**Feedback Management System**

**A Project Report submitted in partial fulfillment of the requirements**

**for the award of the degree of**

**MASTER OF COMPUTER APPLICATION**

**Submitted By**

**Kartikey Sharma MCA IV SEM (2184200045)**

**Praveen Kumar MCA IV SEM (2184200066)**

**Abhishek Gupta MCA IV SEM (2184200002)**

**Under the Guidance of**

**Mr. Anshy Singh**

**Associate Director of Software Development Cell**

**Department of Computer Engineering & Applications**

****

**GLA UNIVERSITY**

**Mathura- 281406, INDIA**

**May, 2023**

**Declaration**

I hereby declare that the work which is being presented in the **MCA** Project **“Feedback Management System”** in partial fulfillment of the requirements for the award ofthe **Masters of Computer Application** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of **GLA University, Mathura**, is an authentic record of my own work carried under the supervision of Mr. Anshy SinghThe contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.I also declare that the contents of this project report are based on my own work and research carried out during the course of the project work. Any reference or contribution made by others has been duly acknowledged in the project report. I understand that in the event of any of the above information being found false or incorrect, the College has the right to reject my project report and take necessary disciplinary action.

Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Candidate: Praveen Kumar Name of Candidate: Kartikey Sharma

University Roll No.:2184200066 University Roll No.:2184200045

Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Candidate: Abhishek Gupta

University Roll No.:2184200002

**Certificate**

This is to certify that the project report entitled **Feedback Management System** submitted by **Abhishek Gupta** to **GLA UNIVERSITY** in partial fulfillment of the requirement for the award of **MCA** is a Bonafede record of the project work carried out by him/her under our guidance and supervision. The project report has been submitted after satisfactory evaluation. To the best of our knowledge, this work has not been submitted elsewhere for the award of any other degree, diploma or fellowship. This is to certify that the above statements made by the candidate are correct to the best of my/our knowledge and belief.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor**

**Mr. Anshy Singh**

**Associate Director**

**(Software Development Cell)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Coordinator Head of Department**

**(Mr. Gaurav Sharma) (Prof. Rohit Agarwal)**

**ACKNOWLEDGEMENTS**

We would like to thank **Mr. Anshy Singh**, (Associate Director Software Development Cell) for their support and guidance in completing our project on the topic (**Feedback Management System**). It was a great learning experience. I would like to take this opportunity to express my gratitude to all Software Development Team members **Mr. Hemanta Kuiry, Ms. Ankita Srivastava, Mr. Adesh Upadhyay and Mr. Ashish Sharma.** The project would not have been successful without their cooperation and inputs. I would like to extend my sincere thanks to my parents, family members, and friends for their constant support, encouragement, and motivation throughout the course of this project work. Finally, I would like to thank all those who have contributed to this project work, directly or indirectly, and helped me in successfully completing it.

Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Candidate: Praveen Kumar Name of Candidate: Kartikey Sharma

University Roll No.:2184200066 University Roll No.:2184200045

Sign \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Candidate: Abhishek Gupta

University Roll No.:2184200002

**ABSTRACT**

In this project, we are creating a Feedback management system where students can share their views and experience. Every individual student has an equal chance to share their views on the topic/matter. The purpose of Feedback Management System 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. The methodology involved in the project includes the study of the existing feedback systems, designing the database schema, developing the user interface, and implementing the feedback submission and analysis modules. The system includes features such as student registration, course feedback submission, dashboard for analysis of feedback, and report generation. The findings of the project suggest that the Feedback Management System is an efficient and effective tool for educational institutions to collect, manage and analyze student feedback. The system provides real-time analysis of feedback data and generates reports that can be used for decision-making and quality improvement. In conclusion, the Feedback Management System developed in this project is a useful tool for educational institutions to collect and manage student feedback. The system provides an efficient and effective way to improve the quality of education and services offered by the institution. The system can be easily customized and integrated into the existing institutional infrastructure.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.NO**. | **Title** | **Page No.** |
| 1. | Chapter 1: Introduction | 6 |
| 2. | Chapter 2: Software Requirement Analysis | 7 |
| 3. | Chapter 3: Software Design | 10 |
| 4. | Chapter 4: Implementation and User Interface | 14 |
| 5. | Chapter 5: Software Testing | 21 |
| 6. | Chapter 6: Conclusion | 28 |

**CONTENTS**

Declaration i Certificate ii Acknowledge iii Abstract iv List of Chapter v CHAPTER 1 Introduction vi

1.1 Overview and Motivation

1.2 Objective

1.3 Summary of Similar Application

1.4 Organization of the Project

CHAPTER 2 Software Requirement Analysis vii 2.1 Technical Feasibility

2.2 Summary of Modules

2.3 Modules

2.4 Security Features

CHAPTER 3 Software Design viii

3.1 DFD

I. 0 Level Or 1 level

3.2 Use Case Diagram

CHAPTER 4 Implementation and User Interface ix CHAPTER 5 Software Testing x CHAPTER 6 Conclusion xi REFERENCES xii

**CHAPTER 1: Introduction**

The "Feedback Management System" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides an error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly.

Feedback Management System, as described above, can lead to error free, secure, reliable and fast management systems. It can assist the user to concentrate on their other activities rather than concentrating on the record keeping. Thus, it will help organizations in better utilization of resources.

This report outlines the feedback management system used in our recent project. The purpose of this report is to provide an overview of the feedback collected from stakeholders and to analyze the effectiveness of the system in managing that feedback. The scope of this report includes a review of the project’s objectives, the methods used to collect feedback, and an evaluation of the feedback management system.

The project aimed to provide a new service to our clients, and the feedback received was critical in ensuring that the final product met their needs. The feedback was collected through a variety of methods, including surveys, interviews, and focus groups. This report will analyze the feedback collected from these methods and provide recommendations for future projects.

The report is structured into several sections, starting with an overview of the project and its objectives. This is followed by a description of the feedback management system and an analysis of the feedback collected. Finally, the report concludes with recommendations for improving the feedback management system and suggestions for future projects.

**1.1Overview and Motivation**:

**Overview:**

The purpose of this project feedback management system report is to provide an overview of the feedback received during a recent college project and to analyze the effectiveness of the feedback management system used. The project aimed to identify areas of improvement in the college's current academic program and to gather feedback from stakeholders, including students, faculty, and staff.

To collect feedback, a variety of methods were employed, including surveys, focus groups, and interviews. The feedback was then analyzed and evaluated for common themes and areas of concern. The report's structure is divided into several sections, starting with an introduction to the project's goals and objectives. This is followed by a description of the feedback management system and an analysis of the feedback collected.

The report concludes with recommendations for improving the feedback management system and suggestions for future projects. The information presented in this report will be used to make informed decisions about the college's academic program and to improve the overall quality of education provided to students.

**Motivation:**

The motivation behind this project was to identify areas of improvement in the college's current academic program and to gather feedback from stakeholders, including students, faculty, and staff. It is important to collect feedback to ensure that the college is meeting the needs of its stakeholders and to identify areas for improvement.

Feedback is critical to the success of any project or program, and the college recognizes the importance of collecting feedback to improve the overall quality of education provided to students. By collecting feedback from a variety of stakeholders, the college can gain a better understanding of the needs and concerns of its community and can make informed decisions about how to improve its academic program.

Furthermore, this project is motivated by a desire to continuously improve the college's academic program to ensure that it remains competitive and provides the best possible education to its students. By identifying areas for improvement and implementing changes based on feedback, the college can stay up-to-date with current trends and technologies and provide a high-quality education that prepares students for success in their careers.

Overall, the motivation behind this project is to gather feedback from stakeholders, identify areas for improvement, and continuously improve the college's academic program to meet the needs of its community and prepare students for success in their future careers.

**1.2Objective:**

The main objective of this project feedback management system report is to collect feedback from stakeholders, including students, faculty, and staff, and to analyze the effectiveness of the feedback management system used. Specifically, the objectives of this report are:

1. To identify areas of improvement in the college's current academic program based on feedback from stakeholders
2. To evaluate the effectiveness of the feedback management system used to collect feedback
3. To make recommendations for improving the feedback management system and for future projects

By achieving these objectives, the college will be able to make informed decisions about how to improve its academic program and ensure that it is meeting the needs of its stakeholders. Additionally, the evaluation of the feedback management system used will provide insights into its effectiveness and identify areas for improvement to ensure that future projects are successful in collecting and utilizing feedback.

Overall, the objectives of this project feedback management system report are to improve the quality of education provided to students by identifying areas for improvement, evaluating the effectiveness of the feedback management system, and making recommendations for future projects.

**1.3 Summary if Similar Application:**

To write a summary for a project feedback management system report for college purpose, you should follow these steps:

1. Start by introducing the project and its objectives briefly.
2. Provide a brief description of the methodology used in the project.
3. Highlight the key findings and outcomes of the project.
4. Mention the limitations and challenges faced during the project.
5. Emphasize the significance of the project and its potential impact on the field.
6. Conclude by summarizing the main points of the report.

Here is an example summary for a project feedback management system report:

The project aimed to develop a feedback management system that enables students to provide feedback on their courses and faculty members to manage and respond to the feedback. The methodology used included a literature review, user requirements gathering, system design, implementation, and testing. The key findings of the project indicate that the system is user-friendly, improves communication between students and faculty, and enhances the quality of education. However, the system faced some limitations, such as data privacy concerns and technical difficulties. Overall, the feedback management system has the potential to revolutionize the way feedback is managed in educational institutions.

|  |  |
| --- | --- |
|  |  |

**1.4 Organization of Project Report:**

PHASES TIME DURATION

Software requirement specification 1 week

System design 1 week

Coding 3 week

Testing 1 week

Documentation 1 week

Implementation 1 week

**Chapter:2 Software Requirement Analysis**

Systems analysis is a detailed study of the various operations performed by a system and their Relationship inside and outside the system. It is a systematic technique that defines goals. The aim of the development is to deliver the system user-friendly

System study has been conducted with the following objectives in mind: -

* Students need to complete its login to get into the feedback page.
* Students should be able to access the site after authentication is fulfilled.
* Admin should be able to delete, update and also reserve the system for his own knowledge.
* Both hardware and software expertise are required to successfully attain the objectives.

**2.1 Requirement Analysis**

Requirements analysis encompasses those tasks that go into. determining the needs or conditions to meet for a new or altered product or project, taking account of the possibility. conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or systems. requirements

The major steps/need for this requirement analysis are: –

● Defining student requirements.

● Studying the present system to verify the problem.

● Defining the performance expected by the student to use the requirement

**2.1.1 Hardware requirements**

Processor: Intel Core i3 or more.

Processor Speed: 1.5 GHz.

RAM: 8 GB.

Hard Disk: 15 -20 GB of free space

**2.1.2 Software requirements**

Operating System: Window 7 and higher.

**2.1.3 Tools and Technology Tools:**

**Tools:**

* Windows 7 & higher
* Visual Studio
* MySQL

**Technology:**

* Bootstrap5
* Angular
* Html5
* CSS
* JS
* Asp.net
* C#
* MVC
* SQL

**2.2 Feasibility Study**

After doing the project Feedback Management System, study and analyze all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

**2.2.1 Technical Feasibility:**

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different types of frontend and backend platforms.

**2.2.2 Operational Feasibility:**

No doubt the proposed system is fully GUI based and is very user friendly and all inputs to be taken are all self-explanatory even to a layman. Besides, proper training has been conducted to let the users know the essence of the system so that they feel comfortable with the new system. As far as our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

**2.2.3 Economic Feasibility:**

This is a very important aspect to be considered while developing a project. We decided the technology based on the minimum possible cost factor. All hardware and software cost has to be borne by the organization. Overall, we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for the system.

**2.3 Modules:**

**Admin Module:** The admin module is designed to provide administrative access to the system. The admin can manage user accounts and access privileges, add new courses to the system, view and manage feedback data, and generate reports on feedback data. The user management feature allows the admin to create and manage user accounts, assign access privileges, and reset passwords.

**User Module:** The user module is designed to provide feedback submission and tracking capabilities to students. The user can submit feedback on courses and faculty members, view their feedback history, receive notifications on feedback updates and system maintenance, and update their personal information.

**About Graph:** The High-charts library comes with all the tools you need to create reliable and secure data visualizations. Built on JavaScript and TypeScript, all our charting libraries work with any back-end database or server stack. We offer wrappers for the most popular programming languages (.Net, PHP, Python, R, Java) as well as iOS and Android, and frameworks like Angular, Vue, and React.

**2.4 Security Features:**

Authentication is the process of verifying the identity of a user or system, and ensuring that they have the appropriate level of access to information or resources. There are several security features that can be implemented to enhance authentication security. Here are some of the common ones:

1. Password policies: Enforce strong password policies that require users to create complex passwords that include a combination of letters, numbers, and special characters. Password policies can also enforce password expiration and prevent the reuse of previous passwords.
2. Session management: Implement session management to ensure that user sessions are securely managed. Session management can include measures such as session timeouts, forced logouts, and session token expiration.
3. User behavior analytics (UBA): Implement user behavior analytics (UBA) to monitor user activity and identify anomalies that may indicate a security threat. UBA can detect suspicious login activity, such as login attempts from unusual locations or devices, and notify security personnel.
4. Risk-based authentication (RBA): Implement risk-based authentication (RBA) to assess the risk level of a user's authentication request based on factors such as their location, device, and behavior. RBA can then adjust the authentication process to increase security for high-risk requests.

By implementing these security features, you can enhance the security of your authentication process and reduce the risk of unauthorized access.

**CHAPTER 3: Software Design**

**ACTIVITY DIAGRAM**

**Use Case Diagram:**

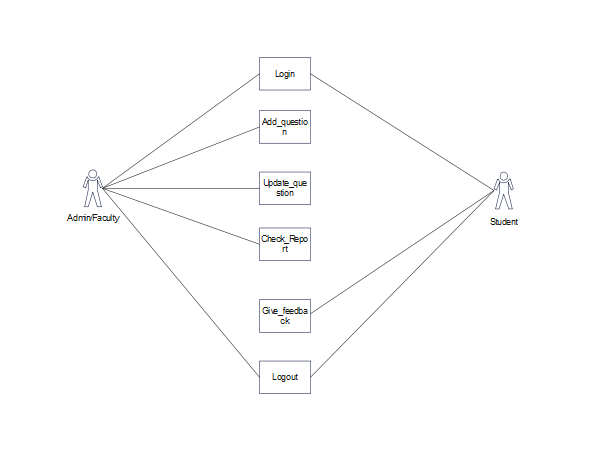
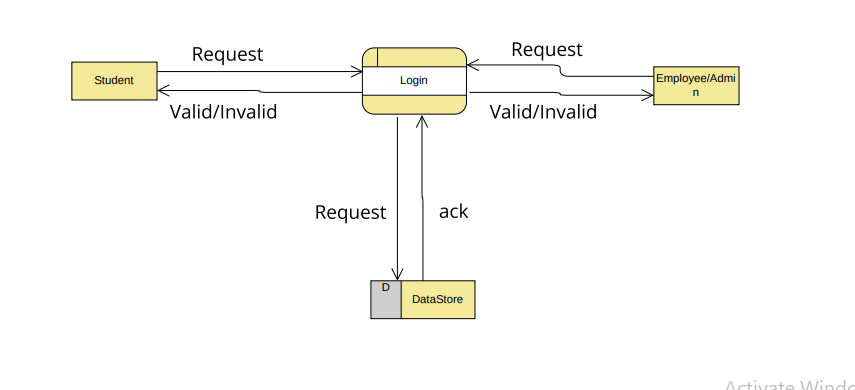


Figure 3.1

**0 Level DFD:**



Figures 3.2

**1 Level DFD:**

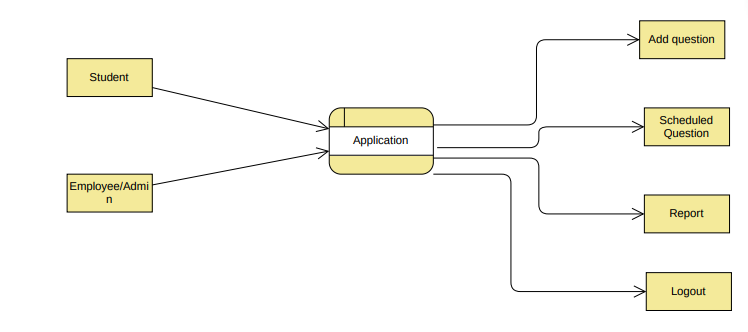


Figure 3.3

**E-R Diagram for Admin/Staff:**

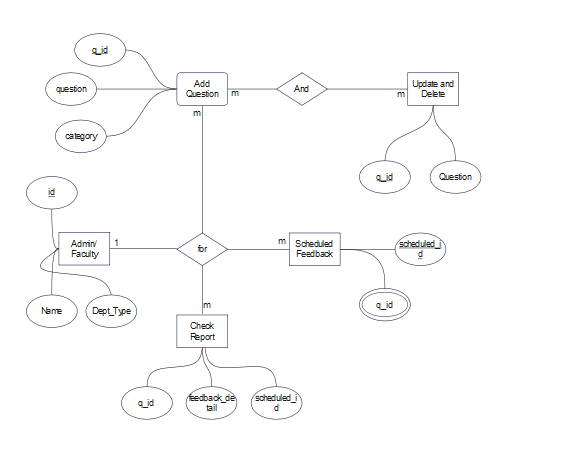


Figure 3.4

**E-R Diagram for Student:**

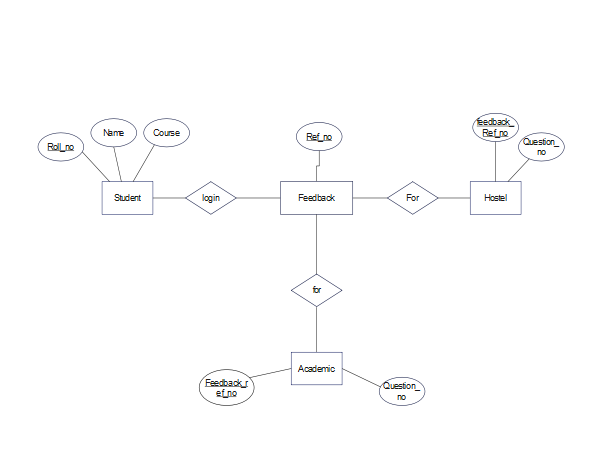


Figure 3.5

Primary Key:

* Roll\_no
* Id
* Q\_Id
* Scheduled\_id

Reference Key:

* Q\_Id

Multi Valued Attribute:

* Q\_id As a Reference in Scheduled Feedback Table

**CHAPTER 4: Implementation and User Interface**

# 

Figure 4.1

# 

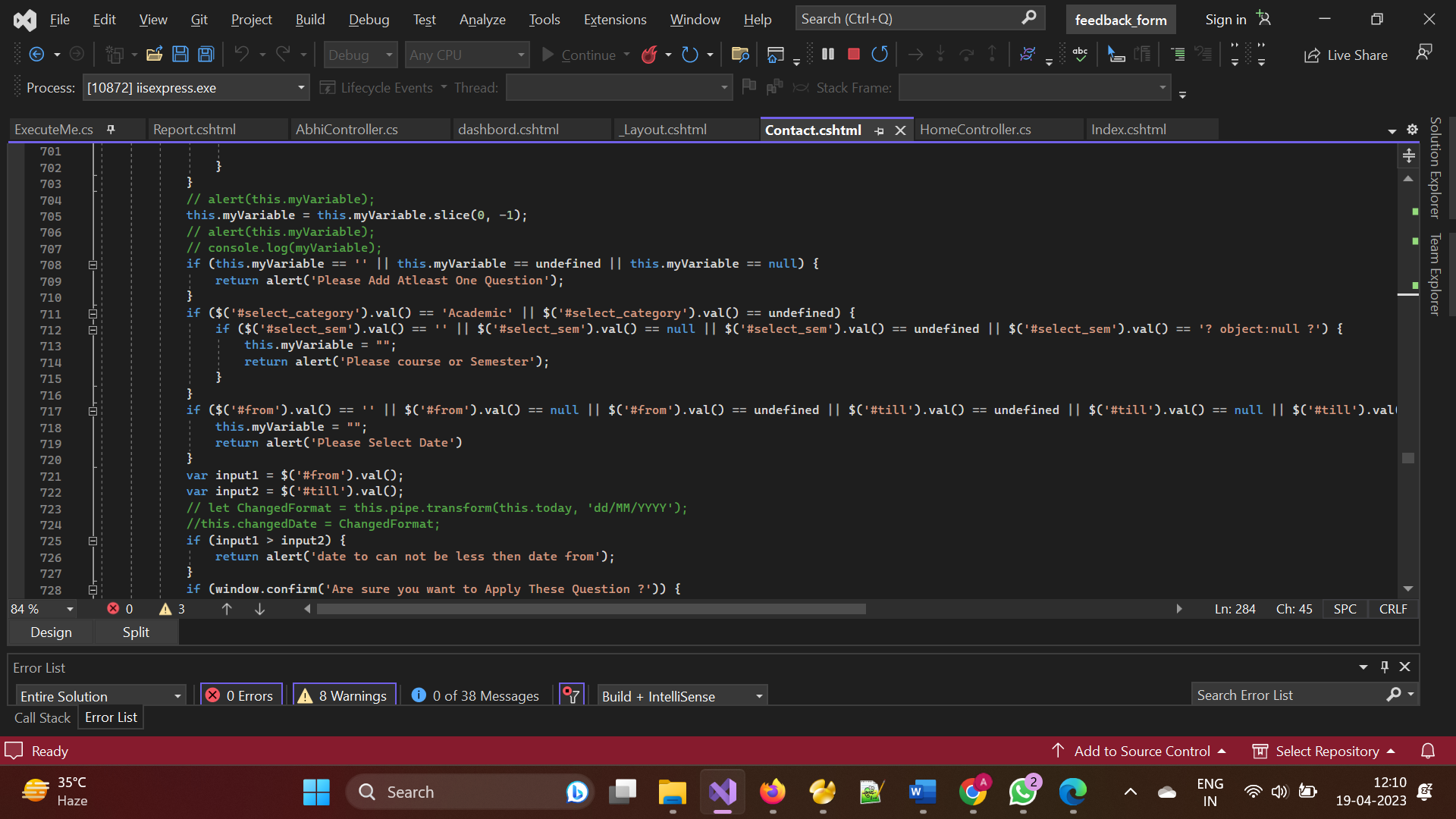


Figure 4.2

# 

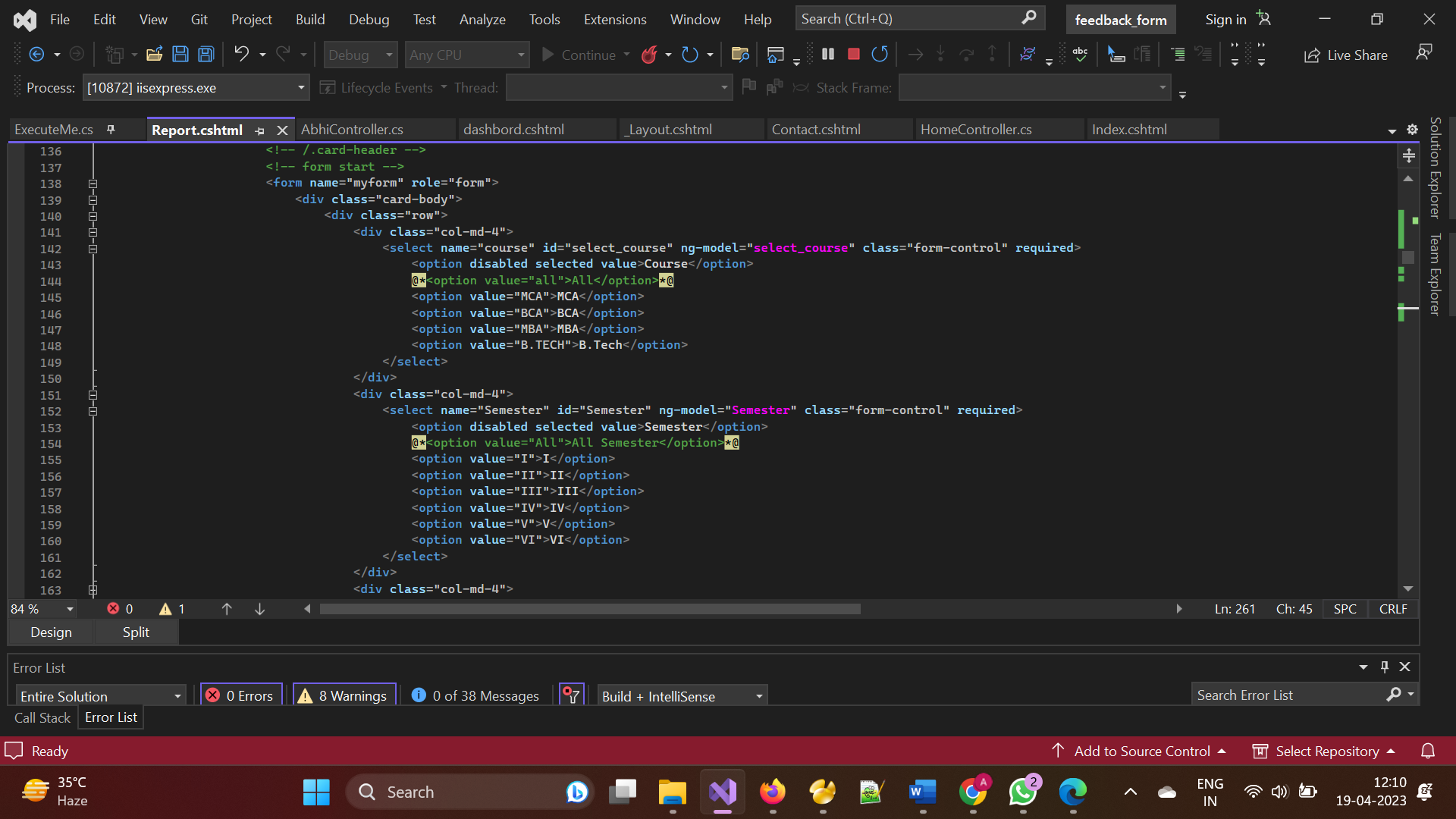


Figure 4.3

**Home page**

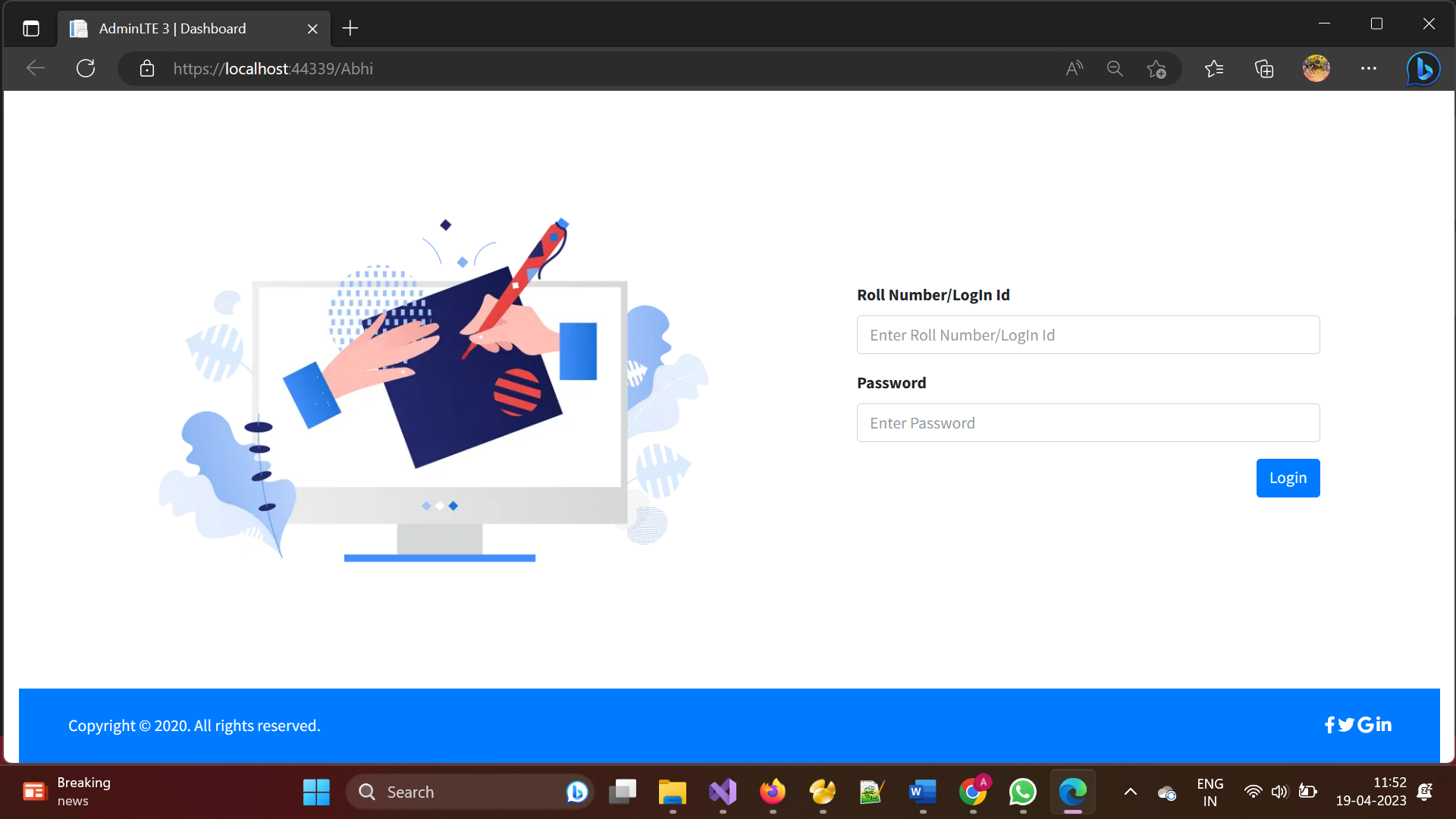


Figure 4.4

**Student Panel**

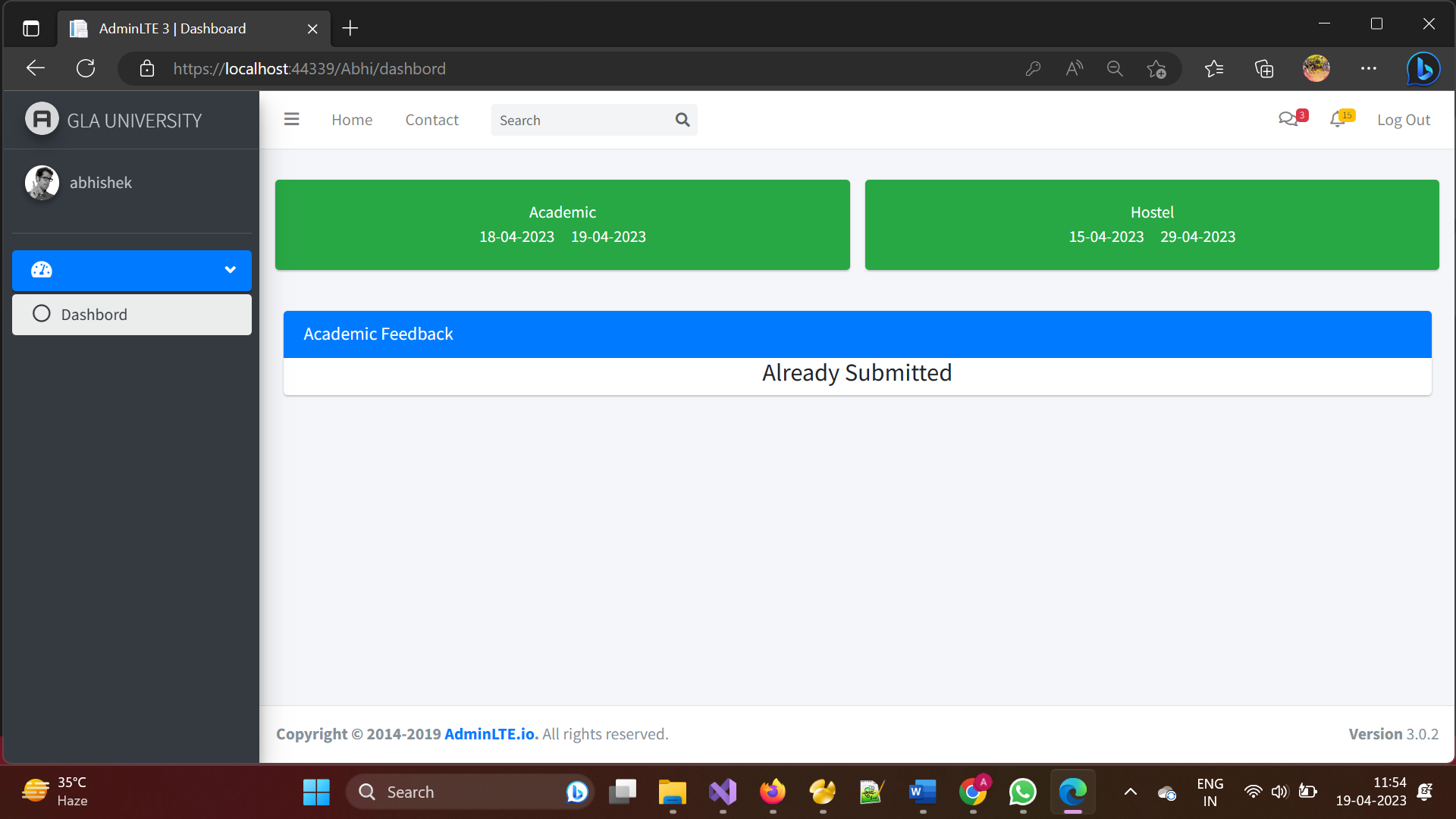


Figure 4.5

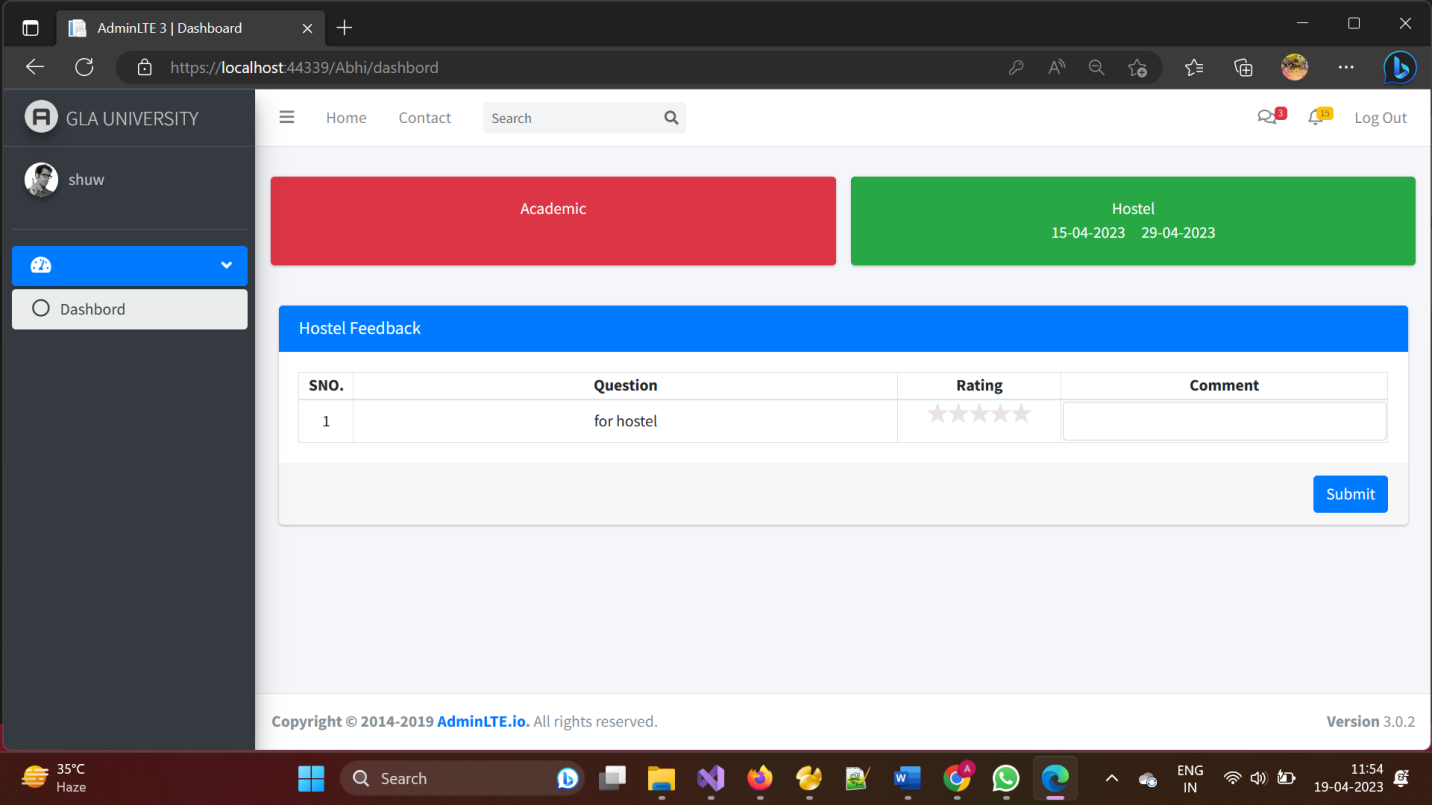


Figure 4.6

**Admin/­­­­Staff Panel**

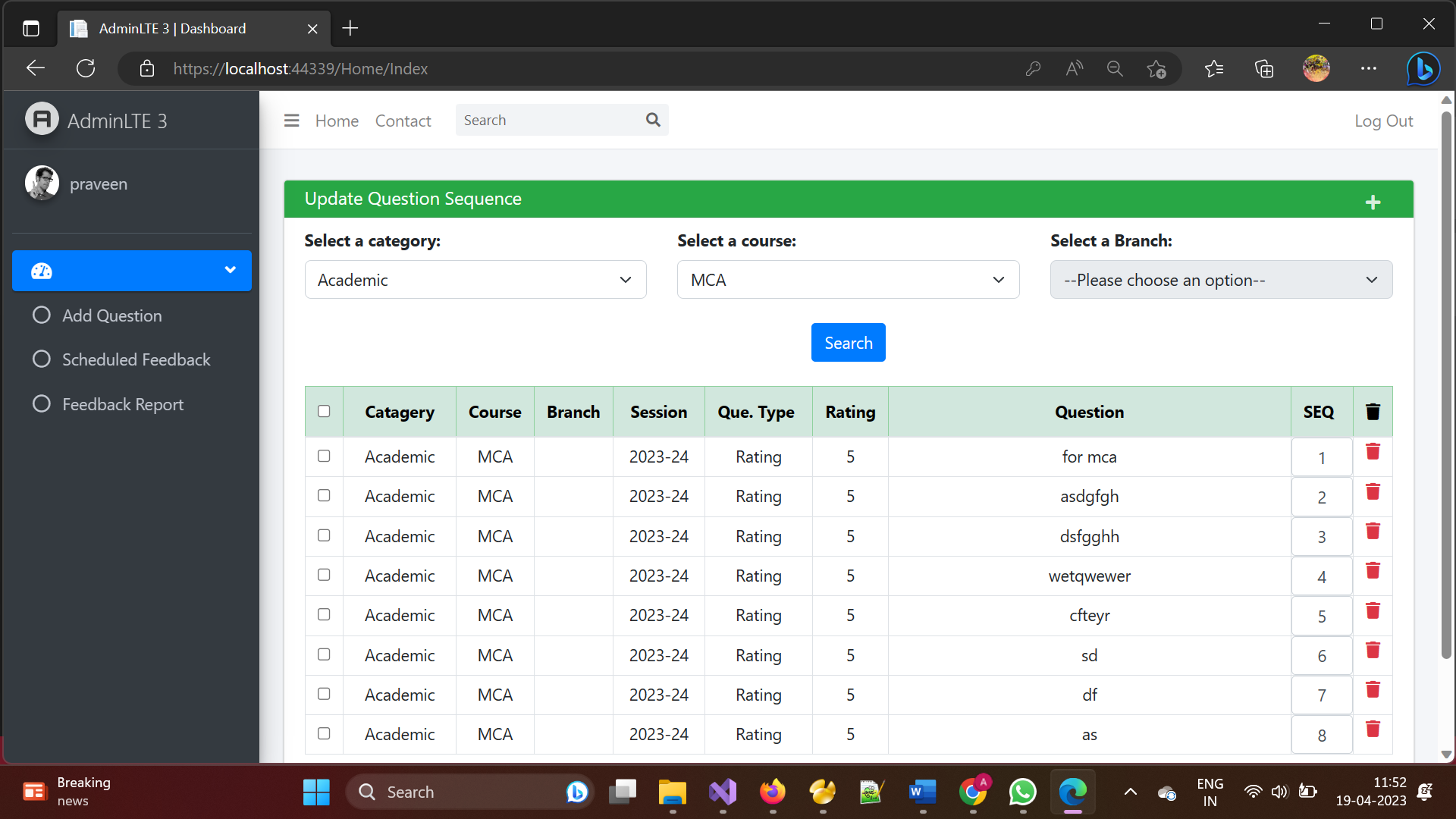


Figure 4.7

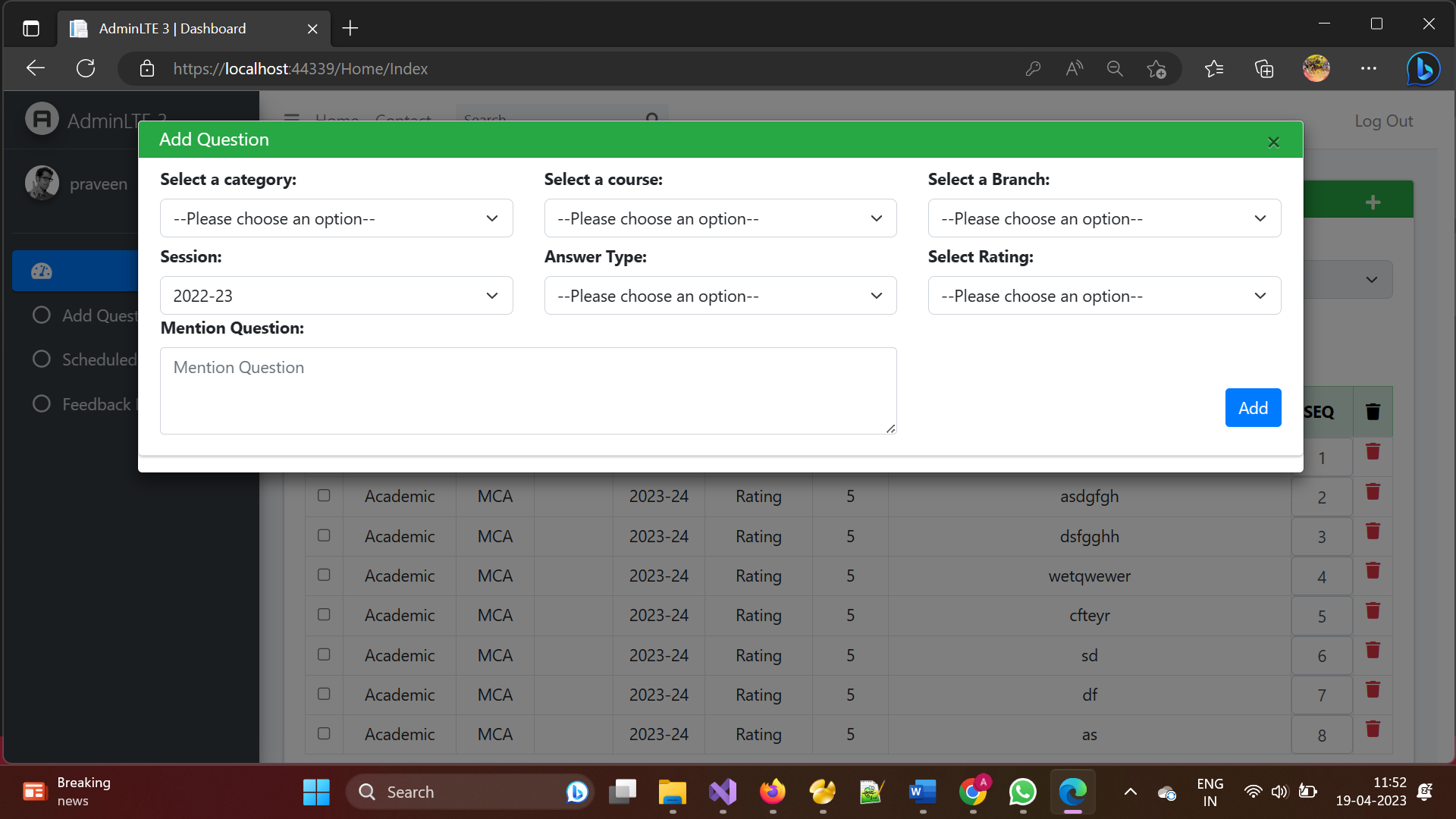


Figure 4.8

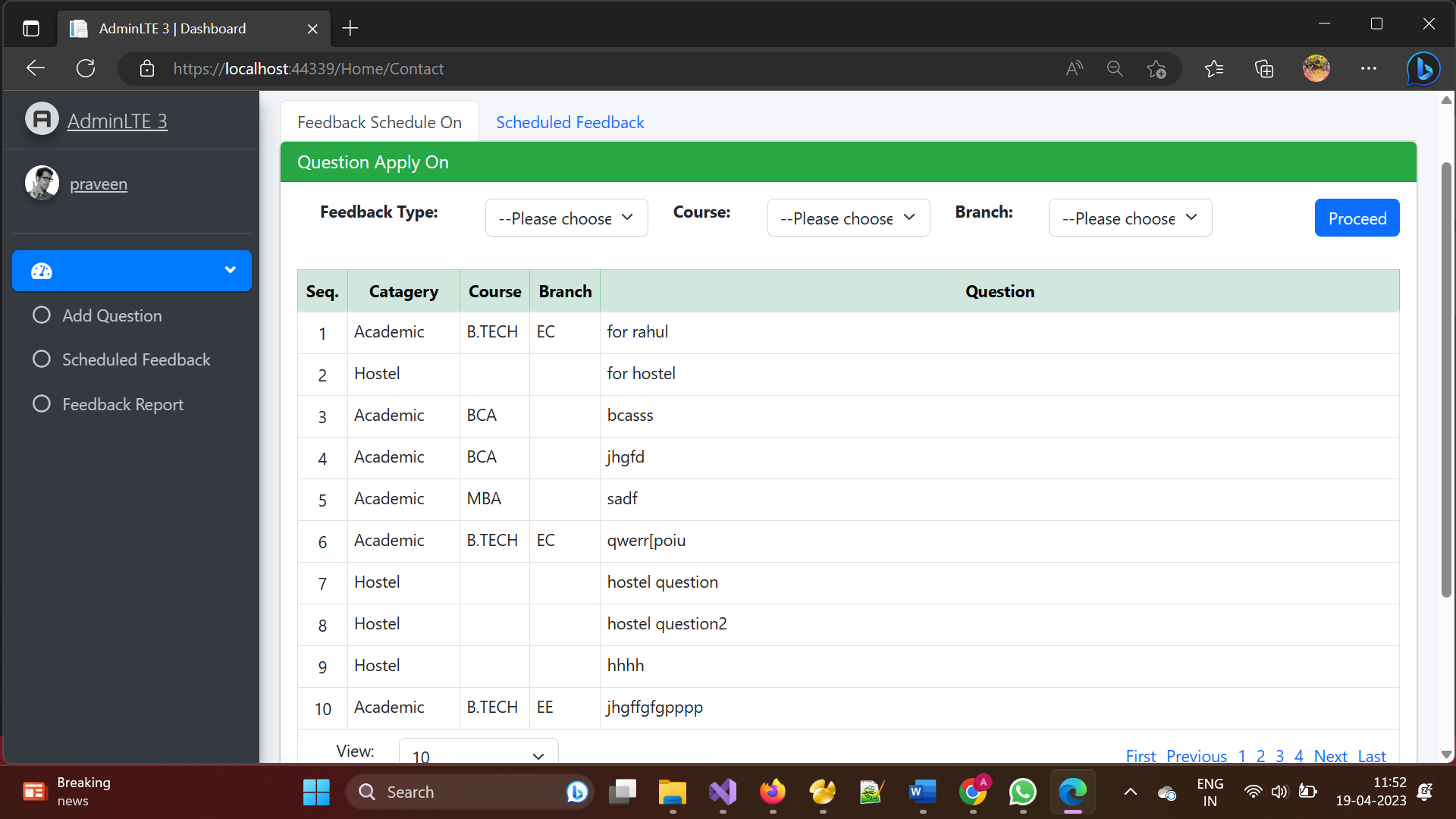


Figure 4.9

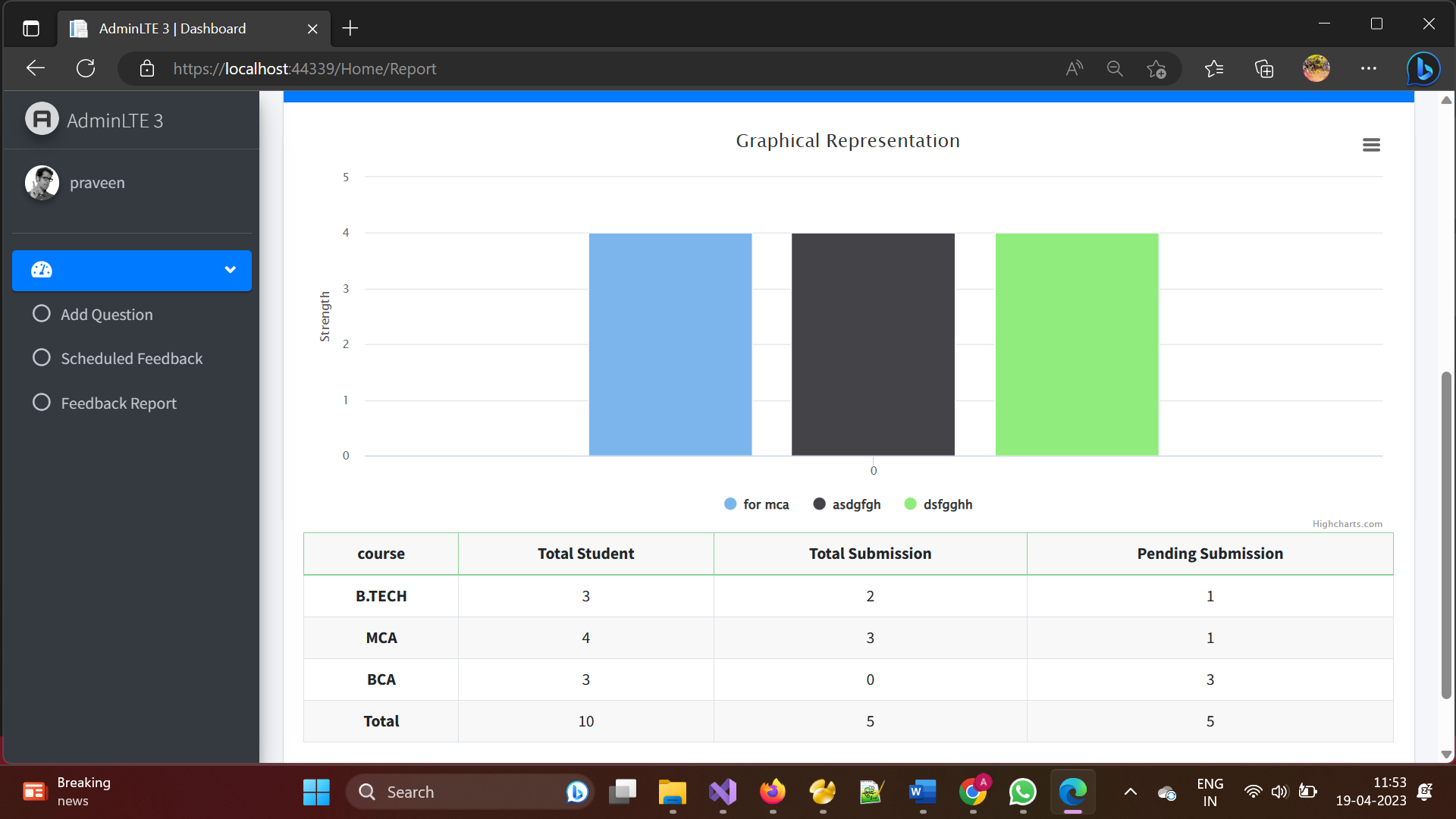


Figure 4.10

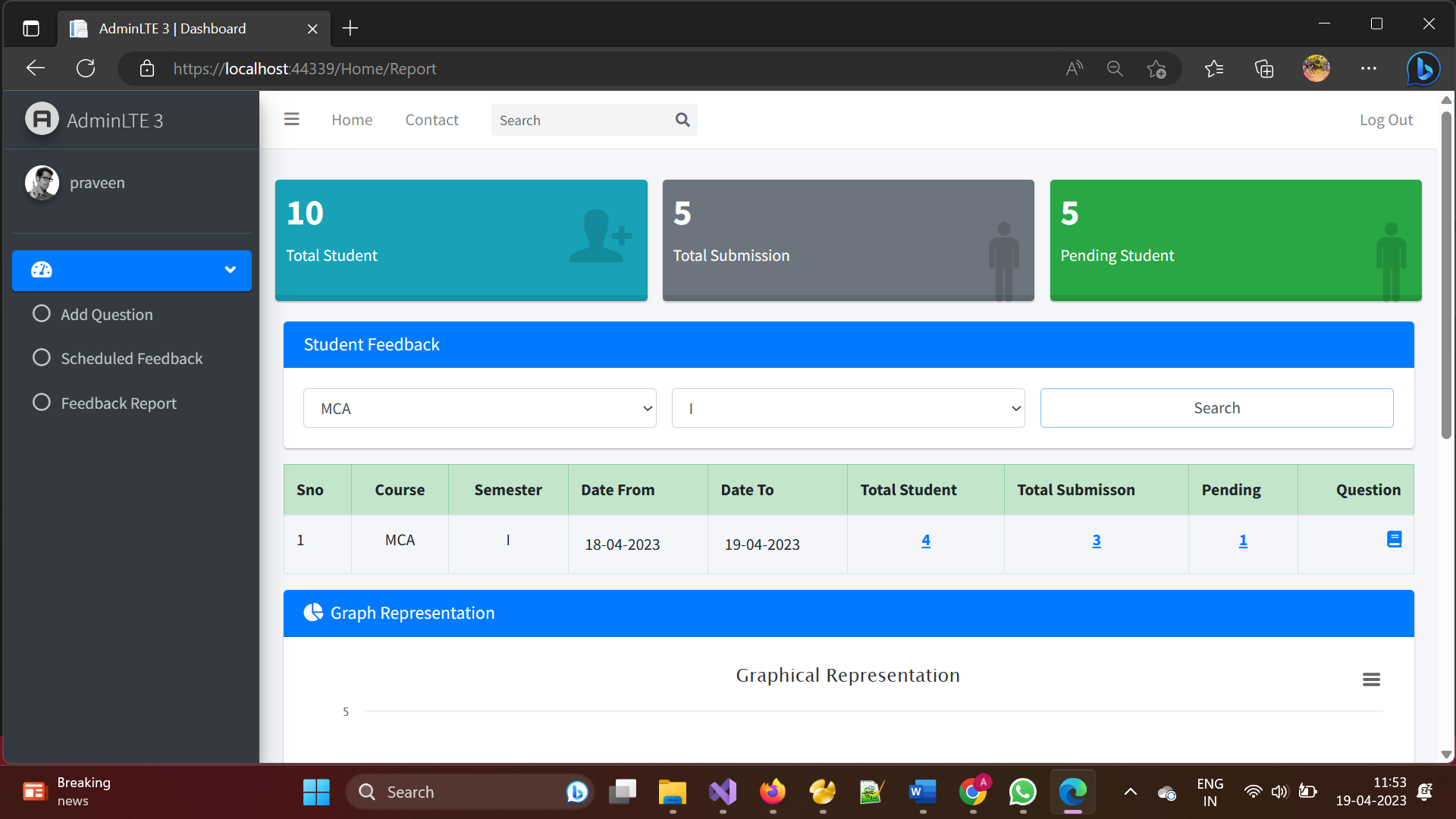


Figure 4.11

**Code:**

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using static feedback\_form.Controllers.HomeController;

using System.Net;

using System.Security.Cryptography;

using Microsoft.Ajax.Utilities;

using System.Collections;

namespace feedback\_form.Controllers

{

public class AbhiController : Controller

{

// GET: Abhi

public ActionResult Index()

{

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetExpires(DateTime.Now.AddSeconds(-1));

Response.Cache.SetNoStore();

Session["RollNo"] = null;

return View();

}

public ActionResult login(User myvar)

{

List<User> list2 = new List<User>();

string query = "select Roll\_No,Password1,Name,type,course,branch,sem from login\_table where Roll\_No='" + myvar.RollNo + "' and Password1='" + myvar.Password1 + "'";

DataTable dt = ExecuteMe.Select(query, "Demo");

if (dt.Rows.Count > 0)

{

ExecuteMe.DeleteInsertUpdate("insert into login\_detail(id) values('" + myvar.RollNo + "')", "Demo");

Session["name"] = dt.Rows[0]["Name"].ToString();

Session["RollNo"] = dt.Rows[0]["Roll\_No"].ToString();

Session["course"] = dt.Rows[0]["course"].ToString();

Session["sem"] = dt.Rows[0]["sem"].ToString();

Session["branch"] = dt.Rows[0]["branch"].ToString();

var pad = dt.Rows[0]["Type"].ToString();

if (pad=="1")

{

//return RedirectToAction("dashbord", "Abhi");

return Content("ok:/Abhi/dashbord");

}

else

{

return Content("ok:/Home/Index");

}

}

else

{

return Content("Sorry! Student Login-Id And Password Mismatched..");

}

}

public ActionResult dashbord()

{

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetExpires(DateTime.Now.AddSeconds(-1));

Response.Cache.SetNoStore();

string rl = (string)Session["RollNo"];

if (rl == null)

{

return RedirectToAction("Index", "Abhi");

}

return View("");

}

public ActionResult Logout()

{

return Content("ok:/Abhi/Index");

}

//login//

[HttpPost]

//fetch record academic//

public ActionResult fetch\_date(mycls myvar)

{

List<User> list0 = new List<User>();

string sem = (string)Session["sem"];

string query4 = "select DATE\_FORMAT(date\_from,'%d-%m-%Y')'date\_from',DATE\_FORMAT(date\_to,'%d-%m-%Y')'date\_to' from feedback\_master where status='Active' and Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and category='" + myvar.category + "' and sem='" + sem + "' and branch='" + Session["branch"] +"' and course='" + Session["course"] +"' ";

DataTable dt9 = ExecuteMe.Select(query4, "Demo");

//test

string query10 = "select ref\_no,date\_from,date\_to from feedback\_master where status='Active' and Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and category='" + myvar.category + "' and sem='" + sem + "' and branch='" + Session["branch"] + "' and course='" + Session["course"] +"' ";

DataTable dt10 = ExecuteMe.Select(query10, "Demo");

if (dt10.Rows.Count > 0)

{

string s1 = "select roll\_no from feedback\_master\_student where ref\_no='" + dt10.Rows[0]["ref\_no"] +"' and roll\_no='" + Session["RollNo"] +"' ";

DataTable s2 = ExecuteMe.Select(s1, "Demo");

if(s2.Rows.Count > 0)

{

for (int i = 0; i < dt9.Rows.Count; i++)

{

list0.Add(new User

{

from1 = dt9.Rows[i]["date\_from"].ToString(),

to1 = dt9.Rows[i]["date\_to"].ToString(),

con1 = "present",

s = "y",

});

}

return Json(new { list0 }, JsonRequestBehavior.AllowGet);

}

}

//test

if (dt9.Rows.Count > 0)

{

for (int i = 0; i < dt9.Rows.Count; i++)

{

list0.Add(new User

{

from1 = dt9.Rows[i]["date\_from"].ToString(),

to1 = dt9.Rows[i]["date\_to"].ToString(),

con1 = "present",

});

}

}

else

{

string query5 = "SELECT DATE\_FORMAT(date\_from,'%d-%m-%Y')'date\_from',DATE\_FORMAT(date\_to,'%d-%m-%Y')'date\_to'\r\nFROM feedback\_master\r\nWHERE course='" + Session["course"] +"' and branch='" + Session["branch"] +"' and category='" + myvar.category + "' AND date\_from BETWEEN DATE(NOW()) AND DATE(DATE\_ADD(NOW(), INTERVAL 1 DAY)) \r\nORDER BY date\_from ASC;";

DataTable dt4 = ExecuteMe.Select(query5, "Demo");

if (dt4.Rows.Count > 0)

{

for (int i = 0; i < dt4.Rows.Count; i++)

{

list0.Add(new User

{

from1 = dt4.Rows[i]["date\_from"].ToString(),

to1 = dt4.Rows[i]["date\_to"].ToString(),

con1 = "comming",

});

}

}

else

{

list0.Add(new User

{

con1 = "close",

});

}

}

return Json(new { list0 }, JsonRequestBehavior.AllowGet);

}

public ActionResult fetch\_dateH(mycls myvar1)

{

List<User> list9 = new List<User>();

string sem = (string)Session["sem"];

string query4 = "select DATE\_FORMAT(date\_from,'%d-%m-%Y')'date\_from',DATE\_FORMAT(date\_to,'%d-%m-%Y')'date\_to' from feedback\_master where status='Active' and Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and category='" + myvar1.category1 + "' ";

DataTable dt4 = ExecuteMe.Select(query4, "Demo");

//test

string query10 = "select ref\_no,date\_from,date\_to from feedback\_master where status='Active' and Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and category='" + myvar1.category1 + "'";

DataTable dt10 = ExecuteMe.Select(query10, "Demo");

if (dt10.Rows.Count > 0)

{

string s1 = "select roll\_no from feedback\_master\_student where ref\_no='" + dt10.Rows[0]["ref\_no"] + "' and roll\_no='" + Session["RollNo"] + "' ";

DataTable s2 = ExecuteMe.Select(s1, "Demo");

if (s2.Rows.Count > 0)

{

for (int i = 0; i < dt4.Rows.Count; i++)

{

list9.Add(new User

{

from1 = dt4.Rows[i]["date\_from"].ToString(),

to1 = dt4.Rows[i]["date\_to"].ToString(),

con3 = "present",

s1 = "y",

});

}

return Json(new { list9 }, JsonRequestBehavior.AllowGet);

}

}

if (dt4.Rows.Count > 0)

{

for (int i = 0; i < dt4.Rows.Count; i++)

{

list9.Add(new User

{

from1 = dt4.Rows[0]["date\_from"].ToString(),

to1 = dt4.Rows[0]["date\_to"].ToString(),

con3 = "present",

});

}

}

else

{

string query5 = "SELECT DATE\_FORMAT(date\_from,'%d-%m-%Y')'date\_from',DATE\_FORMAT(date\_to,'%d-%m-%Y')'date\_to'\r\nFROM feedback\_master\r\nWHERE date\_from BETWEEN DATE(NOW()) AND DATE(DATE\_ADD(NOW(), INTERVAL 1 DAY)) \r\nORDER BY date\_from ASC;";

dt4 = ExecuteMe.Select(query5, "Demo");

if (dt4.Rows.Count > 0)

{

for (int i = 0; i < dt4.Rows.Count; i++)

{

list9.Add(new User

{

from1 = dt4.Rows[0]["date\_from"].ToString(),

to1 = dt4.Rows[0]["date\_to"].ToString(),

con3 = "comming",

});

}

}

else

{

list9.Add(new User

{

con3 = "close",

});

}

}

return Json(new { list9 }, JsonRequestBehavior.AllowGet);

}

public ActionResult myfun(mycls myvar)

{

string str;

str = (string)Session["RollNo"];

List<demo> list2 = new List<demo>();

string cou = (string)Session["course"];

string sem = (string)Session["sem"];

string myquery2 = "SELECT course,sem FROM `feedback\_master` where Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and Status = 'Active' and category='" + myvar.category + "' and sem='" + sem + "' ";

DataTable dt2 = ExecuteMe.Select(myquery2, "Demo");

if (dt2.Rows.Count > 0)

{//test

//test

string c = dt2.Rows[0]["course"].ToString();

if (c == "All")

{

string myquery3 = "SELECT ref\_no,q\_id FROM `feedback\_master` where Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and Status = 'Active' and category='" + myvar.category + "' and sem='" + sem + "' ";

dt2 = ExecuteMe.Select(myquery3, "Demo");

if (dt2.Rows.Count > 0)

{

Session["ref\_no"] = dt2.Rows[0]["ref\_no"].ToString();

Session["q\_id"] = dt2.Rows[0]["q\_id"].ToString();

string ids = (string)Session["q\_id"];

string[] question\_list = ids.Split(',');

foreach (string lst in question\_list)

{

string a = lst;

string myquery1 = "SELECT id,question FROM `insert\_question` where id='" + a + "' ";

DataTable dt1 = ExecuteMe.Select(myquery1, "Demo");

list2.Add(new demo

{

question = dt1.Rows[0]["question"].ToString(),

id = dt1.Rows[0]["id"].ToString(),

});

}

}

}

else

{

string myquery = "SELECT ref\_no,q\_id,course FROM `feedback\_master` where Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and Status = 'Active' and category='" + myvar.category + "' and course='" + cou + "' and sem='" + sem + "' ";

DataTable dt = ExecuteMe.Select(myquery, "Demo");

if (dt.Rows.Count > 0)

{

Session["ref\_no"] = dt.Rows[0]["ref\_no"].ToString();

Session["q\_id"] = dt.Rows[0]["q\_id"].ToString();

string ids = (string)Session["q\_id"];

string[] question\_list = ids.Split(',');

foreach (string lst in question\_list)

{

string a = lst;

string myquery1 = "SELECT id,question FROM `insert\_question` where id='" + a + "' ";

DataTable dt1 = ExecuteMe.Select(myquery1, "Demo");

list2.Add(new demo

{

question = dt1.Rows[0]["question"].ToString(),

id = dt1.Rows[0]["id"].ToString(),

});

}

}

}

}

return Json(new { list2 }, JsonRequestBehavior.AllowGet);

}

//fatch record hostel//

public ActionResult myfun2(mycls myvar1)

{

string str;

str = (string)Session["RollNo"];

string check = "select session22\_23H from login\_table where roll\_no='" + str + "' and session22\_23H='1' ";

DataTable dt0 = ExecuteMe.Select(check, "Demo");

if (dt0.Rows.Count > 0)

{

return Content("already");

}

List<demo> list3 = new List<demo>();

string myquery = "SELECT ref\_no,q\_id FROM `feedback\_master` where Date(date\_from)<=dATE(NOW()) and DATE(NOW()) <= DATE(date\_to) and Status = 'Active' and category='" + myvar1.category1 + "' ";

DataTable dt = ExecuteMe.Select(myquery, "Demo");

if (dt.Rows.Count > 0)

{

Session["ref\_no1"] = dt.Rows[0]["ref\_no"].ToString();

Session["q\_id1"] = dt.Rows[0]["q\_id"].ToString();

string ids = (string)Session["q\_id1"];

string[] question\_list = ids.Split(',');

foreach (string lst in question\_list)

{

string a = lst;

string myquery1 = "SELECT id,question FROM `insert\_question` where id='" + a + "' ";

DataTable dt1 = ExecuteMe.Select(myquery1, "Demo");

list3.Add(new demo

{

question = dt1.Rows[0]["question"].ToString(),

id = dt1.Rows[0]["id"].ToString(),

});

}

}

return Json(new { list3 }, JsonRequestBehavior.AllowGet);

}

//insert record academic table//

//[HttpPost]

public ActionResult myfun1(List<mycls> user)

{

string str;

str = (string)Session["RollNo"];

string str1;

str1 = (string)Session["ref\_no"];

string str2;

str2 = (string)Session["name"];

string hostName = Dns.GetHostName();

string myIP = Dns.GetHostByName(hostName).AddressList[0].ToString();

string tod = DateTime.Now.ToString("yyyy-MM-dd");

ExecuteMe.DeleteInsertUpdate("update login\_table set sesson22\_23A='1' where Roll\_no='" + str + "' ", "Demo");

foreach (mycls item in user)

{

ExecuteMe.DeleteInsertUpdate("update login\_table set submitted\_date='" + tod + "',ref\_no='"+str1+"' where roll\_no='" + str + "'", "Demo");

ExecuteMe.DeleteInsertUpdate("insert INTO feedback\_master\_student(ref\_no,Roll\_no,rating,Comment,submitted\_date,submitted\_by,submitted\_from,q\_id) values('" + str1 + "','" + str + "','" + item.rating + "','" + item.comment + "' ,'" + tod + "' ,'" + str2 + "' ,'" + myIP + "','"+item.q\_id+"' )", "Demo");

}

return Content("feedback submitted successfully");

}

//insert record hostel table//

[HttpPost]

public ActionResult myfun3(List<mycls> user)

{

string str;

str = (string)Session["RollNo"];

string str1;

str1 = (string)Session["ref\_no1"];

string str2;

str2 = (string)Session["name"];

string hostName = Dns.GetHostName();

string myIP = Dns.GetHostByName(hostName).AddressList[0].ToString();

string tod = DateTime.Now.ToString("yyyy-MM-dd");

ExecuteMe.DeleteInsertUpdate("update login\_table set session22\_23H='1' where Roll\_no='" + str + "' ", "Demo");

foreach (mycls item in user)

{

ExecuteMe.DeleteInsertUpdate("insert INTO feedback\_master\_student(ref\_no,Roll\_no,rating,Comment,submitted\_date,submitted\_by,submitted\_from) values('" + str1 + "','" + str + "','" + item.rating + "','" + item.comment + "' ,'" + tod + "' ,'" + str2 + "' ,'" + myIP + "' )", "Demo");

}

return Content("feedback submitted successfully ");

}

}

}

public class demo

{

public string s { get; set; }

public string from1 { get; set; }

public string id { get; set; }

public string category { get; set; }

public string category1 { get; set; }

public string Sno { get; set; }

public string Question1 { get; set; }

public string Comment { get; set; }

public string Rating { get; set; }

public string q\_type { get; set; }

public string question { get; set; }

public string to1 { get; set; }

public string rating { get; set; }

}

public class mycls

{

public string s { get; set; }

public string category { get; set; }

public string from1 { get; set; }

public string comment { get; set; }

public string id { get; set; }

public string rating { get; set; }

public string category1 { get; set; }

public string RollNo { get; set; }

public string to1 { get; set; }

public string q\_id { get; set; }

}

public class User

{

public string s { get; set; }

public string s1 { get; set; }

public string to1 { get; set; }

public string from1 { get; set; }

public int Id { get; set; }

public string RollNo { get; set; }

public string Password1 { get; set; }

public string con1 { get; set; }

public string con2 { get; set; }

public string con3 { get; set; }

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace feedback\_form.Controllers

{

public class Home Controller : Controller

{

private object dt;

public int Total\_submission { get; private set; }

public int Total\_stu { get; private set; }

public ActionResult login form(User user)

{

return View();

}

public ActionResult fetchall()

{

List<User> list1 = new List<User>();

string myquery = "SELECT id,category,course,branch,session,q\_type,rating,question,display\_by FROM `insert\_question` order by display\_by asc";

DataTable dt = ExecuteMe.Select(myquery, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

list1.Add(new User

{

id = dt.Rows[i]["id"].ToString(),

category = dt.Rows[i]["category"].ToString(),

hostelorcourse = dt.Rows[i]["course"].ToString(),

branch = dt.Rows[i]["branch"].ToString(),

session = dt.Rows[i]["session"].ToString(),

q\_type = dt.Rows[i]["q\_type"].ToString(),

rating = dt.Rows[i]["rating"].ToString(),

question = dt.Rows[i]["question"].ToString(),

display\_by = dt.Rows[i]["display\_by"].ToString(),

//dob = dt.Rows[i]["dob"].GetType().t

});

}

return Json(new { list1 }, JsonRequestBehavior.AllowGet);

}

public ActionResult Index()

{

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetExpires(DateTime.Now.AddSeconds(-1));

Response.Cache.SetNoStore();

string rl = (string)Session["RollNo"];

if (rl==null)

{

return RedirectToAction("Index","Abhi");

}

return View();

}

public ActionResult Testing()

{

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult try1()

{

ExecuteMe.DeleteInsertUpdate("insert into demo (name) values('hii')","Demo");

return View();

}

public ActionResult Contact()

{

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetExpires(DateTime.Now.AddSeconds(-1));

Response.Cache.SetNoStore();

string rl = (string)Session["RollNo"];

if (rl == null)

{

return RedirectToAction("Index", "Abhi");

}

return View();

}

[HttpPost]

public ActionResult qbysem(User user)

{

var date1 = user.s\_t.ToString("yyyy-MM-dd");

var date2 = user.e\_d.ToString("yyyy-MM-dd");

ExecuteMe.DeleteInsertUpdate("insert into feedback\_master (category,course,sem,q\_id,branch,date\_from,date\_to) values ('"+user.category+"','"+user.course+"','"+user.sem+"','"+user.q\_id+"','"+user.branch+"','"+date1+"','"+date2+"')", "Demo");

return Content("Question apply on successfully");

}

public ActionResult search1(User user)

{

List<User> list1 = new List<User>();

string myquery = "SELECT id,category,course,branch,session,q\_type,rating,question,display\_by FROM `insert\_question` where category='" + user.category + "' and course='" + user.course + "' and branch='" + user.branch + "' and status='Active' order by display\_by asc";

DataTable dt = ExecuteMe.Select(myquery, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

list1.Add(new User

{

id = dt.Rows[i]["id"].ToString(),

category = dt.Rows[i]["category"].ToString(),

hostelorcourse = dt.Rows[i]["course"].ToString(),

branch = dt.Rows[i]["branch"].ToString(),

session = dt.Rows[i]["session"].ToString(),

q\_type = dt.Rows[i]["q\_type"].ToString(),

rating = dt.Rows[i]["rating"].ToString(),

question = dt.Rows[i]["question"].ToString(),

display\_by = dt.Rows[i]["display\_by"].ToString(),

});

}

return Json(new { list1 }, JsonRequestBehavior.AllowGet);

}

public ActionResult Search(User user)

{

List<User> list1 = new List<User>();

string myquery = "SELECT id,category,course,branch,session,q\_type,rating,question,display\_by FROM `insert\_question` where category='"+user.category + "' and course='"+user.course+"' and branch='"+user.branch+"' and status='Active' order by display\_by asc";

DataTable dt = ExecuteMe.Select(myquery, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

list1.Add(new User

{

id = dt.Rows[i]["id"].ToString(),

category = dt.Rows[i]["category"].ToString(),

hostelorcourse = dt.Rows[i]["course"].ToString(),

branch = dt.Rows[i]["branch"].ToString(),

session = dt.Rows[i]["session"].ToString(),

q\_type = dt.Rows[i]["q\_type"].ToString(),

rating = dt.Rows[i]["rating"].ToString(),

question = dt.Rows[i]["question"].ToString(),

display\_by = dt.Rows[i]["display\_by"].ToString(),

});

}

return Json(new { list1 }, JsonRequestBehavior.AllowGet);

}

public ActionResult update(List<User> user)

{

foreach (User user1 in user)

{

ExecuteMe.DeleteInsertUpdate("update insert\_question set display\_by='" + user1.display\_by + "' where id='" + user1.id + "'", "Demo");

}

return Content("Update Question Sequence successfully");

}

public ActionResult delete(User user)

{

ExecuteMe.DeleteInsertUpdate("update insert\_question set status='In-Active' where id='" + user.id + "'", "Demo");

return Content("Delete Question successfully");

}

public ActionResult UpdateFun(User user)

{

ExecuteMe.DeleteInsertUpdate("update insert\_question set display\_by='" + user.display\_by + "' where id='"+user.id+"'" , "Demo");

return Content("Update Question Sequence successfully");

}

public ActionResult insertQuestion(List<User> user)

{

foreach (User user1 in user)

{

ExecuteMe.DeleteInsertUpdate("insert into insert\_question(category,course,branch,session,q\_type,rating,question) values('" + user1.category + "','"+user1.hostelorcourse + "','"+user1.branch+"','"+user1.session+"','" + user1.q\_type + "','" + user1.rating + "','" + user1.question + "')", "Demo");

}

return Content("insert Question Successfully");

}

public ActionResult Report()

{

return View();

}

public ActionResult FetchRecordctrl2(demo user)

{

List<demo> list4 = new List<demo>();

//string myquery2 = "SELECT id, stu\_name as sub\_name, course as sub\_course, count(if (submitted = 0, submitted, NULL)) as 'Remain\_stu' FROM `` where course = '" + user.category + "' and Semester='" + user.Semester + "' and submitted='0' GROUP BY id ";

//string myquery2 = "SELECT course, COUNT(\*) AS 'Total\_stu',count(if(sesson22\_23A >0, sesson22\_23A, NULL)) as 'Total\_submission', count(if(sesson22\_23A =0, sesson22\_23A, NULL)) as 'Pending' FROM `login\_table` where type = '1' group by course ";

//string myquery = "select submitted, id, stu\_name,category, course,branch,question,rating,comment, Semester,DATE\_FORMAT(datefrom,'%d.%m.%Y')'datefrom',DATE\_FORMAT(dateto,'%d.%m.%Y')'dateto', sub\_date FROM `data\_analysis` where course='" + user.category + "' and Semester='" + user.Semester + "'";

string myquery2 = "SELECT course,COUNT(sesson22\_23A) AS 'Total\_stu', sum(sesson22\_23A) as 'Total\_submission' , COUNT(sesson22\_23A) - SUM(sesson22\_23A) AS 'Pending' FROM `login\_table` where type = '1' group by course union Select 'Total' , count(sesson22\_23A) , sum(sesson22\_23A) , COUNT(sesson22\_23A) - SUM(sesson22\_23A) from login\_table where type = '1'";

DataTable dt = ExecuteMe.Select(myquery2, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

list4.Add(new demo

{

//Total\_stu = dt.Rows[i]["Total\_stu"].ToString(),

//Total\_submission = dt.Rows[i]["Total\_submission"].ToString(),

//id = (int)dt.Rows[i]["id"],

course = dt.Rows[i]["course"].ToString(),

Total\_stu = dt.Rows[i]["Total\_stu"].ToString(),

Total\_submission = dt.Rows[i]["Total\_submission"].ToString(),

Pending = dt.Rows[i]["Pending"].ToString(),

});

}

return Json(new { list4 }, JsonRequestBehavior.AllowGet);

}

[HttpPost]

public ActionResult FetchRecordctrl(demo user)

{

List<demo> list2 = new List<demo>();

string que = "SELECT ref\_no,q\_id,DATE\_FORMAT(date\_from,'%d-%m-%Y')'date\_from',DATE\_FORMAT(date\_to,'%d-%m-%Y')'date\_to' FROM `feedback\_master` where course ='" + user.category + "' and Sem = '" + user.Semester + "' and status = 'Active'";

DataTable re = ExecuteMe.Select(que, "Demo");

if (re.Rows.Count > 0)

{

int j = re.Rows.Count - 1;

Session["ref\_no"] = re.Rows[j]["ref\_no"].ToString();

Session["q\_id"] = re.Rows[j]["q\_id"].ToString();

}

string rf = (string)Session["ref\_no"];

string qid = (string)Session["q\_id"];

string cou\_student = "select count(roll\_no) as roll\_no ,course,sem from login\_table where course ='" + user.category + "' and Sem = '" + user.Semester + "'";

DataTable re1 = ExecuteMe.Select(cou\_student, "Demo");

if (re1.Rows.Count > 0)

{

Session["total\_student"] = re1.Rows[0]["roll\_no"].ToString();

}

string t\_s = (string)Session["total\_student"];

int n = 0;

// string co\_student = "select count(if(sesson22\_23A >0, sesson22\_23A, NULL)) as 'ref\_no' from login\_table ";

string co\_student = "select COUNT(DISTINCT roll\_no) as submit from feedback\_master\_student where Ref\_no = '" + rf + "' group by roll\_no";

DataTable re2 = ExecuteMe.Select(co\_student, "Demo");

if (re2.Rows.Count > 0)

{

for (int i = 0; i < re2.Rows.Count; i++)

{

n++;

}

Session["submit\_student"] = re2.Rows[0]["submit"].ToString();

}

// string s\_s = (string)Session["submit\_student"];

int pending = Convert.ToInt32(t\_s) - n;

string r\_n = (string)Session["ref\_no1"];

//string myquery1 = "SELECT COUNT(\*) AS 'Total\_stu',count(if(sesson22\_23A >0, sesson22\_23A, NULL)) as 'Total\_submission', count(if(sesson22\_23A =0, sesson22\_23A, NULL)) as 'Remain\_stu',course,sem FROM `login\_table` where course = '" + user.category + "' and Sem= '" + user.Semester + "' ";

// DataTable dt1 = MyConnection.Select(myquery1, "Demo1");

//string myquery = "select submitted, id, stu\_name, category, course,branch,question,rating,comment, Semester,DATE\_FORMAT(datefrom,'%d.%m.%Y')'datefrom',DATE\_FORMAT(dateto,'%d.%m.%Y')'dateto', sub\_date FROM `login\_table` where course='" + user.category + "' and Sem='" + user.Semester + "'";

if (re1.Rows.Count > 0 && re.Rows.Count > 0)

{

list2.Add(new demo

{

datefrom = re.Rows[0]["date\_from"].ToString(),

dateto = re.Rows[0]["date\_to"].ToString(),

Total\_stu = re1.Rows[0]["roll\_no"].ToString(),

// Total\_submission= re2.Rows[0]["ref\_no"].ToString(),

Total\_submission = n.ToString(),

Remain\_stu = pending.ToString(),

course = re1.Rows[0]["course"].ToString(),

Semester = re1.Rows[0]["sem"].ToString(),

});

return Json(new { list2 }, JsonRequestBehavior.AllowGet);

}

else

{

return Content("no record found");

}

}

public ActionResult fetch\_q()

{

List<demo> list3 = new List<demo>();

string ids = (string)Session["q\_id"];

string[] question\_list = ids.Split(',');

foreach (string lst in question\_list)

{

string a = lst;

string myquery1 = "SELECT id,question FROM `insert\_question` where id='" + a + "' ";

string myquery2 = "select CAST(AVG(rating)AS DECIMAL(10,1)) as rating FROM `feedback\_master\_student` where q\_id ='" + a + "'";

DataTable dt1 = ExecuteMe.Select(myquery1, "Demo");

DataTable dt2 = ExecuteMe.Select(myquery2, "Demo");

for (int i = 0; i < dt1.Rows.Count; i++)

{

list3.Add(new demo

{

question = dt1.Rows[i]["question"].ToString(),

id = dt1.Rows[i]["id"].ToString(),

rating = dt2.Rows[i]["rating"].ToString(),

});

}

}

return Json(new { list3 }, JsonRequestBehavior.AllowGet);

}

public ActionResult fetchtotal(demo user)

{

List<demo> lis = new List<demo>();

string query1 = "select name,course,roll\_no from login\_table where course ='" + user.category + "' and Sem = '" + user.Semester + "'";

DataTable dt = ExecuteMe.Select(query1, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

lis.Add(new demo

{

Name = dt.Rows[i]["name"].ToString(),

Roll\_No = dt.Rows[i]["roll\_no"].ToString() ,

course = dt.Rows[i]["course"].ToString(),

//sub\_date = dt.Rows[0]["sub\_date"].ToString(),

});

}

return Json(new { lis }, JsonRequestBehavior.AllowGet);

// return View();

}

public ActionResult FetchRecordctrl1(demo user)

{

List<demo> list1 = new List<demo>();

string rf = (string)Session["ref\_no"];

string myquery2 = "SELECT course as sub\_course, login\_table.`Roll\_No` as rollnot, login\_table.`Name` as namenot from login\_table where Type='1' and course='" + user.category + "' and sem='" + user.Semester + "' and login\_table.Roll\_No not IN(SELECT feedback\_master\_student.Roll\_no from feedback\_master\_student where feedback\_master\_student.Ref\_no='" + rf + "') ";

DataTable dt = ExecuteMe.Select(myquery2, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

list1.Add(new demo

{

sub\_course = dt.Rows[i]["sub\_course"].ToString(),

rollnot = dt.Rows[i]["rollnot"].ToString(),

namenot = dt.Rows[i]["namenot"].ToString(),

//sub\_date = dt.Rows[0]["sub\_date"].ToString(),

});

}

return Json(new { list1 }, JsonRequestBehavior.AllowGet);

}

public ActionResult submitstudent(demo user)

{

List<demo> li = new List<demo>();

string rf = (string)Session["ref\_no"];

string myquery2 = "SELECT course as sub\_course, login\_table.`Roll\_No` as roll, login\_table.`Name` as namenot,DATE\_FORMAT(submitted\_date,'%d-%m-%Y')'submitted\_date' from login\_table where Type='1' and course='" + user.category + "' and sem='" + user.Semester + "' and login\_table.Roll\_No IN(SELECT feedback\_master\_student.Roll\_no from feedback\_master\_student where feedback\_master\_student.Ref\_no='" + rf + "') ";

DataTable dt = ExecuteMe.Select(myquery2, "Demo");

for (int i = 0; i < dt.Rows.Count; i++)

{

li.Add(new demo

{

course = dt.Rows[i]["sub\_course"].ToString(),

Roll\_No = dt.Rows[i]["roll"].ToString(),

Name = dt.Rows[i]["namenot"].ToString(),

submitted = dt.Rows[i]["submitted\_date"].ToString(),

//sub\_date = dt.Rows[0]["sub\_date"].ToString(),

});

}

return Json(new { li }, JsonRequestBehavior.AllowGet);

// return View();

}

public class demo

{

public string sub\_course1 { get; set; }

public string rollnot { get; set; }

public string namenot { get; set; }

public string Pending { get; set; }

public string Name { get; set; }

public string sub\_course { get; set; }

public string Roll\_No { get; set; }

public string sem { get; set; }

public string id { get; set; }

public string Total\_stu { get; set; }

public string Total\_submission { get; set; }

public string Remain\_stu { get; set; }

public string category { get; set; }

public string course { get; set; }

public string branch { get; set; }

public string rating { get; set; }

public string question { get; set; }

public string comment { get; set; }

// public string semester { get; set; }

public string submitted { get; set; }

public string datefrom { get; set; }

public string dateto { get; set; }

public string Semester { get; set; }

public string sub\_name1 { get; set; }

public string sesson22\_23A { get; set; }

}

public class User

{

public string id

{

get; set;

}

public string course

{

get;

set;

}

public string category { get; set; }

public string hostelorcourse { get; set; }

public string q\_type { get; set; }

public string branch { get; set; }

public string question { get; set; }

public string session { get; set; }

public string rating { get; set; }

public string display\_by { get; set; }

public string hostel { get; set;}

public string sem { get; set; }

public string q\_id { get; set; }

public DateTime s\_t { get; set; }

public DateTime e\_d { get; set; }

}

}

}

**CHAPTER 5: Software Testing**

**The software testing section of the report should include the following information:**

1. Introduction: Briefly explain the purpose of software testing in the project.
2. Testing methodology: Describe the testing methodology used in the project, including the types of tests performed (e.g., unit testing, integration testing, acceptance testing).
3. Test plan: Explain the test plan developed for the project, including the test cases, test scripts, and test data used.
4. Testing results: Provide an overview of the testing results, including the number of tests performed, the number of tests passed and failed, and the severity of the defects found.
5. Defect management: Describe how defects were managed during the testing process, including how defects were identified, reported, tracked, and resolved.
6. Lessons learned: Discuss any lessons learned from the testing process, including what worked well and what could be improved in future projects.

Here is an example of how you can write about software testing in a project feedback management system report:

**Software Testing:** Software testing is an essential part of the project development lifecycle. The purpose of software testing in the project is to ensure that the feedback management system is functioning as expected, is reliable, and meets the user requirements. The testing methodology used in the project included unit testing, integration testing, and acceptance testing. The test plan developed for the project included test cases, test scripts, and test data to be used during the testing process.

The testing results indicated that the feedback management system was functioning as expected and met the user requirements. A total of 150 test cases were executed, of which 140 were passed, and 10 were failed. The severity of the defects found during testing was low to medium. Defects were identified using a defect tracking system, reported to the development team, and tracked until they were resolved.

During the testing process, the development team learned several lessons that could be applied in future projects. For example, the importance of developing a comprehensive test plan, the need to perform testing in different environments, and the importance of ensuring that the system is compatible with different web browsers. Overall, the software testing process was successful in ensuring the quality of

**White box testing:** White box testing is a type of testing where the tester has knowledge of the internal workings of the system being tested. This means that the tester has access to the source code and understands how the code is implemented. The goal of white box testing is to ensure that the code is working correctly and that all paths through the code are being executed.

**Black box testing**: Black box testing on the other hand, is a type of testing where the tester has no knowledge of the internal workings of the system being tested. The tester is only concerned with the system's inputs and outputs and whether they are working as expected. The goal of black box testing is to ensure that the system is behaving correctly according to its specifications, without regard to how the system is implemented.

**System testing:** System testingis a type of testing where the entire system is tested as a whole. This means that all components of the system are tested together to ensure that they are working correctly and that they are interacting with each other as expected. The goal of system testing is to ensure that the system is meeting its requirements and is functioning correctly in its intended environment.

# 

# 

**CHAPTER 6: Conclusion**

The project feedback management system has been implemented successfully, and the results of this report have demonstrated its effectiveness in improving student engagement and satisfaction. The system has enabled the college to collect and analyze student feedback more efficiently, and has provided valuable insights into areas for improvement. However, the implementation of the system was not without its challenges. Some students were initially resistant to using the system, and there were some technical issues that needed to be resolved. Despite these challenges, the feedback management system has had a significant impact on the quality of education provided by the college. It has facilitated better communication between students and faculty, and has enabled the college to make data-driven decisions that benefit the entire student body. Moving forward, we recommend that the college continue to invest in the development of the feedback management system, with a focus on improving the user experience and addressing any technical issues. By doing so, the college can ensure that the system remains an effective tool for enhancing student engagement and satisfaction.

**REFERENCES**

1. [https://www.stackoverflow.com](https://stackoverflow.com)
2. [https://codepen.io](https://codepen.io/)
3. [https://www.c-sharpcorner.com](https://www.c-sharpcorner.com/)
4. <https://openai.com/blog/chatgpt>
5. [https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide](https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/)