Online Exam

Object Design

<0.1>

<1.1.2016>

Muhammed Bilal Kutlu

Güneycan Demir

Çağla Çolak

Musa Serdar Akmehmet

Prepared for

SE301 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc436772639)

[1.1. Object Design Trade-offs 1](#_Toc436772640)

[1.2. Interface Documentation Guidelines 1](#_Toc436772641)

[1.3. Definitions, Acronyms, and Abbreviations 1](#_Toc436772642)

[1.4. References 1](#_Toc436772643)

[2. Packages 1](#_Toc436772644)

[3. Class Interfaces 1](#_Toc436772645)

OBJECT DESIGN DOCUMENT

* 1. Object Design Trade-Offs

The Online Exam project consists of modules such as exam management, the instructor portal, the student portal, the administration portal, and so on. Therefore, our content management module has to interact with the other modules. Inter-company relationships are an important topic to consider and we have increased our workload to facilitate other modules for the storage and retrieval of data from the database. By providing these interfaces we take control of the database and obey the rules of object-oriented programming. So when you make a modification to a table in the DB, other modules will not get into trouble.

As the user enters data, the module will force the user to follow the predefined pattern designed for this purpose. This will surely prevent a faulty entry of the data into the database. Therefore, the consistency and integration of the data are provided with these patterns. Of course this procedure increases the complexity of our source code.

* + 1. **Guidelines and Convention:**

This sub-section describes about Naming Conventions, boundary cases, exception handling mechanisms.

* + - 1. **Naming Conventions:**

This section is describe in 1.2 subsection in detail.

* + - 1. **Boundary Cases:**

From the user interface, any text input field must have a maximum length indicating a limit.

-Username is not a 5 digit decimal number and Password is minimum 6 characters long.

In Login System, Password and username must match with database

-Password and username don’t match.

-Username is wrong or does not exist.

Also from the user interface, any text input can not be blank.

-Username or password field cant be blank.

-Instructor information, Exam name, date, time, question or answer fields cant be blank.

All mail input field must have "@" character .

- Depending on the type of database we use, the fields might have a maximum width value. We need to make sure there is enough space in each of the fields.

From the user interface, in register or adding text input fields must be unique.

- İnstructor informations cant be exist .

-Same class name is exist.

After we've exceeded the simulated exam time and sees the errors.

**1.1.5 Exception Handling Mechanisms:**

The Online Exam system is dealing with this programming language construct to ensure the fact that normal flow of execution takes place constantly. Not only just to synchronize the integrity of the system but our OES is merely focused about what are the constraints and raising these exceptions in a useful way to deal with any sort of problem or semi-predicate problem as listed below:

**a. Invalid User Name Password Exception:**

The user enters the Username and password into the encrypted text fields, and clicks on “Logon”. The Username communicates with the User Management subsystem to verify whether the Username already exists, and if it does, whether the password is correct or not. If the Username does not exist or the password is incorrect then an Invalid User Name Password exception is displayed.

**b. Overtime Exception:**

This is about the duration in the exam with a time tracker on the screen to display as to how much time has passed since the exam was joined. This page refreshes every seconds to show the latest availability of finish exam dispatched. If the time finish, then an Overtime Exception message pops up.

**c. Dublicate Data Exception:**

This exception is invoked when any information mismatch to the data type is tried to save in class list to add any student by an instructor.

**c. Student Not Found Data Exception:**

This exception is invoked when any information mismatch to the data type is tried to save in class list to delete any student by an instructor.

**d. Exist Email Data Exception:**

The user enters the email into the encrypted text field, and clicks on “Register”. The Email communicates with the system to verify whether the email already exists, and if it does. System throw exist Email data exception.

**e. Exist Username Data Exception:**

The user enters the Username into the encrypted text field, and clicks on “Register”. The Username communicates with the User Management subsystem to verify whether the Username already exists, and if it does. System throw exist username data exception.

**f. is Password equal Data Exception:**

The user enters the password into the text field and confirm password into the text field, and clicks on “Register”. The Password and confirm password are not equal. System throw Passwords are not equal data exception.

**1.2. Interface documentation guidelines**

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier-for example, whether it's a constant, package, or class-which can be helpful in understanding the code.

This section will give an insight about the Naming Convention. They are described individually in the table.

|  |  |  |
| --- | --- | --- |
| Identifier Type | Rules for Naming | Examples |
| Packages | The prefix of a unique package name is always written in capital and all lowercase ASCII letters except abbreviations like UI is User Interface so as not to make lengthy package name. We’ve not used any top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981. | package administration;  package instructor;  package student; |
| Classes | Class names should be nouns, in mixed case with the first letter of each internal word capitalized. We kept our class names simple and descriptive. Nonetheless, we used whole words and avoided acronyms and abbreviations as far as possible (unless the abbreviation is much more widely used than the long form, such as DB for database and UI for User Interface). Some classes’ names also have numeral as it makes us to feel the step of execution. | class addInstructor;  class ListofInstructor; class InsRequestApprove;  class ListStudents; class addClassicQuestion;  class addMultipleChoiceQuestion; class addQuestion;  class createExam; class createLectures;  class editClassicQuestion;  class editLecture; class editMultipleChoiceQuestion;  class editProfile; class manageClassicQuestions;  class manageExams;  class manageQuestions; class showClassList;  class joinexam; class question;  class adminlogin;  class login;  class register;  class index; |
| Interfaces | Interface names should be capitalized like class names. | interface addInstructor;  interface ListofInstructor; interface InsRequestApprove;  interface ListStudents; interface addClassicQuestion;  interface addMultipleChoiceQuestion; interface addQuestion;  interface createExam; interface createLectures;  interface editClassicQuestion;  interface editLecture; interface editMultipleChoiceQuestion;  interface editProfile; interface manageClassicQuestions;  interface manageExams;  interface manageQuestions; interface showClassList;  interface joinexam; interface question;  interface adminlogin;  interface login;  interface register;  interface index; |
| Methods | Our Methods are verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized. | registerBtn\_Click(object sender, EventArgs e)  addInstructorBtn\_Click(object sender, EventArgs e)  addQuestionBtn\_Click(object sender, EventArgs e)  PopulateDataForMultiple()  PopulateDataForClassic()  createExam(object sender, EventArgs e)  btnNext\_Click(object sender, EventArgs e)  addNewQuestion\_Click(object sender, EventArgs e)  startingPoint\_DayRender(object sender, DayRenderEventArgs e)  btnCreate\_Click(object sender, EventArgs e)  UpdateQuestionBtn\_Click(object sender, EventArgs e)  btnDelete\_Click(object sender, EventArgs e)  btnUpdate\_Click(object sender, EventArgs e)  updateProfile\_Click(object sender, EventArgs e)  ClassicQuestionEdit\_RowCommand(object sender, GridViewCommandEventArgs e)  addStudentWithUserName(object sender, EventArgs e)  ExamsGrdView\_RowCommand(object sender, GridViewCommandEventArgs e)  lstQuestions\_PagePropertiesChanging(object sender, PagePropertiesChangingEventArgs e)  deleteStudentWithUserName(object sender, EventArgs e)  loginBtn\_Click(object sender, EventArgs e) |
| Variables | Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names has no start with underscore \_ or dollar sign $ characters, even though both are allowed.  Variable names are short yet meaningful. The choice of a variable name is mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names are avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters. | Person person;  student.student student;  instructor.instructor instructor;  string query;  Database db;  SqlCommand cmd;  string instructorUsername ;  string examID;  string lectureCode ;  string questiontext ;  string answerFirst ;  string answerSecond ;  string answerThird ;  string answerFourth ;  string correctAns ;  string username ;  string code ;  string name ;  string description ;  bool isCreatedBefore; |

**1.3Definitions, acronyms and abbreviations.**

N.A.

## 1.4 References

auzef.istanbul.edu.tr

lms.isikun.edu.tr

# 2. Packages

* 1. Administration: This package represents the Admin interfaces for Online Exam system.
  2. App\_Code: This package contains classes used for making database connections and executing queries on the database.
  3. Assets: This package contains css classes for our bootstrap template.
  4. Instructor: This package represents the Instructor interfaces for Online Exam System.
  5. Student: This package represents the Student side interfaces for Online Exam System.

# 3. Class Interfaces

**Class “index.aspx”**

This is the welcome page of the applications. In this interface application provides sign up and login choices to the user.

**Class “register.aspx”**

This is the sign up interface for both student and instructor actor.

Following are the methods provided by class:

* protected void registerBtn\_Click(object sender, EventArgs e)

The register button in the register page interface calls this method. The method gets the texts from input fields creates a student or an instructor object and inserts to the database. Instructor is inserted unapproved. Admin approves instructor. The method throws exception and shows messages to the users while registering if username is exist or email exist. While choosing password, user must confirm the password if two field doesn’t match.

**Class “login.aspx”**

This is the login interface for all user type.

Following are the methods provided by class:

* protected void Page\_Load(object sender, EventArgs e)

This method redirects users to the main page if they are authanticated.

* protected void loginBtn\_Click(object sender, EventArgs e)

The login button in the login page interface calls this method. The method checks the user name , password and the role. If data matches user goes to main page. If username does not found in database or password doesn’t match with username throws Invalid Username Password Exception.

**Package Administration:**

**Class “index.aspx”**

This is the main page interface of the admin package. Admin can access the pages that has admin functions from here.

**Class “admin.Master”**

This is the master interface of admin package. Other classes inherits from this class.

**Class “addInstructor.aspx”**

This class is adding instructor by admin interface. Admin can add instructor manually on this page.

Following are the methods provided by class:

* protected void registerBtn\_Click(object sender, EventArgs e)

This method is a click event. it creates a new instructor in admin side.

**Class “InsRequestApprove.aspx”**

This interface class is showing request list of instuctor which waiting for approve. Admin’s approve or delete request functions are here.

Following are the methods provided by class:

* protected void PopulateData()

This method populates data for the instructors who have not approved yet.

* protected void GridView1\_RowCommand(object sender, GridViewEventArgs e)

This method sets the instructors that are wait for approve in a grid view. Admin can approve instructor by editing the approve row.

**Class “ListofInstructors.aspx”**

This class represents the admin interface for listing instructors that are approved.Admin can either delete or edit the instructor in this class.

Following are the methods provided by class:

* protected void addInstructorBtn\_Click(object sender, EventArgs e)

This method redirects admin to the addInstructor page.

**Class “ListStudents.aspx”**

This class represents the admin interface for listing students.Admin can either use delete or edit functions on the students in this class.

**Package App\_Code:**

**Class “Database”**

The object oriented Database class. (For Sql). This class is used to connect to the database and execute the queries.

Following are the methods provided by class:

* public Database()

This is the database construct.

* public void startDB()

This method starts the database connection.

* public void stopDB()

This method stops the database connection.

* public SqlCommand SqlCommand(String query)

This method takes the string query returns its sql command

* public DataTable GetData(SqlCommand cmd)

Returns the DataTable created by SqlCommand.

* public DataRow SelectData(SqlCommand cmd)

Returns the DataRow created by SqlCommand.

* public DataTable SelectDataTable(SqlCommand cmd)

Creates and returns a data table with a SQL command.

* public bool Execute(SqlCommand cmd)

This method helps to execute query.

* public int UserLogin(String username, String password)

The login method which takes username and password.

* public bool AdminLogin(String AdminName, String password)

The login method which takes username and password of admin.

**Class “functions”**

This class is used to inherit the common database functions to the system.

**Package assets:**

In this package there are css styles of bootstrap template that we used in our system.

**Package Instructor:**

**Class “addQuestion.aspx”**

This class contains question adding interface and in the code behind database operations.This class is used for adding questions to the created exam. Instructor can add question itself and its answers also can choose the correct answer.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method populates the question creation page for multiple type or classic type questions.

* void PopulateDataForMultiple()

This method is called from Page\_Load(). If selected type of the question that is wanted to create by instructor is “multiple choice question” then this method populates its fields.

* void PopulateDataForClassic()

This method is called from Page\_Load(). If selected type of the question that is wanted to create by instructor is “classic type question” then this method populates its fields.

* protected void createExam(object sender, EventArgs e)

This method inserts the questions and relates it with the exam of lecture.

**Class “createExam.aspx”**

This class contains creating exam interface and in the code behind database operations.Instructor can choose the lecture code from slide bar which are added before, then select question type(Multiple question or Classic) , number of questions, exam duration, also can add description and rule to the exam and can choose exam starting and ending date.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method redirects instructor to the login page if not authanticated.

* void PopulateData()

This method gets the lectures of the instructor to choose and add an exam to that lecture.

* protected void btnNext\_Click (object sender, EventArgs e)

This method gets the informations of the exam from the input fields and directs to the next stage of creating exam, addquestion page. If choosen ending date of the exam is a past day from starting date throws an exception and message to the user.

* protected void startingPoint\_DayRender(object sender,DayrenderEventArgs e)

Sets the starting date data as date.

* protected void EndingPoint\_DayRender(object sender,DayrenderEventArgs e)

Sets the ending date data as date.

**Class “createLectures.aspx”**

This class contains creating lecture interface and in the code behind database operations.Instructor fills the lecture code , lecture description , and lecture name then can create a new lecture with the function.

Following are the methods provided by class:

* protected void btnCreate\_Click (object sender, EventArgs e)

This method is a button event click. Gets the datas from input fields and inserts lecture to the database. If instructor tries to create a lecture with the same id throws exception.

**Class “default.aspx”**

This is the main page of the instructor.This class contains main page interface and in the code behind database operations. In the page, instructor can see the lectures he/she added. The interface nevigates instructor to the class list of the selected lecture with the class list button or to the editing page with the edit button.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method redirects instructor to the login page if not authanticated.

* void PopulateData()

This method gets the lectures of the instructor and show them on the main page.

**Class “editLecture.aspx”**

This is the edit page for the lecture which is added before.The class contains edit lecture interface and in the code behind database operations. In the page, instructor can change lecture name and description but not allowed to change lecture code.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* protected void btnDelete\_click(object sender, EventArgs e)

This method deletes the lecture which is choosed and directed to edit page from mainpage.

* void PopulateData()

This method gets the information of the lecture object which is selected from main page and show informations on the input fields.

* protected void btnUpdate\_Click(object sender, EventArgs e)

This method gets the information from input fields and updates the exam.Lecture code is not allowed to update.

**Class “editProfile.aspx”**

This class represents the editing instructor’s information page.The class contains edit profile interface and in the code behind database operations. Instructor can update his informations if neccessary except username.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets the information of the instructor object and show informations on the input fields.

* protected void updateProfile\_Click(object sender, EventArgs e)

This method gets the information from input fields and updates the profile information of the instructor. User name is not allowed to update.

**Class “instructor.cs”**

This class represents the instructor object. While registering, instructor chooses his role after that the register function in this class is used.

**Class “instructor.Master”**

This is the class that the other classes inherits for interface. Main theme of the page is pulled from here.

**Class “manageClassicQuestions.aspx”**

This class contains managing classing type questions interface and in the code behind database operations for questions. Instructor can change question itself , answers or correct answer.

**Class “manageExams.aspx”**

This class contains managing exams insterface and in the code behinde datrabase operations. Instructor can manage exams , edit or delete on database. Also can navigate to the manage questions page.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets the lectures of the instructor from database and show them on a grid view.

* protected void ExamsGrdView\_RowCommand()

This method sets and shows the populated data to the data row. From this grid blocks instructor can edit or delete exams or go to the questions of the exam.

.

**Class “manageQuestions.aspx”**

This class contains managing multiple choice type questions interface and in the code behind database operations for questions. Instructor can change question itself , answers or correct answer.

**Class “showClassList.aspx”**

This class contains class list interface of the selected lecture and in the code behind database operations.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets and shows the student informations which are added to selected lecture.

* protected void addStudentWithUserName(object sender, EventArgs e)

This is a button click method.The method adds the current lecture’s class list a new student with username. If student’s username is already exist in the current lecture then throws an exception and message “Duplicate”.

* protected void deleteStudentWithUserName(object sender, EventArgs e)

This is a button click method.The method deletes a student from class list with username. If student’s username is not exist throws an exception error “Student not found”.

**Package Student:**

**Class “editProfile.aspx”**

This class represents the editing student’s information page.The class contains edit profile interface and in the code behind database operations. Student can update his informations if neccessary except username.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets the information of the student object and show informations on the input fields.

* protected void updateProfile\_Click(object sender, EventArgs e)

This method gets the information from input fields and updates the profile information of the student. User name is not allowed to update.

**Class “index.aspx”**

This class represents the main page interface of the students.In main page there are shown upcoming exams of the student.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets the exam information of the student object and show the upcoming ones on the main page.

**Class “joinexam.aspx”**

This class is the joining exam interface. In this interface exam informations are shown. Join exam button calls the function and directs student to the questions.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method gets the exam information of the joined student and show to student exam name , rules ,descriptions etc.

* protected void btnStart\_Click (object sender, EventArgs e)

This method directs to the question page.

**Class “question.aspx”**

This class contains the interface of the exam by student side and in the code behind database functions. Student can join the exam then access questions and can answer questions.

Following are the methods provided by class:

* protected void Page\_Load (object sender, EventArgs e)

This method calls the PopulateData method.

* void PopulateData()

This method populates the questions of the exam

**Class “student.cs”**

This class represents the student object. While registering, student chooses his role after that the register function in this class is used.

**Class “student.Master”**

This is the class that the other classes inherits for interface. Main theme of the page is pulled from here.