



Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

B. Tech CE Semester - VI
Subject : Service Oriented Computing (CE-619)

A PROJECT REPORT ON

SMART CONTACT MANAGER

By

PATEL YASH K. (CE127) (19CEUON068)
PATEL LUV P. (CE124) (19CEUON069)

Guided By:

Prof. Prashant M. Jadav
Associate Professor
Dept. Of Computer Engineering

Prof. Ankit P. Vaishnav
Assistant Professor
Dept. Of Computer Engineering



Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of
Service Oriented Computing and recorded in this journal is the
bonafide work of

PATEL YASH K. (CE127) (19CEUON068)

PATEL LUV P. (CE124) (19CEUON069)

of B.Tech semester **VI** in the branch of **Computer Engineering** during the academic
year **2021-2022**.

Prof. Ankit P. Vaishnav
Assistant Professor,
Dept. of Computer Engg.,
Faculty of Technology,
Dharmsinh Desai
University, Nadiad

Prof. Prashant M. Jadav
Associate Professor,
Dept. of Computer Engg.,
Faculty of Technology,
Dharmsinh Desai
University, Nadiad

Dr. C. K. Bhensdadia,
Head,
Dept. of Computer Engg.,
Faculty of Technology,
Dharmsinh Desai
University, Nadiad

Table Of Contents

Abstract	4
Purpose	4
Scope	4
Introduction	5
Brief Introduction	5
Technology/Platform/Tools used	5
Software Requirements Specification - SRS	6
Functional Requirements	6
1 Account module	6
2 Contact module	7
Design	8
Class Diagram	9
Use case Diagram	10
Sequence Diagram	11
Activity Diagram	13
ER Diagram	14
Data Dictionary	14
Implementation Detail	16
Modules	16
Methods and techniques used	17
Testing	22
Screenshots	23
Conclusion	28
Functionalities not implemented	28
Limitations and future extensions	28
Bibliography	29

Abstract

This Smart Contact Manager System is a web application that provides the platform for users to manage their contacts individually. Users can create and manage contacts.

Purpose

The purpose of this document is to collect and present ideas, requirements & analysis done in order to develop this system. This document provides a detailed overview of the system, details of the functionality provided to users, target audience of the system and their user interfaces.

Scope

The scope of the system is providing a handy web application to users for contact management. Any user with a registered account on the system can use the functionality & get benefits of the provided features.

Introduction:

- Brief Introduction

The Smart Contact Manager is a web application for users to minimize their day to day contact management. The system works around only on End Users. Users can Register and login to the system. Users can create, update, view and delete the contacts. System checks all the corner cases for authentication, authorization of the users, validation of input data, duplication of data and gives appropriate error response messages according to the situation.

- Technology/Platform/Tools used

Technology :

- WCF framework.
- ASP.NET framework
- SQL Server database.
- Bootstrap

Platform :

- Windows

Tools :

- Visual Studio

Software Requirements Specification - SRS

Total 1 type of users are there in system:

1. User

Functional Requirements

R.1 Account module

R.1.1 Login

Description: User can login to the system

Exception Flow: If credentials are incorrect or insufficient data is provided

Input: User data

Output: Redirects to Dashboard

R.1.2 Register

Description: User can login to the system

Exception flow: If username is already taken or insufficient data is provided

Input: User data

Output: Redirects to Login page with success message

R.1.3 Forgot Password

Input: Email

Output: Reset Password mail send to provided email address.

R.1.4 Reset Password

Input: password, confirm password

Output: Password change successfully and Redirects to Login page

R.1.5 Logout

Input: User selection

Output: Redirects to Login page with success message

R.2 Contact module

R.2.1 Create contact

Exception flow: If contact number is already taken or insufficient data is provided

Input: Contact data

Output: Success message

R.2.2 View contact

Exception flow: 404 If contact not found or Access denied when data is not allowed to access

Input: User selection

Output: Contact details

R.2.3 Update contact

Exception flow: If new contact number is already taken or insufficient data is provided

Input: Contact data

Output: Success message

R.2.4 Delete contact

Input: User selection

Output: Success message

R.2.5 View Recently added contacts

Description: Last 3 recently added contact list can be viewed

Input: User selection

Output: Contacts list

R.2.6 Import contacts

Description: Contacts can be imported in by .vcf file

Input: .vcf file

Output: Success message

R.2.7 Export contacts

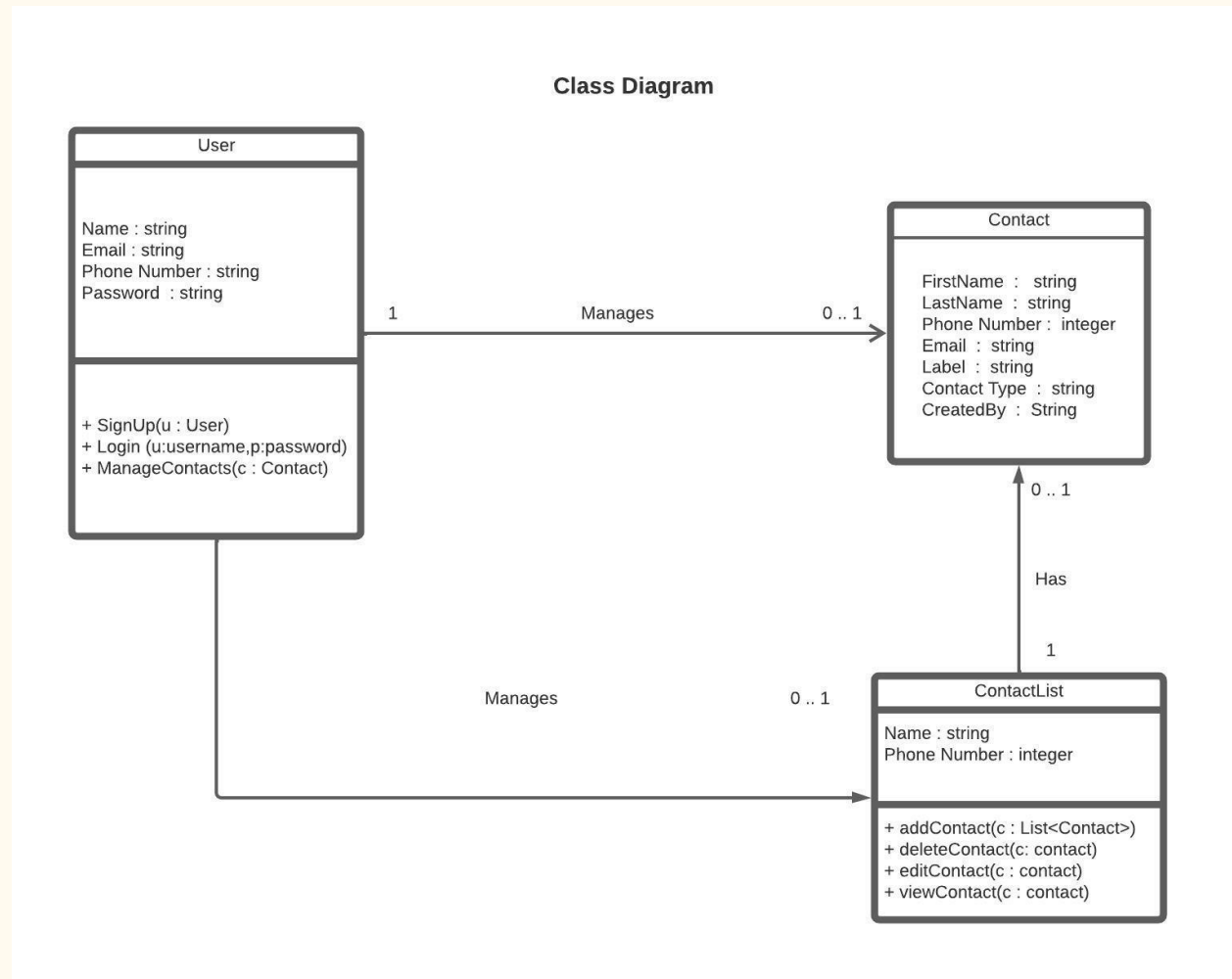
Description: Contacts can be exported in .vcf file

Input: User selection

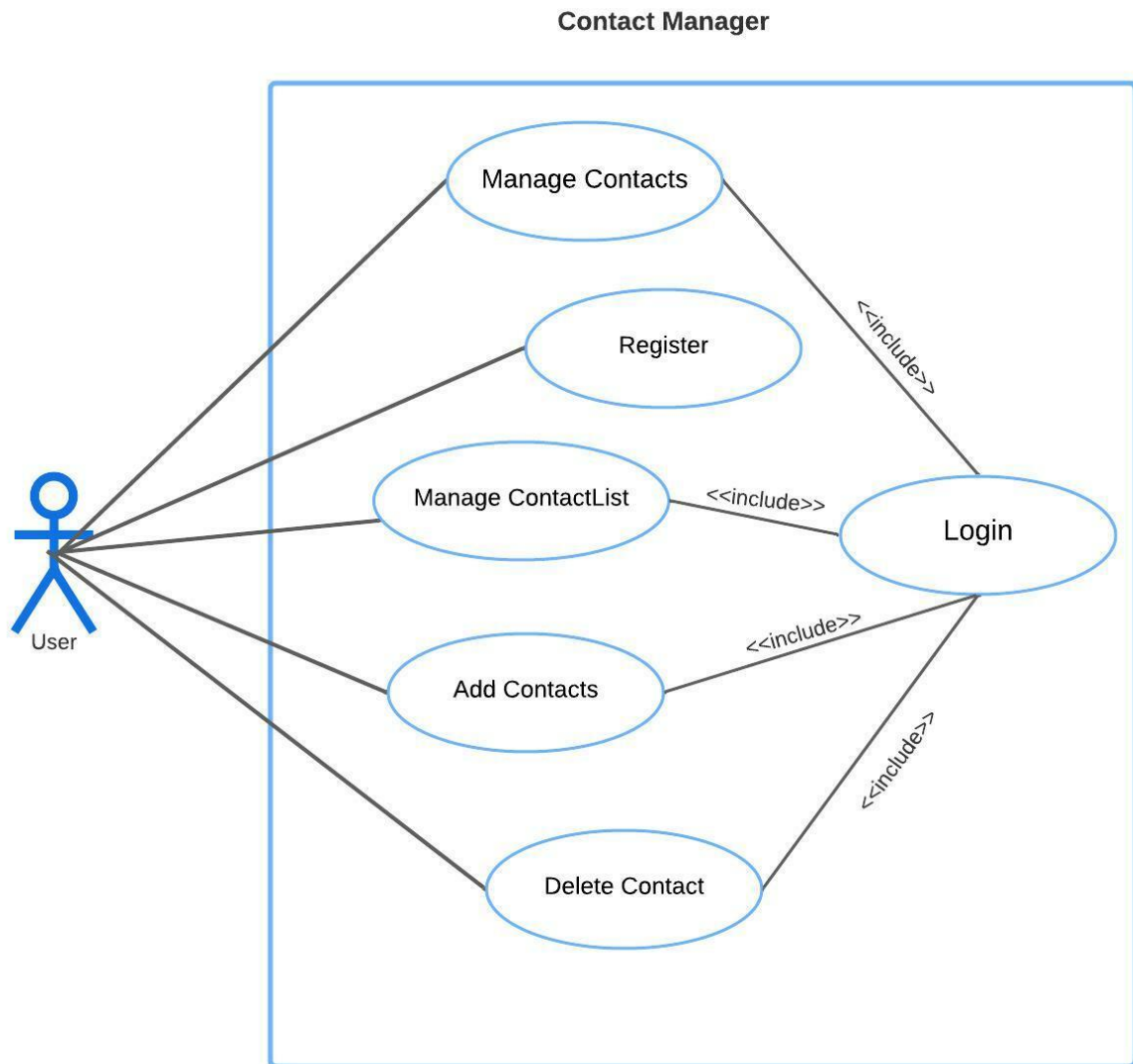
Output: .vcf file

Design

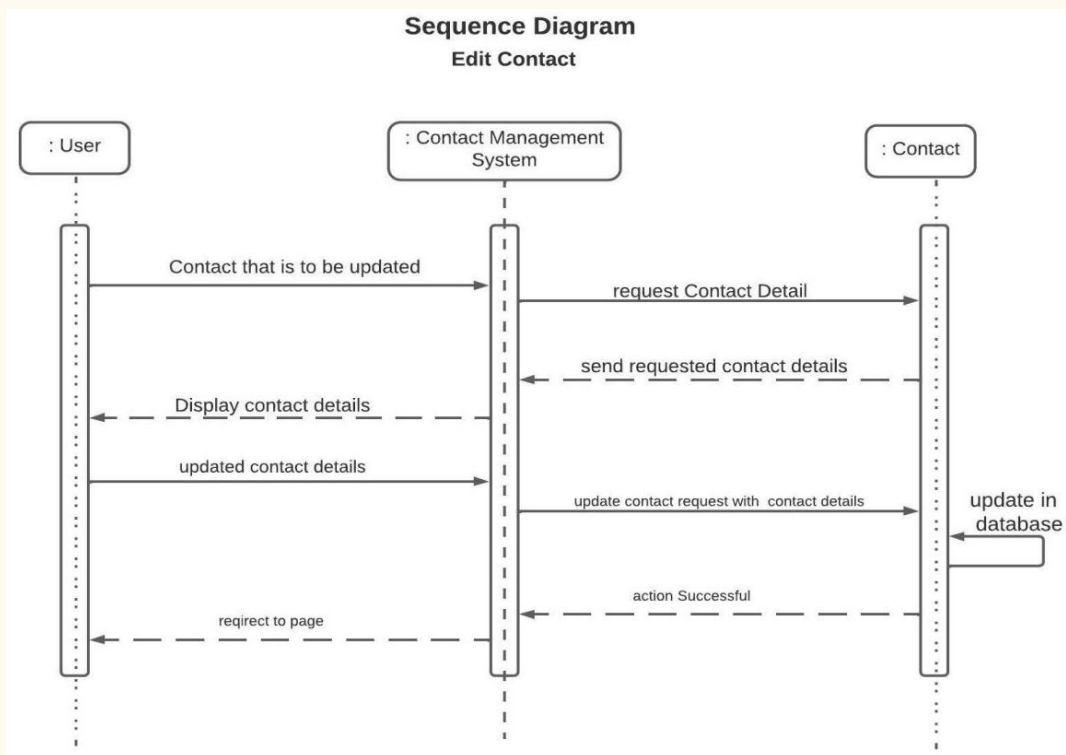
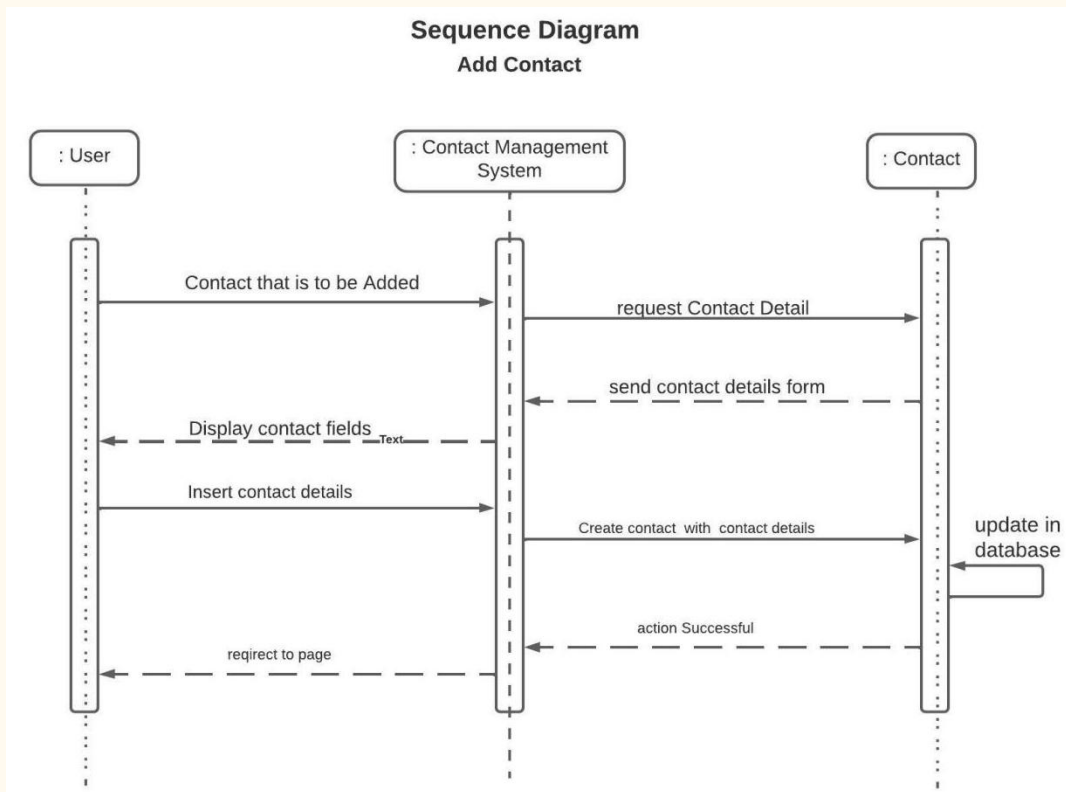
Class Diagram



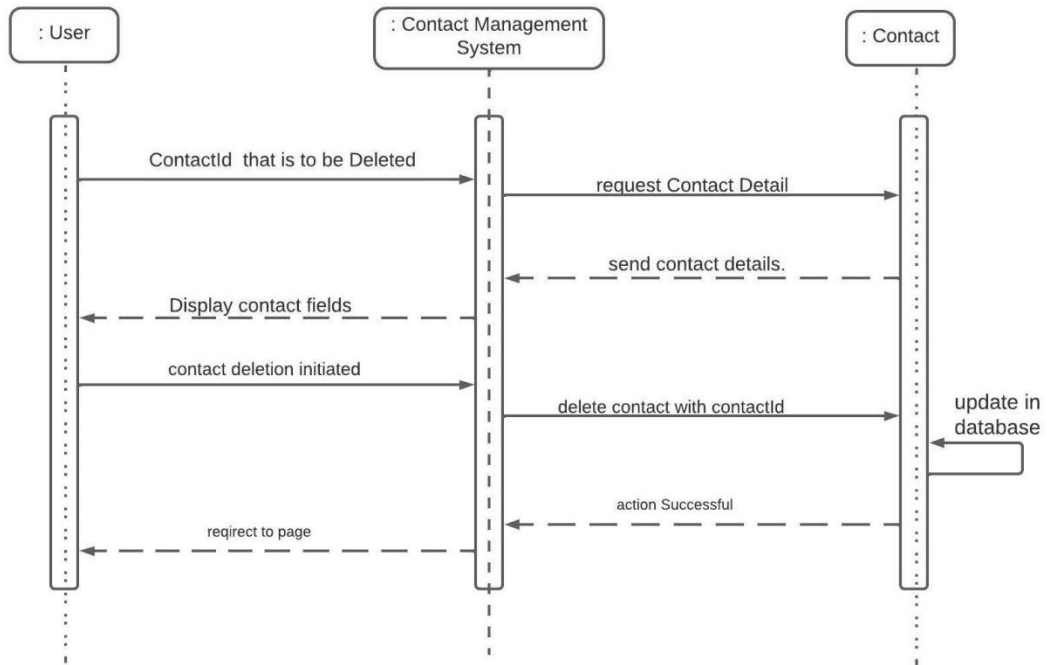
Use case Diagram



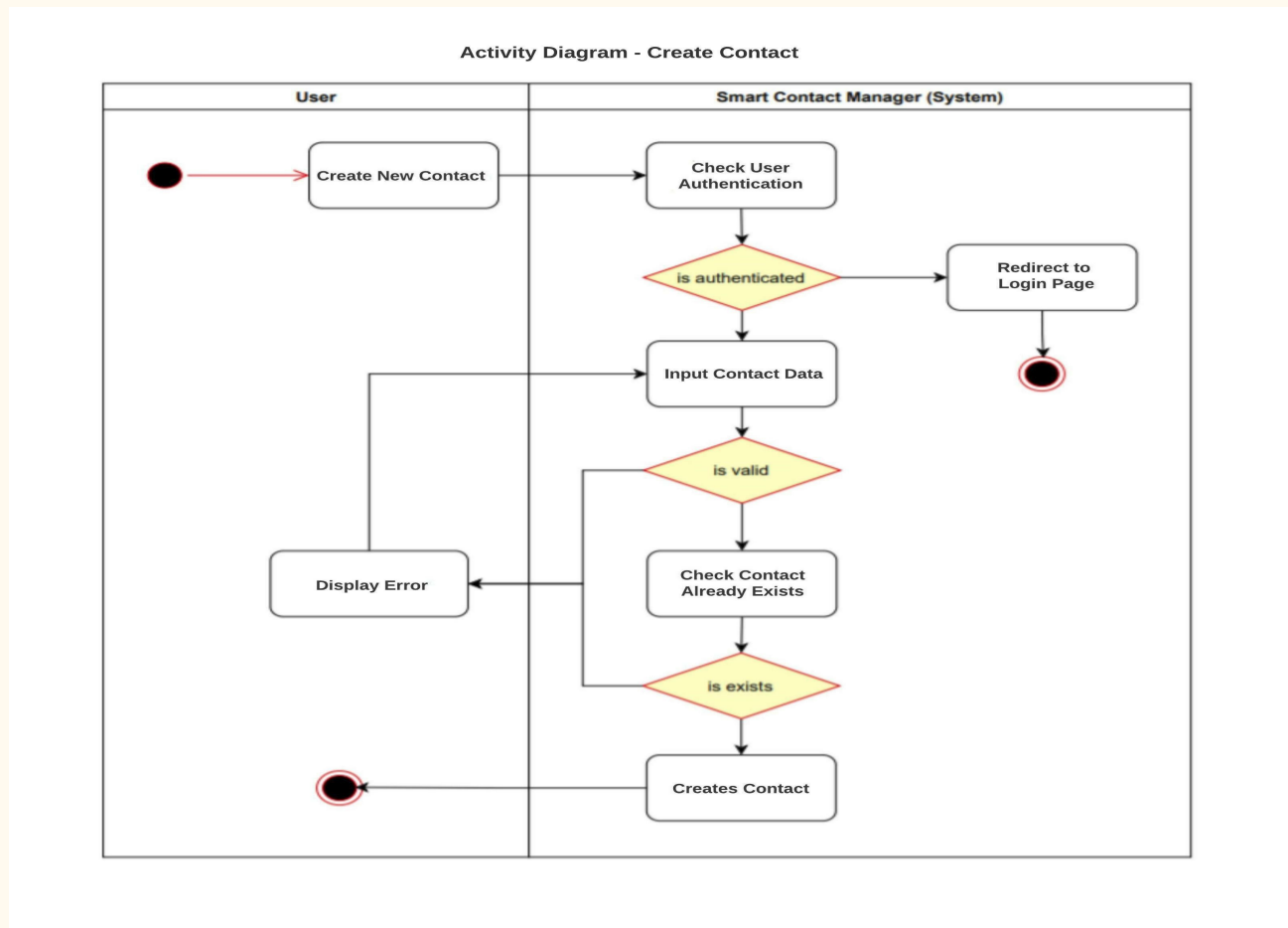
Sequence Diagram



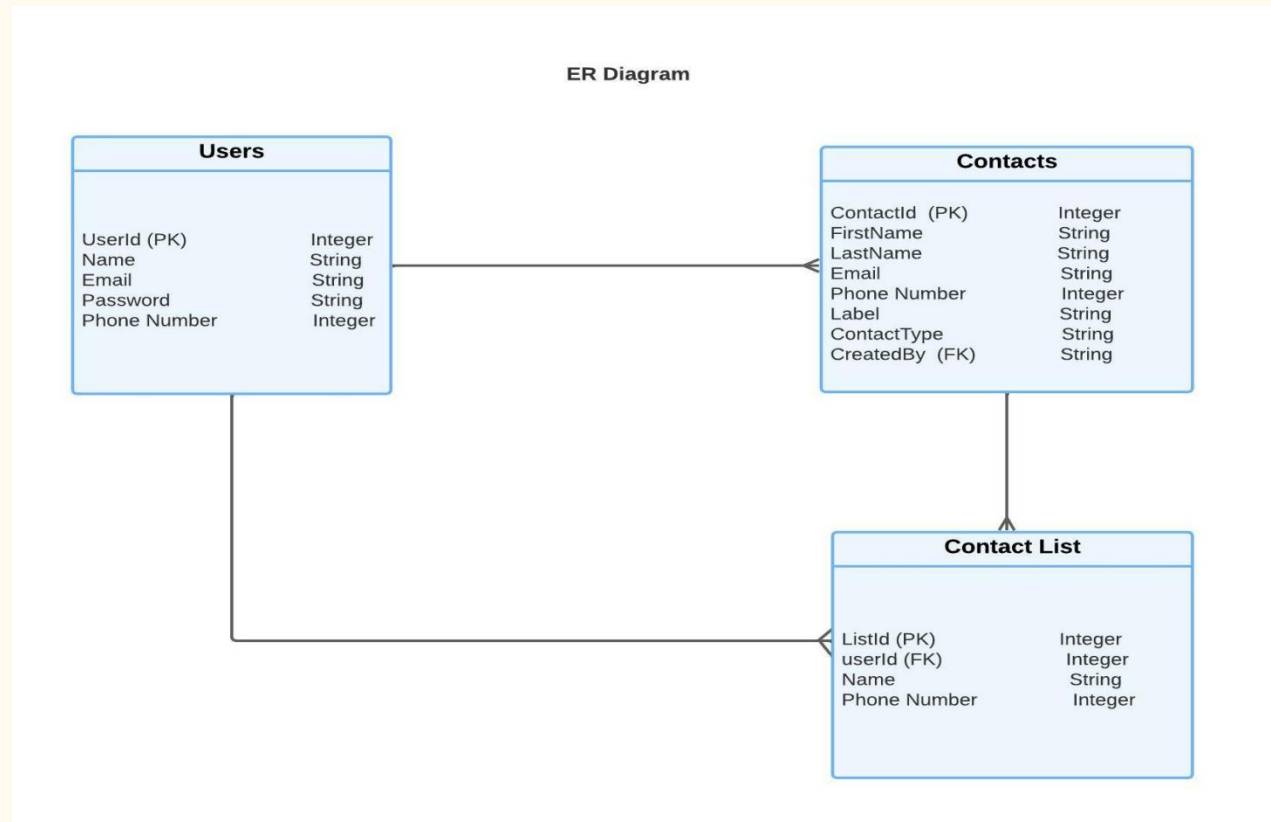
Sequence Diagram Delete Contact



Activity Diagram



ER Diagram



Data Dictionary

Users

No	Field name	Data type	Required	Unique	PK / FK	Ref. Table
1	UserId	int	true	true	PK	-
2	Username	string	true	true	-	-
3	Email	string	true	true	-	-
5	Password	string	true	false	-	-
6	MobileNumber	int	true	false	-	-

Contact

No	Field name	Data type	Required	Unique	PK / FK	Ref. Table
1	ContactId	int	true	true	PK	-
2	First Name	string	true	false	-	-
3	Last Name	string	true	false	-	-
4	Email	string	false	false	-	-
5	MobileNumber	int	true	false	-	-
6	Label	string	false	false	-	-
7	ContactType	string	false	false	-	-
8	CreatedBy	string	true	true	FK	Users

Implementation Detail

Modules

Total modules :- 2

1. Account Module :-

This module manages operations related to users' accounts like login, signup forgot password and reset password. Existing users can login to the system by providing their username and password. New users can create their account by providing the required details.

2. Contact Module :-

This module manages operations related to contacts. Users can add new contacts. They can view a list of existing contacts, detailed information of existing contacts. They can update & delete the contacts as well.

Methods and techniques used

1. ISignup:

```
[ServiceContract]
1 reference
public interface ISignup
{
    [OperationContract]
    1 reference
    bool Signup(user u);
}

[DataContract]
18 references
public class user
{
    0 references
    public int id { get; set; }
    string username = "user1";
    string password = "user1@123";
    string email = "user1@gmail.com";
    int mobileno = 1234567890;

    1 reference
    public user() { }
    0 references
    public user(string uname,string pass,string email,int mobile)
    {
        this.username = uname;
        this.password = pass;
        this.email = email;
        this.mobileno = mobile;
    }

    [DataMember]
    5 references
    public string Username
    {
        get { return username; }
        set { username = value; }
    }

    [DataMember]
    3 references
    public string Password
    {
        get { return password; }
        set { password = value; }
    }

    [DataMember]
    6 references
    public string Email
    {
        get { return email; }
        set { email = value; }
    }

    [DataMember]
    1 reference
    public int Mobileno
    {
        get { return mobileno; }
        set { mobileno = value; }
    }
}
```

Here, user class is defined which has all the required fields like Id, Username, Password, Email and Mobile No. Also, Methods has also been created for each DataMember so as to perform the getter and setter functionality. The Visibility for all the DataMember has been set to public in order to make it accessible across the system.

2. SignupService

```
namespace ContactManagement.User
{
    0 references
    public class SignupService : ISignup
    {
        Model1 m1 = new Model1();
        1 reference
        public bool Signup(user u)
        {
            if (m1.users.Where(se => se.Username == u.Username || se.Email == u.Email).FirstOrDefault<user>() != null)
            {
                return false;
            }
            m1.users.Add(u);
            m1.SaveChanges();
            return true;
        }
    }
}
```

This Service defines object of the Model1. Also, Conditions are validated as to whether the user is already present on our system or not. If not, then the signup is successful and the changes are committed to the database as per the tables created.

3. LoginService

```
0 references
public class LoginService :ILogin
{
    Model1 m1 = new Model1();
    1 reference
    public string Login(string uname, string pass)
    {
        user u1 = m1.users.Where(se => se.Username == uname || se.Email == uname).FirstOrDefault<user>();
        if (u1 != null)
        {
            if (u1.Password == pass && u1.Username == uname)
            {
                return "Login Success";
            }
            return "Invalid Password";
        }
        return "User not exist please signup";
    }
}
```

Here during the login process, the conditions like Username and Password are validated. If there is any mismatch of the parameters, then the login is unsuccessful or else, user is allowed to access further functionalities. Also, on the login page, in case user is unable to login then the user has option to reset the password through email.

4. ForgotService

```
public class ForgotService:IForgot
{
    Model1 m1 = new Model1();
    1 reference
    public string Forgot(string email)
    {
        user u1 = m1.users.Where(se => se.Email == email).FirstOrDefault<user>();
        if (u1 != null)
        {
            try
            {
                string myGUID = Guid.NewGuid().ToString();
                forgot f = new forgot();
                f.Email = email;
                f.Uid = myGUID;
                //f.RegDateTime = new System.DateTime();
                m1.forgots.Add(f);
                m1.SaveChanges();

                string tomail = email;
                string mailbody = "Hi, Click this link to reset you password 
```

Here, if the user initiates the forgot credentials procedure, then the user is redirected to the resert page where he/she is prompted to enter his email in order to get the reset link in inbox. The mail will be sent by the company's own email address custom made for the same and it will have the required steps to guide the user for resetting the Password.

5. IContact

```
[ServiceContract]
1 reference
public interface IContact
{
    [OperationContract]
    1 reference
    bool Contact(contact c);
}

[DataContract]
12 references
public class contact
{
    0 references
    public int id { get; set; }
    string firstname = "abc";
    string lastname = "xyz";
    string email = "user1@gmail.com";
    string label = "mobile";
    string contactType = "work";
    int mobileno = 1234567890;
    string createdby = "user1";

    1 reference
    public contact() { }
    0 references
    public contact(string fname, string lname, string email, string label, string contactType, int mobile, string createdby)
    {
        this.firstname = fname;
        this.lastname = lname;
        this.email = email;
        this.label = label;
        this.contactType = contactType;
        this.mobileno = mobile;
        this.createdby = createdby;
    }
}
```

Here, the interface of the contact is implemented. Here, all the parameters related to the contact information are initialized. Also the public contact method initializes the parameters with the one entered by the user. This will be the exact representation of the data which will be stored in the database for all activities.

6. ResetService

```
0 references
public class ResetService : IReset
{
    Model1 m1 = new Model1();
    1 reference
    public bool Reset(string token, string pass)
    {
        forgot fo = m1.forgots.Where(se => se.Uid == token).FirstOrDefault<forgot>();
        if (fo == null)
        {
            return false;
        }
        user u1 = m1.users.Where(us => us.Email == fo.Email).FirstOrDefault<user>();
        u1.Password = pass;
        m1.SaveChanges();
        return true;
    }
}
```

7. ContactService

```
public class ContactService : IContact
{
    Model1 m1 = new Model1();
    1 reference
    public bool Contact(contact c)
    {
        if (m1.contact.Where(se => se.Email == c.Email || se.Mobileno == c.Mobileno).FirstOrDefault<contact>() != null)
        {
            return false;
        }
        m1.contact.Add(c);
        m1.SaveChanges();
        return true;
    }
}
```

Testing

Unit testing of each module was done after successfully completing the module. Each module was tested individually before integrating them with the whole system.

After integrating each module with the system, integration testing was done in order to check if modules are working properly together.

After completing all integrations, black-box testing of the whole system was carried out to ensure the system works in a correct manner.

Black box testing of Major functions of the system

1. Log in to the system.

Case 1 : Invalid Username or password entered by the user.

Output : Error message on the screen saying “Invalid credentials”

Case 2 : Valid credentials.

Output : The user is redirected to the Home page.

2. Edit Contact

Case 1 : Contact number already exists.

Output : Error message on the screen saying “Contact already exists”

Case 2 : Some of required fields missing in input.

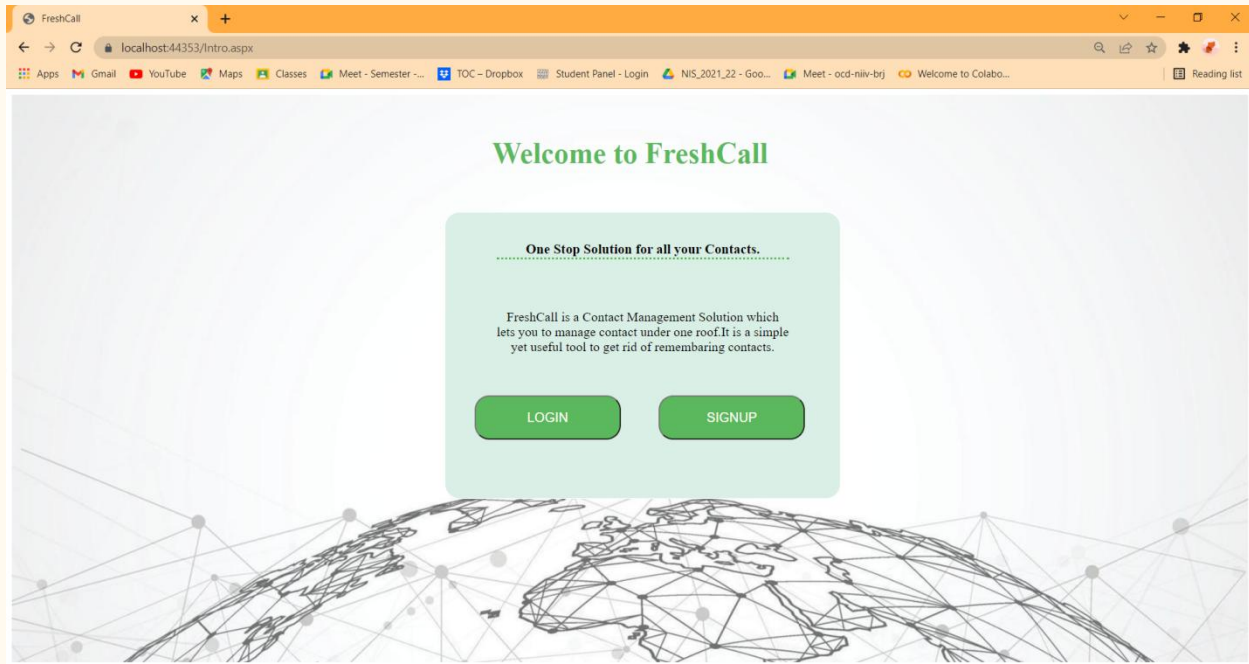
Output : Model validation errors will be displayed to the user.

Case 3 : All input data are valid.

Output : Contact updated successfully.

Screenshots

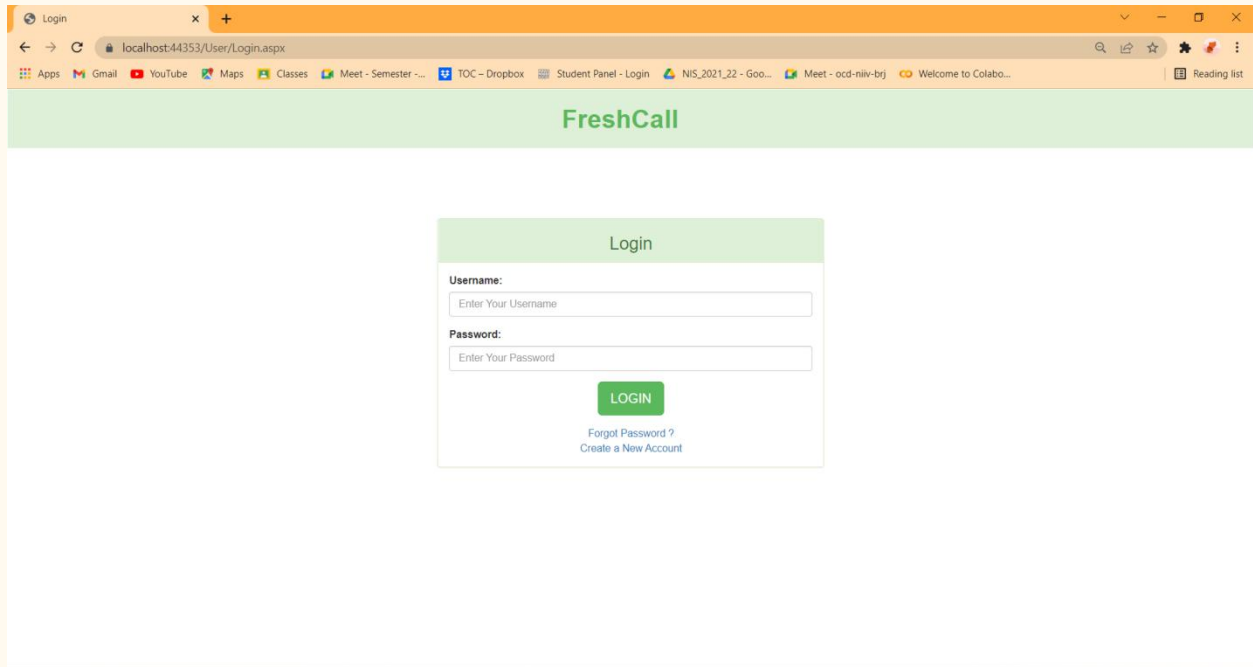
1. Intro



2. Signup

A screenshot of a web browser displaying the FreshCall signup page. The browser's address bar shows 'localhost:44353/User/Signup.aspx'. The page has a light green header with the 'FreshCall' logo. Below the header is a white box titled 'Register' containing a form with the following fields: 'Username:' (with placeholder 'Enter Your Username'), 'Email:' (with placeholder 'Enter Your Email'), 'Password:' (with placeholder 'Enter Your Password'), 'Confirm Password:' (with placeholder 'Re Enter Your Password'), and 'Mobile No.:' (with placeholder 'Enter Your Mobile Number'). A green 'SignUp' button is at the bottom of the form, and a link 'Already Have an Account' is below it.

3. Login



The screenshot shows a web browser window with the address bar displaying 'localhost:44353/User/Login.aspx'. The browser's tab is titled 'Login'. The page features a green header with the 'FreshCall' logo. The main content area is white and contains a centered login form. The form has a green title bar labeled 'Login'. Below the title bar, there are two input fields: 'Username:' with a placeholder 'Enter Your Username' and 'Password:' with a placeholder 'Enter Your Password'. A green 'LOGIN' button is positioned below the password field. At the bottom of the form, there are two links: 'Forgot Password ?' and 'Create a New Account'.

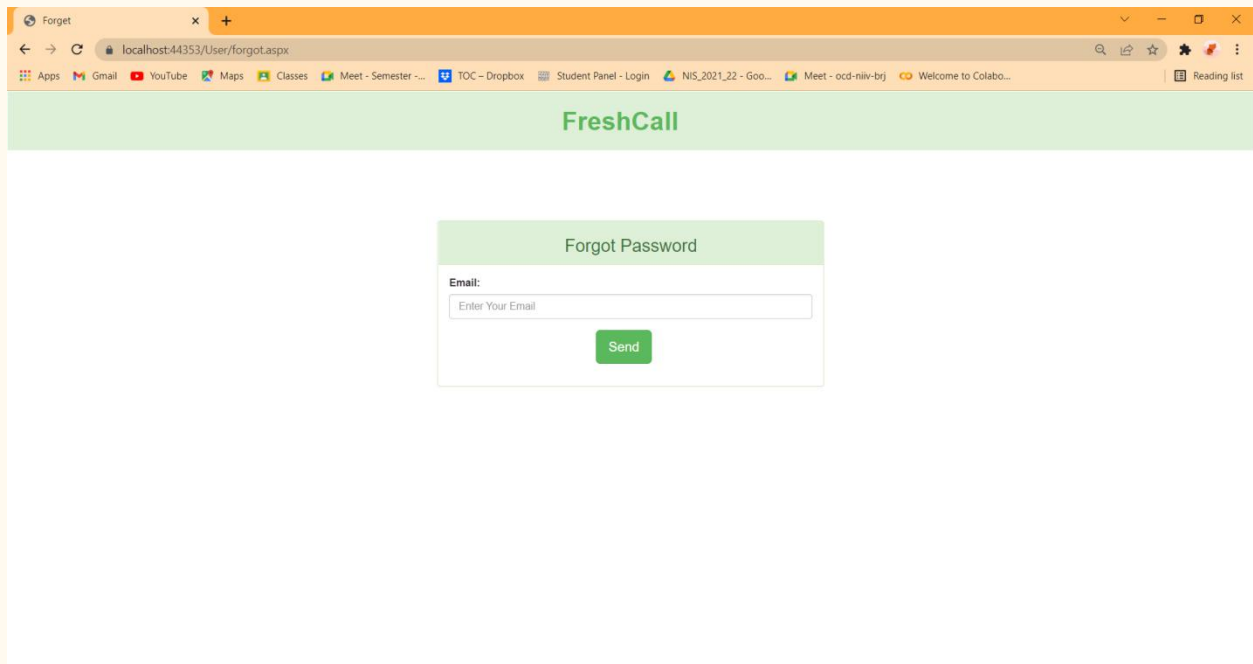
Username:
Enter Your Username

Password:
Enter Your Password

LOGIN

[Forgot Password ?](#)
[Create a New Account](#)

4. Forgot Password



The screenshot shows a web browser window with the address bar displaying 'localhost:44353/User/forgot.aspx'. The browser's tab is titled 'Forgot'. The page features a green header with the 'FreshCall' logo. The main content area is white and contains a centered 'Forgot Password' form. The form has a green title bar labeled 'Forgot Password'. Below the title bar, there is an 'Email:' label followed by an input field with the placeholder 'Enter Your Email'. A green 'Send' button is located below the input field.

Email:
Enter Your Email

Send

5. Reset Password

Reset Password

Password:

Enter Your Username

Confirm Password:

Enter Your Password

Reset

6. Home

Welcome to FreshCall

One Stop Solution for all your Contacts.

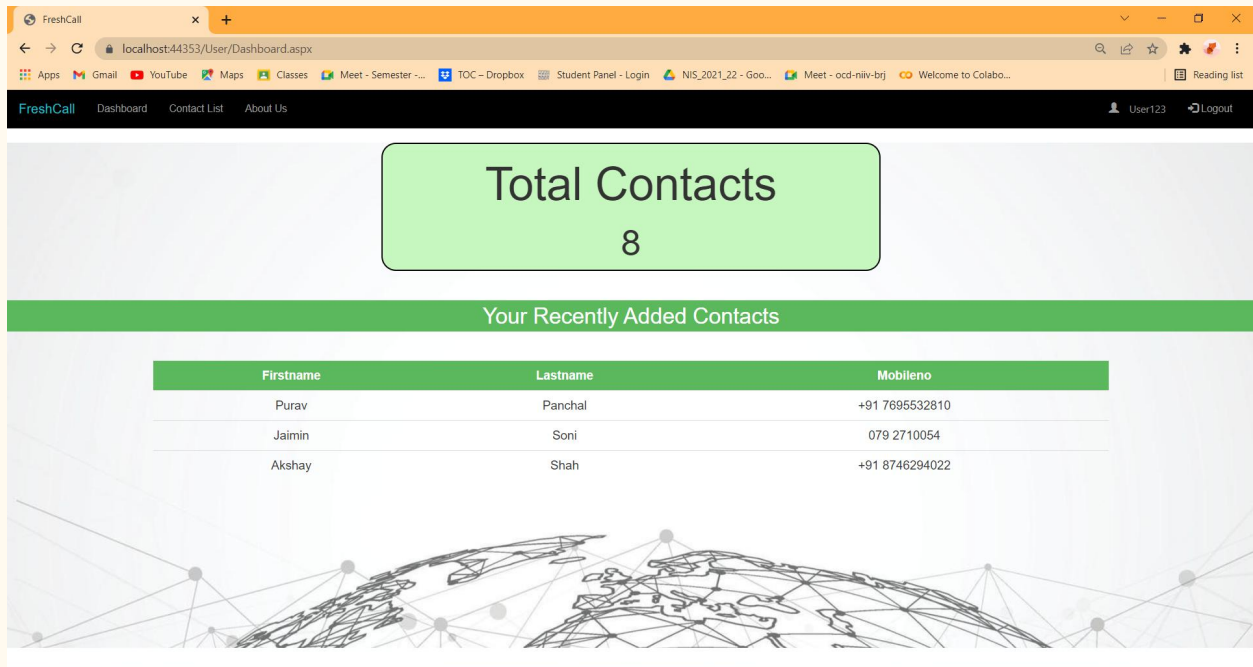
FreshCall is a Contact Management Solution which lets you to manage contact under one roof. It is a simple yet useful tool to get rid of remembering contacts.

Go to DashBoard

Our Services

View Contacts	Add Contacts	Edit Contacts	Delete Contacts
One Place to access all your contact list be it your friend, family or bussiness.	FreshCall allows you to add contact list be it your friend, family or bussiness.	In case of error in contact details changes can be made in contact list.	FreshCall allows you to delete one or many contacts, if there are no further needed.

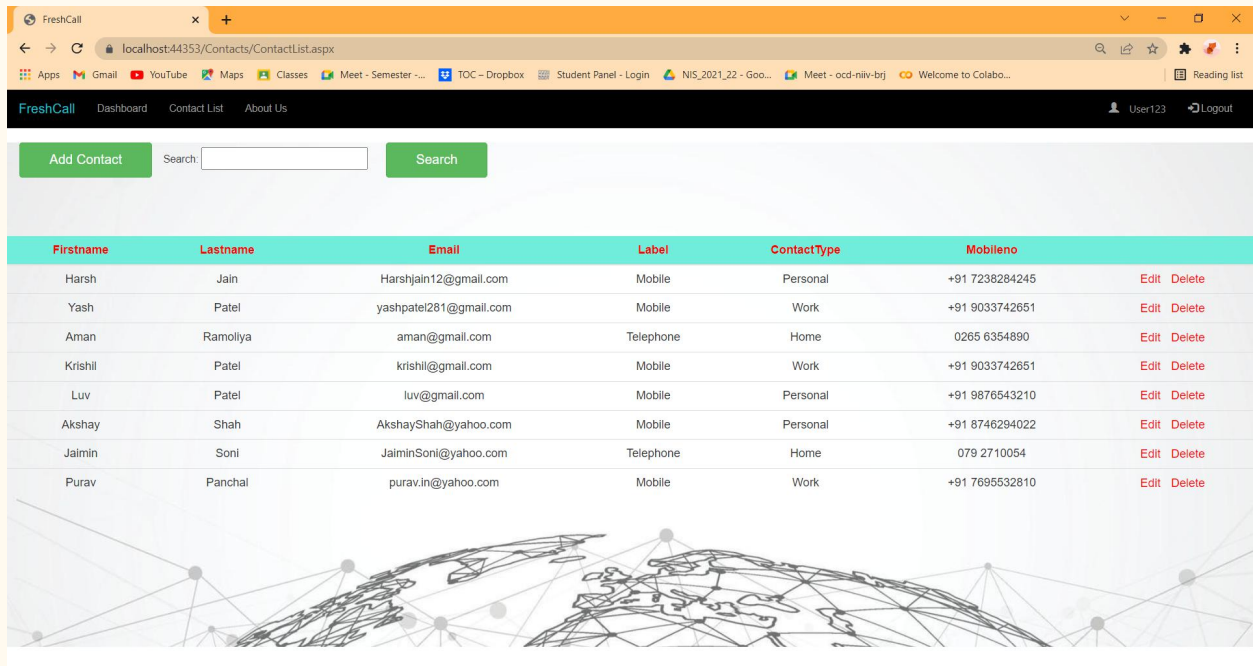
7. Dashboard



The screenshot shows the FreshCall Dashboard. At the top, there's a navigation bar with 'FreshCall', 'Dashboard', 'Contact List', and 'About Us'. The main content area features a large green box displaying 'Total Contacts 8'. Below this, a section titled 'Your Recently Added Contacts' contains a table with three columns: Firstname, Lastname, and Mobileneno. The table lists three contacts: Purav Panchal (+91 7695532810), Jaimin Soni (079 2710054), and Akshay Shah (+91 8746294022). The background of the dashboard has a subtle network diagram pattern.

Firstname	Lastname	Mobileneno
Purav	Panchal	+91 7695532810
Jaimin	Soni	079 2710054
Akshay	Shah	+91 8746294022

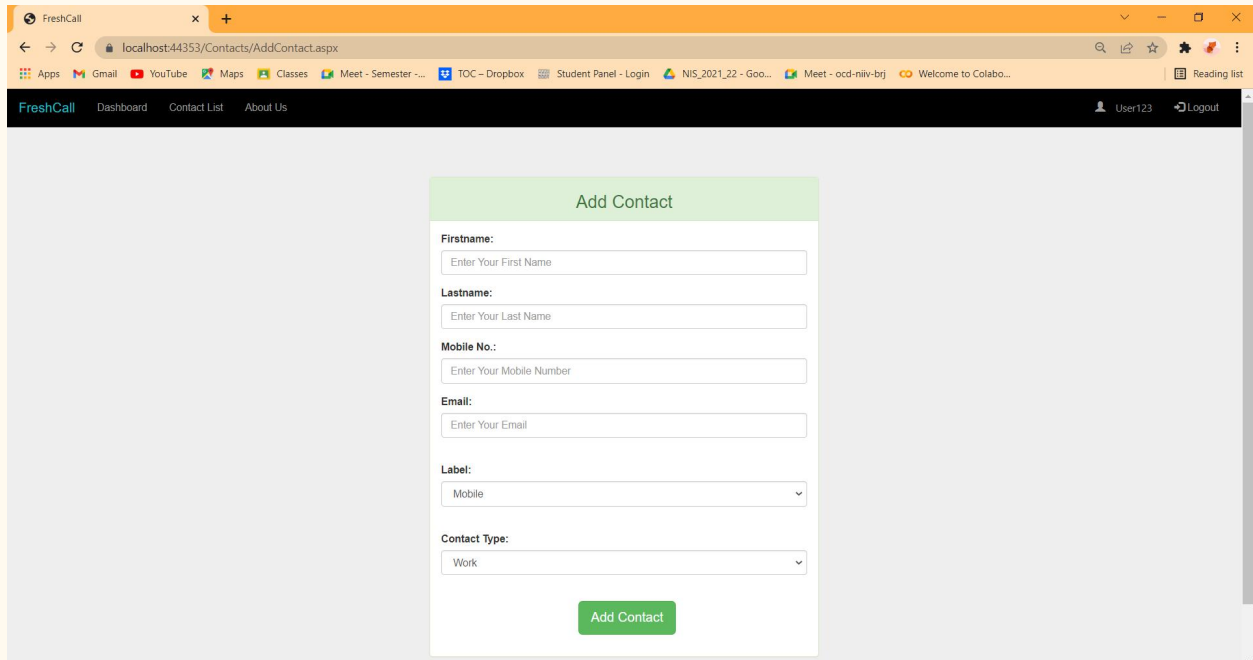
8. Contact List



The screenshot shows the FreshCall Contact List page. At the top, there's a navigation bar with 'FreshCall', 'Dashboard', 'Contact List', and 'About Us'. Below the navigation bar, there's a section with 'Add Contact' and a search bar. The main content area features a table with six columns: Firstname, Lastname, Email, Label, ContactType, and Mobileneno. The table lists eight contacts, each with an 'Edit' and 'Delete' link. The background of the contact list page has a subtle network diagram pattern.

Firstname	Lastname	Email	Label	ContactType	Mobileneno	Edit	Delete
Harsh	Jain	Harshjain12@gmail.com	Mobile	Personal	+91 7238284245	Edit	Delete
Yash	Patel	yashpatel281@gmail.com	Mobile	Work	+91 9033742651	Edit	Delete
Aman	Ramollya	aman@gmail.com	Telephone	Home	0265 6354890	Edit	Delete
Krishil	Patel	krishil@gmail.com	Mobile	Work	+91 9033742651	Edit	Delete
Luv	Patel	luv@gmail.com	Mobile	Personal	+91 9876543210	Edit	Delete
Akshay	Shah	AkshayShah@yahoo.com	Mobile	Personal	+91 8746294022	Edit	Delete
Jaimin	Soni	JaiminSoni@yahoo.com	Telephone	Home	079 2710054	Edit	Delete
Purav	Panchal	purav.in@yahoo.com	Mobile	Work	+91 7695532810	Edit	Delete

9. Add Contact

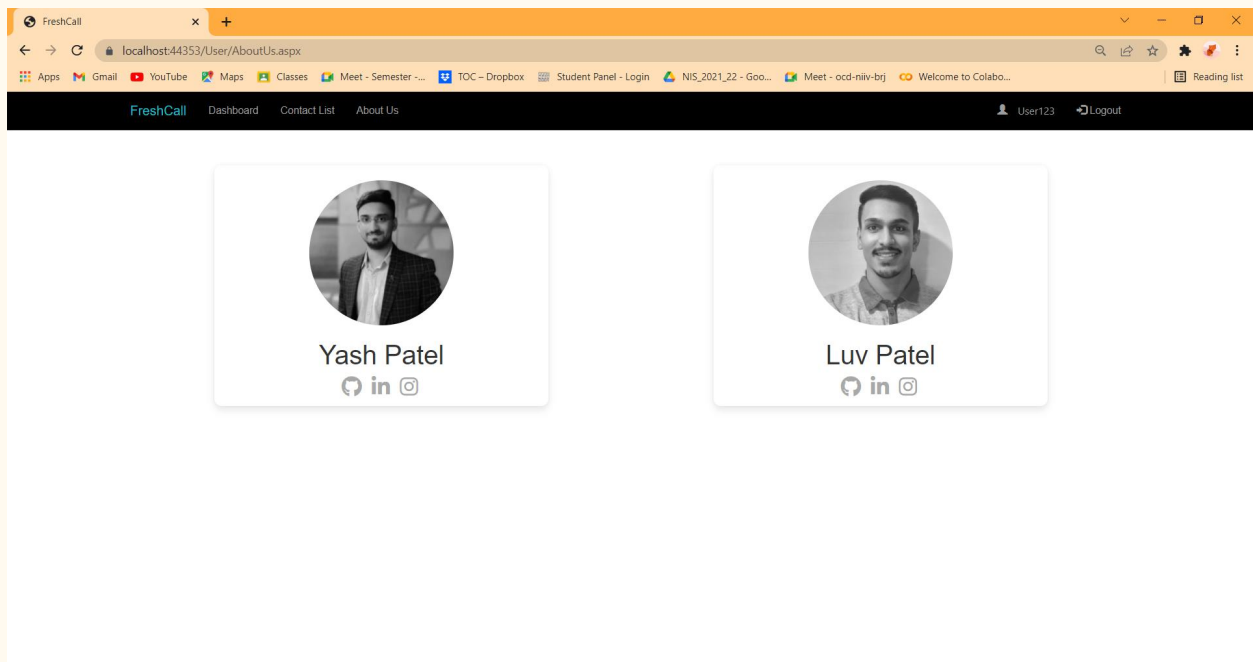


The screenshot shows a web browser window with the URL `localhost:44353/Contacts/AddContact.aspx`. The page title is "FreshCall". The navigation bar includes "Dashboard", "Contact List", and "About Us". The user is logged in as "User123" and can click "Logout". The main content area features a form titled "Add Contact" with the following fields:

- Firstname:**
- Lastname:**
- Mobile No.:**
- Email:**
- Label:**
- Contact Type:**

A green "Add Contact" button is located at the bottom of the form.

10.About Us



Conclusion

Functionalities implemented successfully

- Login by user.
- Registration by user.
- Forgot Password mail send
- Reset Password
- Create contact
- Update contact details
- View contacts list
- View contact details
- Delete contact
- Logout from the system

Limitation and Future Extensions

● Limitations

- Update profile
- Import contacts
- Export contacts

● Possible future extensions

- Users can upload their profile picture & update profile.
- Users can add the contacts by Importing contacts as .vcf file and can export the contacts in .vcf file.

Bibliography

To build this project we have taken references from following websites,

- Frameworks used for development :-
<https://docs.microsoft.com/en-us/dotnet/framework/wcf/> (For backend service)
<https://docs.microsoft.com/en-us/dotnet/framework/wcf/> (For frontend)
- CSS framework for frontend designing :-
<https://getbootstrap.com/>
- For debugging :-
<https://stackoverflow.com/>