                // A surface container using the 'background' color
from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    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
        .background(Color.Black)
        .fillMaxHeight()
        .fillMaxWidth(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
)

{
    Row {
        Text(
            text = "Sign Up",
            color = Color(0xFF6a3ef9),
            fontWeight = FontWeight.Bold,
            fontSize = 24.sp, style =
MaterialTheme.typography.h1,
            letterSpacing = 0.1.em
        )
    }

    Image(
        painter = painterResource(id =
R.drawable.podcast_signup),
        contentDescription = ""
    )

    TextField(
        value = username,
        onValueChange = { username = it },
        leadingIcon = {
            Icon(

```

```

        imageVector = Icons.Default.Person,
        contentDescription = "personIcon",
        tint = Color(0xFF6a3ef9)
    )
},
placeholder = {
    Text(
        text = "username",
        color = Color.White
    )
},
colors = TextFieldDefaults.textFieldColors(
    backgroundColor = Color.Transparent
)
)

```

```

Spacer(modifier = Modifier.height(8.dp))

```

```

TextField(
    value = password,
    onChange = { password = it },
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Lock,
            contentDescription = "lockIcon",
            tint = Color(0xFF6a3ef9)
        )
    },
    placeholder = { Text(text = "password", color =
Color.White) },
    visualTransformation
                                =
    PasswordVisualTransformation(),

```

```
        colors =
TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
    )
```

```
Spacer(modifier = Modifier.height(16.dp))
```

```
TextField(
    value = email,
    onChange = { email = it },
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Email,
            contentDescription = "emailIcon",
            tint = Color(0xFF6a3ef9)
        )
    },
    placeholder = { Text(text = "email", color =
Color.White) },
    colors =
TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
)
```

```
Spacer(modifier = Modifier.height(8.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"
            }
        },
        border = BorderStroke(1.dp, Color(0xFF6a3ef9)),
        colors =
ButtonDefaults.buttonColors(backgroundColor
Color.Black),
        modifier = Modifier.padding(top = 16.dp)
    ) {

```

```
Text(text = "Register",
      fontWeight = FontWeight.Bold,
      color = Color(0xFF6a3ef9)
    )
}
```

```
Row(
  modifier = Modifier.padding(30.dp),
  verticalAlignment = Alignment.CenterVertically,
  horizontalArrangement = Arrangement.Center
) {
  Text(text = "Have an account?", color = Color.White)
```

```
TextButton(onClick = {
  context.startActivity(
    Intent(
      context,
      LoginActivity::class.java
    )
  )
}) {
  Text(text = "Log in",
        fontWeight = FontWeight.Bold,
        style = MaterialTheme.typography.subtitle1,
        color = Color(0xFF6a3ef9)
    )
}

}

}
```

```
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

```
package
com.example.podcastplay
er
```

```
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.BorderStroke
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Person
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import
androidx.compose.ui.text.input.PasswordVisualTransformati
on
```

```

import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.em
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import
com.example.podcastplayer.ui.theme.PodcastPlayerTheme

```

```

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            PodcastPlayerTheme {
                // A surface container using the 'background' color
                from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    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("") }
```

```
Card(  
    elevation = 12.dp,  
    border = BorderStroke(1.dp, Color.Magenta),  
    shape = RoundedCornerShape(100.dp),  
    modifier = Modifier.padding(16.dp).fillMaxWidth()  
) {
```

```
    Column(  
        Modifier  
            .background(Color.Black)  
            .fillMaxHeight()  
            .fillMaxWidth()  
            .padding(bottom = 28.dp, start = 28.dp, end =  
28.dp),  
        horizontalAlignment =  
Alignment.CenterHorizontally,  
        verticalArrangement = Arrangement.Center  
    )
```

```
    {  
  
        Image(  
            painter =  
painterResource(R.drawable.podcast_login),  
            contentDescription = "",  
            Modifier.height(400.dp).fillMaxWidth()  
        )
```

```
        Text(  
            text = "LOGIN",
```

```
        color = Color(0xFF6a3ef9),
        fontWeight = FontWeight.Bold,
        fontSize = 26.sp,
        style = MaterialTheme.typography.h1,
        letterSpacing = 0.1.em
    )
```

```
Spacer(modifier = Modifier.height(10.dp))
```

```
TextField(
    value = username,
    onChange = { username = it },
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Person,
            contentDescription = "personIcon",
            tint = Color(0xFF6a3ef9)
        )
    },
    placeholder = {
        Text(
            text = "username",
            color = Color.White
        )
    },
    colors = TextFieldDefaults.textFieldColors(
        backgroundColor = Color.Transparent
    )
)
```

```
Spacer(modifier = Modifier.height(20.dp))
```

```

TextField(
    value = password,
    onChange = { password = it },
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Lock,
            contentDescription = "lockIcon",
            tint = Color(0xFF6a3ef9)
        )
    },
    placeholder = { Text(text = "password", color =
Color.White) },
    visualTransformation =
PasswordVisualTransformation(),
    colors =
TextFieldDefaults.textFieldColors(backgroundColor
Color.Transparent)
)
Spacer(modifier = Modifier.height(12.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()
        } else {
            error = "Invalid username or password"
        }
    } else {
        error = "Please fill all fields"
    }
},
border = BorderStroke(1.dp, Color(0xFF6a3ef9)),
colors = ButtonDefaults.buttonColors(backgroundColor = Color.Black),
modifier = Modifier.padding(top = 16.dp)
) {
    Text(text = "Log In", fontWeight = FontWeight.Bold, color = Color(0xFF6a3ef9))
}

```

```

Row(modifier = Modifier.fillMaxWidth()) {
    TextButton(onClick = {
        context.startActivity(
            Intent(

```



```
context,
RegistrationActivity::class.java
)))
{
    Text(
        text = "Sign up",
        color = Color.White
    )
}
```

```
Spacer(modifier = Modifier.width(80.dp))
```


```
TextButton(onClick = { /* Do something! */ })
{
    Text(
        text = "Forgot password ?",
        color = Color.White
    )
}
}
}
}
```

```
fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

OUTPUT:

Login Page :

---



*Login*

[Register](#)      [Forget password?](#)

---

Register Page :



*Register*

Username

Email

Password

Register

Have an account? [Log in](#)

Admin page:

## Survey Details

Name: Krishna  
Age: 37  
Mobile\_Number: 6897616678  
Gender: Male  
Diabetics: Diabetic

Name: Pavani  
Age: 23  
Mobile\_Number: 7167816818  
Gender: Female  
Diabetics: Not Diabetic

Name: Pavani  
Age: 23  
Mobile\_Number: 7167816818  
Gender: Female  
Diabetics: Not Diabetic

User Module:  
Login Page :



*Login*

Username

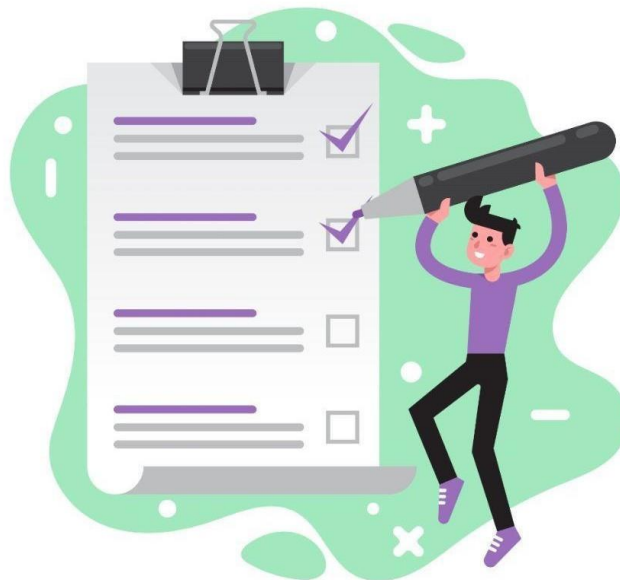
Password

Login

[Register](#)

[Forget password?](#)

Register Page :



*Register*

Username

Email

Password

Register

Have an account? [Log in](#)

Main Page :

## Survey on Diabetics

Name :

Age :

Mobile Number :

Gender :

☐ Male

☐ Female

☐ Other

Diabetics :

☐ Diabetic

☐ Not Diabetic

## **CONCLUSIONS**

Today's Internet user expects to experience personalized interaction with websites. If the company fails to deliver they run the risk of losing a potential customer forever. An important aspect of creating interactive web forms to collect information from users is to be able to check that the information entered is valid, therefore; information submitted through these forms should be extensively validated. Validation could be performed using client script where errors are detected when the form is submitted to the server and if any errors are found the submission of the form to the server is cancelled and all errors displayed to the user. This allows the user to correct their input before re-submitting the form to the server. We can not underestimate the importance of input validation which ensures that the application is robust against all forms of input data obtained from the user.