

Paxcom Internship Report

Internship report submitted in partial fulfillment of the
requirement for the degree of Bachelor of Technology

in

**Computer Science and Engineering/Information
Technology**

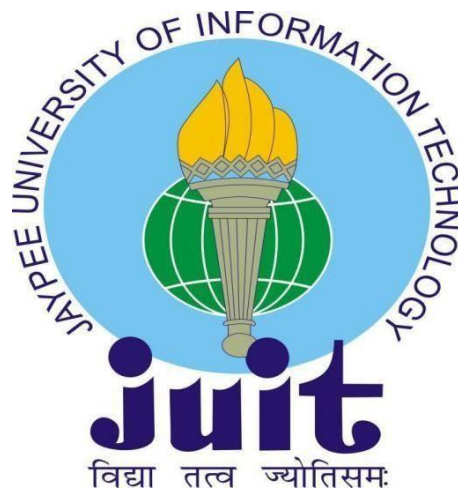
By

Aks Juneja(181261)

Under the supervision of

Dr. Amol Vasudeva

To



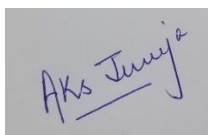
Department of Computer Science & Engineering and
Information Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

Candidate's Declaration

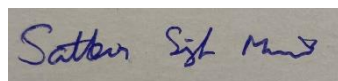
I hereby declare that the work presented in this report entitled “**Paxcom Internship Report**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from February 2022 to June 2022 under the supervision of **Dr. Amol Vasudeva** (Assistant Professor(S.G.), Computer Science Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Aks Juneja, 181261

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



Dr. Amol Vasudeva
Assistant Professor (S.G.)
Computer Science Engineering
Dated: 25-05-2022

Mr. Satbir Singh
Senior Software Engineer
Paxcom
Dated: 25-05-2022

ACKNOWLEDGEMENT

To begin with, I want to express my heartfelt gratitude to almighty God for His divine blessing, which has enabled us to successfully complete the project work.

Supervisor Dr. Amol Vasudeva, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Waknaghat, deserves my deepest gratitude. My supervisor's deep knowledge and keen interest in the field of "Full Stack Development" made it possible to complete this project, as well as his relentless patience, scholarly guidance, persistent and enthusiastic supervision, constructive criticism, insightful advice, and reading many inferior draughts and correcting them at all stages.

I'd like to express my heartfelt gratitude to Dr. Amol Vasudeva of the Department of CSE for his invaluable assistance in completing my thesis.

I'd also like to express my gratitude to everyone who has assisted me in making this project a success, whether directly or indirectly. In this specific circumstance, I'd like to express my gratitude to the numerous staff members, both teaching and non-teaching, who have provided me with valuable assistance and facilitated my project.

Finally, I must express my gratitude for my parents' unwavering and unflagging love, support and utmost patience in putting up with us.

Aks Juneja (181261)

TABLE OF CONTENTS

| Chapter Name | Page No. |
|--|-----------------|
| Candidate's Declaration | i |
| Acknowledgement | ii |
| List of Abbreviations | v |
| List of Figures | vi |
| | |
| Chapter1: Introduction | |
| 1.1 Paxcom | 1 |
| 1.2 Paymentus | 2 |
| 1.3 Description of Industry | 3 |
| 1.3.1How it works | 4 |
| | |
| Chapter 2: OOPS & Data Structures | |
| 2.1 Object Oriented Programming | 6 |
| 2.1.1 Structure of OOPs | 6 |
| 2.1.2 Features of OOPs | 6 |
| 2.2 Data Structures | 7 |
| | |
| Chapter 3: System Development | |
| 3.1 Introduction to C# | 11 |
| 3.1.1 Language Integrated Query(LINQ) | 11 |
| 3.1.2 LINQ Architecture | 12 |
| 3.1.3 Generics in C# | 12 |
| 3.1.4 Delegates in C# | 13 |
| 3.1.5 C# Console Application | 14 |
| 3.2 .NET Framework | 18 |
| 3.2.1 Architecture of .NET Framework | 19 |
| 3.3 Web API | 20 |
| 3.3.1 Web API Project Objective | 21 |
| 3.3.2 Controller | 22 |
| 3.4 Postman | 26 |

Chapter 4: SQL Server

| | |
|--|----|
| 4.1 Introduction to RDBMS | 27 |
| 4.2 Structured Query Language (SQL) | 27 |
| 4.2.1 Application of SQL | 27 |
| 4.2.2 SQL Components | 28 |
| 4.2.3 SQL Architecture | 29 |
| 4.3 Stored Procedures | 30 |
| 4.4 Common Table Expressions (CTE) | 31 |
| 4.4.1 Recursive CTE | 31 |
| 4.5 Pivot | 32 |
| 4.6 Dynamic SQL | 32 |
| 4.7 SQL Server Management Studio | 33 |
| 4.8 Object Relational Model (ORM) | 34 |
| 4.9 Entity Framework Core | 34 |
| 4.9.1 Entity Framework Core Approaches | 35 |
| 4.10 EF Core Database Providers | 36 |

Chapter 5: Front-End

| | |
|----------------|----|
| 5.1 HTML | 37 |
| 5.2 CSS | 38 |
| 5.3 JavaScript | 39 |
| 5.3.1 AJAX | 39 |
| 5.4 TypeScript | 40 |

| | |
|-----------------------------|----|
| Chapter 6 Conclusion | 42 |
| References | 43 |

LIST OF ABBREVIATIONS

SQL- Structured Query Language

EF - Entity Framework

RDBMS - Relational Database Management System

CTE – Common Table Expression

HTML- HyperText Markup Language

CSS- Cascading Style Sheets

JS- Java Script

TS- TypeScript

LIST OF FIGURES

- Figure 1:** Sample Submission File
- Figure 2:** Paxcom Timeline
- Figure 3:** How Paymentus Works
- Figure 4:** Stack Representation
- Figure 5:** Queue Representation
- Figure 6:** Enque
- Figure 7:** Deque
- Figure 8:** Linked List
- Figure 9:** Time Complexity of Linked List
- Figure 10:** C# Logo
- Figure 11:** LINQ Architecture
- Figure 12:** Generics Code
- Figure 13:** Delegates
- Figure 14:** Program.cs file
- Figure 15:** Customer Class
- Figure 16:** Data Model
- Figure 17:** ProcessInitiator.cs
- Figure 18:** Tools.cs
- Figure 19:** UserInterface.cs
- Figure 20:** Output
- Figure 21:** .NET Framework
- Figure 22:** Architecture of .NET Framework
- Figure 23:** Web API
- Figure 24:** API Architecture
- Figure 25:** N Layer Architecture
- Figure 26:** Customer Controller
- Figure 27:** Customer BLL
- Figure 28:** Customer DAL
- Figure 29:** Startup.cs
- Figure 30:** CustomerBookings Table
- Figure 31:** Customers Table
- Figure 32:** Http Client Server Model

Figure 33: Http Methods

Figure 34: Check endpoint of Get API request using Postman

Figure 35: SQL Components Processing

Figure 36: Stored Procedure

Figure 37: CTE

Figure 38: Recursive CTE

Figure 39: Pivot

Figure 40: SQL Server Management Studio Environment

Figure 41: Model of Code First Approach

Figure 42: Model Of Database First Approach

Figure 43: Design of Customer Table

Figure 44: Customer Class automatically generated by EF Core

Figure 45: EF Core Dependencies

Figure 46: HTML Page Structure

Figure 47: FlexBox Structure

Figure 48: Compilation process of TypeScript

CHAPTER 1: INTRODUCTION

1.1 Paxcom



Figure 1: Paxcom Logo

We are a team of 200+Ecommerce enthusiasts, passionate about using technology to simplify Digital Commerce and Payments for brands across the globe. Paxcom is a part of the Paymentus group - a leading global paperless electronic billing and payment solution provider.



Figure 2: Paxcom Timeline

Paxcom characterized way empowers the making of a uniform code base over an association, without agonizing over characterizing certain principles.

Angular as of now accompanies these principles out of the crate. Tailing them doesn't just build the consistency and hence the nature of the code. It makes your application more

obvious, as well. That is, in the event that you are as of now acquainted with angular. This exacting methodology likewise proves to be useful across collaboration fringes. It empowers new engineers to incorporate into another group rapidly, as a result of the high commonality with the code.

What I need to state is, you should follow the angular structure rules to capitalize on the angular system. It will make your life significantly simpler, when coming into new undertakings and will expand the nature of your code consequently.

A typical activity, when learning angular is to put everything into the application module. Obviously, that your application will advance into a total chaos. Modules are there which is as it should be!

Modules help to sort out your code into littler groups to make discovering things simpler. Yet, they are not just a corrective thing. With the assistance of lethargic stacked modules, you can likewise build the client experience by just downloading the pieces of the application, that are required at that point.

Angular is written in typescript by Google development team and is maintained by them regularly. It's first initial release was in beginning of 2016. Very first version was called as Angular-JS so in future updates to make it distinguish from other they named it to just Angular and removed the JS. 11 stable versions has been released by the google team along with the community support as of date 11 Nov 2020. Todo material design in angular, it provides angular material library.

1.2 Paymentus

Paymentus is a North Carolina based software company providing complete billing solutions.

I worked at Paxcom India Pvt Ltd at Gurugram branch. I worked with the OmniChannel Team and was successfully able to understand their working structure and pattern. I also learned soft skills like communicating within a corporate firm and working with a team. I

was successfully able to understand various coding paradigms used by a company to build its application. I got a thorough understanding of some of the company's existing products and some of the upcoming products.

The working culture of the company is great. I thoroughly enjoyed myself working there. Paxcom has atypical blend of work and fun. Although I didn't get to spend much time in the company office and started working from home after the coronavirus pandemic lockdown, I really enjoyed the weekend football and various parties at the company. And even after the lockdown the communication between me and my team was good through various meeting platforms like skype and google meets.

There are numerous things I love about my internship - the experience/information I've acquired, the balance between fun and serious activities climate, my director and colleagues, and surprisingly the gathering of people I exercise with on my mid-day breaks. In any case, in the event that I needed to put it on a certain something, what I like most is, at last, the way that this chance truly opened the entryway into my profession. The most stunning thing about this temporary position experience is that it truly overcame any barrier between learning things and really applying a portion of this information into a reality, and getting paid for it! The way that I presently have an entry-level position straightforwardly identified with what I concentrate on it makes me study more diligently and makes me need to sort out how I can apply a greater amount of what I'm realizing in an internship. It's interesting, in light of the fact that things I've learned in college help me at work.

1.3 Description of Industry:

In 2004, Paymentus was born from a desire to improve the way bills get paid. Vision, innovation and exemplary service have propelled Paymentus to become the leading paperless electronic billing and payment solution on the market, resulting in 1,300 clients including some of the largest billers in NorthAmerica. We know that in order to keep our solutions current and relevant, we need people with the know how, drive and proclivity for fostering a supremely Happy customer experience. Our highly committed, creative employees turned an Idea into a secure, SAAS based Customer Engagement and Payment Platform; one that enables direct bill organizations to provide a unified customer experience

and boost adoption of cost-saving electronic billing and payment services.

Recognized by Deloitte to be among the fastest growing North American companies in 2011, 2013, 2014, and 2016, Paymentus consistently strives to develop better, faster, more secure, cost-efficient billing and payment platforms. We continually seek higher value for our customers, in both solutions and service. It's what has led to our remarkable growth in the last decade. We succeed when our clients succeed. They succeed when their customer relationships are enhanced and, in turn, their customers participate in these costs having electronic services at high rates.

1.3.1 How Paymentus Works:

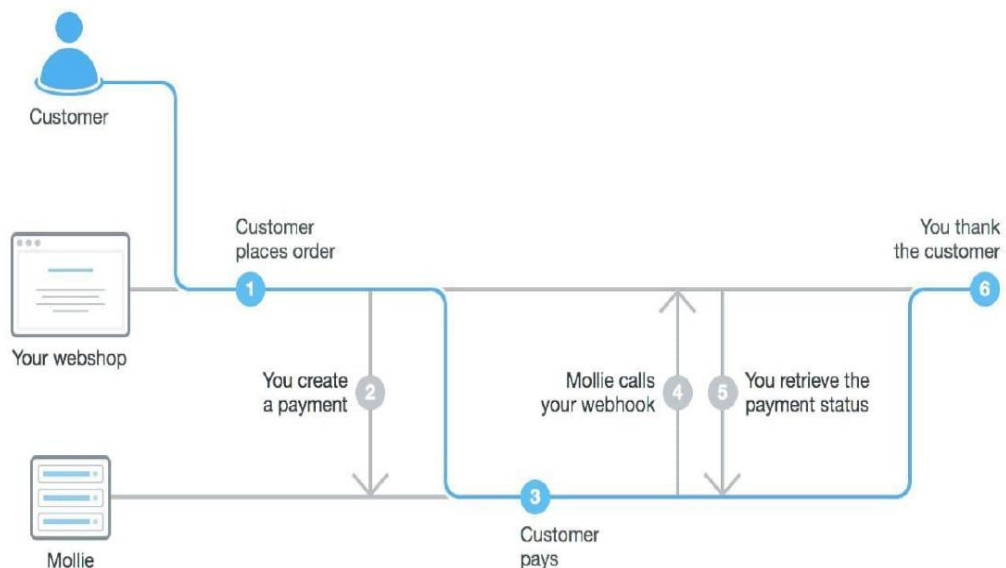


Figure 3: How paymentus works

How to Pay:

Input the URL of the organization you want to pay into your web browser and navigate to the billing section to begin the payment process.

Unsure of the URL?:

It is often found on your paper billing statement. If not, look-up the webpage via your preferred search engine (Google, Bing, etc.).

CHAPTER 2 : OOPs AND DATA STRUCTURES

2.1 Object Oriented Programming

Object-oriented programming (OOP) is a programming paradigm which does not have a major focus on functions and logic, but is more concerned about the functionality that revolves around data and objects.

It merely focuses on the logic required for the management, but strongly on the objects. For the projects and ideology that is large and is prone to update, the developers prefer this method of programming. If the work is concerned with the production systems, designing, or mobile applications, the developers aim at working on the principles and features of OOPs.

2.1.1 Structure of OOPs

1. **Classes:** It is just a blueprint of the methods, attributes and objects. It is considered to be a logical entity and used to define the states of variables, methods and attributes.
2. **Objects:** Object is an actual instance of a class that implements the traits of the methods and attributes of the class. It is an entity that does not exist logically but has a structure and an existence.
3. **Methods:** These are the functions inside a class to describe the behaviors of objects. They are meant to perform certain tasks.
4. **Attributes:** Attributes are needed to store the information and data. Inside a class, they are used to represent the state and quality of the objects.

2.1.2 Features of OOPs

1. **Encapsulation:** As the name suggests, encapsulation means to put the content of the class into a capsule. The important information is bound inside it and all the unnecessary details are hidden from the user.

a) Data Binding: Binding the information, attributes, methods and data inside a class.

b) Data Hiding: Hiding the unnecessary details from the user.

Its major focus is to provide security.

2. Abstraction: This is a feature which is intended to not disclose the backend functionality and details to the user, as it is totally not needed.

3. Inheritance: Inheritance is a feature by virtue of which classes can inherit the properties of some other classes. The classes which inherit the properties are called the derived classes and whose properties are inherited are called the base classes.

4. Polymorphism: Polymorphism is the process by which a piece of code, data, technique, or object acts differently in different situations.

a) Compile-time Polymorphism: Compile time polymorphism, commonly referred to as static polymorphism, is a sort of polymorphism that occurs during the compilation process. The compiler determines the shape or value that the entity in the image must take.

b) Run-time Polymorphism: Runtime polymorphism, also known as Dynamic Polymorphism, is a sort of polymorphism that occurs while the programme is running. It signifies that the compiler cannot decide. As a result, the execution determines what shape or value must be taken.

2.2 Data Structures

Data Structures is a medium of storing data and organizing it. It is a way of arranging data on the system in order to have an efficient access and update. During the internship, before we could actually move to the major work ahead, our mentor wanted us to brush up our skills on Data Structures.

During the course of our proceedings we were not asked to go through all the Data Structures, just a few of them that were needed for our further meetings.

1. Arrays: Arrays are the Data Structures that enable contiguous memory allocation. It follows an index based allocation, which makes it easier to access an element. All the elements have the same data type.

2. Stack: Stack are the special Data Structures that follow the Last In First Out principle. It is a linear data structure.

It is a linear data structure.

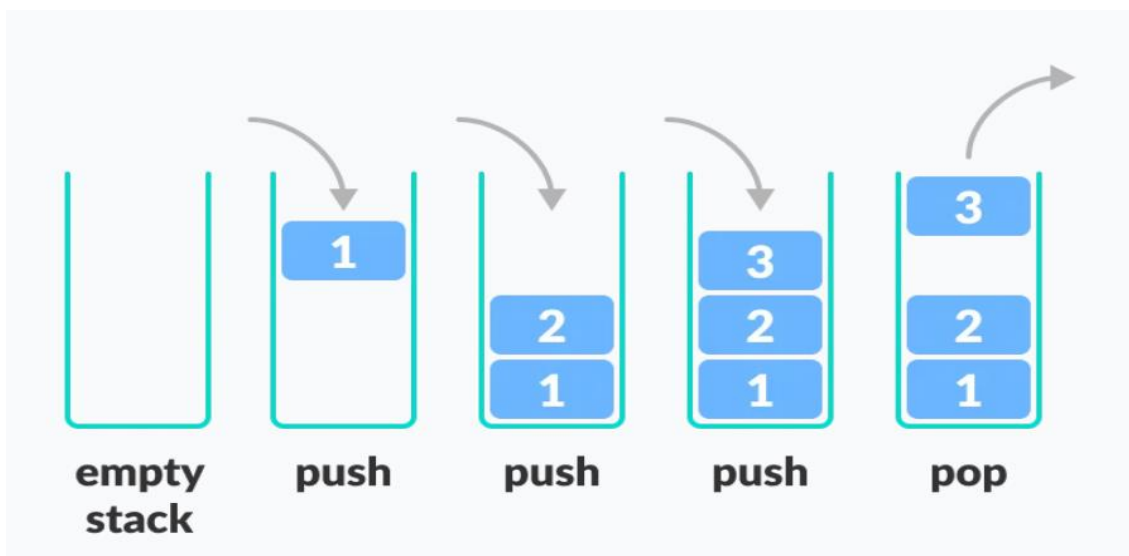


Figure 4: Stack Representation

3. Queue: Queue Data Structure follows the First In First Out principle. It is a linear Data Structure.

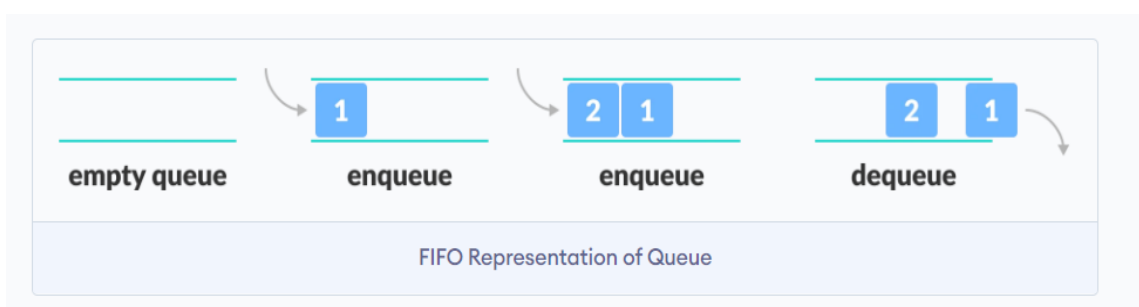


Figure 5: Queue Representation

The two operations are :

a) Enqueue

b) Dequeue

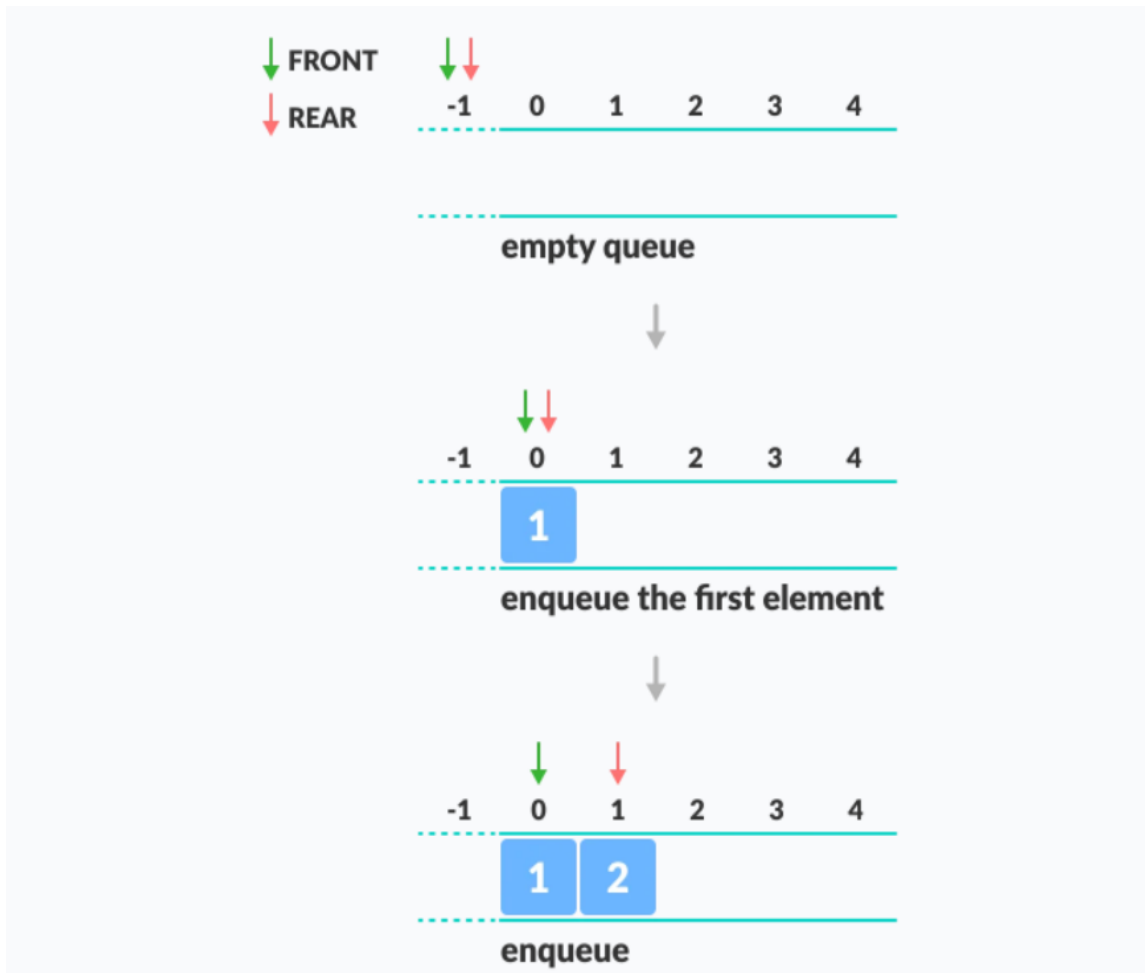


Figure 6: Enque

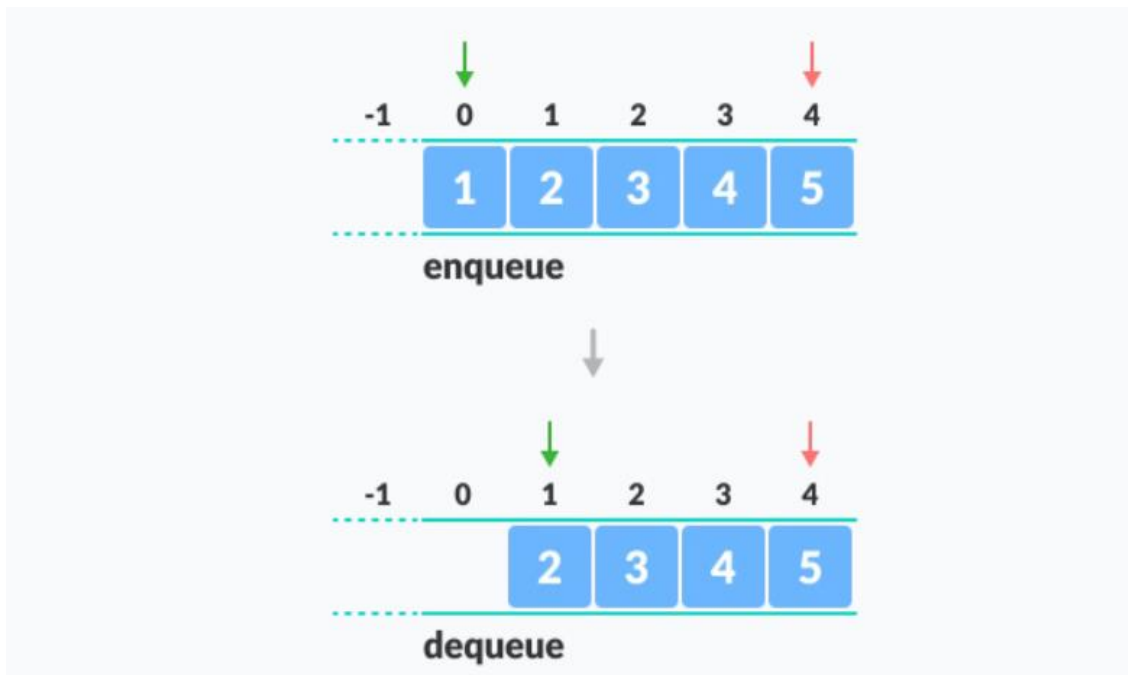


Figure 7: Dequeue

4. **Linked List:** This is a linear Data Structure in which elements are connected to each other in a series of nodes. A node contains two important pieces of information, the value and the address of the next node.

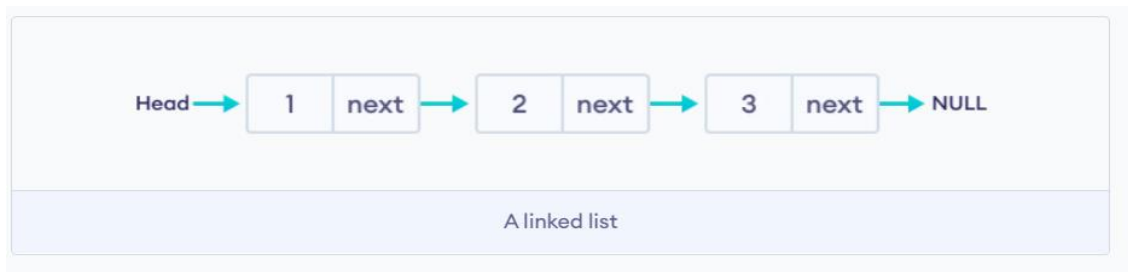


Figure 8: Linked List

Time Complexity

| | Worst case | Average Case |
|-----------------|------------|--------------|
| Search | $O(n)$ | $O(n)$ |
| Insert | $O(1)$ | $O(1)$ |
| Deletion | $O(1)$ | $O(1)$ |

Figure 9: Time Complexity of Linked List

Chapter 3: C# and .NET Framework

3.1 Introduction to C# :

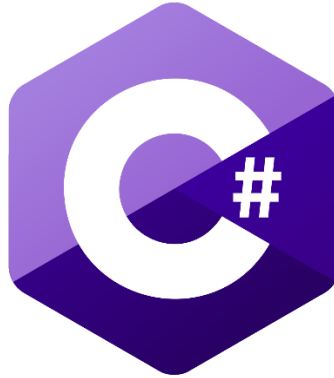


Figure 10: C# Logo

C# is a programming language specially developed by Microsoft for the major purpose of working on .NET framework. It can be used for developing web, desktop and mobile applications.

We were made to brush up on our knowledge and the basic topics of C# including variables, data types, type casting, operators, loops, conditional statements etc. We would not prefer to mention them in detail in this report, but majorly focus on the latest concepts that we learnt during the internship.

3.1.1 Language Integrated Query (LINQ):

Language Integrated Query (LINQ) is referred to as a set of technologies which integrates the query capabilities into the C# language. Earlier, the developer was supposed to learn and understand different query languages for different data sources such as SQL databases, XML, web services etc.

LINQ is capable of providing numerous points of advantages to the users namely, filtering, grouping the operations, ordering and all with a minimum piece of code. In order to query and transform the data in SQL database, XML documents and streams, .NET collection, LINQ prefers a basic query expression pattern.

3.1.2 LINQ Architecture:

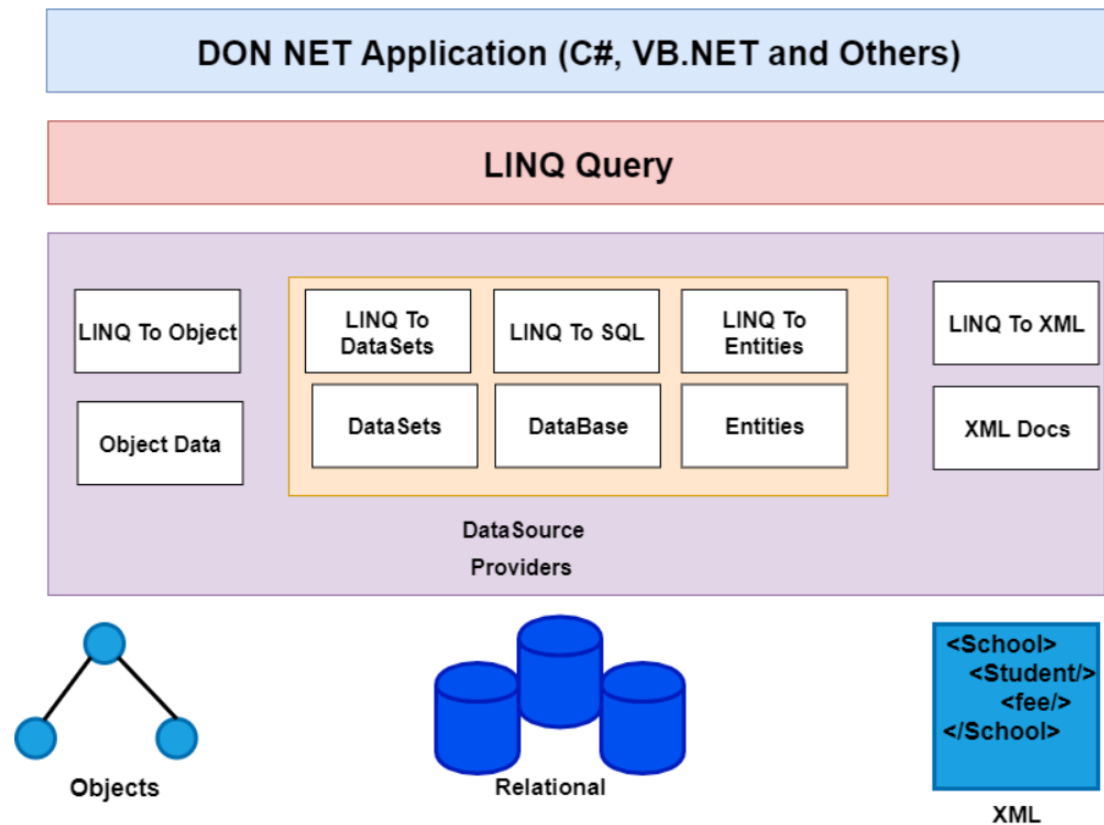


Figure 11: LINQ Architecture

3.1.3 Generics in C# :

In order to defer the specifications of more than single data types, the concept of Generics was made into practice. By using a generic type parameter T, a single class is written that other client code can use without incurring the cost or risk of runtime casts or boxing operations.

By using Generics, the functionalities like reusability, type safety, and efficiency is obtained.

```

using System;
namespace CSharpProgram
{
    class GenericClass<T>
    {
        public GenericClass(T msg)
        {
            Console.WriteLine(msg);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            GenericClass<string> gen = new GenericClass<string> ("This is generic class");
            GenericClass<int> genI = new GenericClass<int>(101);
            GenericClass<char> getCh = new GenericClass<char>('I');
        }
    }
}

```

Figure 12: Generics Code

C# compiler is smart to replace the placeholder at the times, with the specific data types accordingly. The concept of generics is used to create general purpose classes and methods.

3.1.4 Delegates in C#:

A delegate is a type that holds pointers to methods that have a certain argument list and return type. You can associate a delegate's instance with any method that has a compatible signature and return type when you create it. The method can be invoked (or called) using the delegate instance.

Delegates are ideal for establishing callback methods since they can refer to a method as a parameter. You can create a method in your application that compares two objects. A

delegate for a sort algorithm can use that method. The sort function can be more broad because the comparison code is separate from the library.

Delegate figure

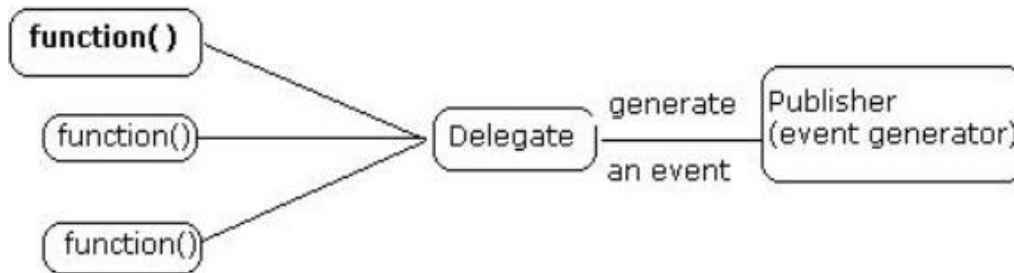


Figure 13: Delegates

A method is typically passed as a parameter to other methods by programmers. Delegates are used for this purpose. A delegate is a class that encapsulates a method signature.

3.1.5 C# Console Application :

In C#, a console program is one that accepts input and displays output on a command line console, with access to three fundamental data streams: standard input, standard output, and standard error.

A console application allows you to read and write characters from the console, either individually or as a line. It's the most basic type of C# application, and it's usually run from the Windows command prompt. A console programme is often a standalone executable file that has little or no graphical user interface (GUI).

Console programmes written in C# feature a single main execution point (static main method), which accepts an optional array of arguments as its only input to represent command-line parameters.

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplications6
7 {
8     4 references
9     class Program
10    {
11        0 references
12        public static void Main()
13        {
14            class ConsoleApplication6.ProcessInitiator
15            {
16                ProcessInitiator processInitiator = new ProcessInitiator();
17                processInitiator.InitiateCustomerOperations();
18            }
19        }
20    }
21 }

```

Figure 14: Program.cs file

Customer Services Class consists of all the function logic to perform different operations on our data.

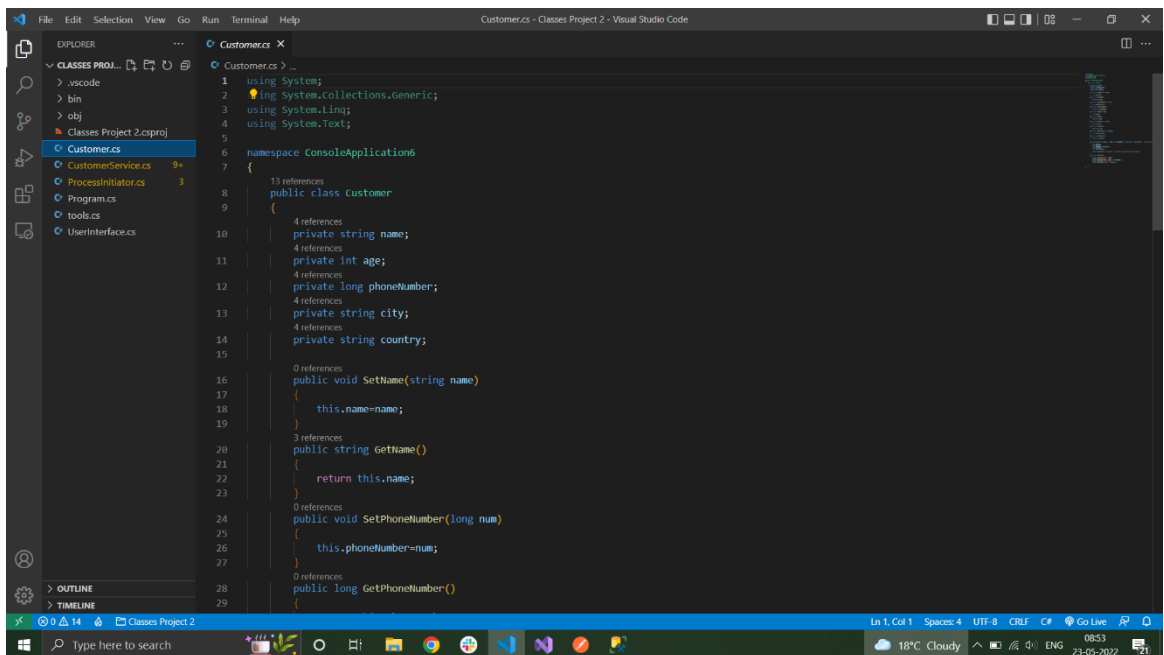
```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplications6
7 {
8     2 references
9     class CustomerServices
10    {
11        1 reference
12        public void AddCustomer(ref List<Customer> CustomerList)
13        {
14            string name,city,country;
15            int age;
16            long phoneNumber;
17            Customer data;
18            Console.WriteLine("Enter customer Full Name: ");
19            name = Console.ReadLine();
20            Console.WriteLine("Enter customer's Age: ");
21            age= Convert.ToInt32(Console.ReadLine());
22            Console.WriteLine("Enter customer's Phone numbers: ");
23            phoneNumber= Convert.ToInt64(Console.ReadLine());
24            Console.WriteLine("Enter customer's city: ");
25            city = Console.ReadLine();
26            Console.WriteLine("Enter customer's country: ");
27            country = Console.ReadLine();
28
29            Customer customerObj = new Customer(name,age,phoneNumber,city,country);
30            CustomerList.Add(customerObj);
31        }
32
33        1 reference
34        public int SearchCustomerByName(ref List<Customer> customerList)
35        {
36            Tools tools = new Tools();
37            Program program = new Program();

```

Figure 15: Customer Class

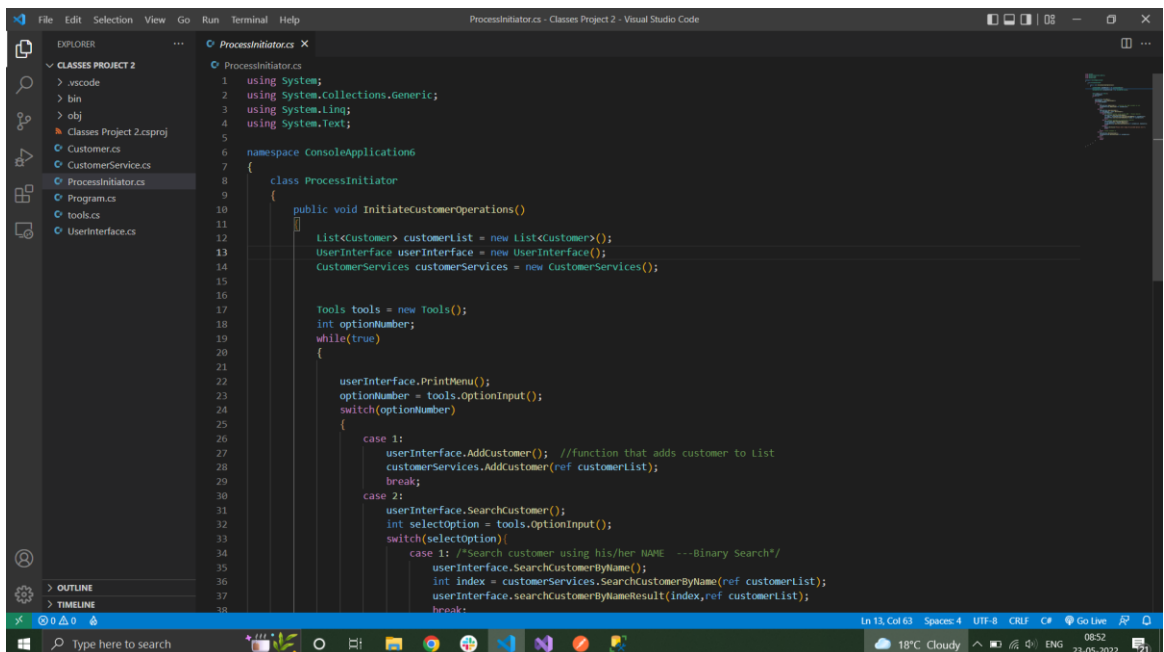
Customer Class serves as our Data Model



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplications
7 {
8     13 references
9     public class Customer
10    {
11        4 references
12        private string name;
13        4 references
14        private int age;
15        4 references
16        private long phoneNumber;
17        4 references
18        private string city;
19        4 references
20        private string country;
21
22        0 references
23        public void SetName(string name)
24        {
25            this.name=name;
26        }
27
28        3 references
29        public string GetName()
30        {
31            return this.name;
32        }
33
34        0 references
35        public void SetPhoneNumber(long num)
36        {
37            this.phoneNumber=num;
38        }
39
40        0 references
41        public long GetPhoneNumber()
42    }
```

Figure 16: Data Model

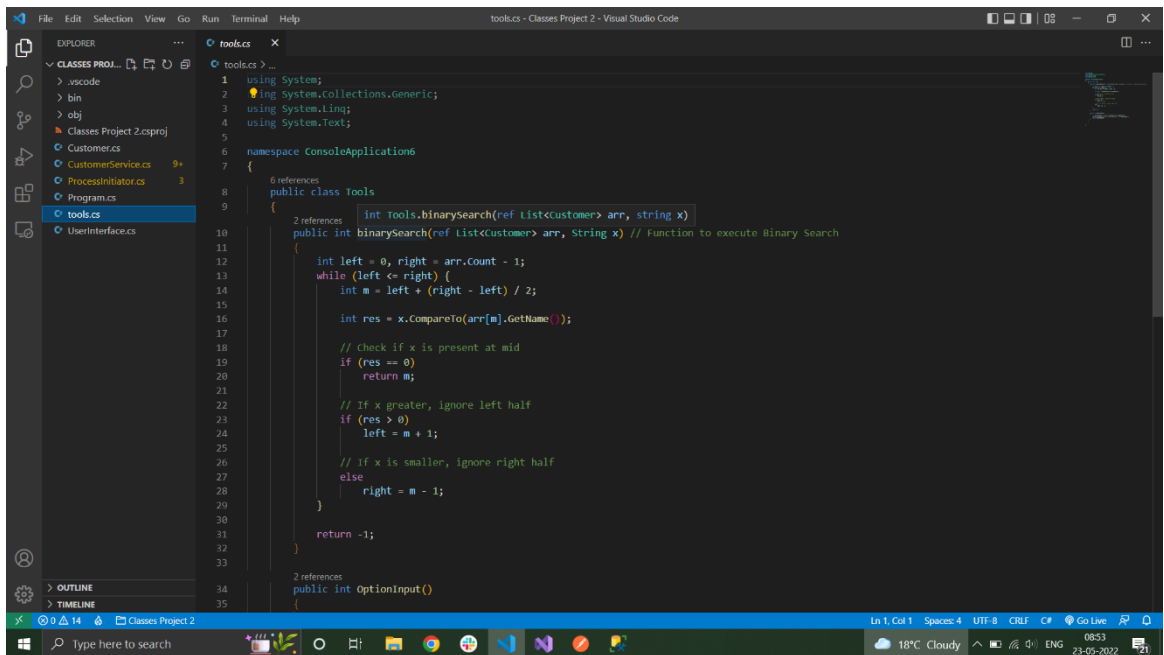
ProcessInitiator.cs: Main function initiates the process Initiator Class which is responsible for all the UI display and I/O operations conducted while the execution of the program.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplications
7 {
8     class ProcessInitiator
9     {
10        public void InitiateCustomerOperations()
11        {
12            List<Customer> customerList = new List<Customer>();
13            UserInterface userInterface = new UserInterface();
14            CustomerServices customerServices = new CustomerServices();
15
16
17            Tools tools = new Tools();
18            int optionNumber;
19            while(true)
20            {
21
22                userInterface.PrintMenu();
23                optionNumber = tools.OptionInput();
24                switch(optionNumber)
25                {
26                    case 1:
27                        userInterface.AddCustomer(); //function that adds customer to List
28                        customerServices.AddCustomer(ref customerList);
29                        break;
30                    case 2:
31                        userInterface.SearchCustomer();
32                        int selectOption = tools.OptionInput();
33                        switch(selectOption)
34                        {
35                            case 1: /*Search customer using his/her NAME ---Binary Search*/
36                                userInterface.SearchCustomerByName();
37                                int index = customerServices.SearchCustomerByName(ref customerList);
38                                userInterface.searchCustomerByNameResult(index,ref customerList);
39                                break;
40                        }
41                }
42            }
43        }
44    }
```

Figure 17: ProcessInitiator.cs

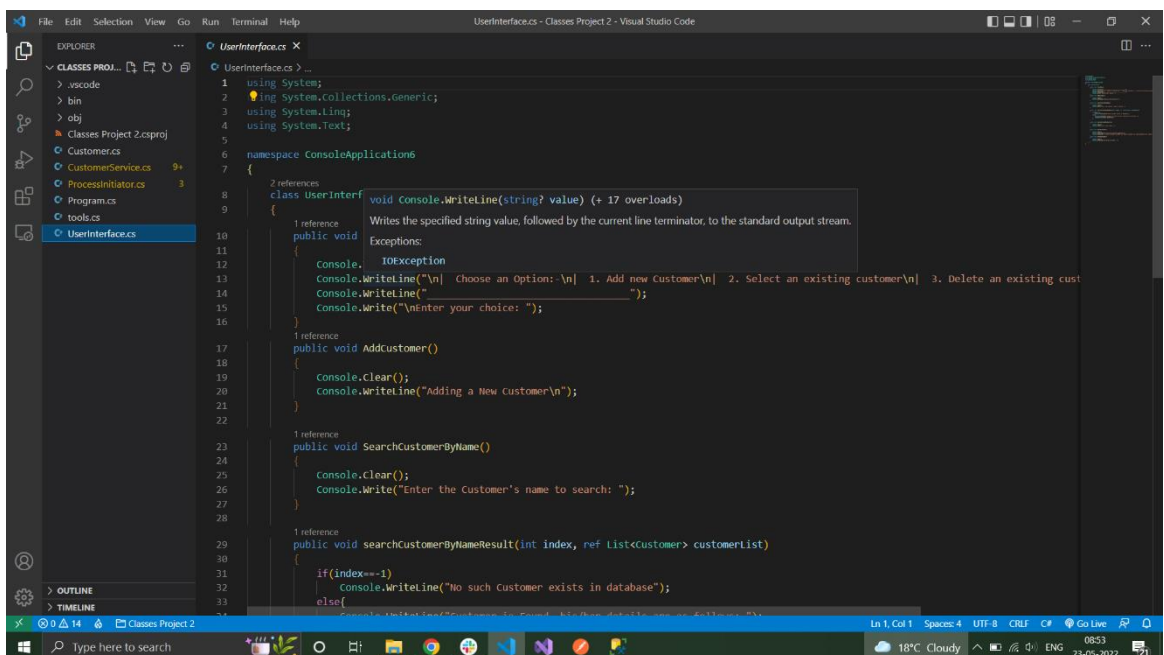
Tools.CS: Tools is a class which contains all the required arbitrary functions which are reused in different operations.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplication6
7 {
8     6 references
9     public class Tools
10    {
11        2 references
12        public int binarySearch(ref List<Customer> arr, string x)
13        {
14            int left = 0, right = arr.Count - 1;
15            while (left <= right) {
16                int m = left + (right - left) / 2;
17
18                int res = x.CompareTo(arr[m].GetName());
19
20                // Check if x is present at mid
21                if (res == 0)
22                    return m;
23
24                // If x greater, ignore left half
25                if (res > 0)
26                    left = m + 1;
27
28                // If x is smaller, ignore right half
29                else
30                    right = m - 1;
31            }
32            return -1;
33        }
34
35        2 references
36        public int OptionInput()
37    }
```

Figure 18: Tools.cs

UserInterface.CS: Its responsible for all the UI display operations, it is instantiated in ProcessInitiator class.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace ConsoleApplication6
7 {
8     2 references
9     class UserInterface
10    {
11        void Console.WriteLine(string? value) (+ 17 overloads)
12        Writes the specified string value, followed by the current line terminator, to the standard output stream.
13        Exceptions:
14        Console.WriteLine("\n| Choose an Option:\n| 1. Add new Customer\n| 2. Select an existing customer\n| 3. Delete an existing customer\n|");
15        Console.WriteLine("");
16        Console.WriteLine("\nEnter your choice: ");
17
18        1 reference
19        public void AddCustomer()
20        {
21            Console.Clear();
22            Console.WriteLine("Adding a New Customer\n");
23        }
24
25        1 reference
26        public void SearchCustomerByName()
27        {
28            Console.Clear();
29            Console.WriteLine("Enter the Customer's name to search: ");
30        }
31
32        1 reference
33        public void searchCustomerByNameResult(int index, ref List<Customer> customerList)
34        {
35            if(index == -1)
36                Console.WriteLine("No such customer exists in database");
37            else
38            {
39            }
40        }
41    }
```

Figure 19: UserInterface.cs

OUTPUT

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication6
{
    2 references
    class UserInterface
    {
        1 reference
    }
}
```

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\ProcessInitiator.cs(41,53): warning CS8600: Converting null literal or possible null value to non-nullable type. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\ProcessInitiator.cs(42,52): warning CS8604: Possible null reference argument for parameter 'keyCountry' in 'void CustomerServices.SearchCustomerByCountry(ref List<Customer> customerList, string keyCountry)'. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\ProcessInitiator.cs(55,25): warning CS0162: Unreachable code detected [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\CustomerService.cs(36,30): warning CS8600: Converting null literal or possible null value to non-nullable type. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\CustomerService.cs(41,62): warning CS8604: Possible null reference argument for parameter 'x' in 'int Tools.BinarySearch(ref List<Customer> arr, string x)'. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\CustomerService.cs(66,27): warning CS8600: Converting null literal or possible null value to non-nullable type. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\CustomerService.cs(67,60): warning CS8604: Possible null reference argument for parameter 'x' in 'int Tools.BinarySearch(ref List<Customer> arr, string x)'. [C:\Users\rnsaini\Desktop\Paxcom work\Customer Management System\Classes Project 2\Classes Project 2.csproj]

```
Choose an Option:-
1. Add new Customer
2. Select an existing customer
3. Delete an existing customer
4. Exit the program

Enter your choice: |
```

```
Adding a New Customer

Enter customer Full Name: Ram
Enter customer's Age: 21
Enter customer's Phone number: 123123123
Enter customer's city: Ghaziabad
Enter customer's Country: India
A new customer is successfully registered in our database

Choose an Option:-
1. Add new Customer
2. Select an existing customer
3. Delete an existing customer
4. Exit the program

Enter your choice: |
```

Figure 20: Output

3.2 .NET Framework :



Figure 21: .NET Framework

.Net is an open source and a cross platform used for different operating systems in order to develop console and web applications. And, .NET Framework is an actual implementation of .NET to develop applications on Windows.

3.2.1 Architecture of .NET Framework :

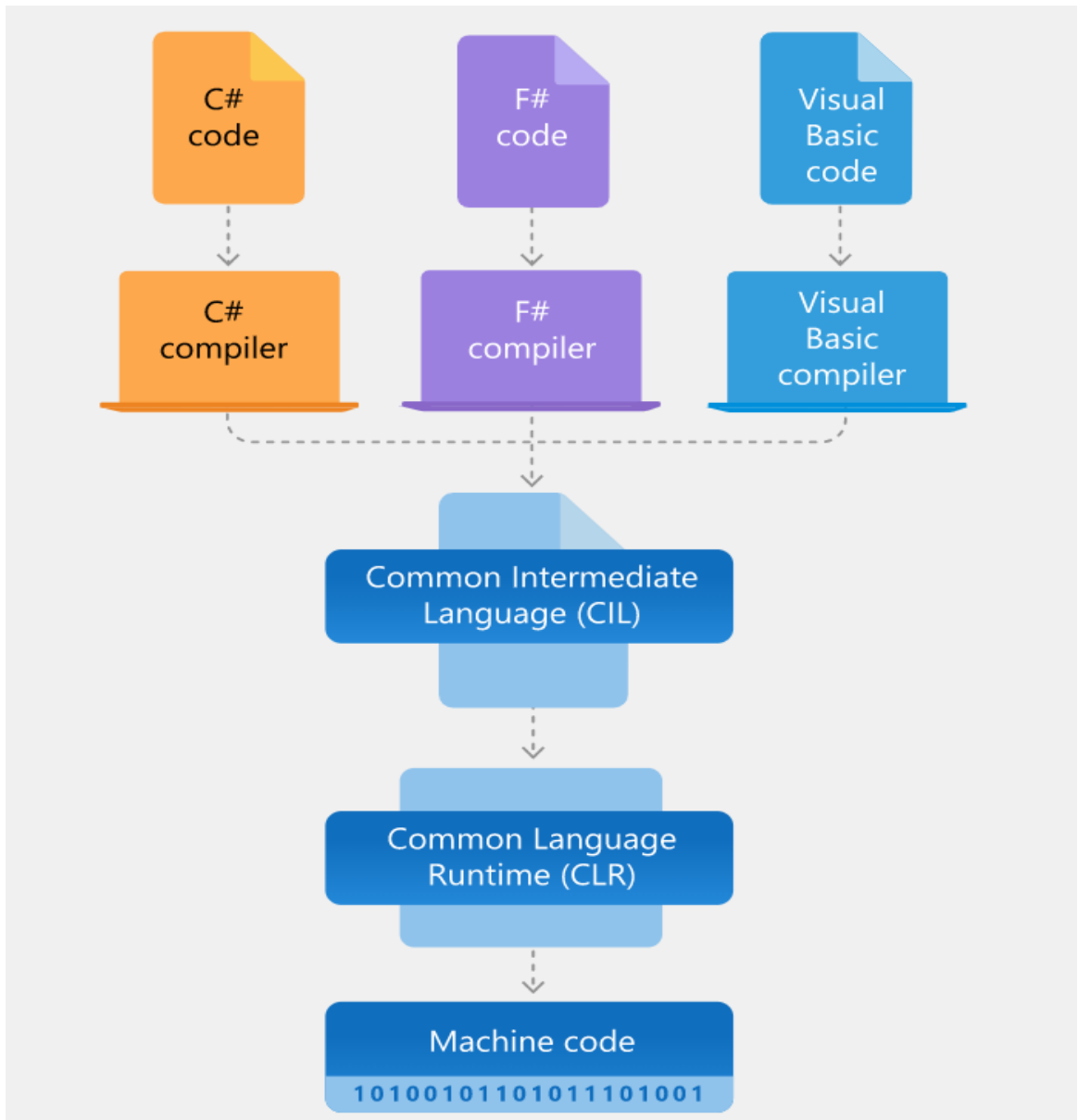


Figure 22: Architecture of .NET Framework

The major components of .NET Framework are:

1. Common Language Runtime (CLR) : The running applications are handled by an engine which is executive in nature. The major beneficiary properties offered by it, includes threading, management, type-safety, and handling the exceptions and errors.

2. Class Library: When the need comes to writing and reading files, connection with the database, or providing types for numbers, dates, strings, the role of Class Library gets into play. Providing a set of Application Interfaces, common functionality types, Class Library performs the job.

The programming languages used for writing the applications in .NET are C#, F# or Visual BASIC. The code compilation is into a language independent and agnostic intermediate language i.e, Common Intermediate Language (CIL) and which is further stored as .dll or .exe files in assemblies.

At the time when the application starts to run, CLR turns the machine code for executing specific architecture with the help of JIT (Just-in-Time) compiler and in the meantime CLR takes the assemblies.

3.3 WEB API:

API stands for Application Programming Interface. It is a software intermediary that allows two or more apps to communicate with one another

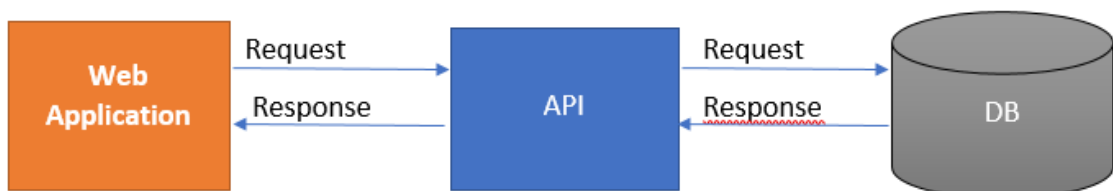


Figure 23:

A web API is an application programming interface for a web application or web server, to put it simply. It communicates with clients and websites using the HTTP protocol for data access.

The Asp.net Core web API is a platform-independent web API.

Why is Web API required?

- The customer wishes to utilize the app on a variety of platforms, including mobile, web, and Google devices. Web API can be handy in this situation.

- Web API receives requests from many devices and responds in JSON format. The majority of gadgets can interpret JSON output.
- Take a look at the web Api Architecture diagram below.

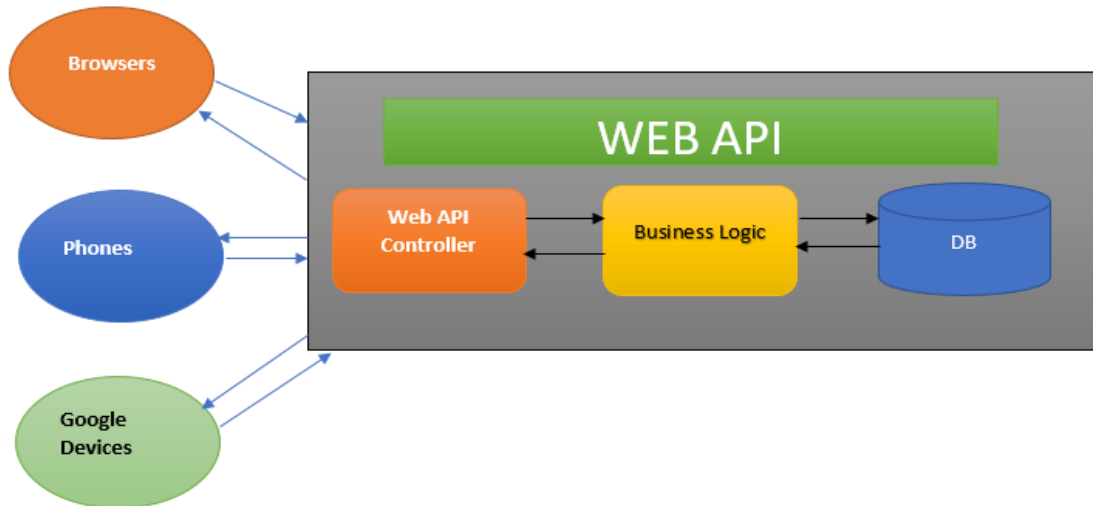


Figure 24:

3.3.1 Web API project Objective:

The objective of this sub project was to convert our previously created C# console application into a .Net based WEB API. This API can handle customer Booking services and operations and stores data into an SQL Server database. This Web API followed N-Layered Architecture

N-Layered Architecture: using N-tier architecture we can divide our project into different layers enabling separation of concerns.

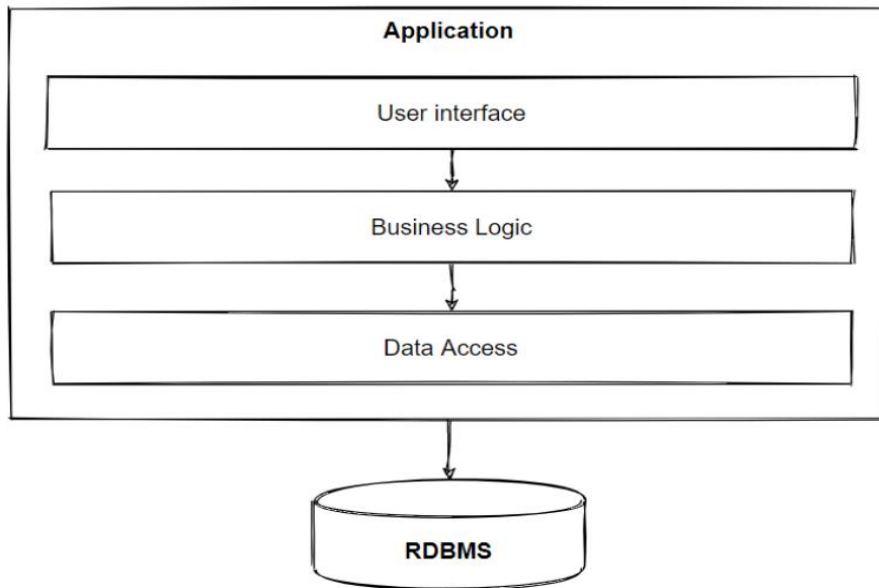


Figure 25: N Layer Architecture

3.3.2 Controller:

A Web API controller is a class that can be built within the Controllers folder or every other location inside the root folder of your project. A controller elegance must have a name that leads to "Controller" and be inherited from the System.Web.Http.ApiController magnificence. Action strategies refer to all the controller's public strategies.

CustomerController has all the movement strategies required by using this WEB API.

```

1  using Microsoft.AspNetCore.Http;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.EntityFrameworkCore;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using WebAPI2704.Models;
8
9  namespace WebAPI2704.Controllers
10 {
11     [Route("Customer")]
12     [ApiController]
13     public class CustomerController : ControllerBase
14     {
15         private BusinessLogicLayer.CustomerBLL _BLL;
16         //Dependency Injection
17         private readonly CustomerDataContext _context;
18         public CustomerController(CustomerDataContext customerDataContext)
19         {
20             _context = customerDataContext;
21             _BLL = new BusinessLogicLayer.CustomerBLL();
22         }
23
24         // GET: api/<CustomerController>
25         [HttpGet]
26         public List<Customers> GetAllCustomers()
27         {
28             return _BLL.GetAllCustomers();
29         }
30
31         // GET api/<CustomerController>/5
32         [HttpGet("GetByName")]
33         public Customers GetByName([FromBody] string keyName)
34         {
35             return _BLL.GetByName(keyName);
36         }
37     }
  
```

Figure 26: Customer Controller

Business Logic Layer: Business Logic Layer is responsible for all the logical operations of WEB API. it consists of different functions which are invoked in the controller. Business Logic Layer further connects to Data Access Layer for storing and retrieving data from our database.

Customer.BLL is an implementation of Business Logic Layer

```
1 using System;
2 using System.Collections.Generic;
3 using WebAPI2704.Models;
4
5 namespace BusinessLogicLayer
6 {
7     public class CustomerBLL
8     {
9         private DataAccessLayer.CustomerDAL _DAL;
10        public CustomerBLL()
11        {
12            _DAL = new DataAccessLayer.CustomerDAL();
13        }
14        public List<Customers> GetAllCustomers()
15        {
16            var data = _DAL.GetAllCustomers();
17            return data;
18        }
19
20        public Customers GetByName(string keyName)
21        {
22            var AllData = _DAL.GetAllCustomers();
23            foreach (var item in AllData)
24            {
25                if (item.Name == keyName)
26                {
27                    return item;
28                }
29            }
30            return null;
31        }
32
33        public IEnumerable<Customers> GetByCountry( string keyName)
34        {
35            List<Customers> customerList = new List<Customers>();
36            List<Customers> outputList = new List<Customers>();
37            customerList = _DAL.GetAllCustomers();
38            foreach (var item in customerList)
39            {
40                if (item.Country == keyName)
41                {
42                    outputList.Add(item);
43                }
44            }
45            return outputList;
46        }
47    }
48 }
```

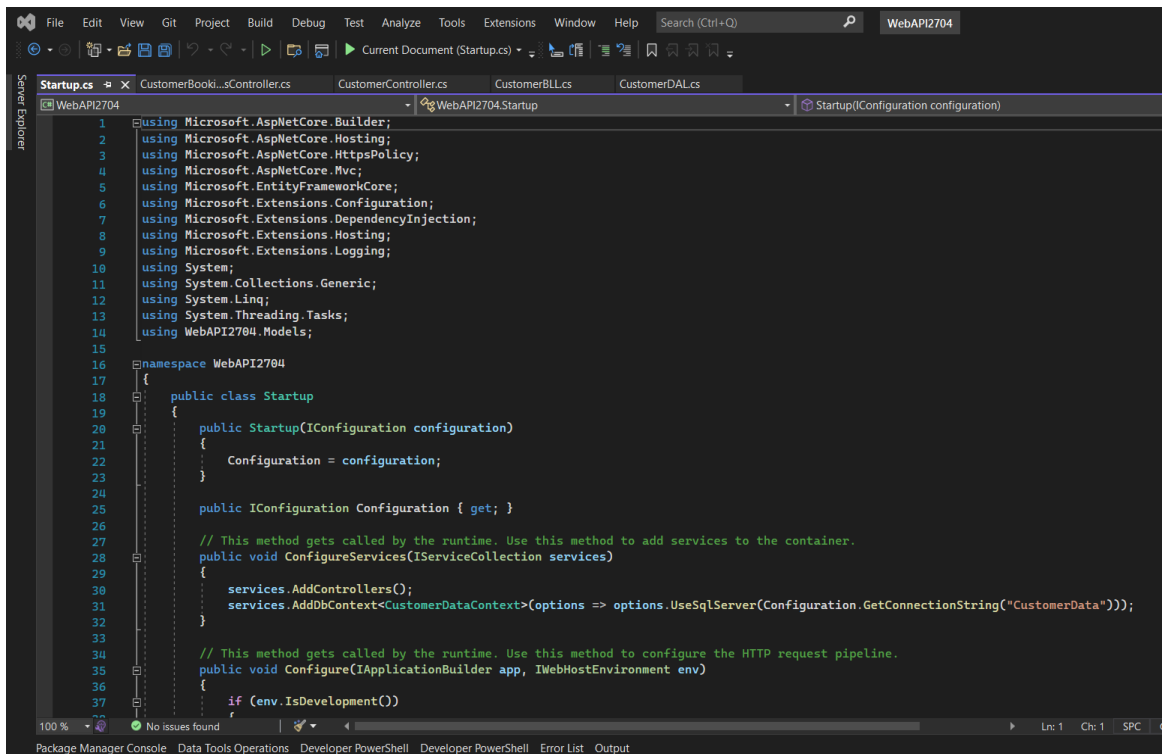
Figure 27: Customer BLL

Data Access Layer: Data access Layer helps in performing all the operations on the database. Custome.DAL is the Data Access Layer of this WEB API.

```
7 namespace DataAccessLayer
8 {
9     public class CustomerDAL
10    {
11        public List<Customers> GetAllCustomers()
12        {
13            var db = new CustomerDataContext();
14            return db.Customers.ToList();
15        }
16
17        public void Create(Customers customerModel)
18        {
19            var db = new CustomerDataContext();
20            db.Customers.Add(customerModel);
21            db.SaveChanges();
22        }
23
24        public void RemoveByName(string keyName)
25        {
26            var db = new CustomerDataContext();
27            var customer = db.Customers.FirstOrDefault(c => c.Name == keyName);
28            if (customer == null)
29            {
30                return;
31            }
32            db.Entry(customer).State = EntityState.Deleted;
33            db.SaveChanges();
34        }
35    }
36 }
```

Figure 28: Customer DAL

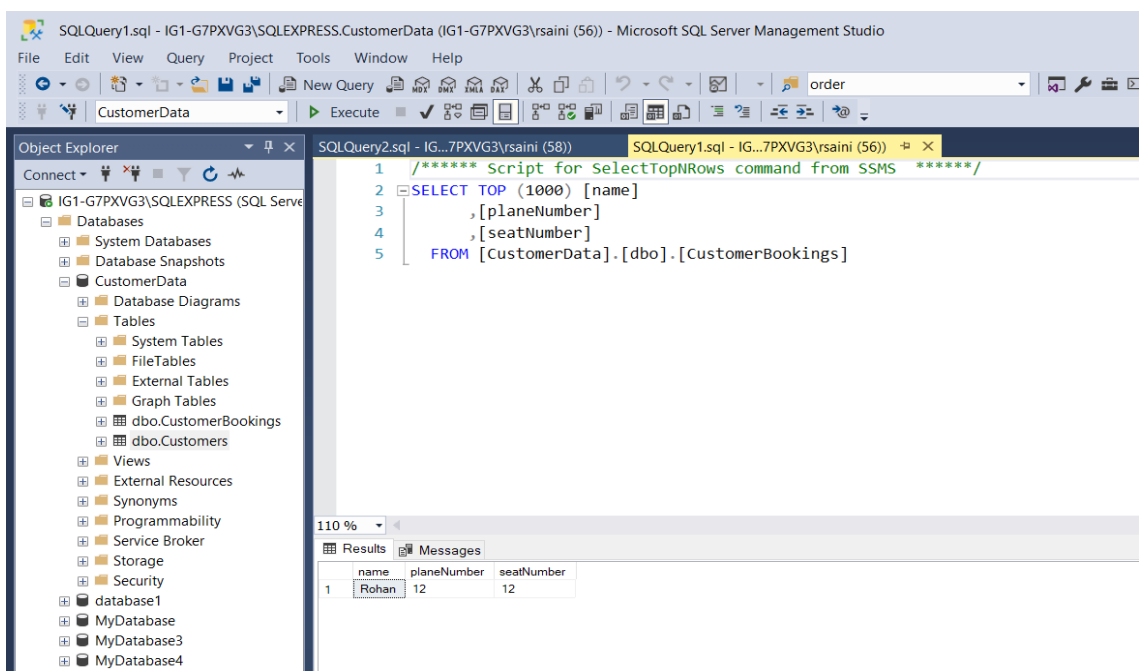
Startup.cs is the root file of this WEB API where the execution of this project begins following pipelining, security features, controller etc.



```
1 using Microsoft.AspNetCore.Builder;
2 using Microsoft.AspNetCore.Hosting;
3 using Microsoft.AspNetCore.HttpsPolicy;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.EntityFrameworkCore;
6 using Microsoft.Extensions.Configuration;
7 using Microsoft.Extensions.DependencyInjection;
8 using Microsoft.Extensions.Hosting;
9 using Microsoft.Extensions.Logging;
10 using System;
11 using System.Collections.Generic;
12 using System.Linq;
13 using System.Threading.Tasks;
14 using WebAPI2704.Models;
15
16 namespace WebAPI2704
17 {
18     public class Startup
19     {
20         public Startup(IConfiguration configuration)
21         {
22             Configuration = configuration;
23         }
24
25         public IConfiguration Configuration { get; }
26
27         // This method gets called by the runtime. Use this method to add services to the container.
28         public void ConfigureServices(IServiceCollection services)
29         {
30             services.AddControllers();
31             services.AddDbContext<CustomerDataContext>(options => options.UseSqlServer(Configuration.GetConnectionString("CustomerData")));
32         }
33
34         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
35         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
36         {
37             if (env.IsDevelopment())
```

Figure 29: Startup.cs

Database: All the bookings and customer related information is processed and stored in the database below, data is also fetched from this database.



```
1 /***** script for SelectTopNRows command from SSMS *****/
2 SELECT TOP (1000) [name]
3     , [planeNumber]
4     , [seatNumber]
5 FROM [CustomerData].[dbo].[CustomerBookings]
```

| name | planeNumber | seatNumber |
|-------|-------------|------------|
| Rohan | 12 | 12 |

Figure 30: CustomerBookings Table

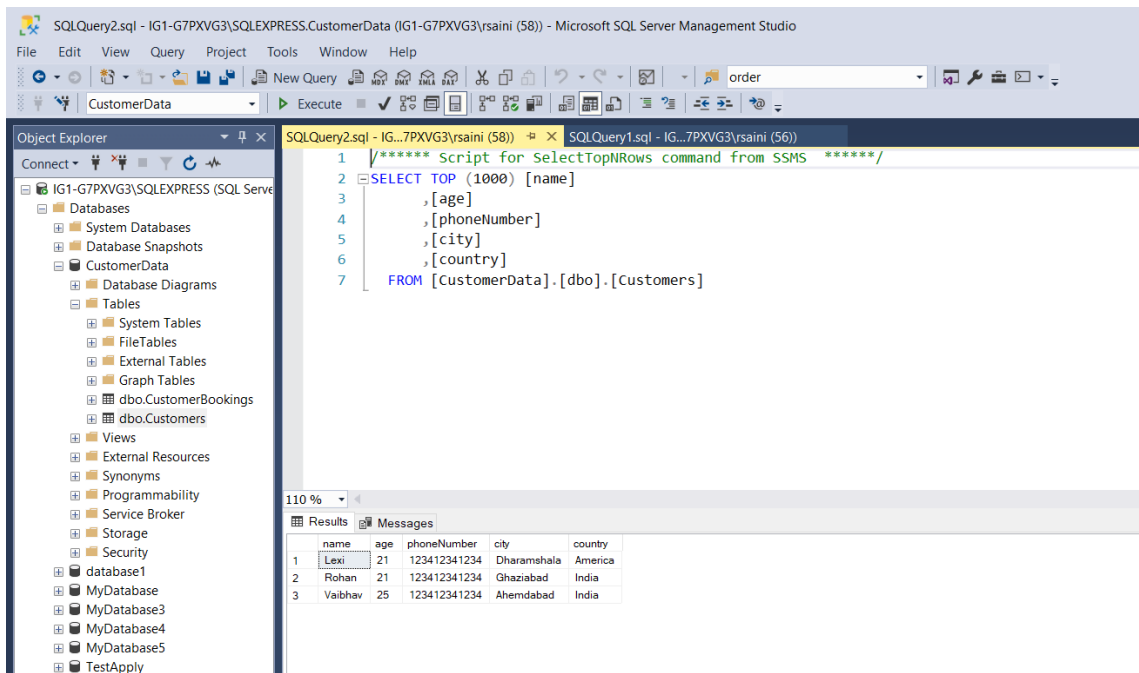


Figure 31: Customers Table

HTTP Request method: A client (browser) sends an HTTP request to the server, which is subsequently answered by the server. The answer may include the requested content as well as status information about the request.

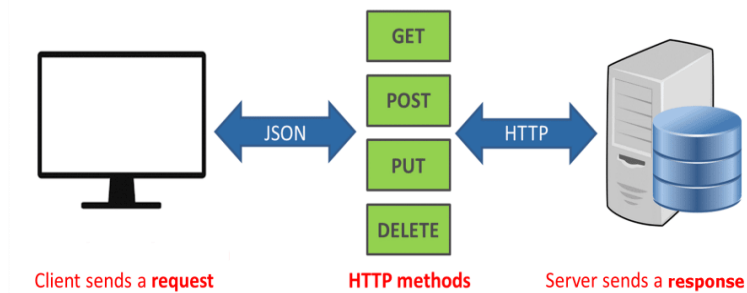


Figure 32: Http Client Server Model

HTTP Methods

- GET
- POST
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS
- CONNECT
- TRACE

Figure 33: Http Methods

3.4 Postman:

Postman is an API creation, testing, and modification tool. This programme contains practically every functionality a developer might want. Over 5 million developers use it every month to make API development simple. It can perform a wide range of HTTP requests (GET, POST, PUT, PATCH), save environments for later use, and convert API to code in a number of languages (like JavaScript, Python).

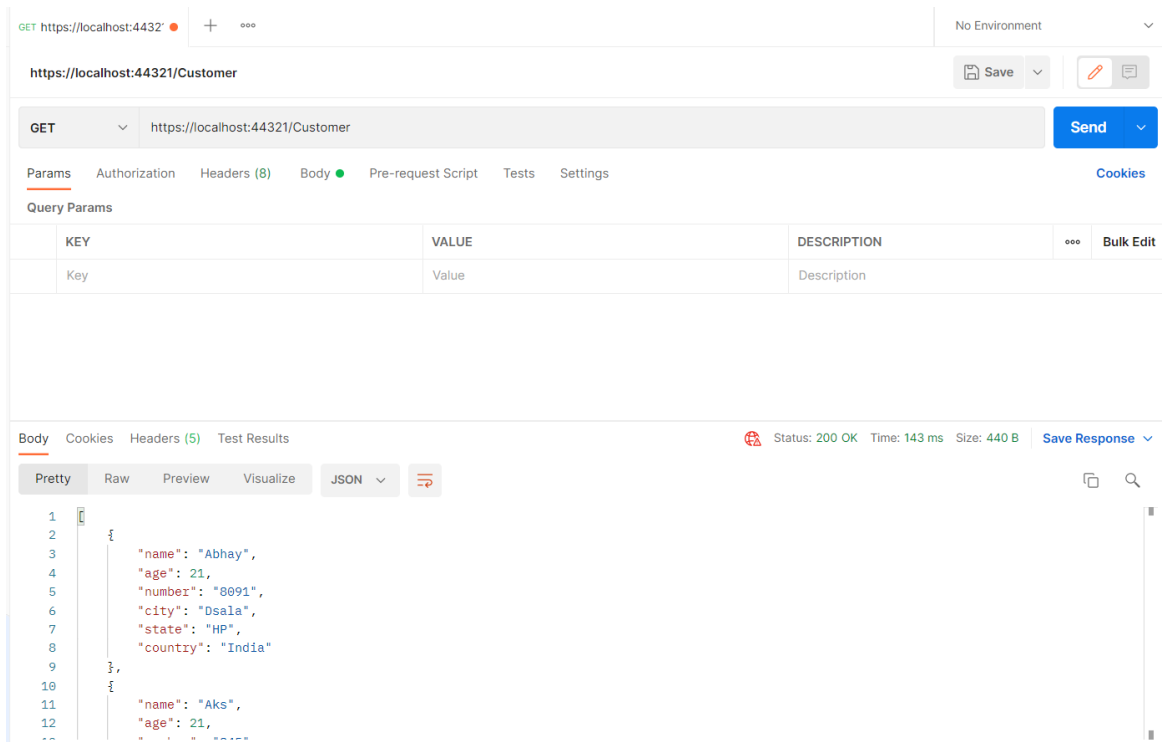


Figure 34: Check endpoint of Get API request using Postman

Chapter 4: SQL Server

4.1 Introduction to RDBMS:

RDBMS stands for Relational Database Management System. A relational database is a database category which will store data points and is able to provide access to data points which are related to each other. Relational databases are built on the concept of relational model which is an intuitive and straightforward method of tabular representation of data. In a relational database there is a unique ID called the key for each row in the. The table's columns include data attributes, and each record typically has a value for each attribute, making it simple to construct links between data points.

4.2 Structured Query Language (SQL) :

SQL is a computer language for retrieving and managing data in relational databases. Structured Query Language stands for Structured Query Language. SQL enables us to conduct operations on the database's records. Update records, delete records, create and edit database tables, insert records views, and so on are examples of such operations. SQL is the standard language which is used for Relational Database System. MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as the standard database language for all the Relational Database Management Systems (RDBMS).

4.2.1 Applications of SQL

SQL is the most popular query language that is used to manage databases.

- It facilitates us to access data in the relational database management systems.
- It helps us to describe the data.
- It provides us with tools to define the data in a database and manipulate that data.
- It provides us with tools to embed within other languages using SQL modules, libraries & pre-compilers.
- It provides us with tools to create and drop databases and tables.

- It provides us with tools to create view, stored procedure, functions in a database.
- It provides us with tools to set permissions on tables, procedures and views.

When we need to run a SQL command on a relational database management system, the system automatically determines the optimal approach to complete our request, and the SQL engine decides how to interpret that command.

4.2.2 SQL Components :

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

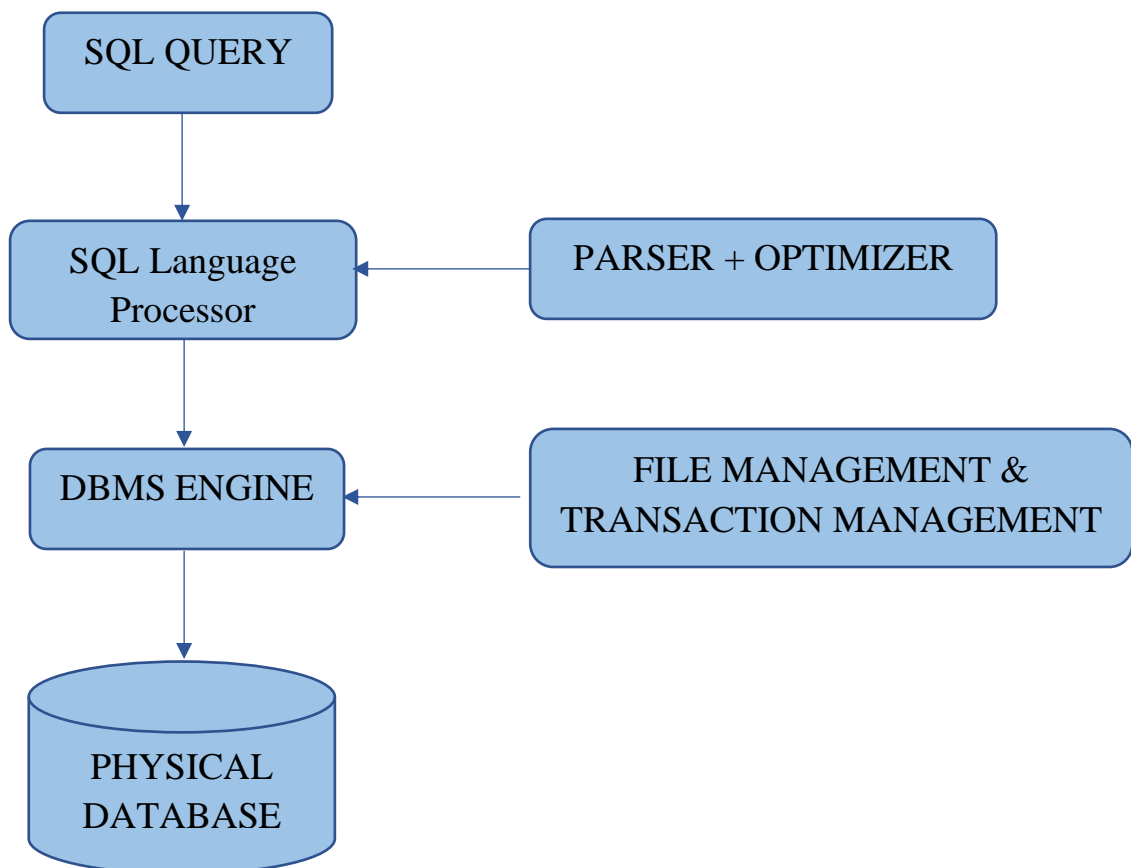


Figure 35: SQL Components Processing

4.2.3 SQL Architecture

SQL Server is a product of Microsoft, which obeys the principles of relational database management system (RDBMS). The basic functionality of the server is to store and thereby performing the retrieval operation on the data. The SQL server is completely based on a standard programming language i.e, SQL particularly used for working with relational databases. For over a span of two decades, SQL Server served the technical world for Windows. And soon in 2016, Microsoft released it for Linux. SQL Server Architecture

SQL Server consists of two main components:

- Database Engine
- SQLOS

Database Engine:

Database Engine is the core component of the SQL Server. The Database Engine consists of two things. A relational engine which is used to processes queries and a storage engine which is used to manage database files, indexes, pages, etc.

- Relational Engine:
The Relational Engine consists of the components which decide the best way to complete a query. The query processor is another name for the relational engine. Based on the input query, the relational engine retrieves data from the storage engine and processes the results. Query processing, memory management and distributed query processing are all tasks performed by the relational engine.
- Storage Engine:
The storage engine is responsible for data storage and retrieval from storage systems like discs and SAN.

SQLOS:

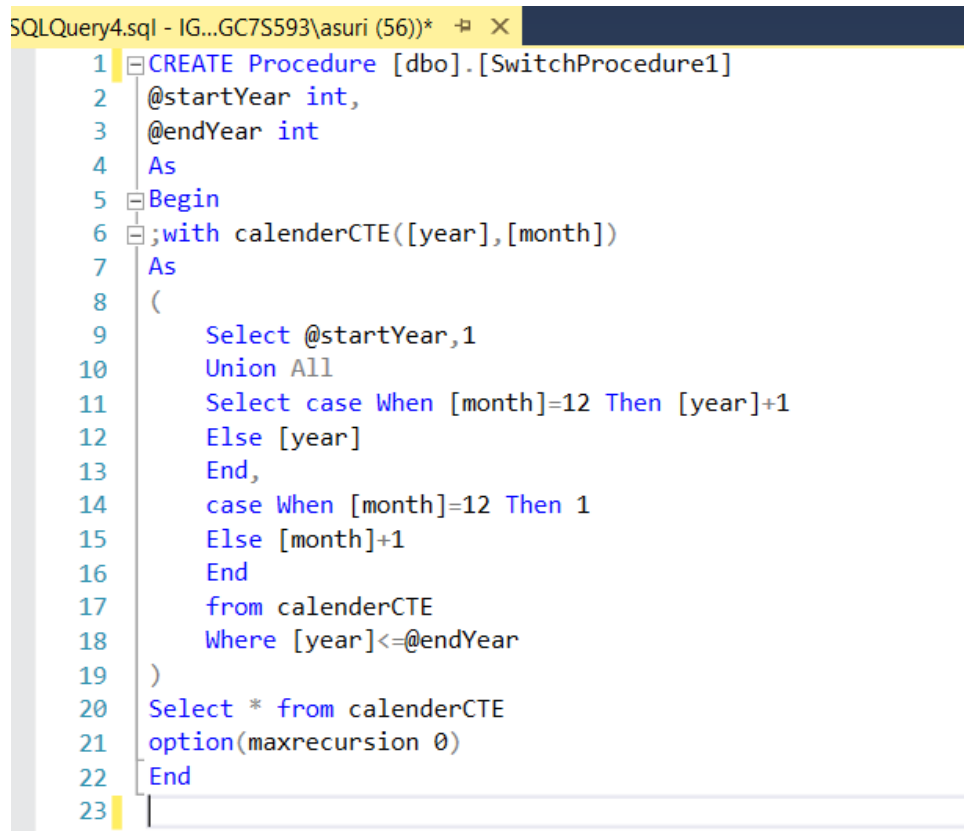
The SQL Server Operating System, or SQLOS, sits beneath the relational engine and garage engine. Many working machine offerings, which include as reminiscence and I/O management, are provided by SQLOS. Exception dealing with and synchronisation services are among the different offerings to be had

4.3 Stored Procedures:

SQL Server provides Stored Procedures with the motive of grouping multiple SQL statements in logical units. This procedure is a names object when its comes to the storage in SQL Server Database Server. On calling it at first, SQL Server creates a plan that handles the execution, called the execution plan and storage is done in the cache. On further executions, SQL Server reuses the plan for a fast and reliable execution.

The main advantages of stored procedure:

1. **Better Performance:** The stored procedure calls are quick and efficient. The reason being stored procedures are compiled once and then are stored in executable form thus giving quick response.
2. **Higher Productivity:** As we are reusing the same piece of code again and again so productivity is higher.
3. **Scalability:** By isolating application processing on the server, SP increase scalability.



```
SQLQuery4.sql - IG...GC7S593\asuri (56)* -# X
1 CREATE Procedure [dbo].[SwitchProcedure1]
2   @startYear int,
3   @endYear int
4   As
5   Begin
6     ;with calenderCTE([year],[month])
7     As
8     (
9       Select @startYear,1
10      Union All
11      Select case When [month]=12 Then [year]+1
12      Else [year]
13      End,
14      case When [month]=12 Then 1
15      Else [month]+1
16      End
17      from calenderCTE
18      Where [year]<=@endYear
19    )
20    Select * from calenderCTE
21    option(maxrecursion 0)
22  End
23
```

Figure 36: Stored Procedure

4.4 Common Table Expressions (CTE):

CTE stands for common table expression. A CTE helps us to create a temporary named result set which has temporary availability in the execution scope of a statement such as SELECT, INSERT, UPDATE, DELETE, or MERGE. This clause can be used in multiple situations like it can be used in a CREATE VIEW statement as part of its defining SELECT statement. A common table expression can even refer to itself. This is known as a recursive CTE.

```
SQLQuery4.sql - IG...GC7S593\asuri (56)* ×
1 CREATE Procedure [dbo].[SwitchProcedure1]
2   @startYear int,
3   @endYear int
4   As
5   Begin
6   ;with calenderCTE([year],[month])
7   As
8   (
9     Select @startYear,1
10    Union All
11    Select case When [month]=12 Then [year]+1
12    Else [year]
13    End,
14    case When [month]=12 Then 1
15    Else [month]+1
16    End
17    from calenderCTE
18    Where [year]<=@endYear
19  )
20  Select * from calenderCTE
21  option(maxrecursion 0)
22  End
23
```

Figure 37: CTE

4.4.1 Recursive CTE:

A CTE that references itself is known as a recursive common table expression. By referring itself, the CTE is continually executed, returning subsets of data until the entire result set is returned. When we need to query hierarchical data, such as organisation charts where one employee reports to a manager or multi-level bill of materials where a product is made up of many components, each of which is made up of many more components, a recursive CTE comes in handy.

```

SQLQuery6.sql - IG...GC7S593\asuri (58)*  X
1 ALTER Procedure [dbo].[finalCTE]
2   @startYear int,
3   @endYear int
4   AS
5   BEGIN
6       DECLARE @year int, @month int
7       SET @year = @startYear
8       SET @month = 1
9       ; with CTE (year1)
10          AS
11          (
12              SELECT @year
13              UNION ALL
14              SELECT year1+1 FROM CTE
15              WHERE year1< @endYear
16          ),
17          CTE2 (month1)
18          AS
19          (
20              SELECT @month
21              UNION ALL
22              SELECT month1+1 FROM CTE2
23              WHERE month1< 12
24          )
25 SELECT * FROM CTE2 CROSS JOIN CTE WHERE
26 NOT(NOT((year1 % 4 = 0 AND year1 % 100 <> 0) OR year1 % 400 = 0)and month1=2)
27 OPTION(maxrecursion 0)
28 END

```

Figure 38: Recursive CTE

4.5 PIVOT:

In SQL Server, the PIVOT operator perform rotation on a table-valued expression. It is used to split a single column's unique values across numerous columns and to aggregate any residual column values in the final result.

```

19 SELECT Teacher, Chef FROM #newTable
20 PIVOT
21 (
22     max(nameData)
23     For professionName
24     IN ([Teacher], [Chef])
25 ) AS PivotTable

```

Figure 39: Pivot

4.6 DYNAMIC SQL:

Dynamic SQL is a programming approach that allows us to create SQL statements on the fly. Because the finalised language of the SQL statements may be unknown at compilation,

it allows us to build more generalised and flexible SQL statements. Dynamic SQL can be used to create a stored procedure that executes a query on data against an unknown table at runtime.

4.7 SQL Server Management Studio:

SQL Server Management Studio (SSMS) is a software tool for configuring, controlling, and administering all Microsoft SQL Server components. It was first released with Microsoft SQL Server 2005. It replaces the Enterprise Manager in SQL 2000 or earlier. The tool offers both script editors and graphical interfaces for working with the server's objects and functionalities.

SSMS is a SQL Server management tool that can be used to construct queries and administer databases and data warehouses through a personal computer or the cloud, independent of your location. SQL Server Management Studio (SSMS) is basically an integrated environment with tools for configuring, monitoring, and administering SQL Server machines and databases.

The Object Explorer is a key element of SSMS, allowing the user to view, select, and act on any of the server's objects

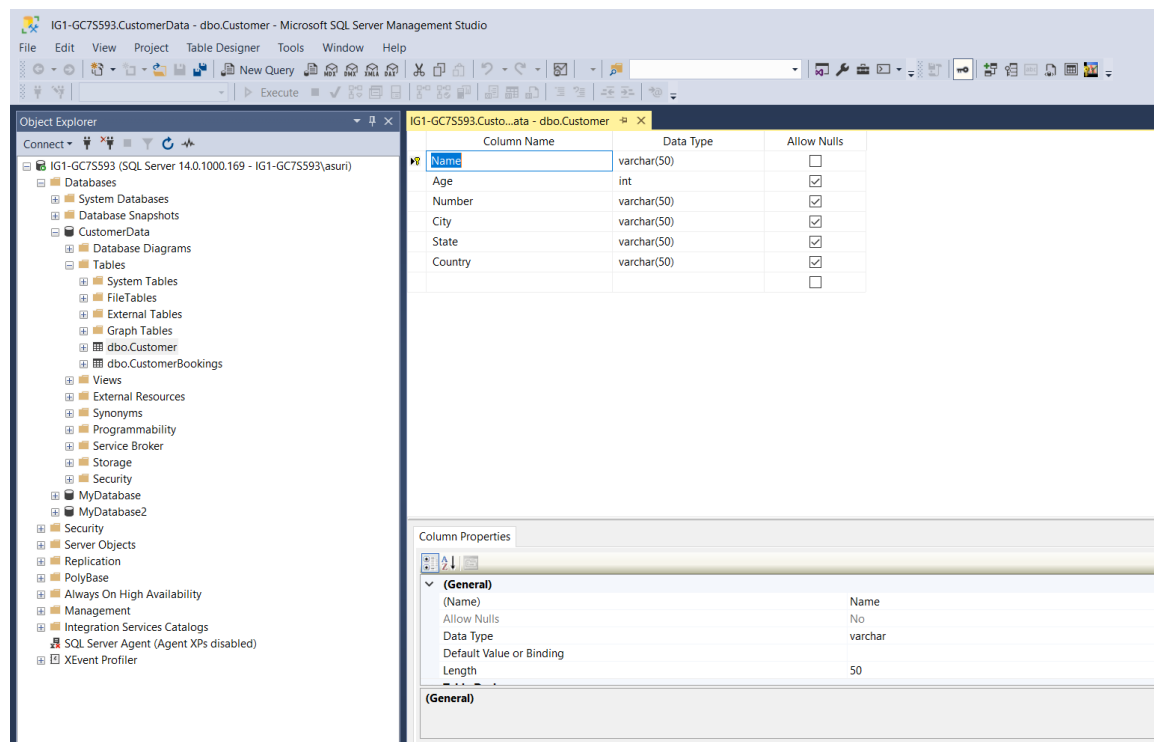


Figure 40: SQL Server Management Studio Environment

4.8 Object Relational Model (ORM):

An ORM stands for object relational mapping. This just means that we map SQL objects which are relational and you have foreign keys and primary keys to actual objects and code so that way we can more easily both read and view respond in objects and also manipulate them so that way you can set properties and say like save changes instead of needing to think about how to do an update statement. ORM allows us to write in the language you are already using. SQL is a very powerful language, but at the same time it is difficult to write.

4.9 Entity Framework Core:

Entity Framework (EF) Core is a data access ORM (Object-Relational Mapper) framework for .Net. It is an extendable, lightweight, Open Source, and cross-platform version of Entity Framework data access technology that was released alongside .NET Core. It runs on a variety of operating systems, including Windows, Mac, and Linux. It is an ADO.NET extension that provides developers with an automated technique for accessing and storing data in databases.

4.9.1 Entity Framework Core has two approaches:

Code-First Approach :

Before proceeding we need to develop our application domain classes such as Student, and a particular class which derives from the DbContext class in the EF Core Code First Approach. The the database and related tables are generated by EF Core based on the application domain classes and the DbContext class. On the basis of default conventions and setup in the code-first approach, the EF Core API generates the database and tables. The default conventions for creating the database and tables can also be changed according to our liking.

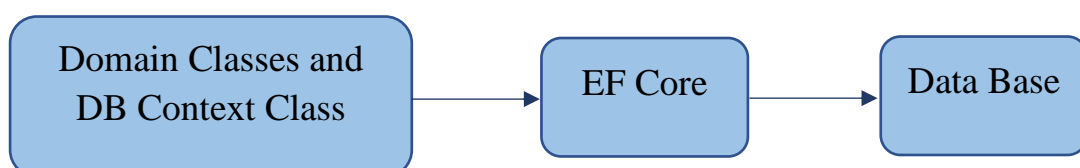


Figure 41: Model of Code First Approach

EF Core Database First Approach:

If we already have the database and tables then we can utilise the EF Core Database First Approach. DbContext and Domain classes are constructed by the EF Core on the basis of existing database schema using EF Core dependency.

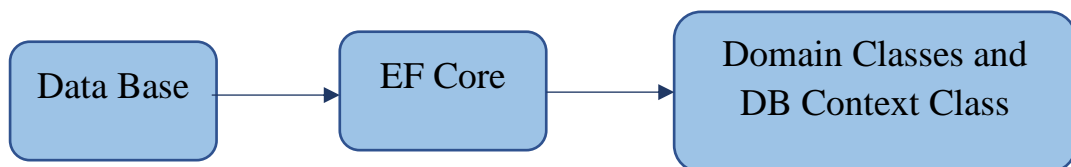


Figure 42: Model Of Database First Approach

| Column Name | Data Type | Allow Nulls |
|-------------|-------------|-------------------------------------|
| Name | varchar(50) | <input type="checkbox"/> |
| Age | int | <input checked="" type="checkbox"/> |
| Number | varchar(50) | <input checked="" type="checkbox"/> |
| City | varchar(50) | <input checked="" type="checkbox"/> |
| State | varchar(50) | <input checked="" type="checkbox"/> |
| Country | varchar(50) | <input checked="" type="checkbox"/> |
| | | <input type="checkbox"/> |

Figure 43: Design of Customer Table

```
using System;
using System.Collections.Generic;

namespace NewCustomerData.Models
{
    10 references
    public partial class Customer
    {
        4 references
        public string Name { get; set; }
        0 references
        public int? Age { get; set; }
        1 reference
        public string Number { get; set; }
        1 reference
        public string City { get; set; }
        1 reference
        public string State { get; set; }
        2 references
        public string Country { get; set; }
    }
}
```

Figure 44: Customer Class automatically generated by EF Core

4.10 EF Core Database Providers:

Because of database providers, the EF Core can support both relational and non-relational databases. NuGet packages are available for the database providers. Between the EF Core and the database it supports, the Database Provider sits.

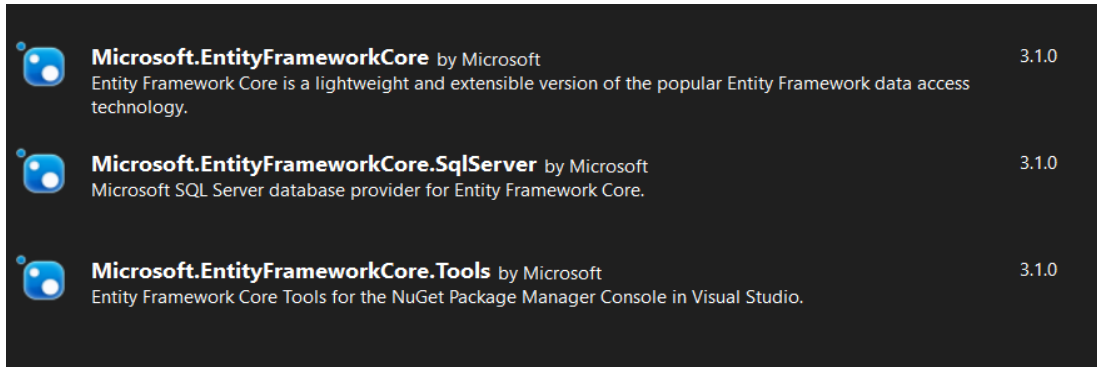


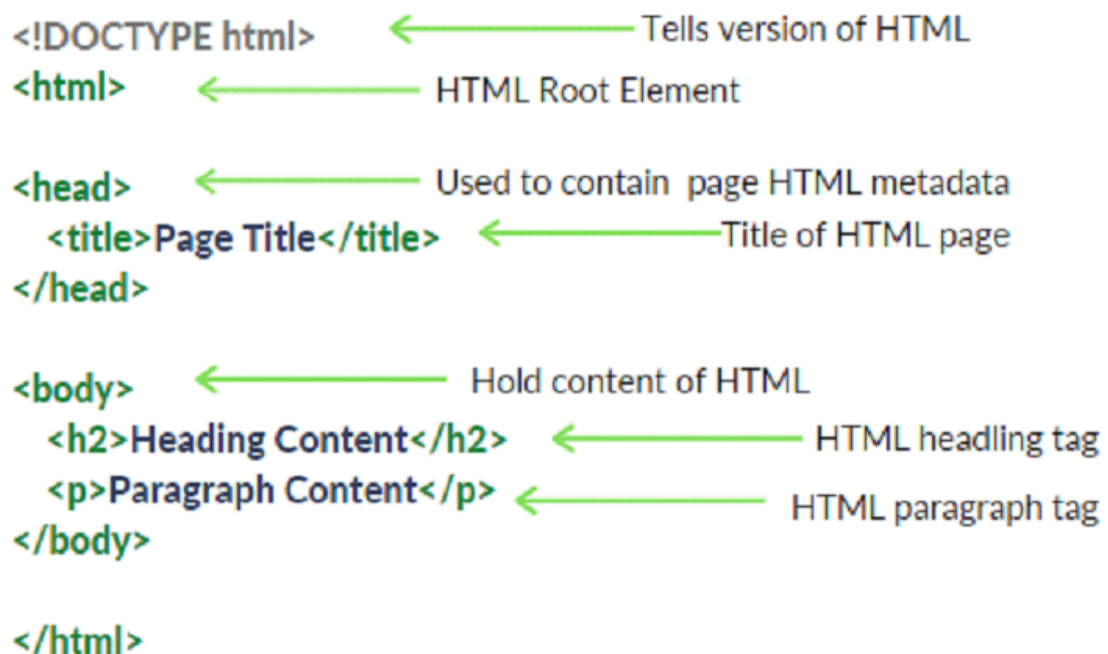
Figure 45: EF Core Dependencies

CHAPTER 5: FRONT-END

5.1 HTML

HTML is an abbreviation Hyper Text Markup Language. A language that is particularly used for creating the web pages is HTML. It performs so well to combine the hypertext with the markup. Hypertext is however is defined as a link between the pages of the web. Markup language is significantly used to define the documents in text inside the tags which further more helps to give a proper definition to the structure of the web pages. Adding notes means annotation of the text in order to make the computer understand and manipulate it accordingly. This language is a very popular for a human readable markup. The tags specifically have the motive to indicate the type of processing required in the text.

This language is used by the developers whereby browser could transform the different content like text, pictures etc. for a desired display in a correct format. Tim Berners-Lee invented HTML in 1991. HTML 1.0 was the first version, but HTML 2.0 was the first standard version, released in 1995.



HTML Page Structure

Figure 46: HTML Page Structure

5.2 CSS- Cascading Style Sheets

CSS (Cascading Style Sheets) is a stylesheet language for describing the presentation of an HTML or XML document (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should look on a screen, on paper, in speech, or in other media.

CSS is a basic language for the open web that is supported by all major browsers. Separate parts of the CSS specification were previously developed in parallel, allowing for the versioning of the most recent suggestions.

CSS is not a programming language like HTML. It's also not a markup language. CSS is basically a style sheet language. CSS is used to style HTML components selectively.

Flex layout :

Flex Layout is included to give the component a neat appearance. The Flexbox is used to specify the children of a component layout. We can archive the correct layout by using the `flexDirection`, `justifyContent`, and `alignItems` attributes.

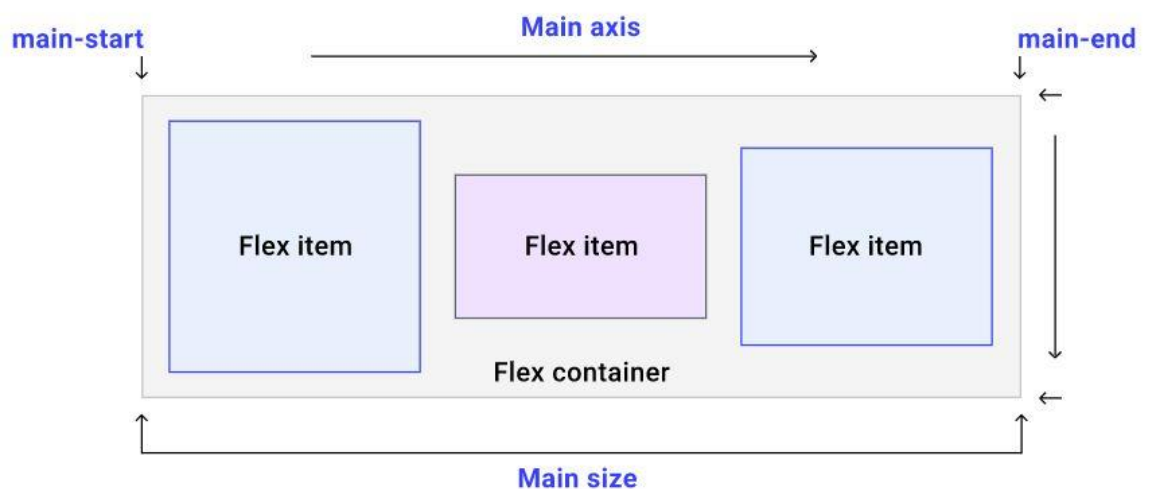


Figure 47: FlexBox Structure

5.3 Javascript

JavaScript is a lightweight, cross-platform object-based scripting language. JavaScript is a translated language, not a compiled language. The JavaScript Translator (which is built within the browser) is in charge of translating JavaScript code for web browsers.

JavaScript is an object-oriented programming language which is lightweight and it has a major focus to script the web pages. On applying JavaScript with document in HTML, it is believed to be a full-featured and interpreted programming language which is capable of interacting dynamically with websites. It was first released in 1995 with a motive to allow the users with the functionality of adding programs to the web pages in the Netscape Navigator browser. Ever since, the popularity of it in the field of graphical web browsers has increased. The developers were advantaged a lot with it, as modern web applications are created which have a great interaction immediately on reloading the pages. The features like simplicity and interaction can be achieved by this language.

There is no connection between JavaScript and the Java programming language. When Java began gaining prominence in the market, the name was suggested and provided. Databases like CouchDB and MongoDB use JavaScript as their scripting and query language, in addition to web browsers.

- Synchronous JavaScript: Synchronous refers to the execution of each statement of code in a sequential order. So, in essence, a statement must wait for the execution of the previous statement.
- Asynchronous JavaScript: Asynchronous code permits the program to be executed immediately, whereas synchronous code prevents the remaining code from being executed until the current one is completed. This may not appear to be a major issue, but when viewed in context, it can result in the User Interface being delayed.

5.3.1 AJAX

AJAX is an acronym that stands for Asynchronous JavaScript and XML. It is the use of the XMLHttpRequest object to communicate with servers in a nutshell. It can send and receive data in JSON, XML, HTML, and text files, among other formats. The most appealing

characteristic of AJAX is its "asynchronous" nature, which allows it to talk with the server, exchange data, and update the website without having to refresh the page.

The two main functionalities of AJAX:

- Make requests to the server without reloading the page
- Receive and work with data from the server

5.4 Typescript

JavaScript was introduced as a client-side programming language. JavaScript has emerged as an emergent server-side technology as a result of the creation of Node.js. However, as JavaScript code develops in complexity, it becomes more difficult to maintain and reuse. Furthermore, JavaScript's failure to embrace Object Orientation, rigorous type checking, and compile-time error checks hinders it from prospering as a full-fledged server-side technology in the enterprise. To fill this void, TypeScript was introduced.

TypeScript is JavaScript for application-scale development.

TypeScript is a compiled, strongly typed, object-oriented language. TypeScript is a set of tools as well as a programming language. JavaScript is compiled to TypeScript, which is a typed superset of JavaScript. To put it another way, TypeScript is JavaScript with a few more features.

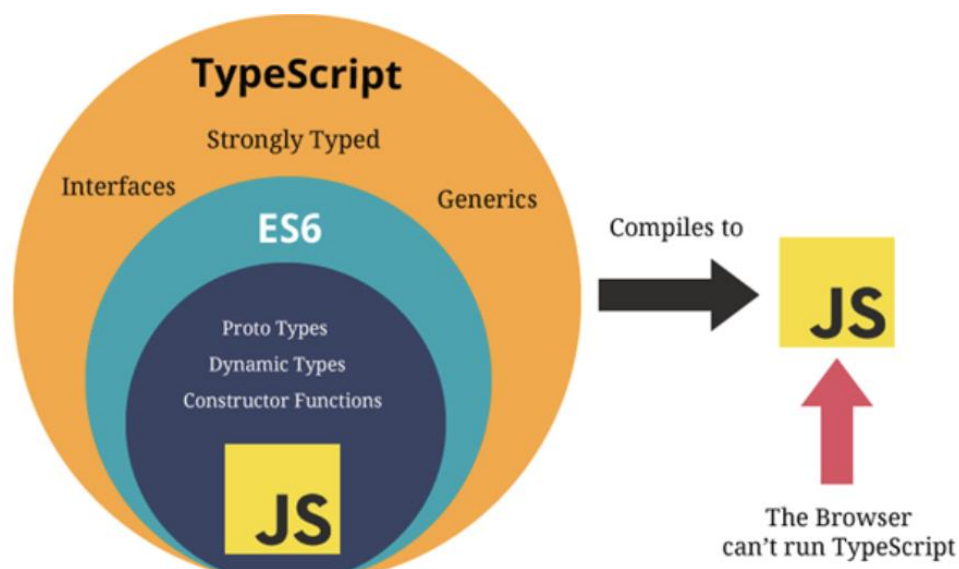


Figure 48: Compilation process of TypeScript

TypeScript is the outer domain of JavaScript. JavaScript is the beginning and the end of TypeScript. The essential building parts of your application are taken from JavaScript by TypeScript. To utilize TypeScript, you simply need to know JavaScript. For the purpose of execution, all TypeScript code is translated to its JavaScript equivalent.

JS libraries are supported by TypeScript. Any JavaScript code can consume compiled TypeScript. All existing JavaScript frameworks, tools, and libraries can be used with TypeScript-generated JavaScript.

JavaScript is TypeScript. JavaScript files can be converted to TypeScript just by renaming the .js file to .ts and compiled with other TypeScript files.

TypeScript is portable. TypeScript is cross-browser, device, and operating system compatible. It can run in any JavaScript-enabled environment. TypeScript does not require a separate virtual machine or a specific runtime environment to run, unlike its competitors.

Advantages of TypeScript

- TypeScript always highlights compilation mistakes during development (pre-compilation). Because JavaScript is an interpreted language, it is less likely to encounter runtime problems.
- Static/strong typing is supported in TypeScript. This means that type correctness can be verified during the compilation process. JavaScript does not support this feature.
- TypeScript is nothing more than JavaScript with some ES6 capabilities thrown in.

CHAPTER 6: CONCLUSION

This internship was indeed a pool of knowledge, not only have I gained knowledge in Full Stack Development but I have also learned about how development of any project takes place, how team works, how the work of each employee is tracked, how work is distributed between different team mates, what are the different stages of development, what are the technical problems that one faces in the development of any project, what all things are required before the development of any project, what the code base should be like and what norms need to be followed in the development. I have learned how to write code in such a manner that its understood by every team member of my team. I have also gained some knowledge on being an active team member and leading my fellow team members to great ideas.

REFERENCES

- BillWagner. “C# Docs - Get Started, Tutorials, Reference.” C# Docs - Get Started, Tutorials, Reference. | Microsoft Docs, docs.microsoft.com, <https://docs.microsoft.com/en-us/dotnet/csharp/>. Accessed 24 May 2022.
- NET Documentation | Microsoft Docs.” .NET Documentation | Microsoft Docs, docs.microsoft.com, <https://docs.microsoft.com/en-us/dotnet/>. Accessed 25 May 2022.
- “Web APIs | MDN.” Web APIs | MDN, developer.mozilla.org, 14 Sept. 2021, <https://developer.mozilla.org/en-US/docs/Web/API>.
- MikeRayMSFT. “Tutorials for SQL Server - SQL Server.” Tutorials for SQL Server - SQL Server | Microsoft Docs, docs.microsoft.com, 17 Dec. 2021, <https://docs.microsoft.com/en-us/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver16>.
- “Object-Oriented Programming (C#) | Microsoft Docs.” Object-Oriented Programming (C#) | Microsoft Docs, docs.microsoft.com, 20 May 2022, <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop>.