|  |  |
| --- | --- |
| **Experiment** | 4 |
| **Aim** | Develop an app for TPO Office at SPIT |
| **Objective** | * To Create TPO App using UI Component * To Develop SideDrawer Navigation * To Use 60-30-10 rule |
| **Name** | Atharva Vasant Angre |
| **UCID** | 2024510001 |
| **Class** | FYMCA |
| **Batch** | A |
| **Date of Submission** | 03.03.25 |

|  |  |
| --- | --- |
| **Technology used** | **Flutter** |
| **Task** | Develop an app for TPO with the following requirement (Use side navigation drawer) :   1. Registration Form (Name, E-mail id, Contact, Roll no, HSC (Which College, Year of Passing, total, out of, should calculate percentage),SSC marks (Which College, Year of Passing, total, out of, should calculate percentage), Marks in sem 1-5 (CGPA and Percentage), upload Resume, Any Additional Courses). 2. User should be able to view the uploaded Resume. (optional) 3. Company details should be visible to the user. (Name of the company, Location, Range of the salary be given for internship/Placement, Short Profile). 4. Student should be able to view their own details. |
| **Code with proper label** | **Main.dart**  import 'package:flutter/material.dart'; import 'package:get/get.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',  theme: ThemeData(  primaryColor: Colors.*white*,  useMaterial3: true,  ),  home: MainScreen(),  );  } }  **commonDrawer.dart**  import 'dart:io';  import 'package:flutter/material.dart'; import 'package:flutter/services.dart'; import 'package:get/get.dart'; import 'package:lab4/screens/companiesPage.dart'; import 'package:lab4/screens/profilePage.dart'; import 'package:lab4/screens/registerPage.dart';  import '../controllers/registrationController.dart';  class CommonDrawer extends StatelessWidget {  @override  Widget build(BuildContext context) {  final FormController formController = Get.put(FormController());  var isProfileComplete = formController.box.read("isProfileComplete");  return Drawer(  child: ListView(  padding: EdgeInsets.*zero*,  children: [  UserAccountsDrawerHeader(  decoration: BoxDecoration(  color: Color(0xff1976D2),  ),  accountName: isProfileComplete != null  ? Text(  "Atharva Angre",  style: TextStyle(color: Colors.*white*),  )  : Text(  "----- -----",  style: TextStyle(color: Colors.*white*),  ),  accountEmail: isProfileComplete != null  ? Text(  "angreatharva08@gmail.com",  style: TextStyle(color: Colors.*white*),  )  : Text(  "-----@gmail.com",  style: TextStyle(color: Colors.*white*),  ),  currentAccountPicture:  isProfileComplete != null  ? CircleAvatar(  radius: 100,  backgroundImage: FileImage(  File(formController.box.read("profileImage"))),  )  : CircleAvatar(  radius: 100,  backgroundColor: Colors.*grey*[200], // Background color for default  child: Stack(  alignment: Alignment.*center*,  children: [  Icon(  Icons.*person*, // Default icon  size: 50,  color: Colors.*grey*[700], // Icon color  ),  ],  ),  ),  ),  ListTile(  leading: Icon(Icons.*home*),  title: Text("Companies"),  onTap: () {  Get.back();  Get.to(CompaniesPage());  }),  ListTile(  leading: Icon(Icons.*person*),  title: Text("Profile"),  onTap: () {  Get.back();  Get.to(ProfilePage());  }),  ListTile(  leading: Icon(Icons.*app\_registration\_rounded*),  title: Text("Register"),  onTap: () {  Get.back();  Get.to(RegisterPage());  }),  Divider(),  ListTile(  leading: Icon(Icons.*logout*),  title: Text("Exit App"),  onTap: () {  SystemNavigator.*pop*();  },  ),  ],  ),  );  } }  **RegistrationController.dart**  import 'package:file\_picker/file\_picker.dart'; import 'package:flutter/material.dart'; import 'package:get/get.dart'; import 'package:get\_storage/get\_storage.dart'; import 'package:permission\_handler/permission\_handler.dart';  class FormController extends GetxController {   @override  void onInit() {  // *TODO: implement onInit* super.onInit();  profileImagePath.value = box.read("profileImage") ?? '';  selectedFilePath.value = box.read("resume") ?? '';  requestPermissions();  }  var selectedFilePath = "".obs;  var profileImagePath = "".obs;  var isPdf = false.obs;  var isImage = false.obs;  var isProfileComplete = false.obs;   Future<void> requestPermissions() async {  await Permission.*storage*.request();  }   Future<void> pickFile() async {  FilePickerResult? result = await FilePicker.*platform*.pickFiles(  allowCompression: true,  type: FileType.custom,  allowedExtensions: ['pdf', 'jpg', 'jpeg', 'png'],  );   if (result != null && result.files.single.path != null) {  selectedFilePath.value = result.files.single.path!;  String fileExtension = result.files.single.extension?.toLowerCase() ?? "";   isPdf.value = fileExtension == 'pdf';  isImage.value = ['jpg', 'jpeg', 'png'].contains(fileExtension);  }  }   Future<void> pickProfileImage() async {  FilePickerResult? result = await FilePicker.*platform*.pickFiles(  allowCompression: true,  type: FileType.image,  );   if (result != null && result.files.single.path != null) {  profileImagePath.value = result.files.single.path!;  }  }   final box = GetStorage();   Rx<TextEditingController> name = TextEditingController().obs;  Rx<TextEditingController> emailId = TextEditingController().obs;  Rx<TextEditingController> phoneNumber = TextEditingController().obs;  Rx<TextEditingController> ucid = TextEditingController().obs;   Rx<TextEditingController> collegeNameGraduation = TextEditingController().obs;  Rx<TextEditingController> sem1 = TextEditingController().obs;  Rx<TextEditingController> sem2 = TextEditingController().obs;  Rx<TextEditingController> sem3 = TextEditingController().obs;  Rx<TextEditingController> sem4 = TextEditingController().obs;  Rx<TextEditingController> sem5 = TextEditingController().obs;  Rx<TextEditingController> sem6 = TextEditingController().obs;  Rx<TextEditingController> CGPA = TextEditingController().obs;  Rx<TextEditingController> percentageGraduation = TextEditingController().obs;   Rx<TextEditingController> collegeNameSSC = TextEditingController().obs;  Rx<TextEditingController> yearOfPassingSSC = TextEditingController().obs;  Rx<TextEditingController> marksObtainedSSC = TextEditingController().obs;  Rx<TextEditingController> totalMarksSSC = TextEditingController().obs;  Rx<TextEditingController> percentageSSC = TextEditingController().obs;   Rx<TextEditingController> collegeNameHSC = TextEditingController().obs;  Rx<TextEditingController> yearOfPassingHSC = TextEditingController().obs;  Rx<TextEditingController> marksObtainedHSC = TextEditingController().obs;  Rx<TextEditingController> totalMarksHSC = TextEditingController().obs;  Rx<TextEditingController> percentageHSC = TextEditingController().obs;    var nameError = ''.obs;  var emailIdError = ''.obs;  var phoneNumberError = ''.obs;  var ucidError = ''.obs;   var collegeNameErrorGraduation= ''.obs;  var sem1Error= ''.obs;  var sem2Error= ''.obs;  var sem3Error= ''.obs;  var sem4Error= ''.obs;  var sem5Error= ''.obs;  var sem6Error= ''.obs;  var CGPAError= ''.obs;  var percentageError= ''.obs;    var collegeNameErrorSSC = ''.obs;  var yearOfPassingErrorSSC = ''.obs;  var marksObtainedErrorSSC = ''.obs;  var totalMarksErrorSSC = ''.obs;  var percentageErrorSSC = ''.obs;   var collegeNameErrorHSC= ''.obs;  var yearOfPassingErrorHSC = ''.obs;  var marksObtainedErrorHSC = ''.obs;  var totalMarksErrorHSC = ''.obs;  var percentageErrorHSC = ''.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]+$').hasMatch(value)) {  nameError.value = 'Name can only contain letters';  } else {  nameError.value = '';  }  }   void validateEmail(String value) {  if (value.isEmpty) {  emailIdError.value = 'Email is required';  } else if (!GetUtils.*isEmail*(value)) {  emailIdError.value = 'Please enter a valid email address';  } else {  emailIdError.value = '';  }  }   void validatePhoneNumber(String value) {  if (value.isEmpty) {  phoneNumberError.value = 'Phone number is required';  } else if (value.length != 10) {  phoneNumberError.value = 'Phone number must be exactly 10 digits';  } else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  phoneNumberError.value = 'Phone number can only contain digits';  } else {  phoneNumberError.value = '';  }  }   void validateUCID(String value) {  if (value.isEmpty) {  ucidError.value = 'UCID is required';  } else if (value.length != 10) {  ucidError.value = 'UCID must be exactly 10 digits';  } else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  ucidError.value = 'UCID can only contain digits';  } else {  ucidError.value = '';  }  }   void validateCollegeNameGraduation(String value) {  if (value.isEmpty) {  collegeNameErrorGraduation.value = 'College Name is required';  } else if (value.length < 2) {  collegeNameErrorGraduation.value = 'College Name must be at least 2 characters';  } else if (!RegExp(r'^[a-zA-Z]+$').hasMatch(value)) {  collegeNameErrorGraduation.value = 'College Name can only contain letters';  } else {  collegeNameErrorGraduation.value = '';  }  }   void validateSem1(String value) {  if (value.isEmpty) {  sem1Error.value = 'Sem1 are required';  } else {  sem1Error.value = '';  }  }  void validateSem2(String value) {  if (value.isEmpty) {  sem2Error.value = 'Sem2 are required';  } else {  sem2Error.value = '';  }  }  void validateSem3(String value) {  if (value.isEmpty) {  sem3Error.value = 'Sem3 are required';  } else {  sem3Error.value = '';  }  }  void validateSem4(String value) {  if (value.isEmpty) {  sem4Error.value = 'Sem4 are required';  } else {  sem4Error.value = '';  }  }  void validateSem5(String value) {  if (value.isEmpty) {  sem5Error.value = 'Sem5 are required';  } else {  sem5Error.value = '';  }  }  void validateSem6(String value) {  if (value.isEmpty) {  sem6Error.value = 'Sem6 are required';  } else {  sem6Error.value = '';  }  }  void validateCGPA() {  double percentage = (double.*parse*(sem1.value.text) + double.*parse*(sem2.value.text) + double.*parse*(sem3.value.text) + double.*parse*(sem4.value.text) + double.*parse*(sem5.value.text) + double.*parse*(sem6.value.text)) / 6;  CGPA.value.text = percentage.toStringAsFixed(2);  print("CGPA: ${CGPA.value.text}");  }  void calculatePercentageGraduation() {  double percentage =(7.1 \*  (double.*parse*(sem1.value.text) + double.*parse*(sem2.value.text) + double.*parse*(sem3.value.text) + double.*parse*(sem4.value.text) + double.*parse*(sem5.value.text) + double.*parse*(sem6.value.text)) / 6  +(11))  ;  percentageGraduation.value.text = percentage.toStringAsFixed(2);  print("percentageGraduation : ${percentageGraduation.value.text}");  }   void validateCollegeNameSSC(String value) {  if (value.isEmpty) {  collegeNameErrorSSC.value = 'College Name is required';  } else if (value.length < 2) {  collegeNameErrorSSC.value = 'College Name must be at least 2 characters';  } else if (!RegExp(r"^[a-zA-Z\s.']+$").hasMatch(value)) {  collegeNameErrorSSC.value = 'College Name can only contain letters';  } else {  collegeNameErrorSSC.value = '';  }  }   void validateYearOfPassingSSC(String value) {  if (value.isEmpty) {  yearOfPassingErrorSSC.value = 'Year of Passing is required';  } else {  yearOfPassingErrorSSC.value = '';  }  }   void validateMarksObtainedSSC(String value) {  if (value.isEmpty) {  marksObtainedErrorSSC.value = 'Marks Obtained is required';  }else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  marksObtainedErrorSSC.value = 'Marks Obtained can only contain digits';  } else {  marksObtainedErrorSSC.value = '';  }  }   void validateTotalMarksSSC(String value) {  if (value.isEmpty) {  totalMarksErrorSSC.value = 'Total Marks is required';  }else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  totalMarksErrorSSC.value = 'Total Marks can only contain digits';  } else {  totalMarksErrorSSC.value = '';  }  }   void calculatePercentageSSC() {  double percentage = (double.*parse*(marksObtainedSSC.value.text) / double.*parse*(totalMarksSSC.value.text)) \* 100;  percentageSSC.value.text = percentage.toStringAsFixed(2);  print("percentageSSC: ${percentageSSC.value.text}");  }    void validateCollegeNameHSC(String value) {  if (value.isEmpty) {  collegeNameErrorHSC.value = 'College Name is required';  } else if (value.length < 2) {  collegeNameErrorHSC.value = 'College Name must be at least 2 characters';  } else if (!RegExp(r"^[a-zA-Z\s.']+$").hasMatch(value)) {  collegeNameErrorHSC.value = 'College Name can only contain letters';  } else {  collegeNameErrorHSC.value = '';  }  }   void validateYearOfPassingHSC(String value) {  if (value.isEmpty) {  yearOfPassingErrorHSC.value = 'Year of Passing is required';  } else {  yearOfPassingErrorHSC.value = '';  }  }   void validateMarksObtainedHSC(String value) {  if (value.isEmpty) {  marksObtainedErrorHSC.value = 'Marks Obtained is required';  }else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  marksObtainedErrorHSC.value = 'Marks Obtained can only contain digits';  } else {  marksObtainedErrorHSC.value = '';  }  }   void validateTotalMarksHSC(String value) {  if (value.isEmpty) {  totalMarksErrorHSC.value = 'Total Marks is required';  }else if (!RegExp(r'^[0-9]+$').hasMatch(value)) {  totalMarksErrorHSC.value = 'Total Marks can only contain digits';  } else {  totalMarksErrorHSC.value = '';  }  }   void calculatePercentageHSC() {  double percentage = (double.*parse*(marksObtainedHSC.value.text) / double.*parse*(totalMarksHSC.value.text)) \* 100;  percentageHSC.value.text = percentage.toStringAsFixed(2);  print("percentageHSC: ${percentageHSC.value.text}");  }    void validateAllFields() {  validateName(name.value.text);  validateEmail(emailId.value.text);  validatePhoneNumber(phoneNumber.value.text);  validateUCID(ucid.value.text);  validateCollegeNameGraduation(collegeNameGraduation.value.text);  validateSem1(sem1.value.text);  validateSem2(sem2.value.text);  validateSem3(sem3.value.text);  validateSem4(sem4.value.text);  validateSem5(sem5.value.text);  validateSem6(sem6.value.text);  validateCGPA();  calculatePercentageGraduation();  validateCollegeNameSSC(collegeNameSSC.value.text);  validateYearOfPassingSSC(yearOfPassingSSC.value.text);  validateMarksObtainedSSC(marksObtainedSSC.value.text);  validateTotalMarksSSC(totalMarksSSC.value.text);  calculatePercentageSSC();  validateCollegeNameHSC(collegeNameHSC.value.text);  validateYearOfPassingHSC(yearOfPassingHSC.value.text);  validateMarksObtainedHSC(marksObtainedHSC.value.text);  validateTotalMarksHSC(totalMarksHSC.value.text);  calculatePercentageHSC();  }   bool isFormValid() {  return nameError.isEmpty &&  emailIdError.isEmpty &&  phoneNumberError.isEmpty &&  ucidError.isEmpty &&  collegeNameErrorGraduation.isEmpty &&  sem1Error.isEmpty &&  sem2Error.isEmpty &&  sem3Error.isEmpty &&  sem4Error.isEmpty &&  sem5Error.isEmpty &&  sem6Error.isEmpty &&  CGPAError.isEmpty &&  percentageError.isEmpty &&  collegeNameErrorSSC.isEmpty &&  yearOfPassingErrorSSC.isEmpty &&  marksObtainedErrorSSC.isEmpty &&  totalMarksErrorSSC.isEmpty &&  percentageErrorSSC.isEmpty &&  collegeNameErrorHSC.isEmpty &&  yearOfPassingErrorHSC.isEmpty &&  marksObtainedErrorHSC.isEmpty &&  totalMarksErrorHSC.isEmpty &&  percentageErrorHSC.isEmpty &&  name.value.text.isNotEmpty &&  emailId.value.text.isNotEmpty &&  phoneNumber.value.text.isNotEmpty &&  ucid.value.text.isNotEmpty &&  collegeNameGraduation.value.text.isNotEmpty &&  sem1.value.text.isNotEmpty &&  sem2.value.text.isNotEmpty &&  sem3.value.text.isNotEmpty &&  sem4.value.text.isNotEmpty &&  sem5.value.text.isNotEmpty &&  sem6.value.text.isNotEmpty &&  CGPA.value.text.isNotEmpty &&  percentageGraduation.value.text.isNotEmpty &&  collegeNameSSC.value.text.isNotEmpty &&  yearOfPassingSSC.value.text.isNotEmpty &&  marksObtainedSSC.value.text.isNotEmpty &&  totalMarksSSC.value.text.isNotEmpty &&  percentageSSC.value.text.isNotEmpty &&  collegeNameHSC.value.text.isNotEmpty &&  yearOfPassingHSC.value.text.isNotEmpty &&  marksObtainedHSC.value.text.isNotEmpty &&  totalMarksHSC.value.text.isNotEmpty &&  percentageHSC.value.text.isNotEmpty;  }   void storeFormValues() {   box.write('name', name.value.text);  box.write('emailId', emailId.value.text);  box.write('phoneNumber', phoneNumber.value.text);  box.write('ucid', ucid.value.text);  box.write('collegeNameGraduation', collegeNameGraduation.value.text);  box.write('sem1', sem1.value.text);  box.write('sem2', sem2.value.text);  box.write('sem3', sem3.value.text);  box.write('sem4', sem4.value.text);  box.write('sem5', sem5.value.text);  box.write('sem6', sem6.value.text);  box.write('CGPA', CGPA.value.text);  box.write('percentageGraduation', percentageGraduation.value.text);  box.write('collegeNameSSC', collegeNameSSC.value.text);  box.write('yearOfPassingSSC', yearOfPassingSSC.value.text);  box.write('marksObtainedSSC', marksObtainedSSC.value.text);  box.write('totalMarksSSC', totalMarksSSC.value.text);  box.write('percentageSSC', percentageSSC.value.text);  box.write('collegeNameHSC', collegeNameHSC.value.text);  box.write('yearOfPassingHSC', yearOfPassingHSC.value.text);  box.write('marksObtainedHSC', marksObtainedHSC.value.text);  box.write('totalMarksHSC', totalMarksHSC.value.text);  box.write('percentageHSC', percentageHSC.value.text);  box.write('profileImage', profileImagePath.value);  box.write('resume', selectedFilePath.value);  box.write('isPdf', isPdf.value);  box.write('isProfileComplete', isProfileComplete.value);    print('Form data stored successfully!');  }   void resetForm() {  name.value.clear();  emailId.value.clear();  phoneNumber.value.clear();  ucid.value.clear();  collegeNameGraduation.value.clear();  sem1.value.clear();  sem2.value.clear();  sem3.value.clear();  sem4.value.clear();  sem5.value.clear();  sem6.value.clear();  CGPA.value.clear();  percentageGraduation.value.clear();  collegeNameSSC.value.clear();  yearOfPassingSSC.value.clear();  marksObtainedSSC.value.clear();  totalMarksSSC.value.clear();  percentageSSC.value.clear();  collegeNameHSC.value.clear();  yearOfPassingHSC.value.clear();  marksObtainedHSC.value.clear();  totalMarksHSC.value.clear();  percentageHSC.value.clear();   nameError.value = '';  emailIdError.value = '';  phoneNumberError.value = '';  ucidError.value = '';  collegeNameErrorGraduation.value = '';  sem1Error.value = '';  sem2Error.value = '';  sem3Error.value = '';  sem4Error.value = '';  sem5Error.value = '';  sem6Error.value = '';  CGPAError.value = '';  percentageError.value = '';  collegeNameErrorSSC.value = '';  yearOfPassingErrorSSC.value = '';  marksObtainedErrorSSC.value = '';  totalMarksErrorSSC.value = '';  percentageErrorSSC.value = '';  collegeNameErrorHSC.value = '';  yearOfPassingErrorHSC.value = '';  marksObtainedErrorHSC.value = '';  totalMarksErrorHSC.value = '';  percentageErrorHSC.value = '';  }  }  **registerPage.dart** import 'dart:io';  import 'package:flutter/material.dart'; import 'package:flutter/services.dart'; import 'package:flutter\_pdfview/flutter\_pdfview.dart'; import 'package:get/get.dart'; import 'package:lab4/screens/mainScreen.dart';  import '../controllers/registrationController.dart';  class RegisterPage extends StatefulWidget {  const RegisterPage({super.key});   @override  State<RegisterPage> createState() => \_RegisterPageState(); }  class \_RegisterPageState extends State<RegisterPage> {  final FormController formController = Get.put(FormController());   @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  iconTheme: IconThemeData(color: Colors.*white*),  title: Text(  "Register",  style: TextStyle(color: Colors.*white*),  ),  backgroundColor: Color(0xff1976D2),  ),  body: SingleChildScrollView(  child: Column(  children: [  Container(  child: Column(  mainAxisAlignment: MainAxisAlignment.start,  crossAxisAlignment: CrossAxisAlignment.start,  spacing: 10,  children: [  Container(  padding:  EdgeInsets.symmetric(vertical: Get.height \* 0.015),  child: Text(  "Details About you",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  )),   //Name  Container(  width: Get.width \* 0.95,  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*,  ),  ),  ),  ),  ],  ),  ),  //Email  Container(  width: Get.width \* 0.95,  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*,  ),  ),  ),  ),  ],  ),  ),  //phno  Container(  width: Get.width \* 0.95,  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*,  ),  ),  ),  ),  ],  ),  ),  //UCID  Container(  width: Get.width \* 0.95,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("UCID"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController  .validateUCID(formController.ucid.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.ucid.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  onChanged: (value) {  formController.ucid.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateUCID(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your UCID',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color:  formController.ucidError.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color:  formController.ucidError.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.ucidError.value.isEmpty  ? null  : formController.ucidError.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  SizedBox(  height: Get.height \* 0.015,  )  ],  ),  ),  Divider(),  Container(  child: Column(  mainAxisAlignment: MainAxisAlignment.start,  crossAxisAlignment: CrossAxisAlignment.center,  spacing: 10,  children: [  Container(  padding: EdgeInsets.symmetric(  vertical: Get.height \* 0.015,  horizontal: Get.width \* 0.025),  alignment: Alignment.*centerLeft*,  child: Text(  "GRADUATION",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  )),   //College  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("College Name"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateCollegeNameGraduation(  formController  .collegeNameGraduation.value.text);  }  },  child: Obx(  () => TextFormField(  controller:  formController.collegeNameGraduation.value,  onChanged: (value) {  formController.collegeNameGraduation.value  .value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController  .validateCollegeNameGraduation(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter College Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorGraduation  .value  .isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorGraduation  .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  .collegeNameErrorGraduation  .value  .isEmpty  ? null  : formController  .collegeNameErrorGraduation.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),   Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //SEM1  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM1 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem1(  formController.sem1.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.sem1.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  onChanged: (value) {  formController.sem1.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateTotalMarksHSC(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM1 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem1Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem1Error.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.sem1Error.value.isEmpty  ? null  : formController.sem1Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //SEM2  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM2 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem2(  formController.sem2.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  controller: formController.sem2.value,  onChanged: (value) {  formController.sem2.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateSem2(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM2 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem2Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem2Error.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.sem2Error.value.isEmpty  ? null  : formController.sem2Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //SEM3  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM3 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem3(  formController.sem3.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  controller: formController.sem3.value,  onChanged: (value) {  formController.sem3.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateSem3(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM3 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem3Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem3Error.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.sem3Error.value.isEmpty  ? null  : formController.sem3Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //SEM4  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM4 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem4(  formController.sem4.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  controller: formController.sem4.value,  onChanged: (value) {  formController.sem4.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateSem4(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM4 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem4Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem4Error.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.sem4Error.value.isEmpty  ? null  : formController.sem4Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //SEM5  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM5 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem5(  formController.sem5.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  controller: formController.sem5.value,  onChanged: (value) {  formController.sem5.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateSem5(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM5 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem5Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem5Error.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.sem5Error.value.isEmpty  ? null  : formController.sem5Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //SEM6  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("SEM6 CGPA"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateSem6(  formController.sem6.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],  controller: formController.sem6.value,  onChanged: (value) {  formController.sem6.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateSem6(value);  formController.validateCGPA();  formController  .calculatePercentageGraduation();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your SEM6 CGPA',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem6Error.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .sem6Error.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.sem6Error.value.isEmpty  ? null  : formController.sem6Error.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //CGPA  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("CGPA"),  Obx(  () => TextFormField(  readOnly: true,  controller: formController.CGPA.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],   decoration: InputDecoration(  hintText: 'Your CGPA',  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: Colors.*black*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color:Colors.*black*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),   filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ],  ),  ),  //percentage  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Percentage"),  Obx(  () => TextFormField(  readOnly: true,  controller:  formController.percentageGraduation.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.allow(  RegExp(r'^\d\*\.?\d\*')),  ],   decoration: InputDecoration(  hintText: 'Your Percentage',  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: Colors.*black*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: Colors.*black*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),   filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ],  ),  ),  ],  ),  SizedBox(  height: Get.height \* 0.015,  )  ],  ),  ),  Divider(),  Container(  child: Column(  mainAxisAlignment: MainAxisAlignment.start,  crossAxisAlignment: CrossAxisAlignment.center,  spacing: 10,  children: [  Container(  padding: EdgeInsets.symmetric(  vertical: Get.height \* 0.015,  horizontal: Get.width \* 0.025),  alignment: Alignment.*centerLeft*,  child: Text(  "HSC",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  )),   //College  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("College Name"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateCollegeNameHSC(  formController.collegeNameHSC.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.collegeNameHSC.value,  onChanged: (value) {  formController.collegeNameHSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateCollegeNameHSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter College Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorHSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorHSC.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  .collegeNameErrorHSC.value.isEmpty  ? null  : formController.collegeNameErrorHSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //Year of passing  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Year of Passing"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateYearOfPassingHSC(  formController.yearOfPassingHSC.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.yearOfPassingHSC.value,  onChanged: (value) {  formController.yearOfPassingHSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateYearOfPassingHSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter Year of Passing',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .yearOfPassingErrorHSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .yearOfPassingErrorHSC.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  .yearOfPassingErrorHSC.value.isEmpty  ? null  : formController  .yearOfPassingErrorHSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),   Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //marks obtained  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Marks Obtained"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateMarksObtainedHSC(  formController  .marksObtainedHSC.value.text);  formController.calculatePercentageHSC();  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  controller:  formController.marksObtainedHSC.value,  onChanged: (value) {  formController.marksObtainedHSC.value  .value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController  .validateMarksObtainedHSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter Marks Obtained',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .marksObtainedErrorHSC  .value  .isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .marksObtainedErrorHSC  .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  .marksObtainedErrorHSC.value.isEmpty  ? null  : formController  .marksObtainedErrorHSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //Total Marks  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Total Marks"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateTotalMarksHSC(  formController.totalMarksHSC.value.text);  formController.calculatePercentageHSC();  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  controller:  formController.totalMarksHSC.value,  onChanged: (value) {  formController.totalMarksHSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateTotalMarksHSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your Total Marks',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.totalMarksErrorHSC  .value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.totalMarksErrorHSC  .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  .totalMarksErrorHSC.value.isEmpty  ? null  : formController  .totalMarksErrorHSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   //Percentage  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Percentage"),  Obx(  () => TextFormField(  readOnly: true,  controller: formController.percentageHSC.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  onChanged: (value) {  formController.percentageHSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Your Percentage',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .percentageErrorHSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .percentageErrorHSC.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  .percentageErrorHSC.value.isEmpty  ? null  : formController.percentageErrorHSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ],  ),  ),  SizedBox(  height: Get.height \* 0.015,  )  ],  ),  ),  Divider(),  Container(  child: Column(  mainAxisAlignment: MainAxisAlignment.start,  crossAxisAlignment: CrossAxisAlignment.center,  spacing: 10,  children: [  Container(  padding: EdgeInsets.symmetric(  vertical: Get.height \* 0.015,  horizontal: Get.width \* 0.025),  alignment: Alignment.*centerLeft*,  child: Text(  "SSC",  style:  TextStyle(fontWeight: FontWeight.*w800*, fontSize: 20),  ),  ),   //College  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("School Name"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateCollegeNameSSC(  formController.collegeNameSSC.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.collegeNameSSC.value,  onChanged: (value) {  formController.collegeNameSSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateCollegeNameSSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter College Name',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorSSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .collegeNameErrorSSC.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  .collegeNameErrorSSC.value.isEmpty  ? null  : formController.collegeNameErrorSSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //Year of passing  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Year of Passing"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateYearOfPassingSSC(  formController.yearOfPassingSSC.value.text);  }  },  child: Obx(  () => TextFormField(  controller: formController.yearOfPassingSSC.value,  onChanged: (value) {  formController.yearOfPassingSSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateYearOfPassingSSC(value);  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter Year of Passing',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .yearOfPassingErrorSSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .yearOfPassingErrorSSC.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  .yearOfPassingErrorSSC.value.isEmpty  ? null  : formController  .yearOfPassingErrorSSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  //marks obtained  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Marks Obtained"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateMarksObtainedSSC(  formController  .marksObtainedSSC.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  controller:  formController.marksObtainedSSC.value,  onChanged: (value) {  formController.marksObtainedSSC.value  .value = TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController  .validateMarksObtainedSSC(value);  formController.calculatePercentageSSC();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter Marks Obtained',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .marksObtainedErrorSSC  .value  .isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .marksObtainedErrorSSC  .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  .marksObtainedErrorSSC.value.isEmpty  ? null  : formController  .marksObtainedErrorSSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  //Total Marks  Container(  width: Get.width \* 0.45,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Total Marks"),  Focus(  onFocusChange: (hasFocus) {  if (!hasFocus) {  formController.validateTotalMarksSSC(  formController.totalMarksSSC.value.text);  }  },  child: Obx(  () => TextFormField(  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  controller:  formController.totalMarksSSC.value,  onChanged: (value) {  formController.totalMarksSSC.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  formController.validateTotalMarksSSC(value);  formController.calculatePercentageSSC();  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Please Enter your Total Marks',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.totalMarksErrorSSC  .value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController.totalMarksErrorSSC  .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  .totalMarksErrorSSC.value.isEmpty  ? null  : formController  .totalMarksErrorSSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ),  ],  ),  ),  ],  ),   //Percentage  Container(  width: Get.width \* 0.94,  child: Column(  spacing: Get.height \* 0.010,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text("Percentage"),  Obx(  () => TextFormField(  readOnly: true,  controller: formController.percentageSSC.value,  keyboardType: TextInputType.*number*,  inputFormatters: [  FilteringTextInputFormatter.*digitsOnly* ],  onChanged: (value) {  formController.phoneNumber.value.value =  TextEditingValue(  text: value,  selection: TextSelection.collapsed(  offset: value.length),  );  },  decoration: InputDecoration(  contentPadding: EdgeInsets.symmetric(  horizontal: 16, vertical: 12),  labelText: 'Your Percentage',  enabledBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .percentageErrorSSC.value.isEmpty  ? Colors.*black* : Colors.*red*,  width: 2,  ),  borderRadius: BorderRadius.circular(8),  ),  focusedBorder: OutlineInputBorder(  borderSide: BorderSide(  color: formController  .percentageErrorSSC.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  .percentageErrorSSC.value.isEmpty  ? null  : formController.percentageErrorSSC.value,  errorStyle: TextStyle(color: Colors.*red*),  filled: true,  fillColor: Colors.*white*,  ),  ),  ),  ],  ),  ),   SizedBox(  height: Get.height \* 0.015,  )  ],  ),  ),  Divider(),  Container(  child: Column(  crossAxisAlignment: CrossAxisAlignment.center,  children: [  Container(  alignment: Alignment.*centerLeft*,  child: Container(  padding: EdgeInsets.symmetric(  vertical: Get.height \* 0.015,  horizontal: Get.width \* 0.025),  alignment: Alignment.*centerLeft*,  child: Text(  "Your Image",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  ),  )),  SizedBox(height: Get.height \* 0.02),  GestureDetector(  onTap: () {  formController.pickProfileImage();  },  child: Container(  margin: EdgeInsets.all(10),  width: Get.width \* 0.92,  height: Get.height \* 0.04,  alignment: Alignment.*center*,  decoration: BoxDecoration(  color: Color(0xff1976D2),  borderRadius: BorderRadius.circular(8)),  child: Text(  "Pick Image",  style: TextStyle(color: Colors.*white*),  ),  ),  ),  SizedBox(height: Get.height \* 0.02),  Obx(() {  if (formController.profileImagePath.value != null) {  return CircleAvatar(  radius: 100,  backgroundImage: FileImage(  File(formController.profileImagePath.value)),  );  } else {  return Text("No profile image selected.");  }  }),  ],  ),  ),  Divider(),  Container(  child: Column(  crossAxisAlignment: CrossAxisAlignment.center,  children: [  Container(  alignment: Alignment.*centerLeft*,  child: Container(  padding: EdgeInsets.symmetric(  vertical: Get.height \* 0.015,  horizontal: Get.width \* 0.025),  alignment: Alignment.*centerLeft*,  child: Text(  "Your Resume",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  ),  )),  SizedBox(height: Get.height \* 0.02),  GestureDetector(  onTap: () {  formController.pickFile();  },  child: Container(  margin: EdgeInsets.all(10),  width: Get.width \* 0.92,  height: Get.height \* 0.04,  alignment: Alignment.*center*,  decoration: BoxDecoration(  color: Color(0xff1976D2),  borderRadius: BorderRadius.circular(8)),  child: Text(  "Pick Resume",  style: TextStyle(color: Colors.*white*),  ),  ),  ),  SizedBox(height: Get.height \* 0.02),  Obx(() {  if (formController.selectedFilePath.isNotEmpty) {  return formController.isPdf.value  ? Container(  height: Get.height \* 0.5,  child: PDFView(  filePath:  formController.selectedFilePath.value),  )  : Image.file(  File(formController.selectedFilePath.value),  height: Get.height \* 0.3,  fit: BoxFit.cover,  );  }  return Container();  }),  ],  ),  ),  GestureDetector(  onTap: () {  if (formController.isFormValid()) {  print('Valid');  formController.storeFormValues();  formController.resetForm();  Get.snackbar(  "Registered",  "You are Registered Successfully",  snackPosition: SnackPosition.TOP,  backgroundColor: Colors.*green*,  );  Get.off(MainScreen());  } else {  print('InValid');  Get.snackbar(  "Form Error",  "Please fill in all the required fields correctly.",  snackPosition: SnackPosition.TOP,  backgroundColor: Colors.*red*,  );  }  },  child: Container(  margin: EdgeInsets.all(10),  width: Get.width \* 0.92,  height: Get.height \* 0.04,  alignment: Alignment.*center*,  decoration: BoxDecoration(  color: Color(0xff1976D2),  borderRadius: BorderRadius.circular(8)),  child: Text(  "Submit",  style: TextStyle(color: Colors.*white*),  ),  ),  ),  ],  ),  ),  );  } }  **profilePage.dart**  import 'dart:io';  import 'package:flutter/material.dart'; import 'package:flutter\_pdfview/flutter\_pdfview.dart'; import 'package:get/get.dart';  import '../controllers/registrationController.dart';  class ProfilePage extends StatefulWidget {  const ProfilePage({super.key});   @override  State<ProfilePage> createState() => \_ProfilePageState(); }  class \_ProfilePageState extends State<ProfilePage> {  final FormController formController = Get.put(FormController());   @override  Widget build(BuildContext context) {  var isProfileComplete = formController.box.read("isProfileComplete");  var resumeBox = formController.box.read("resume");  var imageBox = formController.box.read("profileImage");  var isPdf = formController.box.read("isPdf");  var collegeNameGraduation =  formController.box.read("collegeNameGraduation");  var CGPA = formController.box.read("CGPA");  var collegeNameSSC = formController.box.read("collegeNameSSC");  var percentageSSC = formController.box.read("percentageSSC");  var collegeNameHSC = formController.box.read("collegeNameHSC");  var percentageHSC = formController.box.read("percentageHSC");   return Scaffold(  appBar: AppBar(  iconTheme: IconThemeData(color: Colors.*white*),  title: Text(  "Profile",  style: TextStyle(color: Colors.*white*),  ),  backgroundColor: Color(0xff1976D2),  ),  body: isProfileComplete != null  ? SingleChildScrollView(  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  // Image  Container(  padding: EdgeInsets.all(20),  alignment: Alignment.*center*,  child: (imageBox != null)  ? CircleAvatar(  radius: 100,  backgroundImage: FileImage(File(imageBox)),  )  : Text("No profile image selected."),  ),    Container(  padding: EdgeInsets.symmetric(horizontal: Get.width \* 0.025),  child: Column(  spacing: 0,  crossAxisAlignment: CrossAxisAlignment.start,  children: [  // Graduation  Text(  "Graduation",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  Row(  spacing: 15,  children: [  Icon(Icons.*school\_rounded*),  Text(collegeNameGraduation,style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18),),  ],  ),  Row(  spacing: 15,  children: [  Text(CGPA + " CGPA",style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18)),  Icon(Icons.*leaderboard\_rounded*),  ],  )  ],  ),    // HSC  Text(  "HSC",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20),  ),  Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  Row(  spacing: 15,  children: [  Icon(Icons.*school\_rounded*),  Text(collegeNameHSC,style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18)),  ],  ),  Row(  spacing: 15,  children: [  Text(percentageHSC + "%",style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18)),  Icon(Icons.*leaderboard\_rounded*),  ],  ),  ],  ),    // SSC  Text("SSC",  style: TextStyle(  fontWeight: FontWeight.*w800*, fontSize: 20)),  Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  Row(  spacing: 15,  children: [  Icon(Icons.*school\_rounded*),  Text(collegeNameSSC,style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18)),  ],  ),  Row(  spacing: 15,  children: [  Text(percentageSSC + "%",style: TextStyle(  fontWeight: FontWeight.*w400*, fontSize: 18)),  Icon(Icons.*leaderboard\_rounded*),  ],  ),      ],  ),  ],  ),  ),    // Resume    (resumeBox != null)  ?  // return  isPdf != null  ? Container(  height: Get.height \* 0.5,  child: PDFView(filePath: resumeBox),  )  : Image.file(  File(resumeBox),  height: Get.height \* 0.3,  fit: BoxFit.cover,  )  : Container(  child: Text("No Resume is selected."),  ),  ],  ),  )  : Center(  child: Text("Please Register First!"),  ),  );  } }  **companiesPage.dart**  import 'package:flutter/material.dart'; import 'package:get/get.dart';  import '../controllers/registrationController.dart';  class CompaniesPage extends StatefulWidget {  const CompaniesPage({super.key});   @override  State<CompaniesPage> createState() => \_CompaniesPageState(); }  class \_CompaniesPageState extends State<CompaniesPage> {   @override  Widget build(BuildContext context) {  final FormController formController = Get.put(FormController());  var isProfileComplete = formController.box.read("isProfileComplete");  final List<List<String>> companies = [  [  'ISS stoxx',  'Mumbai, India',  '₹60,000 - ₹75,000',  'SDE-1.',  'assets/images/ISS.jpg'  ],  [  'Nomura',  'Mumbai, India',  '₹80,000 - ₹90,000',  'SDE-1.',  'assets/images/Nomura.jpg'  ],  [  'Oracle',  'Mumbai, India',  '₹70,000 - ₹80,000',  'Associate Consultant.',  'assets/images/oracle.png'  ],  [  'Phone Pe',  'Bangalore, India',  '₹1,20,000 - ₹1,40,000',  'Senior Devops Engineer.',  'assets/images/phonePe.png'  ],  [  'Wissen Infotech',  'Bangalore, India',  '₹70,000 - ₹80,000',  'Database Engineer.',  'assets/images/Wissen.jpg'  ],  ];  return Scaffold(  appBar: AppBar(  iconTheme: IconThemeData(color: Colors.*white*),  title: Text(  "Companies",  style: TextStyle(color: Colors.*white*),  ),  backgroundColor: Color(0xff1976D2),  ),  body:  isProfileComplete != null ?  ListView.builder(  itemCount: companies.length,  itemBuilder: (context, index) {  return Card(  margin: EdgeInsets.symmetric(horizontal: 10, vertical: 5),  child: Padding(  padding: const EdgeInsets.all(10.0),  child: Row(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  // Company Image  ClipRRect(  borderRadius: BorderRadius.circular(10),  child: Image.asset(  companies[index][4],  width: 80,  height: 80,  fit: BoxFit.fill,  ),  ),  SizedBox(width: 15),  // Company Details  Expanded(  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text(  companies[index][0], // Company Name  style: TextStyle(  fontSize: 20,  fontWeight: FontWeight.*bold*,  ),  ),  SizedBox(height: 5),  Text(  'Location: ${companies[index][1]}',  style: TextStyle(color: Colors.*grey*[600]),  ),  Text(  'Salary Range: ${companies[index][2]}',  style: TextStyle(color: Color(0xffFF6600)),  ),  SizedBox(height: 5),  Text(  companies[index][3], // Short Profile  style: TextStyle(fontSize: 16),  ),  ],  ),  ),  ],  ),  ),  );  },  ):  Center(  child: Text("Please Register First!"),  ),  );  } }  **mainScreen.dart**  import 'package:flutter/material.dart'; import 'package:lab4/commonWidgets/commonDrawer.dart';  class MainScreen extends StatefulWidget {  const MainScreen({super.key});   @override  State<MainScreen> createState() => \_MainScreenState(); }  class \_MainScreenState extends State<MainScreen> {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  iconTheme: IconThemeData(color: Colors.*white*),  title: Text("TPO",style: TextStyle(color: Colors.*white*),),  backgroundColor: Color(0xff1976D2),  ),  body: Center(  child: Text(  "Welcome To Training and Placement Office",  textAlign: TextAlign.center,  style: TextStyle(fontSize: 32),  ),  ),  drawer: CommonDrawer(),  );  } } |
| **Screenshots** |  |
| **Question and Answers** | Answer the following Questions:   1. How to create SideDrawer Navigation?   It is implemented using the Drawer widget inside the Scaffold. The CommonDrawer contains a ListView with ListTile items for navigation, using Get.to() for page transitions.   1. How did you use 60-30-10 rule in your application?   The primary color Color(0xff1976D2) (60%) is used for the AppBar and Drawer, white (30%) for text, and Color(0xffFF6600) (10%) for Salary.   1. Which new elements did you use for creating UI components?   PDFView |
| **Conclusion** | The TPO app will help students register their academic details, calculate percentages, and upload resumes. It will have a side navigation drawer for easy access to features. Students can view their details, uploaded resumes, and company information like name, location, and salary range. This app will make the placement process simple and accessible for students and coordinators. |