Internship Final Report
Web Development Intern

By

Raisa Tarannum
21141032

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Raisa Tarannum
21141032

Approval

The thesis/project titled "Internship Final Report Web Development Intern " submitted by

1. Raisa Tarannum (21141032)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Md. Tawhid Anwar
Senior Lecturer
Department of Computer
Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Asif Ahmed
Senior Manager
IT Department
Summit Communications
Limited

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate
Professor
Department of Computer
Science and Engineering
Brac University

Abstract

This report is an outline of the experience I have gained as well all the things that I have learned during my time as a web development intern at Summit Communications Limited. The internship provided hands-on exposure to the real-world software development industry, particularly in web technologies such as C#, ASP.NET, HTML, CSS, JavaScript, and SQL. During my training phase, I developed a few practice projects, including a School Management System, using basic CRUD operations and following MVC architecture. Later on, I contributed to the company's in-house Fuel Management System by maintaining and developing functionalities. The internship has helped me improve my technical abilities as well as soft skills like time management and problem-solving.

Acknowledgement

First and foremost, I am grateful to the Almighty Allah for granting me the strength and opportunity to complete my internship smoothly and on time.
Secondly, I would like to thank my advisor, MD Tawhid Anwar sir for his kind support and advice throughout the internship. I would also like to thank my co-supervisor: Md Asif Ahmed, Senior Manager(Summit Communications Limited) and his team for helping me learn so much throughout my time at Summit Communications Limited. And finally, heartfelt thanks to my parents — without their constant support, encouragement, and prayers, this journey would not have been possible.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Objective

This internship report is being written to provide insight into my five and a half months of experience as an intern at Summit Communications Limited. It is a part of the Computer Science program under the Computer Science and Engineering Department at BRAC University.
In the following, I discuss the projects I have worked on during my time there and how this internship has helped me develop my skills.

1.2 Company Background

Summit Communications Limited is a part of one of Bangladesh's largest conglomerates, Summit Group. It is the leading provider of telecommunication infrastructure services in Bangladesh. Summit Communications has Nationwide Telecommunication Transmission Network(NTTN) and Gateway Licenses.
Summit Communications Limited aims to connect people by providing world-class media services and building a strong telecom network to support the nation's growth in the ICT and telecom sectors.
I chose to do an internship for the completion of my degree and to also get some practical work experience in a real world setting. I chose Summit Communications Limited for my internship where I was assigned to the Software Development team. Since I have a keen interest in web development, this was a great learning experience for me.

1.3 Methodology

This report is based on my work experience and all that I have learned during my time as an intern at Summit Communications. The information I have gathered has been classified into two categories: primary and secondary data. The primary data consists of data and examples that I have obtained through observations throughout the duration of the internship. As for the secondary data, it consists of all the information that I have collected from web-based tutorials as well as documentation from reliable online sources.

1.3.1 Primary Data

- Observations on the job
- Training from senior colleagues

1.3.2 Secondary Data

- Online resources and documentation

Chapter 2

Training Phase

2.1 Overview

This was a 5.5 month internship under Summit Communications Limited. During these past five and a half months, I underwent training and worked on multiple projects, including the company's own in-house data management systems. As I was assigned tasks and web application projects by my supervisor from the company, the experience helped me grow my knowledge of web development.

2.2 Programming Languages and tools used

For my web applications, I used Visual Studio programming, as instructed. I familiarized myself with Model View Controller design (MVC) since that is what I used for my projects.
As for the database, I used Microsoft SQL Server Management Studio to manage the databases. The Software Development team at Scomm mostly uses C# to code and ASP.NET for their api integration and backend development. As I was unfamiliar with the language, I was given some time to learn C#. For the front-end implementation, I also used html, css and Javascript. I got to learn a lot more about Javascript and its JQuery library and AJAX to create dynamic, responsive web applications.
During my training phase, my teammates helped me out on how to implement the language and frameworks. Along with that, I also used websites like w3schools, GeeksforGeeks and Stack Overflow for web tutorials and documentation.

2.3 Student Management System

I worked on a few practice projects there initially, one of them being a student management system. It was a system that manages students, teachers, details about the students and their results to keep data organized. It can also keep track of attendances of the students.

2.3.1 Approach

I used C# ASP.NET for the API integration for the project. I used MVC architecture in order to make the code easier to manage and maintain by separating the application into distinct components- Model, View and Controller. As instructed I used a code-first approach which was new to me, but I got comfortable with it quite quickly.
I began with first normalizing the database tables. This was important since the tables where dependant on each other, creating complications. The table was normalized to 3F.

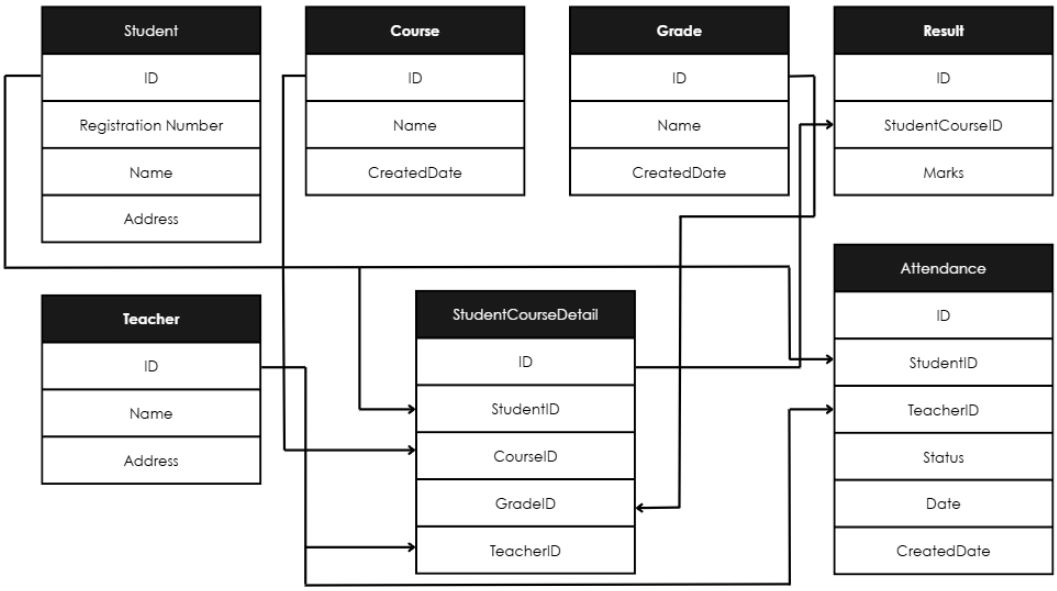


Figure 2.1: Normalized Database Table

Since this was a code-first approach, I used Entity Framework to map the classes onto the database. Using the entity framework, I handled migration and connections and also called methods like .Add(), .Update() or .SaveChanges to change data. I also used LINQ to write queries and interact with the database.

2.3.2 Code Snippets and Implementation Details

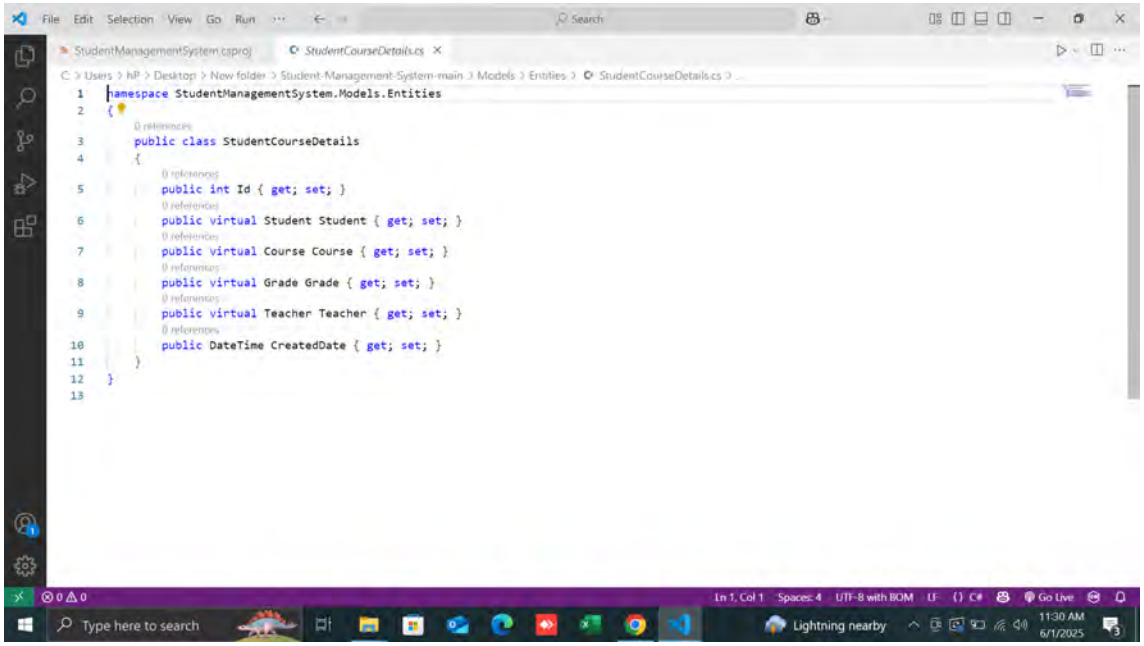


Figure 2.2: StudentCourseDetails Model

```
using Microsoft.AspNetCore.Mvc.Rendering;
using StudentManagementSystem.Models. Entities;
namespace Student ManagementSystem.Models
{

    0 references
    public class AttendanceViewModel : Attendance
    {
        0 references
        public List<SelectListItem> TeacherViewModellist { get; set; }
        0 references
        public int StudentId { get; set; }
        0 references
        public int TeacherId { get; set; }
    }
}
```

Figure 2.3: View Model of Attendance Entity

For the project, I started off by creating 7 models: Grade, Course, Student, Teacher, Attendance, StudentCourseDetail and finally, Result. I used normalization to simplify the database and make the process easier to follow. I also created View Models wherever necessary, especially when dealing with foreign-key relationships.

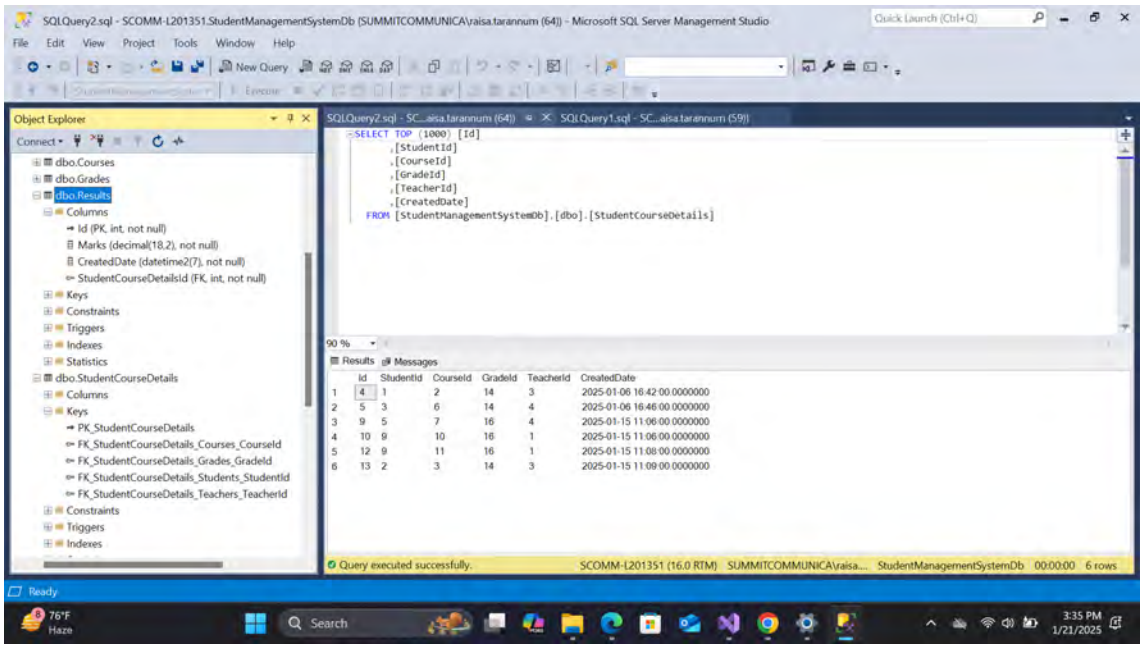


Figure 2.4: Results Database

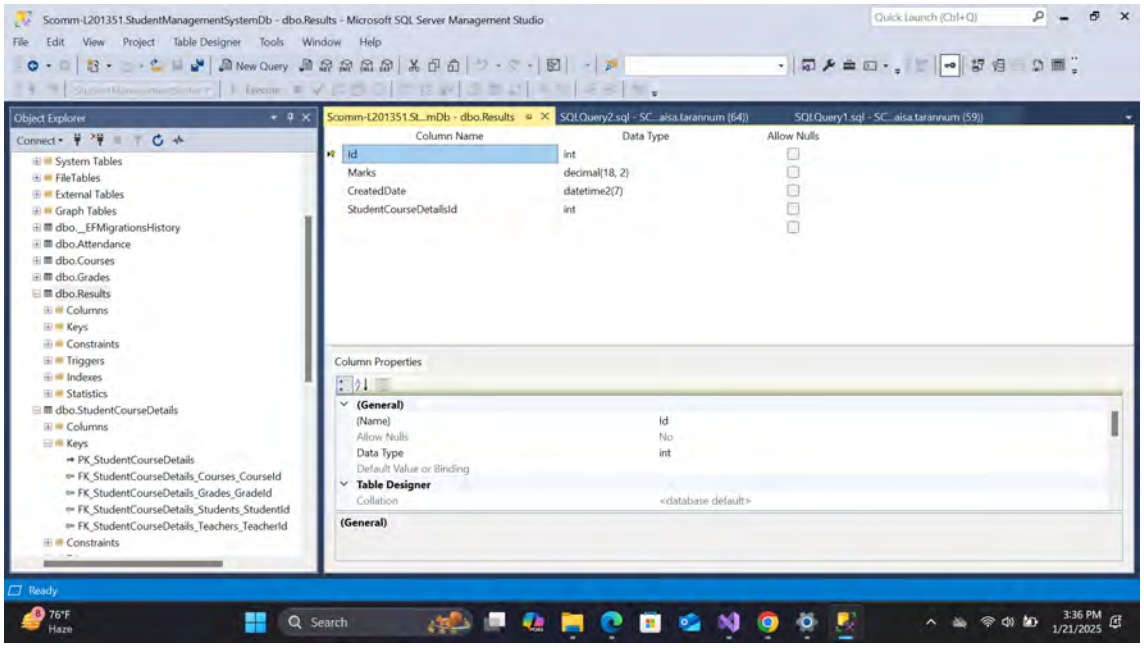
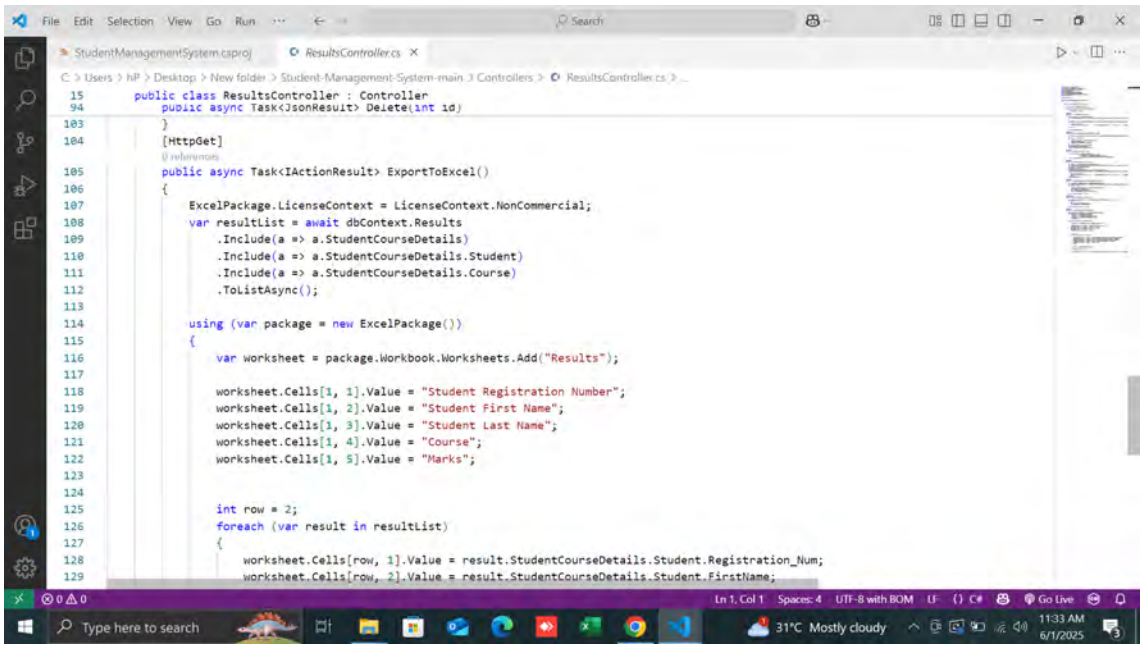


Figure 2.5: Results Database(2)

I used Microsoft SQL Server to manage the database. As seen in the figures above, some of the tables or entities have foreignkey relationships.



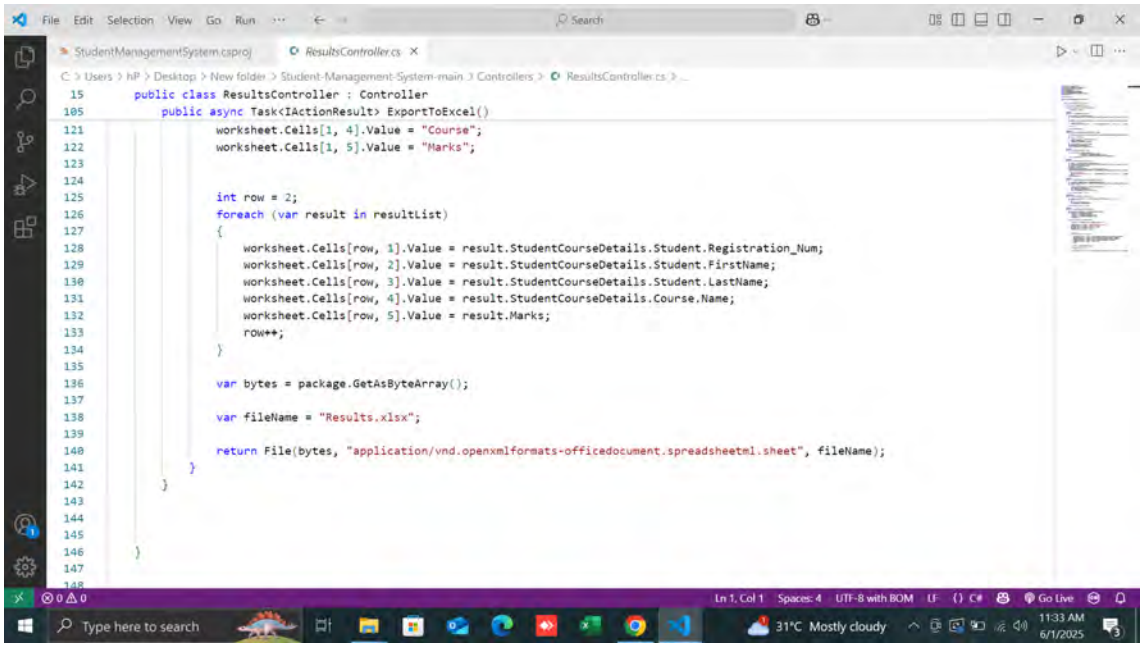
```
public class ResultsController : Controller
    public async Task<JsonResult> Delete(int id)
    }
    [HttpGet]
    public async Task<IActionResult> ExportToExcel()
    {
        ExcelPackage.LicenseContext = LicenseContext.NonCommercial;
        var resultList = await dbContext.Results
            .Include(a => a.StudentCourseDetails)
            .Include(a => a.StudentCourseDetails.Student)
            .Include(a => a.StudentCourseDetails.Course)
            .ToListAsync();

        using (var package = new ExcelPackage())
        {
            var worksheet = package.Workbook.Worksheets.Add("Results");

            worksheet.Cells[1, 1].Value = "Student Registration Number";
            worksheet.Cells[1, 2].Value = "Student First Name";
            worksheet.Cells[1, 3].Value = "Student Last Name";
            worksheet.Cells[1, 4].Value = "Course";
            worksheet.Cells[1, 5].Value = "Marks";


            int row = 2;
            foreach (var result in resultList)
            {
                worksheet.Cells[row, 1].Value = result.StudentCourseDetails.Student.Registration_Num;
                worksheet.Cells[row, 2].Value = result.StudentCourseDetails.Student.FirstName;
```

Figure 2.6: Method to export list into an excel file



```
public class ResultsController : Controller
    public async Task<IActionResult> ExportToExcel()
            worksheet.Cells[1, 4].Value = "Course";
            worksheet.Cells[1, 5].Value = "Marks";


            int row = 2;
            foreach (var result in resultList)
            {
                worksheet.Cells[row, 1].Value = result.StudentCourseDetails.Student.Registration_Num;
                worksheet.Cells[row, 2].Value = result.StudentCourseDetails.Student.FirstName;
                worksheet.Cells[row, 3].Value = result.StudentCourseDetails.Student.LastName;
                worksheet.Cells[row, 4].Value = result.StudentCourseDetails.Course.Name;
                worksheet.Cells[row, 5].Value = result.Marks;
                row++;
            }

            var bytes = package.GetAsByteArray();

            var fileName = "Results.xlsx";

            return File(bytes, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", fileName);
        }
    }


}
```

Figure 2.7: Method to export list into an excel file(2)

The results page has a download button that exports the list data from the page into an excel file. The add pages to insert data makes use of dropdown menus to connect to other tables.

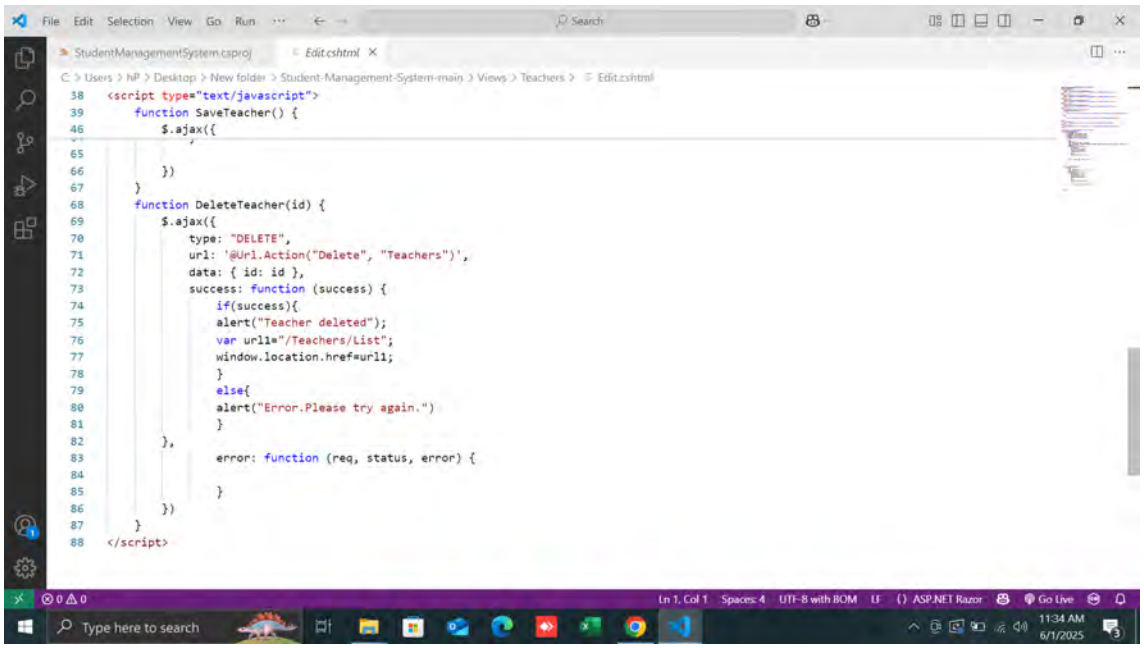


Figure 2.8: Delete function to call Delete() method from controller

I used JQuery and Ajax to handle calling C# functions as well as make the frontend interactive.

Chapter 3

Contributions

3.1 In-House Fuel Management System

During the last few months, I was assigned the task of learning about the company's internal Fuel Management System. I observed my seniors and saw how they managed the system. This system is used internally to handle various administrative and operational tasks, keeping track of requests and user input. Through this, I got to see how real-world software applications are not only developed but also maintained and supported after deployment.

3.2 Maintenance and Support

The in-house Fuel Management System (FMS) has been designed to manage the distribution and usage of fuel across the company's generator-powered infrastructures, including remote areas. It keeps track of their installed generators throughout Bangladesh, as well as their status and capacity. It allows users to request fuel where it is approved by an admin. The fuel requisition and usage records are saved and are also tracked. These were the two key modules that I had the opportunity to work with.

3.2.1 Search and Remove Fuel Usage Records

My first task was to create a feature to remove fuel usage data. The user would enter the unique ID and using that the method would fetch important information regarding that entry and ask user if they want to delete the record.

```
public ActionResult RemoveFuelUsage()
{
    return View();
}

[HttpPost]
public ActionResult Remove FuelUsage(string uniqueid)
{
    var fuelUsage = _fuelUsageService.FindByUniqueId(uniqueid);
    if (fuelUsage == null)
    {
        ViewBag.Message = "No usage found with this ID.";
        return View();
    }
    if (fuelUsage.Status == -404)
    {
        ViewBag.Message = "This usage has been deleted.";
        return View();
    }
    if (fuelUsage.Status==1)
    {
        ViewBag. Message = "This usage has already been completed and cannot be deleted.";
        return View();
    }
    var fuelUsageViewModel = new FuelUsageViewModel
    {
        UniqueId = fuelUsage. UniqueId,
        Type = fuelUsage.Generator.Type,
        RefuelingDate = fuelUsage. RefuelingDate,
        RefueledAmount = fuelUsage. Refueled Amount,
        BeforeRefuelingStock = fuelUsage. BeforeRefuelingStock, TotalFuelUsage=fuelUsage.TotalFuelUsage,
        PresentRunHour=fuelUsage.PresentRunHour, PreviousRunHour=fuelUsage.Previous RunHour, TotalRunHour=fuelUsage.TotalRunHour,
        ConsumptionRate = fuelUsage.Consumption Rate
    };
    return View("DeleteFuelUsage", fuelUsageViewModel);
}
```

Figure 3.1: RemoveFuelUsage method

I created a method named RemoveFuelUsage that deletes the data using unique ID. User enters unique ID and the method uses the ID to fetch all data related to it.
In this code it can be seen that user gets a validation message that informs the user if the usage with that ID does not exist, if the usage has been deleted, or if the usage has already been completed, in which case it cannot be deleted. In order to make this criteria work, I used the status of the usages. If the usage had already been deleted, the status was updated to -404. If it had been completed, the status was set to 1. So every time the method would check the status and if the status was anything except the ones mentioned, it would successfully get the usage data from the database.

```
[HttpPost]
public ActionResult DeleteFuelUsage(string uniqueid)
{
    UserProfile user = _userProfileService.FindByEmail(User.Identity.Name);
    FuelUsage fuelUsage = _fuelUsageService.FindByUniqueId(uniqueid);
    if (fuelUsage == null)
    {
        ViewBag.Message = "No usage found with this ID.";
        return RedirectToAction("Remove FuelUsage");
    }
    foreach (var history in fuelUsage.FuelUsageHistories.ToList())
    {
        _fuelUsageService.DeleteFuelUsageHistory(history);
    }

    foreach (var attachment in fuelUsage.FuelUsageAttachments.ToList())
    {
        _fuelUsageService.DeleteFuelUsageAttachment(attachment);
    }

    _fuelUsageService.Delete(fuelUsage);
    return RedirectToAction("Remove FuelUsage", new { success = true });
}
#endregion
```

Figure 3.2: DeleteFuelUsage method

Next, I created a method DeleteFuelUsage that deletes the fuel usage data using the unique ID. The fuel usage record has dependencies fuel usage history and fuel usage attachments. The method first deletes data from those modules before deleting from fuel usage.

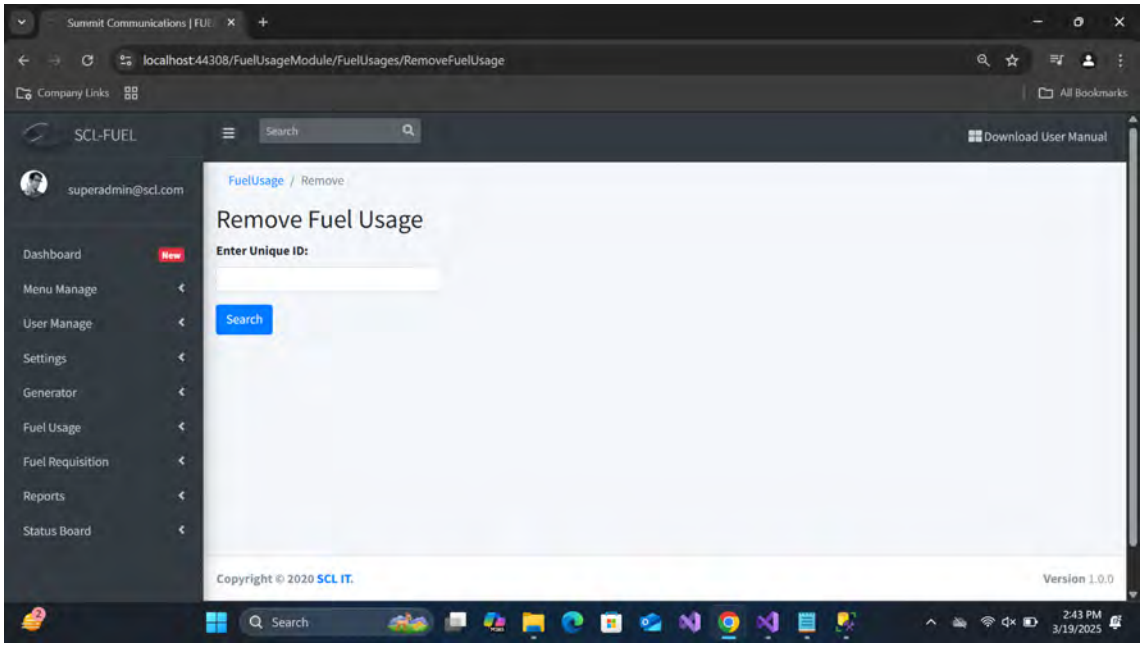


Figure 3.3: Front-end view of search feature

```
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
<script>
    function confirmDelete(event) {
        event.preventDefault();
    Swal.fire({
    title: "Are you sure?",
        text: "You won't be able to revert this!", icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "Yes, delete it!"
    }).then((result) => {
    if (result.isConfirmed) {
            event.target.submit();
        Swal.fire({
        title: "Deleted!",
                text: "Your file has been deleted.",
                icon: "success"
            });
        }
    });
    }
```

Figure 3.4: Confirmation alert using sweetalert

I used sweetalert2 in javascript for user confirmation before running remove method. Sweetalert does not require page reload and is mobile friendly and responsive.

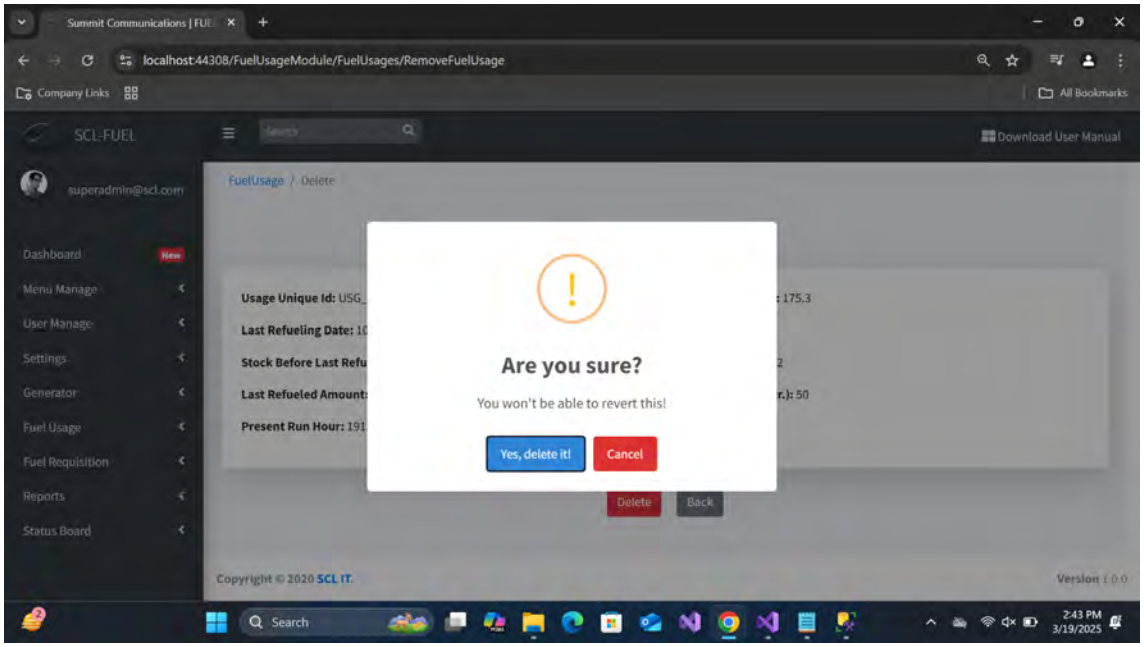


Figure 3.5: Alert UI

```
<form method="post" action="@Url.Action("RemoveFuelUsage", "FuelUsages")">
    <div class="form-group">
        <label for="uniqueid">Enter Unique ID:</label>
        <input type="text" name="uniqueid" id="uniqueid" class="form-control form-control-sm w-25" required />
    </div>
    <button type="submit" class="btn btn-primary">Search</button>
</form>

<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        const urlParams = new URLSearchParams(window.location.search);
        if (urlParams.has("success"))
        {
            Swal.fire({
                title: "Deleted!",
                text: "Fuel usage deleted successfully.",
                icon: "success",
                confirmButtonColor: "#3085d6"
            });
        }
    });
</script>

```

Figure 3.6: Success Message after deletion

User is redirected to previous page and shown a success message that confirms usage has been deleted.

The FMS web application has a fuel usage and fuel requisition section respectively where user can check requests, their status as well all other information. I was given the responsibility to add features there for user convenience. Some of the additional features I added was search filters to allow the user to look up specific data using the filters. I implemented search by UniqueID and search by status filters into the system.

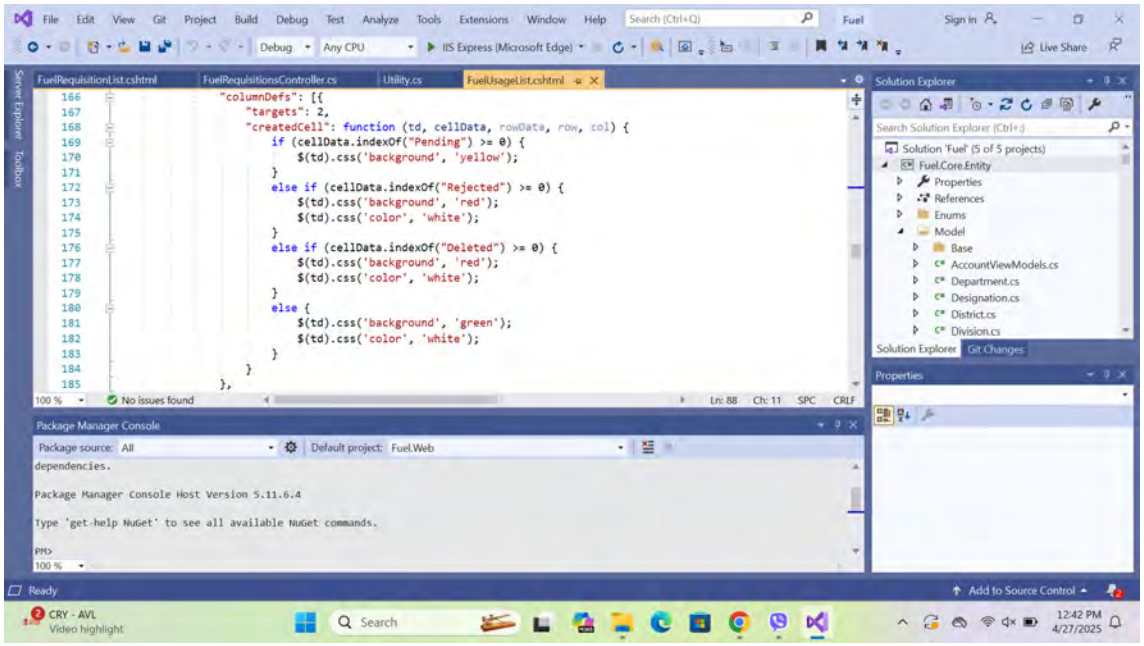


Figure 3.7: Status View

I added distinct colors for the fuel usage view page. The colors depend on the status of the fuel usage. Red for rejected or deleted, yellow for pending and green for others. This helps the user experience as viewers can just see the color and know the status of that certain fuel usage data.

3.2.2 Layered Architecture

Layered architecture follows a pattern that divides the logical layers, typically in the pattern that goes presentation layer (view plus controller), business access layer (service layer) and data access layer. Each of these layers have their own responsibilities and they only interact with the layer below them to complete a task. This ensures a clean code structure and allows easy debugging, especially for large scale applications.

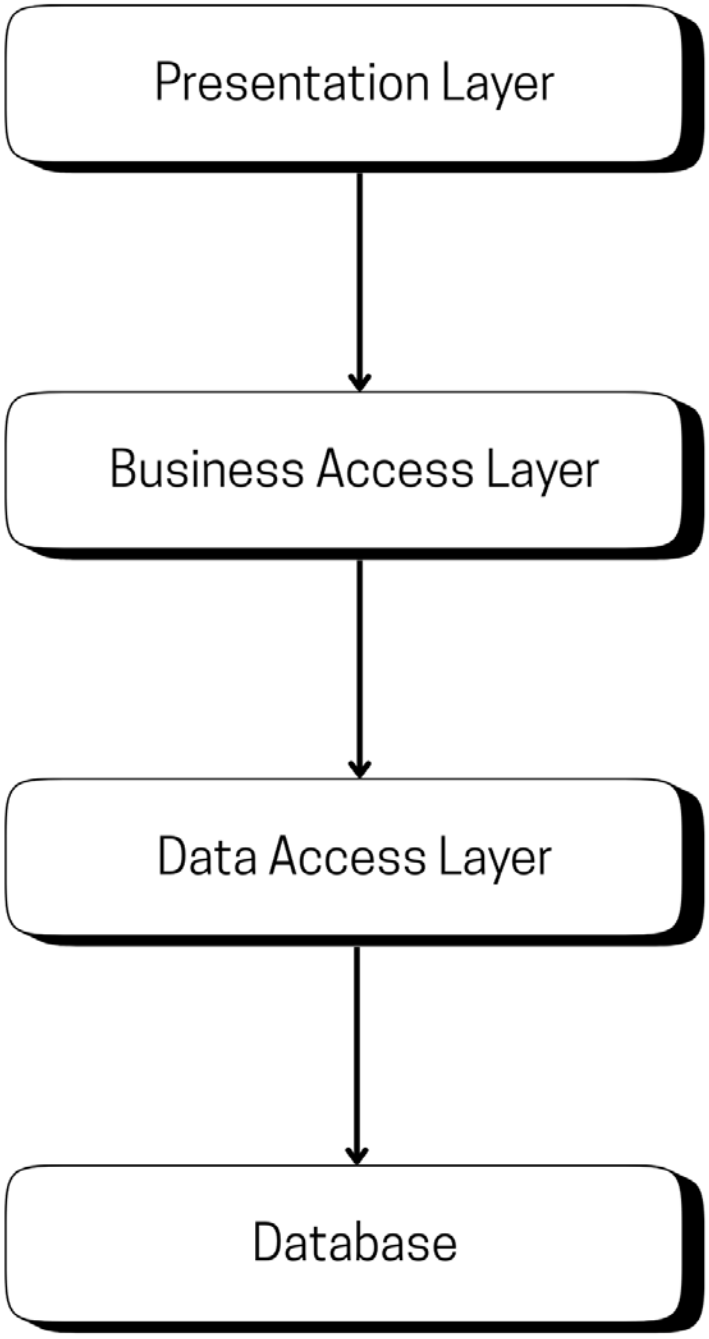


Figure 3.8: Layered Architecture

Since the Fuel Management System is a large and complex system, it uses a layered architecture to keep the code clean and manageable. So in order to add these features, I used the layered architecture. For each filter, I created request parameters in the controller. It calls the service layer to access the request which in

turn calls the data access layer to get the data from the database. The data access layers also uses a utility method in order to reuse the logic for adding parameters, making the code cleaner and easier to maintain. The following code shows this procedure.

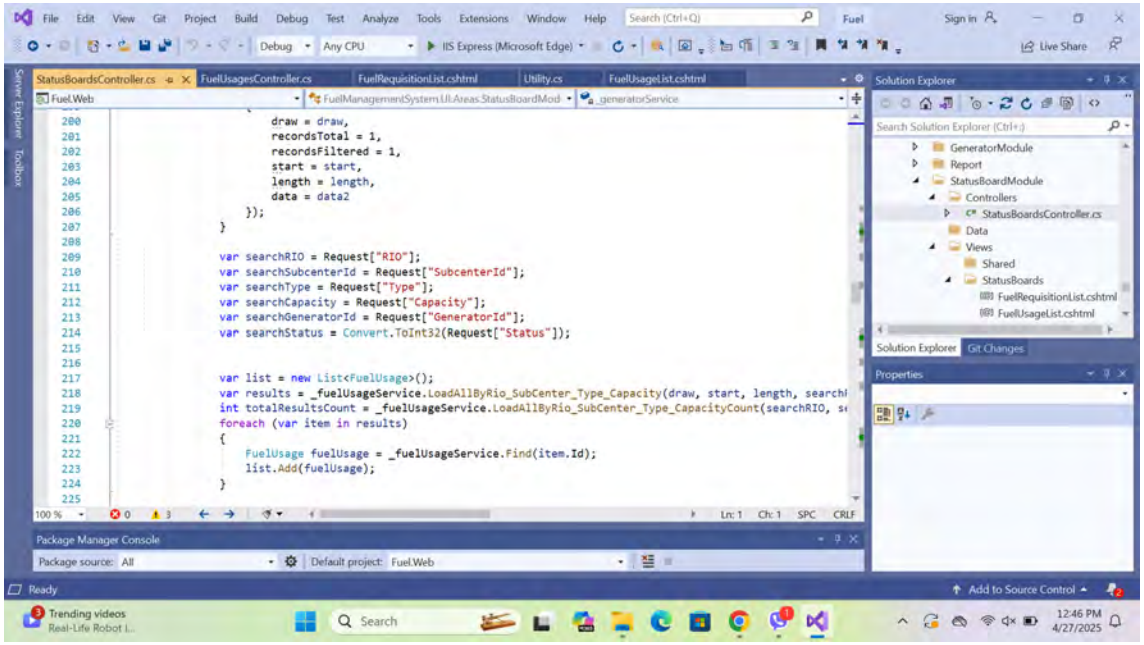


Figure 3.9: Controller snapshot

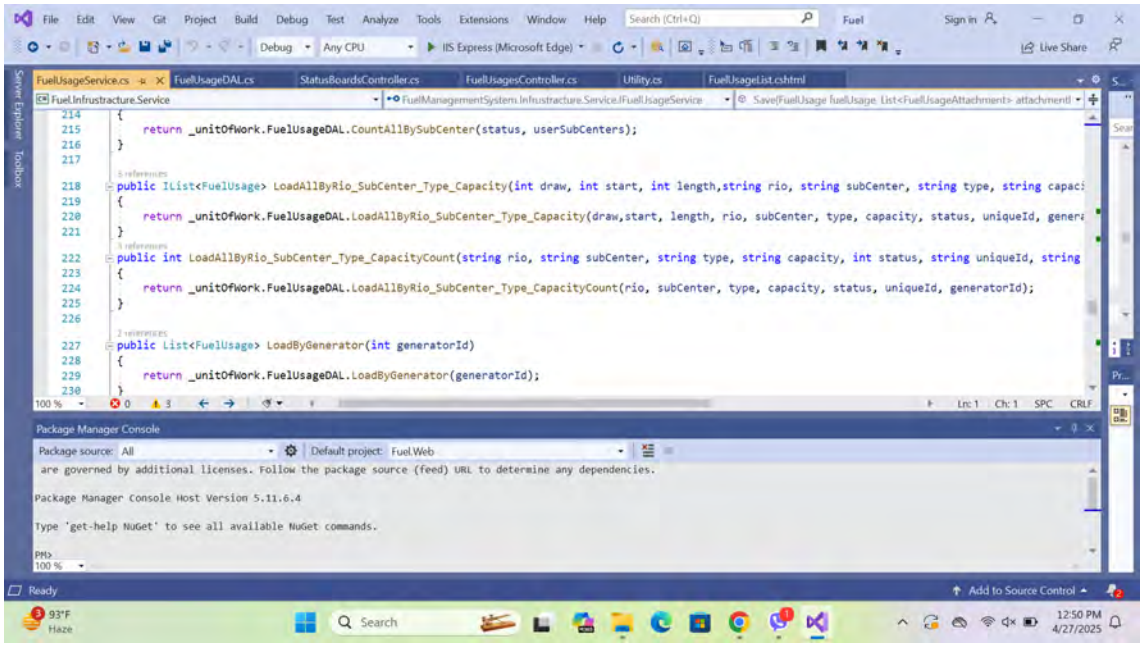


Figure 3.10: Service Layer

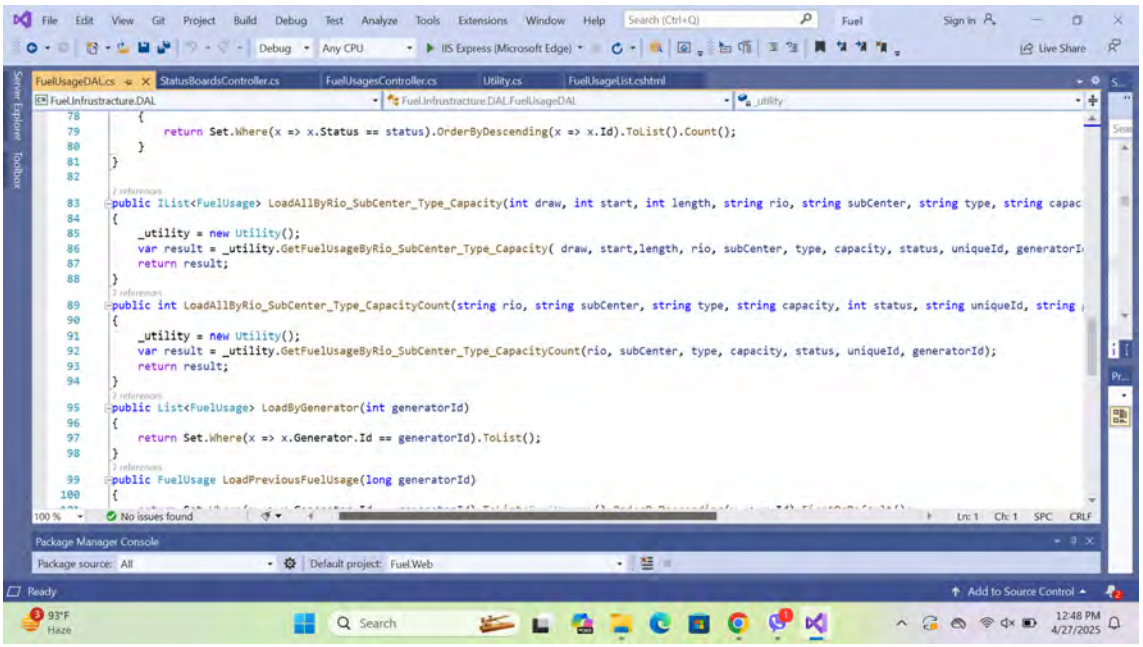


Figure 3.11: Data Access Layer

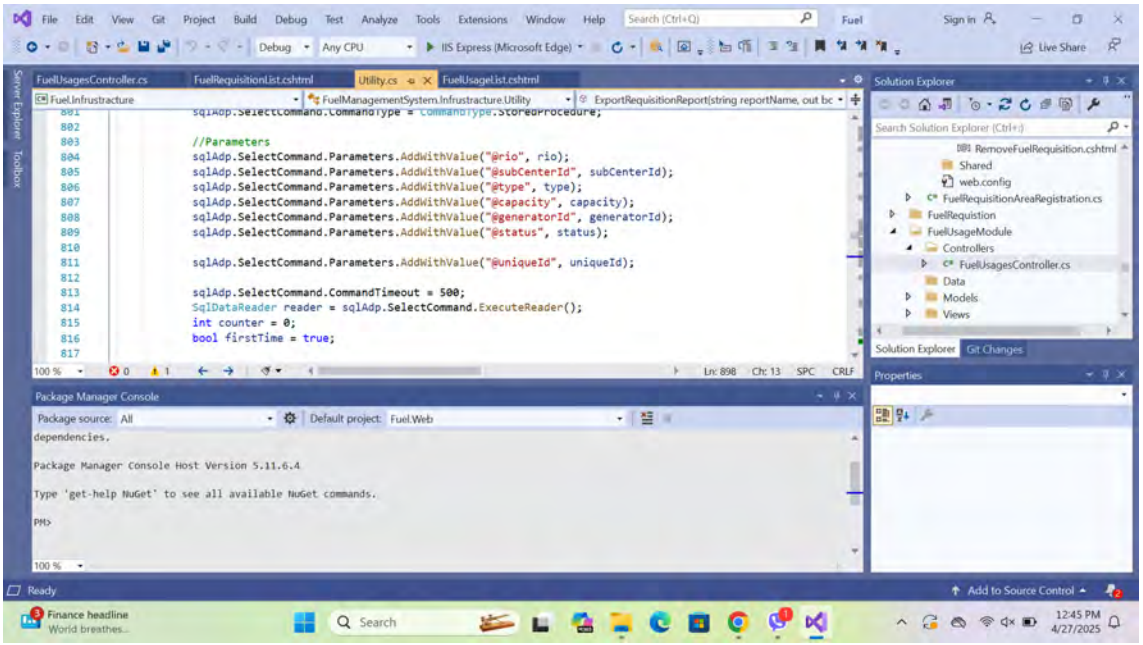


Figure 3.12: Utility

I used a utility method to add SQL parameters. This allows the reuse of the logic for adding parameters, and makes the code cleaner and easier to maintain. Not only that, it also makes sure the parameter values are handled in a consistent way throughout the project.

I created test cases to test the features and fix bugs. I added the features to both fuel usage as well as fuel requisition modules.

The next feature I added was an export functionality to export the fuel usage report. I used AJAX for this task where when user clicks export button, they send an AJAX

request asking for Excel data. The server generates the data and returns the excel file as a downloadable response.

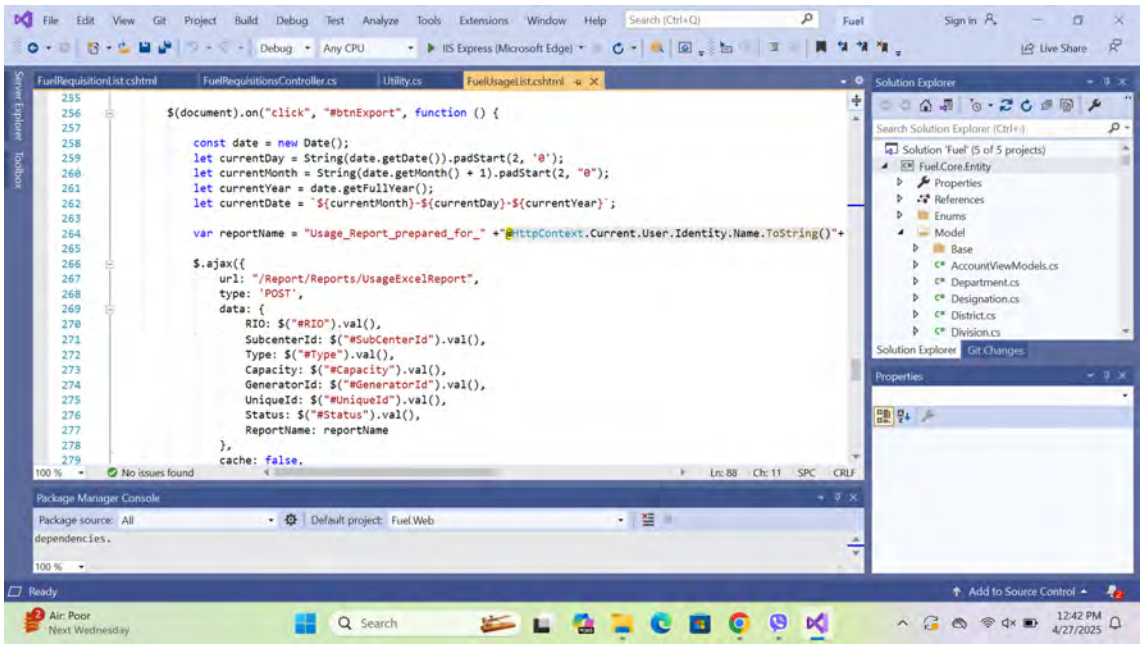


Figure 3.13: Export feature using AJAX

3.3 Selenium and Manual Testing Tasks

During my time at Summit Communications I conducted some manual testing for their Inventory Management System, before it went live. This involved checking if the features worked, adding items, checking amount, whether they are sent properly. I also checked the admin side to see if the requests came through successfully and if there were any issues with dealing with the requests. Through this, I have developed a better understanding of the testing process that a software has to go through before and even after launch.

I also explored automation testing using Selenium. This was a great learning experience as I learned how to locate web elements and automate simple tasks on the web browser. This has helped me understand how automated tests and greatly improve testing efficiency. I hope to apply this knowledge in future projects and dive deeper into automation testing.

Chapter 4

Growth

4.1 Challenges

During my internships I have faced certain challenges that have helped me grow both technically as well as professionally. Since I was not familiar with C sharp or ASP.NET, I had to spend a considerable amount o time learning the syntax and frameworks. I had a difficult time trying to understand how the implementation of model view controller and layered architecture works. However, I was able to manage it and grow more confident over time under the guidance of the senior developers.

4.2 Communication and Time Management

My time at Summit Communications has helped me develop not just technical skills but soft skills, such as time management, as well. There were times when I had to juggle multiple tasks at a time. Although my supervisor was not strict about deadlines, I did my best to complete my tasks in the given time. It got me to stop procrastination and effectively plan my workday.
I have always struggled greatly with communication. However, this experience has helped me improve my communication skills and work effectively in a group setting. It has helped me to not only listen attentively but also to speak up and share my ideas.

4.3 Technical Growth

This internship has helped me to significantly improve my technical skills. I learned C sharp and ASP.net and applied them in building and maintaining web applications. I have gained practical experience in server-side programming using MVC architecture. There was a lot to about writing SQL queries as well and it has helped me write better queries to interact the databases more efficiently.
Not just with the backend, I got to learn a lot about front-end development as well. For front-end I worked with html,css and Javascript. I did not have a lot of Javascript knowledge prior to this. However through this internship, I have learned to create responsive and user-friendly UIs. This internship has provided me with solid knowledge about full-stack development as well as real-world exposure to how software development works.

Chapter 5

Conclusion

5.1 Summary

My internship began on the 5th of November of the year 2024. What began as a 3 month contract, was then extended two and a half more months. This report has been an summary of all I have done and learned throughout my internship at Summit Communications Limited.
I have seen how the cycle of software development works in a real world scenario. Although I have always been more focused on the programming aspect, the little experience with testing that I have had here leads me to see QA testing possibilities as well. I have always had an interest in web development and the internship has shown me what the industry is like. Everything that I have studied during my academic years, I got to actually impplement it in a real world scenario. I hope to implement all that I have learned into future work and continue to grow in the role of a web developer

Bibliography

[1] Microsoft Docs, "ASP.NET Core Documentation," 2024. eprint: https://learn.microsoft.com/en-us/aspnet/core/. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/.

[2] Microsoft Docs, "Entity Framework Core," 2024. eprint: https://learn.microsoft.com/en-us/ef/core/. [Online]. Available: https://learn.microsoft.com/en-us/ef/core/.

[3] W3Schools, "HTML, CSS, JavaScript Tutorials," n.d. eprint: https://www.w3schools.com/. [Online]. Available: https://www.w3schools.com/.

[4] Stack Overflow, "Discussions and solutions related to ASP.NET and C#," [Online]. Available: https://stackoverflow.com/.