Software Requirements and Design Document

for

CodeFast

Prepared by Arban Arfan(22i-0981), Zakariya Abbas(22i-0801), Messam Raza(22i-1194)

NUCES, Islamabad

23rd November 2024

DUALBOOT D SOFTWARE SOLUTIONS

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document defines the requirements for **CodeFast**, an e-learning platform tailored for students at FAST University enrolled in Algorithm and Data Structure courses. **CodeFast** is designed to provide students with standardized learning materials, practical coding exercises, and a mechanism to showcase their skills. This document covers the features, functionality, and objectives of the platform in its entirety, aligning the software's capabilities with the academic and professional needs of FAST University students.

1.2 Product Scope

CodeFast is a web-based platform developed to bridge the gap between theoretical education and practical coding skills for university students. It offers standardized educational resources, competitive programming opportunities, and a public profile for students to present their skills to potential employers. By addressing gaps in current learning methodologies and aligning with modern industry requirements, **CodeFast** aims to revolutionize how students at FAST University learn, practice, and prepare for their careers.

Key benefits include:

- A centralized repository of high-quality, instructor-approved learning materials.
- Tools for self-assessment through quizzes and challenges.
- A platform for competitive programming to enhance coding efficiency.
- A performance evaluation system to guide student improvement.
- Public profiles to increase visibility among employers for internships and job opportunities.

1.3 Title

CodeFast: Enhancing Learning and Career Readiness through an Interactive Coding Platform CodeFast provides an immediate solution for FAST University students to develop, practice, and demonstrate their programming skills, ensuring alignment with academic goals and industry standards.

1.4 Objectives

The primary objectives of **CodeFas**t are:

- To create a unified platform for learning, practicing, and evaluating programming skills.
- To address inconsistencies in teaching methods by offering standardized learning materials.
- To provide tools for students to engage in regular skill assessments through quizzes and competitive programming contests.
- To enable students to build public profiles showcasing their achievements and skill sets.
- To facilitate students in meeting HEC and university requirements for internships and speed programming contests.
- To help employers identify skilled students efficiently through credible performance metrics and competition results.

1.5 Problem Statement

Currently, FAST University lacks a dedicated digital platform to support students in developing and showcasing their programming skills in a structured and meaningful way. Existing initiatives such as the **"Buddy Program"** and **"Padho and Parhao"** have had limited impact due to inconsistent schedules and insufficient reach.

Students face the following challenges:

- Variability in teaching methods and course content delivery.
- Irrelevant or overly complex learning materials that hinder comprehension.
- A lack of practical exercises aligned with both university curricula and industry requirements.
- Difficulty in preparing for and fulfilling HEC's mandatory internship requirements and university-mandated speed programming contests.

CodeFast addresses these challenges by providing a centralized, accessible platform that streamlines skill development, encourages practical learning, and prepares students for academic and professional success. With features like competitive programming contests, detailed performance evaluations, and public profiles,

CodeFast ensures students can confidently meet academic requirements while being well-prepared for future employment opportunities.

2. Overall Description

2.1 Product Perspective

Context and Origin:

CodeFAST is a new, self-contained e-learning platform developed specifically for students of FAST University. Its goal is to provide an interactive and practical environment for students to enhance their coding skills, evaluate their performance, and connect with potential employers. This software is not a follow-on member of any product family but is designed as a comprehensive solution addressing the gap between education and career opportunities in the software industry.

Position in a Larger System:

CodeFAST serves as a bridge between academia and industry by offering tools for:

- Learning and practicing coding through interactive courses, quizzes, and contests.
- Showcasing student achievements via public profiles and leaderboards.
- Connecting students with industries for internships and job opportunities.

Subsystems and Interfaces:

CodeFAST consists of several interconnected subsystems, including:

1. **User Management System:** For managing user roles (students, admins, and industries).
2. **Course and Quiz Management:** Enabling course registration, content access, and quiz participation.
3. **Contest Management:** Hosting and evaluating coding contests.
4. **Performance Evaluation and Feedback:** Providing students with personalized feedback and progress tracking.
5. **Industry Connection Module:** Facilitating job and internship applications.

The system integrates with:

- **Database:** For securely storing user data, courses, quizzes, contest details, and performance metrics.
- **Email System:** For sending notifications and updates to users.
- **Industry Portal:** Allowing industries to analyze student performance and offer internships/jobs.

2.2 Product Functions

CodeFAST provides the following major functions:

- **User Management:**
    - Students, admins, and industry representatives can log in, manage profiles, and perform role-specific tasks.
- **Course Management:**
    - Students can register for and access course content, including coding exercises and quizzes.
- **Contest Management:**
    - Admins can host public or private contests, provide test cases, and evaluate submissions.
    - Students can participate in contests and view rankings.
- **Performance Evaluation and Feedback:**
    - Automated evaluation of quizzes and contests with detailed feedback.
    - Students can access personalized feedback and performance reports.
- **Leaderboards and Rankings:**
    - Students can view their rankings among peers, while industries can analyze top performers.
- **Industry Engagement:**
    - Industries can host job/internship opportunities and manage applications.
    - Students can apply for internships/jobs and receive offers.
- **Dashboard and Analytics:**

- Admins and industries can access a dashboard summarizing user activities, course participation, and system statistics.

2.3 List of Use Cases

2.4 Extended Use Cases

1. Request Premium Plan (Student)

Component	Description
Use Case Name	Request Premium Plan
Scope	CodeFAST Web Application
Level	User Goal
Primary Actor	Student
Stakeholders and Interests	Student: Wants to unlock premium features like student can Access Study Material Admin: Views the request and can approve or disapprove it.
Preconditions	The student is logged in. The student has a Standard plan before.
Postconditions	The student requests admin to approve his premium plan membership Admin views the request and take action (accept/reject)
Main Success Scenario	*Actors Intention* 1. The student accesses the ‘Study Material’ catalog. 2. The student selects the premium plan. *System Responsibility* 3. System Handles the Request and stores it in database. 4. System forwards the request to admin for approval
Extensions	4a. If an error occurs during activation, the system logs the issue, and the student is directed to customer support.

2. Register Course (Student)

Component	Description	
Use Case Name	Register Course	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student	
Stakeholders and Interests	Student: Wants to enroll in relevant courses to learn and practice Course skills. Admin: Needs to ensure students enroll in appropriate courses and approve their registration request.	
Preconditions	The student is logged in. Courses are available for registration.	
Postconditions	The student sends a request to the admin for course enrollment. Course materials are accessible.	
Main Success Scenario	*Actors Intention* 1.The student accesses the course catalog. 2.The student selects a course. 3.The Student requests to register the course.	*System Responsibility* 4.The System records the request in the database. 5.System forwards the request to admin for approval
Extensions	3a. If the course is full, the system shows an error message, and the student is not enrolled. 3b. If the student has not met the course prerequisites, an error message is displayed.	

3. Access Enrolled Course (Student)

Component	Description	
Use Case Name	Access Enrolled Course	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student	
Stakeholders and Interests	Student: Needs to access the enrolled course materials and Quizzes. Admin: Wants to ensure that only registered students access the courses.	
Preconditions	The student is logged in. The student is enrolled in at least one course.	
Postconditions	The student has access to the course content and assignments.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The student navigates to the “Courses” section.	
		2. The system displays a list of enrolled courses.
	3. The student selects a course.	
		4. The system grants access to the course materials and Quizzes.
Extensions	4a. If the student’s enrollment has expired, the system shows a warning and denies access to the course.	

4. Participate in Quiz (Student)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Participate in Quiz</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Student</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Student: Wants to participate in Quiz to practice and improve skills.
Admin: Ensures smooth participation and records performance.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The student is logged in.
There is an active Quiz available for participation.</td></tr>
<tr><td>Postconditions</td><td colspan="2">The student’s participation is recorded, and results are stored.</td></tr>
<tr><td rowspan="6">Main Success Scenario</td><td><i>Actors Intention</i></td><td><i>System Responsibility</i></td></tr>
<tr><td>1. The student navigates to the Course page.</td><td></td></tr>
<tr><td></td><td>2. The system shows available Quizzes relevant to the Course</td></tr>
<tr><td>3.The student selects a course.</td><td></td></tr>
<tr><td>4.The student participates in the quiz.
5.The Student selects the options in the MCQs.
6.The Student clicks submit button</td><td></td></tr>
<tr><td></td><td>7. The system records the performance in database and evaluates the student’s performance.</td></tr>
<tr><td>Extensions</td><td colspan="2">4a. If the Quiz is already attempted, the system displays a message and the student cannot participate.</td></tr>
</table>

5. Participate in Contest (Student)

Component	Description	
Use Case Name	Participate in Contest	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student	
Stakeholders and Interests	Student: Wants to participate in contests to practice and improve skills. Admin: Ensures smooth participation and records performance.	
Preconditions	The student is logged in. There is an active contest available for participation.	
Postconditions	The student's participation is recorded, and results are stored.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The student navigates to the contest page.	
		2. The system shows available contests.
	3.The student selects a contest.	
	4.The student writes the code in the text area	
	5. Student clicks the submit button	
		6. The system records and evaluates the student's performance.
Extensions	4a. If the contest is already attempted then student can't re-attempt it	

6. Evaluate Student Contest (Admin and Student)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Evaluate Student Contest</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Admin and Student</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin: Needs to provide Test Case for a Contest
Student: Wants timely and accurate assessment of submitted assignments.</td></tr>
<tr><td>Preconditions</td><td colspan="2">Admin is logged in for uploading Contest & Student for Participating in it
Admin has uploaded the test cases for the contest</td></tr>
<tr><td>Postconditions</td><td colspan="2">The students’ assessments are evaluated, and student CodeFast performance is updated.</td></tr>
<tr><td>Main Success Scenario</td><td>*Actors Intention*
1. Student writes the code in the text area.
2. Student clicks the submit button.
6.Student can view the status of submission and resubmit if necessary.</td><td>*System Responsibility*
3. The System gets test cases of the contest from database
4. The System writes the submitted code in a C++ file, compiles and runs it
5.System returns status to student.
7.System marks the student submission and updates performance on application.</td></tr>
<tr><td>Extensions</td><td colspan="2">4a. If the submission is incomplete, the system marks it as pending evaluation.</td></tr>
</table>

7. Evaluate Student Quiz (Student)

Component	Description	
Use Case Name	Evaluate Student Quiz	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student	
Stakeholders and Interests	Student: Wants timely and accurate assessment of submitted assignments.	
Preconditions	Student is logged in Quiz is submitted by students.	
Postconditions	The students' assessments are evaluated, and student CodeFast performance is updated.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. Student selects option for MCQs 2.Student clicks submit button	
		3. The system retrieves the correct option from database. 4. The System evaluates the answer of student. 5. System updates student's performance on the application 6. The System updates on database
	7.Student can view his score and updated performance	
Extensions	4a. If the submission is incomplete, the system marks it as pending evaluation.	

8. Give Student Feedback (Student)

Component	Description	
Use Case Name	Give Student Feedback	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student	
Stakeholders and Interests	Student: Wants to receive feedback on the performance.	
Preconditions	The student is logged in.	
Post conditions	The feedback is accessible to student.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The student navigates to the Feedback Tab.	
		2. The system retrieves the student CodeFast performance.
		3. System gives feedback according to parameters set by admin.
	3. The student views the Feedback received.	
Extensions	3a. System provides general feedback to the students if they have not participated in any quiz or contest.	

9. Host Public contest (Admin)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Host Public Contest</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Admin</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin: Hosts contests to engage students and evaluate their performance.
Student: Wants access to participate in high-quality contests.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The admin is logged in.
The contest setup is complete (rules, questions, etc.).</td></tr>
<tr><td>Postconditions</td><td colspan="2">The contest is successfully hosted and accessible to students.</td></tr>
<tr><td rowspan="5">Main Success Scenario</td><td><em>Actors Intention</em></td><td><em>System Responsibility</em></td></tr>
<tr><td>1. The admin navigates to the contest management page.</td><td></td></tr>
<tr><td></td><td>2.The system displays the option to host a contest.</td></tr>
<tr><td>3. The admin adds the Contest Name, Start Date, End Date, Description and Test Cases</td><td>4.The System Handle the Request
5.The System stores the contest in database
6.System updates Admin for successful contest addition.</td></tr>
<tr><td>7. Admin can view the added contest</td><td></td></tr>
<tr><td>Extensions</td><td colspan="2">4a. If the contest cannot be hosted due to an error, the system notifies the admin and logs the issue.</td></tr>
</table>

10. View Contest Ranking (Student and Admin and Industry)

Component	Description	
Use Case Name	View Contest Ranking	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student, Admin, Industry	
Stakeholders and Interests	Student: Wants to know their ranking after participating in a contest. Admin: Oversees rankings to manage results. Industry: Interested in top-performing students for internships and jobs.	
Preconditions	Contest results are finalized. The ranking is publicly accessible.	
Postconditions	The contest ranking is viewed successfully.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The user navigates to the contest page.	2. The system shows Display Contest Rankings button
	3. The user selects the display ranking button	4. System retrieves the contest performance from database. 5. The system displays the contest rankings.
	6. The user views the rankings and performance of participants.	
Extensions	3a. User selects button but no action performed or system shows no data. 4a. If the ranking is not yet available, system shows a pending message.	

11. Analyze Student Performance (Admin and Industry)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Analyze Student Performance</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Admin and Industry</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin: Needs to analyze student performance across various quizzes and contests to provide feedback.
Industry: Interested in students’ performance to shortlist students for internships and jobs.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The admin is logged in.
Student performance data is available.</td></tr>
<tr><td>Postconditions</td><td colspan="2">A detailed performance report is generated for each student.</td></tr>
<tr><td>Main Success Scenario</td><td><em>Actors Intention</em>
1.The user selects the performance analysis option.
4. The user reviews the report and uses it for feedback or decisions.</td><td><em>System Responsibility</em>
2.The system retrieves performance data from database.
3.The system generates and displays a report.</td></tr>
<tr><td>Extensions</td><td colspan="2">4a. Student has no performance to show if he has not participated in any contest or Quizzes.</td></tr>
</table>

12. Maintain Public profile (Student and admin and Industry)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Maintain Public Profile</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Student, Admin and Industry</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin/Industry/Student: Can change their name, email and password for their ease to display updated information and get emails.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The User is logged in.
The profile information is available.</td></tr>
<tr><td>Postconditions</td><td colspan="2">The student’s public profile is updated and accessible.</td></tr>
<tr><td>Main Success Scenario</td><td>*Actors Intention*
1. The user navigates to the setting page.
3. The user makes updates, can change his Name, email and Password.
6.User get notifies and can view the changes</td><td>*System Responsibility*
2. The system displays the profile information.
4. The system saves the changes and updates the information in database.
5.System notifies the user about the information updated successfully.</td></tr>
<tr><td>Extensions</td><td colspan="2">4a. The email is not valid, System alerts the user to enter correct email
4b. To change the Password old password has to be entered correctly.</td></tr>
</table>

13. View top candidates (admin, industry, student)

Component	Description	
Use Case Name	View Top Candidates	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Admin, Industry, Student	
Stakeholders and Interests	Admin: Needs to view top-performing students. Industry: Interested in top candidates for internships and jobs. Student: Wants to know where they rank among peers.	
Preconditions	The user is logged in. Performance data and rankings are available.	
Postconditions	The top candidates are successfully displayed.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The user navigates to Performance section.	
	2. User press the View Leader Boards button	
		3. The system retrieves a list of top candidates from database. 4.The System sorts the data based on performance, cgpa and number of participations.
		5.System displays the list to user on screen
	6.The user views the list.	
Extensions	2a. If there are no top candidates available, the system displays a message indicating no results.	

14. Offer internship (Industry)

Component	Description	
Use Case Name	Offer Internship	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Industry and Student	
Stakeholders and Interests	Student: They can apply for jobs/internship and they can accept or reject the offers of the industries. Industry: Wants to get internees and employees for their organization and they can accept or reject the request of the students.	
Preconditions	The industry is logged in. The student is logged in.	
Post conditions	The students will get an Internship/Job if.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The User navigates to the Job/Internship Tab.	2. The system displays the list of the registered Industries or registered students.
	3. The user selects the student and clicks the offer button.	4. The system records and sends requests to the desired user.
	5. The user can accept and decline the request.	
Extensions	Students who were rejected by the Industry can not apply for Internship and jobs again.	

15. Apply for internship/job (Student)

Component	Description	
Use Case Name	Apply for Job/Internship	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Student and Industry	
Stakeholders and Interests	Student: They can apply for jobs/internship and they can accept or reject the offers of the industries. Industry: Wants to get internees and employees for their organization and they can accept or reject the request of the students.	
Preconditions	The student is logged in.	
Post conditions	The students will get an Internship/Job if a request is accepted. The industry will get an employee or internee if a request is accepted.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The User navigates to the Job/Internship Tab.	2. The system displays the list of the registered Industries.
	3. The user selects the industry and clicks the apply button.	4. The system records and sends requests to the desired user.
	5. The user can accept and decline the request.	
Extensions	Students who were rejected by the industry can not apply for Internship and jobs again.	

16. Manage Internship/Job Students (Industry)

Component	Description	
Use Case Name	Manage Internships/Job	
Scope	CodeFAST Web Application	
Level	User Goal	
Primary Actor	Industry	
Stakeholders and Interests	Industry: Wants to remove internees and employees from their organization.	
Preconditions	The industry is logged in.	
Post conditions	The industry will add and remove Internees.	
Main Success Scenario	*Actors Intention*	*System Responsibility*
	1. The Industry navigates to the Manage Job/Internship Tab.	2. The system displays the list of the students enrolled in Industry.
	3. The Industry selects the student.	
	4. The Industry clicks the remove button.	
		5. The system records the transactions.
Extensions	Once the Industry removes the student, they cannot apply again.	

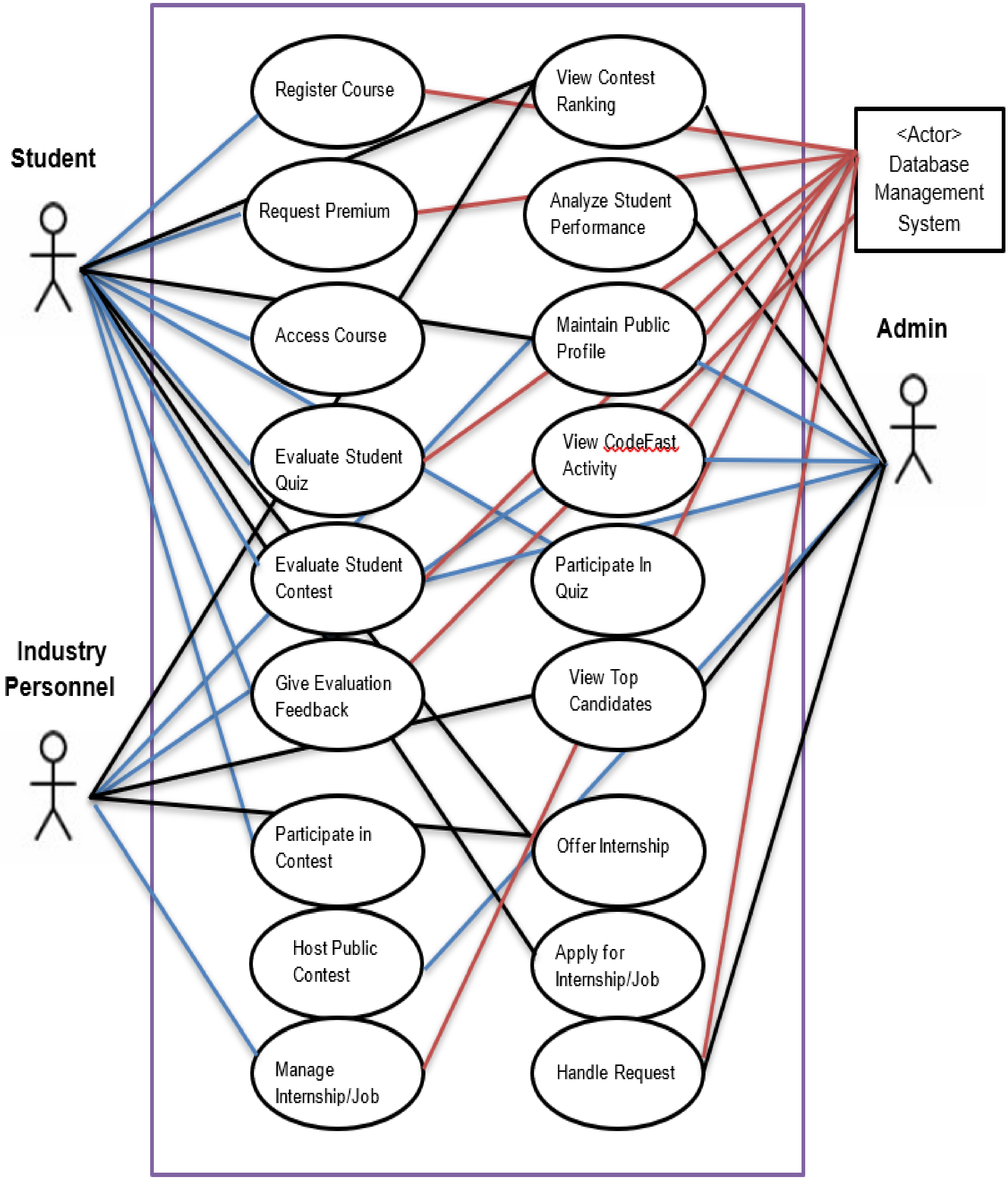
17. Handle Requests (Admin)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">Handle Request</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Admin</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin: Manages requests related to student data or course enrollment, ensuring accurate records.
Student: Expects their requests (e.g., **Student Registration**, C**ourse registrations** and **Premium plan**) to be handled promptly and correctly.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The admin is logged in.
Relevant student data or requests are available.</td></tr>
<tr><td>Postconditions</td><td colspan="2">Student data is updated based on the admin’s decision.
Requests are handled (approved, rejected, or updated).</td></tr>
<tr><td rowspan="7">Main Success Scenario</td><td>*Actors Intention*</td><td>*System Responsibility*</td></tr>
<tr><td>1. Admin navigates to the request management page.</td><td></td></tr>
<tr><td></td><td>2. The system displays pending requests.</td></tr>
<tr><td>3. Admin reviews a specific student request.</td><td></td></tr>
<tr><td></td><td>4. The system retrieves and shows the request details.</td></tr>
<tr><td>5. Admin approves, rejects, or updates the request.</td><td></td></tr>
<tr><td></td><td>6. The system processes the action and updates the relevant data.</td></tr>
<tr><td>Extensions</td><td colspan="2">5a. If the request is invalid (e.g., missing required information), the system notifies the admin with an error message.
4a. If the system fails to retrieve request data, an error message is displayed.</td></tr>
</table>

18. View CodeFast Activity (Admin and Industry)

<table>
<tr><th>Component</th><th colspan="2">Description</th></tr>
<tr><td>Use Case Name</td><td colspan="2">View CodeFast Activity</td></tr>
<tr><td>Scope</td><td colspan="2">CodeFAST Web Application</td></tr>
<tr><td>Level</td><td colspan="2">User Goal</td></tr>
<tr><td>Primary Actor</td><td colspan="2">Admin and Industry</td></tr>
<tr><td>Stakeholders and Interests</td><td colspan="2">Admin: Can view the entire activity for example the total number of registered students, industries, courses, contests and quizzes and could manage the application accordingly.
Industry: Can view the entire activity for example the total number of registered students, industries, courses, contests and quizzes and could use the platform to hire outstanding students and see competition.</td></tr>
<tr><td>Preconditions</td><td colspan="2">The user is logged in.
Relevant student data is available.</td></tr>
<tr><td>Postconditions</td><td colspan="2">Relevant Data is displayed.</td></tr>
<tr><td rowspan="5">Main Success Scenario</td><td><em>Actors Intention</em></td><td><em>System Responsibility</em></td></tr>
<tr><td>1. User navigates the dashboard.</td><td></td></tr>
<tr><td></td><td>2. The system retrieves the data of total registered students, industries, courses, contests, quizzes from the database.</td></tr>
<tr><td></td><td>3. System displays the data in the form of Bar Chart and pie chart.</td></tr>
<tr><td>4. User views the data</td><td></td></tr>
<tr><td>Extensions</td><td colspan="2">5a. If the request is invalid (e.g., missing required information), the system notifies the admin with an error message.
4a. If the system fails to retrieve request data, an error message is displayed.</td></tr>
</table>

2.5 Use Case Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

- **Application Response Time**:
  - Loading any screen (e.g., dashboard, course list) should take less than 1 second.
  - Database queries for retrieving course content, quiz results, or leaderboard data should execute within 500 milliseconds for datasets up to 100,000 records.
- **Startup Time**: The application must fully initialize within 5 seconds on recommended hardware.
- **Concurrency**: Support up to 50 simultaneous users working on the same database via networked connections without performance degradation.
- **Improved Performance**: We have used Hash maps instead of list for quick query retrieval. The Information Expert retrieves all the data at the start of the program and provides to all the classes. This reduces the frequent database operations exponentially increasing the application's performance. All the data is retrieved in Big Oh(1).

3.2 Safety Requirements

- **Data Backup**:
  - The application must create daily backups of the MySQL database to ensure data safety in case of hardware or software failure.
- **Error Recovery**:
  - If a failure occurs during operations (e.g., quiz submission), the system must log the error and allow users to retry the failed action without data loss.
- **Accidental Deletion Prevention**:
  - Critical actions like course deletion should prompt confirmation dialogs to prevent accidental data loss.

3.3 Security Requirements

- **Authentication**:
  - Users must log in with unique credentials; the system must enforce strong password policies (minimum 8 characters, including alphanumeric and special characters).
- **Data Encryption**:
  - Sensitive user information (e.g., passwords) must be encrypted using SHA-256.
  - Data transferred between the application and database over a network must be secured with SSL/TLS.
- **Access Control**:
  - Role-based access control (RBAC) to ensure only authorized users (e.g., admins) can perform sensitive actions like user management or course creation.
- **Secure Local Storage**:
  - Cache or temporary data stored locally must be encrypted to prevent unauthorized access.

3.4 Software Quality Attributes

- **Reliability**:
  - The system should have an uptime of 99.8% during operational hours.
- **Usability**:
  - The interface must follow an intuitive design with minimal user training required, adhering to GUI standards in Scene Builder.
- **Maintainability**:
  - The codebase should be modular and follow industry-standard practices to simplify future updates and debugging.
- **Portability**:

    - The application must run seamlessly on Windows 10/11 and macOS systems.

- **Scalability**:
    - While initially supporting 50 users, the system design should allow scaling to 200 users with minimal changes to architecture.
- **Testability**:
    - The application must support unit and integration testing for major components such as database operations, user interfaces, and business logic.

3.5 Business Rules

- **User Roles**:
    - Students can only view and edit their profiles and access learning materials.
    - Admins have permissions to manage users, accept courses registration requests, quizzes, and contests.
    - Employers can view anonymized student profiles and post job opportunities.
- **Leaderboards**:
    - Only the top students on the bases of their **CodeFast Performance** are displayed, with an option to anonymize the leaderboard on request.
- **Feedback Mechanism**:
    - Students receive feedback on the bases of their overall performance in quizzes and contests they participate in.

3.6 Operating Environment

- **Hardware Platform**:
    - Minimum requirements for the client system:
        - 8 GB RAM

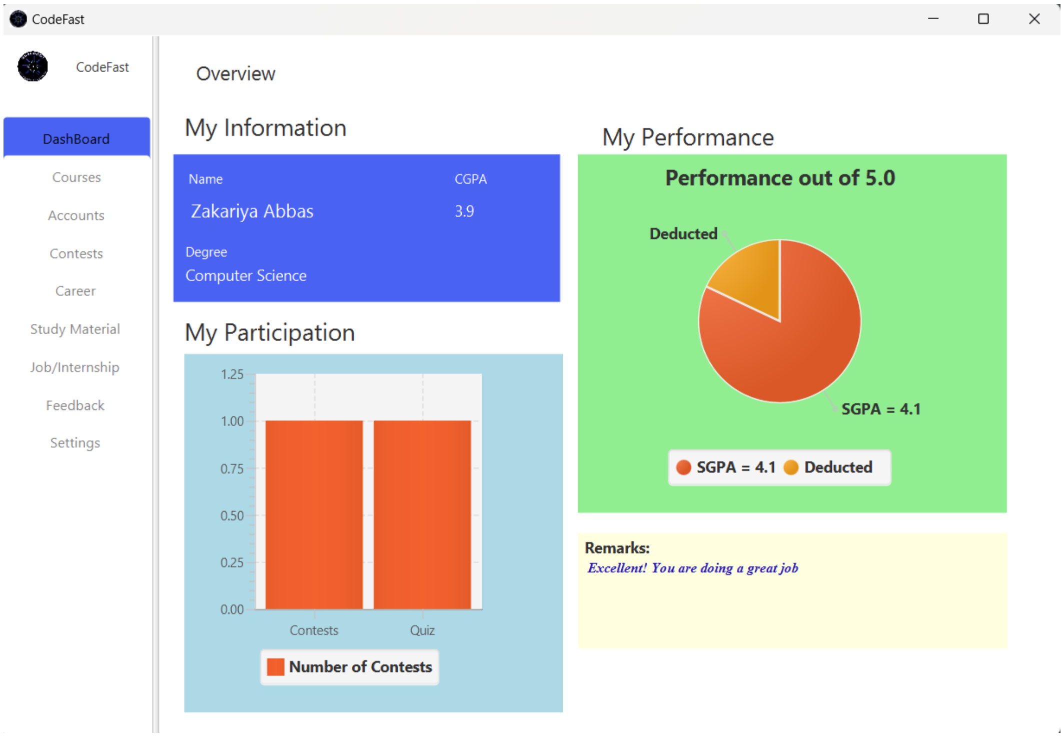
        - 2 GHz dual-core processor
        - 500 MB free storage
    - Server hosting the MySQL database:
        - 16 GB RAM
        - 4-core processor
        - 500 GB SSD
- **Operating System**:
    - The application must run on Windows 10/11 (64-bit) and macOS Monterey or later.
- **Dependencies**:
    - JavaFX SDK 17 for UI.
    - MySQL Connector for Java for database communication.
    - Scene Builder for designing user interfaces.

3.7 User Interfaces

The platform includes three primary user interfaces:

- Student Interface
- Industry Interface
- Admin Interface

- **Student Interface**: Enables material requests, quiz participation, contest engagement, and application submissions.



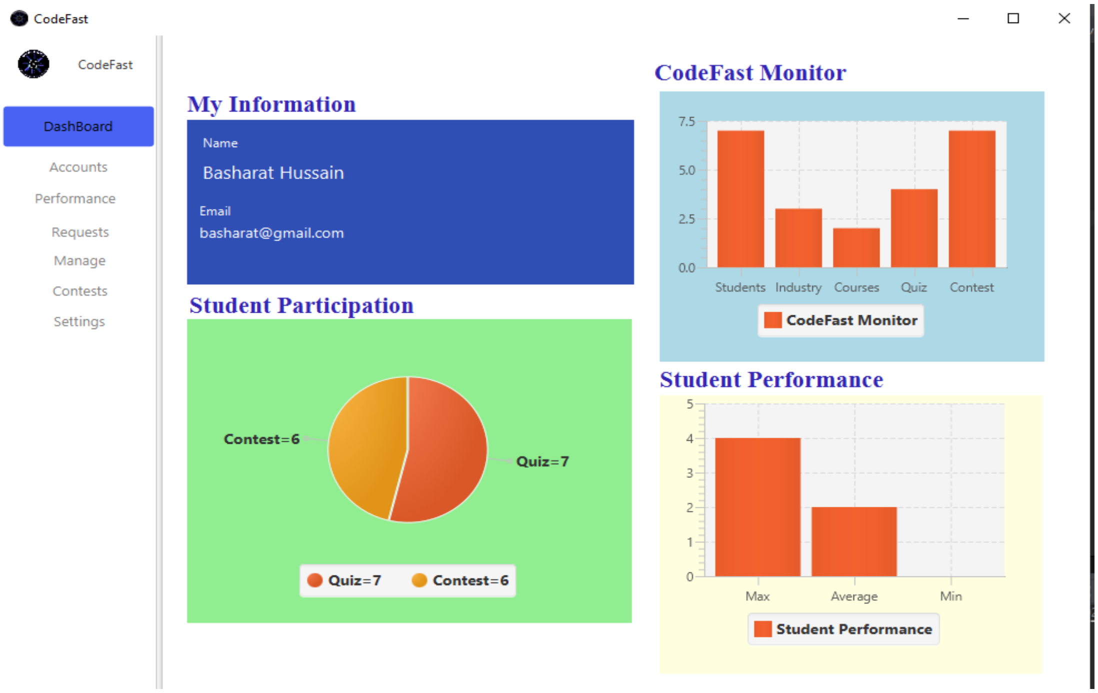
My Information

Name	CGPA
Zakariya Abbas	3.9

Degree
Computer Science

Remarks:
Excellent! You are doing a great job

- **Admin Interface**: Facilitates management of requests, activity provision, and contest hosting.

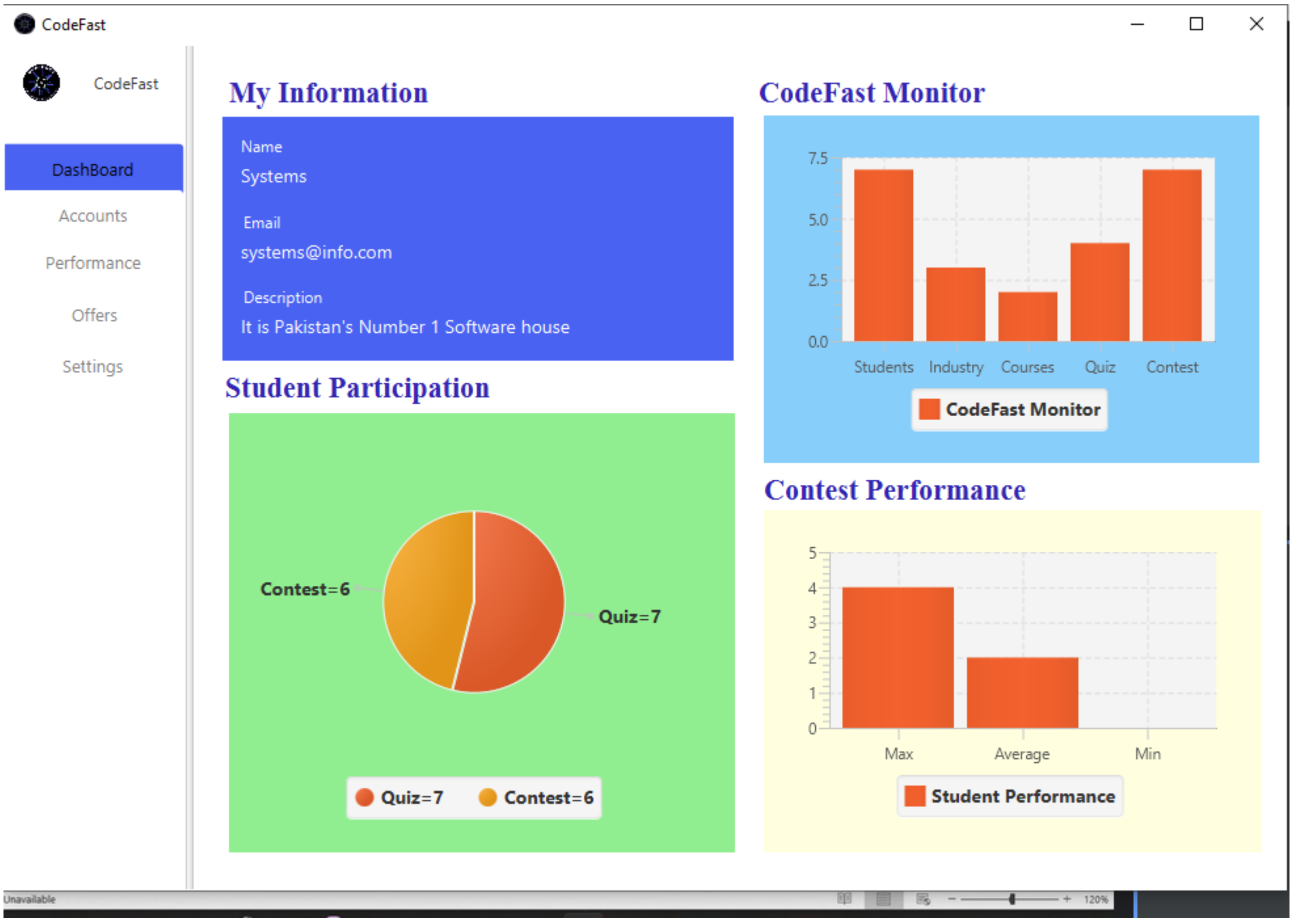


My Information

Name
Basharat Hussain

Email
basharat@gmail.com

- **Industry Interface**: Allows viewing of high achievers and handling of applications.



My Information

Name
Systems

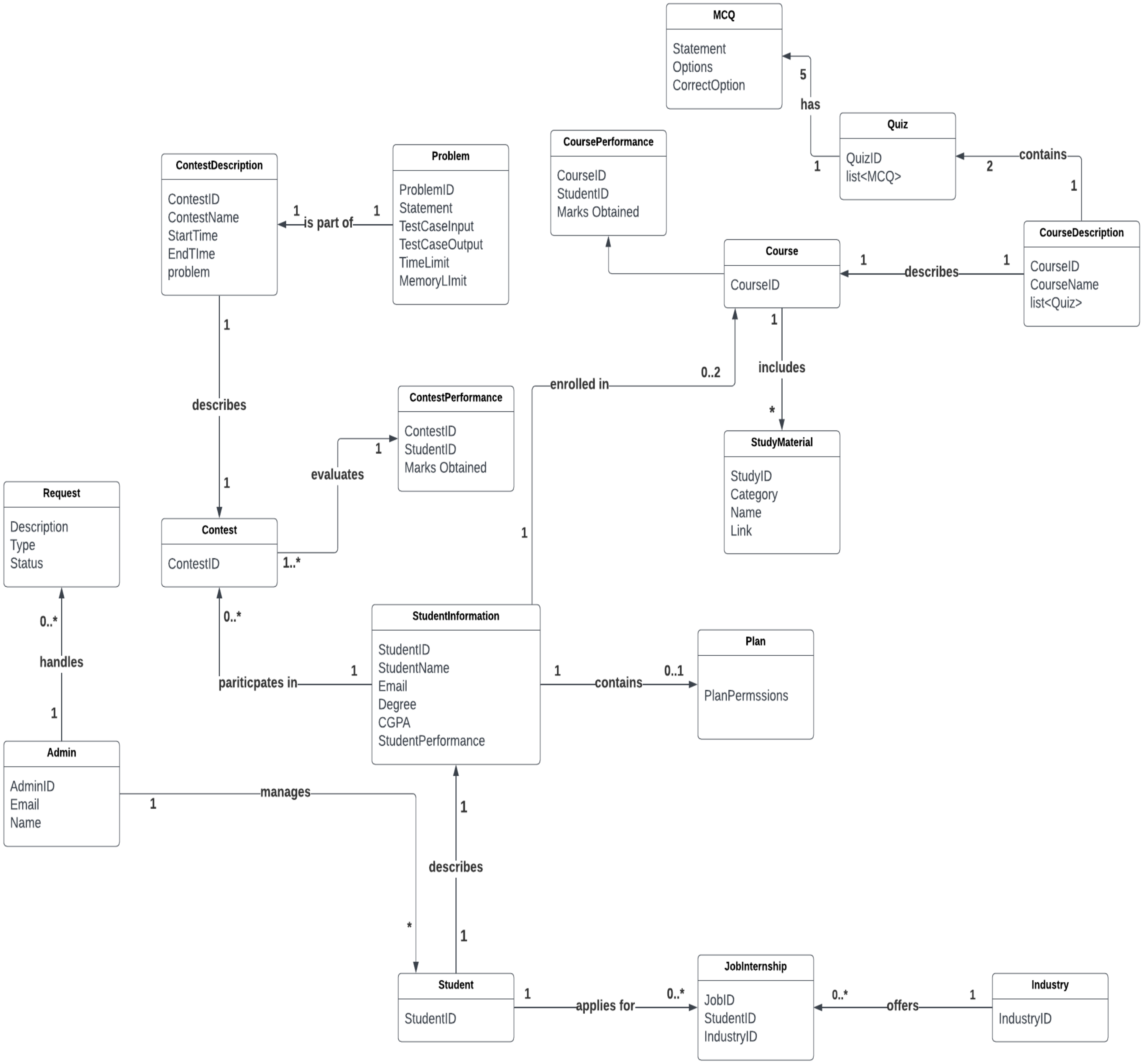
Email
systems@info.com

Description
It is Pakistan's Number 1 Software house

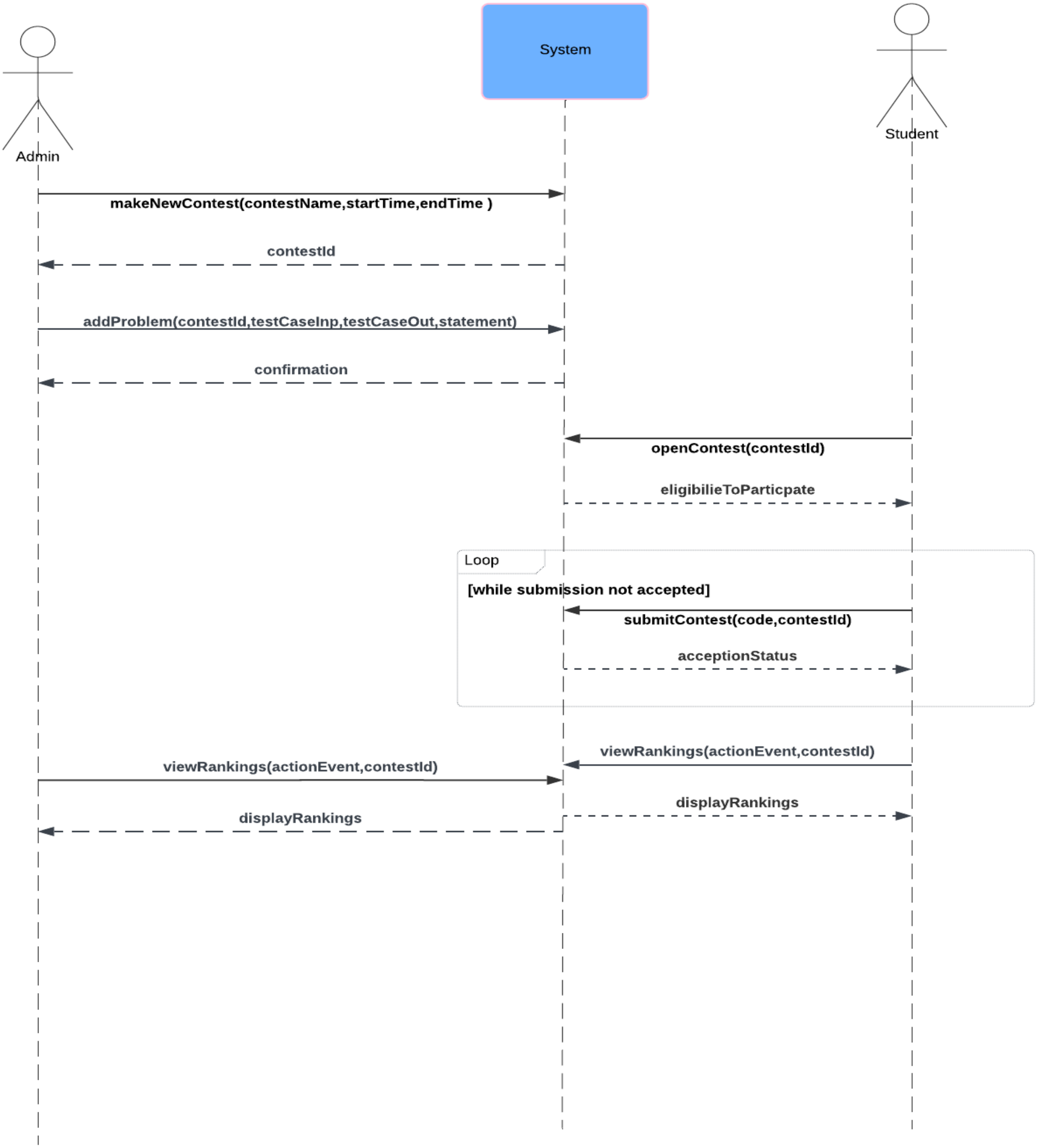
Interface Standards:

- Consistent layout across screens with standard navigation menus and buttons.
- Error messages displayed prominently with actionable guidance.
- Support for keyboard shortcuts and accessible design principles.

4. Domain Model

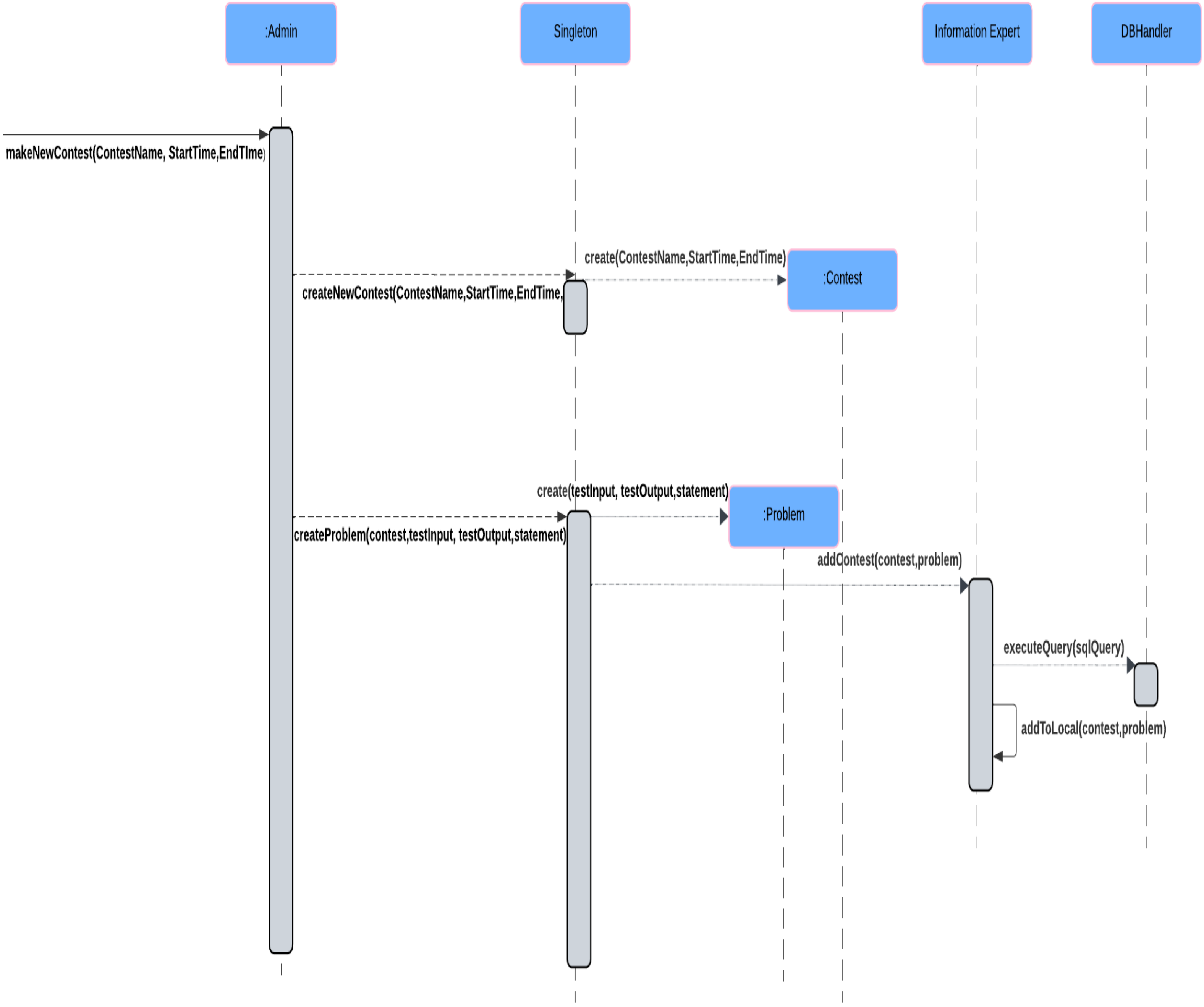


5. System Sequence Diagram

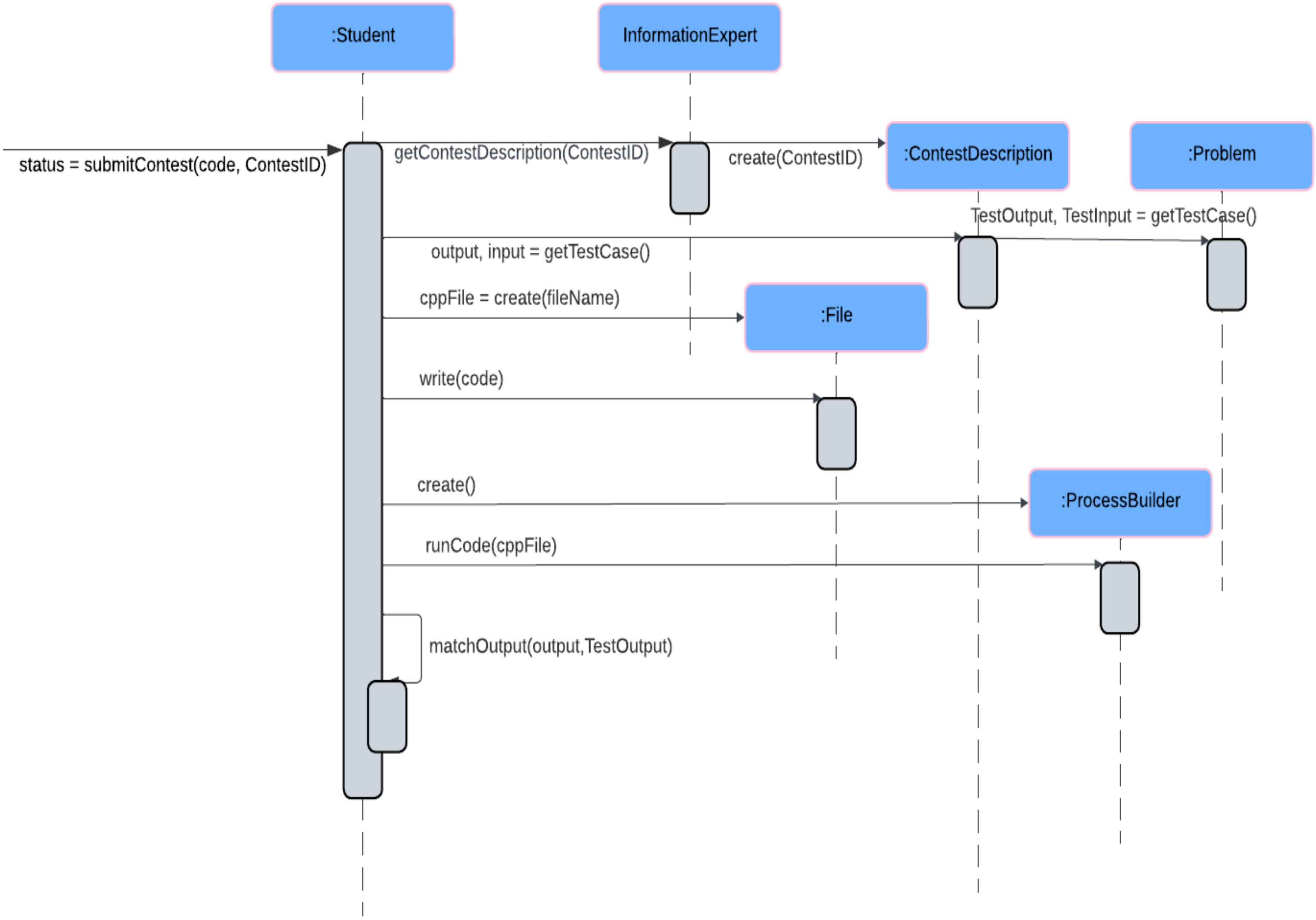

System
Admin
Student
makeNewContest(contestName,startTime,endTime)
contestId
addProblem(contestId,testCaseInp,testCaseOut,statement)
confirmation
openContest(contestId)
eligibilieToParticpate
Loop
[while submission not accepted]
submitContest(code,contestId)
acceptionStatus
viewRankings(actionEvent,contestId)
viewRankings(actionEvent,contestId)
displayRankings
displayRankings

6. Sequence Diagram

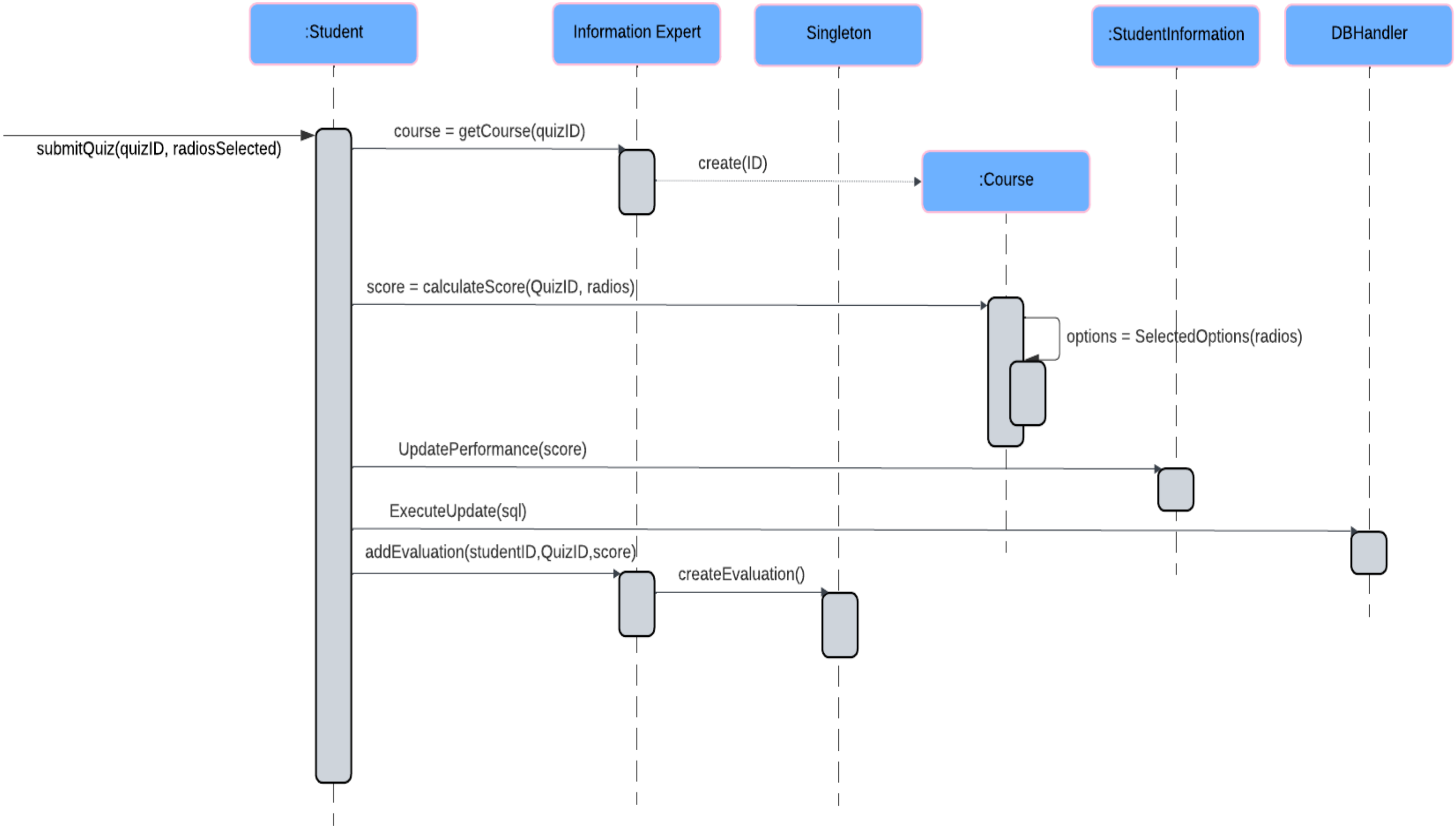
1. Making New Contest Sequence Diagram


:Admin
Singleton
Information Expert
DBHandler
makeNewContest(ContestName, StartTime,EndTIme)
create(ContestName,StartTime,EndTime)
:Contest
createNewContest(ContestName,StartTime,EndTime,
create(testInput, testOutput,statement)
:Problem
createProblem(contest,testInput, testOutput,statement)
addContest(contest,problem)
executeQuery(sqlQuery)
addToLocal(contest,problem)

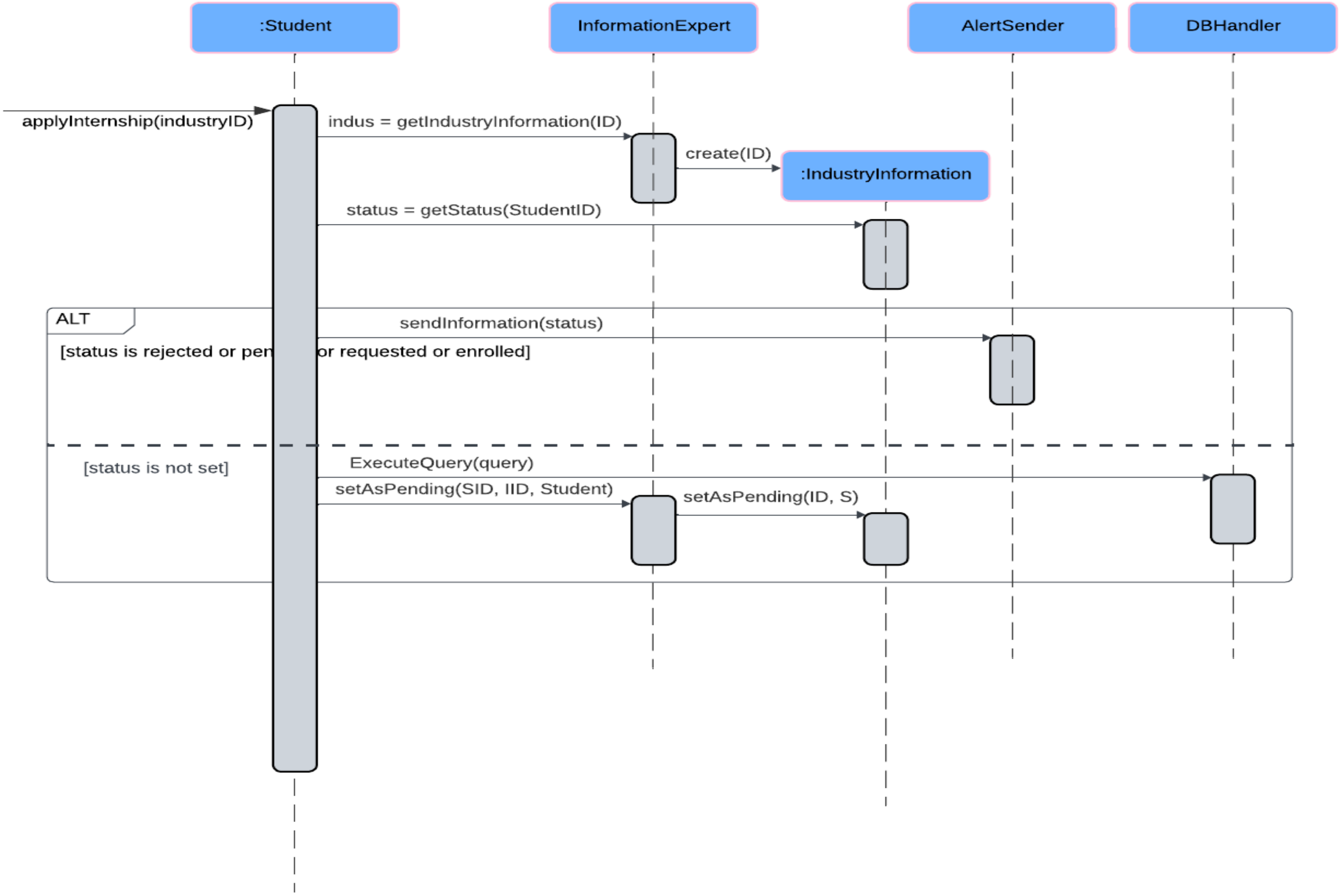
2. Submit Contest (Student)


:Student
InformationExpert
:ContestDescription
:Problem
:File
:ProcessBuilder
status = submitContest(code, ContestID)
getContestDescription(ContestID)
create(ContestID)
TestOutput, TestInput = getTestCase()
output, input = getTestCase()
cppFile = create(fileName)
write(code)
create()
runCode(cppFile)
matchOutput(output,TestOutput)

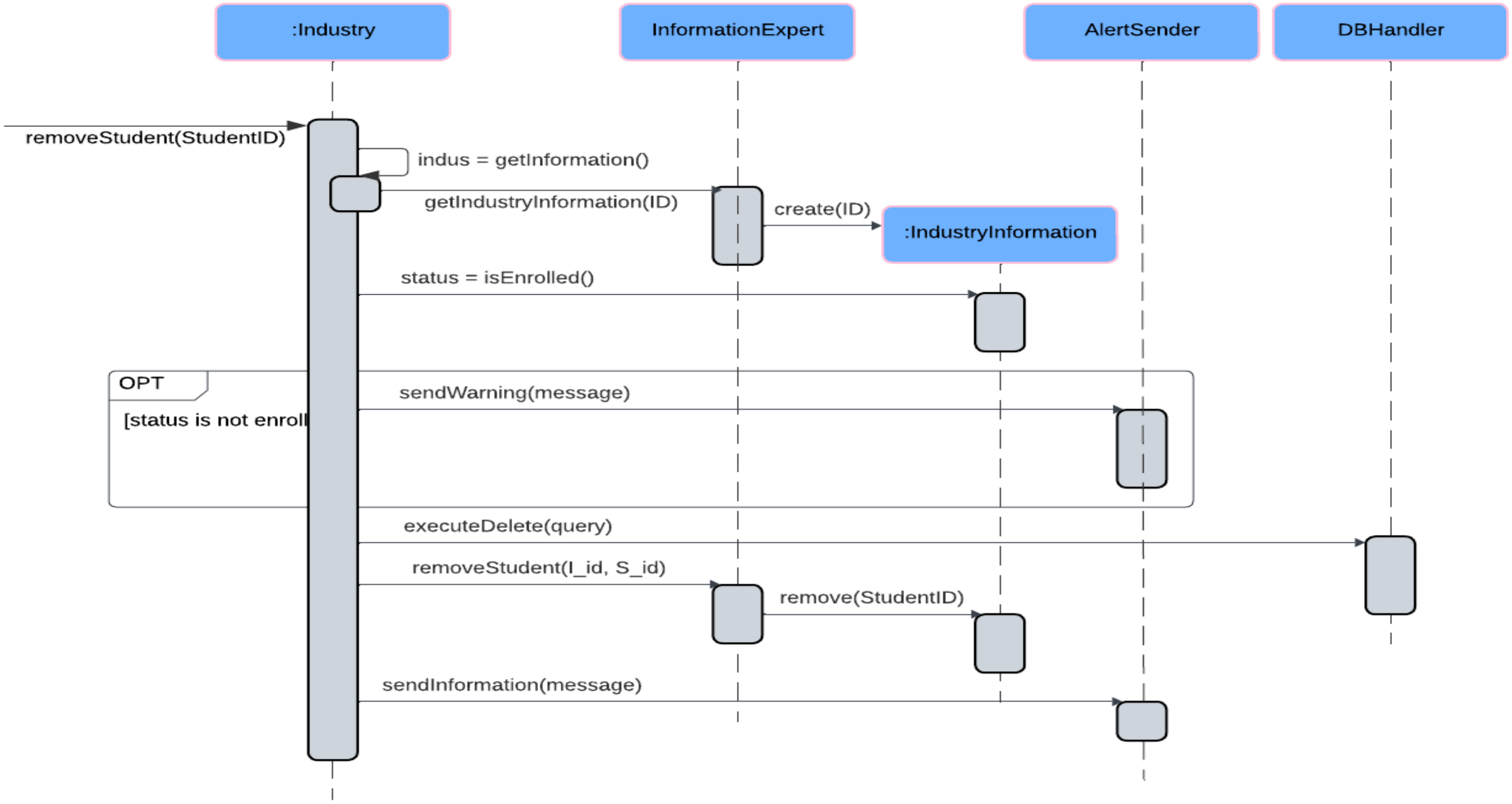
3. Submit Quiz (Student)


:Student
Information Expert
Singleton
:StudentInformation
DBHandler
submitQuiz(quizID, radiosSelected)
course = getCourse(quizID)
create(ID)
:Course
score = calculateScore(QuizID, radios)
options = SelectedOptions(radios)
UpdatePerformance(score)
ExecuteUpdate(sql)
addEvaluation(studentID,QuizID,score)
createEvaluation()

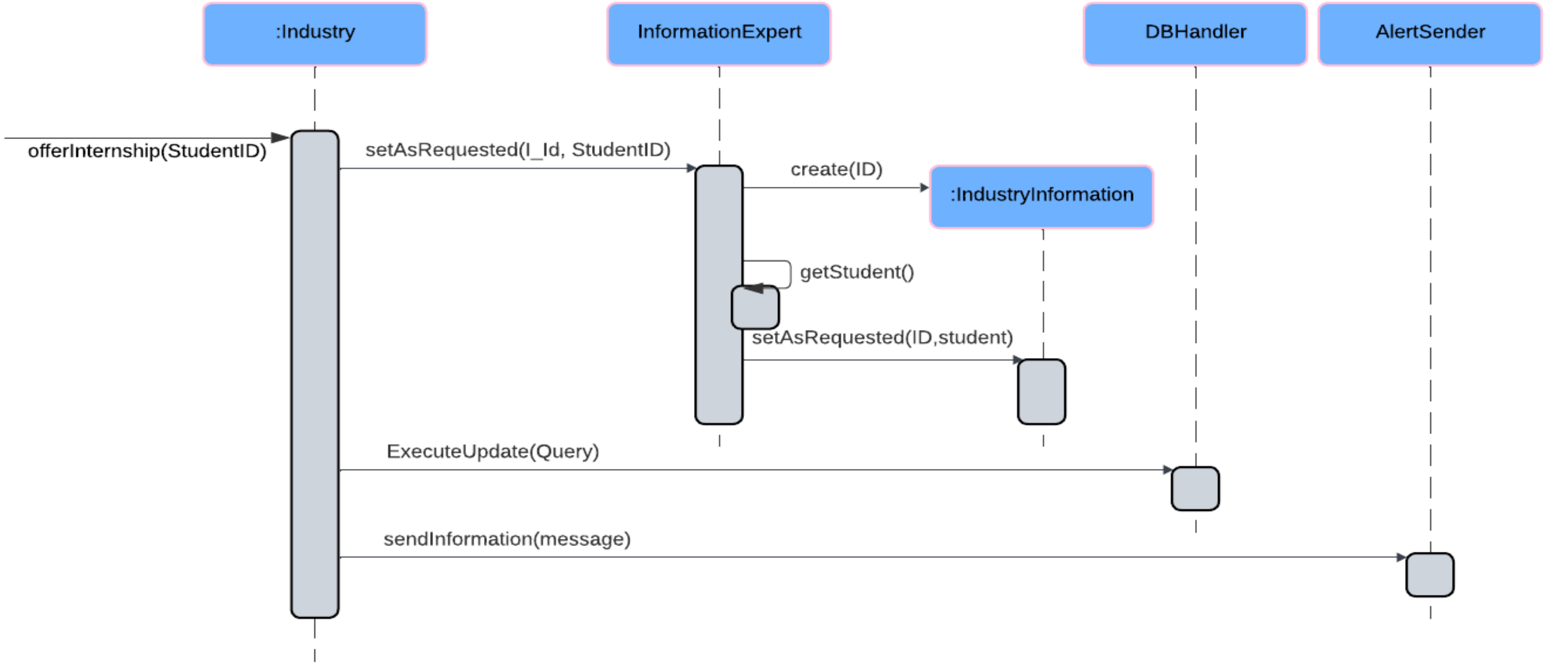
4. Apply for Job/Internship (Student)


:Student
InformationExpert
AlertSender
DBHandler
applyInternship(industryID)
indus = getIndustryInformation(ID)
create(ID)
:IndustryInformation
status = getStatus(StudentID)
ALT
sendInformation(status)
[status is rejected or pen
or requested or enrolled]
[status is not set]
ExecuteQuery(query)
setAsPending(SID, IID, Student)
setAsPending(ID, S)

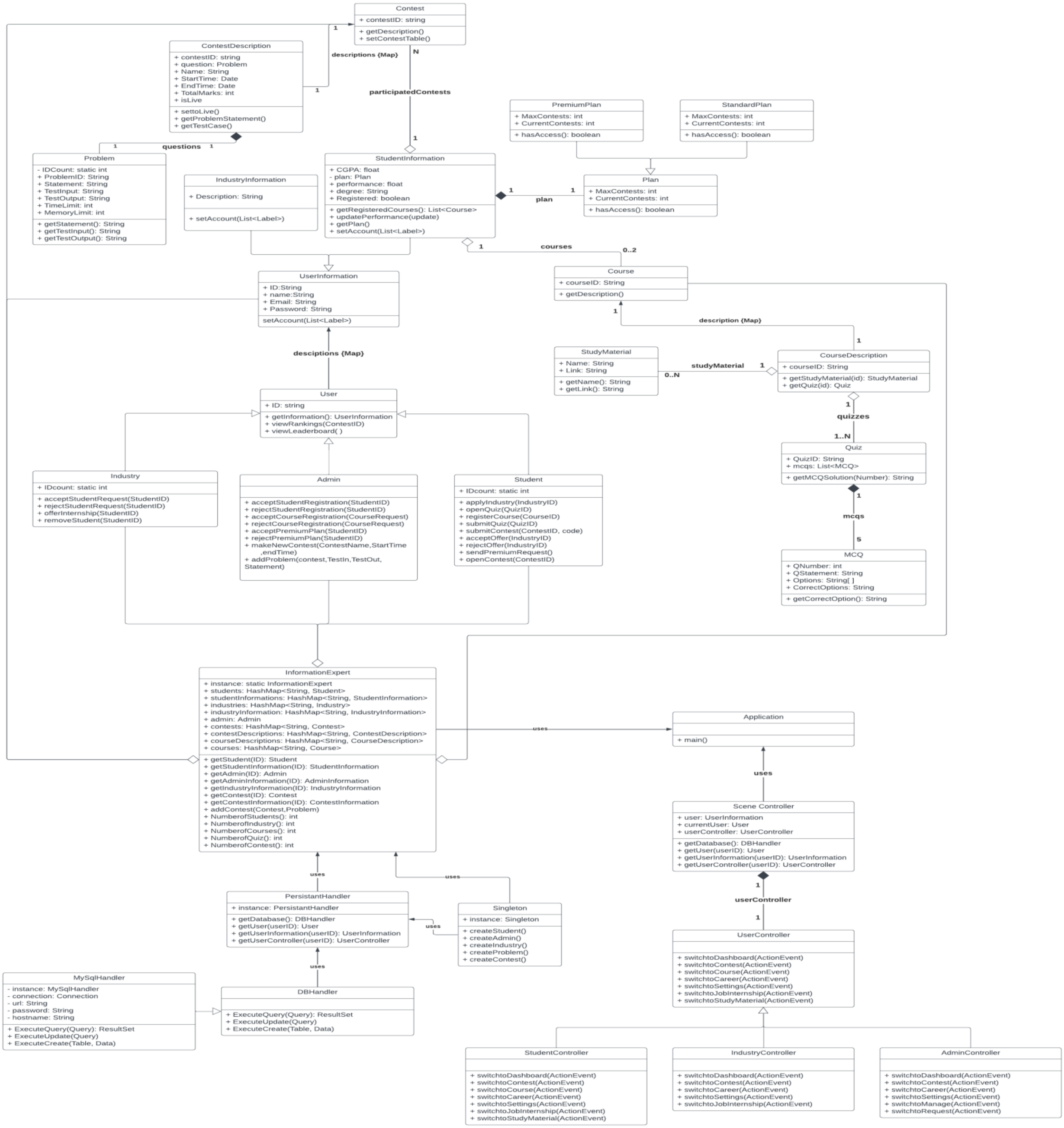
5. Manage for Job/Internship (Industry)



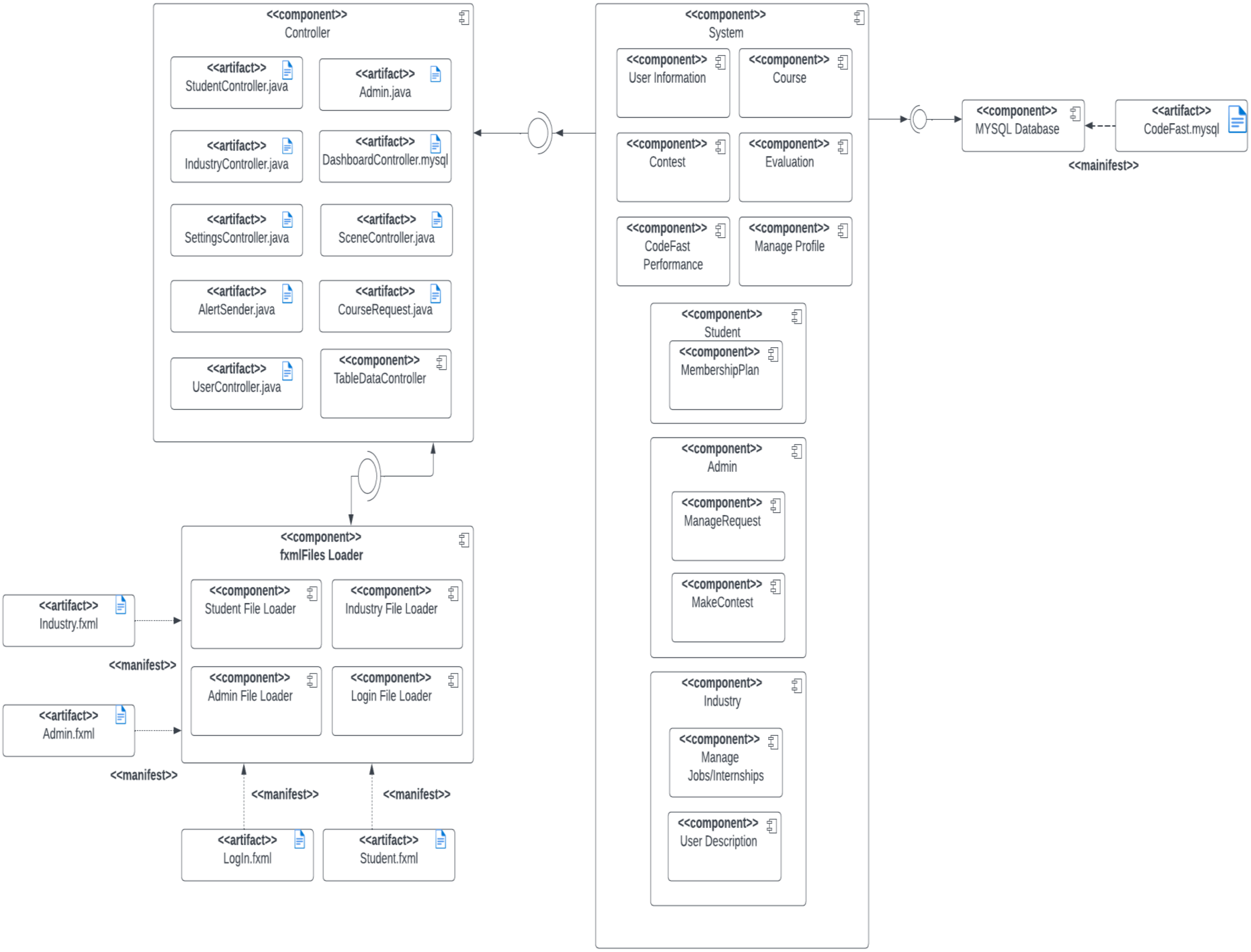
6. Offer for Job/Internship (Industry)



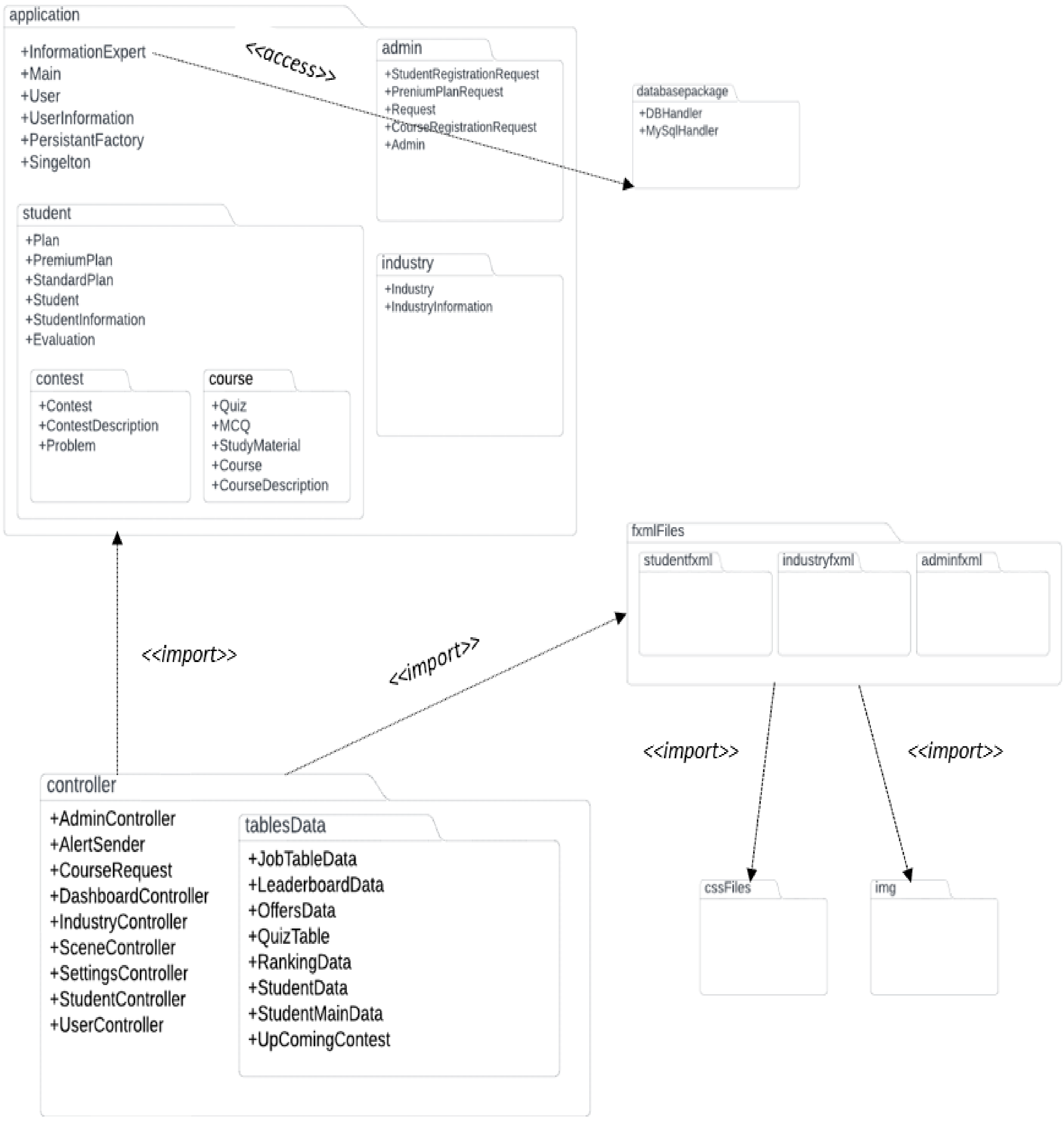
7. Class Diagram


Contest
+ contestID: string
+ getDescription()
+ setContestTable()
ContestDescription
+ contestID: string
+ question: Problem
+ Name: String
+ StartTime: Date
+ EndTime: Date
+ TotalMarks: int
+ isLive
+ settoLive()
+ getProblemStatement()
+ getTestCase()
1
descriptions {Map}
1
N
participatedContests
1
1
questions
1
Problem
- IDCount: static int
+ ProblemID: String
+ Statement: String
+ TestInput: String
+ TestOutput: String
+ TimeLimit: int
+ MemoryLimit: int
+ getStatement(): String
+ getTestInput(): String
+ getTestOutput(): String
IndustryInformation
+ Description: String
+ setAccount(List<Label>)
StudentInformation
+ CGPA: float
- plan: Plan
+ performance: float
+ degree: String
+ Registered: boolean
+ getRegisteredCourses(): List<Course>
+ updatePerformance(update)
+ getPlan()
+ setAccount(List<Label>)
PremiumPlan
+ MaxContests: int
+ CurrentContests: int
+ hasAccess(): boolean
StandardPlan
+ MaxContests: int
+ CurrentContests: int
+ hasAccess(): boolean
1
plan
1
Plan
+ MaxContests: int
+ CurrentContests: int
+ hasAccess(): boolean
1
courses
0..2
Course
+ courseID: String
+ getDescription()
1
description {Map}
1
UserInformation
+ ID:String
+ name:String
+ Email: String
+ Password: String
setAccount(List<Label>)
desciptions {Map}
StudyMaterial
+ Name: String
+ Link: String
+ getName(): String
+ getLink(): String
studyMaterial
0..N
1
CourseDescription
+ courseID: String
+ getStudyMaterial(id): StudyMaterial
+ getQuiz(id): Quiz
1
quizzes
1..N
User
+ ID: string
+ getInformation(): UserInformation
+ viewRankings(ContestID)
+ viewLeaderboard( )
Quiz
+ QuizID: String
+ mcqs: List<MCQ>
+ getMCQSolution(Number): String
1
mcqs
5
Industry
+ IDcount: static int
+ acceptStudentRequest(StudentID)
+ rejectStudentRequest(StudentID)
+ offerInternship(StudentID)
+ removeStudent(StudentID)
Admin
+ acceptStudentRegistration(StudentID)
+ rejectStudentRegistration(StudentID)
+ acceptCourseRegistration(CourseRequest)
+ rejectCourseRegistration(CourseRequest)
+ acceptPremiumPlan(StudentID)
+ rejectPremiumPlan(StudentID)
+ makeNewContest(ContestName,StartTime ,endTime)
+ addProblem(contest,TestIn,TestOut, Statement)
Student
+ IDcount: static int
+ applyIndustry(IndustryID)
+ openQuiz(QuizID)
+ registerCourse(CourseID)
+ submitQuiz(QuizID)
+ submitContest(ContestID, code)
+ acceptOffer(IndustryID)
+ rejectOffer(IndustryID)
+ sendPremiumRequest()
+ openContest(ContestID)
MCQ
+ QNumber: int
+ QStatement: String
+ Options: String[ ]
+ CorrectOptions: String
+ getCorrectOption(): String
InformationExpert
+ instance: static InformationExpert
+ students: HashMap<String, Student>
+ studentInformations: HashMap<String, StudentInformation>
+ industries: HashMap<String, Industry>
+ industryInformation: HashMap<String, IndustryInformation>
+ admin: Admin
+ contests: HashMap<String, Contest>
+ contestDescriptions: HashMap<String, ContestDescription>
+ courseDescriptions: HashMap<String, CourseDescription>
+ courses: HashMap<String, Course>
+ getStudent(ID): Student
+ getStudentInformation(ID): StudentInformation
+ getAdmin(ID): Admin
+ getAdminInformation(ID): AdminInformation
+ getIndustryInformation(ID): IndustryInformation
+ getContest(ID): Contest
+ getContestInformation(ID): ContestInformation
+ addContest(Contest,Problem)
+ NumberofStudents(): int
+ NumberofIndustry(): int
+ NumberofCourses(): int
+ NumberofQuiz(): int
+ NumberofContest(): int
uses
Application
+ main()
uses
Scene Controller
+ user: UserInformation
+ currentUser: User
+ userController: UserController
+ getDatabase(): DBHandler
+ getUser(userID): User
+ getUserInformation(userID): UserInformation
+ getUserController(userID): UserController
1
userController
1
uses
uses
PersistantHandler
+ instance: PersistantHandler
+ getDatabase(): DBHandler
+ getUser(userID): User
+ getUserInformation(userID): UserInformation
+ getUserController(userID): UserController
uses
Singleton
+ instance: Singleton
+ createStudent()
+ createAdmin()
+ createIndustry()
+ createProblem()
+ createContest()
uses
UserController
+ switchtoDashboard(ActionEvent)
+ switchtoContest(ActionEvent)
+ switchtoCourse(ActionEvent)
+ switchtoCareer(ActionEvent)
+ switchtoSettings(ActionEvent)
+ switchtoJobInternship(ActionEvent)
+ switchtoStudyMaterial(ActionEvent)
MySqlHandler
- instance: MySqlHandler
- connection: Connection
- url: String
- password: String
- hostname: String
+ ExecuteQuery(Query): ResultSet
+ ExecuteUpdate(Query)
+ ExecuteCreate(Table, Data)
DBHandler
+ ExecuteQuery(Query): ResultSet
+ ExecuteUpdate(Query)
+ ExecuteCreate(Table, Data)
StudentController
+ switchtoDashboard(ActionEvent)
+ switchtoContest(ActionEvent)
+ switchtoCourse(ActionEvent)
+ switchtoCareer(ActionEvent)
+ switchtoSettings(ActionEvent)
+ switchtoJobInternship(ActionEvent)
+ switchtoStudyMaterial(ActionEvent)
IndustryController
+ switchtoDashboard(ActionEvent)
+ switchtoContest(ActionEvent)
+ switchtoCareer(ActionEvent)
+ switchtoSettings(ActionEvent)
+ switchtoJobInternship(ActionEvent)
AdminController
+ switchtoDashboard(ActionEvent)
+ switchtoContest(ActionEvent)
+ switchtoCareer(ActionEvent)
+ switchtoSettings(ActionEvent)
+ switchtoManage(ActionEvent)
+ switchtoRequest(ActionEvent)

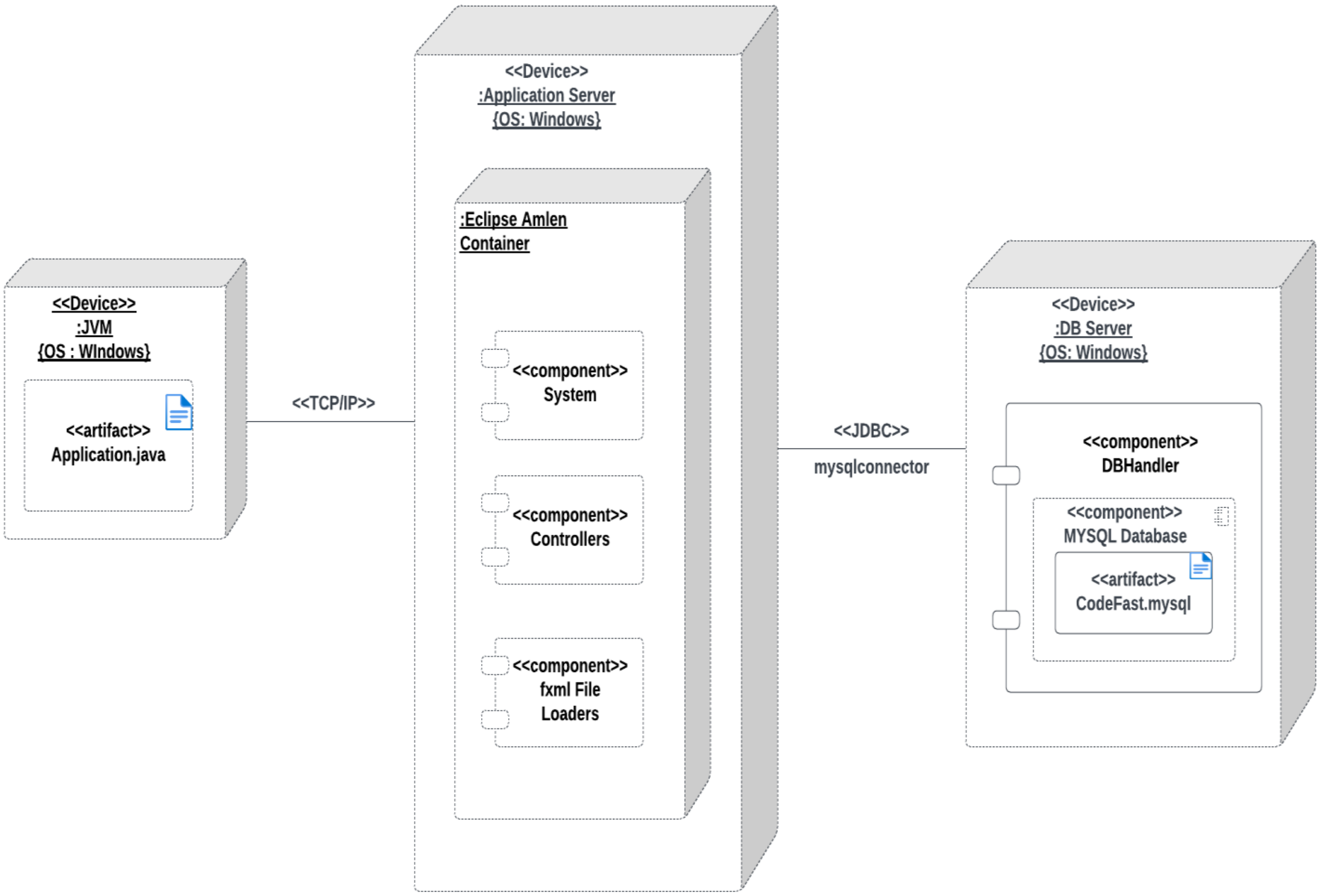
8. Component Diagram



9. Package Diagram



10. Deployment Diagram


<<Device>>
:JVM
{OS : WIndows}
<<artifact>>
Application.java
<<TCP/IP>>
<<Device>>
:Application Server
{OS: Windows}
:Eclipse Amlen
Container
<<component>>
System
<<component>>
Controllers
<<component>>
fxml File
Loaders
<<JDBC>>
mysqlconnector
<<Device>>
:DB Server
{OS: Windows}
<<component>>
DBHandler
<<component>>
MYSQL Database
<<artifact>>
CodeFast.mysql