Alexandria
Branch

Professional Web Development & Bi Track

Intake 44

ITI Examination System

MVC-SQL Server –EFCore&linq

“CODE FIRST”

Team:

- **Mahmoud Fathy Moweina**
- **Ahmed Adly**
- **Mohamed Mahmoud Abdelrahim**
- **David Ayad**
- **Abdelrahman Rehan**

Welcome to the ITI Online Examination System :

The ITI Online Examination System is a pioneering platform designed to enhance the educational experience for students and educators across all ITI branches in Egypt. This system is not just an examination tool; it's a gateway to a more efficient, flexible, and secure way of conducting and managing examinations. With our platform, you can access a wide range of examinations, receive instant feedback, and manage your examination schedule, all from the convenience of your home or anywhere with an internet connection.

Key Features:

- Accessibility Across Branches: Access exams from any ITI branch in Egypt, ensuring convenience and flexibility in your examination schedule.
- Instant Feedback: Receive detailed reports on your exam results instantly, helping you understand your performance and areas for improvement.
- Secure Environment: Conducted in a secure, encrypted environment, ensuring the integrity of the examination process.
- User-Friendly Interface: Designed for ease of use, making it simple for students and educators to navigate and manage their examinations.
- Specific Exam Times: Each exam has a specific time slot, allowing you to plan your study schedule around your exams.

Benefits:

- Convenience: Take your exams at your own pace and on your own time, without the need to travel or wait for exam schedules.
- Efficiency: Streamline the examination process, saving time and resources by conducting exams online.
- Flexibility: Accommodates various exam formats, catering to the diverse needs of ITI students.

- Cost-Effective: Reduce costs associated with traditional examination methods by taking exams online.

Getting Started:

To begin, create an account on our platform. Once registered, you can browse through the available examinations, schedule your exams according to the ITI schedule for each branch, and start preparing.

Welcome aboard, and we look forward to providing you with a seamless and rewarding online examination experience across all ITI branches in Egypt.

System Architecture Overview:

Technologies Used:

- **Backend:**
  - Microsoft ASP.NET MVC
  - Entity Framework with LINQ
  - SQL Server for database management
- **Frontend:**
  - HTML, CSS, JavaScript
  - Bootstrap for responsive design
  - jQuery for DOM manipulation
  - AJAX for asynchronous requests

System Components:

- Controllers: Handle user requests and interact with the model.
- Models: Represent the data structure of the application.
- View Models: Prepare data for views.
- Interfaces, Repositories: Implement solid principles and design patterns.

- Identity Authentication and Authorization: Secure user access.
- Validation: Ensure data integrity and consistency

User Guide for Different Roles:

1. User:

- **Login**: Enter your credentials to access the system.
- **Dashboard**: View your courses, exams, and grades.
- **Profile**: Update your personal information.
- **Logout**: Securely log out of the system.

2. Admin:

- **User Management**: Add, edit, or remove users from the system.
- **Role Assignment**: Assign roles and permissions to users.
- **System Configuration**: Manage system settings and configurations.
- **Reports**: Access system-wide reports and analytics.
- **Profile**: Update admin details.

3. Instructor:

- **Courses**: Create and manage course content.
- **Exams**: Design and schedule exams.
- **Grades**: Evaluate student performance.
- **Students**: Monitor student progress.
- **Profile**: Update personal details.

4. Student:

- **Courses**: Enroll in available courses.
- **Exams**: Take scheduled exams.
- **Grades**: View exam results and overall performance.
- **Profile**: Update personal information.

5. Branch Manager:

- **Branch Management**: Oversee branch operations.
- **Department Management**: Coordinate department activities.
- **Instructor Allocation**: Assign instructors to courses.
- **Reports**: Generate reports on branch performance.
- **Profile**: Update branch manager details.

6. Department Manager:

- **Department Overview**: Manage department resources.
- **Instructor Management**: Supervise department instructors.
- **Student Enrollment**: Monitor student enrollment in courses.
- **Reports**: Generate department-specific reports.
- **Profile**: Update department manager information.

7. Exam:

- **Create Exam**: Design new exams with questions and time limits.
- **Schedule Exam**: Set dates, times, and locations for exams.
- **Monitor Exam**: Supervise exam sessions and ensure security.
- **Evaluate Exam**: Review student responses and grade exams.
- **Reports**: Generate exam performance reports.

8. Question Types:

Multiple Choice: Select one correct answer from multiple options.

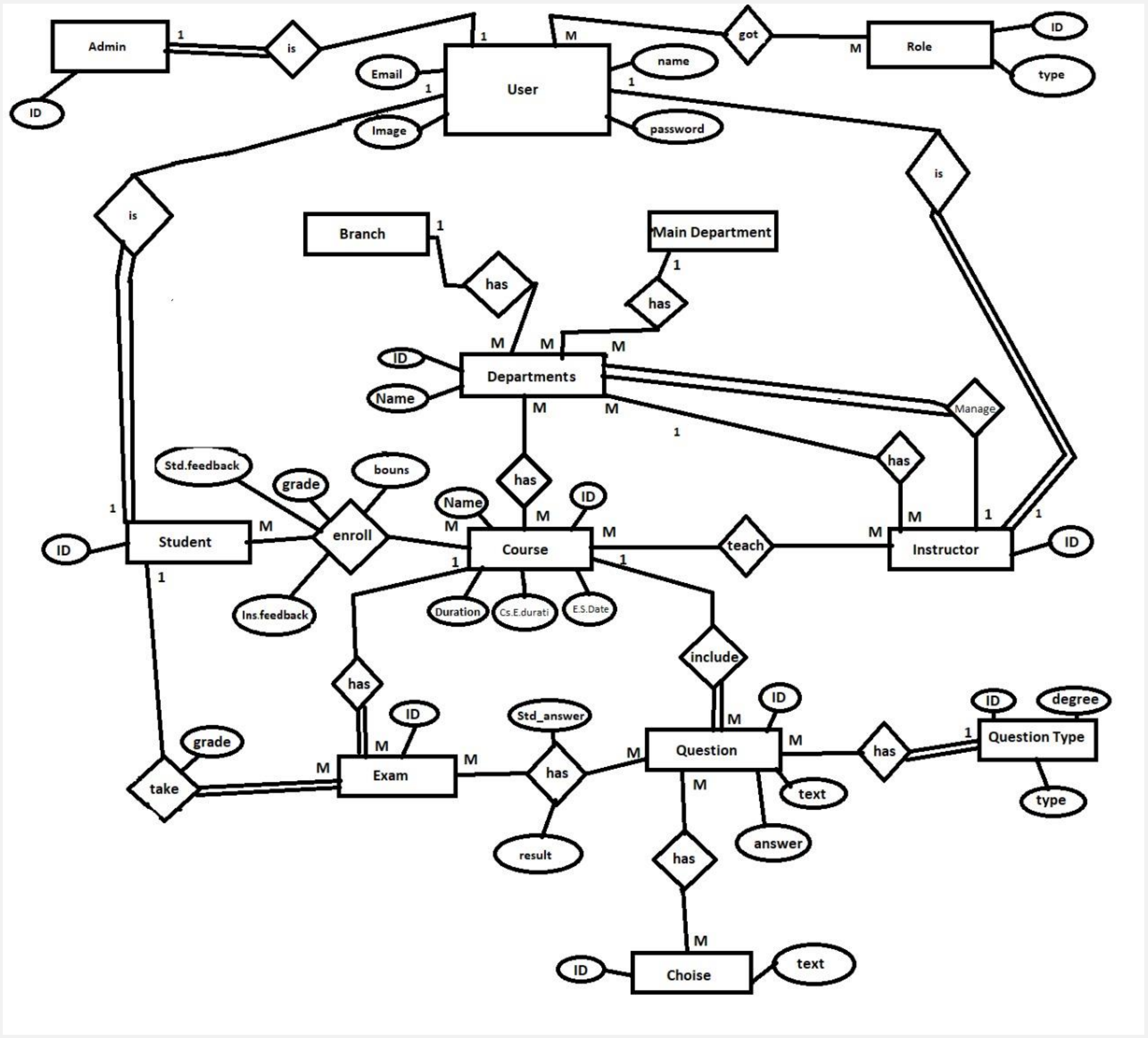
True/False: Determine the correctness of a statement.

Scoring and Grading Process:

- **Immediate Score Display**: Students can view their scores immediately after completing the exam.
- **Model Answer Comparison**: The system compares student answers to model answers for accurate scoring.
- **Feedback**: Students can access feedback on their performance to understand correct and incorrect responses.
- **Grade Reports**: Instructors can generate grade reports to analyze student performance on true/false and multiple-choice questions.

Database Architecture Approach

Entity-Relationship Diagram (ERD):



Entities:

User: Represents users with roles and relationships to other entities.

Role: Defines user roles in the system.

Branch: Represents ITI branches with relationships to departments.

Department: Contains department details and relationships to courses and instructors.

Course: Represents educational courses with relationships to exams.

Instructor: Users responsible for teaching courses.

Student: Users enrolled in courses.

Exam: Assessment component related to courses.

Question: Part of an exam with choices.

Question Type: Classification for types of questions.

Choice: Options for multiple-choice questions and T/F.

Relations:

User: This is a central entity with relationships to many other parts of the system. Attributes include ID, Name, Email, etc. It has various relationships with entities like Role, Branch, Department, Exam, and Course.

Role: This entity seems to define the roles of users in the system.

Branch: Represents a part of an organization, with a relationship indicating that it "has" Departments.

Main Department: An entity that indicates the primary departments within a branch.

Department: With attributes like ID and Name, this entity reflects specific departments, which have relationships with Courses and Instructors.

Course: An educational entity that has attributes like ID, Name, and Code. It is connected with Departments, Students, and Exams.

Instructor: A type of user with a special role related to Courses.

Student: Another role or type of user, related to Feedback and Enrollment.

Exam: Appears to be an assessment component related to a Course, with attributes and a relationship to Questions.

Question: Linked to an Exam and having its own set of Choices. There are also attributes and a relationship to Question Type.

Question Type: A classification for the type of question, related to the Question entity.

Choice: An entity probably representing multiple-choice options for a given Question.

These entities are interlinked, and most of these connections have cardinality indicators (like 1, M, or 1..*) that denote the types of relationships (one-to-one, one-to-many, etc.) between the entities.

System Functionality:

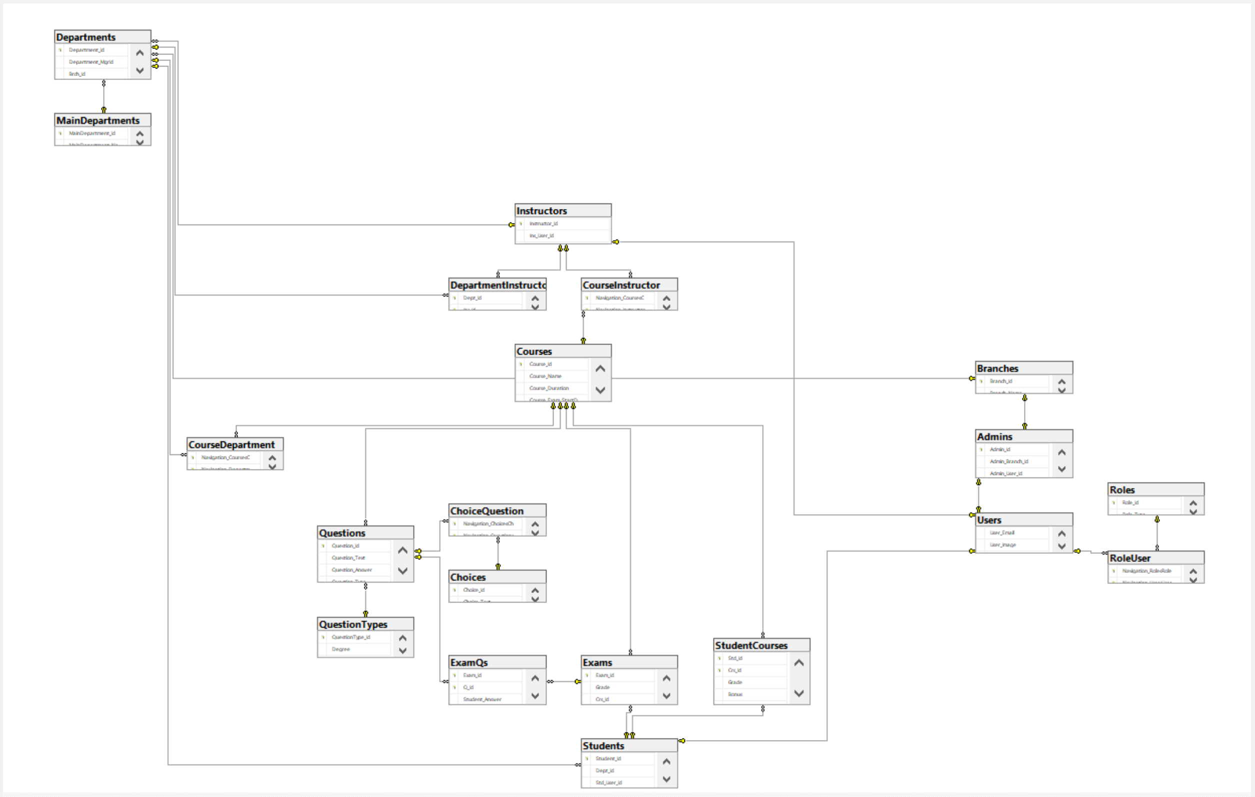
ITI Branches: Managed by Branch Managers.

Departments: Managed by Department Managers, with Instructors and Students.

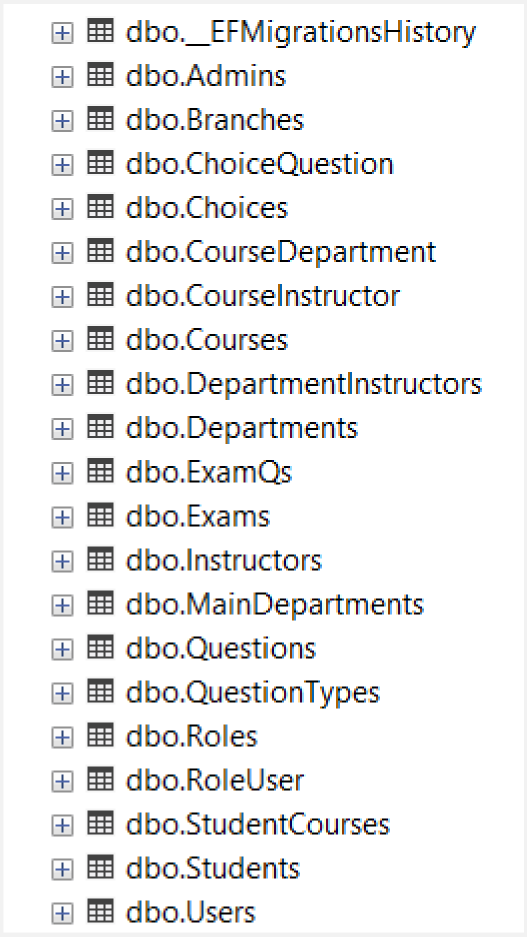
Courses, Exams, Questions, Choices: Educational components.

Admin Controls: Manage users, roles, and system functionality.

Database Diagram:



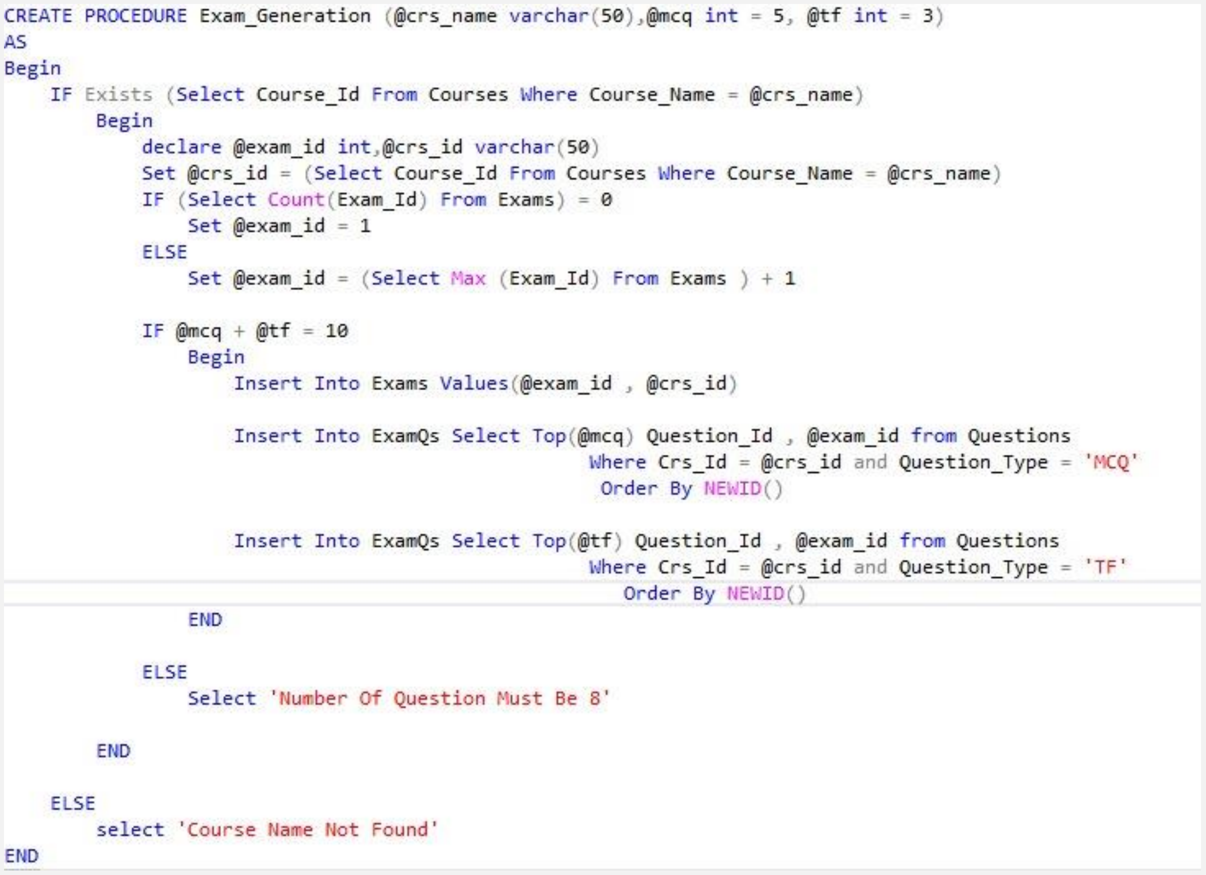
Tables:



dbo.__EFMigrationsHistory
dbo.Admins
dbo.Branches
dbo.ChoiceQuestion
dbo.Choices
dbo.CourseDepartment
dbo.CourseInstructor
dbo.Courses
dbo.DepartmentInstructors
dbo.Departments
dbo.ExamQs
dbo.Exams
dbo.Instructors
dbo.MainDepartments
dbo.Questions
dbo.QuestionTypes
dbo.Roles
dbo.RoleUser
dbo.StudentCourses
dbo.Students
dbo.Users

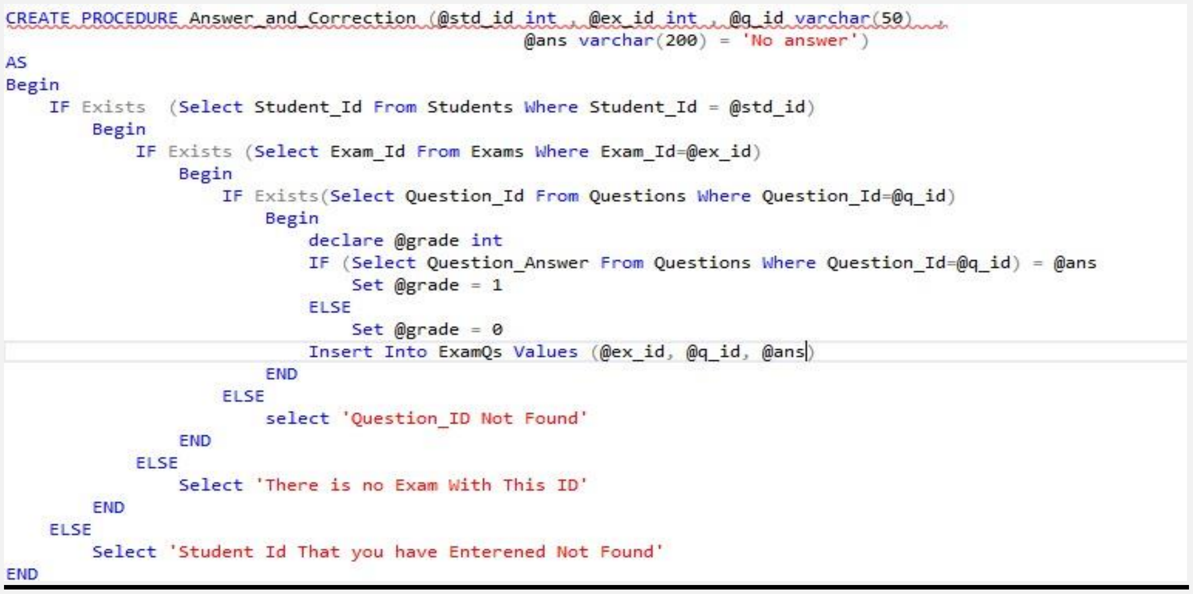
Stored Procedures

Exam Generation:



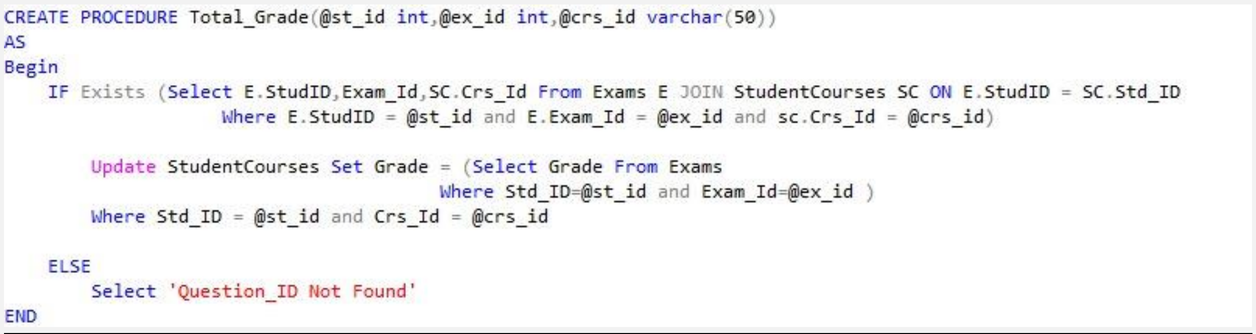
```sql
CREATE PROCEDURE Exam_Generation (@crs_name varchar(50),@mcq int = 5, @tf int = 3)
AS
Begin
    IF Exists (Select Course_Id From Courses Where Course_Name = @crs_name)
        Begin
            declare @exam_id int,@crs_id varchar(50)
            Set @crs_id = (Select Course_Id From Courses Where Course_Name = @crs_name)
            IF (Select Count(Exam_Id) From Exams) = 0
                Set @exam_id = 1
            ELSE
                Set @exam_id = (Select Max (Exam_Id) From Exams ) + 1

            IF @mcq + @tf = 10
                Begin
                    Insert Into Exams Values(@exam_id , @crs_id)

                    Insert Into ExamQs Select Top(@mcq) Question_Id , @exam_id from Questions
                                                Where Crs_Id = @crs_id and Question_Type = 'MCQ'
                                                 Order By NEWID()

                    Insert Into ExamQs Select Top(@tf) Question_Id , @exam_id from Questions
                                                Where Crs_Id = @crs_id and Question_Type = 'TF'
                                                   Order By NEWID()
                END

            ELSE
                Select 'Number Of Question Must Be 8'

        END

    ELSE
        select 'Course Name Not Found'
END
```

Answer&Correction:



```sql
CREATE PROCEDURE Answer_and_Correction (@std_id int , @ex_id int , @q_id varchar(50) ,
                                         @ans varchar(200) = 'No answer')
AS
Begin
    IF Exists  (Select Student_Id From Students Where Student_Id = @std_id)
        Begin
            IF Exists (Select Exam_Id From Exams Where Exam_Id=@ex_id)
                Begin
                    IF Exists(Select Question_Id From Questions Where Question_Id=@q_id)
                        Begin
                            declare @grade int
                            IF (Select Question_Answer From Questions Where Question_Id=@q_id) = @ans
                                Set @grade = 1
                            ELSE
                                Set @grade = 0
                            Insert Into ExamQs Values (@ex_id, @q_id, @ans)
                        END
                    ELSE
                        select 'Question_ID Not Found'
                END
            ELSE
                Select 'There is no Exam With This ID'
        END
    ELSE
        Select 'Student Id That you have Enterened Not Found'
END
```

Total Grade:



```
CREATE PROCEDURE Total_Grade(@st_id int,@ex_id int,@crs_id varchar(50))
AS
Begin
    IF Exists (Select E.StudID,Exam_Id,SC.Crs_Id From Exams E JOIN StudentCourses SC ON E.StudID = SC.Std_ID
                Where E.StudID = @st_id and E.Exam_Id = @ex_id and sc.Crs_Id = @crs_id)

        Update StudentCourses Set Grade = (Select Grade From Exams
                                    Where Std_ID=@st_id and Exam_Id=@ex_id )
        Where Std_ID = @st_id and Crs_Id = @crs_id

    ELSE
        Select 'Question_ID Not Found'
END
```

MVC Architecture Approach:

Interfaces:

1. IAdminRepo

- **Purpose**: Defines methods for managing administrative data, such as creating, updating, deleting, and retrieving administrative users.
- **Methods**: Typically includes methods like GetAllAdmins(), GetAdminById(int id), AddAdmin(Admin admin), UpdateAdmin(Admin admin), and DeleteAdmin(int id).

2. IBranchRepo

- **Purpose**: Manages operations related to branches or locations where exams are conducted.
- **Methods**: Methods might include GetAllBranches(), GetBranchById(int id), AddBranch(Branch branch), UpdateBranch(Branch branch), and DeleteBranch(int id).

3. IChoiceRepo

- **Purpose**: Handles choices or options in multiple-choice questions.
- **Methods**: Methods could include GetAllChoices(), GetChoiceById(int id), AddChoice(Choice choice), UpdateChoice(Choice choice), and DeleteChoice(int id).

4. ICourseRepo

- **Purpose**: Manages courses, including their creation, update, deletion, and retrieval.
- **Methods**: Methods might include GetAllCourses(), GetCourseById(int id), AddCourse(Course course), UpdateCourse(Course course), and DeleteCourse(int id).

5. IDepartmentRepo

- **Purpose**: Deals with departments or academic divisions within the institution.
- **Methods**: Methods could include GetAllDepartments(), GetDepartmentById(int id), AddDepartment(Department department), UpdateDepartment(Department department), and DeleteDepartment(int id).

6. IDeptInstructorRepo

- **Purpose**: Manages the relationship between departments and instructors, possibly including the assignment of instructors to departments.
- **Methods**: Methods might include GetAllDeptInstructors(), GetDeptInstructorById(int id), AddDeptInstructor(DeptInstructor deptInstructor), UpdateDeptInstructor(DeptInstructor deptInstructor), and DeleteDeptInstructor(int id).

7. IExamQuestionRepo

- **Purpose**: Handles the association between exams and questions, allowing for the creation of exam questions.
- **Methods**: Methods could include GetAllExamQuestions(), GetExamQuestionById(int id), AddExamQuestion(ExamQuestion examQuestion), UpdateExamQuestion(ExamQuestion examQuestion), and DeleteExamQuestion(int id).

8. IExamRepo

- **Purpose**: Manages exams, including their creation, update, deletion, and retrieval.
- **Methods**: Methods might include GetAllExams(), GetExamById(int id), AddExam(Exam exam), UpdateExam(Exam exam), and DeleteExam(int id).

9. IInstructorRepo

- **Purpose**: Handles operations related to instructors, such as creating, updating, deleting, and retrieving instructor data.
- **Methods**: Methods could include GetAllInstructors(), GetInstructorById(int id), AddInstructor(Instructor instructor), UpdateInstructor(Instructor instructor), and DeleteInstructor(int id).

10. IMainDeptRepo

- **Purpose**: Manages main departments or primary academic divisions within the institution.
- **Methods**: Methods might include GetAllMainDepts(), GetMainDeptById(int id), AddMainDept(MainDept mainDept), UpdateMainDept(MainDept mainDept), and DeleteMainDept(int id).

11. IQuestionRepo

- **Purpose**: Handles operations related to questions, including their creation, update, deletion, and retrieval.
- **Methods**: Methods could include GetAllQuestions(), GetQuestionById(int id), AddQuestion(Question question), UpdateQuestion(Question question), and DeleteQuestion(int id).

12. IQuestionTypeRepo

- **Purpose**: Manages question types, such as multiple-choice, true/false, etc.
- **Methods**: Methods might include GetAllQuestionTypes(), GetQuestionTypeById(int id), AddQuestionType(QuestionType questionType), UpdateQuestionType(QuestionType questionType), and DeleteQuestionType(int id).

13. IRoleRepo

- **Purpose**: Handles roles within the system, such as student, instructor, or admin.
- **Methods**: Methods could include GetAllRoles(), GetRoleById(int id), AddRole(Role role), UpdateRole(Role role), and DeleteRole(int id).

14. IStudentCourseRepo

- **Purpose**: Manages the relationship between students and courses, including enrollment and course completion.
- **Methods**: Methods might include GetAllStudentCourses(), GetStudentCourseById(int id), AddStudentCourse(StudentCourse studentCourse), UpdateStudentCourse(StudentCourse studentCourse), and DeleteStudentCourse(int id).

15. IStudentRepo

- Purpose: Handles operations related to students, such as creating, updating, deleting, and retrieving student data.
- Methods: Methods could include GetAllStudents(), GetStudentById(int id), AddStudent(Student student), UpdateStudent(Student student), and DeleteStudent(int id).

16. IUserRepo

- **Purpose**: Manages user accounts, including authentication and authorization.
- **Methods**: Methods might include GetAllUsers(), GetUserById(int id), AddUser(User user), UpdateUser(User user), and DeleteUser(int id).

These interfaces and their corresponding repositories play a crucial role in the application's architecture, adhering to the Dependency Inversion Principle and facilitating a clean separation of concerns. This approach makes the application more maintainable, testable, and scalable.

<u>Repositories:</u>

AdminRepo

- **Purpose:** Manages administrative data and related entities.
- **Key Methods:**
- Add(Admin admin): Adds a new admin.
- **Delete(int id):** Deletes an admin by ID.
- **GetAll():** Retrieves all admins.
- **GetAllAdminsbyUsersEmailBranches():** Retrieves all admins with their associated user and branch details.
- **GetById(int id):** Retrieves a specific admin by ID, including related entities.
- **GetByIdfordetails(int id):** Retrieves a specific admin by ID for detailed viewing, including related entities.
- **Update(Admin admin):** Updates an existing admin.
- **AddRole(int id):** Adds an "Admin" role to an admin entity.

BranchRepo

- **Purpose:** Manages branch data.
- **Key Methods:**
- **GetAll():** Retrieves all branches.
- **GetById(int id):** Retrieves a specific branch by ID.
- **Add(Branch branch):** Adds a new branch.
- **Update(Branch branch):** Updates an existing branch.
- **Delete(int id):** Deletes a specific branch.

ChoiceRepo

- **Purpose:** Manages choice data.
- **Key Methods:**

- **GetAll():** Retrieves all choices.
- **GetById(int id):** Retrieves a specific choice by ID.
- **Add(Choice choice):** Adds a new choice.
- **Update(Choice choice):** Updates an existing choice.
- **Delete(int id):** Deletes a specific choice.

CourseRepo

- **Purpose:** Manages course data and related entities.
- **Key Methods:**
- **Add(Course course):** Adds a new course.
- **Delete(int id):** Deletes a course by ID.
- **GetAll():** Retrieves all courses.
- **GetById(int id):** Retrieves a specific course by ID, including related entities.
- **Update(Course course):** Updates an existing course.

DepartmentRepo

- **Purpose:** Manages department data and related entities.
- **Key Methods:**
- **GetAll():** Retrieves all departments with their associated entities.
- **GetDepartmentsByBranchId(int id):** Retrieves all departments associated with a specific branch ID.
- **GetById(int id):** Retrieves a specific department by ID, including related entities.
- **Add(Department department):** Adds a new department.
- **Update(Department department):** Updates an existing department.
- **Delete(int id):** Deletes a department by ID, if it has no associated students, instructors, or courses.

- **GetByBranchAndMainDepartment(int branchId, int mainDepId):** Retrieves a department by branch ID and main department ID.
- **GetCourses(int id):** Retrieves all courses associated with a specific department ID.

DeptInstructorRepo

- **Purpose:** Manages the relationship between departments and instructors.
- **Key Methods:**
- **GetDepartmentsByInsId(int id):** Retrieves all departments associated with a specific instructor ID.
- **GetInstructorsByDeptId(int id):** Retrieves all instructors associated with a specific department ID.
- **Add(int depId, int InsId):** Adds a new association between a department and an instructor.
- **Delete(int depId, int InsId):** Deletes the association between a department and an instructor.

ExamQuestionRepo

- **Purpose:** Manages exam questions and related data.
- **Key Methods:**
- **GetExamQuestions(int id):** Retrieves all questions for a specific exam.
- **GenerateExam(int crsId, int stdId):** Generates a new exam for a course and student.
- **GetByIds(int examId, int questionId):** Retrieves a specific question for an exam.
- **CheckAnswer(ExamQs examQs):** Checks the answer for a specific question and updates grades accordingly.

ExamRepo

- **Purpose:** Manages exam data.
- **Key Methods:**

- **GetAll():** Retrieves all exams.
- **GetById(int id):** Retrieves a specific exam by ID.
- **GetAllByCrsId(int id):** Retrieves all exams for a specific course.
- **GetAllByStdId(int id):** Retrieves all exams for a specific student.
- **Add(Exam exam):** Adds a new exam.
- **Update(Exam exam):** Updates an existing exam.
- **Delete(int id):** Deletes a specific exam.

InstructorRepo

- **Purpose:** Manages instructor data and related entities.
- Key Methods:
- **GetAll():** Retrieves all instructors.
- **GetById(int id):** Retrieves a specific instructor by ID, including related entities.
- **Add(Instructor instructor):** Adds a new instructor.
- **Update(Instructor instructor):** Updates an existing instructor.
- **Delete(int id):** Deletes a specific instructor.
- **GetCourses(int id):** Retrieves all courses for a specific instructor.
- **AddCourse(int insId, Course course):** Adds a course to an instructor.
- **RemoveCourse(int insId, Course course):** Removes a course from an instructor.
- **AddRole(int id):** Adds a role to an instructor.

MainDeptRepo

- **Purpose:** Manages main department data.
- **Key Methods:**
- **GetAll():** Retrieves all main departments.
- **GetById(int id):** Retrieves a specific main department by ID.

- **Add(MainDepartment mainDepartment):** Adds a new main department.
- **Update(MainDepartment mainDepartment):** Updates an existing main department.
- **Delete(int id):** Deletes a specific main department.

QuestionRepo

- **Purpose:** Manages question data and related entities.
- **Key Methods:**
- **GetAll():** Retrieves all questions.
- **GetById(int id**): Retrieves a specific question by ID, including related entities.
- **Add(Question question):** Adds a new question.
- **Update(Question question):** Updates an existing question.
- **Delete(int id):** Deletes a specific question.

QuestionTypeRepo

- **Purpose:** Manages question type data.
- **Key Methods:**
- **GetAll():** Retrieves all question types.
- **GetById(int id):** Retrieves a specific question type by ID.
- **Add(QuestionType questionType):** Adds a new question type.
- **Update(QuestionType questionType):** Updates an existing question type.
- **Delete(int id):** Deletes a specific question type.

RoleRepo

- **Purpose:** Manages role data and related entities.
- **Key Methods:**
- **GetAll():** Retrieves all roles.
- **GetById(int id):** Retrieves a specific role by ID, including related entities.

- **Add(Role role):** Adds a new role.
- **Update(Role role):** Updates an existing role.
- **Delete(int id):** Deletes a specific role.
- **GetUsers(int id):** Retrieves all users for a specific role.

StudentCourseRepo

- **Purpose:** Manages the relationship between students and courses.
- **Key Methods:**
- **GetByIds(int crsId, int stdId):** Retrieves a specific student-course relationship by IDs.
- **GetCourseFeedbacks(int courseId):** Retrieves all feedbacks for a specific course.
- **GetCourseStudents(int id):** Retrieves all students for a specific course.
- **GetStudentCourseDetails(int studentId, int courseId):** Retrieves details of a student-course relationship.
- **GetStudentCourses(int id):** Retrieves all courses for a specific student.
- **GetStudentFeedbacks(int studentId):** Retrieves all feedbacks for a specific student.
- **UpdateStudentFeedback(int studentId, int courseId, string feedback):** Updates feedback for a student-course relationship.
- **Add(int studentId, int courseId):** Adds a new student-course relationship.
- **GetStudentCourseList(int courseId):** Retrieves a list of student-course relationships for a specific course.

UserRepo

- **Purpose:** Manages user data and related entities.
- **Key Methods:**
- **GetAll():** Retrieves all users.
- **GetById(int id):** Retrieves a specific user by ID, including related entities.

- **Add(User user):** Adds a new user.
- **Update(User user):** Updates an existing user.
- **Delete(int id):** Deletes a specific user.
- **AddImage(User user, IFormFile image):** Adds an image to a user's profile.
- **GetNonAssignedUsers():** Retrieves users not assigned to any role.
- **Login(string email, string password):** Attempts to log in a user.

Models:

Admin Model

- **Purpose:** Represents an admin user within the ITI Online Examination System.
- **Key Properties:**
- **Admin_Id:** Unique identifier for the admin.
- **Admin_User_Id:** Foreign key referencing the user account associated with the admin.
- **Admin_Branch_Id:** Foreign key referencing the branch the admin is associated with.
- **Navigation Properties:**
- **Navigation_User:** Relates to the user account associated with the admin.
- **Navigation_Branch:** Relates to the branch the admin is associated with.

Branch Model

- **Purpose:** Represents a branch within the ITI Online Examination System.
- **Key Properties:**

- **Branch_Id:** Unique identifier for the branch.
- **Branch_Name:** Name of the branch.
- **Branch_Manager_Name:** Name of the branch manager.
- **Navigation Properties:**
- **Navigation_Departments:** Relates to the departments associated with the branch.
- **Navigation_Admin:** Relates to the admin user associated with the branch.

Choice Model

- **Purpose:** Represents a choice in the ITI Online Examination System, typically used in multiple-choice questions.
- **Key Properties:**
- **Choice_Id:** Unique identifier for the choice.
- **Choice_Text:** Text of the choice.
- **Navigation Properties:**
- **Navigation_Questions:** Relates to the questions associated with the choice.

Department Model

- **Purpose:** Represents a department within the ITI Online Examination System.
- **Key Properties:**
- **Department_Id:** Unique identifier for the department.
- **Department_Name:** Name of the department.
- **Department_MgrId:** Foreign key referencing the department manager (an instructor).
- **Brch_Id:** Foreign key referencing the branch the department belongs to.
- **MainDept_Id:** Foreign key referencing the main department (if applicable).

- **Navigation Properties:**
- **Navigation_Department_Manager_Instructor:** Relates to the instructor who manages the department.
- **Navigation_Department_Instructor:** Relates to the department-instructor associations.
- **Navigation_Courses:** Relates to the courses offered by the department.
- **Navigation_Students:** Relates to the students enrolled in the department.
- **Navigation_Branch:** Relates to the branch the department is part of.
- **Navigation_MainDepartment:** Relates to the main department (if applicable).

DepartmentInstructors Model

- **Purpose:** Represents the many-to-many relationship between departments and instructors.
- **Key Properties:**
- **Dept_Id:** Foreign key referencing the department.
- **Ins_Id:** Foreign key referencing the instructor.
- **Navigation Properties:**
- **Navigation_Instructor:** Relates to the instructor associated with the department.
- **Navigation_Department:** Relates to the department associated with the instructor.

ErrorViewModel

- **Purpose:** Represents a view model for error handling, providing a structured way to manage and display error information to the user.
- **Key Properties:**
- **RequestId:** Nullable string representing the request ID associated with the error.

- **ShowRequestId:** Boolean indicating whether the request ID should be displayed.

Course Model

- **Purpose:** Represents a course within the ITI Online Examination System.
- **Key Properties:**
- **Course_Id:** Unique identifier for the course.
- **Course_Name:** Name of the course.
- **Course_Duration:** Duration of the course.
- **Crs_Exam_Duration:** Duration of the course examination.
- **Course_Exam_StartDate:** Start date and time of the course examination.
- **Navigation Properties:**
- **Navigation_Instructors:** Relates to the instructors associated with the course.
- **Navigation_Departments:** Relates to the departments offering the course.
- **Navigation_StudentCourses:** Relates to the student-course associations.
- **Navigation_Exam:** Relates to the exams associated with the course.
- **Navigation_Question:** Relates to the questions associated with the course.

Exam

- **Purpose:** Represents an exam in the system.
- **Key Properties:**
- **Exam_Id:** Primary key, identifies the exam.
- **Grade:** Optional, represents the grade achieved in the exam.
- **Crs_Id:** Optional foreign key, links to the course associated with the exam.
- **StudId:** Optional foreign key, links to the student who took the exam.
- **Navigation Properties:**

- **Navigation_Student:** Links to the student entity.
- **Navigation_Course:** Links to the course entity.
- **Navigation_ExamQs:** Links to the exam questions entity.

Exam_Context

- **Purpose:** Defines the database context for the ITI Online Examination System.
- **Key Components:**
- DbSet properties for each entity in the system.
- Constructors for creating instances of the context.
- OnModelCreating method for configuring the model, including defining composite keys and unique indexes.

ExamQs

- **Purpose:** Represents the relationship between an exam and a question, including the student's answer and the result.
- **Key Properties:**
- **Exam_Id:** Optional foreign key, links to the exam entity.
- **Q_Id:** Optional foreign key, links to the question entity.
- **Student_Answer:** Represents the student's answer to the question.
- **Result:** Represents the result of the question.
- **Navigation Properties:**
- **Navigation_Question:** Links to the question entity.
- **Navigation_Exam:** Links to the exam entity.

Instructor

- **Purpose:** Represents an instructor in the system.
- **Key Properties:**
- **Instructor_Id:** Primary key, identifies the instructor.

- **Ins_User_Id:** Optional foreign key, links to the user entity associated with the instructor.
- **Navigation Properties:**
- **Navigation_Departments:** Links to the department entities associated with the instructor.
- **Navigation_Department_Instructor:** Links to the department-instructor association entities.
- **Navigation_Courses:** Links to the course entities associated with the instructor.
- **Navigation_User:** Links to the user entity associated with the instructor.

MainDepartment

- **Purpose:** Represents a main department in the system.
- **Key Properties:**
- **MainDepartment_Id:** Primary key, identifies the main department.
- **MainDepartment_Name:** Represents the name of the main department.
- **Navigation Properties:**
- **Navigation_Departments:** Links to the department entities associated with the main department.

Question

- **Purpose:** Represents a question in the system.
- **Key Properties:**
- **Question_Id:** Primary key, identifies the question.
- **Question_Text:** Represents the text of the question.
- **Question_Answer:** Represents the correct answer to the question.
- **Question_Type:** Optional foreign key, links to the question type entity.
- **Crs_Id:** Optional foreign key, links to the course entity.
- **Navigation Properties:**

- **Navigation_ExamQs:** Links to the exam questions entity.
- **Navigation_QuestionType:** Links to the question type entity.
- **Navigation_Course:** Links to the course entity.
- **Navigation_Choices:** Links to the choice entities associated with the question.

QuestionType

- **Purpose:** Represents a type of question in the system.
- **Key Properties:**
- **QuestionType_Id:** Primary key, identifies the question type.
- **Degree:** Represents the degree of difficulty associated with the question type.
- **Type:** Represents the type of the question (e.g., MCQ, T&F).
- **Navigation Properties:**
- **Navigation_Question:** Links to the question entities associated with the question type.

Role

- **Purpose:** Represents a role in the system.
- **Key Properties:**
- **Role_Id:** Primary key, identifies the role.
- **Role_Type:** Represents the type of the role (e.g., Admin, Instructor, Student).
- **Navigation Properties:**
- **Navigation_Users:** Links to the user entities associated with the role.

Student

- **Purpose:** Represents a student in the system.
- **Key Properties:**
- **Student_Id:** Primary key, identifies the student.

- **Std_User_Id:** Optional foreign key, links to the user entity associated with the student.
- **Dept_Id:** Optional foreign key, links to the department entity.
- **Navigation Properties:**
- **Navigation_Department:** Links to the department entity associated with the student.
- **Navigation_StudentCourses:** Links to the student course entities associated with the student.
- **Navigation_StudentExam:** Links to the exam entities associated with the student.
- **Navigation_User:** Links to the user entity associated with the student.

StudentCourse

- **Purpose:** Represents the relationship between a student and a course, including details about the student's enrollment.
- **Key Properties:**
- **Std_Id:** Optional foreign key, links to the student entity.
- **Crs_Id:** Optional foreign key, links to the course entity.
- **Grade:** Optional, represents the grade achieved by the student in the course.
- **Bonus:** Represents any bonus points awarded to the student for the course.
- **Ins_Feedback:** Represents feedback from the instructor about the student's performance in the course.
- **Std_Feedback:** Represents feedback from the student about the course.
- **HasTakenExam:** Indicates whether the student has taken the exam for the course.
- **Navigation Properties:**
- **Navigation_Course:** Links to the course entity.
- **Navigation_Student:** Links to the student entity.

11. User

- **Purpose:** Represents a user in the system, which could be an instructor, student, or admin.
- **Key Properties:**
- **User_Id:** Primary key, identifies the user.
- **User_Name:** Represents the name of the user.
- **User_Email:** Represents the email address of the user.
- **User_Image:** Represents the image associated with the user.
- **User_Password:** Represents the password of the user.
- **Navigation Properties:**
- **Navigation_Roles:** Links to the role entities associated with the user.
- **Navigation_Instructor:** Links to the instructor entity associated with the user.
- **Navigation_Student:** Links to the student entity associated with the user.
- **Navigation_Admin:** Links to the admin entity associated with the user.

<u>Controllers:</u>

AccountController

- **Purpose:** Manages user authentication and authorization, including login, logout, and displaying user details based on their role.
- **Key Methods:**
- **Login (GET):** Displays the login view.
- **Login (POST):** Handles the login process, validates credentials, and sets user claims.
- **LogOut:** Handles the logout process.
- **MyDetails:** Displays user details based on their role.

AdminController

- **Purpose:** Manages admin users, including listing all admins, adding new branches, adding new admins, editing existing admins, and viewing admin details.
- **Key Methods:**
- **Index: Lists all admins.**
- **AddnewBranch:** Displays the view for adding a new branch.
- **Add (GET):** Displays the view for adding a new admin.
- **Save (POST):** Handles the addition of a new admin.
- **Edit (GET):** Displays the view for editing an existing admin.
- **Edit (POST):** Handles the editing of an existing admin.
- **Details:** Displays the details of a specific admin.

BranchController

- **Purpose:** Manages branches, including listing all branches, adding new branches, deleting branches, editing existing branches, and viewing branch details.
- **Key Methods:**
- **Index:** Lists all branches.
- **AddBranch:** Displays the view for adding a new branch.
- **Save (POST):** Handles the addition of a new branch.
- **Delete:** Deletes a specific branch.
- **Edit (GET):** Displays the view for editing an existing branch.
- **Update (POST):** Handles the editing of an existing branch.
- **Details:** Displays the details of a specific branch.

CourseController

- Purpose: Manages courses, including listing all courses, displaying students enrolled in a course, creating new courses, editing existing courses, viewing course details, and deleting courses.
- **Key Methods:**
- **Index:** Lists all courses.
- **DisplayStudents:** Displays the students enrolled in a specific course.
- **Create:** Displays the view for creating a new course.
- **Create (POST):** Handles the creation of a new course.
- **Details:** Displays the details of a specific course.
- **Edit (GET):** Displays the view for editing an existing course.
- **Edit (POST):** Handles the editing of an existing course.
- **Delete:** Deletes a specific course.

DepartmentController

- **Purpose:** Manages departments, including listing all departments, displaying department details, managing courses within departments, creating new departments, editing existing departments, and deleting departments.
- **Key Methods:**
- **Index:** Lists all departments.
- **Details:** Displays the details of a specific department.
- **DisplayStudents:** Displays the students enrolled in a specific department.
- **DisplayInstructors:** Displays the instructors associated with a specific department.
- **DisplayCourses:** Displays the courses offered by a specific department.
- **Create:** Displays the view for creating a new department.

- **Create (POST):** Handles the creation of a new department.
- **Edit (GET):** Displays the view for editing an existing department.
- **Edit (POST):** Handles the editing of an existing department.
- **Delete:** Deletes a specific department.
- **ManageCourses:** Displays the courses that can be added or removed from a specific department.
- **ManageCourses (POST):** Handles the addition or removal of courses from a specific department.

ExamController

- **Purpose:** Manages exams, including taking exams, displaying exam templates, and handling the submission of exam answers.
- **Key Methods:**
- **TakeExam (GET):** Displays the view for taking an exam.
- **TakeExam (POST):** Handles the submission of exam answers.
- **DisplayExamTemplate:** Displays the template of an exam.

HomeController

- **Purpose:** Manages the home and error pages of the application.
- **Key Methods:**
- **Index:** Displays the default view for the home page.
- **Privacy:** Displays the privacy policy page.
- **Error:** Displays the error page, ensuring it is not cached and includes an ErrorViewModel for tracking purposes.

InstructorController

- **Purpose:** Manages instructors, including listing all instructors, displaying instructor details, managing courses for instructors, adding and editing instructors, and handling department management for instructors.

- **Key Methods:**
- **Index:** Lists all instructors.
- **Details:** Displays the details of a specific instructor.
- **InstructorCourses:** Displays the courses associated with a specific instructor.
- **Edit:** Displays the view for editing an existing instructor.
- **SubmitEdit:** Handles the editing of an existing instructor, including updating the instructor's image, email, and name.
- **RemoveDepartmentManager:** Removes a department manager from a department.
- **AddDepartment:** Adds a department to an instructor's list of managed departments.
- **DeleteDepartment:** Deletes a department from an instructor's list of managed departments.
- **ManageCourses:** Manages the courses that can be added or removed from an instructor's list of managed courses.
- **Add:** Displays the view for adding a new instructor.
- **AddInstructor:** Handles the addition of a new instructor.

QuestionController

- **Purpose:** Manages questions, including listing all questions, creating new questions, displaying question details, editing existing questions, and deleting questions.
- **Key Methods:**
- **Index:** Lists all questions.
- **Create:** Displays the view for creating a new question.
- **Create (POST):** Handles the creation of a new question.
- **Details:** Displays the details of a specific question.
- **Edit (GET):** Displays the view for editing an existing question.

- **Edit (POST):** Handles the editing of an existing question.
- **Delete:** Deletes a specific question based on its ID.

RoleController

- **Purpose:** Manages roles, including listing all roles, displaying role details, creating new roles, editing existing roles, and deleting roles.
- **Key Methods:**
- **Index:** Lists all roles.
- **Details:** Displays the details of a specific role.
- **Create:** Displays the view for creating a new role.
- **Create (POST):** Handles the creation of a new role.
- **Edit (GET):** Displays the view for editing an existing role.
- **Edit (POST):** Handles the editing of an existing role.
- **Delete:** Deletes a specific role based on its ID.

StudentController

- **Purpose:** Manages students, including listing all students, displaying student details, managing courses for students, adding new students, and editing existing students.
- **Key Methods:**
- **Index:** Lists all students.
- **Details:** Displays the details of a specific student.
- **StudentCourses:** Displays the courses associated with a specific student.
- **Edit (GET):** Displays the view for editing an existing student.
- **Edit (POST):** Handles the editing of an existing student, including updating the student's image, email, name, and courses.
- **Add (GET):** Displays the view for adding a new student.
- **AddToBranch (GET):** Displays the view for adding a student to a branch.

- **AddToBranch (POST):** Handles the addition of a new student to a branch, including assigning the student to a department and adding courses.
- **AddCourses:** Adds courses to a student.
- **DisplayStudents:** Displays the students associated with a specific course.

UserController

- Purpose: Manages users, including listing all users, displaying user details, creating new users, editing existing users, and deleting users.
- **Key Methods:**
- **Index:** Lists all users.
- **Details:** Displays the details of a specific user.
- **UserDetails:** Displays the details of a specific user, possibly intended for a different view or functionality.
- **Create:** Displays the view for creating a new user.
- **Create (POST):** Handles the creation of a new user, including adding the user and their image to the repository.
- **Edit (GET):** Displays the view for editing an existing user.
- **Edit (POST):** Handles the editing of an existing user, including updating the user and their image in the repository.
- **Delete:** Deletes a specific user based on their ID, checking if the user is associated with an instructor, student, or admin role and deleting the associated role before deleting the user.