6COSC007C.1 Enterprise Application Development

Coursework: 02

Informatics Institute of Technology

Department of Computing

BSc Computer Science (IIT Sri Lanka)

Name: Shenal Anthony

IIT ID: 2018383

UOW ID: w1742306

Table of Contents

Table of Figures

1. Design Changes and Reasons

The coursework specification had to design and implement a project management system. The implementation for the project management system which is the coursework two which will be discussed in this document. The first section of this document will discuss the design changes and the reasons for these changes. The designs for this were completed during the coursework one and have been implemented according to the designs. But for some reasons during the implementation the design has been changed due to some reasons.

The software requirements that had been identified has been implemented other than for some few requirements. Mainly the view of the features of the system had been created other than the report generations and prediction. The reason for that was not able to connect the tables with each other and was not success in creating the assigning parts in the system. The complexity of the coding played a huge down fall during the implementation. Even the CRUD operations took a huge amount of time to implement.

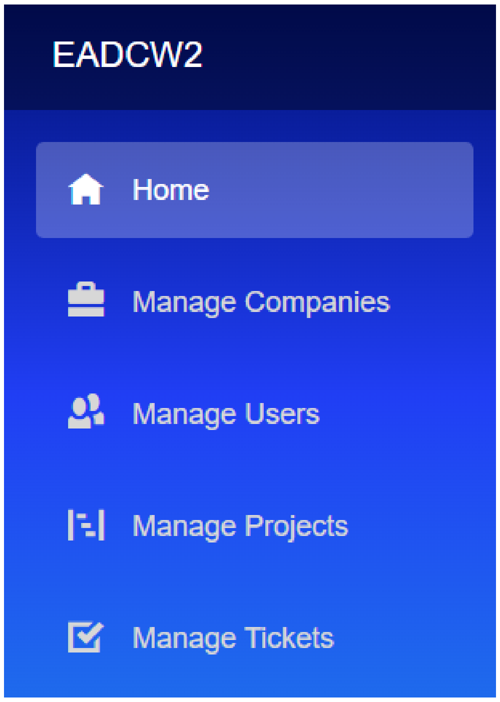


Figure 1: Navigation Bar

The login interface that was design was also not implemented incorrectly. The system has the blazors built in login component for the admin to login. In creating the login was unsuccessful due to not been able to connect the database tables. The user table was initial supposed to be the table that should have been used in the login face but since the system was using the built in login and not using a custom login form made the login for other users not been able to login to the system. The project assigning to company and user and ticket assigning to the user was designed only on the use cases but was not added to the wireframes. The assigning was created using a drop down where the user can select the assignee and assign the project to a company, user and ticket to user. Apart from the assigning same as that the project assigning for the ticket, selecting the project to which the ticket belongs to and assign user to company for set the company of which the user is working for. Those changes were new and added just as the three main assigning.

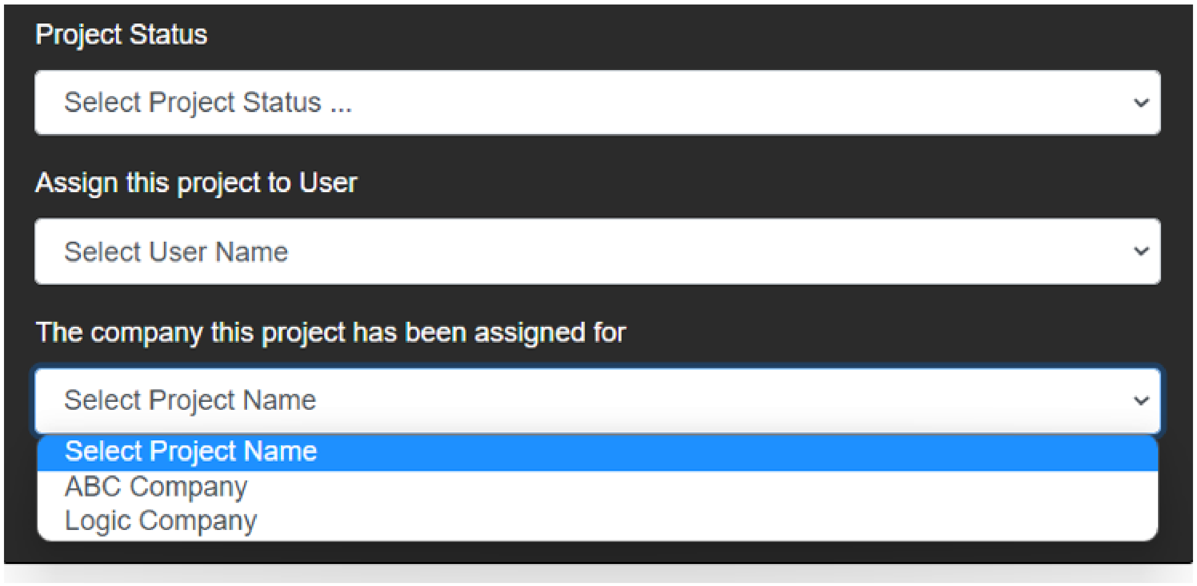
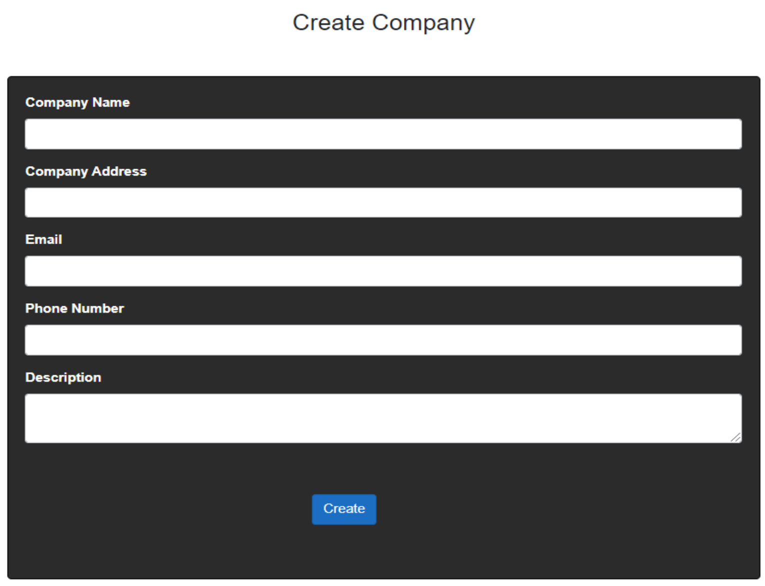

Project Status
Select Project Status ...
Assign this project to User
Select User Name
The company this project has been assigned for
Select Project Name
Select Project Name
ABC Company
Logic Company

Figure 2 : Project Assign fields

Some of the classes that was identified in the CRC table were not implemented but created in a single class to have maintainability of the system and the project. Such as the CRUD operation classes were created inside one class. Although the report generation and the prediction class and the model views have not been implemented.

When it came to the user interface, the wireframes that was originally designed had to be changed since the designs were not very user friendly and to be modified in order to get the user friendliness and also to make the readability even so. Also, the uses of confirmation and alert boxes to alert the user of the changes made and to ask confirmation of deleting a record form the system.


Create Company

Company Name

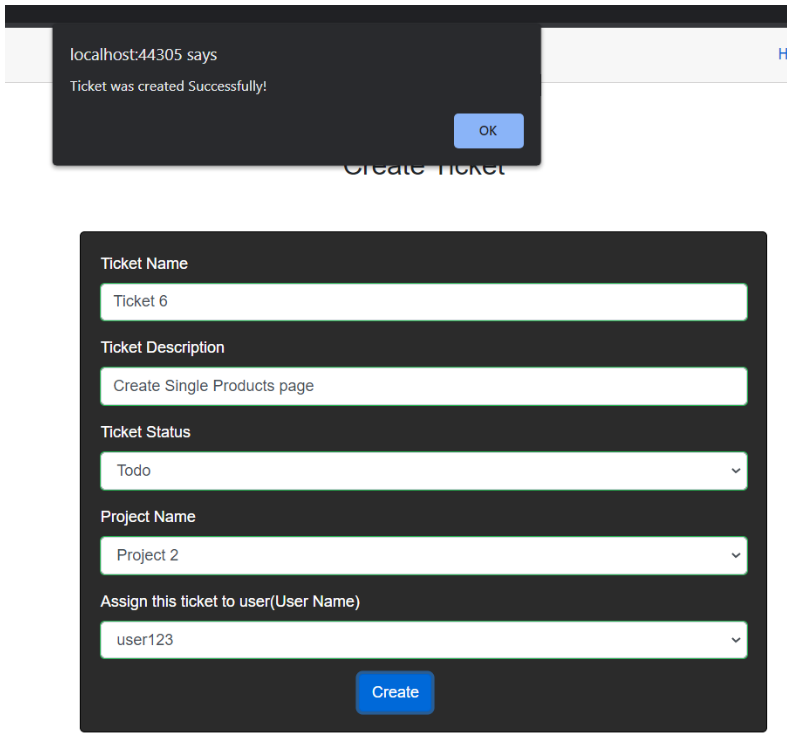
Company Address

Email

Phone Number

Description

Figure 3:Create form


localhost:44305 says

Ticket was created Successfully!

Ticket Name: Ticket 6

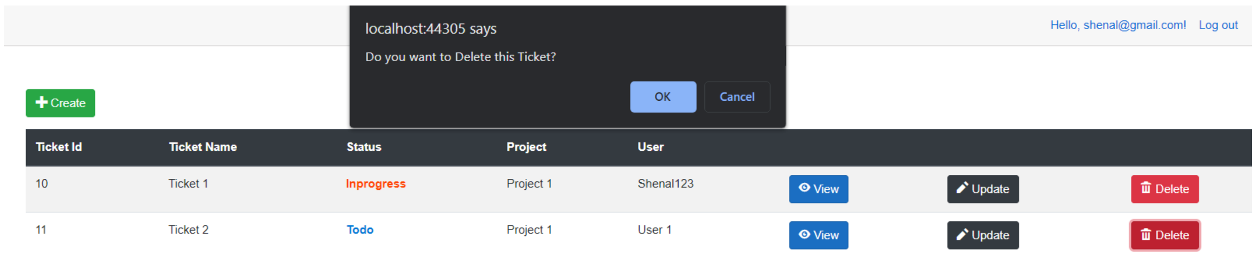
Ticket Description: Create Single Products page

Ticket Status: Todo

Project Name: Project 2

Assign this ticket to user(User Name): user123

Figure 4 : Alert Box



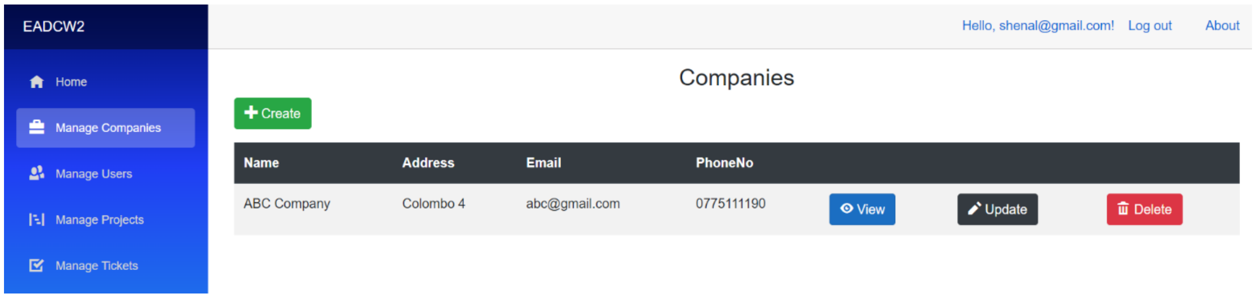
localhost:44305 says

Do you want to Delete this Ticket?

Ticket Id	Ticket Name	Status	Project	User			
10	Ticket 1	Inprogress	Project 1	Shenal123	View	Update	Delete
11	Ticket 2	Todo	Project 1	User 1	View	Update	Delete

Figure 5 : Confirmation Box

A final design change that was created was the implementing the edit functionality for the manage company section. During the design stage the update company was not included in the use case diagram since it was not that important. But during the implementation it was identified as it should exist since some details of the company might change during the time of creation like the address or the phone number. So, the update company feature was implemented so during a situation like that the system data could be changed.



Companies

Name	Address	Email	PhoneNo			
ABC Company	Colombo 4	abc@gmail.com	0775111190	View	Update	Delete

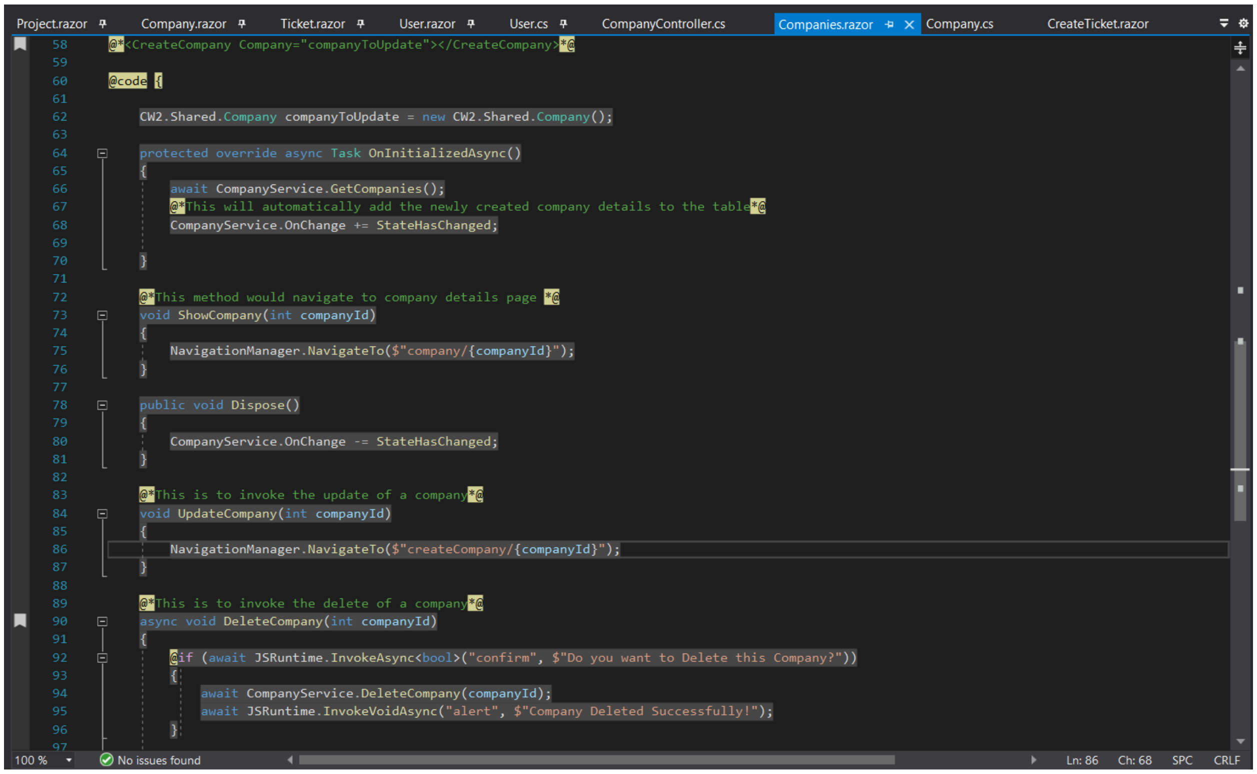
Figure 6 : Company Update

2. Evaluation: Implementation Choices

In the evaluation section the implementation would be discussed as what technics that has been used to keep the readability and the maintainability of the code and the UI along with the design patterns that was used to implement and how it was helpful.

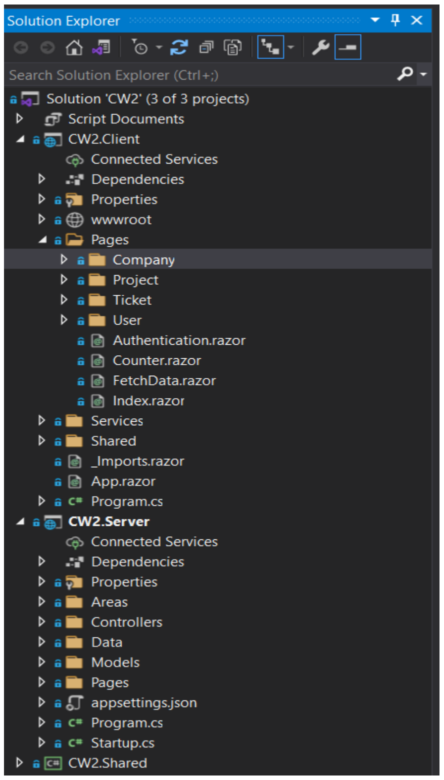
2.1 Maintainability and Readability

In order to keep a good maintainability and readability of the code, comments have been used through out the project and formatting of the code also gives the code the great readability and the file hierarchy of having the controllers the views and the models in different folders to make it easier for check files even after some time.



```
@*<CreateCompany Company="companyToUpdate"></CreateCompany>*@

@code {

    CW2.Shared.Company companyToUpdate = new CW2.Shared.Company();

    protected override async Task OnInitializedAsync()
    {
        await CompanyService.GetCompanies();
        @*This will automatically add the newly created company details to the table*@
        CompanyService.OnChange += StateHasChanged;

    }

    @*This method would navigate to company details page *@
    void ShowCompany(int companyId)
    {
        NavigationManager.NavigateTo($"company/{companyId}");
    }

    public void Dispose()
    {
        CompanyService.OnChange -= StateHasChanged;
    }

    @*This is to invoke the update of a company*@
    void UpdateCompany(int companyId)
    {
        NavigationManager.NavigateTo($"createCompany/{companyId}");
    }

    @*This is to invoke the delete of a company*@
    async void DeleteCompany(int companyId)
    {
        @if (await JSRuntime.InvokeAsync<bool>("confirm", $"Do you want to Delete this Company?"))
        {
            await CompanyService.DeleteCompany(companyId);
            await JSRuntime.InvokeVoidAsync("alert", $"Company Deleted Successfully!");
        }
```

Figure 7: Comments On the code

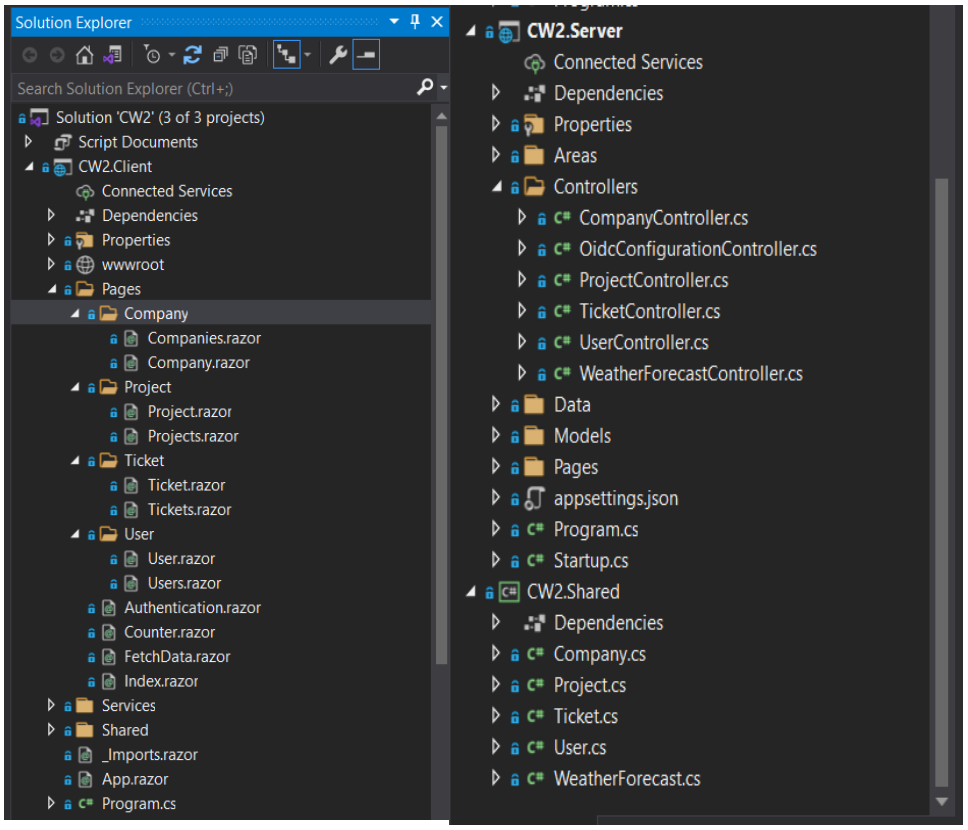


```
Solution 'CW2' (3 of 3 projects)
  Script Documents
  CW2.Client
    Connected Services
    Dependencies
    Properties
    wwwroot
    Pages
      Company
      Project
      Ticket
      User
      Authentication.razor
      Counter.razor
      FetchData.razor
      Index.razor
    Services
    Shared
    _Imports.razor
    App.razor
    Program.cs
  CW2.Server
    Connected Services
    Dependencies
    Properties
    Areas
    Controllers
    Data
    Models
    Pages
    appsettings.json
    Program.cs
    Startup.cs
  CW2.Shared
```

Figure 8 : File Hierarchy

2.2 Design Pattern

The MVC structure was the design pattern that was used to implement the solution. Creating the models that was identified along with their attributes and then creating the controllers with the crud operations and finally the view pages to view the output. Since this implementation is tested along the way Test Driven Development has also been used. The structure of the project is easy to access and edit since it follows the MVC structure.

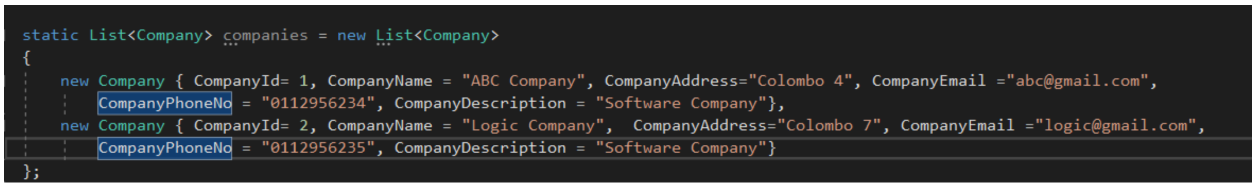


```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CW2' (3 of 3 projects)
  Script Documents
  CW2.Client
    Connected Services
    Dependencies
    Properties
    wwwroot
    Pages
      Company
        Companies.razor
        Company.razor
      Project
        Project.razor
        Projects.razor
      Ticket
        Ticket.razor
        Tickets.razor
      User
        User.razor
        Users.razor
      Authentication.razor
      Counter.razor
      FetchData.razor
      Index.razor
    Services
    Shared
    _Imports.razor
    App.razor
    Program.cs
  CW2.Server
    Connected Services
    Dependencies
    Properties
    Areas
    Controllers
      CompanyController.cs
      OidcConfigurationController.cs
      ProjectController.cs
      TicketController.cs
      UserController.cs
      WeatherForecastController.cs
    Data
    Models
    Pages
    appsettings.json
    Program.cs
    Startup.cs
  CW2.Shared
    Dependencies
    Company.cs
    Project.cs
    Ticket.cs
    User.cs
    WeatherForecast.cs
```

Figure 9 : MVC Structure

2.3 Testability

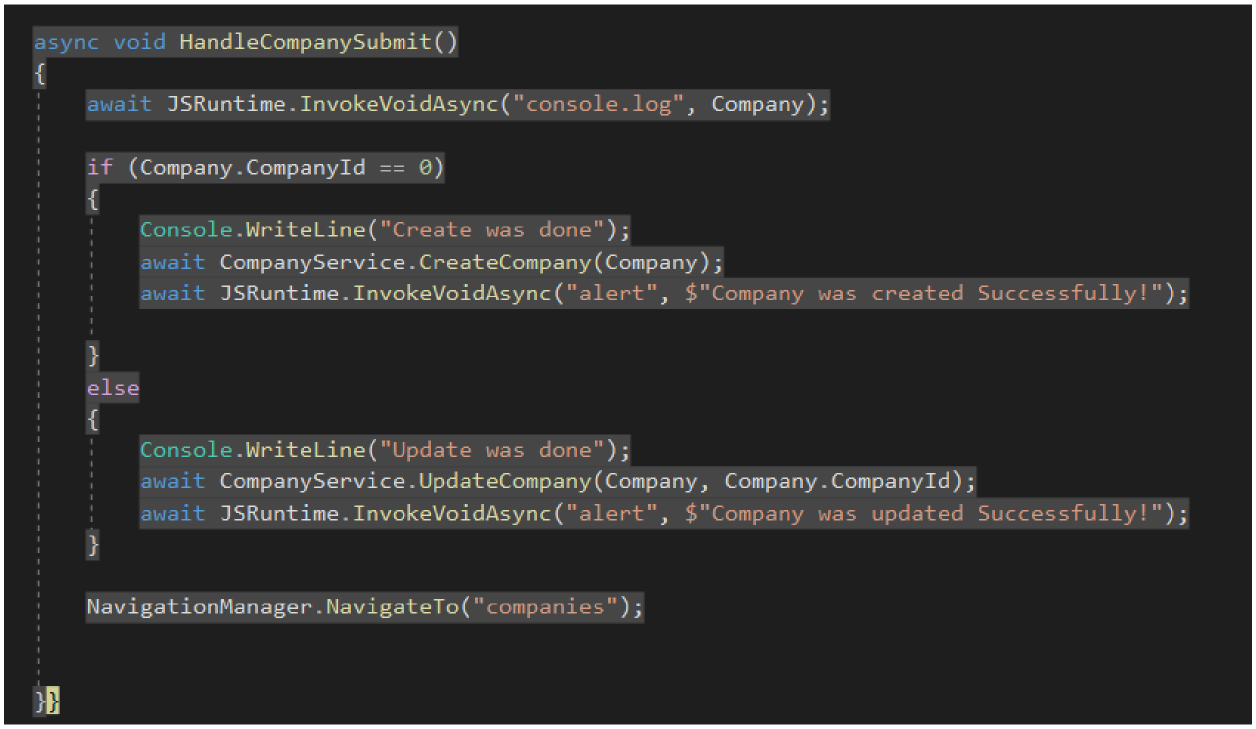
Before connecting to the database, the controllers and methods were tested out using Lists. Lists containing the data made it easy to use and high performance since it takes time to do operations with database. That helped to ease the productivity and prevent the bugs and have a clean solution before creating database and connecting to it.



```
static List<Company> companies = new List<Company>
{
    new Company { CompanyId= 1, CompanyName = "ABC Company", CompanyAddress="Colombo 4", CompanyEmail ="abc@gmail.com",
        CompanyPhoneNo = "0112956234", CompanyDescription = "Software Company"},
    new Company { CompanyId= 2, CompanyName = "Logic Company",  CompanyAddress="Colombo 7", CompanyEmail ="logic@gmail.com",
        CompanyPhoneNo = "0112956235", CompanyDescription = "Software Company"}
};
```

Figure 10 : Array used in the system

Also console messages to identify the operations that are performed are executing correctly and to debug any issues that would occur when going through the system. Also, the use of alert boxes were help in displaying the success messages as a sign of completing the task that was given

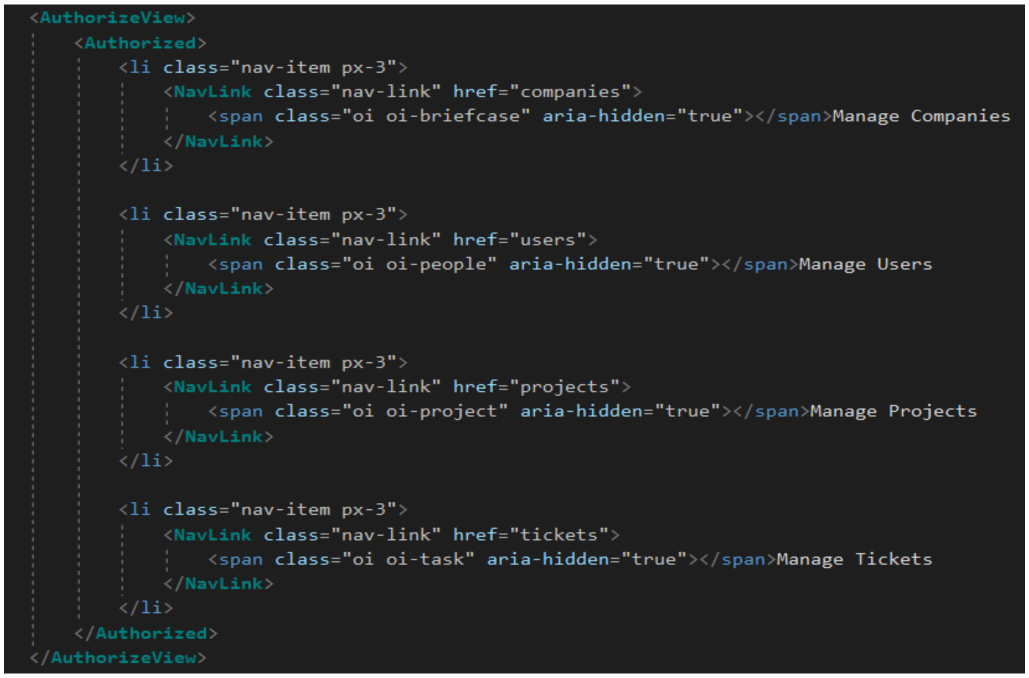


```
async void HandleCompanySubmit()
{
    await JSRuntime.InvokeVoidAsync("console.log", Company);

    if (Company.CompanyId == 0)
    {
        Console.WriteLine("Create was done");
        await CompanyService.CreateCompany(Company);
        await JSRuntime.InvokeVoidAsync("alert", $"Company was created Successfully!");

    }
    else
    {
        Console.WriteLine("Update was done");
        await CompanyService.UpdateCompany(Company, Company.CompanyId);
        await JSRuntime.InvokeVoidAsync("alert", $"Company was updated Successfully!");
    }

    NavigationManager.NavigateTo("companies");


}}
```

Figure 11 : Console messages and alert messages

2.4 Security

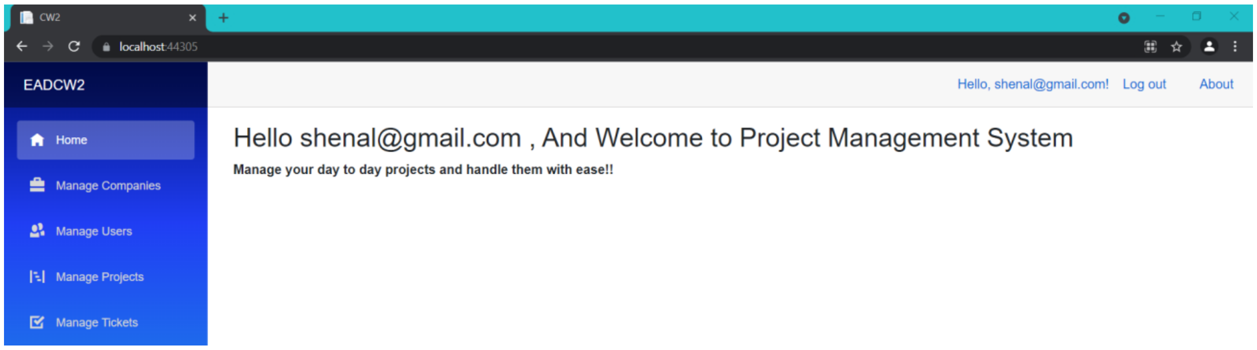
For security purpose the system cannot be used without getting logged in only the login link would be displayed in the home since if the features would be available freely anyone would be able to perfume any CRUD operation in the system and it would be a disaster.



```
<AuthorizeView>
    <Authorized>
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="companies">
                <span class="oi oi-briefcase" aria-hidden="true"></span>Manage Companies
            </NavLink>
        </li>

        <li class="nav-item px-3">
            <NavLink class="nav-link" href="users">
                <span class="oi oi-people" aria-hidden="true"></span>Manage Users
            </NavLink>
        </li>

        <li class="nav-item px-3">
            <NavLink class="nav-link" href="projects">
                <span class="oi oi-project" aria-hidden="true"></span>Manage Projects
            </NavLink>
        </li>

        <li class="nav-item px-3">
            <NavLink class="nav-link" href="tickets">
                <span class="oi oi-task" aria-hidden="true"></span>Manage Tickets
            </NavLink>
        </li>
    </Authorized>
</AuthorizeView>
```

Figure 12 : Authorization to access the navigation features

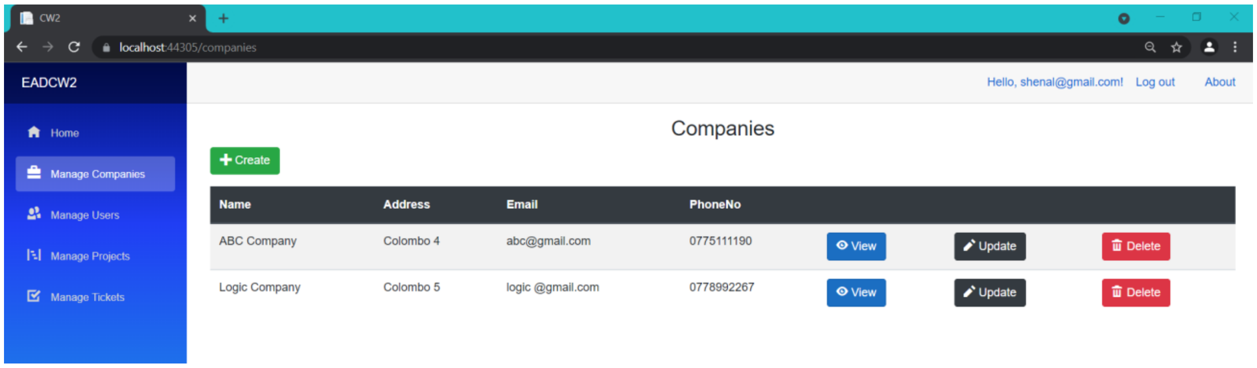
3. User Interfaces



Hello shenal@gmail.com , And Welcome to Project Management System

Manage your day to day projects and handle them with ease!!

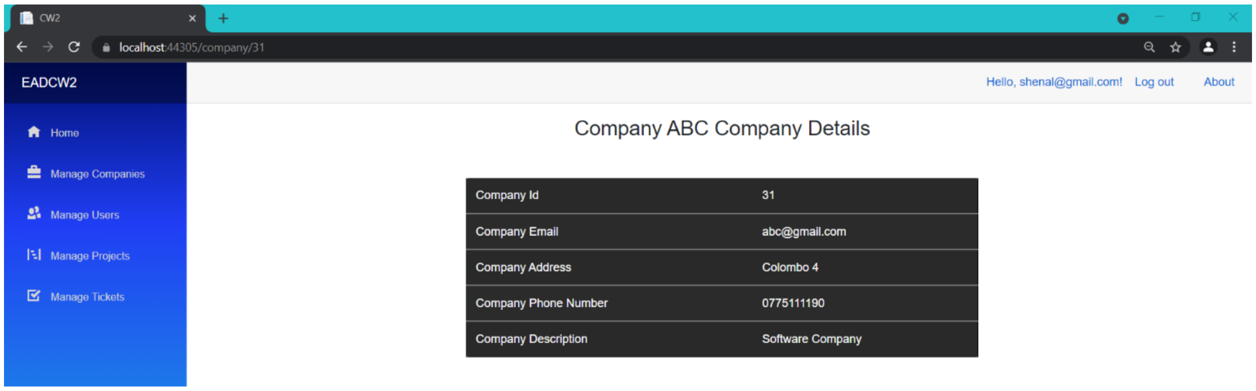
Figure 13 : Home Page



Companies

Name	Address	Email	PhoneNo			
ABC Company	Colombo 4	abc@gmail.com	0775111190	View	Update	Delete
Logic Company	Colombo 5	logic @gmail.com	0778992267	View	Update	Delete

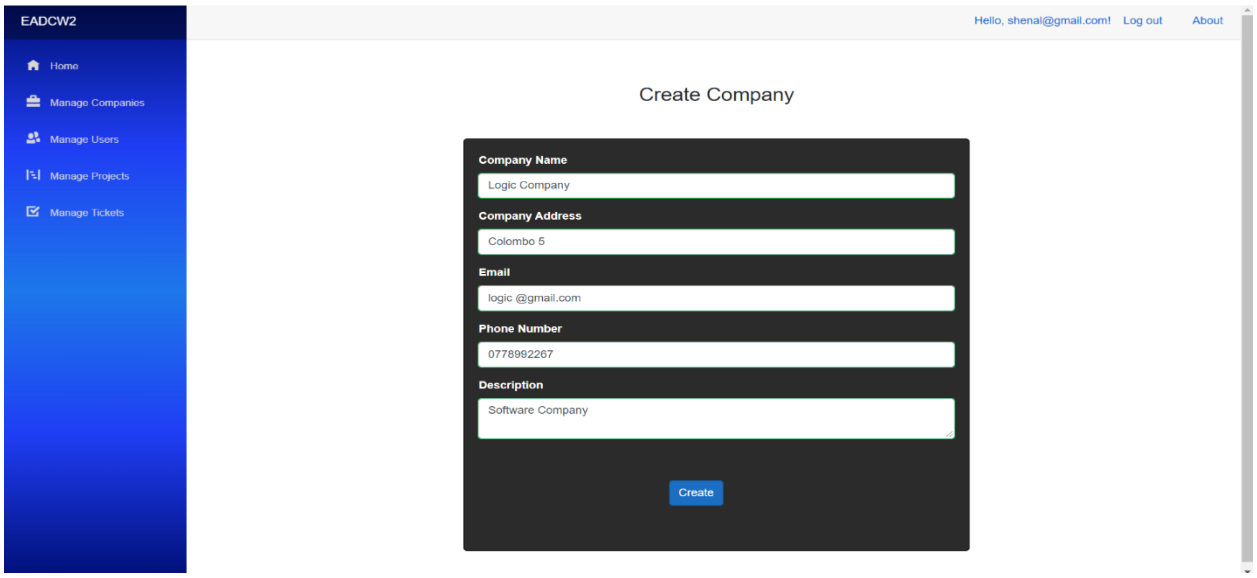
Figure 14 : Manage Company Page



Company ABC Company Details

Company Id	31
Company Email	abc@gmail.com
Company Address	Colombo 4
Company Phone Number	0775111190
Company Description	Software Company

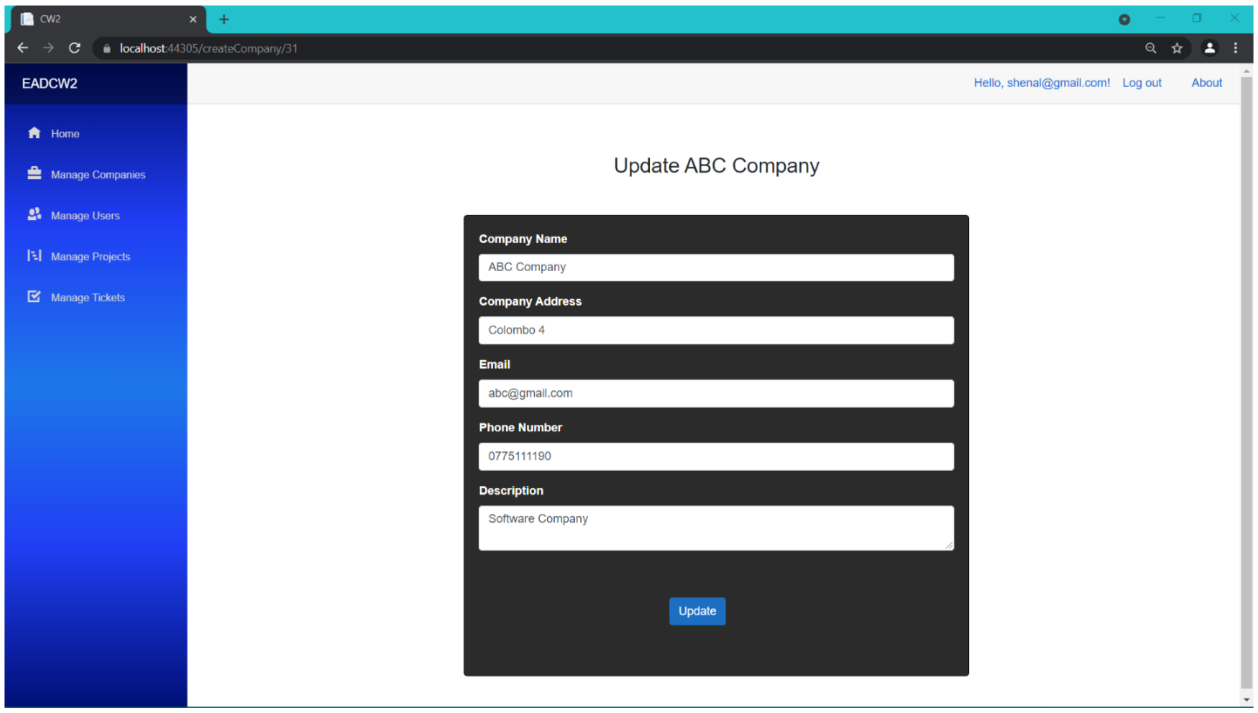
Figure 15 : Company Details Page



Create Company

Company Name	Logic Company
Company Address	Colombo 5
Email	logic @gmail.com
Phone Number	0778992267
Description	Software Company

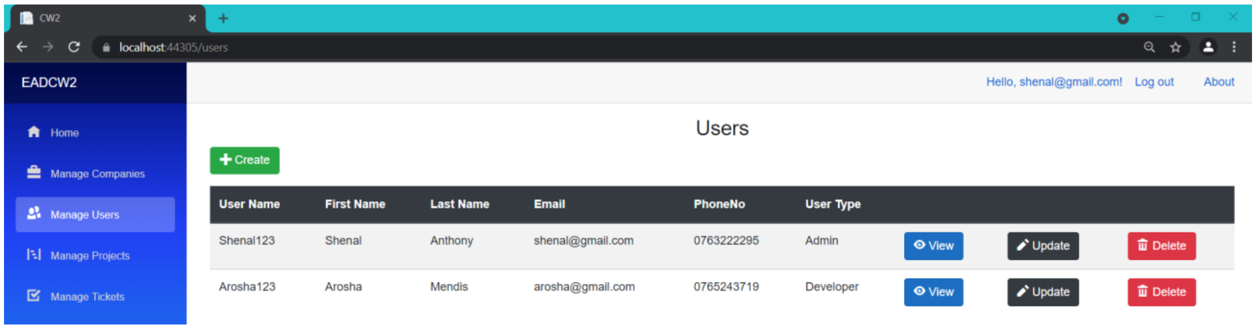
Figure 16 : Create Company Form



Update ABC Company

Company Name	ABC Company
Company Address	Colombo 4
Email	abc@gmail.com
Phone Number	0775111190
Description	Software Company

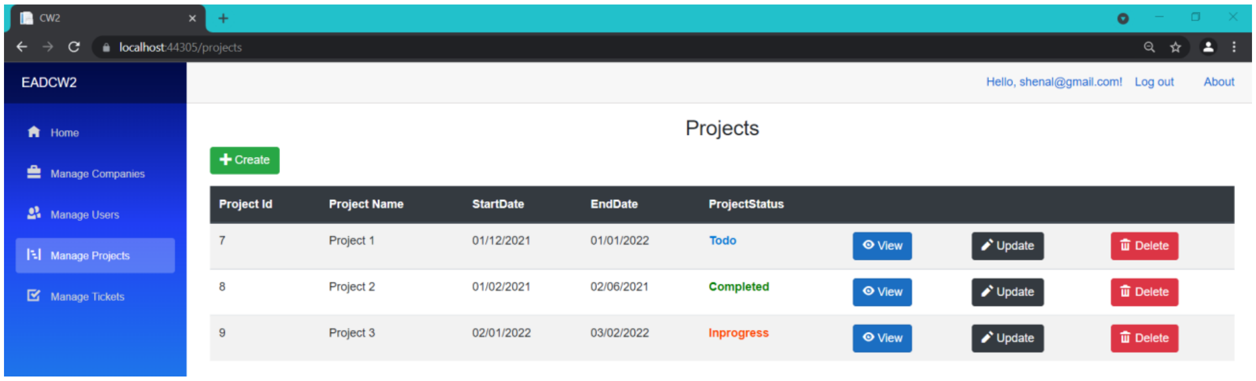
Figure 17 :Update Company Form



Users

User Name	First Name	Last Name	Email	PhoneNo	User Type			
Shenal123	Shenal	Anthony	shenal@gmail.com	0763222295	Admin	View	Update	Delete
Arosha123	Arosha	Mendis	arosha@gmail.com	0765243719	Developer	View	Update	Delete

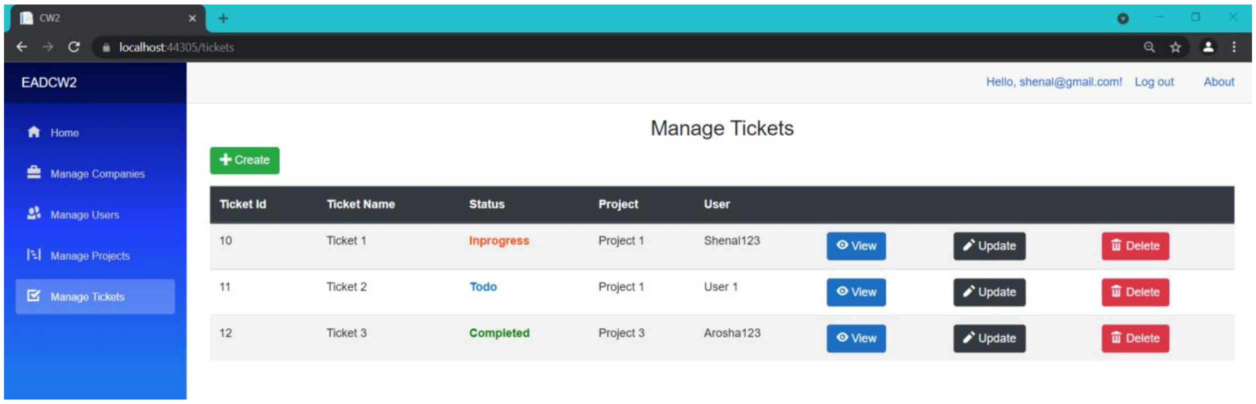
Figure 18 : User Manage Page



Projects

Project Id	Project Name	StartDate	EndDate	ProjectStatus			
7	Project 1	01/12/2021	01/01/2022	Todo	View	Update	Delete
8	Project 2	01/02/2021	02/06/2021	Completed	View	Update	Delete
9	Project 3	02/01/2022	03/02/2022	Inprogress	View	Update	Delete

Figure 19 : Projects Manage Page



Manage Tickets

Ticket Id	Ticket Name	Status	Project	User			
10	Ticket 1	Inprogress	Project 1	Shenal123	View	Update	Delete
11	Ticket 2	Todo	Project 1	User 1	View	Update	Delete
12	Ticket 3	Completed	Project 3	Arosha123	View	Update	Delete

Figure 20 : Manage Tickets Page