K.S.RANGASAMY COLLEGE OF TECHNOLOGY
(Autonomous)
TIRUCHENGODE-637215

A MINI PROJECT REPORT

CONTACT MANAGEMENT SYSTEM

60 IT L04 – C# and .NET FRAMEWORK

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

ROHITH S (73772214187)

SABAREESH K S (73772214188)

SAKTHI R (73772214189)

K.S.RANGASAMY COLLEGE OF TECHNOLOGY

(Autonomous)
TIRUCHENGODE-637215

BONAFIDE CERTIFICATE

Certified that this project report titled "**CONTACT MANAGEMENT SYSTEM**" is the bonafide work of **SABAREESH K S (73772214188), SAKTHI R (73772214189),** who carried out the project under my guidance.

ABSTRACT

The **Contact Management System** is a web-based application developed using the **ASP.NET Core MVC framework**, designed to efficiently handle the organization and management of contact information. The project incorporates essential CRUD (Create, Read, Update, Delete) functionalities, enabling users to seamlessly add, view, edit, and delete contacts. Each contact includes key details such as the First Name, Last Name, Contact Number, and Email Address, providing a structured way to store and retrieve information.

The application leverages the powerful scaffolding feature of ASP.NET Core MVC to automate the creation of controllers, views, and models for CRUD operations. This approach not only accelerates the development process but also ensures a consistent and maintainable codebase. With a dynamic and responsive user interface built using Razor Pages, the system offers an intuitive experience that caters to both personal and professional contact management needs.

A robust backend is implemented using **Entity Framework Core** for seamless interaction with the **SQL Server database**, where all contact information is stored. The database is designed to include a Contacts table with fields such as Id, FirstName, LastName, MobileNumber, and Email. Additionally, the system includes built-in data validation mechanisms to ensure the integrity and correctness of user inputs, such as proper email formatting and mandatory fields.

The modular architecture of the Contact Management System makes it highly scalable, allowing for future enhancements and integration of additional features. Its design emphasizes simplicity and practicality, making it suitable for individuals or businesses that require a reliable and user-friendly solution for managing contacts. Overall, this project exemplifies the practical application of **ASP.NET Core MVC** in creating real-world web applications, demonstrating its capability to deliver efficient and scalable solutions.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The **Contact Management System** provides a user-friendly interface to manage contacts efficiently. It offers a structured navigation system and pages tailored for specific operations such as creating, editing, viewing, and deleting contact information. Below is an overview of the functionality provided on each page:

- **Index Page**:
  The central hub of the application where all contacts are listed. From here, users can create a new contact, edit existing entries, delete contacts, or view detailed information about a selected contact.
- **Create Page**:
  This page provides a form for adding a new contact. Users can input essential details such as the First Name, Last Name, Contact Number, and Email Address. Upon saving, the contact is added to the list on the index page. A "Go Back to List" option is also available for easy navigation.
- **Edit Page**:
  The edit page allows users to modify the details of a selected contact. The form is pre-filled with the current information of the chosen contact, enabling users to make necessary changes. After saving the updates, users can return to the index page to see the revised list.
- **Details Page**:
  This page displays all the details of a selected contact in a read-only format. Users can edit the contact directly from this page or navigate back to the index page. This ensures flexibility and quick access to editing options.
- **Delete Page**:
  The delete page presents the details of the selected contact and asks for confirmation with the message: *"Are you sure you want to delete this contact?"*. Upon confirmation, the contact is removed from the database, and users are redirected back to the index page.

This well-structured navigation and feature-rich interface ensure that managing contacts is an intuitive and seamless process for users. The application emphasizes ease of use while maintaining robust functionality for all operations.

CHAPTER 2

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

Contact Management

- Provide functionality to create, edit, and delete contact records with fields such as First Name, Last Name, Contact Number, and Email Address.
- Display all contacts in a searchable and sortable list on the index page.
- Ensure seamless navigation between different pages, including Create, Edit, Details, and Delete pages.

Create Contact

- Allow users to fill out a form with the required contact details, including validations for mandatory fields and proper formatting (e.g., valid email).
- Provide an option to navigate back to the index page without saving changes.

Edit Contact

- Pre-fill the form with existing details of the selected contact, enabling users to update information.
- Save updated information and navigate back to the index page to view the changes.

View Contact Details

- Display the full details of a selected contact in a read-only format.
- Provide an option to edit the contact directly from the details page.
- Allow navigation back to the index page.

Delete Contact

- Display a confirmation prompt before deleting a selected contact, asking: *"Are you sure you want to delete this contact?"*.
- Remove the contact upon confirmation and redirect back to the index page.

NON-FUNCTIONAL REQUIREMENTS

User Interface (UI) and User Experience (UX)

- Design a responsive, clean, and intuitive interface compatible with various devices (desktop, tablet, mobile).
- Ensure consistent styling and clear navigation across all pages to minimize the learning curve for users.

Performance and Scalability

- Support fast data retrieval and updates for efficient management of contacts, even with large datasets.
- Ensure that the system can handle a high number of simultaneous users without performance degradation.

Data Management and Security

- Store contact details securely in a relational database using SQL Server, ensuring data integrity and consistency.
- Implement basic input validations and prevent SQL injection or other vulnerabilities by using best practices in coding and data handling.
- Regularly back up the database to prevent data loss.

System Reliability and Availability

- Design the system to minimize downtime, ensuring continuous availability for users.
- Implement error handling and logging mechanisms to diagnose and resolve issues promptly.
- Ensure reliable navigation between pages, even during network disruptions.

Compliance and Accessibility

- Adhere to data protection guidelines, such as encrypting sensitive data like contact details.
- Follow accessibility standards like WCAG to make the application usable for individuals with disabilities.

CHAPTER 3

WORK FLOW

The design of the **Contact Management System** ensures a clear structure for managing contacts, supported by a well-defined flow and class relationships. Below are detailed explanations along with the workflow.

WORK FLOW

1. **Index Page:**
    - Displays all contacts with options to create, edit, view, or delete.
2. **Create Contact:**
    - User fills out a form on the Create Page to add a new contact.
    - Validates inputs and redirects back to the index page upon successful creation.
3. **Edit Contact:**
    - User selects Edit on a contact, updates the pre-filled form, and saves changes.
    - Redirects back to the index page after updating.
4. **View Contact Details:**
    - Displays full details of a contact on the Details Page in read-only format.
    - Options to edit or return to the index page.
5. **Delete Contact:**
    - User confirms deletion on the Delete Page, and the contact is removed.
    - Redirects back to the index page after deletion.
6. **Validation & Security:**
    - Input fields are validated for correctness.
    - Secure database operations ensure data integrity and prevent vulnerabilities.

The system ensures seamless navigation between operations, maintaining an intuitive user experience.

CHAPTER 4
IMPLEMENTATION

CODE:
CONTACTS CLASS (Contacts.cs):

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace ContactManagement.Models
{
   [Table("Contacts", Schema = "dbo")]
   public class Contacts
   {
      [Key]
      public int Id { get; set; }

      [Required(ErrorMessage = "First Name is required.")]
      [StringLength(50)]
      public required string FirstName { get; set; }

      [Required(ErrorMessage = "Last Name is required.")]
      [StringLength(50)]
      public required string LastName { get; set; }

      [Required(ErrorMessage = "Mobile Number is required.")]
      [Phone]
      [StringLength(15, ErrorMessage = "Mobile Number must be exactly 10 digits.")]
      [RegularExpression(@"^\d{10}$", ErrorMessage = "Mobile Number must be 10 digits.")]
      public required string MobileNumber { get; set; }

      [EmailAddress(ErrorMessage = "Invalid Email Address.")]
      [StringLength(100)]
      [RegularExpression(@"^[^@\s]+@[^@\s]+\.[^@\s]+$", ErrorMessage = "Invalid Email Address.")]

      public required string Email { get; set; }
```

```
    }
 }
```

INDEX PAGE (Index.cshtml):

```
@model IEnumerable<ContactManagement.Models.Contacts>

@{
   ViewData["Title"] = "Index";
}
<div class="container">
   <h1 class="my-4">Contacts</h1>

   <p>
      <a class="btn btn-success" asp-action="Create">Create New</a>
   </p>
   <table class="table table-striped table-bordered">
      <thead class="thead-dark">
         <tr>
            <th>
               @Html.DisplayNameFor(model => model.FirstName)
            </th>
            <th>
               @Html.DisplayNameFor(model => model.LastName)
            </th>
            <th>
               @Html.DisplayNameFor(model => model.MobileNumber)
            </th>
            <th>
               @Html.DisplayNameFor(model => model.Email)
            </th>
            <th></th>
         </tr>
      </thead>
      <tbody>
      @foreach (var item in Model) {
         <tr>
            <td>
               @Html.DisplayFor(modelItem => item.FirstName)
            </td>
            <td>
               @Html.DisplayFor(modelItem => item.LastName)
            </td>
            <td>
               @Html.DisplayFor(modelItem => item.MobileNumber)
```

```
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                <a class="btn btn-primary btn-sm" asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                <a class="btn btn-info btn-sm" asp-action="Details" asp-route-id="@item.Id">Details</a> |
                <a class="btn btn-danger btn-sm" asp-action="Delete" asp-route-id="@item.Id">Delete</a>
            </td>
        </tr>
    }
    </tbody>
</table>
</div>
```

CREATE PAGE (Create.cshtml):

```
@model ContactManagement.Models.Contacts

@{
    ViewData["Title"] = "Create Contact";
}

<div class="container">
    <h1 class="text-center my-4">Create New Contact</h1>

    <div class="card shadow">
        <div class="card-header bg-primary text-white">
            <h4 class="mb-0">Contact Details</h4>
        </div>
        <div class="card-body">
            <form asp-action="Create">
                <div asp-validation-summary="ModelOnly" class="alert alert-danger"></div>

                <div class="mb-3">
                    <label asp-for="FirstName" class="form-label fw-bold"></label>
                    <input asp-for="FirstName" class="form-control" />
                    <span asp-validation-for="FirstName" class="text-danger"></span>
                </div>

                <div class="mb-3">
                    <label asp-for="LastName" class="form-label fw-bold"></label>
                    <input asp-for="LastName" class="form-control" />
                    <span asp-validation-for="LastName" class="text-danger"></span>
                </div>
```

```
            <div class="mb-3">
                <label asp-for="MobileNumber" class="form-label fw-bold"></label>
                <input asp-for="MobileNumber" class="form-control" />
                <span asp-validation-for="MobileNumber" class="text-danger"></span>
            </div>

            <div class="mb-3">
                <label asp-for="Email" class="form-label fw-bold"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>

            <div class="d-flex justify-content-between">
                <input type="submit" value="Create" class="btn btn-success" />
                <a asp-action="Index" class="btn btn-secondary">Back to List</a>
            </div>
        </form>
    </div>
  </div>
</div>

@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

DELETE PAGE (Delete.cshtml):

```
@{
  ViewData["Title"] = "Delete";
}

<div class="container my-5">
  <div class="card shadow-lg">
    <div class="card-header bg-danger text-white">
      <h3 class="mb-0">Delete Contact</h3>
    </div>
    <div class="card-body">
      <h4 class="text-center text-danger mb-4">Are you sure you want to delete this contact?</h4>
      <dl class="row">
        <dt class="col-sm-4 fw-bold">First Name</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.FirstName)</dd>

        <dt class="col-sm-4 fw-bold">Last Name</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.LastName)</dd>

        <dt class="col-sm-4 fw-bold">Mobile Number</dt>
```

```
            <dd class="col-sm-8">@Html.DisplayFor(model => model.MobileNumber)</dd>

            <dt class="col-sm-4 fw-bold">Email</dt>
            <dd class="col-sm-8">@Html.DisplayFor(model => model.Email)</dd>
        </dl>
    </div>
    <div class="card-footer d-flex justify-content-between">
        <form asp-action="Delete" method="post">
            <input type="hidden" asp-for="Id" />
            <button type="submit" class="btn btn-danger">Delete</button>
        </form>
        <a asp-action="Index" class="btn btn-secondary">Back to List</a>
    </div>
  </div>
</div>
```

DETAILS PAGE (Details.cshtml):

```
@model ContactManagement.Models.Contacts

@{
  ViewData["Title"] = "Details";
}

<div class="container my-5">
  <h1 class="text-center mb-4">Contact Details</h1>
  <div class="card shadow-lg">
    <div class="card-header bg-success text-white">
      <h4 class="mb-0">View Contact</h4>
    </div>
    <div class="card-body">
      <dl class="row">
        <dt class="col-sm-4 fw-bold">First Name</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.FirstName)</dd>

        <dt class="col-sm-4 fw-bold">Last Name</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.LastName)</dd>

        <dt class="col-sm-4 fw-bold">Mobile Number</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.MobileNumber)</dd>

        <dt class="col-sm-4 fw-bold">Email</dt>
        <dd class="col-sm-8">@Html.DisplayFor(model => model.Email)</dd>
      </dl>
    </div>
    <div class="card-footer d-flex justify-content-between">
```

```
            <a asp-action="Edit" asp-route-id="@Model?.Id" class="btn btn-primary">Edit</a>
            <a asp-action="Index" class="btn btn-secondary">Back to List</a>
        </div>
    </div>
</div>
```

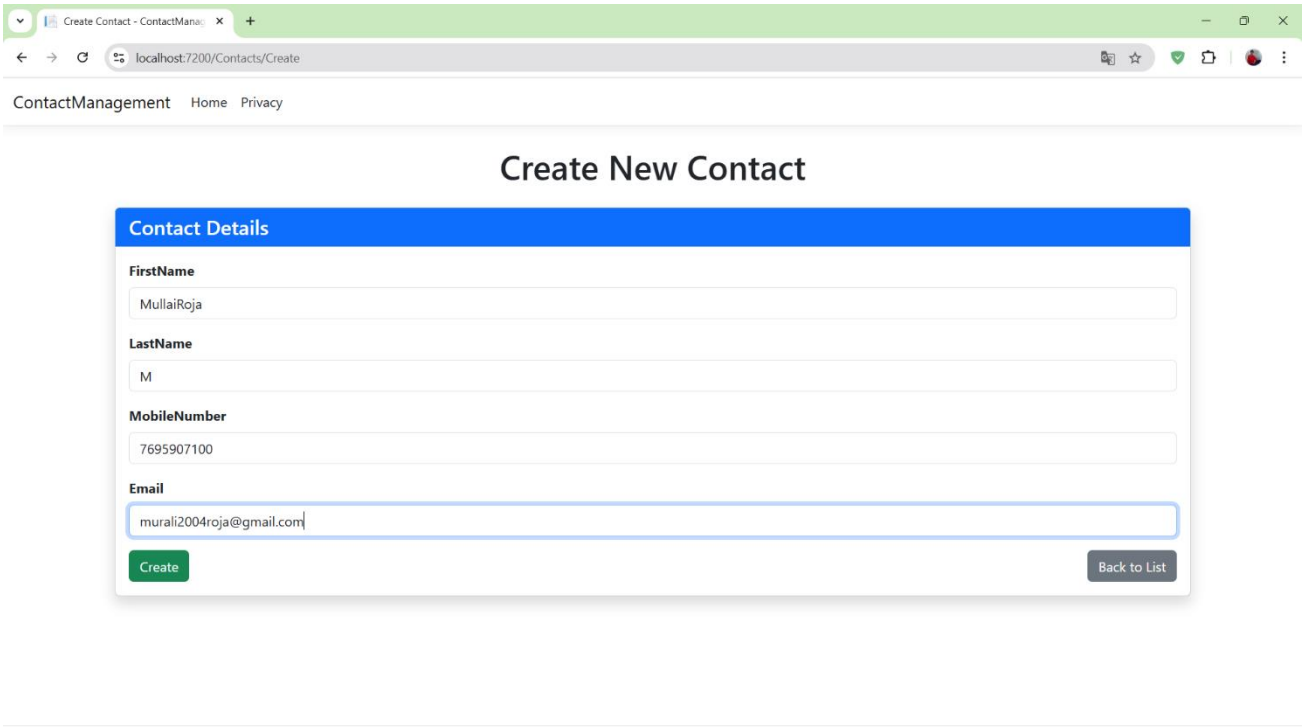
EDIT PAGE (Edit.cshtml):

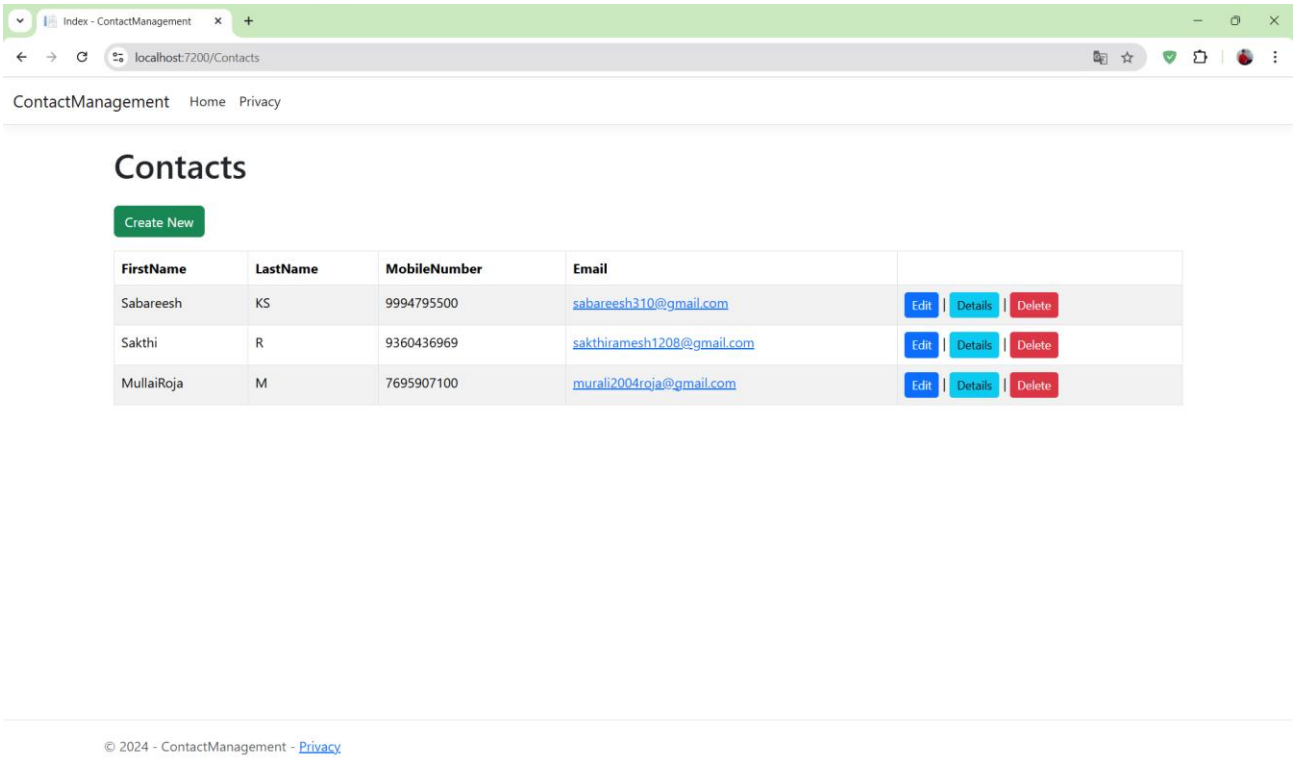
```
@model ContactManagement.Models.Contacts

@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>

<h4>Contacts</h4>
<hr />
<div class="container my-5">
    <h1 class="text-center mb-4">Edit Contact</h1>
    <div class="card shadow-lg">
        <div class="card-header bg-info text-white">
            <h4 class="mb-0">Update Contact Details</h4>
        </div>
        <div class="card-body">
            <form asp-action="Edit" method="post">
                <div asp-validation-summary="ModelOnly" class="alert alert-danger"></div>
                <input type="hidden" asp-for="Id" />
                <div class="mb-3">
                    <label asp-for="FirstName" class="form-label fw-bold"></label>
                    <input asp-for="FirstName" class="form-control" placeholder="Enter First Name" />
                    <span asp-validation-for="FirstName" class="text-danger"></span>
                </div>
                <div class="mb-3">
                    <label asp-for="LastName" class="form-label fw-bold"></label>
                    <input asp-for="LastName" class="form-control" placeholder="Enter Last Name" />
                    <span asp-validation-for="LastName" class="text-danger"></span>
                </div>
                <div class="mb-3">
                    <label asp-for="MobileNumber" class="form-label fw-bold"></label>
                    <input asp-for="MobileNumber" class="form-control" placeholder="Enter Mobile Number"
/>
                    <span asp-validation-for="MobileNumber" class="text-danger"></span>
                </div>
                <div class="mb-3">
```

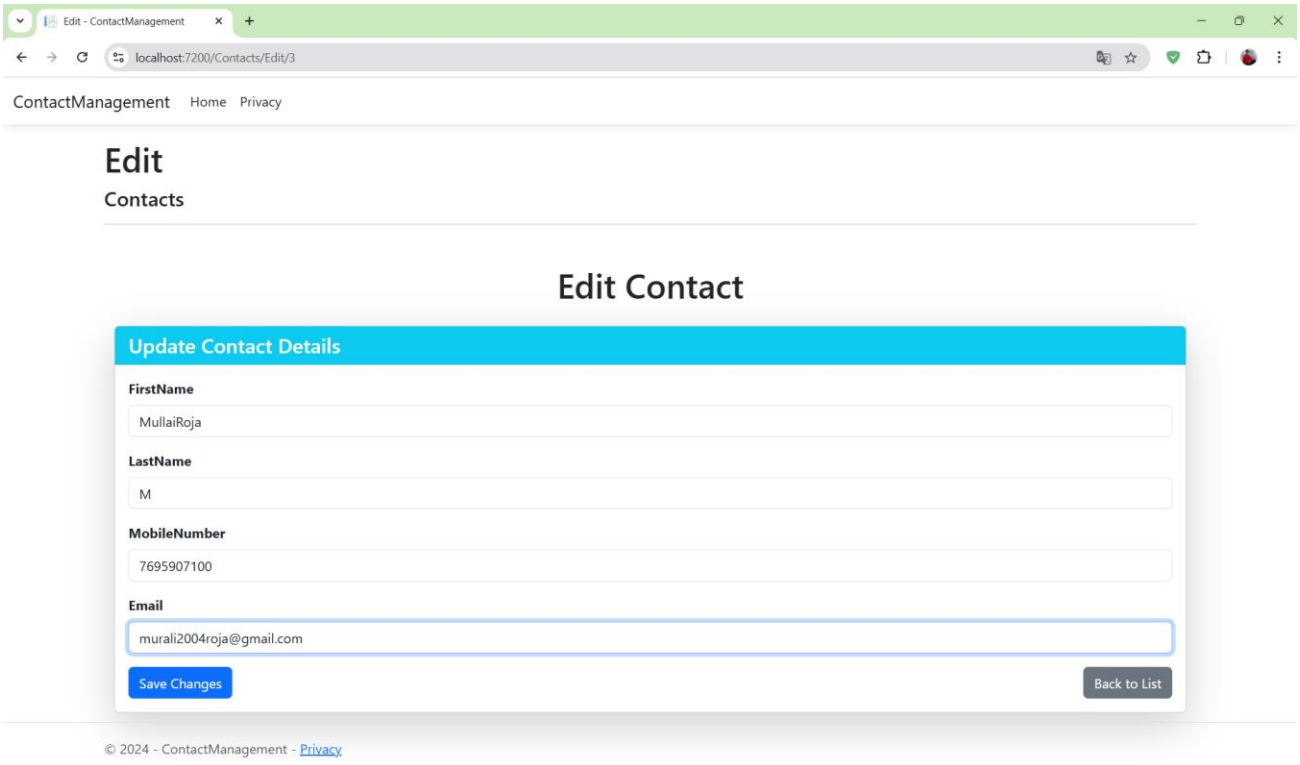
```
                <label asp-for="Email" class="form-label fw-bold"></label>
                <input asp-for="Email" class="form-control" placeholder="Enter Email Address" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="d-flex justify-content-between">
                <input type="submit" value="Save Changes" class="btn btn-primary" />
                <a asp-action="Index" class="btn btn-secondary">Back to List</a>
            </div>
        </form>
    </div>
  </div>
</div>
@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

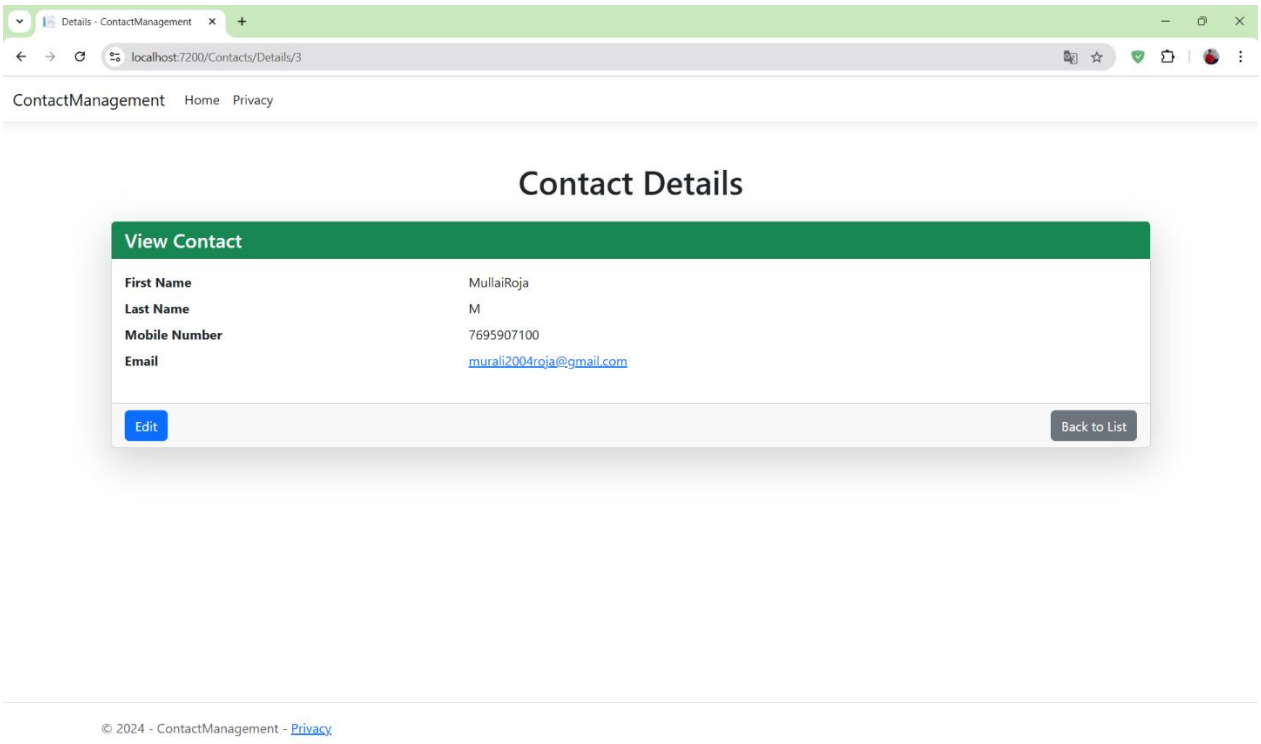
CHAPTER 5

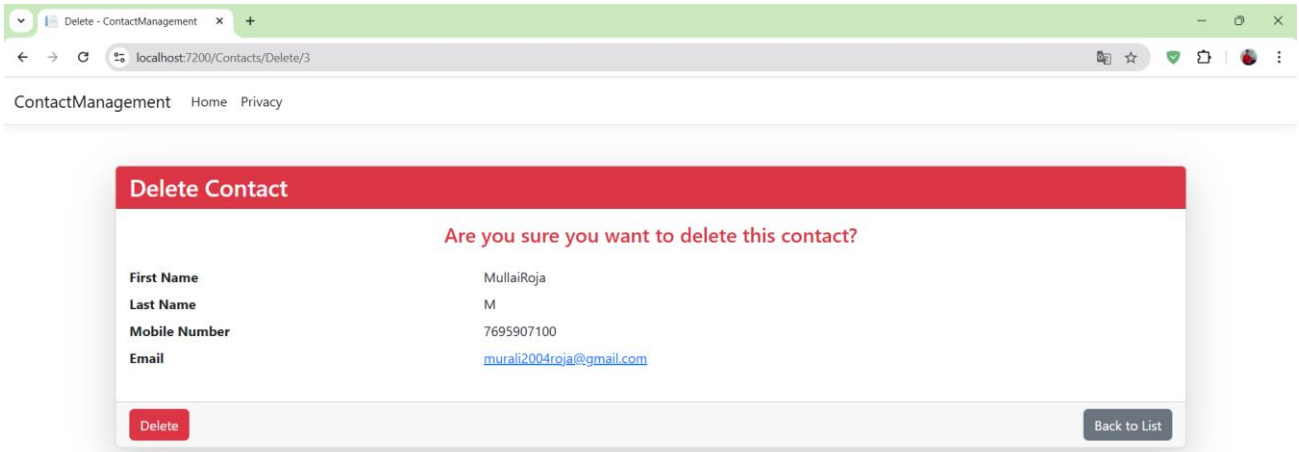
OUTPUT

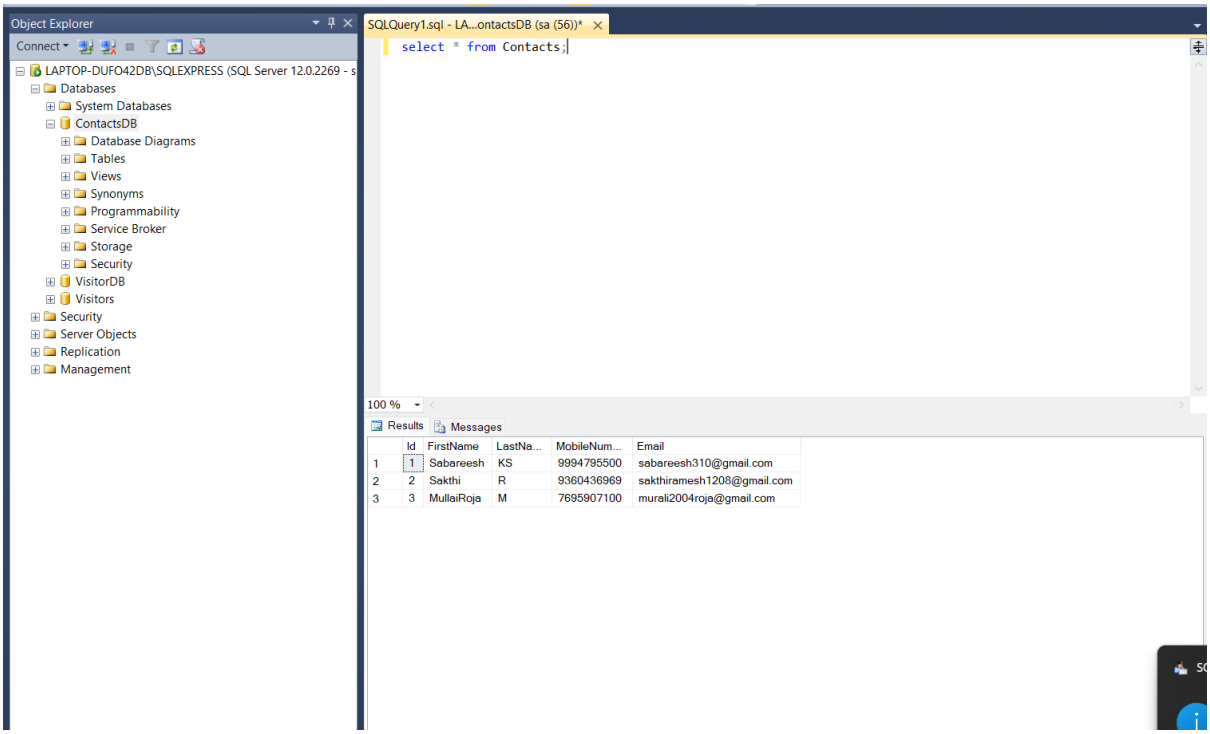





Edit - ContactManagement
localhost:7200/Contacts/Edit/3
ContactManagement
Home
Privacy
Edit
Contacts
Edit Contact
Update Contact Details
FirstName
MullaiRoja
LastName
M
MobileNumber
7695907100
Email
murali2004roja@gmail.com
Save Changes
Back to List
© 2024 - ContactManagement - Privacy


Details - ContactManagement
localhost:7200/Contacts/Details/3
ContactManagement
Home
Privacy
Contact Details
View Contact
First Name
MullaiRoja
Last Name
M
Mobile Number
7695907100
Email
murali2004roja@gmail.com
Edit
Back to List
© 2024 - ContactManagement - Privacy





CHAPTER 6
CONCLUSION

CONCLUSION

The **Contact Management System** is a comprehensive web application designed to simplify and enhance the management of contact information. Developed using the **ASP.NET Core MVC framework**, it provides a seamless and efficient way to perform CRUD (Create, Read, Update, Delete) operations. Users can easily create new contacts, edit existing ones, view detailed information, and delete outdated entries. Each contact record includes essential fields such as First Name, Last Name, Contact Number, and Email Address, catering to both personal and professional needs.

The application is built with a focus on user experience and functionality. Its intuitive interface ensures smooth navigation across pages, including the index, create, edit, details, and delete pages. Robust data validation and secure database management ensure that all contact details are accurate and protected against unauthorized access. Features such as pre-filled forms for editing and confirmation prompts for deletion make the system user-friendly while minimizing errors.

Furthermore, the use of **Entity Framework Core** for database interactions and scaffolding for rapid development underscores the scalability and maintainability of the project. These modern tools enable efficient handling of data and simplify the addition of new features in the future. The responsive design ensures compatibility with various devices, enhancing accessibility and convenience for users.

In conclusion, the **Contact Management System** fulfills its purpose of efficiently managing contact information with a blend of functionality, security, and ease of use. Its well-structured design and modern implementation make it a reliable and scalable solution, showcasing the potential of ASP.NET Core MVC for developing practical and user-centric applications.

CHAPTER 7
REFERENCES

1) **Microsoft ASP.NET Core Documentation**: Official guide for ASP.NET Core and Entity Framework Core.
2) **W3Schools ASP.NET Tutorial**: Beginner-friendly ASP.NET Core and MVC resources.
3) **Entity Framework Core Documentation**: Guide for database integration with ASP.NET Core.
4) **YouTube Tutorials**: Practical CRUD demos in ASP.NET Core:
   Code With Mukesh.
5) **Stack Overflow**: Troubleshooting and solutions for ASP.NET Core issues.