

PG Admission Portal

Overview:

This application is built to help customers to get PG Admission online.

Users of the System:

1. Admin
2. User

Functional Requirements:

- Build a portal that enables customers can get PG Admission online.
- The customers can add/edit/view/delete admission.
- The admin can add/edit/delete/view courses.
- The admin can add/edit/delete/view institutes.
- The admin can add/edit/delete/view students.
- Customer can provide reviews.

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- ☐ Have appropriate filters for search.
- ☐ Email integration for intimate customers.
- ☐ Multi-factor authentication for the sign-in process
- ☐ Payment Gateway (if required)

Output/ Post Condition:

- ☐ Records Persisted in Success & Failure Collections
- ☐ Standalone application / Deployed in an app Container

Non-Functional Requirements:

Security	<ul style="list-style-type: none">● App Platform – Username/Password-Based Credentials● Sensitive data has to be categorized and stored in a secure manner● Secure connection for transmission of any data
Performance	<ul style="list-style-type: none">● Peak Load Performance (during Festival days, National holidays etc.)

	<ul style="list-style-type: none"> • Admin application < 2 Sec • Non-Peak Load Performance • Appointment Application< 2 Sec • Admin Application < 2 Sec
Availability	<ul style="list-style-type: none"> • 99.99 % Availability
Standard Features	<ul style="list-style-type: none"> • Scalability • Maintainability • Usability • Availability • Failover
Logging & Auditing	<ul style="list-style-type: none"> • The system should support logging(app/web/DB) & auditing at all levels
Monitoring	<ul style="list-style-type: none"> • Should be able to monitor via as-is enterprise monitoring tools
Cloud	<ul style="list-style-type: none"> • The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	<ul style="list-style-type: none"> • All latest browsers

Technology Stack

Front End	Angular 10+/ React 16+ Material Design Bootstrap / Bulma
Server Side	Spring Boot / .NET Web API/ Node
Database	MySQL or Oracle or MSSQL

Platform Prerequisites (Do's and Don'ts):

1. The angular app or react app should run in port 8081.
 2. Spring boot app and .NET app should run in port 8080. Key points to remember:
 1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
 2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
 3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
 4. Adhere strictly to the endpoints given below.
-

5. This is a basic SRS document, so understand them well and please feel free to explore and come with new ideas.

Application assumptions:

1. The login page should be the first page rendered when the application loads.
2. Manual routing should be restricted by using Auth Guard by implementing the canActivate interface. For example, if the user enters as <http://localhost:8080/signup> or <http://localhost:8080/home> the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile number validation should be performed.

Project Tasks:

API Endpoints:

Admin Side:

Action	URL	Method	Response
Admin Login	/admin/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/admin/signup	POST-Sends Admin Model data	Admin added
Add Courses	/admin/addCourse	POST – Sends Course data	Courses added
View Courses	/admin/courses/id?id=1	GET – Fetches course data	Retrieve the courses
View All Courses	/admin/courses	GET – Fetches course data	Retrieve all the courses
Delete Courses	/admin/deleteCourse	DELETE – Send course Id	Course deleted

Edit Courses	/admin/editCourse/{courseId}	PUT – Send course Id	Course edited
Add Institutes	/admin/addInstitute	POST – Sends Institute data	Institute added
View Institutes	/admin/institute/id?id=1	GET – Fetches course data	Retrieve the institute
View Institutes	/admin/institute	GET – Fetches course data	Retrieve all the institute
Edit Institutes	/admin/editInstitute/{instituteId}	PUT – Sends institute Id	Institute edited
Delete Institutes	/admin/deleteInstitutes	DELETE – Sends Institute Id	Institute deleted
Add Student	/admin/addStudent	POST – Sends student data	Student added
View Student	/admin/student/id?id=1	GET – Fetches student details	Retrieve all the student details
View Student	/admin/student	GET – Fetches student details	Retrieve all the student details
Edit Student	/admin/editStudent/{studentId}	PUT – sends student id	Student details edited
Delete Student	/admin/deleteStudent/{studentId}	DELETE – sends student id	Student details deleted

User Side:

Action	URL	Method	Response
User Login	/user/login	POST-Sends email ID and password	Return True/False
Admin SignUp	/user/signup	POST-Sends User Model data	User added

Add Admission	/admin /addAdmission	POST – Sends admission data	Course enrolled
View All Admission	/admin/admission	GET – Fetches admission data	Retrieve all the admission details
View Admission	/admin/admission/id?id=1	GET – Fetches admission data	Retrieve the admission details
Edit Admission	/admin/editAdmission/{enrollmentId}	PUT – Sends admissionId	Admission details edited
Delete Admission	/admin /deleteAdmission/{enrollmentId}	DELETE – Sends admissionId	Admission details deleted
View Status	/admin /viewStatus	GET – Fetches Admission status	Admission Application Status

Frontend:

Customer:

1. Auth: Design an auth component (Name the component as auth for angular app whereas Auth for react app. Once the component is created in react app, name the jsx file as same as component name i.e Auth.jsx file) where the customer can authenticate login and signup credentials
2. Signup: Design a signup page component (Name the component as signup for angular app whereas Signup for react app. Once the component is created in react app, name the jsx file as same as component name i.e Signup.jsx file) inside the auth where the new customer has options to sign up by providing their basic details.
 - a. Ids: Refer to the screenshot below for the id details.
 - b. Output screenshot:

Register

Enter admin/user

id=admin/user

Enter email

id= email

Enter Username

id=username

Enter Mobilenumber

id=mobileNumber

Password

id= password

Confirm Password

id= confirmPassword

Submit

id= submitButton

Already a user? Login

id=signinLink

3. Login: Design a login page component (Name the component as login for angular app whereas Login for react app. Once the component is created in react app, name the jsx file as same as component name i.e Login.jsx file) inside the auth where the existing customer can log in using the registered email id and password.
 - a. Ids: Refer to the screenshot below for the id details.
 - b. Output Screenshot:

Login

id= email

Enter email

Enter Password

id= password

id= loginButton

Login

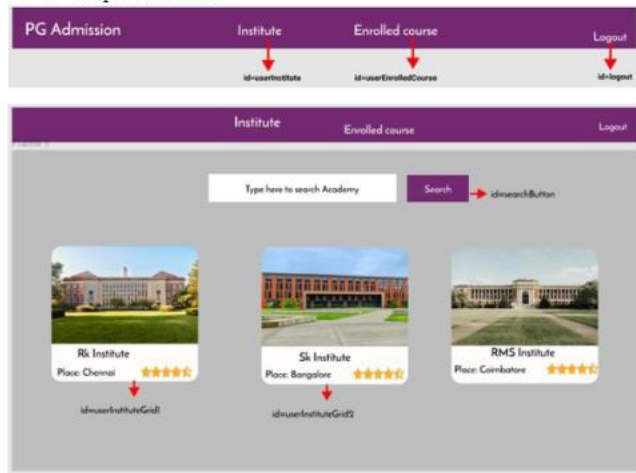
New User/admin? Sign Up

id= signupLink

4. View Academy: Design a component (Name the component as viewacademy for angular app whereas Viewacademy for react app. Once the component is created in react app, name the jsx file as same as component name i.e Viewacademy.jsx file)

a. Ids: Refer to the screenshot below for the id details.

b. Output Screenshot:



Page 2 Institute Enrolled course Logout

Type here to search course Search id=searchCourse

Course name : M.E(VLSI) Number of Students : 222 id=userCourseGrid

Course Duration : 2 years Course Description : yyyyy

Course Available Timings : 9am to 4pm id=courseGrid

Enroll course

Course name : M.COM(CA) Number of Students : 122 id=userCourseGrid2

Course Duration : 2 years Course Description : yyyyy

Course Available Timings : 9am to 4pm id=courseGrid2

Enroll course

Page 3 Institute Enrolled course Logout

enter your first name id=firstName

enter your last name id=lastName

enter male or female id=gender

enter your father name id=fatherName

enter phone number id=phoneNumber1

enter alternate number id=phoneNumber2

enter your mother name id=motherName

enter email id id=emailId

enter you age id=age

Enter SSLC/HSC marks id=enterSSLC/HSCMarks

Address information id=houseNo

House No : id=houseNo

Street Name : id=streetName

Area Name : id=areaName

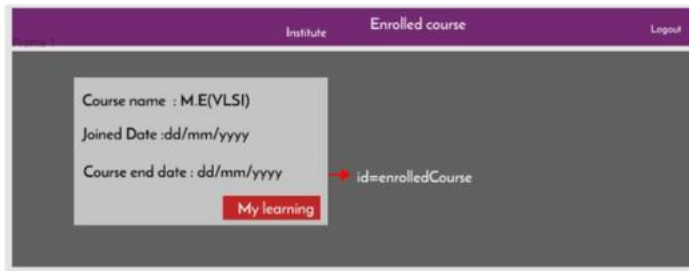
Pincode : id=pincode

State : id=state

Nationality : id=nationality

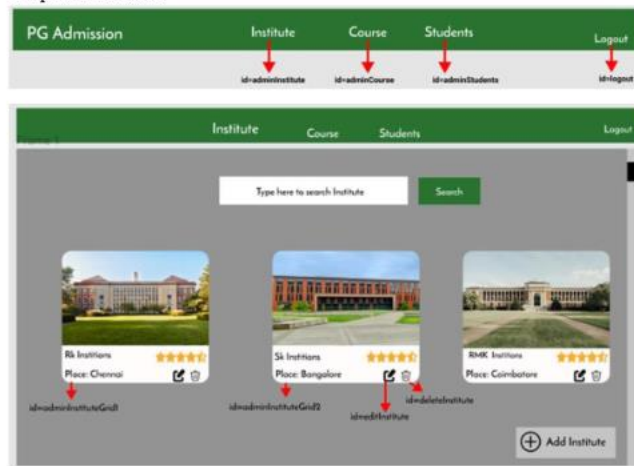
id=enrollNowButton Enroll now

5. Enrolled Course: Design a component (Name the component as enrolledcourse for angular app whereas Enrolledcourse for react app. Once the component is created in react app, name the jsx file as same as component name i.e, Enrolledcourse.jsx file)
 - a. Ids: Refer to the screenshot below for the id details.
 - b. Output Screenshot:

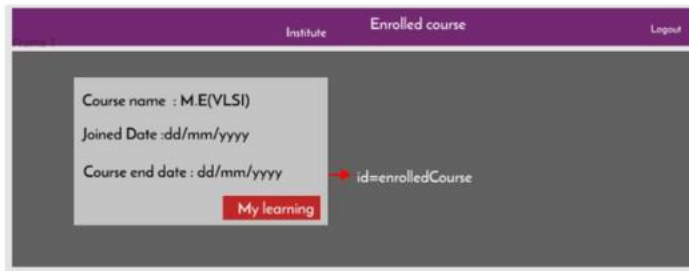


Admin:

6. Admin Academy: Design a component (Name the component as adminacademy for angular app whereas Adminacademy for react app. Once the component is created in react app, name the jsx file as same as component name i.e, Adminacademy.jsx file)
 - a. Ids: Refer to the screenshot below for the id details.
 - b. Output Screenshot

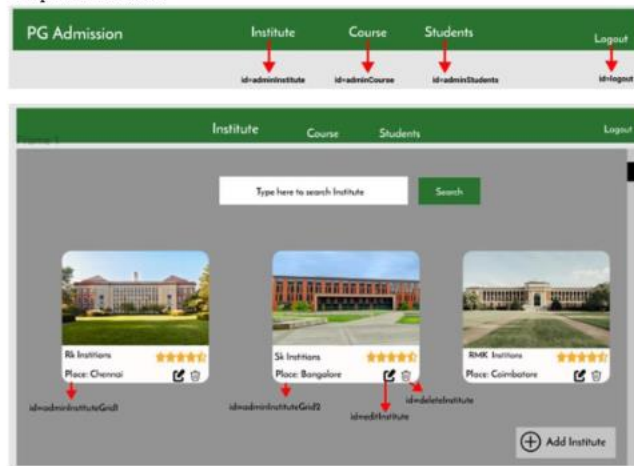


Add:

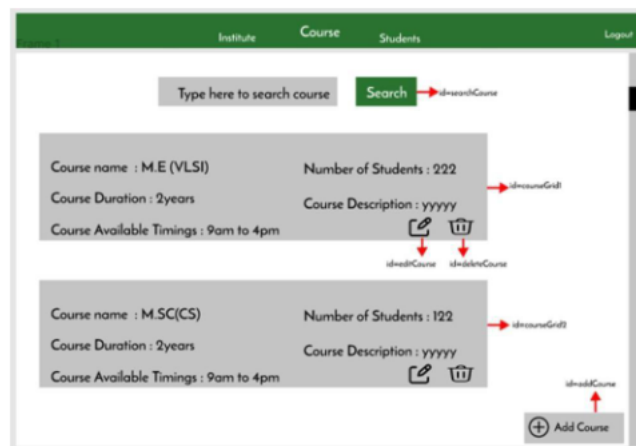


Admin:

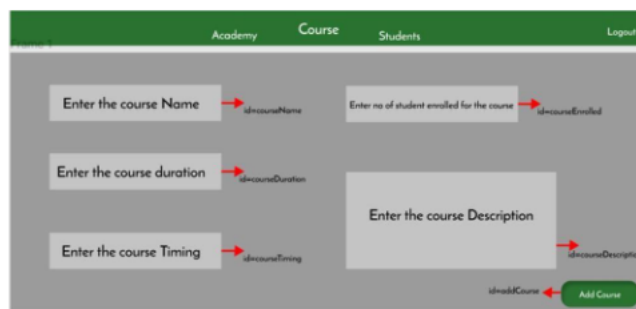
6. Admin Academy: Design a component (Name the component as adminacademy for angular app whereas Adminacademy for react app. Once the component is created in react app, name the jsx file as same as component name i.e, Adminacademy.jsx file)
 - a. Ids: Refer to the screenshot below for the id details.
 - b. Output Screenshot



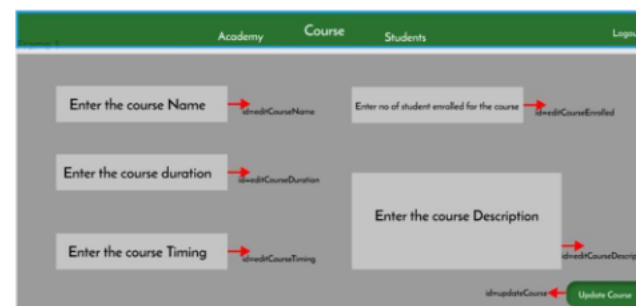
Add:



Add:










Edit:



8. Admin Students: Design a component (Name the component as adminstudent for angular app whereas Adminstudent for react app. Once the component is created in react app, name the jsx file as same as component name i.e, Adminstudent.jsx file)

a. Ids: Refer to the screenshot below for the id details.

b. Output Screenshot:

Academy Course Students Logout				
type here student id to search				Search
Student ID	Student name	Enrolled Course	Mobile number	Actions
1232213	Arul	Boxing drills	3393992020	 
2321232	Kavya	Strength training	3393992020	 
2321233	Manoj	Conditioning work	3393992020	 
				 Add Student

Add:

Academy Course Students Logout

enter your first name id=firstName

enter your last name id=lastName

enter male or female id=gender

enter your father name id=fatherName

enter phone number id=phoneNumber

enter alternate number id=altPhoneNumber

enter your mother name id=motherName

enter email id id=emailId

enter your age id=age

Address information id=houseNo

House No : id=houseNo

Street Name : id=streetName

Area Name : id=areaName

Pincode : id=pincode

State : id=state

Nationality : id=nationality

Add Student

Edit:

Academy Course Students Logout

enter your first name id=firstName

enter your last name id=lastName

enter male or female id=gender

enter your father name id=fatherName

enter phone number id=phoneNumber

enter alternate number id=altPhoneNumber

enter your mother name id=motherName

enter email id id=emailId

enter your age id=age

Address information id=houseNo

House No : id=houseNo

Street Name : id=streetName

Area Name : id=areaName

Pincode : id=pincode

State : id=state

Nationality : id=nationality

Update Student

Backend:

Functional Requirements:

Create 4 folders inside the WORKSPACE/springapp/src/main/java/com/examly /springapp

1. controller
2. model
3. repository
4. service

For example:

Inside the controller, create a Java file named "AdmissionController.java"

Inside the model, create a Java file named "Courses.java"

Create variables for the model class and getters and setters and constructors for the corresponding variables.

Inside the repository, create a Java file named "CoursesRepository.java"

Inside the service, create a Java file named "CoursesService.java"

Class and Method description:

Model Layer:

1. UserModel: This class stores the user type (admin or the customer) and all user information.
 - a. Attributes:
 - i. email: String
 - ii. password: String
 - iii. username: String iv. mobileNumber: String
 - v. userRole: String
 2. LoginModel: This class contains the email and password of the user.
 - a. Attributes:
 - i. email: String
 - ii. password: String
 3. AdminModel: This class stores the details of the admin.
 - a. Attributes:
 - i. email:String
 - ii. password:String
 - iii. mobileNumber:String
-

iv. userRole:String

4. CourseModel: This class stores the details of the course

a. Attributes:

i. courseId: int ii. courseName: String iii. courseDescription: String iv.
courseDuration: int

5. InstituteModel: This class stores the details of the Institute or College

a. Attributes:

i. instituteId: int ii. instituteName: String iii. instituteDescription: String iv.
instituteAddress: String

v. mobile: String vi.
email: String

6. Student Model: This class stores the details of the students.

a. Attributes:

i. studentId: int ii. studentName: String iii. studentDOB: Date iv. address:
string

v. mobile: String

vi. SSLC: int vii.

HSC: int

viii. Diploma: int ix.

eligibility: string

6. Admission Model: This class stores the details of the admission.

a. Attributes:

admissionId: Int

coursesId: Int

InstituteId: Int

Status: String

Student: student

Controller Layer:

1. AuthController: This class control the user /admin signup and signin

a. Methods:

i. isUserPresent(LoginModel data): This method helps to check whether the
user present or not and check the email and password are correct and return
the boolean value.

- ii. `isAdminPresent(LoginModel data)`: This method helps to check whether the admin present or not and check the email and password are correct and return the boolean value.
 - iii. `saveUser(UserModel user)`: This method helps to save the user data in the database.
 - iv. `saveAdmin(UserModel user)`: This method helps to save the admin data in the database.
 - 2. **AdmissionController**: This class helps to add/edit/view/delete admission process.
 - a. Methods:
 - i. `addAdmission(Admission a)`: This method adds new admission.
 - ii. `editAdmission(int admissionId)`: This method helps to edit admission details
 - iii. `viewAdmission(int admissionId)`: This method helps to view the admission details
 - iv. `deleteAdmission(int admissionId)`: This method helps to delete the admission
 - v. `ViewStatus(int admissionId)`: This method helps to view the status of the admission.
 - vi. `viewAllAdmission()`: This method helps to view the admission details
 - 3. **AdminController**: This class helps to add/edit/view/delete various details necessary with admission process.
 - a. Methods:
 - i. `addStudent(StudentModel student)`: This method helps to add student.
 - ii. `viewStudent(int studentId)`: This method helps to view student.
 - iii. `editStudent(int studentId)`: This method helps to edit student.
 - iv. `deleteStudent(int studentId)`: This method helps to delete student.
 - v. `addCourse(CourseModel course)`: This method helps to add course.
 - vi. `editCourse(int courseId)`: This method helps to edit course.
 - vii. `deleteCourse(int courseId)`: This method helps to delete course.
 - viii. `viewCourse(int courseId)`: This method helps to view course.
 - ix. `addInstitute(int instituteId)`: This method helps to add institute.
 - x. `editInstitute(int instituteId)`: This method helps to edit institute.
 - xi. `deleteInstitute(int instituteId)`: This method helps to delete the institute
 - xii. `ViewInstitute(int instituteId)`: This method helps to view the institute details
 - xiii. `viewAllStudent()`: This method helps to view all student
 - xiv. `viewAllCourses()`: This method helps to view all course.
-

xv. viewAllInstitute (): This method helps to view all institue.
