**British English Certification (BEC) Enquiry Management System**

**Problem Statement:**  
The BEC Enquiry Management System revolutionizes certification program interaction. With a user-friendly platform, candidates can effortlessly submit enquiries, access comprehensive details on the BEC certification program, and streamline registration. Real-time responses from dedicated support staff bridge communication gaps, ensuring timely and accurate information. This innovative system enhances the candidate experience, mitigating frustration, and facilitating informed decision-making

**Users of the System:**

**Authentication and Authorization : Role based authentication.**

**Role:**

1.     Admin

2.     Student

Admin should have permission for CRUD operation.

Role based Menu Options

Admin:

* Register
* Login
* Post Course details
* Edit Course Details
* Delete Course Details
* View All Courses
* View All Enquiries
* Change Status of Enquiry
* View Payment History
* Logout

Student:

* + Register
  + Login
  + View All Courses
  + Check the Status of Enquiry
  + Delete Enquiry
  + Make Payment for the course
  + View Payment History
  + Logout

**Functional Requirements:**

1**. Enquiry Submission**: The application allows prospective candidates to submit enquiries regarding the BEC certification course, including questions about the course structure, content, examination details, and registration procedures.

2**. Course Information**: Detailed information about the BEC certification program, including course modules, examination format, fees, and available test dates, is readily accessible to prospective candidates.

3. **User Registration:** Candidates can create accounts, enabling them to track their enquiries, monitor the status of their registration, and access additional course-related resources.

4**.  Real-time Enquiry Response:** Dedicated support staff can promptly respond to enquiries, providing detailed responses to candidate questions and concerns.

5.  **Payment Process**: The student can pay the exam fees, and other payment in online.

**Non-Functional Requirements:**

1. Security: The system must implement robust security measures to protect user data, including user authentication, secure data storage, and encrypted data transmission
2. Scalability: The system should be designed to handle an increasing number of course listings, enquiries, payments, and users.
3. Usability: The user interface should be intuitive and user-friendly, with responsive design for mobile and desktop users.
4. Availability: The system should be available 24/7 with minimal downtime for maintenance.
5. Logging and Auditing: Support logging and auditing of system activities for monitoring and troubleshooting.

**Application Flow:**

**Student side:**

The application flow for the portal begins with user registration, where prospective students create accounts by providing personal information. Upon logging in, users access the user dashboard and view the available courses with payment details. The student can enquire about the course timing, course details, syllabus doubts etc.

The student can do the payment and view all the payment history.

**Admin side:**

The administrative flow within the portal begins with administrators accessing the admin dashboard, providing a comprehensive overview of courses offered with payment details. The admin can view the list of student’s profiles.

The admin can view the enquiries submitted by the students and change the enquiry change status.

The admin can view the payment history.

**Abstract**

The British English Certification (BEC) Enquiry Management System is a specialized application crafted to enhance the efficiency of managing enquiries, registrations, and information related to the BEC certification course. Tailored to meet the specific needs of potential candidates, this system provides a seamless platform for submitting enquiries, accessing comprehensive course details, and facilitating the payment process for the BEC certification program. By leveraging technology, the BEC Enquiry Management System aims to streamline communication, improve information accessibility, and enhance the overall experience for candidates interested in pursuing the BEC certification.

**Modules of the Application:**

**ADMIN:**

Ø Register

Ø Login

Ø Dashboard

·      Add course details

·      View, edit and delete course details

·      View enquiry details

·      View Payment details

**STUDENT:**

Ø Register

Ø Login

Ø Dashboard

·      View course details

·      Add enquiry details

·      View enquiry details

- Make Payment

**Technology Stack**

**Front End**

React,HTML,CSS

**Back End**

Java, Spring Boot, MySQL Database

**Authentication**

JWT for User Authentication

**Application assumptions**:

1.    The login page should be the first page rendered when the application loads.

2.    Unless logged into the system, the user cannot navigate to any other pages.

3.    Logging out must again redirect to the login page.

4.    Design forgot password and forgot email buttons in login page.

**Validation**

**Client-Side Validation:**

Implement client-side validation using HTML5 attributes and JavaScript to validate user input before making API requests.

Provide immediate feedback to users for invalid input, such as displaying error messages near the input fields.

1. Basic email validation should be performed.

2. Basic mobile number validation should be performed.

3. Basic password should be performed

**Server-Side Validation:**

Validate user input and API responses to prevent unexpected or malicious data from affecting the application.

Return appropriate validation error messages to the user interface for any validation failures.

**Exception Handling**

Implement exception handling mechanisms in the controllers to gracefully handle errors and exceptions.

Define custom exception classes for different error scenarios, such as API communication errors or database errors.

Log exceptions for debugging purposes while presenting user-friendly error messages to users. Record all the exceptions and errors handled store in separate table “**ErrorLogs**”.

**Error Pages:**

Create custom error pages for different HTTP status codes (e.g., **404** Not Found, **500** Internal Server Error) to provide a consistent and user-friendly error experience.

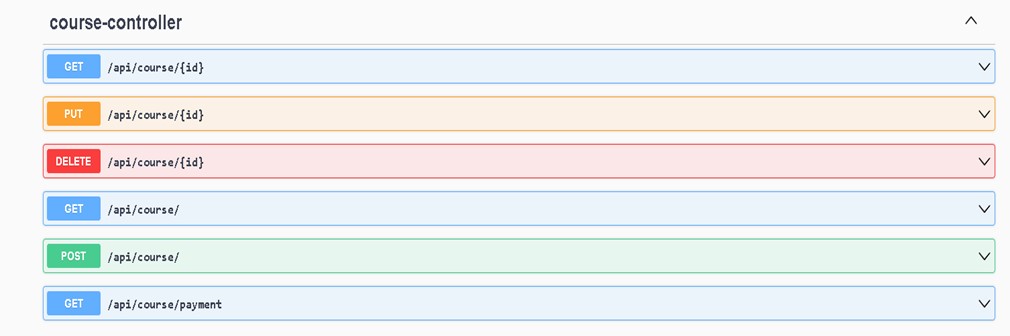
Ensure that error pages contain helpful information and guidance for users.

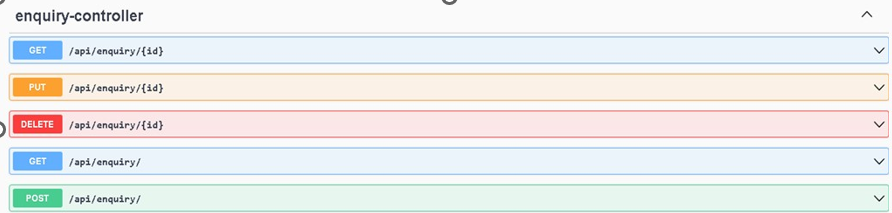
Thus, create a reliable and user-friendly web application that not only meets user expectations but also provides a robust and secure experience, even when faced with unexpected situations.

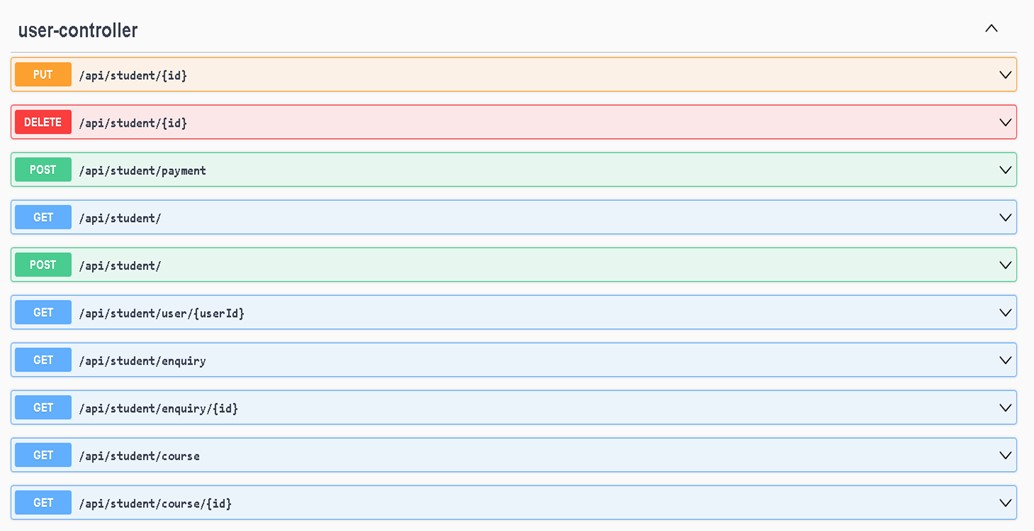
**Project Tasks:**

1. API Endpoints:









**Additional End Points:**

Admin – Can view all payment details.

URL - /admin/payment/ - GET all payment details.

Student – Will process the payment details.

URL - /student/payment/ - POST the payment details and return the status of the payment

**Backend**

class User

{

     Long userId;

     String email;

     String password;

     String username;

String mobilenumber;

     String userRole ;

}

class Student

{

     Long studentId;

     String studentName;

     String studentMobileNumber;

@OneToMany

     @JsonIgnore

     List<Enquiry> enquiries

  @ManyToMany

     List<Course> courses

     @OneToMany

     @JsonIgnore

     private Set<Payment> payments

     @OneToOne

     private User user;

}

class Course

{

     Long courseId;

     String courseName;

     String description;

     String duration;

     Long cost;

     @ManyToMany

     private Set<Student> students

     @OneToMany

     List<Enquiry> enquiries

}

class Enquiry

{

     Long enquiryId;

     Date enquiryDate;

     String title;

     String description;

     String email;

     String enquiryType;

     @ManyToOne

     Student student;

     @ManyToOne

     private Course course;

}  
  
  
class Payment

{

     Long paymentId;

     String status; //paid=true/false

     Double amountPaid;

     Date paymentDate;

     String modeOfPayment;

     @ManyToOne

     Student student;

}

Note:

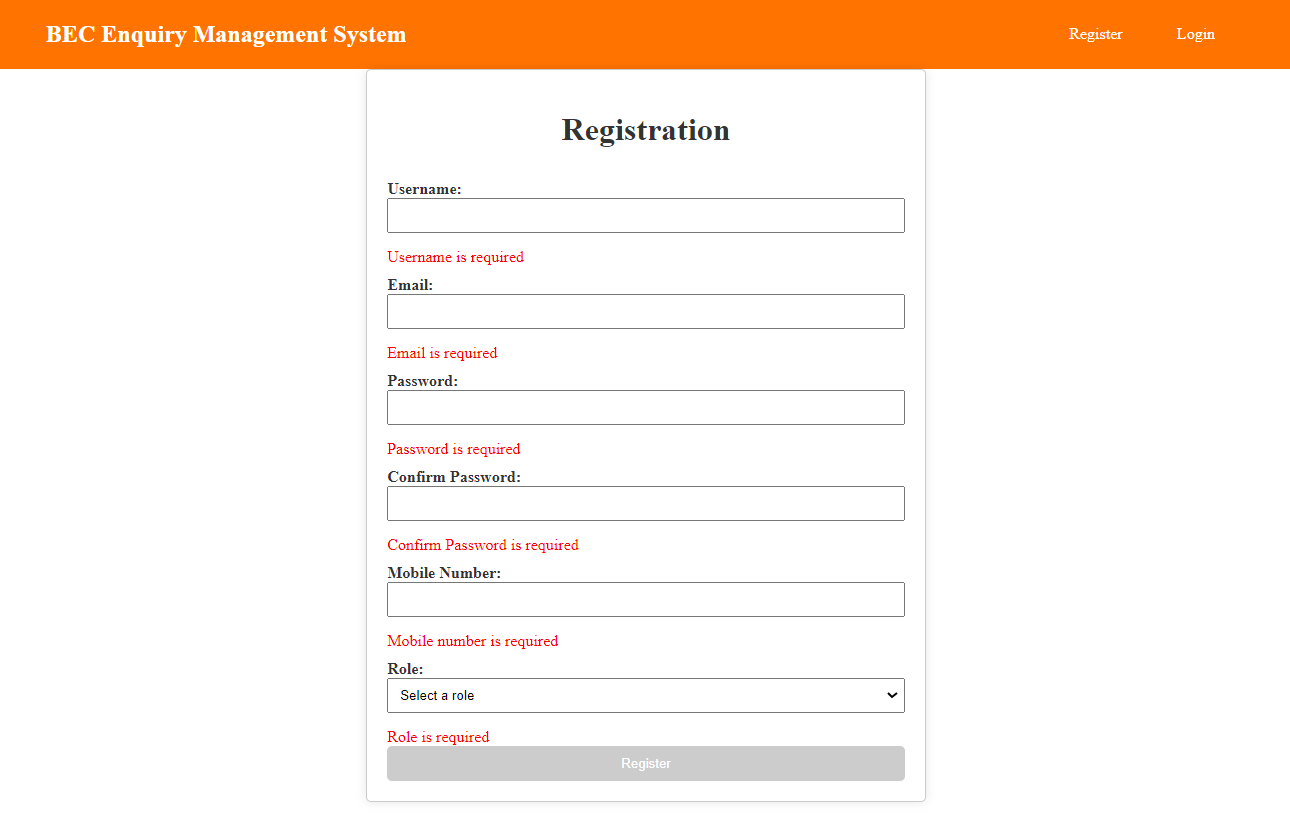
* Declare a public property **apiUrl** to store the backend URL.
* Import model files, services and components as required.

**Frontend Sample Screenshots:    
  
Home page For Admin**

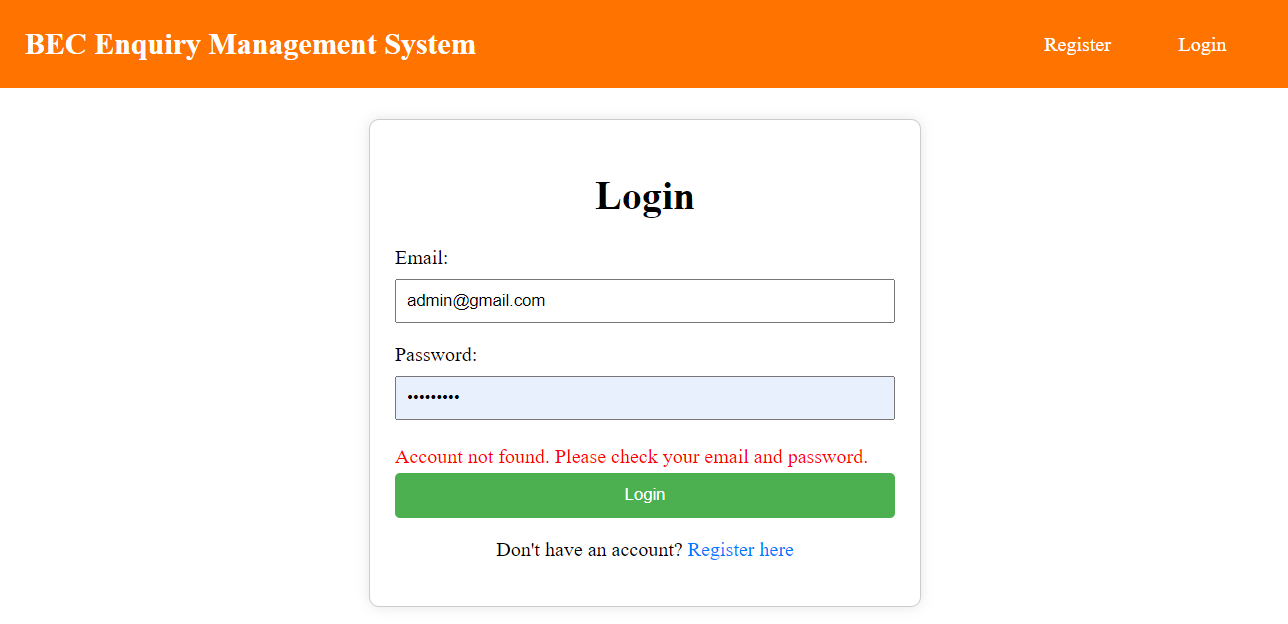
A close-up of a certificate

Description automatically generated

**Registration**: User can register the details.



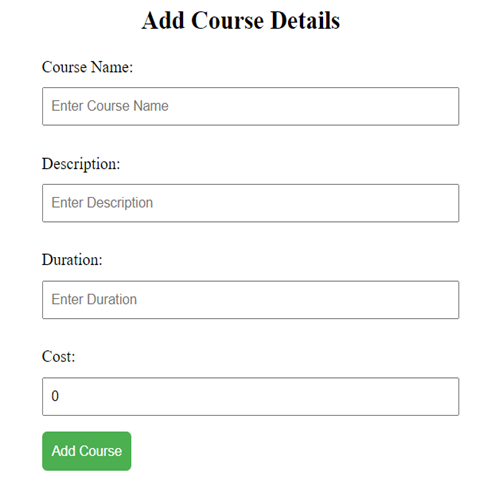
Validation error message is displayed with empty input



**Admin Side Dashboard**



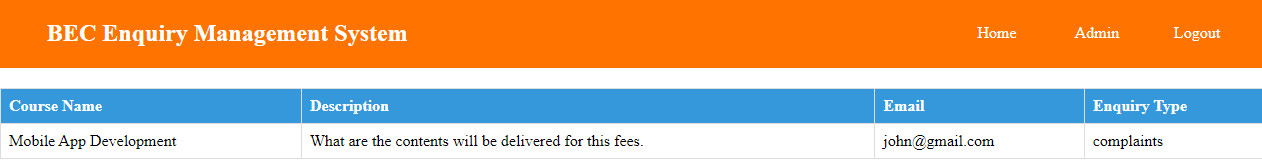
Admin can add the course details on clicking “**Add Course**” in dashboard



Admin can view all the courses and have access to edit/delete the courses.



The Admin can view the student enquiries



The Admin can view the payment history.

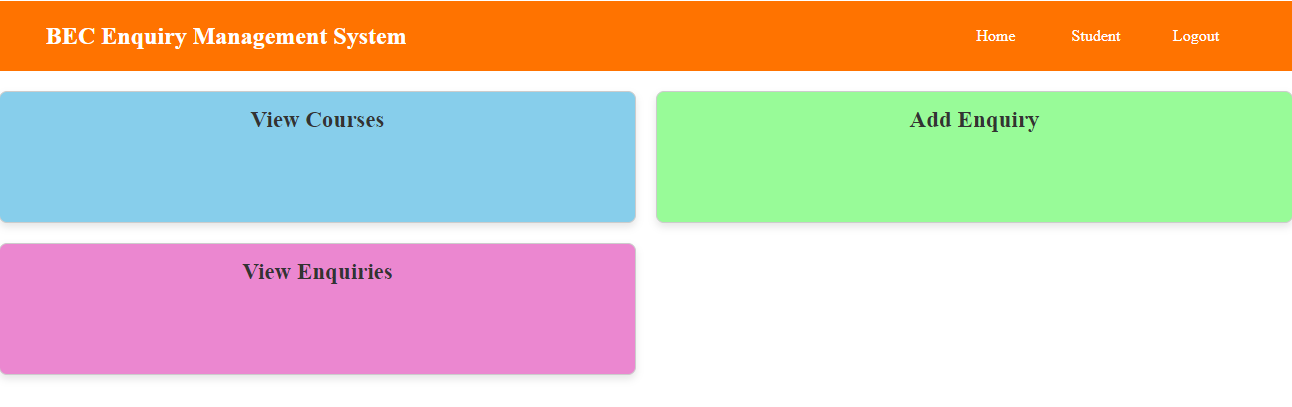
A green and white rectangular box with black text

Description automatically generated

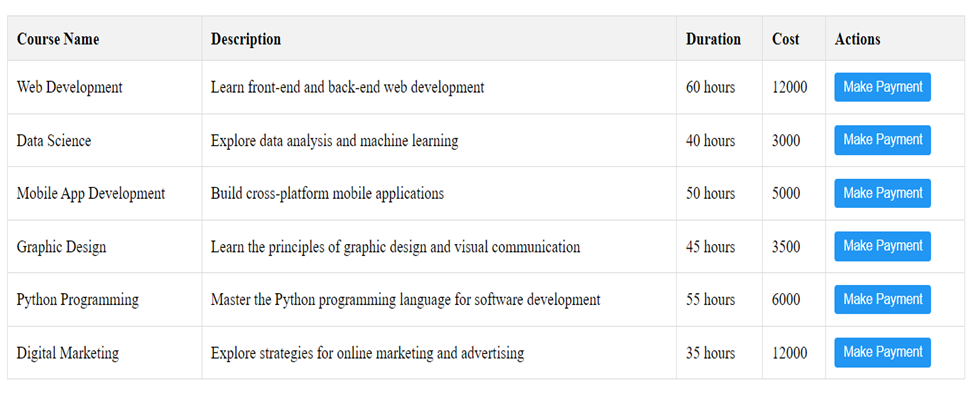
STUDENT Side:

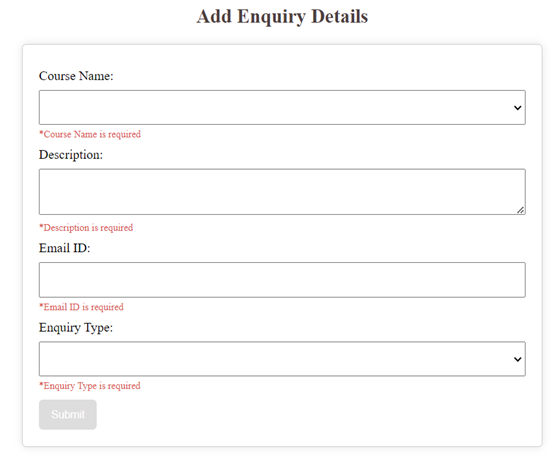


Student side Dashboard

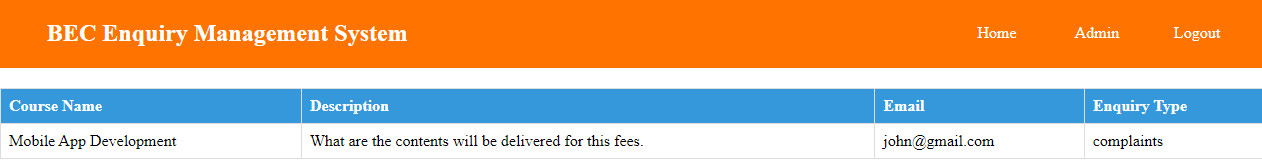


The Student can see the list of course

  
  
On clicking “Add Enquiry” on dashboard , form will be displayed , as shown in the below screenshot

The enquiry form to post the student enquiries.

Student can view the list of enquiries and have the access to delete the enquiries.



The student can do the payment

A screen shot of a form

Description automatically generated

**Other Important Key factors in the application:**

• Should use Custom Exceptions mandatory

• Tables should have proper relationship and keys

• Frontend Application should be menu driven.

• Proper Menu / Navigation for corresponding role

• Client side Validations and server side validations are mandatory

• Error should be handled

• Follow best programmer practice while developing

• Provide proper Naming Conventions

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

1. The react app should run in port 8081.

2. The springboot app should run in port 8080.

**﻿HOW TO RUN THE PROJECT :**

**FRONTEND:**

**Step 1:**

Open the terminal

Use “nvm use 14” command to change node version to 14

**Step 1:**

Use "cd reactapp" command to go inside the reactapp folder

Install Node Modules **- "**npm install**"**

**Step 2:**

Write the code inside src folder

Create the necessary components

**Step 3:**

Click the run test case button to run the test cases

**Note :**

* Click PORT 8081 to view the result / output
* If any error persists while running the app , delete the node modules and reinstall them

**BACKEND:**

**API endpoint:**

8080

**Platform Guidelines:**

To run the command use **Terminal**in the platform.

**Spring Boot:**

Navigate to the springapp directory => **cd springapp**

To start/run the application '**mvn spring-boot:run**'

Click on the Run Test Case button to pass all the test cases

**To Connect Database open terminal**

Cmd: Mysql -u root –protocol=tcp -p

Password: examly