ONLINE QUIZ SYSTEM

A PROJECT REPORT

Submitted By

Omveer Singh
(2000290140082)
Rahul Dadoo
(2000290140093)
Vinay Kumar
(2000290140132)

Submitted in partial fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Ms. Vidushi Mishra
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY 2022)

CERTIFICATE

Certified that Omveer Singh (2000290140082), Rahul Dadoo (2000290140093),Vinay Kumar (2000290140132)has carried out the project work presented in this report entitled " ONLINE QUIZ SYSTEM" for the award of Master of Computer Application from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision.

The report embodies result of original work, and studies are carried out by the students themselves and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Ms Vidushi Mishra External Examiner

Assistant Professor

Dept. of Computer Applications

KIET Group of Institutions, Ghaziabad

Dr. Ajay Kumar Srivastava

Professor & Head

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Date:28 may 2022

ABSTRACT

The purpose of MCQ Quiz Application is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

MCQ Quiz Application, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. We extend my sincere and heartfelt thanks to our esteemed guide, Ms Vidushi Mam, for providing us with the right guidance and advice at the crucial junctures and for showing us the right way. We extend our sincere thanks to our respected Head of the department Dr. Ajay Kumar Shrivastava, for allowing us to use the facilities available. We would like to thank the other faculty members also, at this occasion. Last but not the least, we would like to thank my friends and family for the support and encouragement they have given us during our work.

Omveer Singh (2000290140082)

Rahul Dadoo (2000290140093)

Vinay Kumar (2000290140132)

TABLE OF CONTENTS

DECLARATION

We hereby declare that the work presented in this report entitled “ONLLINE QUIZ SYSTEM”, was carried out by us.

We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution.

We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated.

In the event of a complaint of plagiarism and the manipulation of the experiments and results,

We shall be fully responsible and answerable.

Name: Omveer Singh

Rahul Dadoo

Vinay Kumar

Roll. No.: 2000290140082

2000290140093

2000290140132

Branch: Master of Computer Application

SYNOPSIS

Introduction

Online Examination System refers to service as conduct online examination or test. It will use for student progress evaluation using modern computer technology. It replaced the paperwork and overcome the outcomes of traditional way of examinations using paper or pen.It is web based platform can be used by Admin at any remote location. Online Examination System is fully developed automated system is to efficiently evaluate the candidate progress that not only save the time of Examination Controller and also gives fast result. The Administrator of the system has authority to propose tests or papers. It is cost effective and time effective.The candidate can login through proposed computer with their Enrolment number matching the details to the student's database, Then they can take the exam. Candidate can give their course's examination in a specific duration and in specific number of questions. The questions can be appear in both mode MCQ (Multiple Choice Questions) and answer in paragraph.

Literature Review

Xue (2006), adapted the online examination system using a mode known as "B/W Mode". Web server was used to control the tests and provide information to students. The main goal of the study is to reduce the use of paper and develop a test system that is safe to use. Among the features utilized was only allow the system to be used after receiving the application from student to use it.

Ria Mae H (2013), studied about an online examination system that can be administered manually or automatically by lecturers. Lecturers can send questions via the web and students will answer and send their answers online via the Internet. By using this online examination system, grading process can be done automatically. Researcher has conducted a survey about the form and type of examination questions that will be included in the online examination system.

Objective

The main objective of the Project on MCQ Quiz Application is to manage the details of Students, Examinations, Marks, Courses, Papers. It manages all the information about Students, Results, Papers, Students. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Students, Examinations, Results, Marks. It tracks all the details about the Marks, Courses, Papers.

Functionalities provided by MCQ Quiz Application are as follows:

Provides the searching facilities based on various factors. Such as Students, Marks, Courses, Papers

MCQ Quiz Application also manage the Results details online for Courses details, Papers details, Students.

It tracks all the information of Examinations, Results, Courses ect

Manage the information of Examinations

Shows the information and description of the Students, Marks

To increase efficiency of managing the Students, Examinations

Research Methodology

The research design depends on the purpose of the study (Maxwell, 2005). This study uses a quantitative approach carried out using a survey method. The survey method is appropriate for the management of the study because the reviews to be conducted will focus on the usability online testing system and based on the questions submitted. In addition, it involves data collection and information from the respondent involved only

Method of Data Collection

There are numbers of approach to data collection depending on the nature of the research being conducted. In this project, the methods adopted include the following: Interview, World Wide Web, references to published and unpublished collection. The data collected for this research can be broadly classified into two types, namely: the primary and secondary data.

PRIMARY DATA

Primary data can be defined as data collected directly from respondent relevant to the subject under investigation. The primary data used in this case is interview method according to Enr. D. O Dimoji (2022) says that primary source data collection is source from first-hand information can be obtained. The tools for gathering the primary source of data collection include interview, observation, and questionnaire etc.

In outcome we gather some data from end user for further process and verify and then validate it may be defined as the fundamental data which is needed for registration.

In this registration process there is need to have a personal email id which is valid format and

- He/she needs to a password which helpful for login

- Through password she can login in the site and able to verify his or himself so they have to create a password for the login and after creating the password they will enter in the next module so this is the fundamental data which the user need in this case the end user is that the student which is trying to register itself for the course it may be the courses like MCA and other courses like MBA or B Tech after then login the next phase is started here

- Now in the next phase we need to add some students extra details like students first name and then in second column students middle name it will be the optional for the student to seal it or not and after the middle name he needs to be registered with his last name or which we can call it's a surname after entering the surname and last name or middle name in it's to enter the mobile number for the verification and contact

- Now in the next box needs to enter this father or mother name his father's first name and surname also his mother's first name and surname in both names middle name is optional and it is up to the student he/she wants to enter or not after that he or she needs to be entered his city as address is needed for the registration in the college admission process It's to be entered in both addresses current and permanent address.

- after his student in need to enter his previous educational details like 10th, 12th, and graduation, if they are applying for that post-graduation course Like MCA and MBA but if they apply for the BTech there is no need for graduation details.

Outcome

Online Quiz system is a application softwareand website have done work in two part first is User and second is Admin .

User:-

- **Registration**
- **Login**

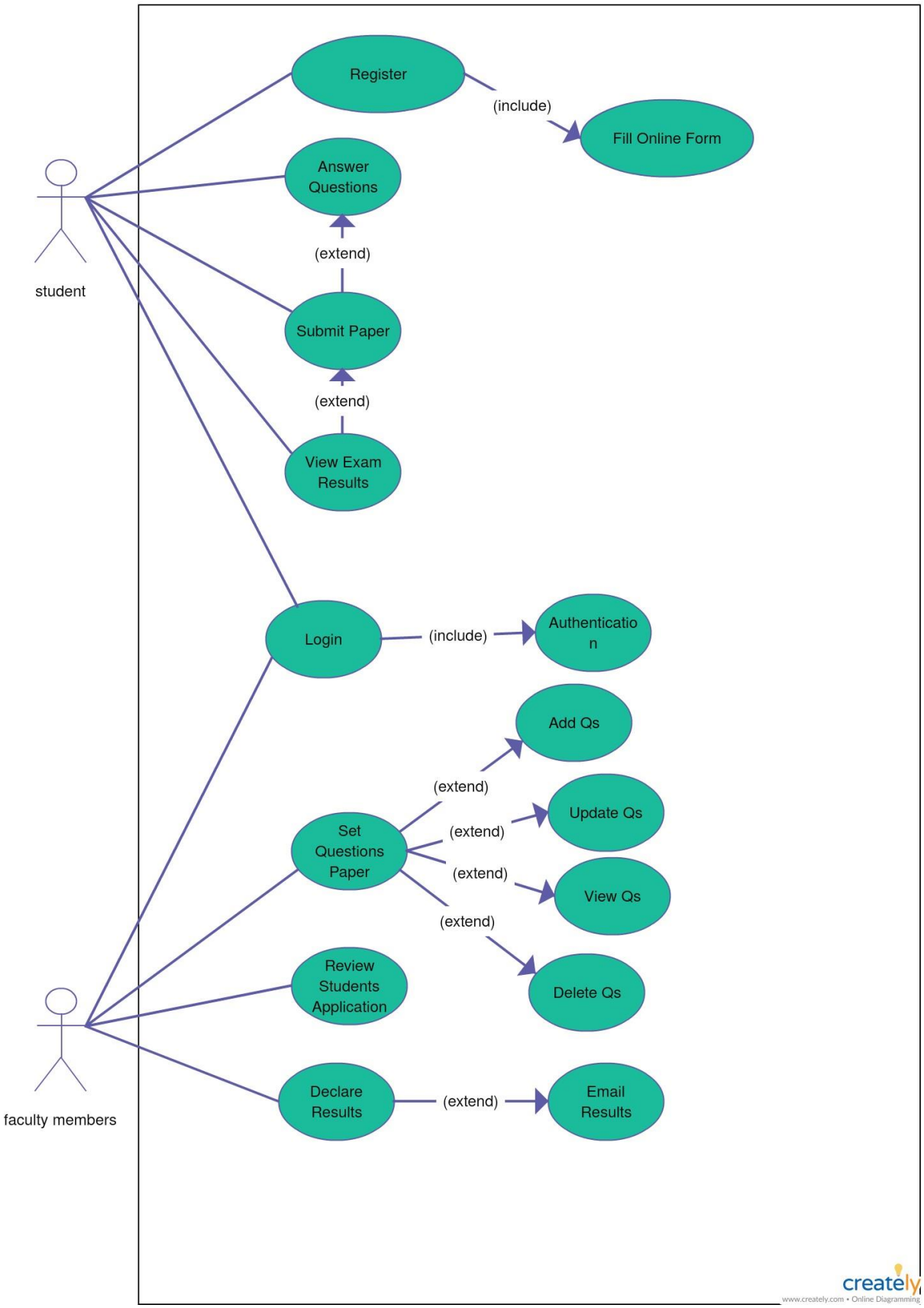
- **Take quiz**
- **Show performance**

Admin-

Second is Admin here admin can create the quiz and manage the settings of quiz like time duration and number of questions and also see the performance of user.

USE CASE MODEL

Online Examination System



Time Duration

PROJECT TASK	MARCH	APRIL	MAY
Planning & Requirement			
Design			
Coding & Implementation			
Testing & Report			

3.Specific Requirements

3.1 Use Case Reports

1. Administrator: Responsible for managing student details.

Use-case: Login into the website

Goal in context: Gain access to the website

Brief Description: This use case is used when the administrator wants to access the website to enable/disable/update the personal details of the student.

Preconditions: The Administrator must be logged onto the website for this use case to begin.

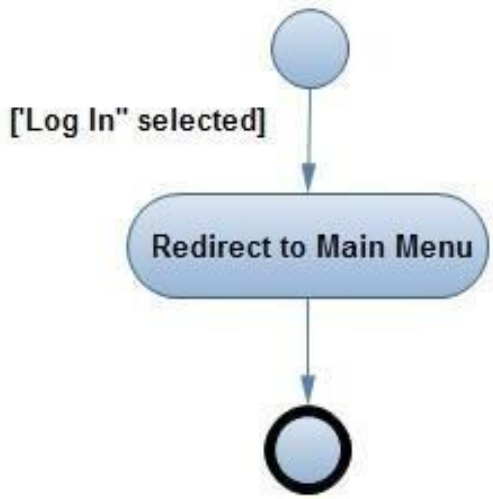
Basic Flow:

- The Website prompts the administrator for the username and password.
- The Administrator enters the username and password.
- The Website verifies the password and sets the user's authorization.
- The Administrator is given access to the Website to perform his tasks

Alternative Flow:

- The administrator enters invalid username and password then hewill not be allowed to enter the website**.**

Post conditions: The website state is unchanged by this use case.



Use Case Report- Login into the website

Use Case: Display student details

Goal in context: View the details of a student

Brief Description: This use case is used when the administrator wants to view the personal details of the students already existing in the database on the screen.

Preconditions:

- The Administrator must be logged into the system for this use case to begin
- The details of the student must pre-exist in the database
- The student id must be entered correctly.

Basic Flow:

- The Administrator logs onto the System.
- The Administrator search the student from following keys: -
- The System prompts for the student

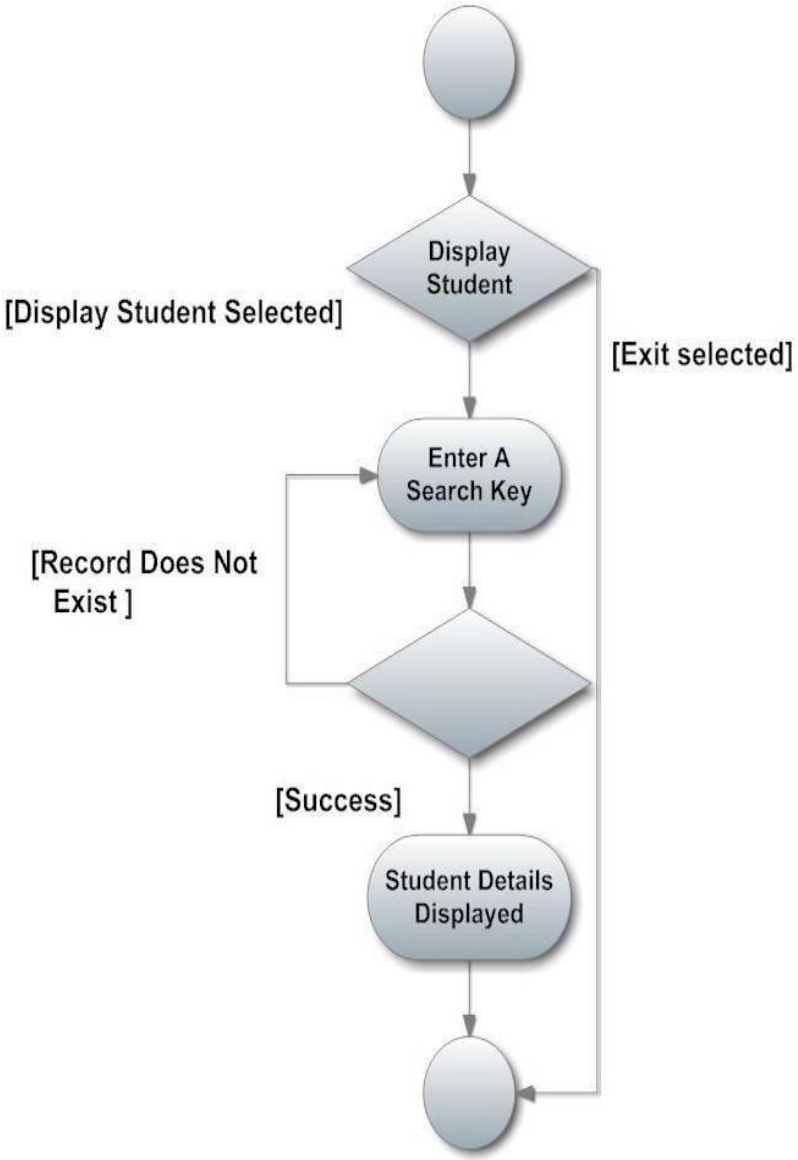
  detail from one of the above keys.
- The student details are displayed on

  the screen.

Alternative Flow:

Student Not Found

If in the **Display a student** sub-flows, a student with the specified id number does not exist, the system displays an error message. The Administrator can then enter a different id number or cancel the operation, at which point the use case ends.



Use Case Report-Display Student Details

Use Case: Edit student details

Goal in context: Edit the details of a student

Brief Description: This use case is used when the administrator wants to edit the personal details of himself/herself already existing in the database.

Preconditions:

- The Administrator must be logged into the system for this use case to begin.
- The details of the student must pre-exist in the database

Basic Flow:

The Administrator logs onto the System.

The Administrator can edit following keys: -

- First/last name
- Gender
- DOB
- Contact no Qualification
- City

- Email1
- Email2
- Address
- Description

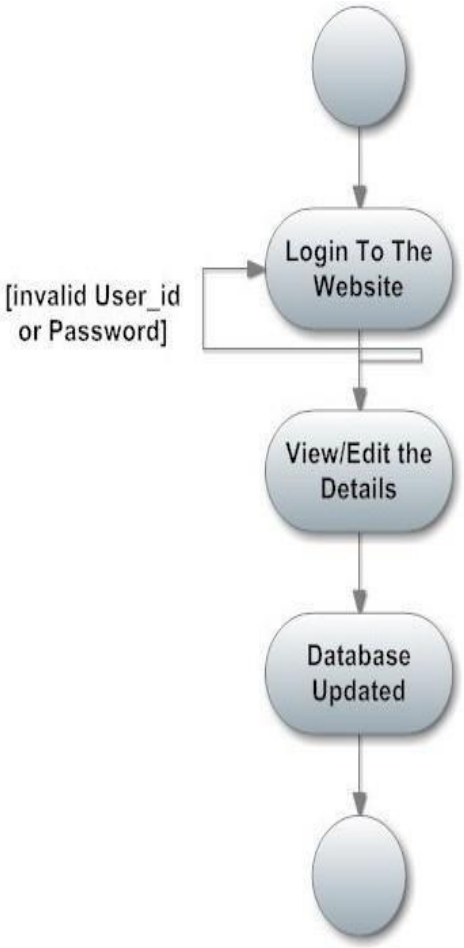
- The Website updates the database according to edited details.
- The student details are edited in the database.

Alternative Flow:

- There is no alternative flow of this use case diagram.

Post conditions:

- The student details get updated in the database.



Use Case Report- Edit student detail into thwebsite

2. Student

Use Case:student registration

Goal in context: Registration of a student

Brief Description: This use case is used when the student register himself/herself in the database online.

Preconditions:

- The student must access the website for this use case to begin.
- The user id must be unique and entered correctly.

Basic Flow:

The studententers the website.

The student fills his/her details from the following keys: -

- Student id
- password
- First/last name
- Status
- Gender
- DOB
- Contact no
- Qualification

- City
- Email1
- Email2
- Address
- Description
- Resume
- Image

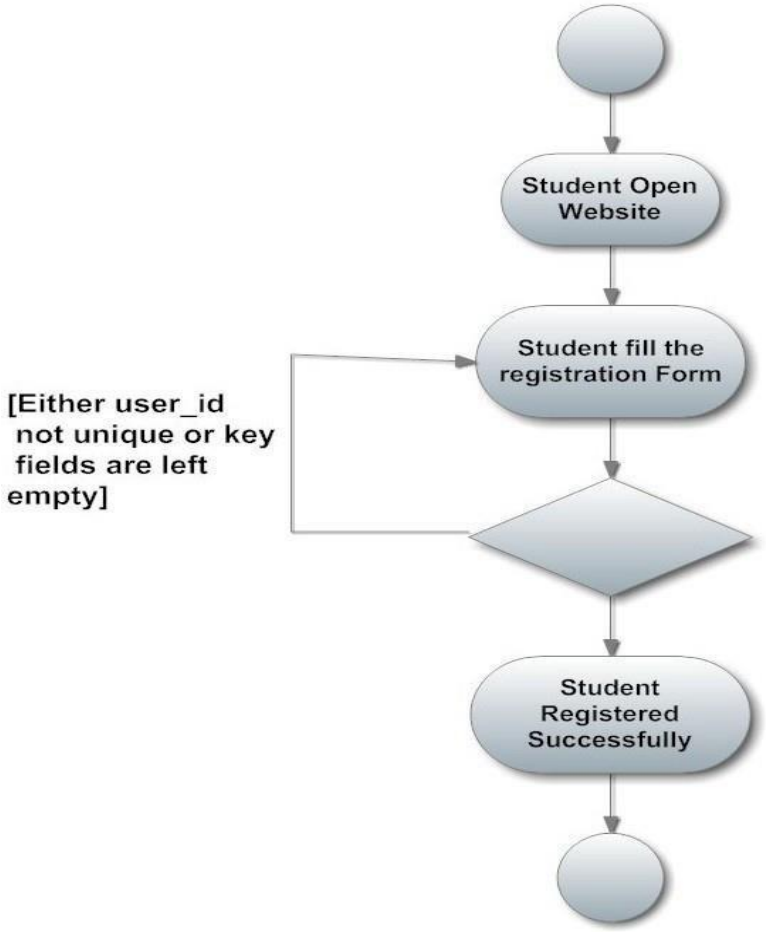
- The System details are added to the database.
- The student details are displayed on the screen.

Alternative Flow:

User ID not unique: if the user id entered is not unique then it will show an error message.

Post conditions:

The student gets registered on the website and to login into that particular the administrator must enable it.



Use Case Report- Register student on website

Use-case: Login into the website

Goal in context: Gain access to the website

Brief Description: This use case is used when the student wants to access the website

Preconditions: The Administrator must enable the student onto the website in order for this use case to begin.

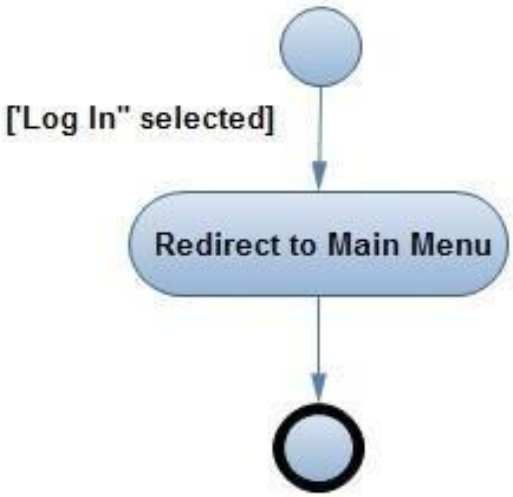
Basic Flow:

- The website prompts the student for the username and password.
- The student enters the username and password.
- The website verifies the password and sets the user's authorization.
- The student is given access to the website to perform his tasks.

Alternative Flow:

The student enters invalid username and password then he will not be allowed to enter the website**.**

Post conditions: The website state is unchanged by this use case.



Use Case Report- Login into the system

Use Case: Edit student details

Goal in context: Edit the details of a student

Brief Description: This use case is used when the student wants to edit the personal details of himself/herself already existing in the database.

Preconditions:

- The student must be logged into the system for this use case to begin.
- The details of the student must pre-exist in the database
- The student must be enabled by administrator.

Basic Flow:

- The student logs onto the System.
- The student can edit following keys: -

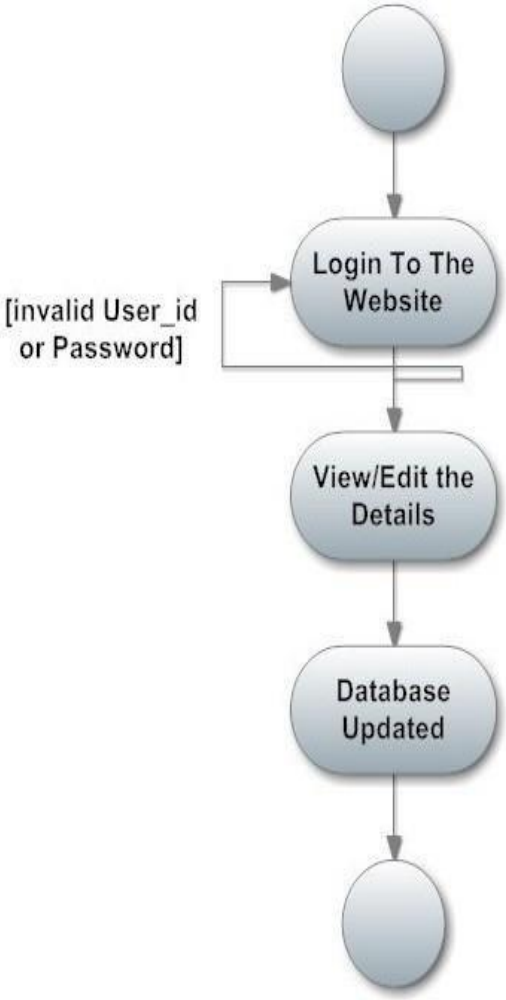
- First/last name
- Gender
- DOB
- Contact no
- Qualification
- City
- Email1
- Email2
- Address
- Description

- The Website updates the database according to edited details.
- The student details are edited in the database.

Alternative Flow: There is no alternative flow of this use case diagram.

Post conditions:

The student details get updated in the database**.**



Use Case Report- Edit Student Details into Database

SOURCE CODE

Controller

HomeController.cshtml

```
using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Web;
usingSystem.Web.Mvc;
usingQuizApp.Models;
using System.IO;

namespaceQuizApp.Controllers
{
public class HomeController : Controller
   {
       QuizAPPEntities1 db = new QuizAPPEntities1();

      [HttpGet]
```

```
publicActionResultAdmin_Signup()
        {
return View();
        }

        [HttpPost]
publicActionResultAdmin_Signup(Admin_tbsnup)
        {
if (ModelState.IsValid == true)
            {
db.Admin_tb.Add(snup);
int a = db.SaveChanges();
if (a > 0)
                {
ViewBag.InsertMessage = "<script>swal('Congradulations', 'Registered Successfully','success')</script>";
ModelState.Clear();
                }
else
                {
ViewBag.InsertMessage = "<script>alert('Registration Failed!!')</script>";
                }

            }
return View();
        }
```

```
        [HttpGet]
publicActionResultAdmin_login()
        {
return View();
        }


        [HttpPost]
publicActionResultAdmin_login(Admin_tb ad)
        {
Admin_tb admin = db.Admin_tb.Where(x =>x.ADMIN_NAME ==
ad.ADMIN_NAME&&x.ADMIN_PASSWORD == ad.ADMIN_PASSWORD).SingleOrDefault();
if (admin != null)
            {
Session["Admin_id"] = admin.ADMIN_ID.ToString();
Session["AdminName"] = admin.ADMIN_NAME;
TempData.Keep();
returnRedirectToAction("Dashboard");
            }
else
            {
                ViewBag.msg = "Invalid UserName Or Password!";
            }
return View();
        }
```

```
        [HttpGet]
publicActionResultAdd_Category()
        {
if (Session["Admin_id"]==null)
            {
returnRedirectToAction("Admin_login");
            }

            //Session["ad_id"]  = 2;//remove  it
intAdmin_id = Convert.ToInt32(Session["Admin_id"]);
            List<CATEGORY> cat = db.CATEGORies.Where(x=>x.CAT_ADMIN_ID == Admin_id).OrderBy(x
=>x.CAT_ID).ToList();
ViewData["list"] = cat;
return View();
        }

        [HttpPost]
publicActionResultAdd_Category(CATEGORY cat)
        {

            //------
            List<CATEGORY>cat_Li = db.CATEGORies.OrderBy(x =>x.CAT_ID).ToList();
ViewData["list"] = cat_Li;


            CATEGORY c = new CATEGORY();

            Random r = new Random();
```

```
c.CAT_NAME = cat.CAT_NAME;
c.CAT_ADMIN_ID = Convert.ToInt32(Session["Admin_id"].ToString());

c.cat_encrypted_string = Crypt.Encrypt(cat.CAT_NAME.Trim() + r.Next().ToString(), true);
db.CATEGORies.Add(c);
Session["catid"] = c.CAT_ID.ToString();
db.SaveChanges();

returnRedirectToAction("Add_Category");
    }

    [HttpGet]
publicActionResultAddQuestions()
    {

if (Session["Admin_id"] == null)
        {
returnRedirectToAction("Admin_login");
        }
intsid = Convert.ToInt32(Session["Admin_id"]);



        List<CATEGORY>cat_li = db.CATEGORies.Where(x=>x.CAT_ADMIN_ID== sid).ToList();
ViewBag.list = new SelectList(cat_li, "CAT_ID", "CAT_NAME");
```

```
            List<QUESTION>que = db.QUESTIONS.ToList();
ViewData["que"] = que;

return View();
        }

        [HttpPost]
publicActionResultAddQuestions(QUESTION q)
        {

int aid = Convert.ToInt32(Session["Admin_id"]);

TempData["QUE_ID"] = q.QUE_ID.ToString();
TempData.Keep();

            List<CATEGORY>cat_li = db.CATEGORies.Where(x =>x.CAT_ADMIN_ID == aid).ToList();

ViewBag.list = new SelectList(cat_li, "CAT_ID", "CAT_NAME");

            QUESTION qa = new QUESTION();

qa.QUE_TEXT = q.QUE_TEXT;
qa.OPT_A = q.OPT_A;
qa.OPT_B = q.OPT_B;
qa.OPT_C= q.OPT_C;
qa.OPT_D = q.OPT_D;
```

```
qa.CORRECT_OPT = q.CORRECT_OPT;

qa.QUE_CAT_ID = q.QUE_CAT_ID;

db.QUESTIONS.Add(qa);
db.SaveChanges();
        ViewBag.msg = "Question Added Successfully";
ModelState.Clear();
return View();
    }

    [HttpGet]
publicActionResultStudentSignup()
    {
return View();
    }

    [HttpPost]
publicActionResultStudentSignup(STUDENT_TBL st)
    {

string filename = Path.GetFileNameWithoutExtension(st.ImageFile.FileName);
string extension = Path.GetExtension(st.ImageFile.FileName);
HttpPostedFileBasepostedFile = st.ImageFile;
int length = postedFile.ContentLength;

if (extension.ToLower() == ".jpg" || extension.ToLower() == ".jpeg" || extension.ToLower() == ".png")
```

```
            {
if (length <= 1000000)
                {
filename = filename + extension;
st.STUDENT_IMAGE = "~/Content/Studentimg/" + filename;
filename = Path.Combine(Server.MapPath("~/Content/Studentimg/"),filename);
st.ImageFile.SaveAs(filename);
db.STUDENT_TBL.Add(st);
int a= db.SaveChanges();

if (a>0)
                    {
TempData["CreateMsg"] = "<script>alert('Registered Successfully.');</script>";
ModelState.Clear();
returnRedirectToAction("Student_login","Home");
                    }
else
                    {
TempData["CreateMsg"] = "<script>alert('Registration Failed...');</script>";
                    }
                }
else
                {
TempData["SizeMsg"] = "<script>alert('Image Size should be less than 1 MB');</script>";
                }
            }
else
```

```
            {
TempData["ExtensionMsg"] = "<script>alert('Format Not Supported');</script>";
            }



return View();
        }



publicActionResultStudent_login()
        {
return View();
        }

        [HttpPost]
publicActionResultStudent_login(STUDENT_TBL st)
        {
            STUDENT_TBL student = db.STUDENT_TBL.Where(x =>x.STUDENT_NAME ==
st.STUDENT_NAME&&x.STUDENT_PASSWORD == st.STUDENT_PASSWORD).SingleOrDefault();
if (student != null)
            {
Session["studentId"] = student.STUDENT_ID.ToString();
returnRedirectToAction("ExamDashboard");
            }
else
            {
```

```
            ViewBag.msg = "<script>alert('Invalid Name and Password..');</script>";
        }

return View();
    }



publicActionResult Dashboard()
    {
if (Session["Admin_id"] == null)
        {
returnRedirectToAction("Admin_login");
        }



return View();
    }

publicActionResultExamDashboard()
    {
if (Session["studentId"] == null)
        {
returnRedirectToAction("Student_login");
        }
return View();
    }
```

```
    [HttpPost]
publicActionResultExamDashboard(string room)
    {
        List<CATEGORY> list = db.CATEGORies.ToList();

foreach (var item in list)
        {
if (item.cat_encrypted_string==room)
            {

                List<QUESTION> li = db.QUESTIONS.Where(x =>x.QUE_CAT_ID == item.CAT_ID).ToList();
                Queue<QUESTION> queue = new Queue<QUESTION>();

foreach (QUESTION a in li)
                {
queue.Enqueue(a);
                }

TempData["questions"]=queue;

TempData["Score"] = 0;



                //TempData["examid"] = item.CAT_ID;
TempData.Keep();
```

```
returnRedirectToAction("StartQuiz");
            }

else
            {
ViewBag.error = "<script>alert('No Room Found. Please Enter Correct Room Name....');</script>";
            }
          }

return View();
      }



publicActionResultStartQuiz()
      {
if (Session["studentId"]==null)
          {
returnRedirectToAction("Student_login");
          }

          QUESTION q = null;

if (TempData["questions"]!=null)
          {
            Queue<QUESTION>qlist = (Queue<QUESTION>)TempData["questions"];
if (qlist.Count>0)
            {
```

```
                q = qlist.Peek();

qlist.Dequeue();
TempData["questions"] = qlist;
TempData.Keep();
            }
else
            {
returnRedirectToAction("EndExam");

            }
        }
else
        {
returnRedirectToAction("ExamDashboard");
        }

return View(q);

    }

    [HttpPost]
publicActionResultStartQuiz(QUESTION q)
    {

stringcorrectans = null;
```

```
if (q.OPT_A!=null)
        {
correctans = "A";
        }
else if(q.OPT_B!=null)
        {
correctans = "B";
        }
else if (q.OPT_C != null)
        {
correctans = "C";
        }
else if (q.OPT_D != null)
        {
correctans = "D";
        }

if (correctans.Equals(q.CORRECT_OPT))
        {
TempData["Score"] = Convert.ToInt32(TempData["Score"]) + 1;
        }

TempData.Keep();

returnRedirectToAction("StartQuiz");
      }
```

```
publicActionResultEndExam()
        {
return View();
        }

publicActionResultAdminLogout()
        {
Session.Abandon();
returnRedirectToAction("Index", "Home");
        }

publicActionResult Index()
        {
return View();
        }



publicActionResult About()
        {
ViewBag.Message = "Your application description page.";

return View();
        }

publicActionResult Contact()
```

```
        {
ViewBag.Message = "Your contact page.";


return View();
        }
    }
}
```

Views

Admit_Signup.cshtml

```
  @model QuizApp.Models.Admin_tb

@{
ViewBag.Title = "Admin_Signup";
    Layout = "~/Views/Shared/_Layout.cshtml";
}


<scriptsrc="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
<h2>Admin Sign Up</h2>

@Html.Raw(ViewBag.InsertMessage)

@using (Html.BeginForm())
{
@Html.AntiForgeryToken()

<divclass="form-horizontal">
<hr/>
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<divclass="form-group">
```

```
@Html.LabelFor(model =>model.ADMIN_NAME, htmlAttributes: new { @class = "control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.ADMIN_NAME, new { htmlAttributes = new { @class = "form-control" } })
@Html.ValidationMessageFor(model =>model.ADMIN_NAME, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Email, new { htmlAttributes = new { @class = "form-control" } })
@Html.ValidationMessageFor(model =>model.Email, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.ADMIN_PASSWORD, htmlAttributes: new { @class = "control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.ADMIN_PASSWORD, new { htmlAttributes = new { @class = "form-control" } })
@Html.ValidationMessageFor(model =>model.ADMIN_PASSWORD, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.ConfirmPassword, htmlAttributes: new { @class = "control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.ConfirmPassword, new { htmlAttributes = new { @class = "form-control" } })
@Html.ValidationMessageFor(model =>model.ConfirmPassword, "", new { @class = "text-danger" })
</div>
</div>
```

```
<divclass="form-group">
<divclass="col-md-offset-2 col-md-10">
<inputtype="submit"value="Sign Up"class="btnbtn-success"/>
</div>
</div>
</div>
}

<div>
<ahref="@Url.Action("Admin_login","Home")"class="btnbtn-primary">If you have Registered
already Please LOGIN</a>

</div>
```

Admin_Login.cshtml

```
@model QuizApp.Models.Admin_tb

@{
ViewBag.Title = "Admin_login";
}
<scriptsrc="~/Scripts/jquery-3.4.1.min.js"></script>
<scriptsrc="~/Scripts/jquery.validate.min.js"></script>
<script>
   $(document).ready(function () {
      $("#left-div").animate({ width: '30%' }, 900);
      $("#teacherimg").animate({ height: '450', width: '320', }, 900);
      $("#login_card").animate({ width: '70%', height: '450px' }, 900);
   });

</script>

<style>
#teacherimg{
margin-top:20px;

  }
.invalid-error {
```

```
font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode',
Geneva, Verdana, sans-serif;
    }
</style>


<divclass="col-lg-6 col-md-6 col-sm-6"id="left-div"style="width:50%; float:left;">

<imgsrc="~/Content/img/Teacher.png"id="teacherimg"height="500"width="400"alt="Teacher
image"/>


</div>


<divclass="card text-info mb-3 col-lg-6 col-md-6 col-sm-6"id="login_card"style="width: 5px;
height:5px; float: right; background-color: #fffffc; margin-top: 3%; border: 1pxsolid#fffffc;
border-radius: 10px; box-shadow: 4px4px15px-2pxrgba(0,0,0,0.78)";>

<divclass="card-body">
<div>
<center>
<pclass="invalid-error"style="color:red; margin-left:5px;">@ViewBag.msg</p>
</center>
<center>
<div>
<imgsrc="~/Content/img/logincard.png"alt="login image"width="200"height="200"/>
</div>
</center>
@using (Html.BeginForm())
        {
@Html.AntiForgeryToken()

<divclass="form-horizontal">
<center><h4> Please Login your Account</h4></center>
<hr/>
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<divclass="form-group">
@Html.LabelFor(model =>model.ADMIN_NAME, htmlAttributes: new { @class = "control-
label col-md-4" })
<divclass="col-md-8">
@Html.EditorFor(model =>model.ADMIN_NAME, new { htmlAttributes = new { @class =
"form-control" } })
```

```
@Html.ValidationMessageFor(model =>model.ADMIN_NAME, "", new { @class = "text-
danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.ADMIN_PASSWORD, htmlAttributes: new { @class =
"control-label col-md-4" })
<divclass="col-md-8">
@Html.EditorFor(model =>model.ADMIN_PASSWORD, new { htmlAttributes = new {
@class = "form-control" } })
@Html.ValidationMessageFor(model =>model.ADMIN_PASSWORD, "", new { @class =
"text-danger" })
</div>
</div>


<divclass="form-group">
<divclass="col-md-offset-4 col-md-8">
<inputtype="submit"value="Login"class="btnbtn-info"/>
</div>
</div>
</div>
        }



</div>
<br/>
<ahref="@Url.Action("Admin_Signup","Home")"class="btnbtn-primary"style="margin-
top:2px;">Go to Registration</a>
</div>
</div>


@section Scripts {
@Scripts.Render("~/bundles/jqueryval")
}
```

StudentSignup.cshtml

```
@model QuizApp.Models.STUDENT_TBL

@{
ViewBag.Title = "StudentSignup";
   Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>StudentSignup</h2>

@Html.Raw(TempData["CreateMsg"])
@Html.Raw(TempData["SizeMsg"])
@Html.Raw(TempData["ExtensionMsg"])


@using (Html.BeginForm("StudentSignup", "Home", FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
@Html.AntiForgeryToken()

<divclass="form-horizontal">
<hr/>
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<divclass="form-group">
@Html.LabelFor(model =>model.STUDENT_NAME, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.STUDENT_NAME, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.STUDENT_NAME, "", new { @class = "text-
danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.StudentEmail, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.StudentEmail, new { htmlAttributes = new { @class =
"form-control" } })
```

```
@Html.ValidationMessageFor(model =>model.StudentEmail, "", new { @class = "text-danger"
})
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.STUDENT_IMAGE, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
<inputtype="file"name="ImageFile"class="form-control btnbtn-primary"required/>
@Html.ValidationMessageFor(model =>model.STUDENT_IMAGE, "", new { @class = "text-
danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.STUDENT_PASSWORD, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.STUDENT_PASSWORD, new { htmlAttributes = new {
@class = "form-control" } })
@Html.ValidationMessageFor(model =>model.STUDENT_PASSWORD, "", new { @class =
"text-danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.ConfirmPassword, htmlAttributes: new { @class = "control-
label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.ConfirmPassword, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.ConfirmPassword, "", new { @class = "text-
danger" })
</div>
</div>

<divclass="form-group">
<divclass="col-md-offset-2 col-md-10">
<inputtype="submit"value="Signup"class="btnbtn-info"/>
</div>
</div>
</div>
}
```

```
<div>
@Html.ActionLink("If you have already Registered Please LOGIN", "Student_login")
</div>
```

Student_Login.cshtml

```
@model QuizApp.Models.STUDENT_TBL

@{
ViewBag.Title = "Student_login";
}

<h2style="color:deepskyblue;">Student Log In</h2>
@Html.Raw(ViewBag.msg);

<div>
@using (Html.BeginForm())
    {
@Html.AntiForgeryToken()

<divclass="form-horizontal">
<center><h4> Please Login your Account</h4></center>
<hr/>
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<divclass="form-group">
@Html.LabelFor(model =>model.STUDENT_NAME, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.STUDENT_NAME, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.STUDENT_NAME, "", new { @class = "text-
danger" })
</div>
</div>

<divclass="form-group">
```

```
@Html.LabelFor(model =>model.STUDENT_PASSWORD, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.STUDENT_PASSWORD, new { htmlAttributes = new {
@class = "form-control" } })
@Html.ValidationMessageFor(model =>model.STUDENT_PASSWORD, "", new { @class =
"text-danger" })
</div>
</div>

<divclass="form-group">
<divclass="col-md-offset-2 col-md-10">
<inputtype="submit"value="Login"class="btnbtn-info"/>
</div>
</div>
</div>
    }
</div>

@section Scripts {
@Scripts.Render("~/bundles/jqueryval")
}
```

Models

Admin Table

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------
usingSystem.ComponentModel.DataAnnotations;
namespaceQuizApp.Models
{
using System;
```

```
usingSystem.Collections.Generic;

publicpartialclassAdmin_tb
   {
      [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
publicAdmin_tb()
      {
this.CATEGORies = newHashSet<CATEGORY>();
      }

publicint ADMIN_ID { get; set; }

      [Required(ErrorMessage ="Admin Name is Required")]
      [Display(Name ="Admin Name")]
publicstring ADMIN_NAME { get; set; }

      [Required(ErrorMessage = "Admin Password is Required")]
      [Display(Name = "Admin Password")]
      [DataType(DataType.Password)]
publicstring ADMIN_PASSWORD { get; set; }

      [Required(ErrorMessage = "Confirm Password is Required")]
      [Display(Name = "Confirm Password")]
      [DataType(DataType.Password)]
publicstringConfirmPassword { get; set; }

      [Required(ErrorMessage = "Admin Email is Required")]
      [Display(Name = "Email")]
      [DataType(DataType.EmailAddress, ErrorMessage ="Email is not Valid")]
      [RegularExpression("^[a-zA-Z0-9_\\.-]+@([a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$",
ErrorMessage = "E-mail is not valid")]
publicstring Email { get; set; }

      [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
publicvirtualICollection<CATEGORY>CATEGORies { get; set; }
   }
}
```

Subject Category Table

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------
usingSystem.ComponentModel.DataAnnotations;
namespaceQuizApp.Models
{
using System;
usingSystem.Collections.Generic;

publicpartialclassCATEGORY
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
publicCATEGORY()
        {
this.QUESTIONS = newHashSet<QUESTION>();
        }

publicint CAT_ID { get; set; }

        [Required(ErrorMessage = "Admin Name is Required")]
        [Display(Name = "Subject")]
publicstring CAT_NAME { get; set; }
publicNullable<int> CAT_ADMIN_ID { get; set; }
publicstringcat_encrypted_string { get; set; }

publicvirtualAdmin_tbAdmin_tb { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
publicvirtualICollection<QUESTION> QUESTIONS { get; set; }
    }
}
```

Questions Table

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------
usingSystem.ComponentModel.DataAnnotations;
namespaceQuizApp.Models
{
using System;
usingSystem.Collections.Generic;

publicpartialclassQUESTION
    {
publicint QUE_ID { get; set; }

        [Required]
        [Display(Name = "Question")]
publicstring QUE_TEXT { get; set; }

        [Required]
        [Display(Name = "A")]
publicstring OPT_A { get; set; }
        [Required]
        [Display(Name = "B")]
publicstring OPT_B { get; set; }

        [Required]
        [Display(Name = "C")]
publicstring OPT_C { get; set; }

        [Required]
        [Display(Name = "D")]
publicstring OPT_D { get; set; }

        [Required]
        [Display(Name = "Correct Option")]
publicstring CORRECT_OPT { get; set; }
```

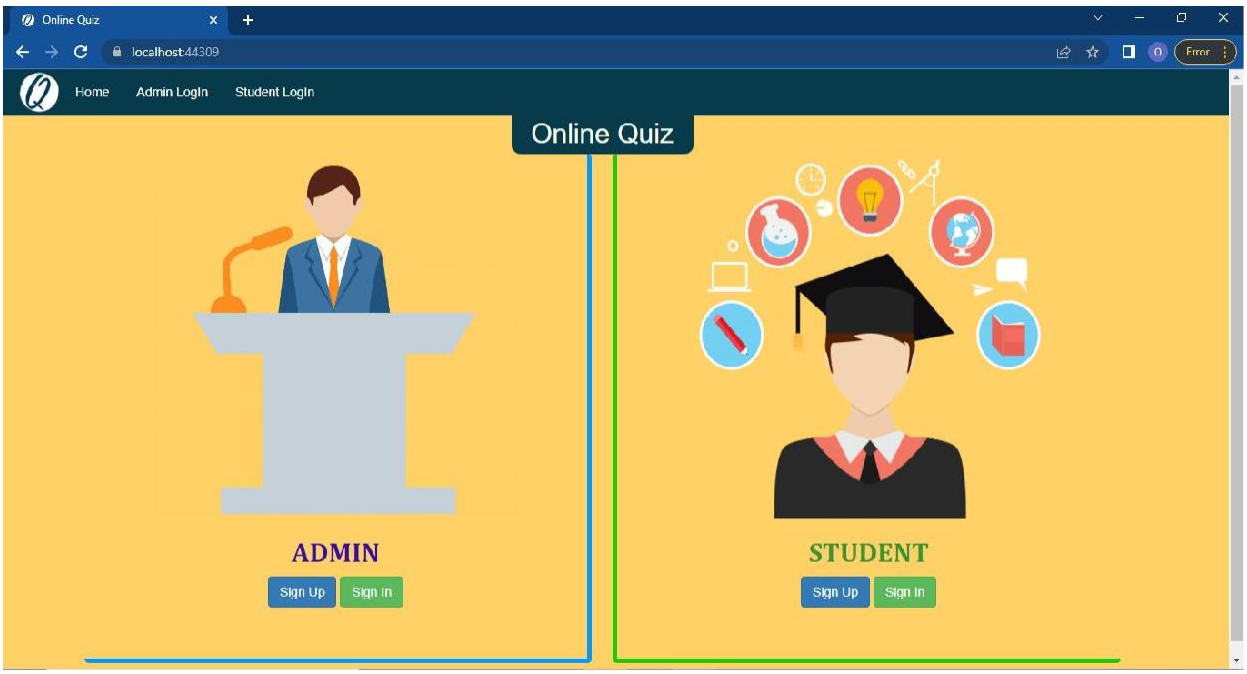
```
        [Required]
        [Display(Name = "Subject Category")]
publicNullable<int> QUE_CAT_ID { get; set; }


publicvirtual CATEGORY CATEGORY { get; set; }
    }
}
```

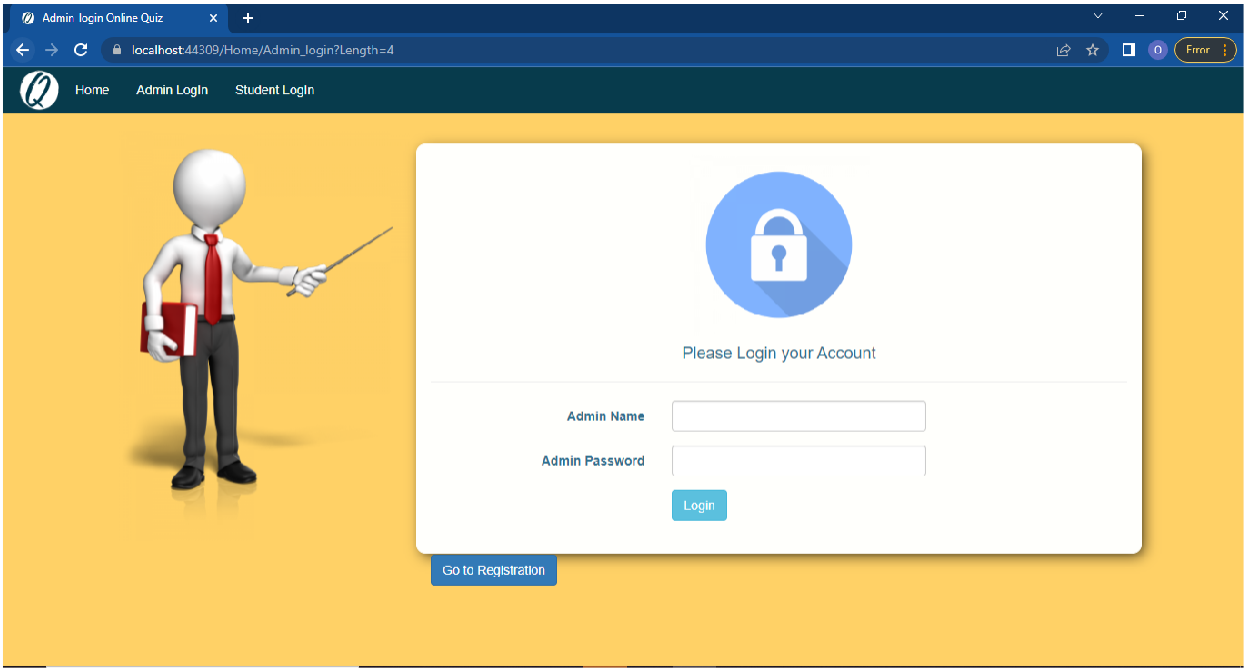
Student Table

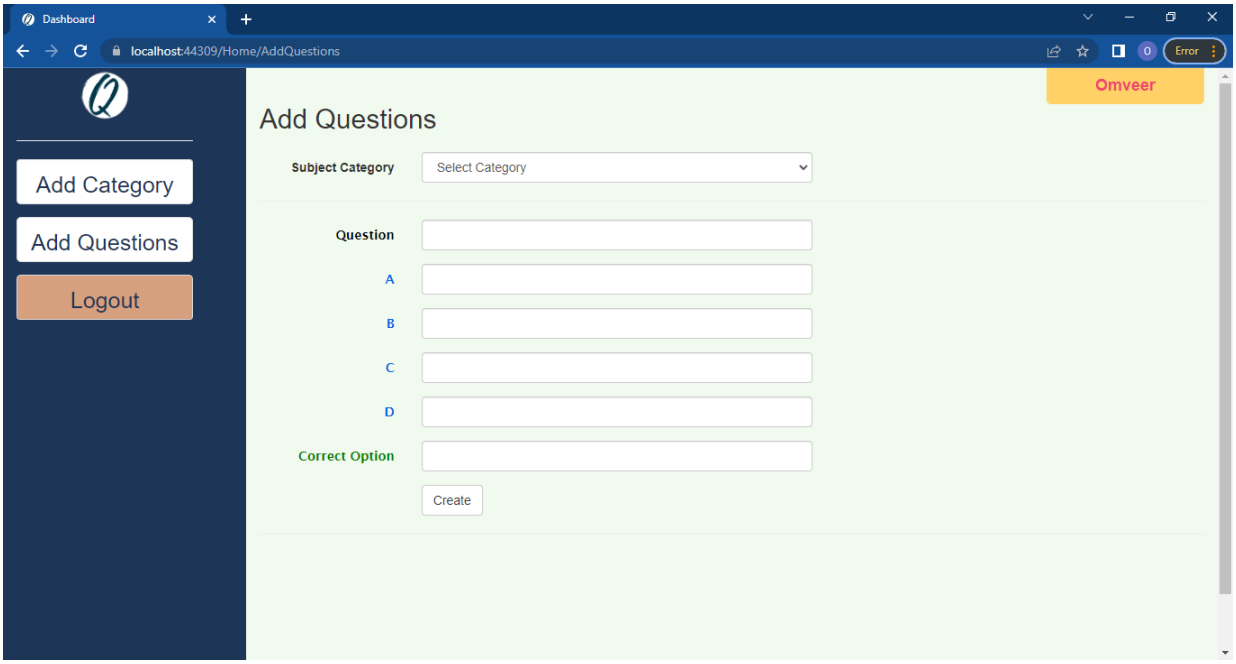
```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespaceQuizApp.Models
{
using System;
usingSystem.Collections.Generic;
usingSystem.Web;
usingSystem.ComponentModel.DataAnnotations;
publicpartialclassSTUDENT_TBL
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
publicSTUDENT_TBL()
        {
this.SET_EXAM = newHashSet<SET_EXAM>();
        }

publicint STUDENT_ID { get; set; }

        [Required(ErrorMessage = "Student Name is Required")]
```

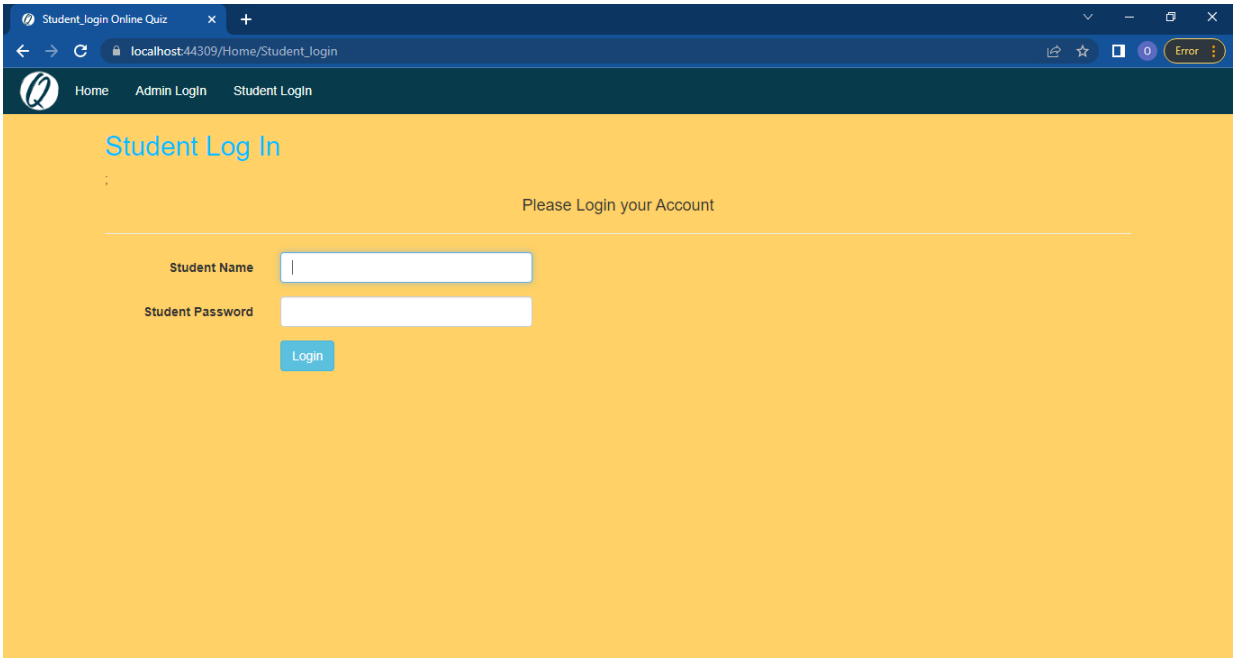
```
        [Display(Name = "Student Name")]
publicstring STUDENT_NAME { get; set; }

        [Required(ErrorMessage = "Student Password is Required")]
        [Display(Name = "Student Password")]
        [DataType(DataType.Password)]
publicstring STUDENT_PASSWORD { get; set; }

        [Required(ErrorMessage = "Student image is Required")]
        [Display(Name = "Student Image")]
publicstring STUDENT_IMAGE { get; set; }

        [Required(ErrorMessage = "Student Email is Required")]
        [Display(Name = "Email")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Email is not Valid")]
        [RegularExpression("^[a-zA-Z0-9_\\.-]+@([a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$",
ErrorMessage = "E-mail is not valid")]
publicstringStudentEmail { get; set; }

        [Required(ErrorMessage = "Confirm Password is Required")]
        [Display(Name = "Confirm Password")]
        [DataType(DataType.Password)]
publicstringConfirmPassword { get; set; }

publicHttpPostedFileBaseImageFile { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
publicvirtualICollection<SET_EXAM> SET_EXAM { get; set; }
    }
}
```

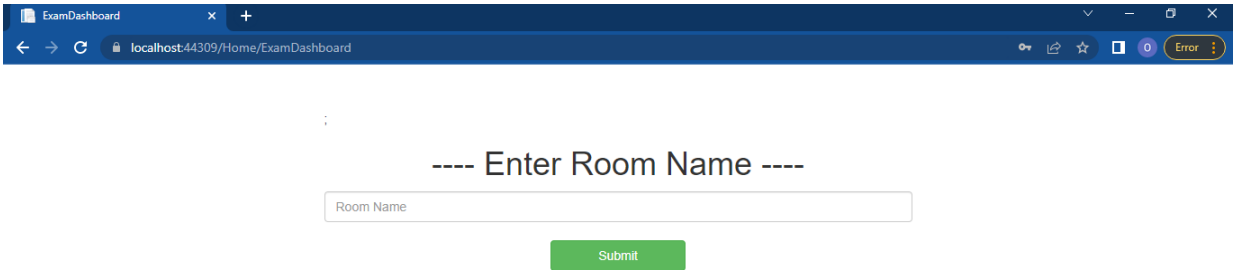
SNAPSHOTS


Online Quiz
localhost:44309
Home
Admin LogIn
Student LogIn
Online Quiz
ADMIN
Sign Up
Sign In
STUDENT
Sign Up
Sign In


Admin login Online Quiz
localhost:44309/Home/Admin_login?Length=4
Home
Admin Login
Student Login
Please Login your Account
Admin Name
Admin Password
Login
Go to Registration

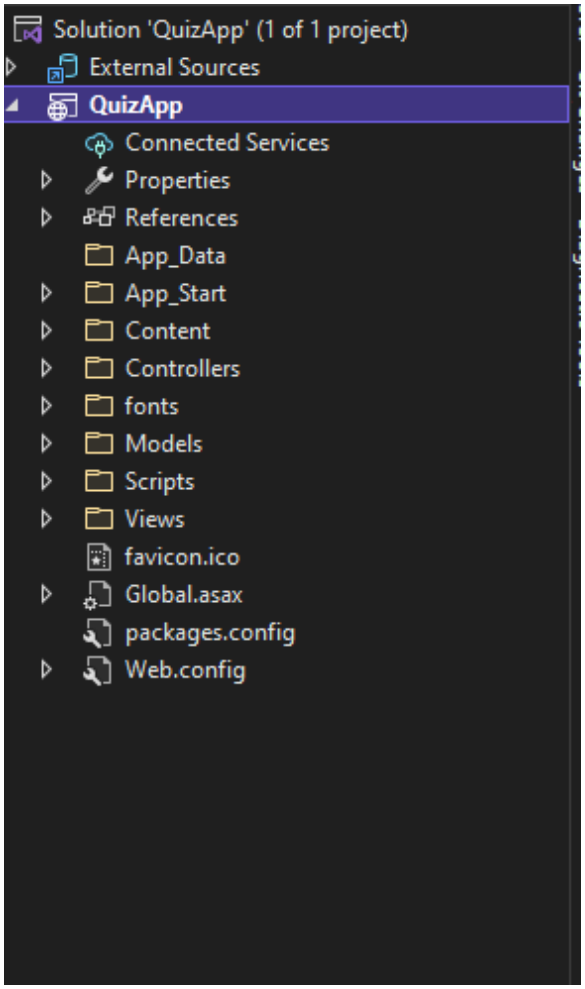

Dashboard
localhost:44309/Home/AddQuestions
Error
Omveer
Add Category
Add Questions
Logout
Add Questions
Subject Category
Select Category
Question
A
B
C
D
Correct Option
Create

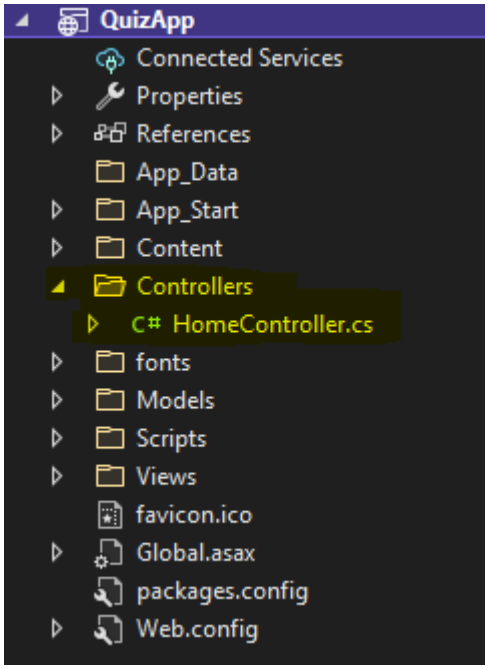

Student_login Online Quiz
localhost:44309/Home/Student_login
Error
Home
Admin LogIn
Student LogIn
Student Log In
Please Login your Account
Student Name
Student Password
Login

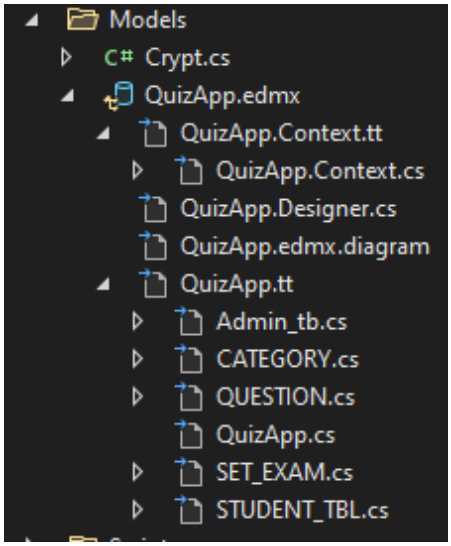

ExamDashboard
localhost:44309/Home/ExamDashboard
Error
---- Enter Room Name ----
Room Name
Submit

Project Folders

Folders




QuizApp
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
HomeController.cs
fonts
Models
Scripts
Views
favicon.ico
Global.asax
packages.config
Web.config


Models
Crypt.cs
QuizApp.edmx
QuizApp.Context.tt
QuizApp.Context.cs
QuizApp.Designer.cs
QuizApp.edmx.diagram
QuizApp.tt
Admin_tb.cs
CATEGORY.cs
QUESTION.cs
QuizApp.cs
SET_EXAM.cs
STUDENT_TBL.cs

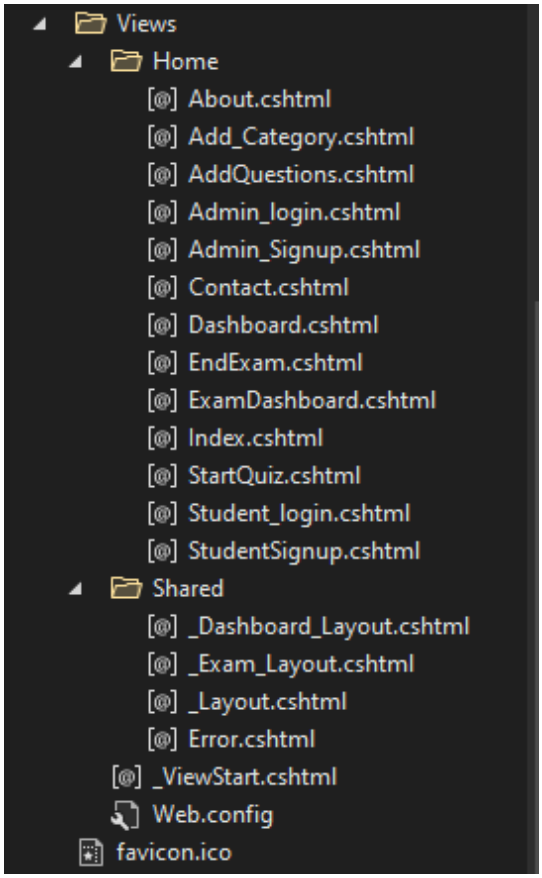
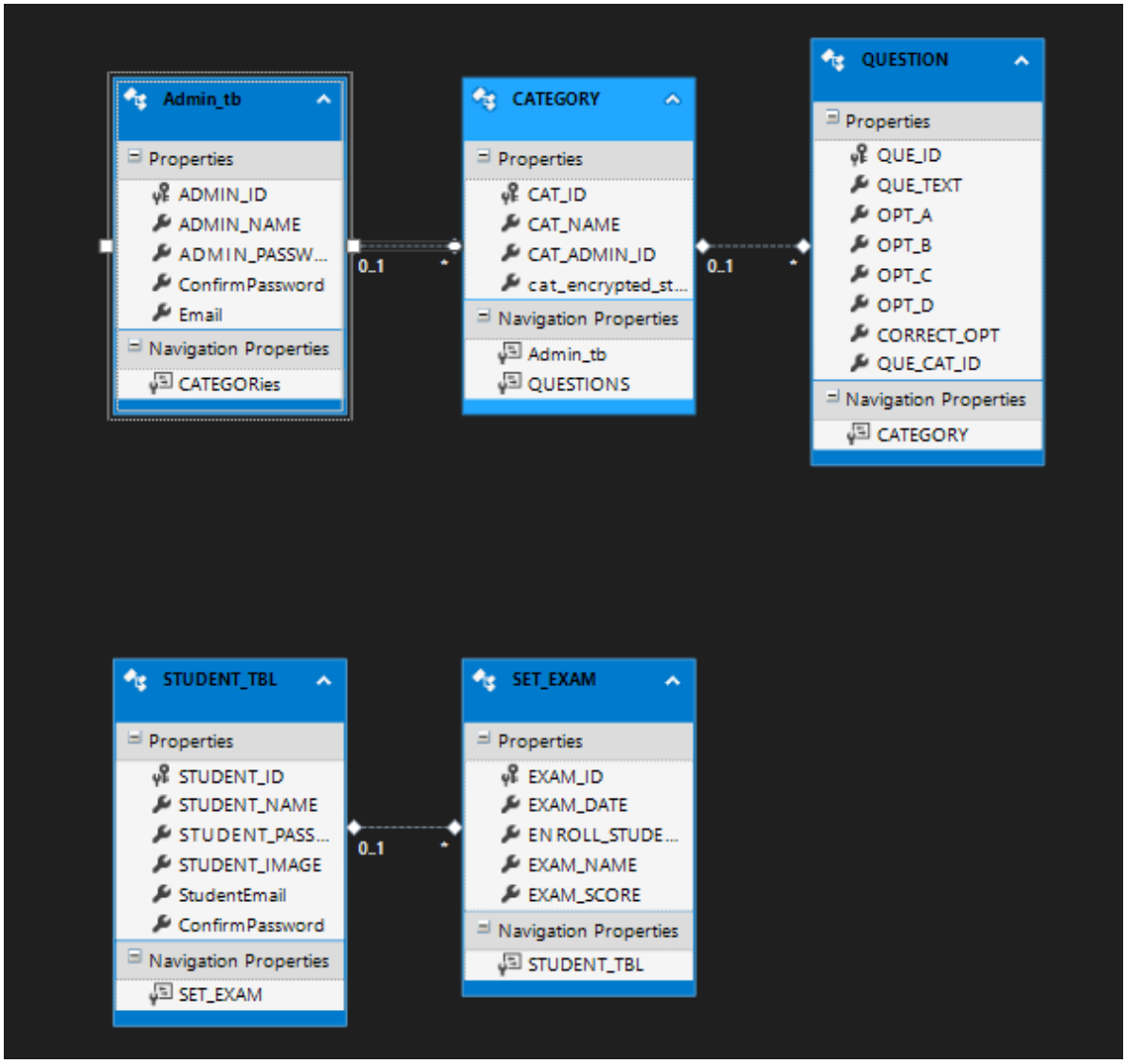

Views
Home
About.cshtml
Add_Category.cshtml
AddQuestions.cshtml
Admin_login.cshtml
Admin_Signup.cshtml
Contact.cshtml
Dashboard.cshtml
EndExam.cshtml
ExamDashboard.cshtml
Index.cshtml
StartQuiz.cshtml
Student_login.cshtml
StudentSignup.cshtml
Shared
_Dashboard_Layout.cshtml
_Exam_Layout.cshtml
_Layout.cshtml
Error.cshtml
_ViewStart.cshtml
Web.config
favicon.ico

Table Structure



Testing

Test Cases

- **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.

  Usually Test Engineers are involved in the black box testing.

- **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.

  Usually The Developers are involved in white box testing.

- **Gray Box Testing**: is the process in which the combination of black box and white box techniques are used.

- **Smoke Testing**: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)
- **Sanity Testing**: is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.
- **Regression Testing**: is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.
- **Re-Testing**: is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.
- **Static Testing**: is the testing, which is performed on an application when it is not been executed. ex: GUI, Document Testing

- **Alpha Testing**: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.
- **Beta-Testing**: it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.
- **Monkey Testing**: is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.
- **Compatibility testing**: it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.
- **Installation Testing**: it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.
- **Adhoc Testing**:Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmatic issues.

TCD (Test Case Document):

Test Case Document Contains

- Test Scope (or) Test objective
- Test Scenario
- Test Procedure
- Test case

This is the sample test case document for the Acadamic details of student project:

Test scope:

- Test coverage is provided for the screen " Acadamic status entry" form of a student module of university management system application

- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively