|  |  |
| --- | --- |
| **Experiment** | 2 |
| **Aim** | Create Dynamic Registration form |
| **Objective** | To Create Dynamic Registration form using UI components and Scaffold elements. |
| **Name** | Atharva Vasant Angre |
| **UCID** | 2024510001 |
| **Class** | FYMCA |
| **Batch** | A |
| **Date of Submission** | 02.02.2025 |

|  |  |
| --- | --- |
| **Technology used** | Flutter, Android Studio |
| **Task** | Create Dynamic Webpage for submitting review for movie app along with giving ratings for the same. The webpage should include, name, surname, date of birth, address, email id, phone number, gender, review and ratings for the movie in terms of stars/smiley etc. include validation. The information should be fetched and displayed on the next page for viewing it. Use snackbar, alert dialog box, try adding fan button (static) etc. |
| **Code with proper label** | **Main.dart**  import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:movie\_rating/Screens/thankYouPage.dart';  import 'Screens/mainScreen.dart';  void main() {  runApp(const MyApp()); }  class MyApp extends StatelessWidget {  const MyApp({super.key});   // This widget is the root of your application.  @override  Widget build(BuildContext context) {  return GetMaterialApp(  debugShowCheckedModeBanner: false,  title: 'Flutter Demo',  home: MainScreen(),  );  } }  **mainScreen.dart**  import 'package:flutter/material.dart'; import 'package:flutter/services.dart'; import 'package:flutter\_rating\_bar/flutter\_rating\_bar.dart'; import 'package:get/get.dart'; import 'package:movie\_rating/Screens/thankYouPage.dart';  import '../Controllers/formController.dart';  class MainScreen extends StatefulWidget {  const MainScreen({super.key});   @override  State<MainScreen> createState() => \_MainScreenState(); }  class \_MainScreenState extends State<MainScreen> {  final FormController formController = Get.put(FormController());   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  backgroundColor: Color(0xffbed5ea),  centerTitle: true,  title: Text("Welcome to Movies Rating App"),  ),  body: Container(  color: Color(0xffffd46d),  padding: EdgeInsets.all(8),  child: Column(  spacing: Get.height \* 0.025,  children: [  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Name"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateName(formController.name.value.text);  }  },  child: Obx(()=>  TextFormField(  controller: formController.name.value,  onChanged: (value) {  formController.name.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateName(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.nameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.nameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.nameError.value.isEmpty ? null : formController.nameError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Surname"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSurname(formController.surname.value.text);  }  },  child: Obx(()=>  TextFormField(  controller: formController.surname.value,  onChanged: (value) {  formController.surname.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateSurname(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Surname',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.surnameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.surnameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.surnameError.value.isEmpty ? null : formController.surnameError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  )  ],  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Email Id"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateEmail(formController.emailId.value.text);  }  },  child: Obx(()=>  TextFormField(  controller: formController.emailId.value,  onChanged: (value) {  formController.emailId.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateEmail(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Email Id',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.emailIdError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.emailIdError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.emailIdError.value.isEmpty ? null : formController.emailIdError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Phone Number"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validatePhoneNumber(formController.phoneNumber.value.text);  }  },  child: Obx(()=>  TextFormField(  controller: formController.phoneNumber.value,  keyboardType: TextInputType.*number*,  inputFormatters: [FilteringTextInputFormatter.*digitsOnly*],  onChanged: (value) {  formController.phoneNumber.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validatePhoneNumber(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Phone Number',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.phoneNumberError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.phoneNumberError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.phoneNumberError.value.isEmpty ? null : formController.phoneNumberError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  )  ],  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Date of Birth"),  Obx(()=>  TextFormField(  controller: formController.dob.value,  readOnly: true,  onTap: () async {  DateTime? pickedDate = await showDatePicker(  context: context,  initialDate: DateTime.now(),  firstDate: DateTime(1900),  lastDate: DateTime.now(),  );   if (pickedDate != null) {  formController.dob.value.text = "${pickedDate.day.toString().padLeft(2, '0')}-${pickedDate.month.toString().padLeft(2, '0')}-${pickedDate.year}";  }  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Date of Birth',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.dobError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.dobError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.dobError.value.isEmpty ? null : formController.dobError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ],  ),  ),  Container(  width: Get.width \* 0.45,   child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Gender"),  Container(  width: Get.width \* 0.21,   padding: EdgeInsets.all(5),  decoration: BoxDecoration(  color: Colors.*white*,  border: Border.all(color: Colors.*black*, width: 2),  borderRadius: BorderRadius.circular(10)),  child: Obx(  ()=> Row(  spacing: Get.height \* 0.050,  mainAxisAlignment: MainAxisAlignment.start,  children: [  Row(  children: [  Radio(  value: 'Male',  groupValue: formController.selectedGender.value,  onChanged: (value) {  formController.selectedGender.value = value!;  },  ),  Text('Male'),  ],  ),  Row(  children: [  Radio(  value: 'Female',  groupValue: formController.selectedGender.value,  onChanged: (value) {  formController.selectedGender.value = value!;  },  ),  Text('Female'),  ],  ),  ],  ),  ),  ),  ],  ),  )  ],  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Movie Name"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateMovieName(formController.movieName.value.text);  }  },  child: Obx(()=>  TextFormField(  controller: formController.movieName.value,  onChanged: (value) {  formController.movieName.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateMovieName(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter Movie Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.movieNameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.movieNameError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.movieNameError.value.isEmpty ? null : formController.movieNameError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  Container(  width: Get.width \* 0.45,  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  spacing: Get.height \* 0.010,  children: [  Text("Rating"),  Container(  padding: EdgeInsets.symmetric(horizontal: Get.width \* 0.015),  decoration: BoxDecoration(  color: Colors.*white*,  borderRadius: BorderRadius.all(Radius.circular(8)),  border: Border.all(color: Colors.*black*,width: 2)  ),  child: RatingBar.builder(  initialRating: formController.rating.value,  itemCount: 5,  minRating: 1,  itemBuilder: (context, index) {  switch (index) {  case 0:  return Icon(Icons.*sentiment\_very\_dissatisfied*, color: Colors.*red*);  case 1:  return Icon(Icons.*sentiment\_dissatisfied*, color: Colors.*redAccent*);  case 2:  return Icon(Icons.*sentiment\_neutral*, color: Colors.*amber*);  case 3:  return Icon(Icons.*sentiment\_satisfied*, color: Colors.*lightGreen*);  case 4:  return Icon(Icons.*sentiment\_very\_satisfied*, color: Colors.*green*);  default:  return SizedBox.shrink();  }  },  onRatingUpdate: (rating) {  formController.rating.value = rating;  },  ),  ),  ],  ),  ),  ],  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Address"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateAddress(formController.address.value.text);  }  },  child: Obx(()=>  TextFormField(  maxLines: 3,  controller: formController.address.value,  onChanged: (value) {  formController.address.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateAddress(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.addressError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.addressError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.addressError.value.isEmpty ? null : formController.addressError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Review"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateReview(formController.review.value.text);  }  },  child: Obx(()=>  TextFormField(  maxLines: 3,  controller: formController.review.value,  onChanged: (value) {  formController.review.value.value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(offset: value.length),  );  formController.validateReview(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),  labelText: 'Please Enter your Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.reviewError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.reviewError.value.isEmpty ? Colors.*black* : Colors.*red*,  width: 1.5,  ),  borderRadius: BorderRadius.circular(8),  ),  errorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  focusedErrorBorder: OutlineInputBorder(  borderSide: BorderSide(color: Colors.*red*, width: 1.5),  borderRadius: BorderRadius.circular(8),  ),  errorText: formController.reviewError.value.isEmpty ? null : formController.reviewError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   GestureDetector(  onTap: () {  formController.validateAllFields();  if (formController.isFormValid()) {  print("name: "+formController.name.value.text);  formController.box.write('name', formController.name.value.text);  print("surname: " +formController.surname.value.text);  formController.box.write('surname', formController.surname.value.text);  print("emailId: "+formController.emailId.value.text);  formController.box.write('emailId', formController.emailId.value.text);  print("phoneNumber: "+formController.phoneNumber.value.text);  formController.box.write('phoneNumber', formController.phoneNumber.value.text);  print("dob: "+formController.dob.value.text);  formController.box.write('dob', formController.dob.value.text);  print("selectedGender: "+formController.selectedGender.value);  formController.box.write('selectedGender', formController.selectedGender.value);  print("address: "+formController.address.value.text);  formController.box.write('address', formController.address.value.text);  print("review: "+formController.review.value.text);  formController.box.write('review', formController.review.value.text);  print("Rating : "+formController.rating.value.toString());  formController.box.write('Rating',formController.rating.value );  print("Form submitted!");   Get.to(ThankYouPage());  } else {  Get.snackbar(  'Error',  'Please fill all fields correctly',  snackPosition: SnackPosition.BOTTOM,  backgroundColor: Colors.*red*,  colorText: Colors.*white*,  );  }  },  child: Container(  padding: EdgeInsets.symmetric(horizontal: Get.width \* 0.025,vertical: Get.height \* 0.005),  decoration: BoxDecoration(  color: Color(0xffbed5ea),  borderRadius: BorderRadius.all(Radius.circular(8)),  border: Border.all(color: Colors.*black*, width: 2.5),  ),  child: Text("Submit")),  ),  ],  ),  ),  );  } }  **thankYouPage.dart**  import 'package:flutter/material.dart'; import 'package:flutter\_rating\_bar/flutter\_rating\_bar.dart'; import 'package:get/get.dart'; import 'package:get/get\_core/src/get\_main.dart'; import 'package:movie\_rating/Screens/mainScreen.dart';  import '../Controllers/formController.dart';  class ThankYouPage extends StatefulWidget {  const ThankYouPage({super.key});   @override  State<ThankYouPage> createState() => \_ThankYouPageState(); }  class \_ThankYouPageState extends State<ThankYouPage> {  final FormController formController = Get.put(FormController());   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  leading: BackButton(  onPressed: (){  formController.resetForm();  Get.to(MainScreen());  },  ),  centerTitle: true,  backgroundColor: Color(0xffbed5ea),  title: Text("Movies Rating App"),   ),  body: Container(  color: Color(0xffffd46d),  child: Center(  child: Column(  crossAxisAlignment: CrossAxisAlignment.center,  children: [  Icon(  Icons.*mail\_outline\_rounded*,size: 200,  color: Colors.*black*,  ),  Text("Thank you for your valuable Review"),  Container(  padding: EdgeInsets.all(8),  decoration: BoxDecoration(  color: Colors.*white*,  border: Border.all(color: Colors.*black*,width: 2),  borderRadius: BorderRadius.all(Radius.circular(8))  ),  child:Column(  children: [  Text("Your Review"),  Text("Name: "+formController.box.read("name")),  Text("Surname: "+formController.box.read("surname")),  Text("Email Id: "+formController.box.read("emailId")),  Text("Phone Number: "+formController.box.read("phoneNumber")),  Text("Date of Birth: "+formController.box.read("dob")),  Text("Gender: "+formController.box.read("selectedGender")),  Text("Address: "+formController.box.read("address")),  Text("Review: "+formController.box.read("review")),  RatingBar.builder(  initialRating: formController.rating.value,  itemCount: 5,  minRating: 1,  itemBuilder: (context, index) {  switch (index) {  case 0:  return Icon(Icons.*sentiment\_very\_dissatisfied*, color: Colors.*red*);  case 1:  return Icon(Icons.*sentiment\_dissatisfied*, color: Colors.*redAccent*);  case 2:  return Icon(Icons.*sentiment\_neutral*, color: Colors.*amber*);  case 3:  return Icon(Icons.*sentiment\_satisfied*, color: Colors.*lightGreen*);  case 4:  return Icon(Icons.*sentiment\_very\_satisfied*, color: Colors.*green*);  default:  return SizedBox.shrink();  }  },  ignoreGestures: true,  onRatingUpdate: (rating) {   },  ),  ],  ),  ),    SizedBox(height: Get.height \* 0.05,),  Row(  mainAxisAlignment: MainAxisAlignment.center,  spacing: Get.width \* 0.055,  children: [  GestureDetector(  onTap: (){  Get.to(MainScreen());  },  child: Container(  padding: EdgeInsets.all(10),  decoration: BoxDecoration(  borderRadius: BorderRadius.all(Radius.circular(8)),  border: Border.all(color: Colors.*black*,width: 2),  color: Color(0xffbed5ea),  ),  child: Text("Make Some Changes"),  ),  ),  GestureDetector(  onTap: (){  Get.to(MainScreen());  formController.resetForm();  },  child: Container(  padding: EdgeInsets.all(10),  decoration: BoxDecoration(  borderRadius: BorderRadius.all(Radius.circular(8)),  border: Border.all(color: Colors.*black*,width: 2),  color: Color(0xffbed5ea),  ),  child: Text("Add Another"),  ),  ),  ],  ),   ],  ),  ),  ),  );  } }  **formController.dart**  import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:get/get\_state\_manager/src/simple/get\_controllers.dart'; import 'package:get\_storage/get\_storage.dart';  class FormController extends GetxController {   final box = GetStorage();    Rx<TextEditingController> name = TextEditingController().obs;  Rx<TextEditingController> surname = TextEditingController().obs;  Rx<TextEditingController> dob = TextEditingController().obs;  Rx<TextEditingController> emailId = TextEditingController().obs;  Rx<TextEditingController> phoneNumber = TextEditingController().obs;  Rx<TextEditingController> address = TextEditingController().obs;  Rx<TextEditingController> review = TextEditingController().obs;  Rx<TextEditingController> movieName = TextEditingController().obs;   var rating = 0.0.obs;  var selectedGender = 'Male'.obs;   var nameError = ''.obs;  var surnameError = ''.obs;  var emailIdError = ''.obs;  var phoneNumberError = ''.obs;  var dobError = ''.obs;  var addressError = ''.obs;  var reviewError = ''.obs;  var movieNameError = ''.obs;   void validateName(String value) {  if (value.isEmpty) {  nameError.value = 'Name is required';  } else if (value.length < 2) {  nameError.value = 'Name must be at least 2 characters';  } else if (!RegExp(r'^[a-zA-Z\s]+$').hasMatch(value)) {  nameError.value = 'Name can only contain letters';  } else {  nameError.value = '';  }  }   void validateSurname(String value) {  if (value.isEmpty) {  surnameError.value = 'Surname is required';  } else if (value.length < 2) {  surnameError.value = 'Surname must be at least 2 characters';  } else if (!RegExp(r'^[a-zA-Z\s]+$').hasMatch(value)) {  surnameError.value = 'Surname can only contain letters';  } else {  surnameError.value = '';  }  }   void validateEmail(String value) {  if (value.isEmpty) {  emailIdError.value = 'Email is required';  } else if (!GetUtils.*isEmail*(value)) {  emailIdError.value = 'Please enter a valid email address';  } else {  emailIdError.value = '';  }  }   void validatePhoneNumber(String value) {  if (value.isEmpty) {  phoneNumberError.value = 'Phone number is required';  } else if (value.length != 10) {  phoneNumberError.value = 'Phone number must be exactly 10 digits';  } else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  phoneNumberError.value = 'Phone number can only contain digits';  } else {  phoneNumberError.value = '';  }  }   void validateDob(String value) {  if (value.isEmpty) {  dobError.value = 'Date of birth is required';  } else {  dobError.value = '';  }  }   void validateAddress(String value) {  if (value.isEmpty) {  addressError.value = 'Address is required';  } else if (value.length < 10) {  addressError.value = 'Please enter a complete address';  } else {  addressError.value = '';  }  }   void validateReview(String value) {  if (value.isEmpty) {  reviewError.value = 'Review is required';  } else {  reviewError.value = '';  }  }  void validateMovieName(String value) {  if (value.isEmpty) {  movieNameError.value = 'Movie Name is required';  }  else{  movieNameError.value = '';  }  }   void validateAllFields() {  validateName(name.value.text);  validateSurname(surname.value.text);  validateEmail(emailId.value.text);  validatePhoneNumber(phoneNumber.value.text);  validateDob(dob.value.text);  validateAddress(address.value.text);  validateReview(review.value.text);  }   bool isFormValid() {  return nameError.value.isEmpty &&  surnameError.value.isEmpty &&  emailIdError.value.isEmpty &&  phoneNumberError.value.isEmpty &&  dobError.value.isEmpty &&  addressError.value.isEmpty &&  reviewError.value.isEmpty &&  name.value.text.isNotEmpty &&  surname.value.text.isNotEmpty &&  emailId.value.text.isNotEmpty &&  phoneNumber.value.text.isNotEmpty &&  dob.value.text.isNotEmpty &&  address.value.text.isNotEmpty &&  review.value.text.isNotEmpty &&  rating.value > 0;  }   void resetForm() {  name.value.text = '';  surname.value.text = '';  dob.value.text = '';  emailId.value.text = '';  phoneNumber.value.text = '';  address.value.text = '';  review.value.text = '';  rating.value = 0.0;  selectedGender.value = 'Male';   nameError.value = '';  surnameError.value = '';  emailIdError.value = '';  phoneNumberError.value = '';  dobError.value = '';  addressError.value = '';  reviewError.value = '';  } } |
| **Screenshots** |  |
| **Question and Answers** | Answer the following Questions:   1. How to create snackbar? Which component of flutter it belongs to?  * The default Snackbar belongs to Flutter’s material library (ScaffoldMessenger).   But as I am using GetX, GetX Snackbar is part of the GetX package, which simplifies state management and UI interactions.  **Default snackbar**:  ScaffoldMessenger.of(context).showSnackBar(  SnackBar(  content: Text("This is a Snackbar"),  duration: Duration(seconds: 3), // How long the Snackbar stays  backgroundColor: Colors.blue, // Background color  action: SnackBarAction(  label: "Undo",  onPressed: () {  print("Undo action clicked");  },  textColor: Colors.white,  ),  ),  );  **GetX snackbar**:  Get.snackbar(  'Error', // Title  'Please fill all fields correctly', // Message  snackPosition: SnackPosition.BOTTOM, // Position  backgroundColor: Colors.red, // Background color  colorText: Colors.white, // Text color  );   1. How to give validation to any field in flutter?  * Every field has a validation method that checks for empty values, length, format, etc.   Example: validateEmail  void validateEmail(String value) {  if (value.isEmpty) {  emailIdError.value = 'Email is required';  } else if (!GetUtils.isEmail(value)) {  emailIdError.value = 'Please enter a valid email address';  } else {  emailIdError.value = '';  }  }  Here in the above given function,  If the user does not enter anything in the email field, it sets the error message to "**Email is required**".  GetUtils.isEmail(value) is a utility function from GetX that verifies if the input matches a valid email format. If the email format is incorrect, it sets the error message to **"Please enter a valid email address"**.  If the email is not empty and is correctly formatted, the error message is cleared   1. How to navigate from one page to another?  * As I am using GetX, navigation can be done using:   Get.to(Page\_Name());  or the normal way:  Navigator.push( context, MaterialPageRoute(builder: (context) => Page\_Name()) );   1. How are values of variables fetched in flutter.   In this project values are stored using GetX's reactive variables (Rx) and GetStorage.  Using **.obs** (Reactive State Management)  var name = ''.obs;  Fetching:  Obx(() => Text(formController.name.value))  Using **GetStorage** (Local Storage)  formController.box.write('name', formController.name.value.text);  Fetching:  String storedName = formController.box.read("name");  **Fetching Controller Text Values**  String enteredName = formController.name.value.text; |
| **Conclusion** | I learned how to use Flutter's default and GetX Snackbar for showing messages, with GetX offering more customization. I also understood how to validate form fields, like checking if an email is valid, and displaying error messages with GetX's reactive variables. Additionally, I learned the two ways to navigate between pages: using GetX's Get.to() and the standard Navigator.push(). Lastly, I explored how to store and retrieve values using GetX's reactive state and GetStorage for local data storage, making it easier to manage app data efficiently. |