NAME: HETAL NIKAM
CLASS: D15A
BATCH : B
ROLL NO. 39

MAD PWA LAB-4

AIM:
To create an interactive Form using form widget

THEORY:
Grouping Form Fields: The Form widget allows us to group multiple form fields together. This grouping is important for managing the state of the form and handling form submission.

State Management: The Form widget manages the state of the form fields within it. It automatically keeps track of the current value of each form field, their validation status, and any errors associated with them.

Validation: The Form widget provides built-in support for form field validation. We can specify validation logic for each form field, and the Form widget will automatically validate the form when it's submitted. Validation ensures that the user input meets certain criteria, such as required fields, valid email addresses, or password length.

Error Reporting: If a form field fails validation, the Form widget displays error messages associated with the invalid fields. These error messages are typically displayed below the corresponding form fields to inform us about what went wrong.

Submission Handling: When the user submits the form, the Form widget can trigger a callback function, allowing us to handle form submission. This callback function can perform actions such as sending data to a server, saving data locally, or navigating to another screen.

GlobalKey: To access the state of a Form widget, we typically use a GlobalKey<FormState>. This key allows us to interact with the Form widget programmatically, such as triggering form validation or resetting the form fields.

Nested Forms: We can nest Form widgets within each other to create complex forms with different sections or groups of fields. Each nested Form manages its own set of form fields independently.

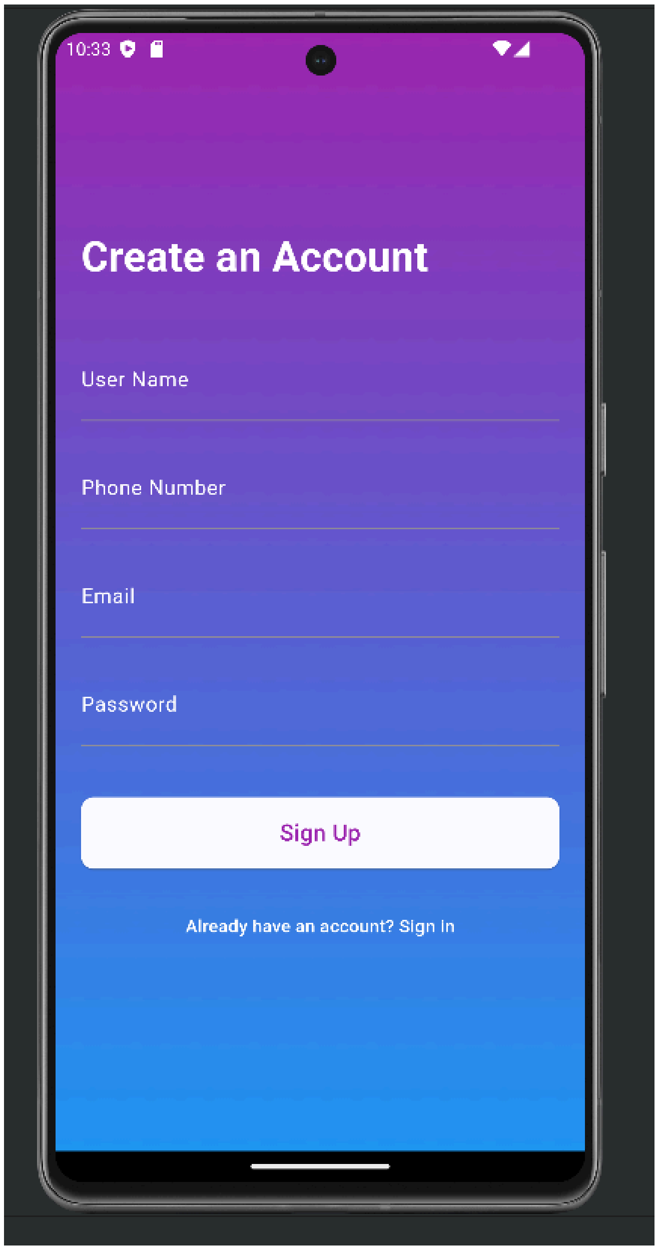
CODE:

```
import 'package:flutter/material.dart';

import 'package:firebase_auth/firebase_auth.dart';

import 'package:firebase_database/firebase_database.dart';

import 'package:uber_practice/methods/common_methods.dart';

import 'package:uber_practice/pages/homepage.dart';

import '../widgets/loading.dart';

import 'login_screen.dart';


class SignUpScreen extends StatefulWidget {

 const SignUpScreen({Key? key}) : super(key: key);


 @override

 State<SignUpScreen> createState() => _SignUpScreenState();

}


class _SignUpScreenState extends State<SignUpScreen> {

 TextEditingController userNameTextEditingController = TextEditingController();

 TextEditingController userPhoneTextEditingController = TextEditingController();

 TextEditingController emailTextEditingController = TextEditingController();

 TextEditingController passwordTextEditingController = TextEditingController();
```

```
CommonMethods cMethods = CommonMethods();


checkIfNetworkIsAvailable() {

  cMethods.checkConnectivity(context);

  signUpFormValidation();

}


signUpFormValidation() {

  if (userNameTextEditingController.text.trim().length < 4) {

    CommonMethods.showToast("Your name must be at least 4 characters");

  } else if (userPhoneTextEditingController.text.trim().length < 10) {

    CommonMethods.showToast("Your number must be at least 10 characters");

  } else if (!emailTextEditingController.text.trim().contains("@")) {

    CommonMethods.showToast("Please enter a valid email address");

  } else if (passwordTextEditingController.text.trim().length < 6) {

    CommonMethods.showToast("Your password must be at least 6 characters");

  } else {

    registerNewUser();

  }

}
```

```
registerNewUser() async {
  showDialog(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) => LoadingDialog(messageText: "Registering your account..."),
  );


  try {
    UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
      email: emailTextEditingController.text.trim(),
      password: passwordTextEditingController.text.trim(),
    );


    if (userCredential != null) {
      DatabaseReference usersRef = FirebaseDatabase.instance.ref().child("users").child(userCredential.user!.uid);
      Map userDataMap = {
        "name": userNameTextEditingController.text.trim(),
        "email": emailTextEditingController.text.trim(),
        "phone": userPhoneTextEditingController.text.trim(),
        "id": userCredential.user!.uid,

        "password":passwordTextEditingController.text.trim(),

        "blockStatus": "no",

      };

      await usersRef.set(userDataMap);



      // Navigate to the home page upon successful registration

      Navigator.pushReplacement(context, MaterialPageRoute(builder: (c) =>
HomePage()));

    }

  } catch (e) {

    print("Error during registration: $e");

    CommonMethods.showToast("Registration failed. Please try again later.");

  } finally {

    Navigator.pop(context); // Close loading dialog

  }

}



@override

Widget build(BuildContext context) {

  return Scaffold(

    backgroundColor: Colors.transparent,
    body: Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          colors: [Colors.purple, Colors.blue],
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
          stops: [0.0, 1.0],
          tileMode: TileMode.clamp,
        ),
      ),
      child: Center(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(20),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  "Create an Account",
                  style: TextStyle(
      fontSize: 32,

      fontWeight: FontWeight.bold,

      color: Colors.white,

    ),

),

SizedBox(height: 40),

TextField(

  controller: userNameTextEditingController,

  keyboardType: TextInputType.text,

  style: TextStyle(color: Colors.white), // Set input text color to white

  decoration: InputDecoration(

    labelText: "User Name",

    labelStyle: TextStyle(color: Colors.white),

  ),

),

SizedBox(height: 20),

TextField(

  controller: userPhoneTextEditingController,

  keyboardType: TextInputType.number,

  style: TextStyle(color: Colors.white), // Set input text color to white

  decoration: InputDecoration(
      labelText: "Phone Number",
      labelStyle: TextStyle(color: Colors.white),
    ),
),
SizedBox(height: 20),
TextField(
  controller: emailTextEditingController,
  keyboardType: TextInputType.emailAddress,
  style: TextStyle(color: Colors.white), // Set input text color to white
  decoration: InputDecoration(
    labelText: "Email",
    labelStyle: TextStyle(color: Colors.white),
  ),
),
SizedBox(height: 20),
TextField(
  controller: passwordTextEditingController,
  obscureText: true,
  keyboardType: TextInputType.text,
  style: TextStyle(color: Colors.white), // Set input text color to white
  decoration: InputDecoration(
      labelText: "Password",
      labelStyle: TextStyle(color: Colors.white),
    ),
),
SizedBox(height: 40),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: () {
      checkIfNetworkIsAvailable();
    },
    style: ElevatedButton.styleFrom(
      primary: Colors.white,
      padding: EdgeInsets.all(15),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(10),
      ),
    ),
    child: Text(
      "Sign Up",
      style: TextStyle(fontSize: 18, color: Colors.purple),
            ),
           ),
          ),
          SizedBox(height: 20),
          Center(
           child: TextButton(
            onPressed: () {
             Navigator.push(context, MaterialPageRoute(builder: (c) =>
LoginScreen()));
          },
          child: Text(
           "Already have an account? Sign in",
           style: TextStyle(
            color: Colors.white,
           ),
          ),
         ),
        ),
       ],
      ),
     ),
    ),
        ),
      ),
    );
   }
}
```

OUTPUT:



CONCLUSION:
Learnt about form widget and applied it in my project.