Compose Input: A Demonstration of Text Input and Validation with Android Compose

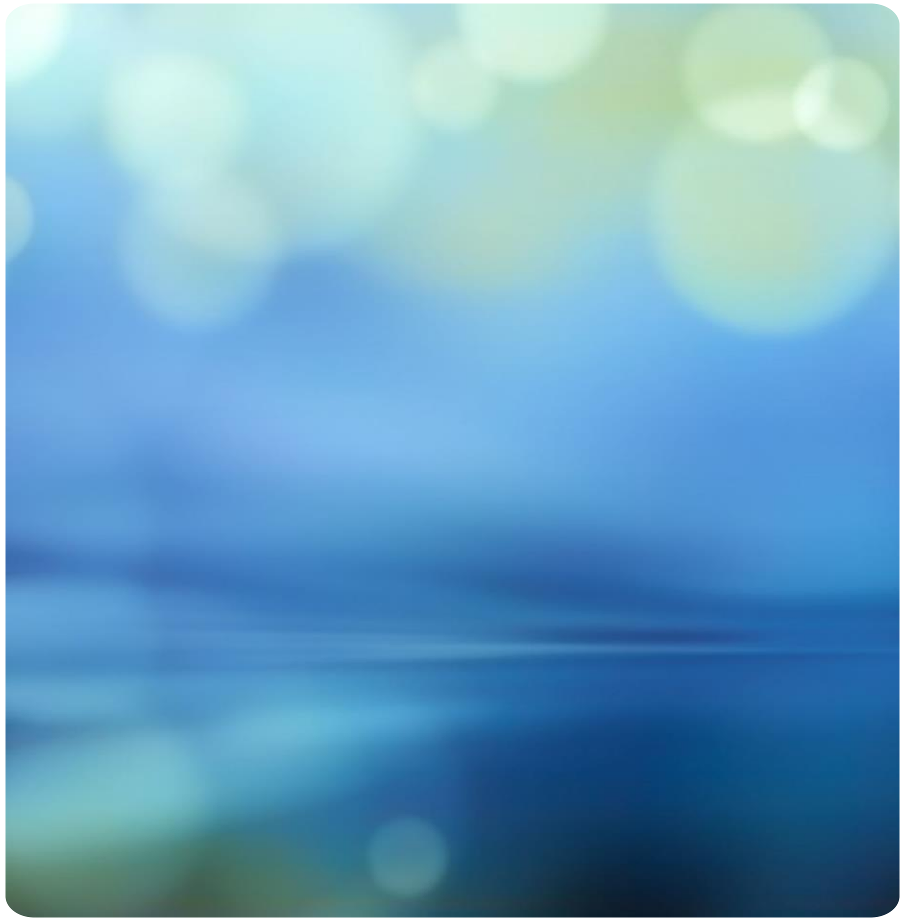
MOHAMAD ALI A (AUT810622CS19)
KEERTHANA S (AUT810623LCS41)

Project Description

- Compose Input is a sample project designed to showcase efficient text input handling and validation in Android applications using Jetpack Compose. This project focuses on demonstrating the capabilities of Android's modern UI toolkit, Jetpack Compose, in building intuitive, user-friendly forms with real-time validation feedback.

Application Description

- **Compose Input** is a sample Android application built with Jetpack Compose to demonstrate efficient text input handling, form validation, and dynamic user feedback. This app is designed as a learning tool for Android developers exploring the Compose UI toolkit, specifically for handling user inputs in forms and validating data in real-time.



Key Features

Flexible Text Input Fields:

Demonstrates various types of input fields including single-line, multi-line, and secure password fields.

Customizable labels, hints, and placeholder text for clear and intuitive data entry.

Comprehensive Input Validation:

Real-time validation rules applied to different field types like email, password strength, and required fields.

Instant feedback on user input errors through visual cues, error messages, and dynamic styling.

Custom validation logic, including regex-based validation and conditional checks, to cover complex requirements.

Enhanced User Experience:

Visual indicators such as focus changes, color highlights, and icons to guide users through form completion.

Error messages that appear and disappear dynamically as inputs change, creating an intuitive and responsive form experience.

Key Features

- **Composable and Modular Components**:
- Reusable components built in a modular way, following best practices for Compose, making it easy to adapt or extend the code for other applications.
- Clean, documented code structure that allows developers to easily understand and apply Compose concepts in their own projects.
- **Theme Compatibility**:
- Supports both light and dark modes, ensuring a consistent experience across Android's theme settings.

Purpose and Audience

- **Compose Input** is intended for Android developers who want to learn how to implement form-based user input and validation using Jetpack Compose. The application demonstrates best practices for handling UI state, validation logic, and user feedback, making it a practical reference for building production-ready Android apps with Compose. The app serves as both a tutorial and a starting point for developers aiming to integrate robust input handling in Compose applications.

Project Goals

- The goal of **Compose Input** is to provide a comprehensive guide for developers aiming to create input forms using Jetpack Compose with robust validation and user-friendly interaction. This project highlights Compose's strengths in managing UI state and offers practical insights into developing responsive, production-ready applications. **Compose Input** is designed to be a learning tool and a reference for developers interested in building forms and handling user input effectively with Jetpack Compose.

MAIN ACTIVITY

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">
  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.SurveyApplication"
    tools:targetApi="31">
    <activity
      android:name=".RegisterActivity"
      android:exported="false"
      android:label="@string/title_activity_register"
      android:theme="@style/Theme.SurveyApplication" />
    <activity
      android:name=".MainActivity"
    android:exported="false"
```

SOURCE CODE

```
  android:label="@string/title_activity_admin"
    android:theme="@style/Theme.SurveyApplication" />
   <activity
    android:name=".LoginActivity"
    android:exported="true"
    android:label="@string/app_name"
    android:theme="@style/Theme.SurveyApplication">
    <intent-filter>
     <action android:name="android.intent.action.MAIN" />



     <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
   </activity>
  </application>



 </manifest>
```

CODE

```
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
```

CODE

```
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
  ) {
```

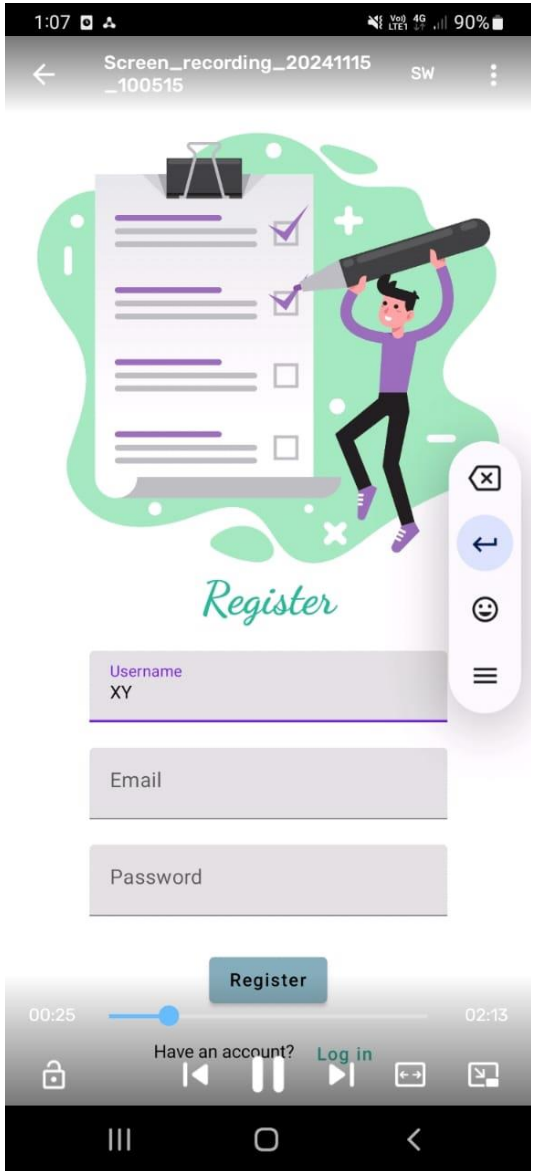
CODE

```
 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)

    )
```

CODE

```
    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)
}
```

OUTPUT



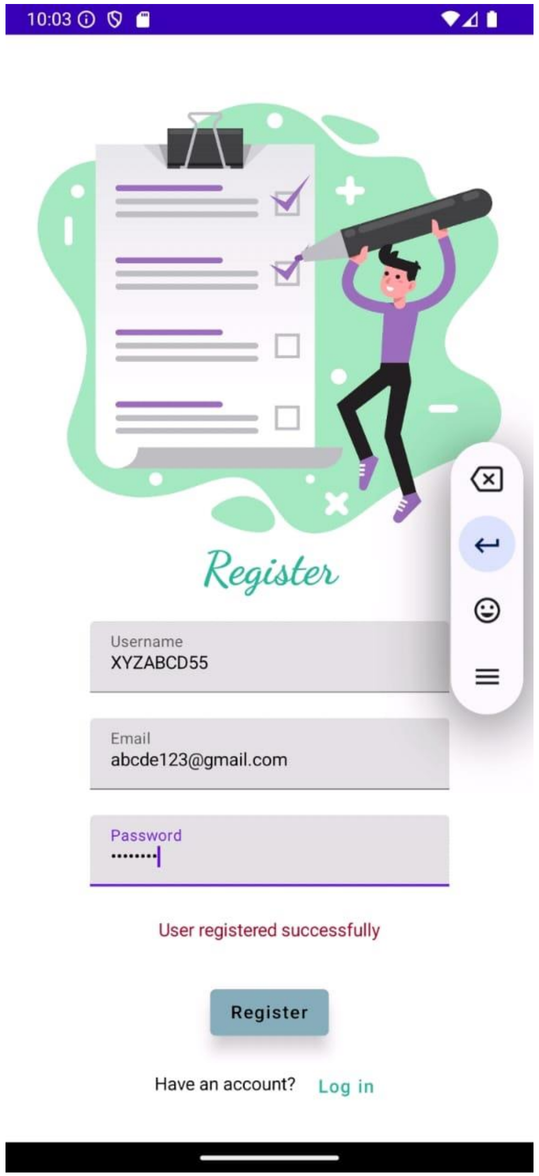
Username
XY

Email

Password

Register

Have an account? Log in

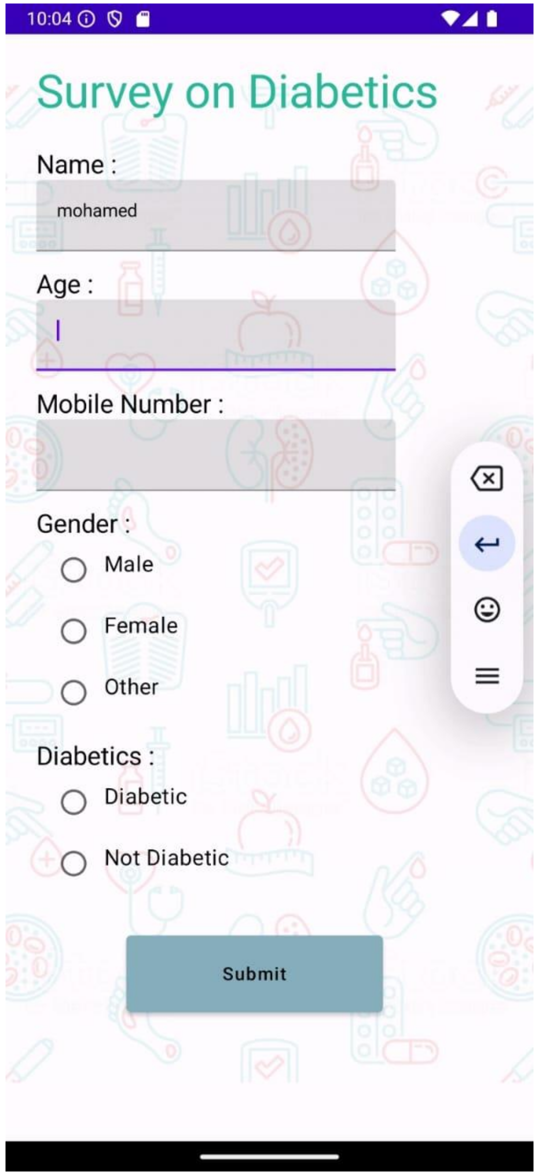


Username
XYZABCD55

Email
abcde123@gmail.com

Password
••••••••

User registered successfully



10:04

Survey on Diabetics

Name : mohamed

Age :

Mobile Number :

Gender :

Male

Female

Other

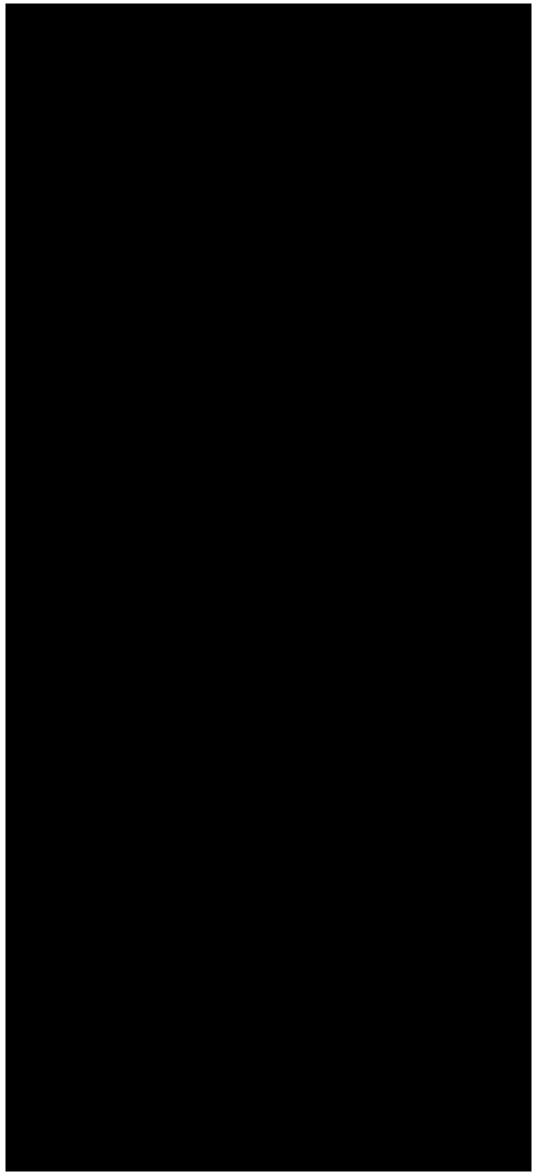
Diabetics :

Diabetic

Not Diabetic

Submit

DEMO VIDEO



Conclusion

The **Compose Input** application serves as a practical demonstration of how Jetpack Compose simplifies the development of dynamic and responsive user interfaces for Android applications. By focusing on text input and validation, the app highlights Compose's capabilities for building intuitive, user-friendly forms with real-time feedback.

Through reusable components, modular code structure, and seamless theme adaptability, the application provides developers with a solid foundation for handling user inputs and implementing validation logic effectively. It bridges the gap between basic Compose concepts and production-ready implementation, making it a valuable resource for Android developers of all skill levels.

Ultimately, **Compose Input** showcases the power and flexibility of Jetpack Compose, empowering developers to create modern, efficient, and visually appealing Android applications while reducing complexity in UI development