Documentation on Online Quiz System

Online Quiz System, this project is basically for two sections such as teachers and students. Teachers will be able to make questions and students will be attending exams on those questions prepared by teachers. In our project, we have used MVC pattern to represent our website. So, the classes, methods, parameters exist in the controller folder of our project.

In total, we have used 14 classes throughout the project which are,

- HomeController.cs
- CategoryController.cs
- NewCategoryController.cs
- SRegistrationController.cs
- StudentLoginController.cs
- StudentProfileController.cs
- TRegistrationController.cs
- TeacherLoginController.cs
- TeacherProfileController.cs
- LogOutController.cs
- RequestController.cs
- QuizController.cs
- ViewQuestionController.cs
- ReportController.cs

HomeControllerClass:

This class is for controlling the home page of our Online Quiz System. When we will run our system, the landing page will basically come by appending in this controller class.

The methods that are used in this class are,

- **Index():**
  This method is called for the view page of our design. Whenever this method will be called it will return to the home page view.
- **About():**
  This method will be called for the About Us page. That means, this method will return the view of about us page to show.

CategoryController Class:

This class is for creating subject categories by teachers. The methods that are used in this class are given below descriptively,

- **Addcategory() :**
  This method is [HttpGet] process, which is used for sending data using query string. Here, a list is taken from the Category table separating by the TeacherID, means particular teacher will be able to set their own categories or subjects. When the application gets to be run, this will pass the value of category.
  For example,
  ```
  List<Category> li = db.Categories.Where(x => x.CategoryTeacher == tid).OrderByDescending(x =>
  x.CategoryId).ToList();
          ViewData["list"] = li;
  ```
  Teacher will be creating Category list and will pass the value to the ViewData.

- **Addcategory(Category cat):**
  This is a [HttpPost] method, which is used for fetching data from the input and passing the data into the database. Here, the listed categories by different teachers will be fetched and those values of CategoryId and CategoryName will be posted into the database table through Model class. By that, the categories will be saved selecting by teachers.
  Here a parameter cat of Category model class is used as a parameter for passing the data into the database.

For example,

```
Category c = new Category();
        c.CategoryName = cat.CategoryName;
        c.CategoryTeacher = Convert.ToInt32(Session["TeacherID"].ToString());

        db.Categories.Add(c);
        db.SaveChanges();
```

Category is a Model class and c is an object of that class and the values are getting passed through the parameter, cat and as db is the object of our database model, we are saving the values in db.

NewCategoryController Class:

This class is basically for adding questions by the categories selected by the teachers. This class will be working for making the questions under different categories with options and selecting the right answer and post it in the database so that, when it will be available, students can get the exam data from the database and give the exam.

This class contains two methods for [HttpGet] method and [HttpPost] method. Below, these are descripted properly,

- **NewCategory():**
  In this method, we have used the ViewModel class, CategoryViewModel and made an object with that class, objCategoryViewModel so that the teacher can select the category or subject through the values listed from the Category Model.
  For example,

```
objCategoryViewModel.ListofCategory = (from objCat in db.Categories
                                       select new SelectListItem()
                                       {
                                          Value = objCat.CategoryId.ToString(),
                                          Text = objCat.CategoryName
                                       }).ToList();
```

  Here, the Categories are listed with the help of ListOfCategory, which is a list component set in the ViewModel class before. So, here a teacher will be selecting categories, from which he already saved as his chosen categories for making questions. For the Viewmodel to be passed, we have passed the ViewModel object in the return View process.

- **NewCategory(quesOptionViewModel quesOption):**
  It is a JsonResult type method. JsonResult is an ActionResult type in MVC. This format is used for the data to look very easy to understand and here the data objects consist of attribute-value pairs.

In this method, we have used another ViewModel class, quesOptionViewModel and taken a parameter of this class, quesOption. This method is for creating questions, giving otions with the questions and selecting the right answer of a question from the options. We have taken three model classes which are Question, Option and Answer and created object with those class for passing the created values into the database.
As example for Question,

```
Question objQuestion = new Question();
          objQuestion.Question_Text = quesOption.Question_Text;
          objQuestion.QuesCategoryId = quesOption.CategoryId;
           db.Questions.Add(objQuestion);
          db.SaveChanges();
```

Here, we have created object for Question Model class and passed the values inserted by the parameter which will save the data into the database.
Then for Option,

```
foreach(var item in quesOption.ListOfOptions)
          {
             Option objOption = new Option();
             objOption.OptionName = item;
             objOption.OptQuesId = questionid;

             db.Options.Add(objOption);
             db.SaveChanges();
          }
```

We have created object of the Option Model and inserted the options in the foreach loop which was set by List and passed through the parameter quesOption. This will save the options for the questions respectively into the database.
And then for Answer,

```
Answer objAns = new Answer();
          objAns.AnswerText = quesOption.AnswerText;
          objAns.AnsQuesId = questionid;
          db.Answers.Add(objAns);
          db.SaveChanges();
```

We have created an object for the Answer Model and passed the data through the parameter as the answer of that particular question to be saved in the database.
At last we have used a property of JsonResult, JsonRequestBehavior and set it as AllowGet and returned as Json to pass the values successfully.

RequestController Class:

This class is mainly created for controlling the request by a student to a teacher and the response of the teacher. Our project has the feature of request sending and receiving system. By this class, a student will be able to see the teacher list as the registered teachers and he can send a request to the teacher for attending his exam, also if he wants to delete the request, he will be able to do so. Later if the teachers accepts his request, he will see the teacher into his added teacher list folder.

For the teacher, he will be allowed to accept or delete the student request and when a student request is accepted, the student info will be shown in his added student list.

This class contains several methods which are explained below.

- **TeacherListForStudent():**

  This method is for showing the list of teacher so that the student can add the teacher or send request to the teacher.

  For example,

```
int sID = Convert.ToInt32(Session["StudentId"]);
        String sql = "select * from Teacher where TeacherID not in" +
          " ( select Teacher.TeacherID from Teacher join FriendListForStudnet on " +
          "Teacher.TeacherID = FriendListForStudnet.teacherid where studentid = " + sID
+ ")";

        var data = db.Teachers.SqlQuery(sql).ToList();
```

  Here, we are taking a session and passing the StudentId to the session so that it will be saved which student sent the request. Then we are passing the session through the string variable with the sql query, where two tables, Teacher and FriendListForStudent is joined so that the teacher list will be shown without the already added teachers.

- **TeacherListForStudentadd(int tid):**

  This is the [HttpPost] method for sending request to the teacher from the teacher list. If the student send add request to the teacher, the request id of the student will be sent to the particular TeacherID, which is passed by the integer type parameter tid. These information will be saved to the database through the FriendlistForStudent Model class.

  For example,

```
FriendListForStudnet AF = new FriendListForStudnet();
        AF.studentid = sID;
        AF.teacherid = tid;
        AF.timedate = DateTime.UtcNow.Date;
        db.FriendListForStudnets.Add(AF);
        db.SaveChanges();
```

  Here, an object is created of the Model class and data are inserted into the model through the session id and the parameter declared before. Then database is saving the request information which will get passed to the teacher profile.

- **Teacherprofileforstudent(string id):**
  This method is for showing the teacher profile to the student. When the student will see the teacher list, for recognizing the teacher or for seeing the teacher's profile he will be able to see the teacher profile by this method.
  Here, a string type parameter, id was passed so that when the TeacherName will appear in the list, the student will be able to go to his profile by that parameter sent.
  For example,
  ```
   var del = (from OnlineQuizSystem in db.Teachers
                      where OnlineQuizSystem.TeacherName == id
                      select OnlineQuizSystem).FirstOrDefault();
  ```
  In the above code, a query is inserted to the variable del where the teacher information from the Teacher Model will be passed through the parameter id. Later the del variable is sent to the return function so that it will take to the view of teacher profile.

- **Studentprofileforteacher(string id):**
  This method is for showing the student profile to the teacher. When the teacher will see the request sent by a student, he will be able to see the student profile for getting the information of the student.
  A string type parameter, id was passed so that when the StudentName will appear in the list, the teacher will be able to go to the student's profile by the parameter sent.
  For example,
  ```
  var del = (from OnlineQuizSystem in db.Students
                      where OnlineQuizSystem.StudentName == id
                      select OnlineQuizSystem).FirstOrDefault();
  ```
  Here, a query is inserted to the variable del, where the student information from the Student Model will be passed through the parameter id. Later the del variable is sent to the return function so that it will take to the view of student profile.

- **Studentlistforteacher():**
  The method is for showing the student list who is already listed as the teacher's accepted request. When the teacher will add the request of the student, that particular student will be listed in the Added Student list for teacher.
  For example,
  ```
          List<Student> addfriend = db.Students.ToList();

          ViewData["jointables"] = from fr in friendrequest
                          join ad in addfriend on fr.studentid equals ad.StudentID into table1
                          from ad in table1.DefaultIfEmpty()
                          select new AcceptStudentRequestClass { friendrequest = fr,
  addfriend = ad };
  ```

  Here, from the Student Model an object was taken as list and then the data was matched that if the studentId matches the AcceptedStudentRequest class which was separately made. If that matches, then the students will be listed to the list for Added Student list for teacher as ViewData["jointables"] is sent to the return function.

- **StudentRequestNotifictionForTeacher():**
  This method is for showing to the teacher which students have sent him request. The students who are already listed in his accepted student list, won't be shown to that list again. If the teacher deletes a student from his student list, then the student will be able to send request to the teacher if the student wants to. Anyways, here the teacher will be able to see his request list.
  For example,

```
List<FriendListForStudnet> friendrequest = db.FriendListForStudnets.Where(x =>
x.teacherid == sessId && x.friendlist == 0).ToList();
        List<Student> addfriend = db.Students.ToList();

        ViewData["jointables"] = from fr in friendrequest
                        join ad in addfriend on fr.studentid equals ad.StudentID into table1
                        from ad in table1.DefaultIfEmpty()
                        select new AcceptStudentRequestClass { friendrequest = fr,
addfriend = ad };
```

  Here, an object was made from the FriendListForStudent Model as list and the condition was given that the students who are not in the added list will be able to be seen in the request list of a teacher. Then the query was inserted to check that if the student request id matched the conditions, the student request will be shown in the request list of the teacher. After that ViewData["jointables"] was passed to the return function so that the whole process goes to the teacher view panel.

- **StudentRequestNotifictionForTeacher(int sid, int delete):**
  The method is called for the process where the teacher can delete the request of a student. The students who sent him friend request, if the teacher wants, he can delete the request by this method. Then the student won't be listed in his student list. Here, two parameters were passed which are sid and delete and the datatype of the parameters are integer type. By the parameter sid, the studentID is passing and the delete parameter is stating if the request is deleted or not.
  For example,

```
 if (delete == 1)
        {
           db.FriendListForStudnets.Remove(del);
           db.SaveChanges();
        }
```

  Here, when the value of the delete parameter is true, then the request will be removed from the list of the teacher's request list and the procedure will be saved in the model.

- **StudentRequestNotifictionForTeacheraccept(int sid, int accept):**
  This method is called for accepting the student request. The students who sent request to the teacher, accepting their request is done here. In this method, two parameters are declared of integer type which are sid and accept. By the parameter sid, the StudentID is passing and the accept parameter is stating if the request of that StudentId is accepted or not.
  For example,
  ```
  if (accept == 1)
          {
             del.friendlist = 1;
             db.SaveChanges();
          }
  ```
  Here, when the value of the accept parameter is true, then the request will be accepted in the list of the teacher's request list and the procedure will be saved in the model.

- **FriendListForStudent():**
  This method is called in the student panel, where the added teacher list will be shown. The teachers who have accepted the request of the student, enlisting them in the added teacher list is done here.
  For example,
  ```
  ViewData["jointabless"] = from fr in friendlist
                            join ad in teacherlist on fr.teacherid equals ad.TeacherID into
                            table1
                            from ad in table1.DefaultIfEmpty()
                            select new friendlistforstudent { friendlist = fr, teacherlist = ad };
  ```
  Here, from the query, it is shown that if the teacherid matches the accepting friendlist id, then it will enlist that the teacher has accepted the request and so the teacher will be listed into the added teacher list of the student.

- **deletefrndforstudent(int tid, int delete):**
  This method is called for deleting teacher from the student's added teacher list. Here, two integer type parameters are passed which are tid and delete. When the student will press the delete button, this method will be called and it will execute the procedure of deleting the teacher from the added teacher list. From the parameter tid, it will get the teacher id and with the parameter delete, the program will be able to delete that teacher id along with information from the added teacher list.
  For example,
  ```
  if (delete == 1)
          {
             db.FriendListForStudnets.Remove(del);
             db.SaveChanges();
          }
  ```

Here, it is shown that if the delete parameter gets the true value, then it will call the remove function and make the changes into the model which will delete the teacherID from the added teacher list.

- **requestsentforstudent():**
  This method is for showing that a student has sent request to which teachers. When this method will be called, it will execute work of viewing the list of which teacher are there to have the request sent, also when the request is sent to a teacher, that teacher won't be shown in the teacher list, where students generally find the teachers.
  For example,
  ```
   ViewData["jointabless"] = from fr in friendlist
                                       join ad in teacherlist on fr.teacherid equals ad.TeacherID into
                                       table1
                                       from ad in table1.DefaultIfEmpty()
                                       select new friendlistforstudent { friendlist = fr, teacherlist = ad };
  ```
  Here, the query is matching from the teacher list that if that teacher has been recorder as request sent and if that matches then the fetched data, will show the request list in the view, means sent request list.

- **deleterequestsentforstudent(int tid, int delete):**
  It is a method for deleting the request by student which was sent by a teacher before. That means by implementing this method, students will be able to delete the request which he sent to a teacher before accepting the request. Here, we have taken two parameters of integer types which are tid and delete. By tid parameter, we are sending the teacher id which is indicated to be deleted and by delete parameter, we are making the remove operation done.
  For example,
  ```
  var del = (from OnlineQuizSystem in db.FriendListForStudnets
                 where OnlineQuizSystem.teacherid == tid && OnlineQuizSystem.studentid
                 == sessId
                 select OnlineQuizSystem).FirstOrDefault();
        if (delete == 1)
        {
          db.FriendListForStudnets.Remove(del);
          db.SaveChanges();
        }
  ```
  Here we can see that, by passing the parameter tid we are matching that with the teacher id and then by making the delete parameter true, the remove operation is implementing and the it is being saved in the database model to have the record.

SRegistrationController Class:

This class is mainly created for controlling the registration of student panel. Students will give the required information properly which is required and then they will be able to register as a student and then they can continue to log in to their account. Without registration, no students can give any exam.

This class contains two methods for [HttpGet] method and [HttpPost] method. Below, these are descripted properly,

- **StudentRegister():**
  This method is executed for calling the view page or cshtml file of registration page.

- **StudentRegister(Student stv):**
  This is an [HttpPost] method, which means here the work of insertion into database or model is done. When the view page of registration is called, after that there has to be taken some values into the database model with the help of a parameter and here, a parameter is passed which is stv and it is a parameter of Student Model type.
  For example,
  ```
  Student s = new Student();
          s.StudentName = stv.StudentName;
          s.ID = stv.ID;
          s.Passwords = stv.Passwords;
          s.Email = stv.Email;
          db.Students.Add(s);
          db.SaveChanges();
  ```
  Here, an object of Student class was taken to pass the values into the model for which the parameter stv is used. After the values were passed, it is saved into the database model for having the record to register.

StudentLoginController Class:

This controller class controls the login system of a student. After registration the student need to login to go to their profile and give exam. For login, they need to give the proper information and if the information matches with the record from the database, students will be able to login to their account.

This class contains two methods for [HttpGet] method and [HttpPost] method. Below, these are descripted properly,

- **SLogin():**
  This method is for viewing the page of student login. After calling this method, a view page will be shown where the login page for student will be open and there the login information will go.

- **SLogin(Student s):**
  This is an [HttpPost] method, where student will pass the login information into the database with the help of the parameter s, which is a type of Student class. In this method, verification is done if the login info doesn't match with the updated info from the Student database model. If the info doesn't match, then an error message will be shown in the design page. For passing the values, an object of Student model class was created so that it can enter the new values into the database.
  For example, we have used session for this process, where we passed the student id into the session and with that session students are getting logged in.

StudentProfileController Class:

This controller class will show the student profile after login. Also, when the student will want to see his profile when he's in the student panel, he will be able to see his profile information. The profile information is taken from the records which is given in the registration form.

This class contains a method which is mentioned below,

- **StudentProfile():**
  This method is executed for showing the profile of a student after signing in. For having the track of sign in or login, session was used here and after signing in, a view page will appear which will be of the profile of that signed in student.

TRegistrationController Class:

This class is mainly created for controlling the registration of teacher panel. Teachers will give the required information properly which is required and then they will be able to register as a teacher and then they can continue to log in to their account. Without registration, no teachers can create any questions or take any exam.

This class contains two methods for [HttpGet] method and [HttpPost] method. Below, these are descripted properly,

- **TeacherRegister():**
  By calling this method, a view page of registration will be called where a new page will open for the registration of reacher. This is an [HttpGet] method, and here, we have to insert the values that is shown in the page.

- **TeacherRegister(Teacher tch):**
  This method is for posting the values of registration into the database to save the records. Here we have passed a parameter of Teacher Model class which is tch and when the view page of registration is called, after that there has to be taken some values into the database model with the help of that parameter.

For example,

```
Teacher t = new Teacher();
            t.FullName = tch.FullName;
            t.Email = tch.Email;
            t.TeacherName = tch.TeacherName;
            db.Teachers.Add(t);
            db.SaveChanges();
```

Here, an object of Teacher class was taken to pass the values into the model for which the parameter tch is used. After the values were passed, it is saved into the database model for having the record to register.

TeacherLoginControllerClass:

This controller class controls the login system of a teacher. After registration the teachers need to login to go to their profile and take exam with creating questions. For login, they need to give the proper information and if the information matches with the record from the database, teachers will be able to login to their account.

This class contains two methods for [HttpGet] method and [HttpPost] method. Below, these are descripted properly,

- **TLogin():**
  When this method is called, a view page for login as a teacher will appear where the login information has to be inserted.

- **TLogin(Teacher t):**
  This is an [HttpPost] method, where a teacher will pass the login information into the database with the help of the parameter t, which is a type of Teacher class. In this method, verification is done if the login info doesn't match with the updated info from the Teacher database model. If the info doesn't match, then an error message will be shown in the design page. For passing the values, an object of Teacher model class was created so that it can enter the new values into the database.
  For example, we have used session for this process, where we passed the teacherId into the session and with that session teachers are getting logged in.

TeacherProfileController Class:

This controller class will show the teacher profile after login. Also, when the teacher will want to see his profile when he's in the teacher panel, he will be able to see his profile information. The profile information is taken from the records which is given in the registration form.

This class contains a method which is mentioned below,

- **TeacherProfile():**
  This method is called for showing the profile of a teacher after signing in. For having the track of sign in or login, session was used here and after signing in, a view page will appear which will be of the profile of that signed in teacher.

LogOutController Class:

In our quiz system there are two separate panels for student and teacher. This controller works for logout system. When a teacher in in his account, if he wants to logout, then this controller will make that work proceed and return him to the home page. Also the student will be able to logout from his account and go to homepage by this controller

This class contains a method which is mentioned below,

- **Logout():**
  This method is called when the student or the teacher will press for logout. When this method will be executed, all the sessions will get abandoned, and will return to the view page of home.

QuizController Class:

The quiz controller mainly controls the whole quiz system of this project. With this controller, the students will be able to attend the exam on the questions that teacher has made before and also made the exam available for the students and the students will be able to see the questions then answer the questions and later they will be able to see the result of their given exam. This controller works for the student in typical ways.

This class contains several methods which are explained below,

- **QuizStudent():**
  This method is called when the student wants to give a quiz. Here, it will show the exam names which is categorized by teacher or made available by teacher for the student. Here we have used a ViewModel, QuizCategoryViewModel and created an object of it to select and pass the values into the database.

For example,

```
objQuizViewModel.ListofCategory = (from objcat in db.Categories
                        join objfrndlist in db.FriendListForStudnets on
objcat.CategoryTeacher equals objfrndlist.teacherid
                        where objfrndlist.studentid == sID && objfrndlist.friendlist ==
1 && objcat.available == 1
                        select new SelectListItem()
                        {
                          Value = objcat.CategoryId.ToString(),
                          Text = objcat.CategoryName
                        }).ToList();
```

Here, we have joined the tables of Category and FriendListForStudnets because the students who are not added in the teacher list won't be able to give exam and also it is checked that if the category is made available by the teacher or not. If the teacher didn't make available the quiz category, then it won't appear in that page.

- **QuizStudent(int CategoryId):**

  This method is executed when the student will select the exam category and want to start the exam by showing questions. Here, the selected category will be inserted to the database of ExamStudent table. There it will be recorded which student has selected which category and we have used session for saving the studentId.

  For example,

```
ExamStudent objstudent = new ExamStudent();
      objstudent.StuCategoryId = CategoryId;
      db.ExamStudents.Add(objstudent);
      db.SaveChanges();
```

  Here, we have created an object for the model class and inserted the record through the parameter CategoryId, which is of integer type. That passes the record of which category of exam is selected.

  Also, by this method, some clarification was done for showing the student result.

  For example,

```
 IEnumerable<Result> resultAll = db.Results
             .Where(u => u.ResQuesId == data.QuestionID && u.ResStudent == sID);
```

  As we have shown the result of a particular student, which student has given which exam, that data is needed. So, from this method we have inserted the result by giving the condition of matching the question id and the result question id and these will be of that particular student who has given the exam.

➢ **UserQuestionAnswer(StudentAnswer objStudentAnswer):**

This is the method for executing the partial view where the question and options are displaying. For partial view, we have made the method as PartialViewResult. In this method, we have worked for showing questions, options and answers properly, which was set by the teacher earlier.

For showing questions, we have taken a list for question as there will be various questions displaying one by one and showed them from the database. We have used a ViewModel for recording into database which is QuizAnswerViewModel and by making an object of that model, we have matched the questionId and questionText with the values of Question class.

For example,

```
 objAnsViewModel.IsLast = IsLast;
            objAnsViewModel.QuestionID = objQuestion.QuestionID;
            objAnsViewModel.Question_Text = objQuestion.Question_Text;
```

Here, we have also checked if the question is the last one or not as it is a list of questions. And also, we have sent the values of the Question Model to the QuizAnswerViewModel to appear in the exam question.

For showing the options of particular questions, we have used the QuizAnswerViewModel and made an object and with that object we have checked the options should be of some particular question which the teacher has set before.

For example,

```
objAnsViewModel.ListOfQuizOption = (from obj in db.Options
                                    where obj.OptQuesId == objQuestion.QuestionID
                                    select new QuizOption()
                                    {
                                       OptionName = obj.OptionName,
                                       OptionID = obj.OptionID
                                    }).ToList();
```

Here, we have put the values into the ViewModel where the option question id has to be matched with the question id which was created before. Then we set it with list function so that the options will appear as a list.

Now for answering the questions, student has to select the answer from the options given. Which student is giving which answer, this process is controlled by a session.

For example,

```
if (objStudentAnswer.AnswerText != null)
            {
                List<StudentAnswer> objstudentAnswers = Session["StdQuestionAnswer"] as
List<StudentAnswer>;
                objstudentAnswers.Add(objStudentAnswer);
                Session["StdQuestionAnswer"] = objstudentAnswers;
            }
```

Here, when the student is answering the question, the answers are saved by a list as there will be many answers and the data is being recorded by session. And it is passing through the parameter, **objStudentAnswer** which is the data type of StudentAnswer. Also, the answers is saving into the StudentAnswer which is the data for saving the answers.
All of these will appear in the partial view and later when the JSON method will be called, all of the data will be saved into the database.

- **SaveStudentAnswer(StudentAnswer objStudentAnswer):**
  This method is executed for saving the answers of the students and passing it to the Result table. This is a JsonResult method because we have saved the values by using json. By calling this method, the data which will be passed through the parameter objStudentAnswer will be saved into the database.
  For example,

```
foreach (var item in canAnswer)
      {
        Result objResult = new Result();
        objResult.AnswerText = item.AnswerText;
        objResult.ResQuesId = item.QuestionID;
        objResult.ResStudent = Convert.ToInt32(Session["StudentID"]);
        db.Results.Add(objResult);
        db.SaveChanges();
      }
```

  Here we have created an object of Result class and passed the answers of the students with the condition of selecting answers and verified the values are matching and then saved in database to have the record. For recognizing the student we have used session here to differentiate which student has given which question's answer.

- **GetFinalResult():**
  This method is called when the student will submit the answers and complete his quiz. Then there will be a view page where the students will be able to see the result of their given quiz.
  For example,

```
var UserResult = (from objResult in listOfQuestionAnswers
               join objAnswer in db.Answers on objResult.QuestionID equals
               objAnswer.AnsQuesId
               join objQuestion in db.Questions on objResult.QuestionID equals
               objQuestion.QuestionID
               select new ResultModel()
               {
                 Question = objQuestion.Question_Text,
                 Answer = objAnswer.AnswerText,
                    AnswerByStudent = objResult.AnswerText,
                    Status = objAnswer.AnswerText == objResult.AnswerText ? "Correct" :
                        "Wrong",
                }).ToList();
```

Here from the query, the checking is happening that which answer, which question and which option is getting executed and through the ViewModel, ResultModel the values are passing and showing which answer is given correct or wrong by the student.

We are also calculating the marks of the students by their correct answers. For that we are counting each answers of that particular question given by a particular student and also which are right.

For understanding,

```
Resultshow showresult = new Resultshow();
        showresult.studentID = studnetId;
        showresult.QuesCategoryId = quizCatId;
        showresult.totalmarks = count;
        db.Resultshows.Add(showresult);
        db.SaveChanges();
```

Here, we have made an object of the class Resultshow and passed the data of student category and put the value of count in total marks which is shown in the method. After that the values are getting saved in the database.

ViewQuestionsController Class:

This controller class is basically for viewing the questions after creating and also, for the view panel the teacher would be able delete any question if he wants to. Then the teacher can also delete the whole exam he has created by deleting the category overall. Here, the process of seeing sample question is also done. This class also controls the system of exam availability, this means if the teacher makes the exam available, then the student will be able to give exam for that particular time. But if the teacher makes the exam unavailable, then no student will be able to attend the exam.

This class contains several methods which are explained below,

- **ViewQuestion():**

  This method is [HttpGet] process and this is used for sending data using query string. Here a list taken from Category table separating by TeacherId, means particular teacher will be able to see only his/her category or subject. When the application gets to be run, this will pass the value of the category.

  For example,

```
List<Category> cat = db.Categories.Where(x => x.CategoryTeacher == sessId).ToList();
            ViewData["list"] = cat;
```

Here, the Category class will be set as list because particular teachers will have several different categories. By the query which is sent in the object cat, it is ensuring that particular teacher will see his created category question sets which will be passed by ViewData.

- **showquestions(int id):**

  This method is [HttpGet] process which is executed for showing the question which the teacher has created already. Teacher will be able to see the questions he made and also the answers of that question. In this method, a parameter was also passed of integer type, which is id. This paramtere passes the particular category for questions.

  For example,

```
List<Question> ques = db.Questions.Where(x => x.QuesCategoryId == id).ToList()
 List<Answer> ans = db.Answers.ToList();
var allQuestionData = from que in ques
                          join an in ans on que.QuesCategoryId equals an.AnsQuesId into
                          table
                          from an in table.DefaultIfEmpty()
                         select new showsubjectnumberClass { ques_text = que, catid = que,
                         quesid = que, anstext = an, ansquesid = an }
```

  Here, a list is taken from Question model separating by CategoryId. This CategoryId was passed by the parameter id in this method from ViewQuestion, so that only those question will be selected which are only in that particular question set.

  Also, answers of the particular question will be fetched from Answer table.

  To understand,

```
  foreach (var data in ques)
{
  anstextViewModel sing = new anstextViewModel();
  Answer answer = db.Answers.Where(u => u.AnsQuesId ==
                        data.QuestionID).SingleOrDefault();
  sing.answerId = answer.AnswerID;
  sing.answerText = answer.AnswerText;
  sing.questionId = data.QuestionID;
  sing.questionText = data.Question_Text;
  sv.Add(sing);
}
```

  Here, Question and Answer list will be joined by showsubjectnumberClass and all the values will be passed to the View through anstextViewModel. The teacher will see his created Questions which will be passed by ViewData.

- **makeavailable(int cid ,int available):**

  This method is [HttpPost] process which is executed for controlling the quiz process by the teacher. Here, if the teacher makes the exam available, then the student will be able to attend the exam for the time until the teacher makes the exam unavailable. For passing the values to the database, two parameters are used of integer type, which are cid and available.

  For example,

  ```
  var del = (from OnlineQuizSystem in db.Categories
                  where OnlineQuizSystem.CategoryId == cid
                  select OnlineQuizSystem).FirstOrDefault();
  int questionCount = db.Questions.Where(u => u.QuesCategoryId == cid).Count();
  if (available == 1)
     (if (questionCount > 1)
            {
              del.available = 1;
              db.SaveChanges();
            }
          }
       else{del.available = 0;
            db.SaveChanges();  }
  ```

  Here we can see, First the category will be selected by the CategoryId. Parameter cid is passing the Category id and parameter available represents the value of the available button. If the value is sent zero, the particular category will be unavailable for the student and if the value is 1 then it will make the particular category available for students. Also, questionCount is counting the number of questions in that particular category. And we have put a condition where a category must have at least 2 question for availability. Otherwise it won't be available.

- **deleteques(int delete,int qid):**

  This method is executed for deleting the questions which are created by the teachers. Suppose, a teacher has created some questions in particular exam category, now he wants to delete one question from that exam. By executing this methos, he will be able to make this process happen. In this method, two parameters have been used of integer types which are delete and qid that will hep to pass the deleting condition and the question id respectively.

  For example,

  ```
  var del3 = (from OnlineQuizSystem in db.Questions where OnlineQuizSystem.QuestionID
  == qid select OnlineQuizSystem).FirstOrDefault();
  if (delete == 1) {db.Questions.Remove(del3);
                     db.SaveChanges();
                     }
  ```

Here we can see, if the value of the delete is one then that particular question will be deleted. First it will be deleted from Result and Answer table. Then all the options will be deleted and after that the question will be deleted from the Question table. For passing the questioned we are using qid parameter and for delete operation we are using delete parameter. After executing this, that particular question will be fully deleted from the database.

- **deleteset(int delete, int qid):**

  This method is for deleting the whole exam set the teacher has created questions for. But for deleting the whole set, a teacher must need to delete all the questions under that set. Otherwise the category won't be deleted. If the category doesn't contain any question it will be deleted from all the database.

  Fr example,

```
  if (questionCount>0)
          return RedirectToAction("ViewQuestion");
else{
 var del3 = (from OnlineQuizSystem in db.Categories  where
OnlineQuizSystem.CategoryId == qid select OnlineQuizSystem).FirstOrDefault();
if (delete == 1) {
db.Categories.Remove(del3);
db.SaveChanges();
  }
```

  Here, questionCount is checking if that particular category have no questions or not. If the value of the parameter delete is 1 that particular category(qid) will be deleted from the category model.

ReportController Class:

This Controller class is basically assembled for showing the whole report of a student to himself and to the teacher of all his exams given by every student. By this class, a teacher will be able to see every students' result of the exams taken by him as report view and also, he will be able to see the profile of any student from there also. Again, the student will also be able to see a report of his given exams. There is also a feature of seeing the questions as sample questions by the student. Like, after giving the exams, he will be able to see the questions and correct answers from the report and a sample question view.

This class contains several methods which are explained below,

- **SendResults():**
  This method is called for showing the report to the student. Here report means a student will be able to show all the results of all the given exam of him. All the exams will be categorized and by this method view he will be able to see total marks of each exam he has given.
  For example,
  ```
  ViewData["jointables"] = from r in res
                          join c in cat on r.QuesCategoryId equals c.CategoryId into table
                          from c in table.DefaultIfEmpty()
                          select new showsubjectnumberClass { CategoryName = c, marks = r
                          ,CategoryId=c};
  ```
  Here, the required values are passing to showsubjectnumberClass which are the exam category id, name and marks of that exam. With surpassing the ViewData["jointables"] into the return function, the view is displaying.

- **showsamplequestions(int id):**
  This method is [HttpGet] process which is for showing the questions after attending the exam for the students. By executing this method, students will be able to see the questions and correct answers anytime from the report of his given exams. Here a parameter of integer type was passed to take the value of CategoryId, by which the questions of that exam will be shown.
  For example,
  ```
  List<Question> ques = db.Questions.Where(x => x.QuesCategoryId == id).ToList();
  List<Answer> ans = db.Answers.ToList();
  var allQuestionData = from que in ques
                          join an in ans on que.QuesCategoryId equals an.AnsQuesId into table
                          from an in table.DefaultIfEmpty()
                          select new showsubjectnumberClass { ques_text =que,  catid = que,
  quesid = que, anstext =an  ,ansquesid=an  };
  ```

Here, all the questions from Question table and also their answers from Answer table will be listed here by joining them through showsubjectnumberClass. After that through anstextViewModel values are shown to the student. By using this process, student will see the sample questions with correct answers which will be passed by ViewData.

- **ShowResultToTeacher():**

  This method is [HttpGet] process where the teachers will be able to check the results of the students of each exam set categories. That means, this page will show the exam sets of his created quizzes.

  For Example,

  ```
  string set = "(select Distinct  CategoryId,CategoryName,CategoryTeacher,available from
                  Category join Resultshow  on Category.CategoryId =
                   Resultshow.QuesCategoryId where Category.CategoryTeacher = " + tid + ")";
  var data = db.Categories.SqlQuery(set).ToList();
  ```

  Here, by joining the Category and Resultshow model, the query is approaching that all the question sets that is created by a particular teacher will be selected from database and that teacher will see this through a ViewData.

- **Showresultstudentid(int id):**

  This method is [HttpGet] process, which is for showing every students' marks in each exam. In the report section, the teacher will want to see the marks of every student in each exam and this method will execute the process of showing the results of each student of a particular exam represented as category. To pass the value of the particular question, we have used a parameter of integer type which is id.

  For example,

  ```
  List<showsubjectnumberViewModel> sv = new List<showsubjectnumberViewModel>();
  foreach (var data in res) {
  showsubjectnumberViewModel sing = new showsubjectnumberViewModel();
  Student stud = db.Students.Where(u => u.StudentID ==
  data.studentID).SingleOrDefault();
  sing.studentfullID = stud.ID;
  sing.totalmarks = Convert.ToInt32( data.totalmarks);
  sv.Add(sing); }
  ```

  Here, a list will be created from ResultShow categorized by CategoryId and after that Student table and Result table has been joined through showsubjectnumberClass and after that StudentId and their marks is showed by showsubjectnumberViewModel.

  That means questions of a particular category (parameter id) will be selected by teacher and the teacher can see all the students' id and also there marks on the particular category. Teacher will see his StudentId and their marks which will be passed by ViewData.

- **Studentprofileofstudents(int id):**
  By executing this method, the teacher will be able to view the student profile from his report lists. When a teacher will go to his report view, he will be able to see the results of every student who has given his exams. So, from that view also, teacher can go to the profile view of the pointed student. For passing the student info, a parameter id is used wich is of integer type.
  For example,
  ```
   var del = (from OnlineQuizSystem in db.Students
                    where OnlineQuizSystem.StudentID == id
                    select OnlineQuizSystem).FirstOrDefault();
  ```
  Here, we have passed the parameter id to the StudentID from Student class to proceed the work to go the view of that student profile.