Fall 2022

Registration (Management) System for an Academic Conference

Imaduddin Mohammed
Governors State University

Registration(management) system for an Academic Conference

By

Mohammed Imaduddin

Bachelors in computer science engineering, Shadan College of Engineering,2020

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science

Governors State University
University Park, IL 60484

2022

ABSTRACT

In every college or university, people will conduct different academic conferences so that students will know what is going on in the world related to the latest technological updates. The conference information should be distributed globally to reach a larger number of people. For many people, physical communication is insufficient, and it won't help people clear their doubts. We propose an online registration management system for the conduct of academic conferences in any college or university to overcome all of the limitations of a physical system in conducting the conference.

This new web application will reach every nook and cranny of the globe. People can view the information, they can register with this application, can communicate with the conference people, can get suggestions or recommendations from experts in the same field, etc. To achieve these things, the user needs to sign up with the application, which he should be logged into for doing the different operations in the application. If any of the users want to change their personal information, they will be given the edit option, and they can search for people who are experts in the related field. If needed,they can communicate with them by using this application. Based on the information provided by theuser, experts will recommend the related research field so that he or she can opt for it.

To create the above application, we will use.Net to create the GUI and implement the application's various operations. We will be using HTML, CSS, BOOTSTRAP, JavaScript, and jQuery in coding, andto store the data, we will be using SQL Server management studio. The framework we are going to use isASP.Net Framework 4.5. According to the schedule, the project will be completed by December 10, 2022.

Table of Content

Contents

1. Project Description

This web application development is used by different users. Every user will be having their own operations. All these functionalities need to be implemented one after the other so that the user can be satisfied.

Following are the different types of users.

a. Administrator.
b. Scholar.
c. Experts
d. Conference handler

a. Administrator: He is the privileged user of the application and will monitor all the user's activities. All the information that is being provided by the scholars and experts can be viewedby the administrator. He has the right to delete any of the information from the application. With the help of the application, he can contact any of the users, block them, etc.

b. Scholar: Scholar is one of the users who will be registering with the application, and he will know the information about the conferences wherever they are happening throughout the globe. Scholars can view the conference's details, such as all the dates, contact information for the conference, the schedule, and so on. They can post any of the queries on the site, where experts can view them and provide suggestions. Scholars can get one-stop information about all the conferences in the world. [4]

c. Experts: These people can register with the application if they want to give any kind of suggestion or recommendation, and then they can reply to the queries asked by the scholars. They can recommend the conferences that will be suitable for them, etc.

d. Conference handler: They need to register with the application; there will be a template provided for preparing the conference site; all this information will be posted by the conference handler. This link will be given to the scholars to view the conference details. If the scholar clicks the links,then this static site or template should open and provide the conference details. [4].

1.1 Competitive Information

This application is designed and developed for the registration of academic conferences. As managing all the data physically, an automated system is needed. There is an application known as an easy chair which is having similar functionalities. But the easy chair application is having more functionalities when compared to our application. Our application is focusing only on the registration system. That is the reason, this application is not a competition for any other similar application.

1.2 Relationship to Other Applications/Projects

This project is not related to any other application; it is an individual application.

1.3 Assumptions and Dependencies

- As there are different users like administrators, scholars, experts, and conference handlers, every user will be depending on one or the other. For example, the conference handler will be posting information regarding the conference, which will be visible to all the scholars. They can contact and obtain the information if they see this information.

1.4 Future Enhancements

We hope to make this application a product in the future so that people can integrate it into their applications. For those who want to conduct the conferences for them, we want to make some changes in the existing application and develop this as a product so that it will be useful for many other academic organizations.

1.5 Definitions and Acronyms

ASP – Active server pages

HTML – Hypertext markup language

CSS – Cascading style sheet.

SQL – Structured query language

2. Project Technical Description

To develop this application, we have used the following technologies. They are

Operating System: Windows.

Database: SQL server management studio 2014.

Application Software: ASP .Net

Interface Design: HTML, CSS.

2.1 Application Architecture

Here we are developing a web application that will be retrieved by many users. This application should be hosted on the server, and every user will be connected to the server as a client. With this information, we can determine the client-server architecture. As there will be clients, applications, and databases, we will have a 3-tier architecture.[1].

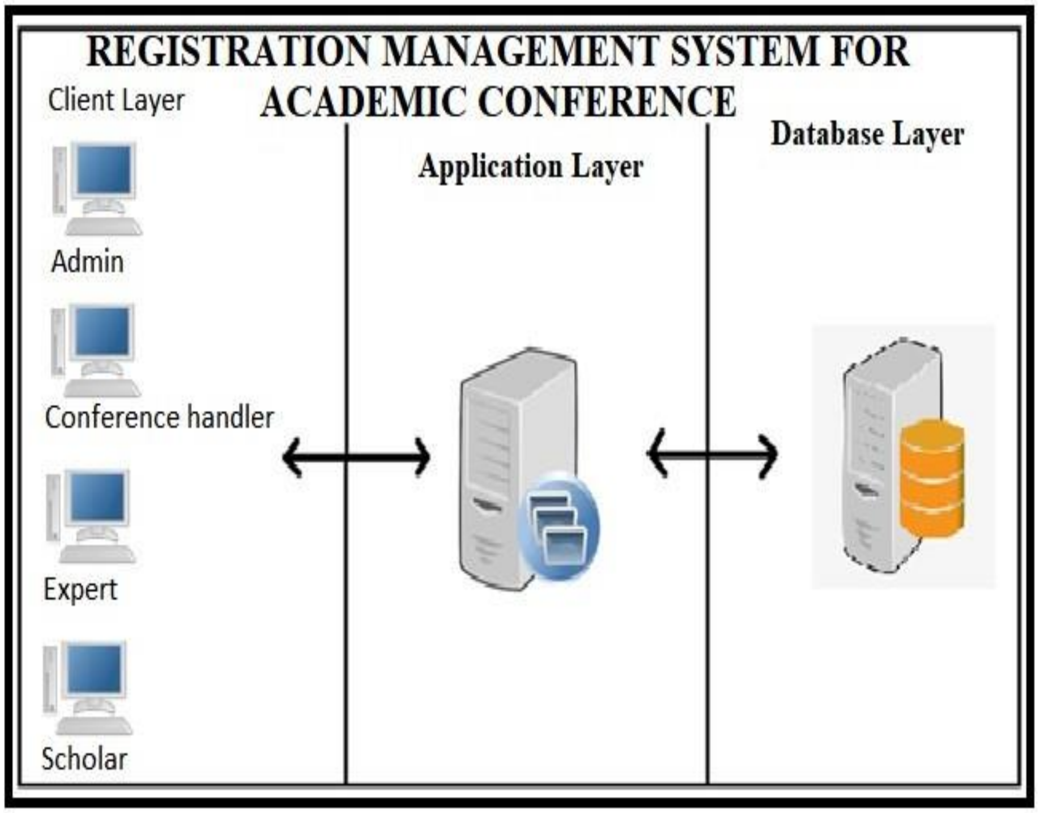


Figure 1: 3- Tier Architecture

2.2 Application Information flows

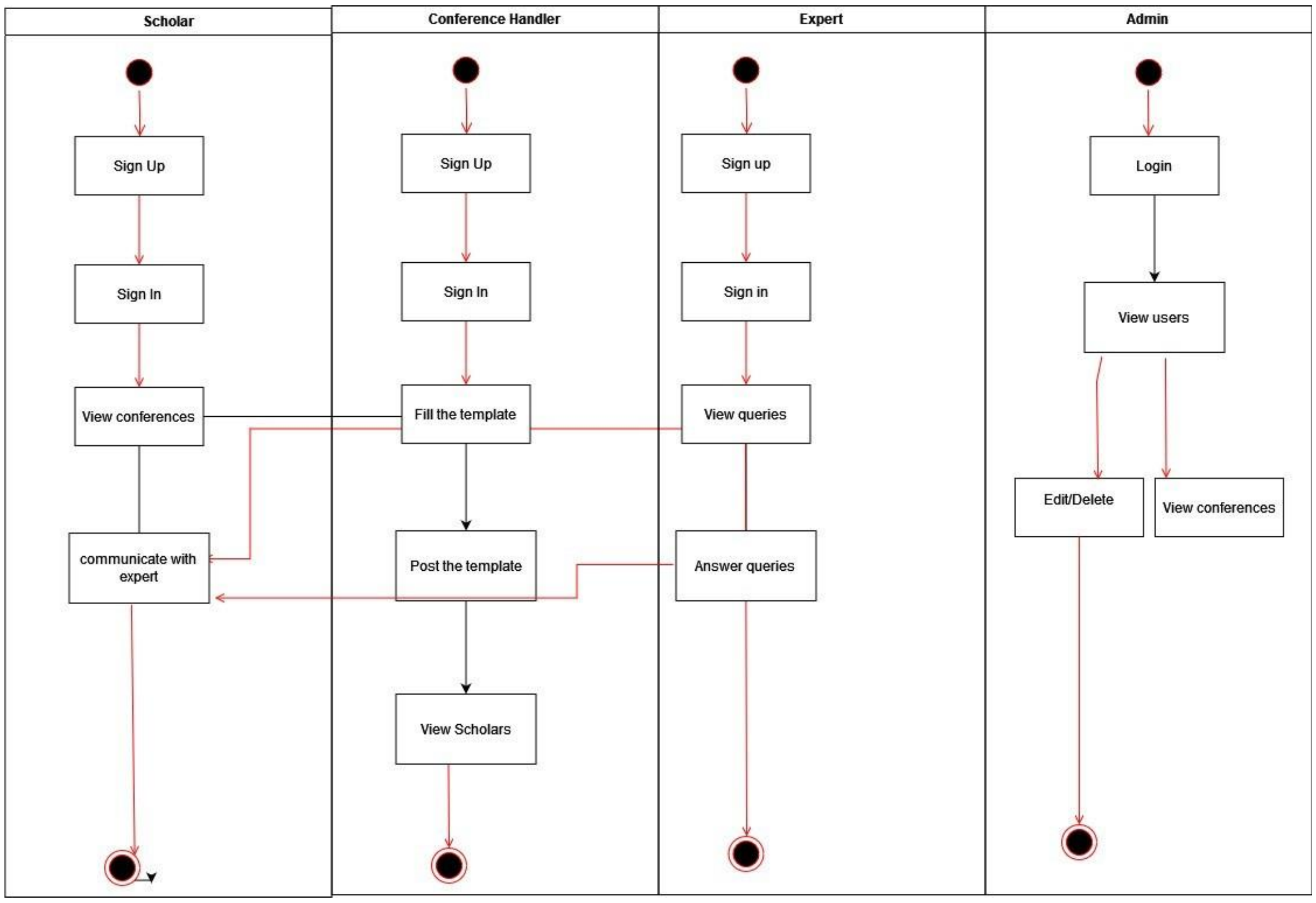


Figure 2: Application flow

2.3 Interactions with other Projects (if Any)

There are no interactions with any project.

2.4 Interactions with other Applications

No Interactions.

2.5 Capabilities

Following are the capabilities of different users.

Scholar:

Registration

Login

View conferences.

Communication

Expert

Registration

Login

View queries

Reply queries

Conference Handler:

Post conference.

Communicate with

scholars.View scholars

Admin:

View users

Delete users.

Delete conference

Edit conference.

2.6 Risk Assessment and Management

This project has started with the framing requirements. We analyzed the requirements at the end of the requirements. At the time of analysis, we understood all these requirements from the perspective of different types of users. Then some of the requirements are not on the list, and some of the requirements are not complete. If this process is not followed in the beginning stages, there is the possibility of project failure. That is the reason we need to do the risk assessment. This assessment process involves risk identification, analysis, prioritization, monitoring, mitigation, etc. All these steps are implemented in our project, and all the risks are handled properly.

3. Project Requirements

3.1 Identification of Requirements

This section provides the requirements of the application. All these requirements are implemented one after the other, and finally, all these requirements are verified, whether they are implemented or not.

GSU-Conference management_Scholar_2022-Registration-000100:

This requirement is related to the scholar; if the scholar wants to use the application, he needs to register with the application.

GSU-Conference management_Scholar_2022-Login-000102:

After registering with the application, scholars need to login with the application.

GSU-Conference management_Scholar_2022-View-000103:

Scholars should be able to view the conferences.

GSU-Conference management_Scholar_2022-Communicate-000104:

Scholars will be able to communicate with the conference manager.

GSU-Conference management_Scholar_2022-Querying expert-000105:

Scholars should be able to post any kind of query to the experts.

Expert:

GSU-Conference management_Expert_2022-Registration-000100:

Experts should register with the application.

GSU-Conference management_Expert_2022-Login-000102:

Experts should be able to login with the system.

GSU-Conference management_Expert_2022-View queries-000103:

Experts should be able to view the queries posed by the scholars.

GSU-Conference management_Expert_2022-Reply-000104:

Experts should be able to give replies to queries.

Conference Handler:

GSU-Conference management_Conference handler_2022-Registration-000100:

Conference handlers should register with the application.

GSU-Conference management_Conference handler_2022-Login-000101:

Conference handlers should login with the application.

GSU-Conference management_Conference handler_2022-post conference-000102:

Conference handlers should be able to post the new conference.

GSU-Conference management_Conference handler_2022-View scholar-000103:

He should be able to view the scholars who have registered with the conference.

GSU-Conference management_Conference handler_2022-communication-000104:

Conference handlers should be able to communicate with the scholars

Admin:

GSU-Conference management_Admin_2022-login-000100:

Administrators will be directly logging into the application.

GSU-Conference management_Admin_2022-View users-000101:

Administrators will be viewing the users.

GSU-Conference management_Admin_2022-delete-000102:

Administrators will be able to delete the users.

GSU-Conference management_Admin_2022-View conferences-000103:

Administrators will be able to view the conferences that are registered with the application.

GSU-Conference management_Admin_2022-Delete conferences-000104: Administrators will be able to delete the conferences.

3.2 Operations, Administration, Maintenance, and Provisioning (OAM&P)

As this application is related to the academic conference, there will be huge data storage, a chance of data leakage, and security and privacy issues. It is the responsibility of the management to provide security and privacy. As the application is implementing the username and password mechanism, it will prevent unauthorized access.

If there is any data loss because of the floods or because of any disaster, then organizations should implement different data recovery tools and mechanisms to get back the data that is lost.

3.3 Security and Fraud Prevention

To protect the data, organizations should implement internal and external security. All the people who are using this application need to be trained so that they understand the importance of security. The data that is being stored in the database should be protected with the help of access rights. One should not give the access right to every user to view all the data. Based on their level, access rights should be given to the users. This way, fraud can be avoided, and security can be established.

3.4 Release and Transition Plan

For every project, we need to have a schedule. This schedule provides information on when we need to start and when the project will be completed. In this schedule, there will be different phases, and all of these phases will be implemented one after the other. While implementing, if anything goes wrong, we can check the schedule and take the necessary steps to overcome it. Following is the schedule that gives information related to the phases of the development.

Phase No	Phase name	Number of days	initial date	Final date	Description	status
1	Preparing the abstract	7 days	05-09-22	13-09-22	Abstract preparation	Completed
2	Software requirements specification	10 days	15-09-22	25-09-22	Gathering the requirements	completed
3	Building a prototype	10	25-09-22	05-10-22	A prototype needs to build tounderstand before implementation.	completed
4	UML design	8 days	05-10-22	15-10-22	Project design should be had with different UML and other tools	completed
5	Database design	5 days	15-10-22	20-10-22	Storage structures need to be created	Completed
6	coding	20 days	20-10-22	15-11-22	Code should be written to implement the requirements	completed
7	User testing	3 days	15-11-22	19-11-22	Testing the application	completed.
8	Documentation	10 days	15-11-22	27-11-22	Project document should be	completed

9	Project execution and presentation	3 days	27-11-22	01-12-22	prepared as per the given template presentation	Not completed

Table 1: Project Schedule

4. Project Design Description

This section deals with the application design. The requirements are transformed into a representationby design. To represent the requirements, we will be using UML notations.

Module design:

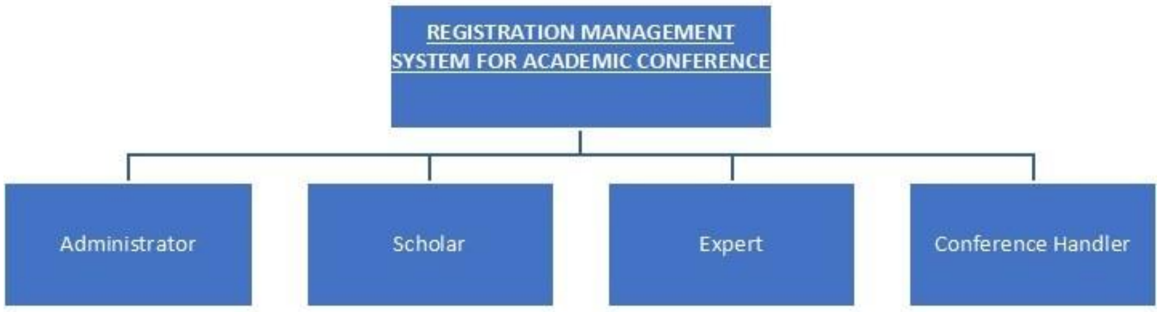


Figure 3: Module Design for Registration management of Academic conference

Use case diagram:

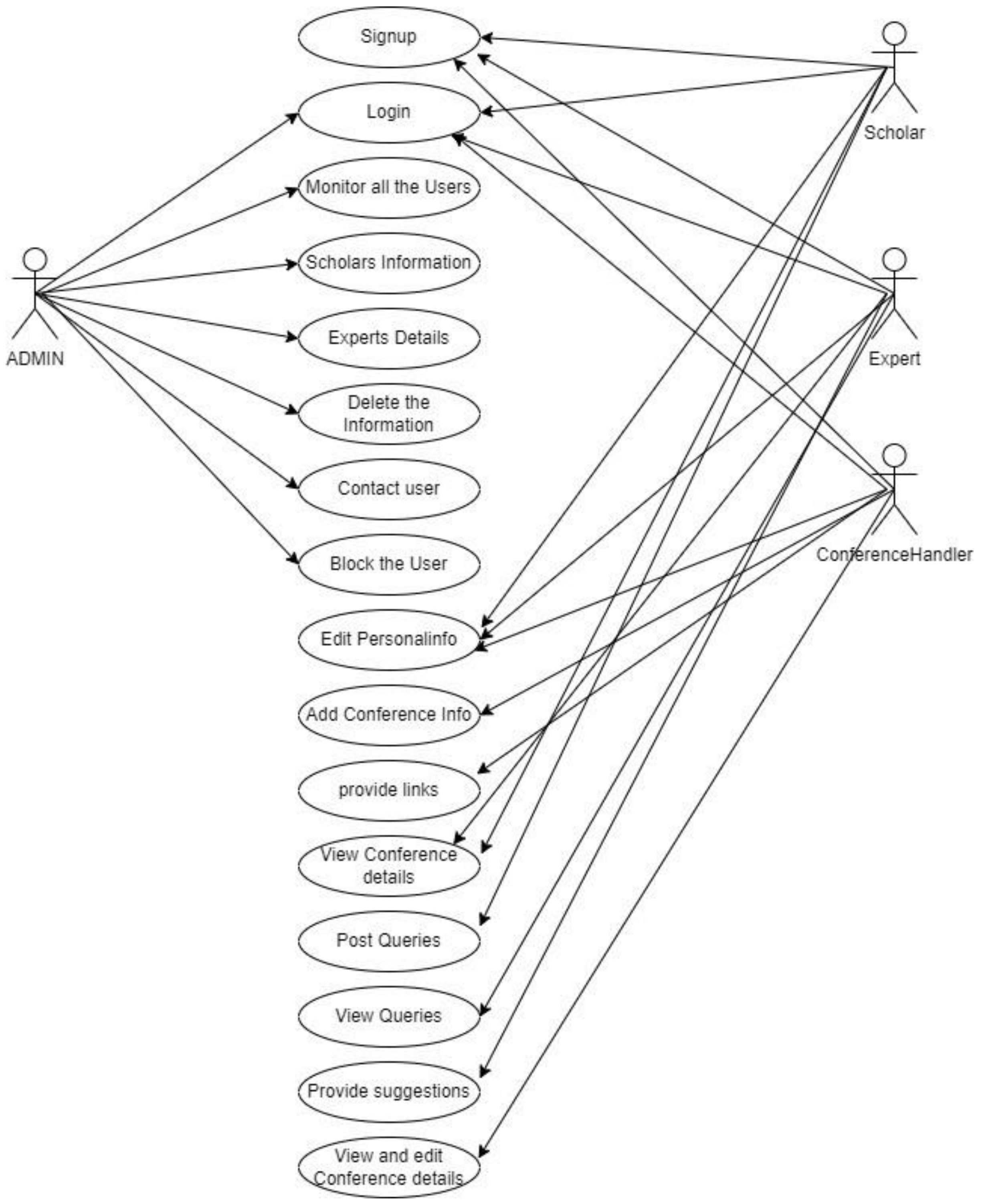


Figure 4: Use case diagram

Sequence Diagram:
Admin

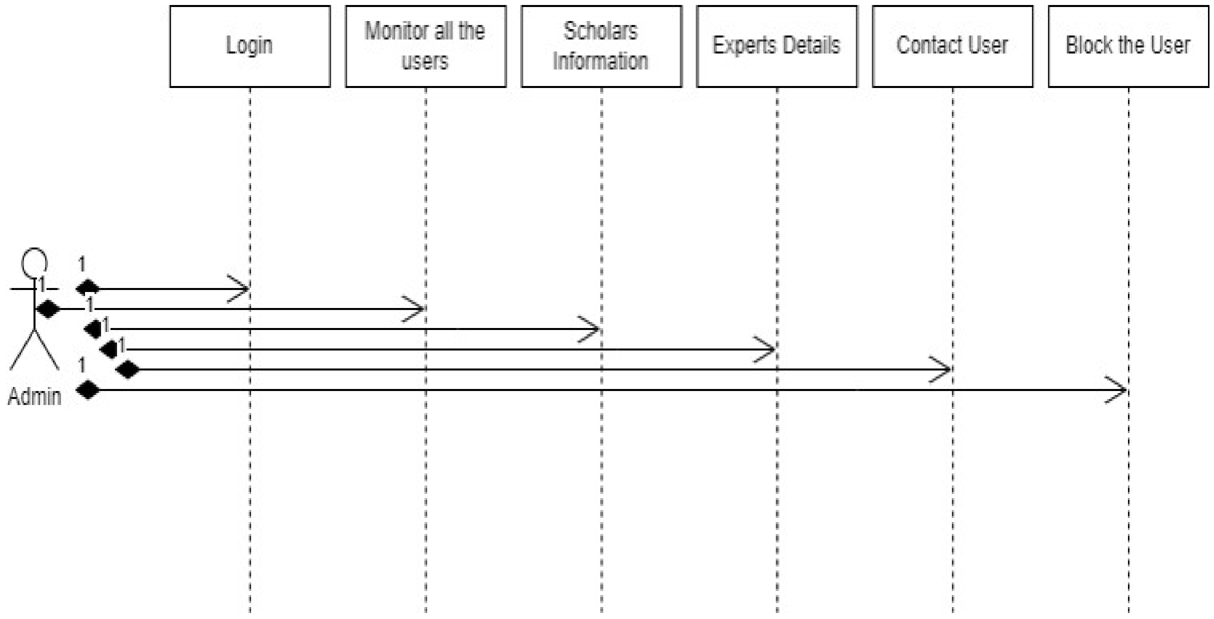


Figure 5: Sequence diagram for Admin operations

Scholar

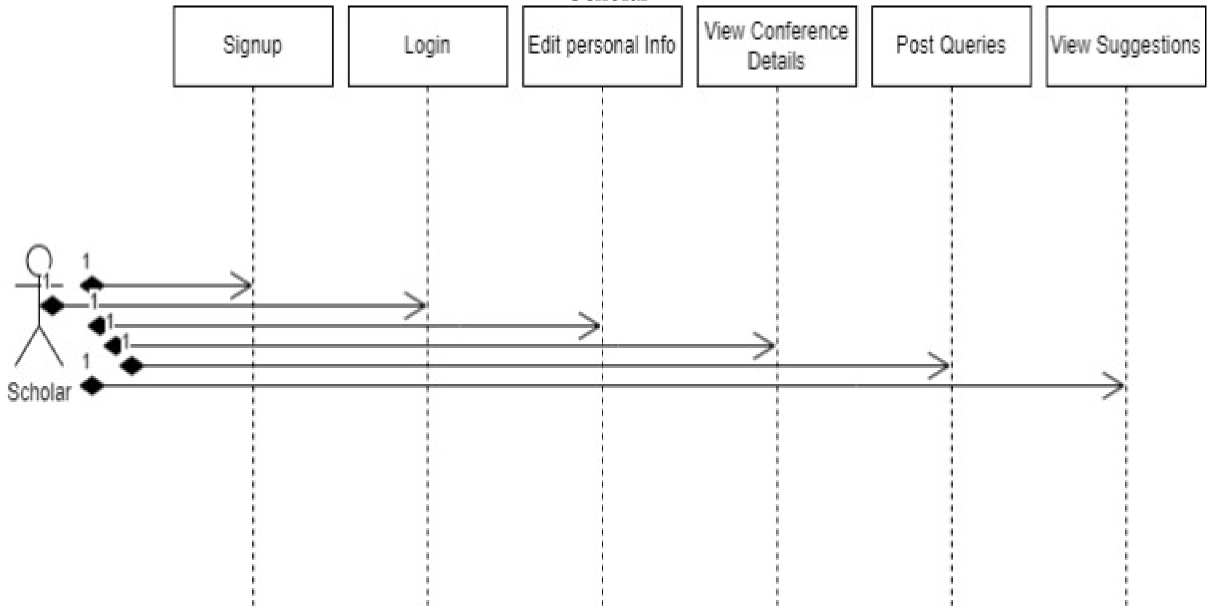


Figure 6: Sequence diagram for Scholar operations

Expert

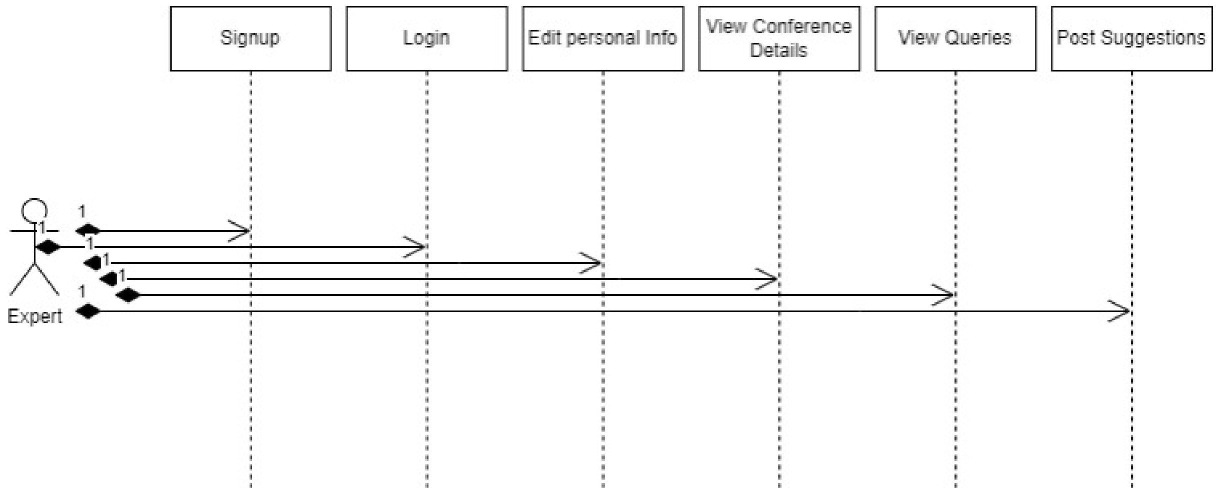


Figure 7: Sequence diagram for Expert operations

Conference Handler

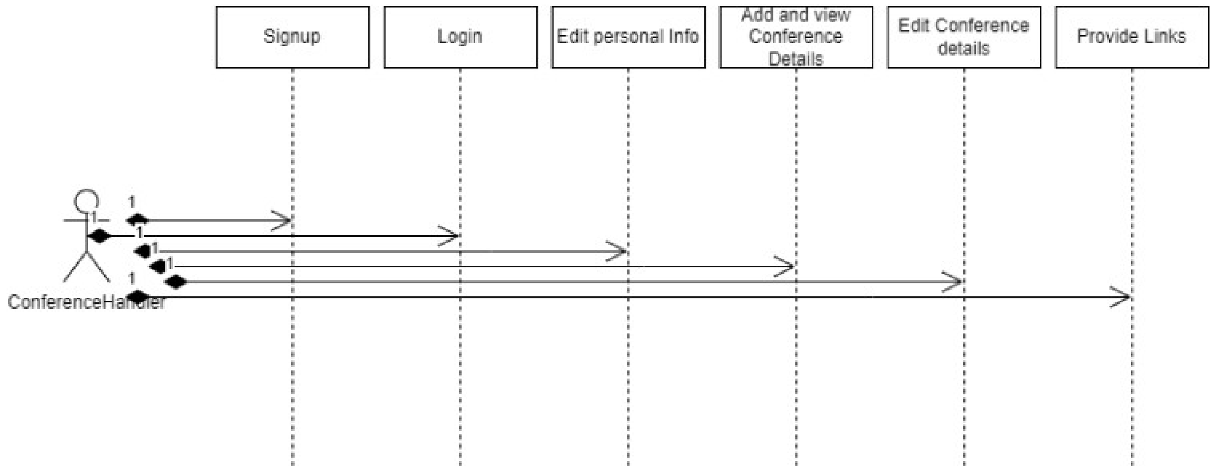
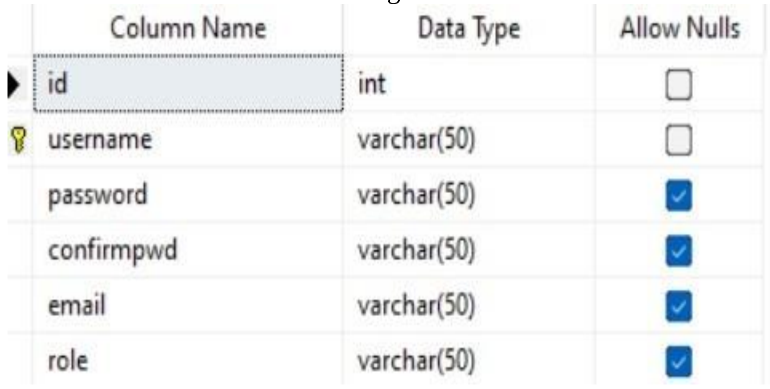


Figure 8: Sequence diagram for Conference handler operations

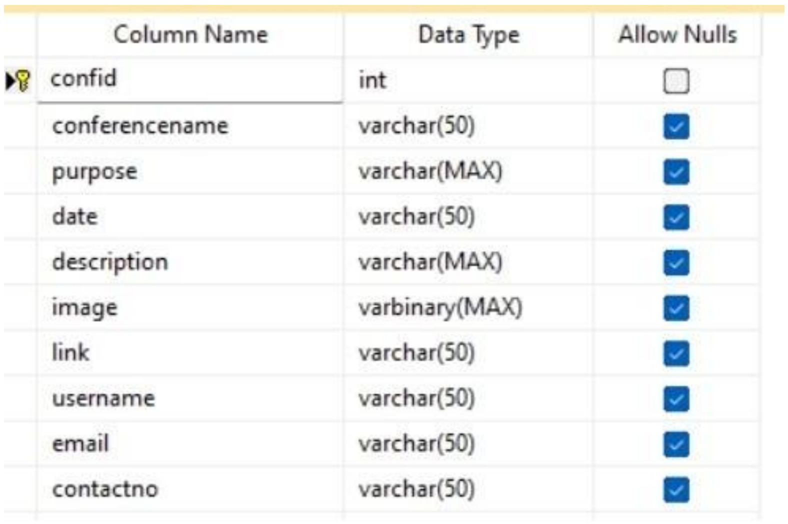
Database:
Tblregister:



Column Name	Data Type	Allow Nulls
id	int	☐
username	varchar(50)	☐
password	varchar(50)	☑
confirmpwd	varchar(50)	☑
email	varchar(50)	☑
role	varchar(50)	☑

Table 2: Registration table

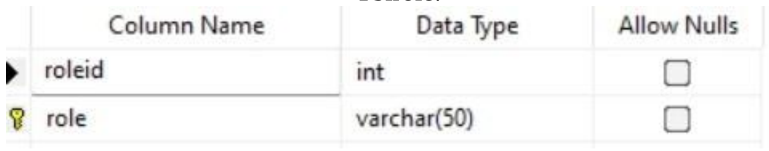
TblConference:



Column Name	Data Type	Allow Nulls
confid	int	☐
conferencename	varchar(50)	✓
purpose	varchar(MAX)	✓
date	varchar(50)	✓
description	varchar(MAX)	✓
image	varbinary(MAX)	✓
link	varchar(50)	✓
username	varchar(50)	✓
email	varchar(50)	✓
contactno	varchar(50)	✓

Table 3: Conference Table

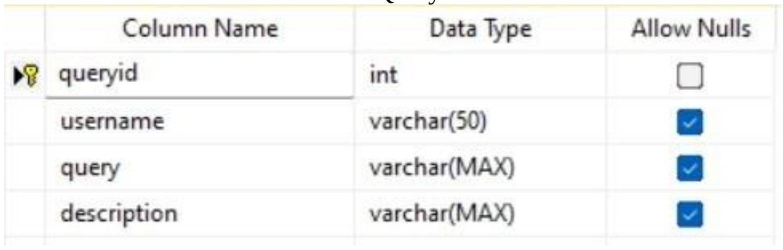
Tblrole:



Column Name	Data Type	Allow Nulls
roleid	int	☐
role	varchar(50)	☐

Table 4: Roles Table

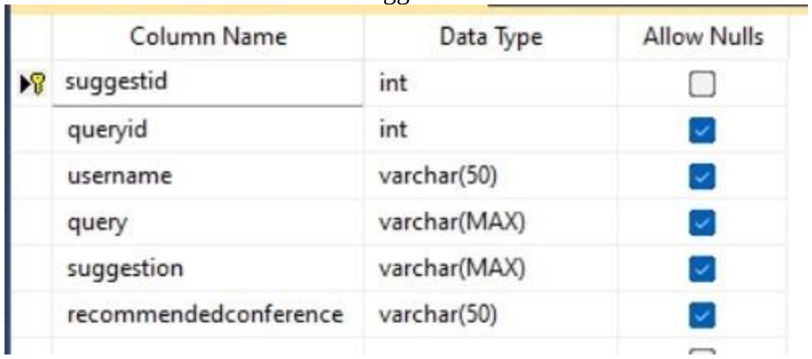
TblQuery:



Column Name	Data Type	Allow Nulls
queryid	int	☐
username	varchar(50)	✓
query	varchar(MAX)	✓
description	varchar(MAX)	✓

Table 5: Query Table

TblSuggestions:



Column Name	Data Type	Allow Nulls
suggestid	int	☐
queryid	int	✓
username	varchar(50)	✓
query	varchar(MAX)	✓
suggestion	varchar(MAX)	✓
recommendedconference	varchar(50)	✓

Table 6: Suggestions table

5. Internal/external Interface Impacts and Specification

This application will have four users. They are scholars, experts, conference handlers, and administrators. The application is developed in such a way that every user operation will not coincide with any other user operation. No user will be able to view other users' operations. Every user will have their own dashboard. They can register with the application, they can login to the application, and they can perform their operation. The privacy of the application is preserved.

Users of the application will interact with it with the help of user interfaces. They can store the data in the database with the help of user interfaces. They can even view the data through the user interfaces. The user can easily understand the functionality of the application. No additional training is required for any of the users.

6. Design Units Impacts

If the design is not made according to the requirements, then it will not satisfy the user's needs. As a result, the designer must design the database or user interface to meet the needs of the user. The design's impact is determined by the precise gathering of requirements. If the requirements are incorrect, the design willbe inadequate. If the design is not according to the needs of the user, then the implementation also will not satisfy the user. Enough care should be taken at the time of requirements, design, and implementation.

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

as the application is developed based on the requirements of the users. As the functions of all the users are clearly explained in the previous sections, their implementation and impacts are explained in this section.

6.1.2 Impacts

In terms of the effects on each user, scholars will be able to find all conference-related information in one place. People who want to conduct the conferences can make use of this application. With this application, information related to the conferences will go around the globe. Reachability will be increased with this application. Another thing is that scholars can contact experts and conference handlerswith this application. This application is very flexible for all its users.

6.1.3 Requirements

As the requirements are discussed in the earlier sections, these requirements are designed with the user interfaces in mind. These user interfaces are user-friendly.

The following page is the registration page, where the user has to sign up with:

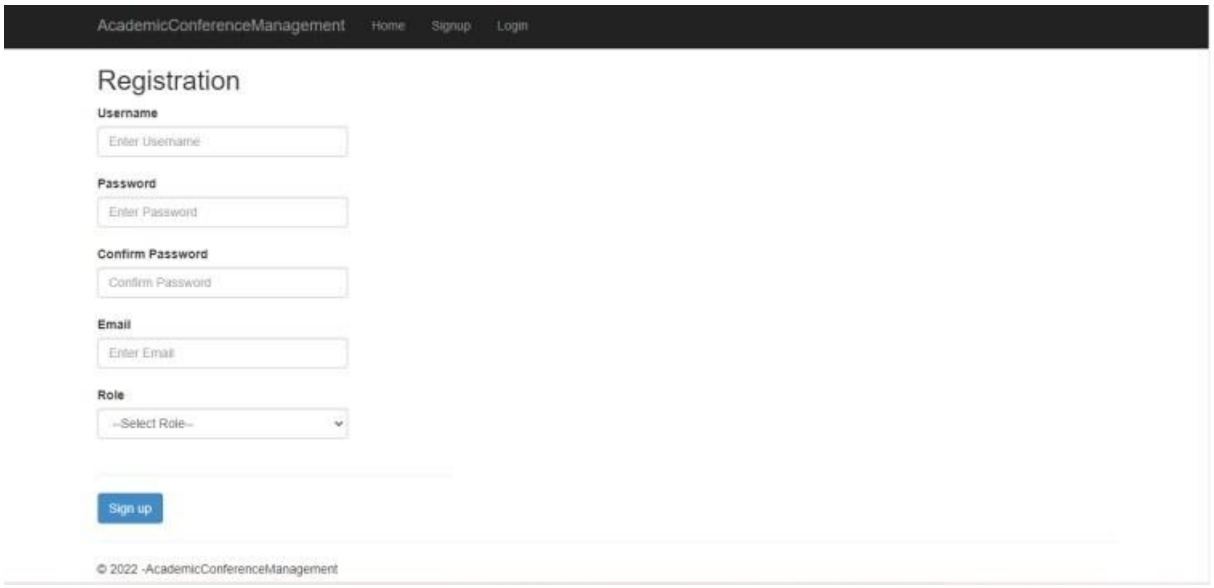


Figure 9: Registration User Interface

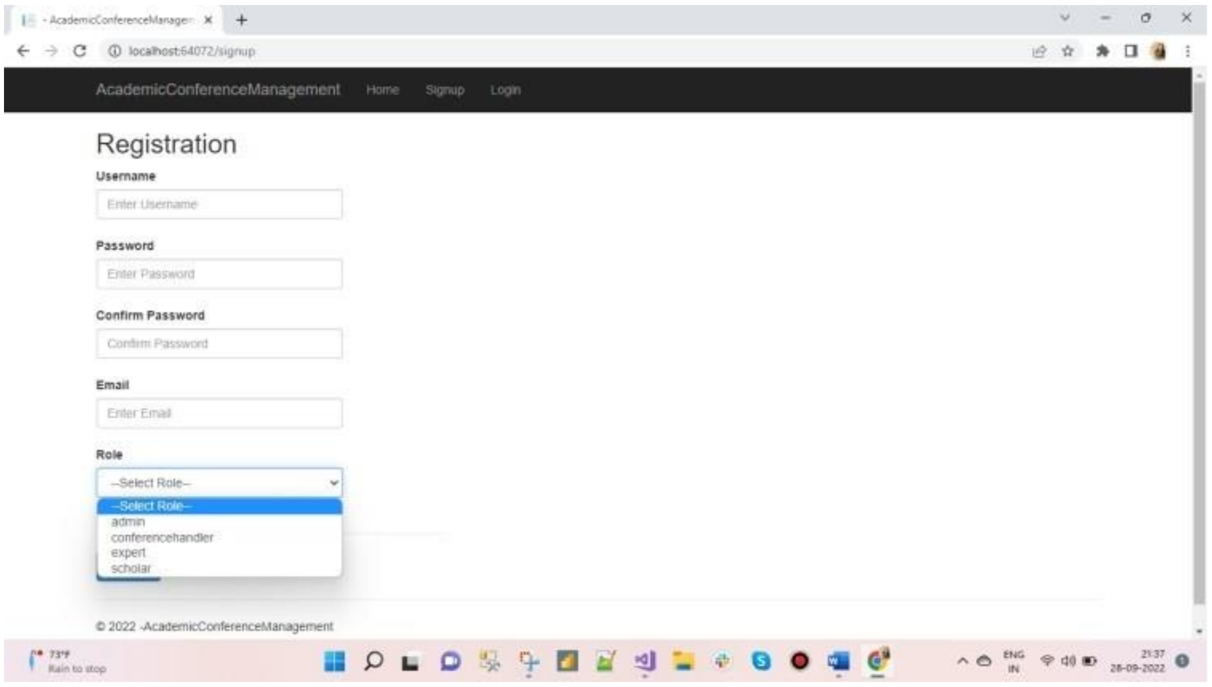


Figure 10: User interface showing the Roles in the drop-down list:

After the signup page user can log in through here and based on his Role he will redirect to his console:

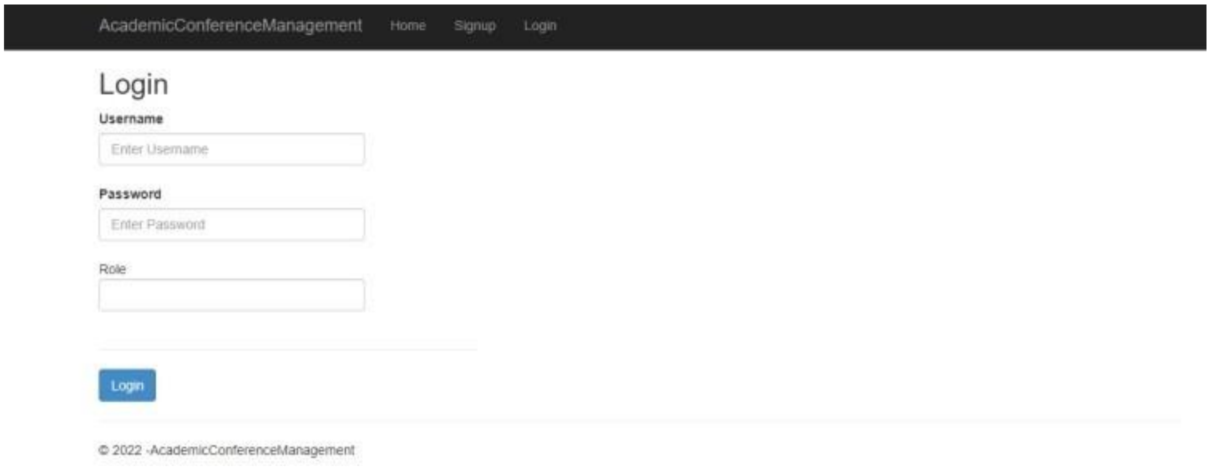


Figure 11: Login Interface

Login.aspx:

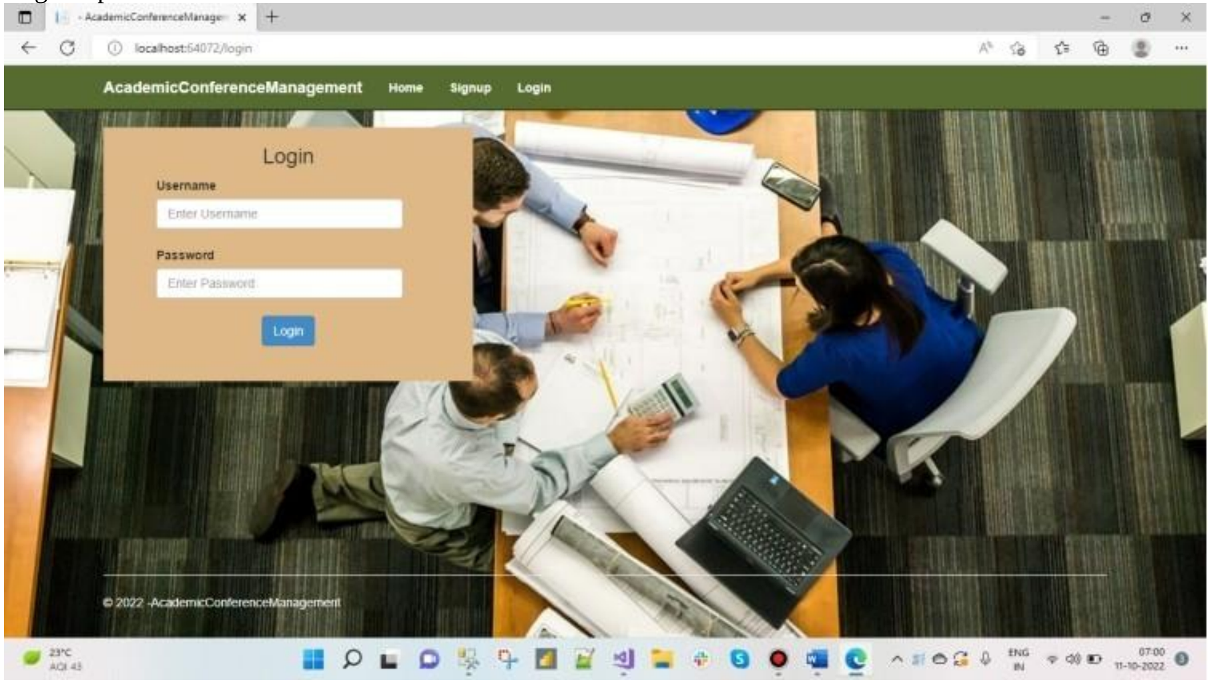


Figure 12: Admin homepage:

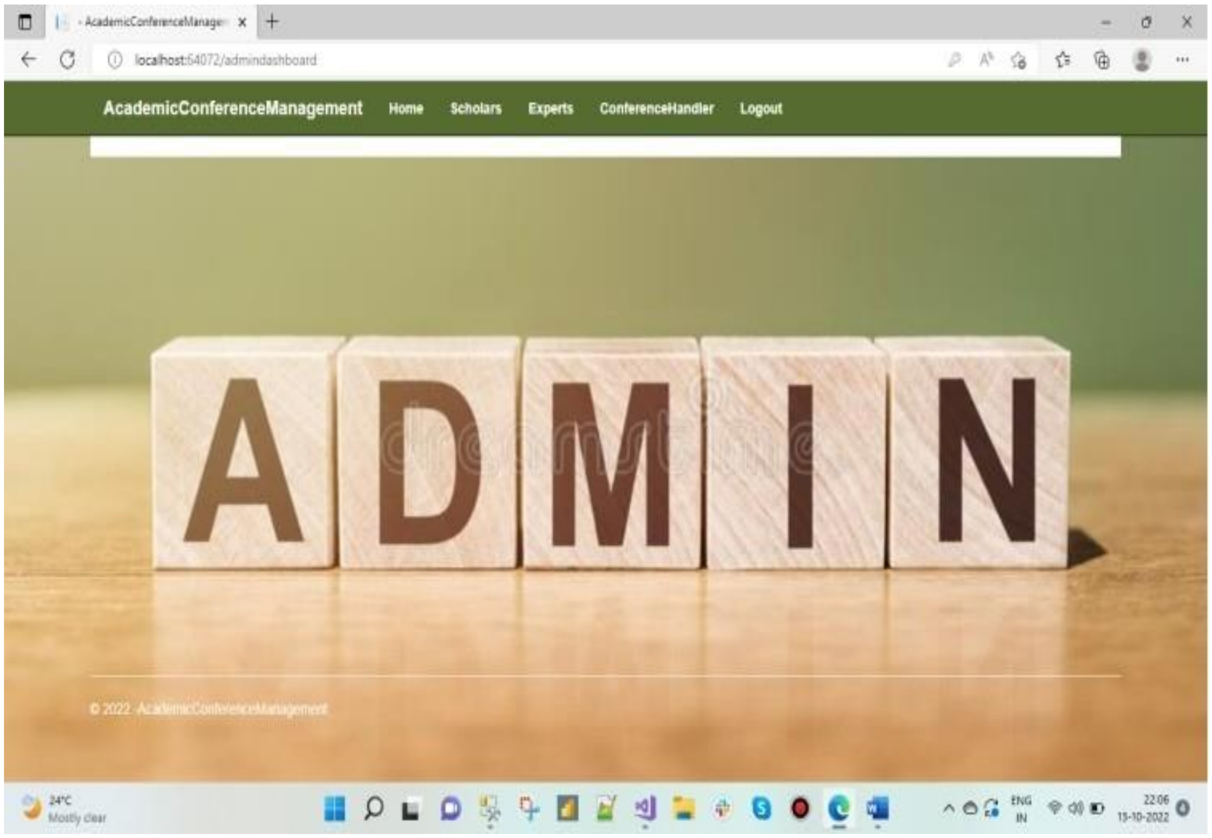

AcademicConferenceManagement
localhost:64072/admindashboard
AcademicConferenceManagement
Home
Scholars
Experts
ConferenceHandler
Logout
ADMIN
© 2022 - AcademicConferenceManagement

Figure 13: Admin Home page

Conference handler:

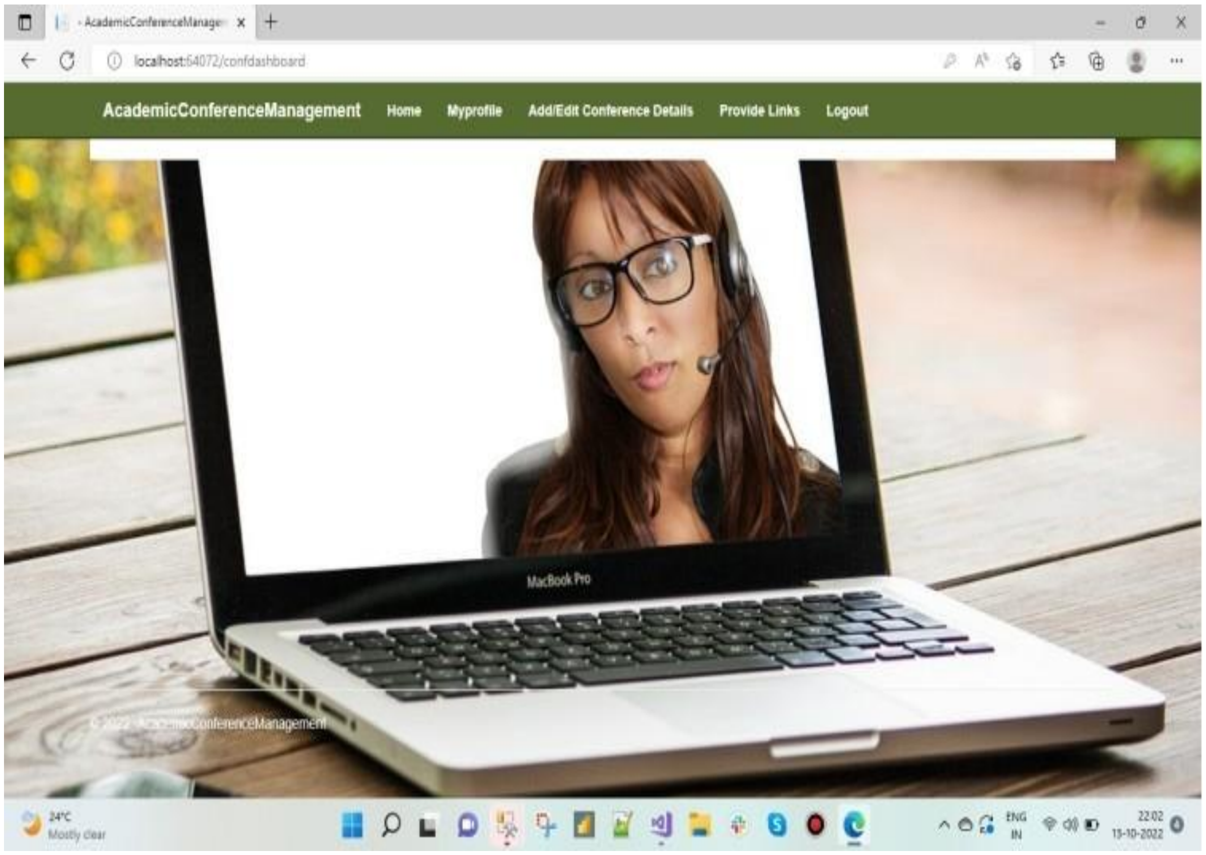
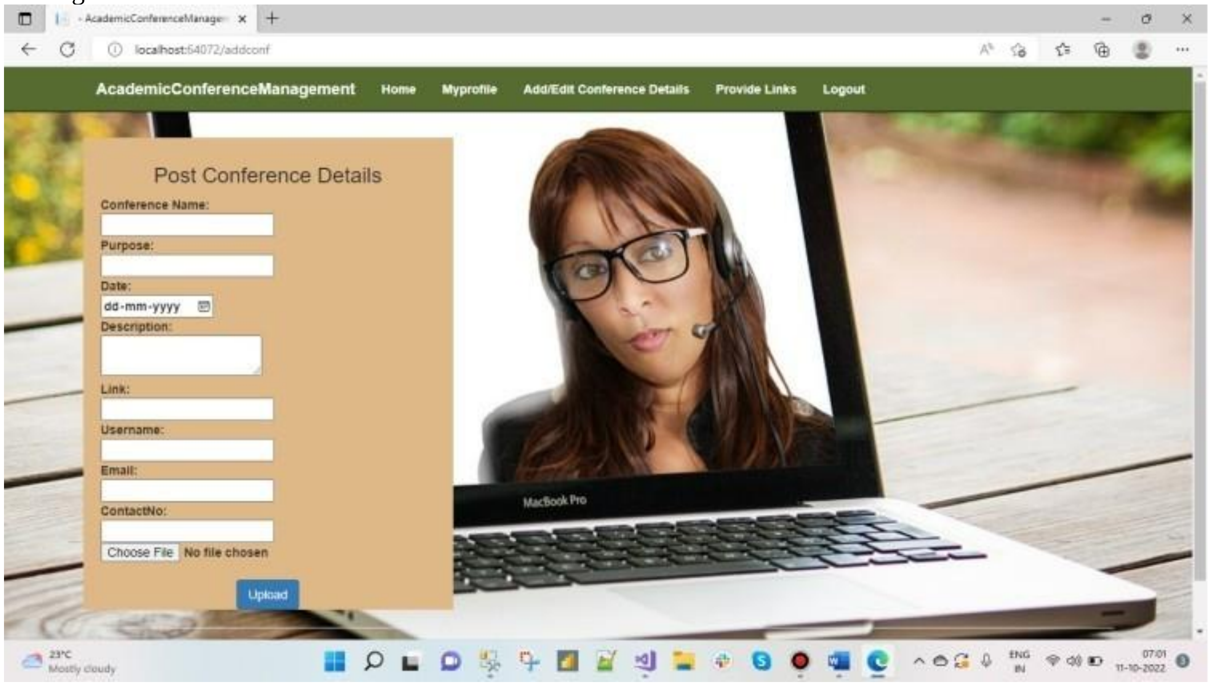

AcademicConferenceManagement
localhost:64072/confdashboard
AcademicConferenceManagement
Home
Myprofile
Add/Edit Conference Details
Provide Links
Logout
© 2022 - AcademicConferenceManagement

Figure 14: Conference handler home page

Adding conference details:



Post Conference Details

Conference Name:

Purpose:

Date: dd-mm-yyyy

Description:

Link:

Username:

Email:

ContactNo:

Choose File No file chosen

Upload

Figure 15: Adding conference details

Expert homepage:

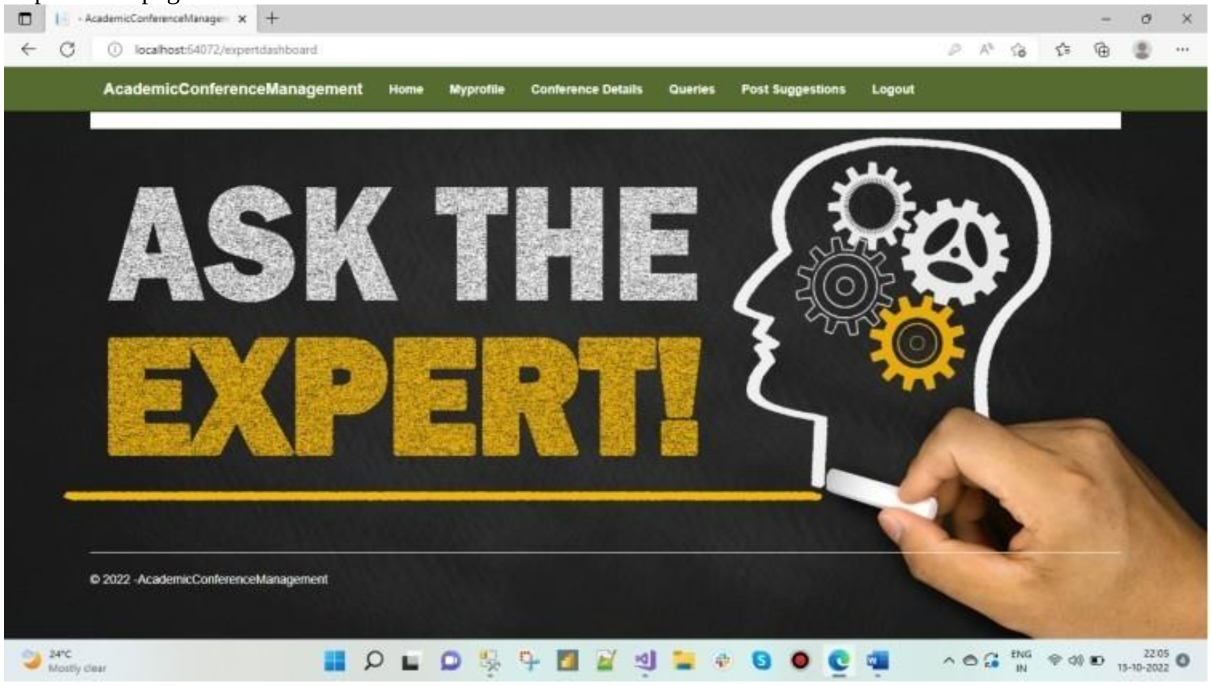

localhost:54072/expertdashboard
AcademicConferenceManagement
Home
Myprofile
Conference Details
Queries
Post Suggestions
Logout
ASK THE EXPERT!
© 2022 -AcademicConferenceManagement

Figure 16: Expert Home page

Scholar homepage:

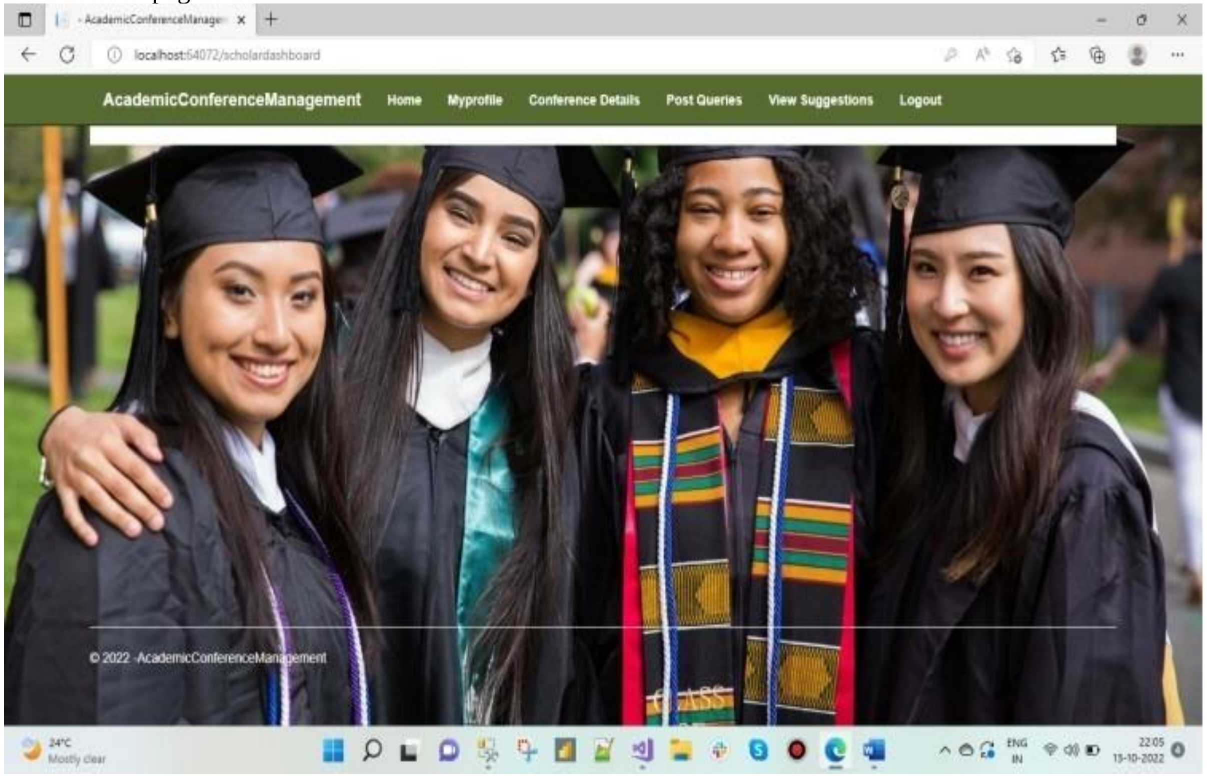

localhost:64072/scholardashboard
AcademicConferenceManagement
Home
Myprofile
Conference Details
Post Queries
View Suggestions
Logout
© 2022 -AcademicConferenceManagement

Figure 17: Scholar home page

Design:

Signup page:

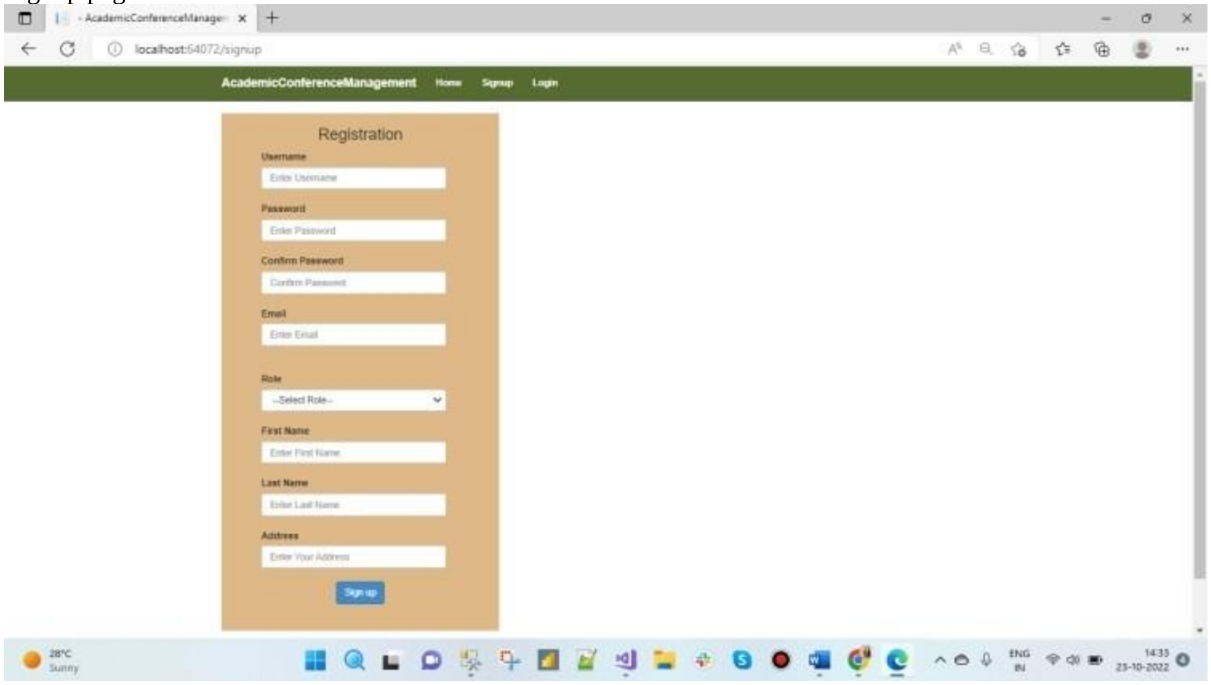
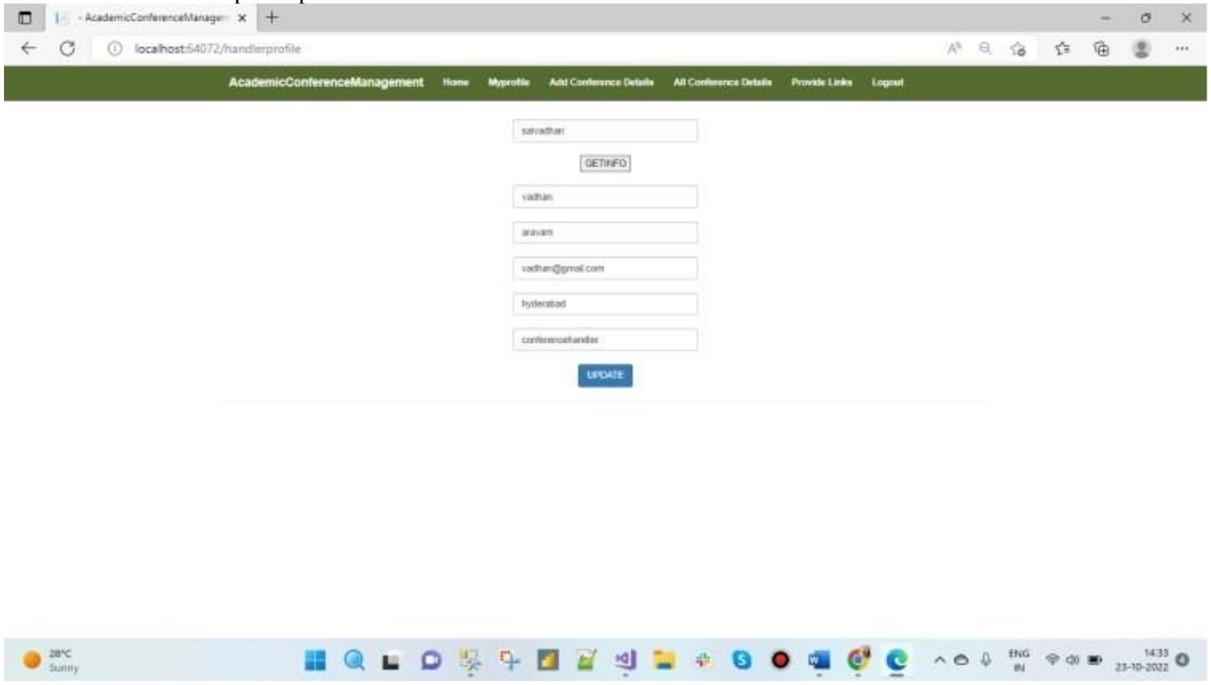


Figure 18: Signup page

Conference handler update profile:


localhost:54072/handlerprofile

salvadhan

vadhan

aravam

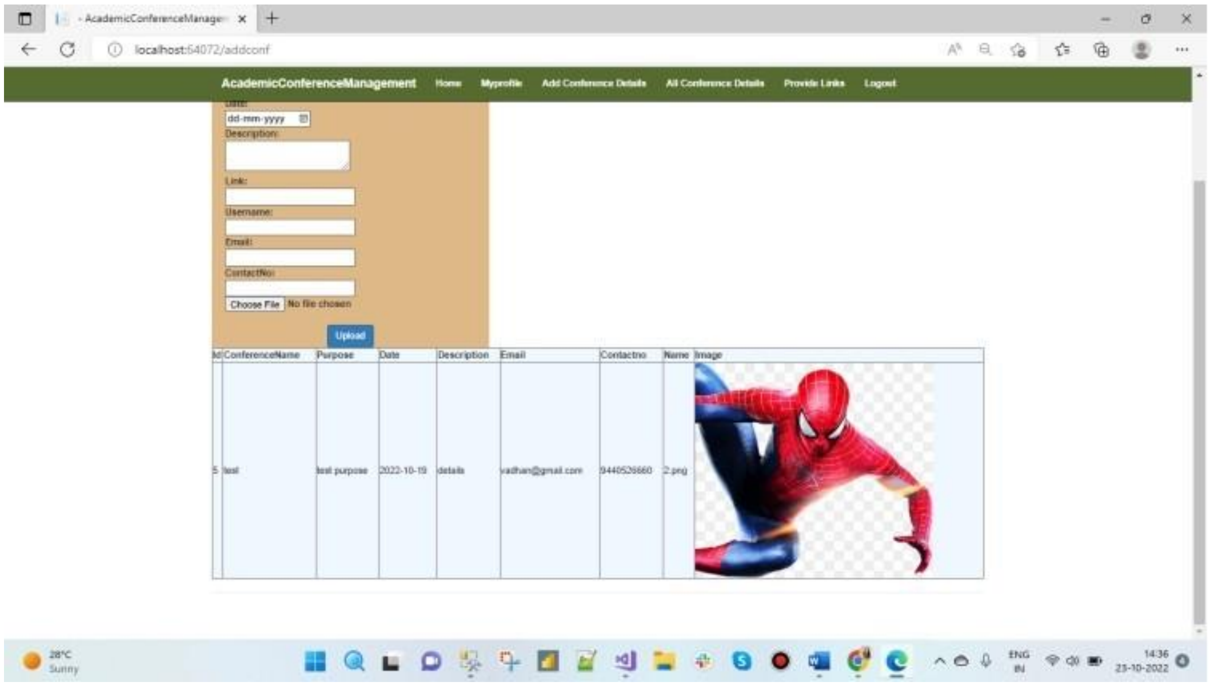
vadhan@gmail.com

hyderabad

conferencehandler

Figure 19: Conference handler page

Add conference details:


localhost:54072/addconf

Date: dd-mm-yyyy

Description:

Link:

Username:

Email:

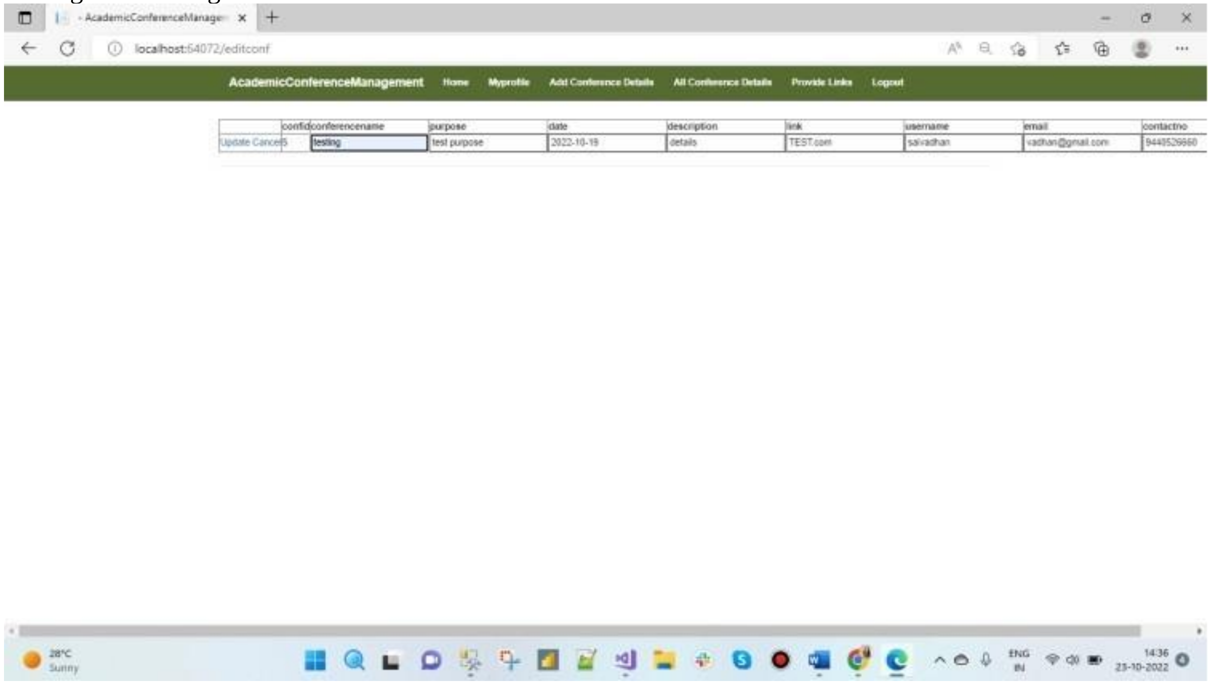
ContactNo:

Choose File No file chosen

Id	ConferenceName	Purpose	Date	Description	Email	Contactno	Name	Image
5	test	test purpose	2022-10-19	details	vadhan@gmail.com	9440526660	2.png	

Figure 19: Add conference details

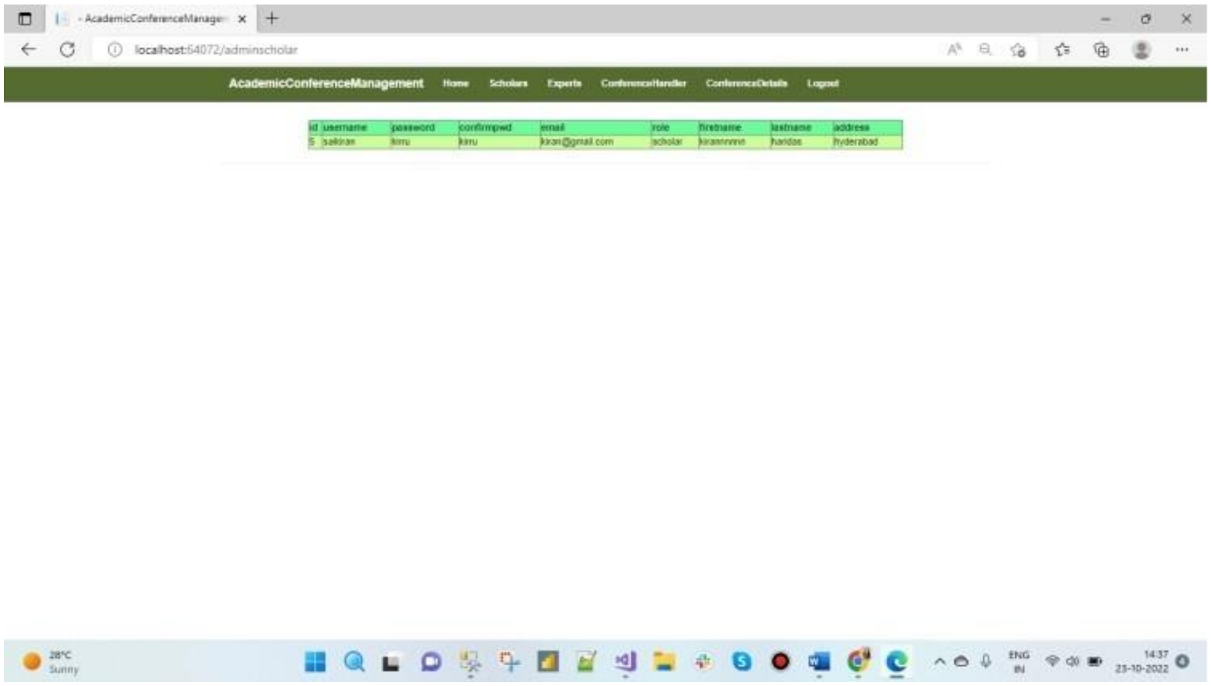
Editing and deleting conference details:



	confid	conferencename	purpose	date	description	link	username	email	contactno
Update Cancel	5	testing	test purpose	2022-10-19	details	TEST.com	saivadhan	vadhan@gmail.com	9440526660

Figure 20: Editing and deleting conference details

Admin can view scholar details:



id	username	password	confirmpwd	email	role	firstname	lastname	address
5	saikiran	kiru	kiru	kiran@gmail.com	scholar	kirannnnnn	haridas	hyderabad

Figure 21: Admin view

7. Open Issues

When we started the development of this application, we were not sure about the application's requirements. Then we gathered the requirements from similar applications, and we understood what should be in our application. Data flow between the users is one of the main issues, and the integration of all the modules is another issue. Somehow, we completed the project by implementing all the requirements.

8. Acknowledgements

We would like to thank everyone who assisted us in completing the project successfully.

9. References

[1]. Ramakrishnan, R., Gehrke, J., & Gehrke, J. (2003). *Database management systems* (Vol. 3). New York:McGraw-Hill.

[2]. Galloway, J., Haack, P., Wilson, B., & Allen, K. S. (2012). *Professional ASP. NET MVC 4*. John Wiley &Sons.

[3]. Freeman, A., & Sanderson, S. (2013). *Pro Asp. net Mvc 4* (Vol. 832). Apress.

[4]. Kanav, S., Lammich, P., & Popescu, A. (2014, July). A conference management system with verified document confidentiality. In *International Conference on Computer Aided Verification* (pp. 167-183). Springer, Cham.

10. Appendices

Edit conference details

Editconf.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/ConferenceHandler.Master"
AutoEventWireup="true" CodeBehind="editconf.aspx.cs"
Inherits="AcademicConferenceManagement.editconf" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">

<div style="padding-top:30px">
    <center>

    <asp:GridView ID="confgrid" runat="server" OnRowDataBound="confgrid_RowDataBound"
DataKeyNames="confid" Width="900px"
        OnRowEditing="confgrid_RowEditing"
        OnRowCancelingEdit="confgrid_RowCancelingEdit"
        OnRowDeleting="confgrid_RowDeleting"
        OnRowUpdating="confgrid_RowUpdating" AutoGenerateDeleteButton="true"
AutoGenerateEditButton="true"></asp:GridView>

    </center>[2]
```

```
</div>

</asp:Content>
```

Editconf.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AcademicConferenceManagement
{
    public partial class editconf : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                this.BindGrid();
            }
        }
        private void BindGrid()
        {
            string constr = ConfigurationManager.ConnectionStrings["connect"].ConnectionString;
            using (SqlConnection con = new SqlConnection(constr))
            {
                using (SqlCommand cmd = new SqlCommand("SELECT
confid,conferencename,purpose,date,description,link,username,email,contactno  FROM
tblconference"))
                {
                    using (SqlDataAdapter sda = new SqlDataAdapter())
                    {
                        cmd.Connection = con;
                        sda.SelectCommand = cmd;
                        using (DataTable dt = new DataTable())
                        {
                            sda.Fill(dt);
                            confgrid.DataSource = dt;
                            confgrid.DataBind();
                        }
                    }
                }
            }
        }
        protected void confgrid_RowDataBound(object sender, GridViewRowEventArgs e)
        {
            if (e.Row.RowType == DataControlRowType.DataRow && e.Row.RowIndex !=
confgrid.EditIndex)
            {
                (e.Row.Cells[0].Controls[2] as LinkButton).Attributes["onclick"] = "return
confirm('Do you want to delete this row?');";
            }
        }

        protected void confgrid_RowEditing(object sender, GridViewEditEventArgs e)
        {
            confgrid.EditIndex = e.NewEditIndex;
            this.BindGrid();
        }

        protected void confgrid_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
        {
            confgrid.EditIndex = -1;
            this.BindGrid();

        }

        protected void confgrid_RowDeleting(object sender, GridViewDeleteEventArgs e)
        {
            int confid = Convert.ToInt32(confgrid.DataKeys[e.RowIndex].Values[0]);
            string constr = ConfigurationManager.ConnectionStrings["connect"].ConnectionString;
            using (SqlConnection con = new SqlConnection(constr))
            {
                using (SqlCommand cmd = new SqlCommand("DELETE FROM tblconference WHERE confid =
@confid"))
                {
                    cmd.Parameters.AddWithValue("@confid", confid);
                    cmd.Connection = con;
                    con.Open();
                    cmd.ExecuteNonQuery();
                    con.Close();
                }
            }
            this.BindGrid();
        }

        protected void confgrid_RowUpdating(object sender, GridViewUpdateEventArgs e)
        {
            GridViewRow row = confgrid.Rows[e.RowIndex];
            int confid = Convert.ToInt32(confgrid.DataKeys[e.RowIndex].Values[0]);
            string conferencename= (row.Cells[2].Controls[0] as TextBox).Text;
            string purpose = (row.Cells[3].Controls[0] as TextBox).Text;
            string date = (row.Cells[4].Controls[0] as TextBox).Text;
            string description= (row.Cells[5].Controls[0] as TextBox).Text;
            string link= (row.Cells[6].Controls[0] as TextBox).Text;
            string username= (row.Cells[7].Controls[0] as TextBox).Text;
            string email= (row.Cells[8].Controls[0] as TextBox).Text;
            string contactno= (row.Cells[9].Controls[0] as TextBox).Text;

            string constr = ConfigurationManager.ConnectionStrings["connect"].ConnectionString;
            using (SqlConnection con = new SqlConnection(constr))
            {
                using (SqlCommand cmd = new SqlCommand("UPDATE tblconference SET conferencename =
@conferencename,
purpose=@purpose,date=@date,description=@description,link=@link,username=@username,email=@email,c
ontactno=@contactno where confid=@confid"))
                {
                    cmd.Parameters.AddWithValue("@confid", confid);
                    cmd.Parameters.AddWithValue("@conferencename", conferencename);
                    cmd.Parameters.AddWithValue("@purpose", purpose);
                    cmd.Parameters.AddWithValue("@date", date);
                    cmd.Parameters.AddWithValue("@description", description);
                    cmd.Parameters.AddWithValue("@link", link);
                    cmd.Parameters.AddWithValue("@username", username);
                    cmd.Parameters.AddWithValue("@email", email);
                    cmd.Parameters.AddWithValue("@contactno", contactno);
```

```
                        cmd.Connection = con;
                        con.Open();
                        cmd.ExecuteNonQuery();
                        con.Close();
                    }
                }
                confgrid.EditIndex = -1;
                this.BindGrid();[3]
            }
        }
    }
```

Edit profile:

Expertprofile.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Expert.Master" AutoEventWireup="true"
CodeBehind="expertprofile.aspx.cs" Inherits="AcademicConferenceManagement.expertprofile" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">

     <div style="padding-top:30px">

        <center>
            <asp:TextBox ID="txtusername" runat="server" CssClass="form-control"
Enable="false"></asp:TextBox><br />
            <asp:Button ID="btngo" runat="server" Text="GETINFO"
OnClick="btngo_Click"></asp:Button><br /><br />
<asp:TextBox ID="txtfname" runat="server" CssClass="form-control"></asp:TextBox><br />
            <asp:TextBox ID="txtlname" runat="server" CssClass="form-control"></asp:TextBox><br
/>
            <asp:TextBox ID="txtemail" runat="server" CssClass="form-control"></asp:TextBox><br
/>
            <asp:TextBox ID="txtaddress" runat="server" CssClass="form-control"></asp:TextBox><br
/>

            <asp:TextBox ID="txtrole" runat="server" CssClass="form-control"></asp:TextBox><br />

          <asp:Button ID="btnupdate" runat="server" Text="UPDATE" CssClass="btn btn-primary"
OnClick="btnupdate_Click"></asp:Button>

        </center>
    </div>



</asp:Content>
```

Expertprofile.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
namespace AcademicConferenceManagement
{
    public partial class expertprofile : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            txtusername.Text = Session["username"].ToString();

        }

        protected void btnupdate_Click(object sender, EventArgs e)
        {
            string constr = ConfigurationManager.ConnectionStrings["connect"].ToString();
            SqlConnection con = new SqlConnection(constr);
            con.Open();
            SqlCommand cmd = new SqlCommand("update tblregister set
username=@username,email=@email,role=@role,firstname=@firstname,lastname=@lastname,address=@addre
ss where username='" + txtusername.Text + "'", con);
            cmd.Parameters.AddWithValue("@username", txtusername.Text);
            cmd.Parameters.AddWithValue("@email", txtemail.Text);
            cmd.Parameters.AddWithValue("@role", txtrole.Text);
            cmd.Parameters.AddWithValue("@firstname", txtfname.Text);
            cmd.Parameters.AddWithValue("@lastname", txtlname.Text);
            cmd.Parameters.AddWithValue("@address", txtaddress.Text);
            cmd.ExecuteNonQuery();
            con.Close();
            Response.Write("<script>alert('Your profile updated Successfully');</script>");
        }

        protected void btngo_Click(object sender, EventArgs e)
        {
            try
            {
                string strcon = ConfigurationManager.ConnectionStrings["connect"].ToString();
                SqlConnection con = new SqlConnection(strcon);
                if (con.State == ConnectionState.Closed)
                {
                    con.Open();
                }
                SqlCommand cmd = new SqlCommand("SELECT * from tblregister where username='" +
txtusername.Text + "'", con);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                da.Fill(dt);
                if (dt.Rows.Count >= 1)
                {
                    txtfname.Text = dt.Rows[0]["firstname"].ToString();
                    txtlname.Text = dt.Rows[0]["lastname"].ToString();
                    txtemail.Text = dt.Rows[0]["email"].ToString();
                    txtaddress.Text = dt.Rows[0]["address"].ToString();
                    txtrole.Text = dt.Rows[0]["role"].ToString();

                }
                else
                {
                    Response.Write("<script>alert('user does not exists');</script>");
                }

            }
            catch(Exception ex)
            {
            }
        }
    }
}
```

Scholar posting query:

Postqueries.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Scholar.Master" AutoEventWireup="true"
CodeBehind="postqueries.aspx.cs" Inherits="AcademicConferenceManagement.postqueries" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">

     <div style="padding-top:30px">

        <center>
            Username:
<asp:TextBox ID="txtusername" runat="server" CssClass="form-control"
Enabled="false"></asp:TextBox><br />
            Query:
            <asp:TextBox ID="txtquery" runat="server" CssClass="form-control"></asp:TextBox><br
/>
            Details:
            <asp:TextBox ID="txtdescription" runat="server" CssClass="form-
control"></asp:TextBox><br />

          <asp:Button ID="btnpost" runat="server" Text="POST" CssClass="btn btn-primary"
OnClick="btnpost_Click"></asp:Button>

        </center>
    </div>




</asp:Content>
```

Postqueries.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AcademicConferenceManagement
{
    public partial class postqueries : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            txtusername.Text = Session["username"].ToString();
        }

        protected void btnpost_Click(object sender, EventArgs e)
        {
            string constr = ConfigurationManager.ConnectionStrings["connect"].ToString();
            SqlConnection con = new SqlConnection(constr);
            con.Open();
            SqlCommand cmd = new SqlCommand("insert into tblquery 
values(@username,@query,@description)", con);
            cmd.Parameters.AddWithValue("@username", txtusername.Text);
            cmd.Parameters.AddWithValue("@query", txtquery.Text);
            cmd.Parameters.AddWithValue("@description", txtdescription.Text);
            cmd.ExecuteNonQuery();
            Response.Write("<script>alert('Your Query Posted Successfully');</script>");

        }
    }
}
```

Experts can view queries from a scholar:
Expertqueries.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Expert.Master" AutoEventWireup="true" 
CodeBehind="expertqueries.aspx.cs" Inherits="AcademicConferenceManagement.expertqueries" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">

    <div style="padding-top:30px">
        <center>
            <h4>Queries</h4>
            <br />
<asp:GridView ID="GridView1" runat="server" Width="900px" HeaderStyle-BackColor="#ccffff" 
BackColor="#ccff99" OnSelectedIndexChanging="GridView1_SelectedIndexChanging" 
AutoGenerateSelectButton="true"></asp:GridView>

        </center>
    </div>

</asp:Content>
```

Expertqueries.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace AcademicConferenceManagement
{
    public partial class expertqueries : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                string constr = 
ConfigurationManager.ConnectionStrings["connect"].ConnectionString;
                using (SqlConnection conn = new SqlConnection(constr))
                {
                    using (SqlDataAdapter sda = new SqlDataAdapter("SELECT 
queryid,username,query,description FROM tblquery", conn))
                    {
                        DataTable dt = new DataTable();
                        sda.Fill(dt);
                        GridView1.DataSource = dt;
                        GridView1.DataBind();
                    }
                }
            }
        }

        protected void GridView1_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
        {

        }
    }
}
```