

**NOPCOMMERCE CUSTOMIZATION FOR IMPROVED
FUNCTIONALITY AND USER EXPERIENCE**

Project report submitted in partial fulfilment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Aashima Juneja (191345)

Under the supervision of

Mr. Bobin Sondhi

&

Dr. Aman Sharma



to

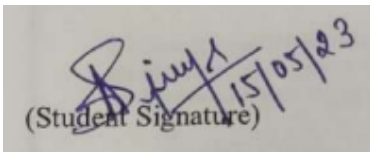
Department of Computer Science & Engineering and Information
Technology

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

CERTIFICATE

I hereby declare that the work presented in this report entitled “**NopCommerce Customization for Improved Functionality and User Experience**” in partial fulfilment 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 Wanknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Mr. Bobin Sondhi, Manager, Paxcom India Pvt. Ltd and Dr. Aman Sharma, Assistant Professor, Senior Grade.**

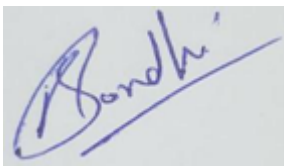
The matter embodied in the report has not been submitted for the award of any other degree or diploma.



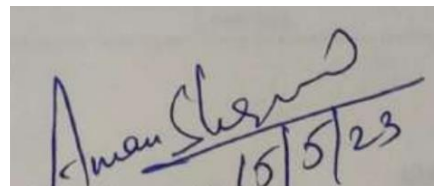
(Student Signature)

Aashima Juneja, 191345

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



Mr. Bobin Sondhi
Manager
Paxcom India Pvt. Ltd
Dated: 08/05/2023



Dr. Aman Sharma
Assistant Professor, Senior Grade
Computer Science & Engineering
Dated: 15/05/2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the Head of the Computer Science & Engineering and Information Technology Department and the Training and Placement Officer at Jaypee University of Information Technology for providing me with the wonderful opportunity to work with Paxcom India Pvt Limited- A Paymentus Company, during the final semester of my bachelors' course. I am immensely grateful for their support and guidance, which helped me gain invaluable real-world experience and develop important skills that will undoubtedly prove useful in my future endeavours.

During my training at Paxcom India Pvt Limited, I had the privilege of working with some of the most dedicated and passionate professionals in the industry. I am particularly grateful to my manager, Mr. Bobin Sondhi, who not only provided me with valuable guidance and support but also served as a great source of inspiration and motivation. The entire team at Paxcom India Pvt Limited was incredibly supportive and provided me with constant supervision, advice, and guidance whenever I needed it. Their unwavering commitment to excellence was truly inspiring, and I feel honoured to have had the opportunity to work with such a talented and dedicated group of professionals. I would also like to express my appreciation for the outstanding faculty at Jaypee University of Information Technology who prepared me well for this industrial experience. They not only imparted their knowledge but also provided me with all the necessary support, facilities, and co-operation that I needed to succeed.

Finally, I would like to extend my deepest gratitude to my parents, who have always been my biggest supporters and sources of inspiration. They instilled in me the value of hard work, dedication, and perseverance through their own example, and I will always be grateful for their unwavering love and support.

TABLE OF CONTENTS

TITLE	PAGE NO.
CERTIFICATE	i
PLAGIARISM CERTIFICATE	ii
ACKNOWLEDGEMENT	iii
LIST OF ABBREVIATIONS	vi
LIST OF FIGURES	vii
LIST OF TABLES	x
ABSTRACT	xi
CHAPTER 1: INTRODUCTION	
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Methodology	3
1.4.1 Frameworks	3
1.4.2 Application layers and Architecture	4
1.4.3 Technologies and Languages	5
1.4.3.1 SOLID Principles	5
1.4.3.2 C# Programming Language	8
1.4.3.3 Structured Query Language	9
1.4.3.4 JavaScript	10
1.4.4 IDE, Software	10
1.4.4.1 Visual studio	10
1.4.4.2 Microsoft Sequel Server	11
1.4.4.3 Git	12
1.4.4.4 Kestrel Server	13
1.4.4.5 Language Integrated Query Language	15
1.5 Organization	15
CHAPTER 2: LITERATURE SURVEY	
2.1 Application Programming Interface	17

2.2 Working of API	17
2.3 REST API	19
2.4 Routing	20
2.5 Razor View Engine	21
2.5.1 Razor View Engine Working	21
2.6 ASP.NET Core	22
2.7 Cross Origin Resource Sharing	23
CHAPTER 3: SYSTEM DESIGN AND DEVELOPMENTS	
3.1 Hardware requirements	25
3.2 Software Requirements	26
3.3 Functional Requirements	26
3.4 Project Development	30
3.4.1 MVC Lifecycle	30
3.4.2 Services in NopCommerce	32
3.4.3 Repositories in NopCommerce	34
3.4.4 Razor Pages	38
3.4.5 Partial Views	38
3.4.6 Layout	39
3.4.7 Database Implementation	40
3.4.8 Dependency Injection	41
3.4.9 Task Implementation	43
CHAPTER 4: RESULTS & DISCUSSIONS	
4.1 Results	49
CHAPTER 5: CONCLUSION	
5.1 Conclusion	58
5.2 Future Scope	59
REFERENCES	61
PLAGIARISM REPORT	62

LIST OF ABBREVIATIONS

API	Application Programming Interface
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
JS	JavaScript
MVC	Model View Controller
DBMS	Database Management System
LINQ	Language Integrated Query Language
SDK	Software Development Kit
RAM	Random Access Memory
NET	Network Enabled Technologies
ORM	Object Relational Mapping
REST	Representational State Transfer
CORS	Cross Origin Resource Sharing

LIST OF FIGURES

Figure Number	Caption [Reference]	Page Number
1.	Omni Channel	2
2.	Onion View of Application Layers in ASP.NET [1]	5
3.	SOLID principles for software development	6
4.	C# Logo [2]	8
5.	JavaScript Logo	10
6.	Visual Studio Logo [3]	11
7.	Microsoft SQL Server Logo	12
8.	Git Logo	12
9.	Working of Kestrel Server as Reverse Proxy [5]	14
10.	Code Snippet for retrieving list of customers from database	15
11.	Status codes [11]	18
12.	Working of API [10]	19
13.	REST API Architecture	20
14.	Razor View Engine Principle [15]	21
15.	NopCommerce Technologies [6]	25
16.	NopCommerce Store-front side [7]	28

17.	Home page of NopCommerce website	28
18.	Login page of NopCommerce Website	29
19.	Shopping Cart page of NopCommerce Website	29
20.	MVC Request Life Cycle [8]	31
21.	Working of MVC [9]	31
22.	Approaches of Entity Core Framework	36
23.	Code Snippet showing the Customer Class	37
24.	Code Snippet to retrieve the list of customers from database	37
25.	C# code in razor page to display product-name and product-price	38
26.	Common Layout used in Web Applications [19]	39
27.	Code Snippet showing the IEmailService Interface	41
28.	Implementation of IEmailService interface	42
29.	Code Snippet for CustomerService class	42
30.	Code Snippet for providing an instance of email service to its constructor	43
31.	Customer form fields (admin view)	44
32.	Register page of the store front side (default view)	45
33.	Customers Table (default admin view)	47
34.	Default Product Details view for the Gift Card (without the Link)	48

35.	Admin view for customer form fields (with middle name property)	49
36.	Store-front side view for the account info (with middle name field)	50
37.	Register page on store-front side (with middle name property)	52
38.	Store Front View showing the Register page for already registered customer	53
39.	Register page View on Successful Registration	54.
40.	Customers data table (with middle name property appended in the Name column)	54.
41.	Shopping cart (with the manufacturer and link of the product (if given))	55
42.	Admin side view of the Products Catalog page	56
43.	Edit Product Details page (admin view)	56
44.	Product Details page for the Gift Card with the Link of the product	57

LIST OF TABLES

Table Number	Title	Page Number
1.	Difference between ASP.NET and ASP.NET Core [18]	22
2.	Hardware Requirements for NopCommerce	25
3.	Software Requirements for NopCommerce	26

ABSTRACT

E-commerce has become an essential part of our daily lives, with the advancements in technology playing a crucial role in shaping its growth. With the ease of delivering products to your doorstep and the convenience of shopping from anywhere, anytime, e-commerce has revolutionized the way we shop. To keep up with the changing customer needs and preferences, e-commerce businesses need to continuously update themselves with the latest technologies and innovative strategies.

As part of my training project at Paxcom India Pvt Ltd, I am working on enhancing and customizing the NopCommerce site, which is an open-source e-commerce platform that provides a comprehensive set of features and functionalities to create a powerful and customizable online store. It is built on the ASP.NET framework, making it highly scalable and flexible to meet the diverse needs of businesses of all sizes. The aim of this project is to make the site more user-friendly, efficient, and tailored to meet the needs of our clients. With the help of the latest technologies and tools, we are optimizing the site's performance, improving its functionality, and enhancing the overall customer experience. By doing so, we are ensuring that our clients can leverage the power of e-commerce to grow their business and stay ahead in the competitive market.

CHAPTER- 1

INTRODUCTION

1.1 Introduction

With the arrival of ultramodern technology, ecommerce has changed a lot. With this change comes a complete change in the way consumers interact with products, making the buying process better and easier than ever ahead. moment, where ecommerce has come an important part of our diurnal life, products can fluently come to our door. still, keeping up with the rearmost technological developments is no easy task for eCommerce associations. In order to contend in moment's business world, ecommerce companies should be encouraged to use new technologies, develop good strategies and hire experts.

These technological inventions have made shopping easier for consumers and have enabled them to fluently meet their shopping requirements. Cutting edge technology makes ecommerce briskly and lightly, offering consumers a unique shopping experience. With just many clicks, buyers can find what they want and order fluently. In addition, guests can track their orders, find the stylish deals and stay informed about the rearmost deals. Due to these developments, new businesses are arising in the field of e-commerce. The elaboration of omnichannel shopping Figure1, which uses multiple channels and defenses to interact with guests, is getting more and more important. In fact, recent data shows that around 90% of consumers want flawless relations across channels.



Figure 1 - Omni Channel

By using the right technology, e-commerce companies can provide consumers with exactly what they want, when they want it. This can help to not only attract and retain customers but also boost sales and revenue. Ultimately, the continued evolution of technology will undoubtedly play a crucial role in shaping the future of e-commerce.

1.2 Problem Statement

The aim of the NopCommerce training project is to improve and alter the current e-commerce platform in order to satisfy changing consumer and commercial demands. A seamless and customized online shopping experience is essential to attracting and keeping clients in the rapidly evolving digital market. The development and profitability of an e-commerce business might be hampered by out-of-date features, ineffective functioning, and a subpar user experience. In order to improve the performance, functionality, and usability of the NopCommerce platform, this project aims to address these problems by applying creative ideas and utilizing cutting-edge technology. By doing this, we hope to establish a robust and scalable ecommerce system that can assist companies in staying ahead of their competition in the market and meet the demands of the evolving customers.

1.3 Objectives

The objectives of the training project are-

1. To provide an overview of the importance of technology in e-commerce and its impact on businesses.
2. To get an understanding of key technical concepts such as -
 - a. C# Programming Language
 - b. HTTP Context
 - c. Web API
 - d. SOLID principles
 - e. Kestrel Server
 - f. Add Singleton, Add Scoped, Add Transient functions in .Net core
 - g. Dependency Injection/ Inversion of Control principle
 - h. MVC architecture-
 - MVC lifecycle
 - Model State
 - Model Attributes
 - Routing
 - Razor Pages
 - Action Result Types
 - Model Binding
 - Action Filters
3. To outline the steps involved in enhancing and customizing the NopCommerce platform to meet the unique needs of businesses.

1.4 Methodology

1.4.1 Frameworks

ASP.NET and ASP.NET Core are two popular frameworks developed by Microsoft for building web applications. While both frameworks share some similarities, they have significant differences in terms of architecture, development approach, and platform support.

Since the early 2000s, ASP.NET has existed as a more seasoned and established framework. It is based on the .NET Framework and primarily intended for creating online apps that interact with Windows-based OSes. Due to its monolithic architecture and need for a complete installation of the .NET Framework on the web server, ASP.NET can be less resource-efficient and flexible.

In comparison, the 2016 version of ASP.NET Core, a more recent and lightweight framework. It is constructed on top of .NET Core, a cross-platform, modular version of the .NET Framework. Because ASP.NET Core is modular and flexible, developers can pick and choose only the components they require. It is a more platform-independent solution because it can be used with Windows, Linux, and macOS. Either framework might be a good choice for developing web applications, depending on the particular needs and specifications of a project.

1.4.2 Application layers and architecture

ASP.NET is based on a multi-tier architecture that divides an application into several layers. These layers are essential for organizing the application's code, simplifying complexity, and promoting modularity and maintainability.

The common layers used in ASP.NET applications are as follows:

1. **Presentation Layer:** This layer is in charge of overseeing the application's user interface, which includes user controls, master pages, and web pages.
2. **Business Layer:** This layer executes the application's business logic, which includes data processing and enforcing business rules. It serves as a liaison between the data access layer and the display layer.
3. **Data Access Layer:** The third layer, known as the data access layer, is in charge of gaining access to information kept in databases or other data sources. It consists of data models, data access classes, and data access objects.

4. **Infrastructure Layer:** This layer includes all supporting elements needed to run the application, including configuration, caching, security, and logging.

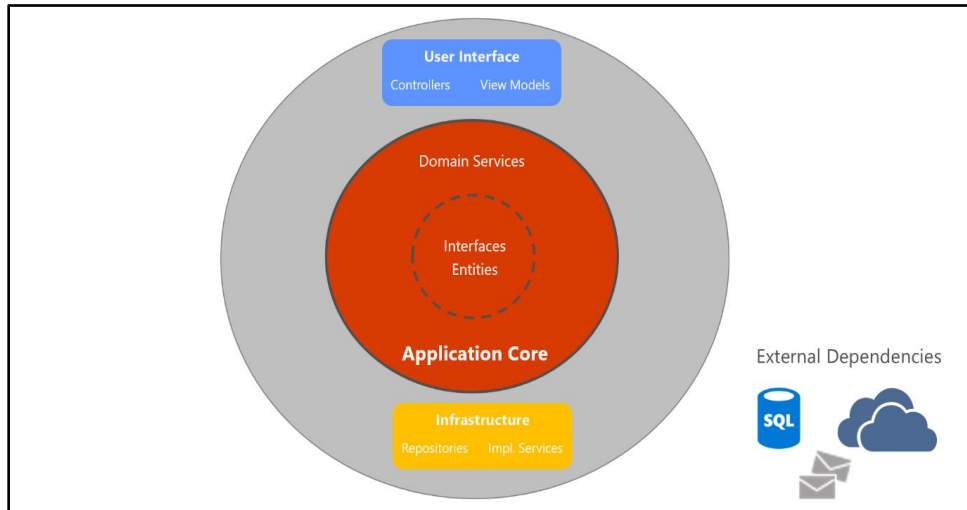


Figure 2- Onion View of Application Layers in ASP.NET [1]

Figure 2 shows the onion view of ASP.NET's layer-based architecture. It is highly adaptable and expandable, allowing developers to modify and add functionality to an application more efficiently. By enabling developers to work independently on different components of an application, the architecture can enhance productivity and decrease development time.

1.4.3 Technologies and languages

1.4.3.1 SOLID principles

When it comes to creating flexible, scalable, manageable, and reusable code in software development, object-oriented design is essential. Robert C. Martin, often known as Uncle Bob, established the SOLID principle, which is a coding convention for programmers.

This principle is the abbreviation for the five principles listed below.

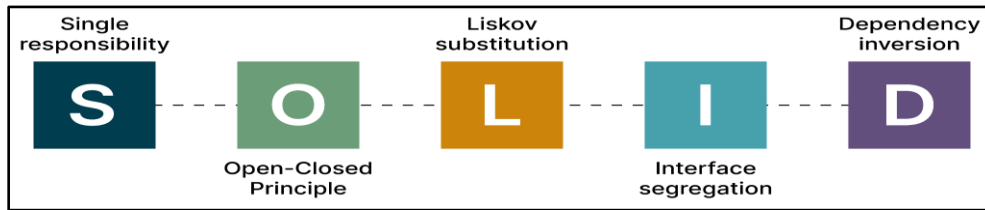


Figure 3- SOLID principles for software development

1. **Single Responsibility Principle-** According to this rule, "a class should have only one reason to change," which means each class should have only one duty, one task, or one goal. Consider the creation of software. We may argue that everyone has a single job or responsibility because the task is separated into distinct individuals doing different tasks, such as front-end designers performing design, testers conducting testing, and back-end developers taking care of the back-end development aspect. When programmers need to add features or new behavior, they frequently integrate everything within the existing class, which is utterly incorrect. When later something needs to be changed, it makes their code lengthy, complex, and time-consuming.
2. **Open/Closed Principle-** According to the "open for extension, but closed for modification" (or "open/closed") approach, "software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification." This indicates that you should be able to modify a class's behavior without being able to extend its behavior. If developer A must update a library or framework and developer B wants to make changes or add new features, developer B may extend the current class developed by developer A, but developer B is not permitted to make direct changes to the class. By separating the modified code from the existing code using this technique, your code will be more stable, maintainable, and subject to fewer modifications.

3. **Liskov's Substitution Principle-** According to this principle, "Derived or child classes must be substitutable for their base or parent classes," it was established by Barbara Liskov in 1987. The use of any class that is a child of a parent class should be possible without resulting in any unexpected behavior, according to this principle. It makes sense that a farmer's son would pick up farming skills from his father and be able to take over the family farm if necessary. Even though they both fall under the same family hierarchy, the son can replace his father if he wants to become a farmer, but if he wants to play cricket, he most definitely cannot. A rectangle with four edges is among the classic illustrations of this idea. Height and width of a rectangle can both be any value. A rectangle with equal width and height is called a square. Therefore, we can state that the properties of the rectangle class can be extended to the square class. To achieve this, you must replace the parent (rectangle) class with the child (square) class. However, a derived class does not impact the behavior of the parent class, thus if you do this, you will be in violation of the Liskov Substitution Principle.

4. **Single Responsibility Principle-** The single responsibility concept is analogous to the interface segregation principle, which is the first SOLID principle to apply to interfaces rather than classes. "Do not force any client to implement an interface that is irrelevant to them," the statement reads. The major objective in this situation is to give preference to numerous small client-specific interfaces over one large interface. One general client interface should be avoided in favor of several client interfaces, each of which should have a distinct function. Imagine going to a restaurant as a strict vegetarian. You received the menu card from the waiter at that restaurant, which lists vegetarian and non-vegetarian options as well as beverages and desserts. In this situation, as a customer, you should have a menu card that only features vegetarian options, not anything you don't eat. Here, the menu ought to vary depending on the customers. Instead of having

just one common or general menu card for everyone, there can be multiple cards. Using this principle helps in reducing the side effects and frequency of required changes.

5. **Dependency Inversion Principle-** This principle's primary goal is to decouple dependencies so that class B doesn't need to be concerned about or aware of changes to class A. Think about the battery in a TV remote as an actual example. Regardless of the battery's manufacturer, the remote needs a battery. Any XYZ brand can be used; it will function. Thus, we can say that the brand name is only tangentially associated with the TV remote. The code is more reusable because of dependency inversion.

1.4.3.2 C# programming language

As part of the .NET Framework project, Microsoft developed the object-oriented programming language C# in the early 2000s. It is a well-liked option for creating games as well as software programs for the desktop, online, and mobile platforms. The strength of a low-level language and the simplicity of a high-level language are combined in C#.



Figure 4- C# Logo [2]

Numerous features are supported, including automatic garbage collection, strong typing, and concepts from object-oriented programming like inheritance, encapsulation, and polymorphism. It also has sophisticated capabilities like

generics, LINQ, and asynchronous programming that make it appropriate for creating intricate and scalable applications. One of C#'s advantages is its adaptability, which makes it simple to learn and use for developers of all skill levels. Because C#'s syntax is similar to that of other well-known programming languages like Java and C++, developers can switch between them more easily. Additionally, C# offers a vast array of resources and tools, including the Visual Studio development environment, which gives programmers a full set of tools for creating, testing, and deploying C# applications.

1.4.3.3 Structured Query Language

For handling relational databases, programmers frequently use SQL (Structured Query Language). Popular databases including MySQL, Oracle, Microsoft SQL Server, PostgreSQL, and SQLite can utilize it to store, retrieve, and edit data.

The way SQL functions are through statements that help you carry out various activities like building and modifying database structures, adding, updating, and removing data, and retrieving data from the database. Since SQL is a declarative language, you only need to specify what you want to do with the data—the database management system (DBMS) handles the specifics. This is one of the language's main advantages. For those with little or no programming knowledge, in particular, this makes it simpler to understand and use.

In addition to supporting numerous users and transactions, SQL also efficiently handles big datasets and guarantees data confidentiality and integrity. Furthermore, it is highly portable, allowing for the use of the same code with little to no modifications on various DBMS.

1.4.3.4 JavaScript

For building interactive web pages and applications, developers frequently use JavaScript, a high-level, dynamic, and interpreted programming language. When Brendan Eich was still employed at Netscape, he introduced it in 1995. Due to the fact that JavaScript is a client-side scripting language, it executes on the user's web browser as instead of the web server. It is used to enhance web sites with interaction, animations, and dynamic functionality.



Figure 5- JavaScript Logo

In addition, Node.js is employed for developing server-side programs, online apps, and APIs. JavaScript is a powerful language that can be used for a variety of tasks, from straightforward form validation to sophisticated online applications. It has a big developer community that contributes to its ongoing development and is lightweight, simple to learn, and easy to use.

1.4.4 IDE, Software

1.4.4.1 Visual studio

For creating software applications, Microsoft created the highly renowned integrated development environment (IDE) known as Visual Studio. By offering developers the ability to change code, debug it, test it, and deploy it all from one platform and with a variety of features and tools, Visual Studio speeds the development process. Numerous programming languages, including C#, VB.NET, F#, C++, and JavaScript, as well as a number of frameworks, including the.NET

framework, are supported by this potent IDE. Developers can work together on projects and effectively manage codebase changes because to its integration with source control tools like Git.

One benefit of using Visual Studio is how adaptable it is. By utilizing its add-ons and extensions, developers may modify the IDE to meet their unique needs. The ability to customize one's development environment and simplify workflows is provided by this feature.

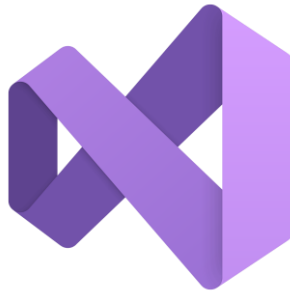


Figure 6- Visual Studio Logo [3]

In conclusion, Visual Studio is a thorough and adaptable IDE that gives developers the tools they need to produce high-quality software applications for a variety of platforms and programming languages. Its variety of features and customization choices make it an excellent tool for developers who want to streamline their processes and strengthen their skill sets.

1.4.4.2 Microsoft Sequel Server

Due to its scalability and capability to store, manage, and retrieve data for many industries, Microsoft SQL Server is a relational database management system that has gained prominence. The flexible and dependable SQL Server platform allows developers to create efficient database solutions for both modestly sized apps and massive enterprise-level solutions. A few of the cutting-edge features that come standard with SQL Server and aid in preserving the security, integrity, and accessibility of crucial data are data encryption, backup and recovery, replication,

and high availability choices. Additionally, SQL Server supports a variety of programming languages, including T-SQL, which provides more complex data manipulation and querying capabilities than regular SQL.



Figure 7- Microsoft SQL Server Logo

For companies that already use Microsoft-based solutions, Microsoft SQL Server is a desirable alternative because it interfaces well with other Microsoft products like Visual Studio and Power BI. For companies looking for a dependable and scalable solution, SQL Server's excellent capabilities and interoperability with a wide range of applications make it a widely sought-after database management system.

1.4.4.3 Git

To manage and track changes to software code, documents, and other digital items, one can use the distributed version control system known as Git. Users can communicate with others, keep track of changes made to files over time, and undo or roll back changes as necessary. In 2005, it was developed by Linus Torvalds, who also built the Linux operating system.



Figure 8- Git Logo

All files and their modifications are kept in a repository, which can be hosted locally or on a distant server, in Git. Every time a change is made, a commit is made and a message detailing the change is included. After then, these commits are categorized into branches, enabling the concurrent development of many code versions.

1. **Merging:** Git allows developers to merge changes from different branches or repositories into a single, unified codebase. This can be done automatically or manually, depending on the level of control required.
2. **Reverting:** If a mistake is made, Git allows developers to revert to a previous version of the code, undoing all changes since that point.
3. **Cloning:** Developers can make a copy of an existing repository, including all its history and branches, by cloning it.
4. **Forking:** Forking allows developers to create a new repository based on an existing one, which can then be modified and developed independently.

1.4.4.4 Kestrel Server

Specifically designed to execute ASP.NET Core apps, Kestrel is a cross-platform, open-source web server. It is a speedy and lightweight web server that has low latency and can handle thousands of requests per second because it is built on top of the .NET Core runtime.

In order to function, Kestrel monitors a particular port for incoming HTTP requests and routes those requests to the appropriate ASP.NET Core application for processing. As soon as a request is received, Kestrel starts a new thread to handle it, keeping the server responsive even when multiple requests are being processed simultaneously. In addition to serving as a standalone web server, Figure 9 shows how Kestrel may also be used as a reverse proxy server.

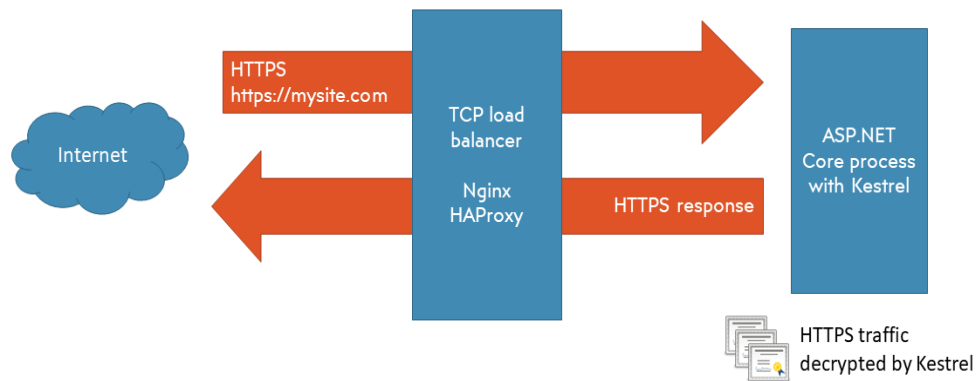


Figure 9- Working of Kestrel Server as Reverse Proxy [5]

Between a client and a web server, an intermediary server called a reverse proxy relays client requests to the latter and the latter's responses back to the former. By acting as an intermediary between the client and the web server, a reverse proxy can provide a number of advantages, including:

1. **Load balancing:** Client queries can be split among several web servers using reverse proxies, which helps to balance the load and prevents any one server from becoming overloaded.
2. **Security:** Reverse proxies can provide an additional layer of security by hiding the details of the web server from external clients and filtering out malicious requests.
3. **Caching:** By caching frequently requested content, reverse proxies can lighten the pressure on web servers and speed up response times.
4. **Scalability:** Reverse proxies can increase the scalability and availability of web applications by spreading traffic across numerous web servers.

Kestrel is a reliable and adaptable web server that is good for hosting ASP.NET Core apps. Its ability to function as a reverse proxy gives the hosting environment an additional level of flexibility and scalability.

1.4.4.5 Language Integrated Query Language

With the help of .NET's LINQ (Language Integrated Query) technology, you can build queries with a uniform syntax against a range of data sources. A range of built-in operators can be used with LINQ queries to alter, filter, and sort the data obtained from in-memory collections, databases, and other data sources. LINQ queries can be created using either a fluid syntax or query expressions, which resemble SQL queries in appearance. Here is an illustration of a LINQ query that retrieves a list of clients from a database and sorts them according to last name:

```
using (var context = new MyDbContext())
{
    var customers = from c in context.Customers
                    orderby c.LastName
                    select c;
}
```

Figure 10- Code Snippet for retrieving list of customers from database

In this illustration, figure 10, the "from" keyword is used to identify the data source, which is the Customers table in the database, and the "orderby" clause is used to arrange the results by the Last Name field. The "select" keyword is used to specify the attributes that should be present in the result set.

1.5 Organization

The way in which this project report is organized is that:

1. Chapter 1 comprises of the Introduction about the project, objectives, tools and technologies used, the frameworks and programming languages used.

2. Chapter 2 covers the literature survey and the study that was done before the actual development phase. It covers a brief introduction about all the necessary topics that are required to be studied before beginning the project.
3. Chapter 3 covers the Hardware, Software and functional Requirements of the project. In addition, it includes the description of all phases of the project development.
4. Chapter 4 covers the results along with the screenshots of the admin view and store-front view that were obtained after the implementation of the tasks assigned.
5. Chapter 5 covers the Conclusions obtained after the finishing of the project and its future scope.

CHAPTER-2

LITERATURE SURVEY

2. Literature Survey

2.1 Application Programming Interface

The acronym API stands for "Application Programming Interface." It serves as a software bridge that permits communication between several software applications. An API is a collection of standards, instructions, and programming languages that specify how software components should communicate with one another and are used to create software applications. APIs can be used to connect to third-party services, send data to distant servers, perform activities on those servers, or obtain data from distant servers. They enable users to communicate with remote resources in a standardized way, enabling programmers to create applications that are more dependable, scalable, and integrated. Finally, an API specifies a set of rules and protocols that allow different software applications to communicate with one another and exchange data without the programs themselves needing to do it. [12]

2.2 Working of API

When a client application wants to use an API, it sends a request with the desired action and any additional parameters or data to the API endpoint. The API responds with a response after processing the request, often with the client's requested data and a status code.

The client program examines the answer and performs the required action based on the status code, figure 11, and response data.

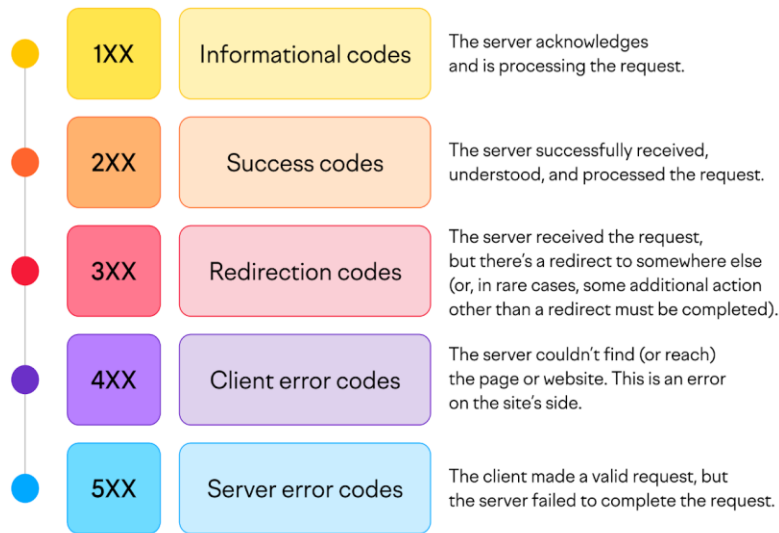


Figure 11- Status codes [11]

Many different tasks can be accomplished using APIs, including data retrieval from distant servers, data submission to distant servers, action triggering on distant servers, and integration with third-party services.

Access to APIs is possible through a variety of protocols by other applications, websites, or mobile devices.

In order to connect with an API endpoint, developers often create code that makes requests to it using a library or tool that maintains the underlying communication protocol, such as HTTP. Although there are many different approaches to implement APIs, they should generally be transparent, safe, and scalable.[12]

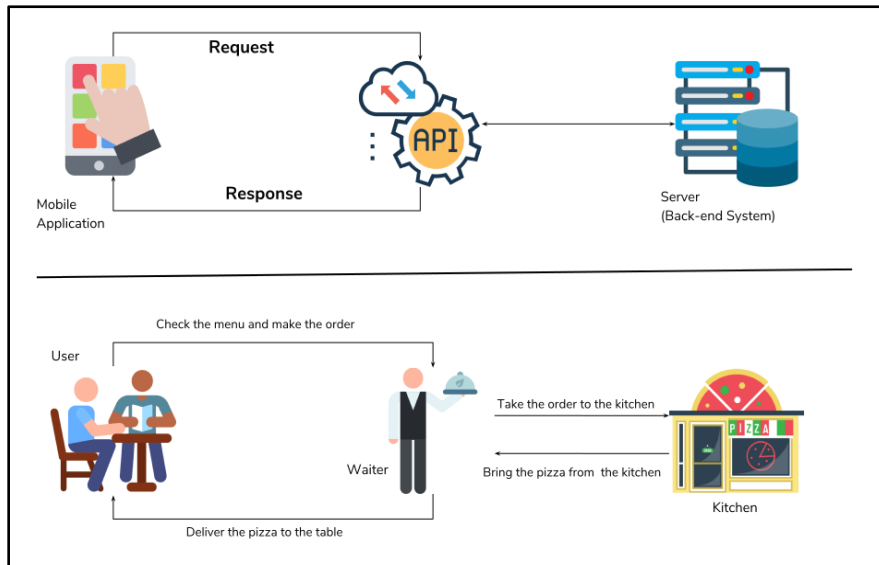


Figure 12- Working of API [10]

Figure 12 shows a very simple example of a restaurant to demonstrate the working of API. Just like waiter acts a mediator between the customer and the chef and brings the requested food to the respective customer, same does the API to process the requests on the server. Developers may build more reliable, scalable, and integrated applications thanks to APIs, which offer a standardized method of interacting with external resources.

2.3 REST API

REST represents representational state transfer and uses the compact JSON format to send data. Because of its quick performance, dependability, and capacity to scale by reusing modular components without harming the system, the majority of public APIs employ this [13].

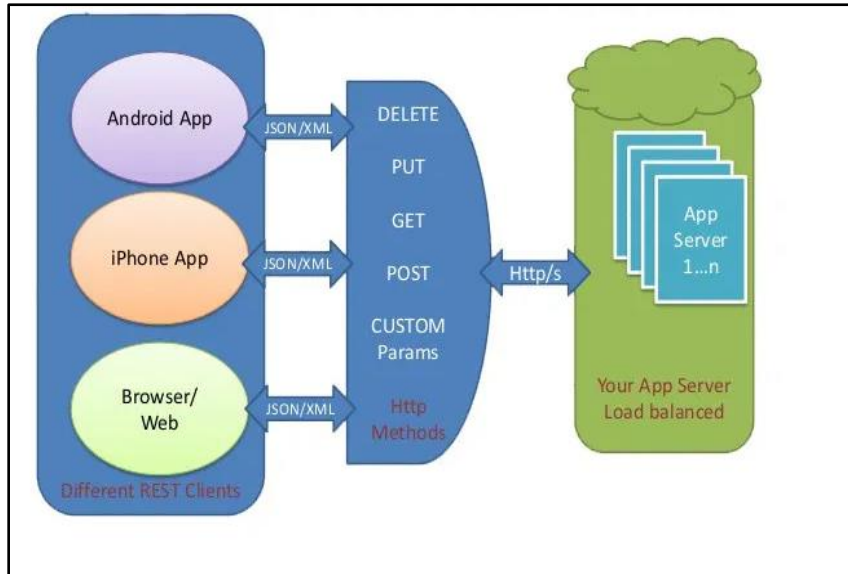


Figure 13- REST API Architecture

Figure 13 shows the REST API structure. Using a standard and predefined set of operations, this API provides access to data. The HTTP protocol and URLs provide the foundation of REST APIs.

2.4 Routing

The process of mapping an incoming HTTP Request (URL) to a specific resource, such as a controller action method, is known as routing in an ASP.NET Core Web API application. We often set a few URLs for each resource when using the Routing Concept in the ASP.NET Core Web API. The Route table, which contains the mapping details between the URL and the Resource, is created when the application is started. Therefore, the application will check the URL in the Route table when a request is sent from the client to the server, and if it finds an exact match, it will send the request to that specific resource; otherwise, it will throw an error saying that the resource was not found. In the ASP.NET Core Web API Application, any resource is accessible to us by using a specific URL. A resource could also have several different, distinct URLs. [14] However, multiple resources cannot share the same URL, as doing so causes the application to become confused about which

action method to invoke, which results in an ambiguity error. Therefore, based on the routes that are set up for the application, the ASP.NET Core Framework maps the incoming HTTP Requests, i.e., URLs, to the action methods of Controllers. Multiple routes can be set up in ASP.NET Core, and each route can have its own specific configurations such as default values, constraints, message handlers, etc.

2.5 Razor View Engine

The rendering of the view into an HTML form for the browser is done by the view engine. Razor View (.cshtml) and web form (.aspx) are supported by ASP.NET MVC [15].

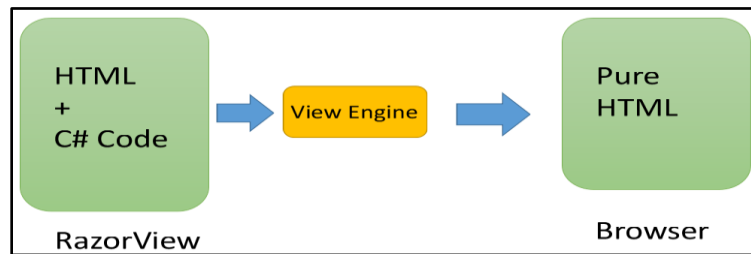


Figure 14- Razor View Engine Principle [15]

To transform HTML + Programming language to pure HTML, a View Engine is used, as shown in figure 14. A view may include both C# and HTML code, as seen in the figure above. The C# code on the view is turned into plain HTML once it has rendered on the browser. View Engine's function is to translate C# code into pure HTML.

2.5.1 Razor View Engine Working

The views offer the website's fundamental structure and operation, including the page layout and data rendering. When a user accesses a page on a NopCommerce website, the platform initially searches the "Views" folder of the active theme for the relevant view. The page is rendered using a theme-specific view if one is discovered. The default view from the "Views" folder in the root directory is utilized in the absence of a theme-specific view.

2.6 ASP.NET Core

A well-liked web development framework for the .NET platform is ASP.NET. The free and open-source ASP.NET Core platform is compatible with Windows, Linux, and macOS. An updated version of older Windows-only ASP.NET versions, ASP.NET Core was originally made available in 2016 [16].

Table 1- Difference between ASP.NET and ASP.NET Core [18]

	ASP.NET	ASP.NET Core
Platform Support	Windows only	Windows, Linux and Mac
Architecture	Based on .NET Framework	Based on .NET Core
Components	Web Forms, MVC and Web API	MVC, Web API, Razor Pages, and Blazor
Dependencies	Less control over dependencies	Strict control over dependencies
File support	WCF, WF, WPF, VB, Web config, and more	No support for Global.asax, Web config
Visual Studio	Support of all versions	Support of latest versions from 2015

While still offering a stable and maintained foundation for apps to run on, ASP.NET Core is designed to allow runtime components, APIs, compilers, and languages to evolve quickly.

The same server can host many instances of ASP.NET Core concurrently. Meaning that one app can use the most recent version, while the other apps continue to use the version that they were tested on. In ASP.NET Core, performance is a top priority. Compared to other well-known web frameworks, it is quicker.

2.7 Cross Origin Resource Sharing

CORS, or cross origin resource sharing, is its full name. It permits a server to accept calls from the designated domains while refusing others. Due to browser security, a web page cannot by default make an Ajax request from one domain to another.

The software phrase for this security is "same-origin policy," and it prevents a suspensive site assault from reading sensitive data from another site. An error message will be returned by the AJAX request: On the requested resource, there is no 'Access-Control-Allow-Origin' header [17].

However, we frequently use multi-domain applications that require calls to be made from one domain to another; in this situation, we need to allow cross-origin policy. If the browser in question is CORS-compatible, it will automatically set the headers for cross-origin requests. If everything goes according to plan on the server, "Access-Control-Allow-Origin" header is added to the response. This type of request will fail if the answer does not contain the Access-Control-Allow-Origin header.

If two URLs are on the same domain, they have the same origin.

The origin of these two URLs is the same.

`https://test.com/index.html`

`https://test.com/about.html`

The origins of the subsequent URLs are distinct from those of the prior two URLs,

`https://hello.net`

`https://www.hello.com/foo.html`

CHAPTER-3

SYSTEM DESIGN AND DEVELOPMENTS

NopCommerce is a well-known e-commerce platform that is accessible as free software, giving companies the ability to easily launch an online store. It is built on the Microsoft.NET framework and offers a wide range of capabilities that let companies build fully functional e-commerce websites that can be tailored to meet particular needs. NopCommerce is an all-in-one solution for online businesses thanks to its capabilities, which include product administration, order processing, payment processing, shipment management, and customer management.

NopCommerce versatility is one of its key benefits because it was created utilizing a modular design, which enables simple addition or removal of features and functionality to suit business requirements. NopCommerce is also very scalable, so it can manage an increasing volume of merchandise, clients, and orders.

Furthermore, NopCommerce is equipped with a range of built-in security features that safeguard the website from malicious attacks. Regular updates ensure that NopCommerce stays current with the latest security threats and patches.

Overall, NopCommerce is a powerful and adaptable e-commerce platform, suitable for businesses of all sizes looking to establish an online presence and increase sales. Its wealth of features and customization options make it an appealing choice for businesses seeking to build an e-commerce website.



Figure 15- NopCommerce Technologies [6]

Figure 15, shows different NopCommerce Technologies that are used for working on this framework.

3.1 Hardware requirements

The size of the shop, the quantity of products, the number of concurrent users, and the anticipated volume of traffic will all affect the hardware requirements for hosting NopCommerce. However, the following are some general recommendations for the hardware needed to run NopCommerce:

Table 2- Hardware Requirements for Running NopCommerce

Processor	1 GHz
RAM	512 MB
Minimum Disk space	4.5 GB
Operating system	Windows, Linux, mac

3.2 Software Requirements

It's crucial to note that the specific software requirements may vary depending on the version of NopCommerce being used.

Table 3- Software Requirements for Running NopCommerce

Web browser	Google Chrome, Mozilla Firefox, or Microsoft Edge (any suitable browser)
Web Server	IIS, Apache, Nginx, Kestrel (any suitable server)
.NET Sdk	Dotnet-sdk-3.1.426 (or higher)
Operating system	Windows, Linux, mac (any)
NopCommerce framework	Version 4.30 (or higher)
Software	Microsoft SQL Server Management Studio 2019 and Microsoft Visual Studio

3.3 Functional Requirements

The functional requirements of NopCommerce include:

1. **Product management:** NopCommerce allows store owners to manage their products efficiently by adding or removing products, updating product descriptions, pricing, and availability, and managing inventory.
2. **Order processing:** NopCommerce allows store owners to manage their orders from the admin panel, view order details, update order status, and manage returns and refunds.

3. **Payment processing:** Numerous payment methods are supported by NopCommerce, including credit cards, PayPal, and other payment gateways. From the admin panel, store owners can set up payment options and control payments.
4. **Shipping management:** Store owners may manage shipping choices with NopCommerce, including setting up shipping providers, adjusting shipping rates, and tracking orders.
5. **Customer management:** Numerous customer management features are offered by NopCommerce, such as managing customer accounts, viewing order histories, managing customer groups, and sending customers marketing emails.
6. **Marketing and promotions:** NopCommerce include a range of marketing and promotional features, including managing discounts and coupon codes, managing product reviews, and setting up email marketing campaigns.
7. **Localization:** NopCommerce is appropriate for enterprises that operate in various countries since it supports multiple languages, currencies, and tax regulations.
8. **Security:** Numerous security features are already built into NopCommerce, such as PCI compliance, SSL support, and defenses against SQL injection and cross-site scripting attacks.

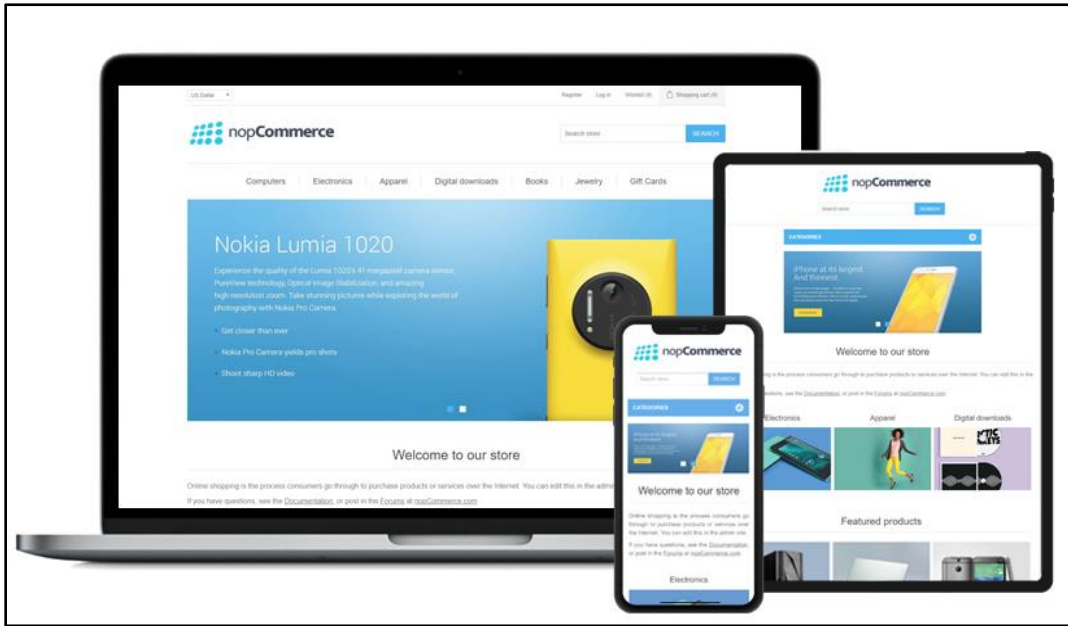


Figure 16- NopCommerce Store-front side [7]

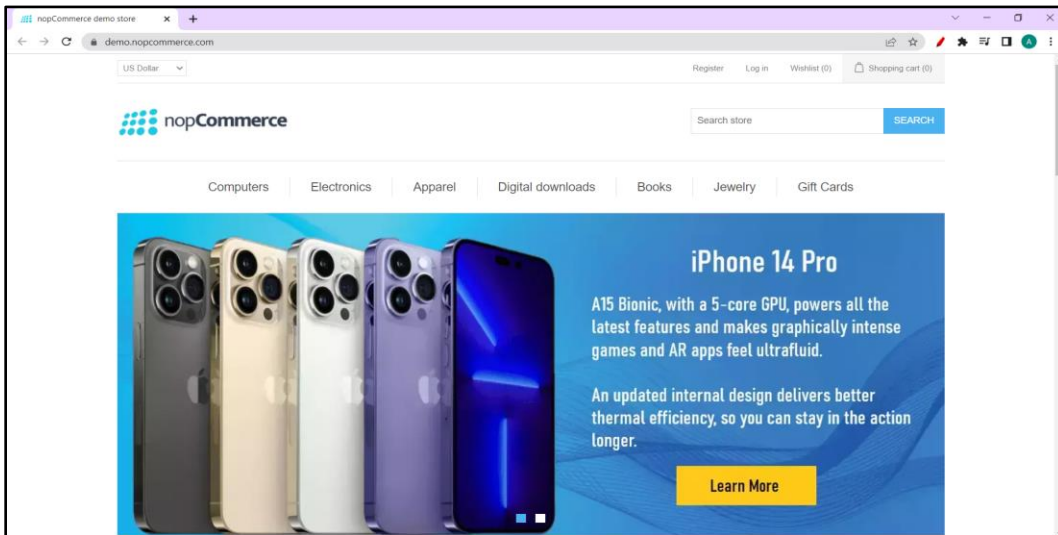


Figure 17- Home page of NopCommerce website

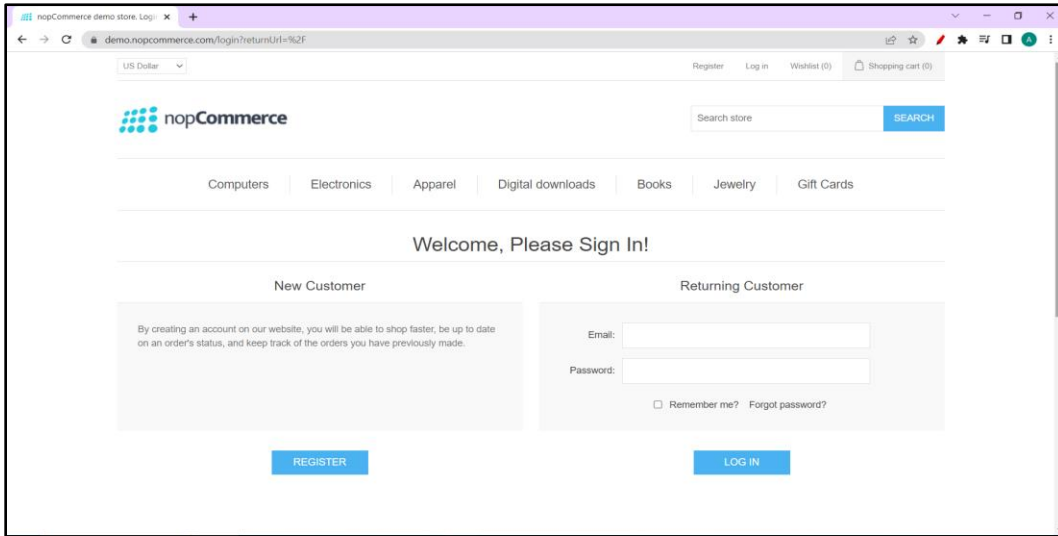


Figure 18- Login page of NopCommerce Website

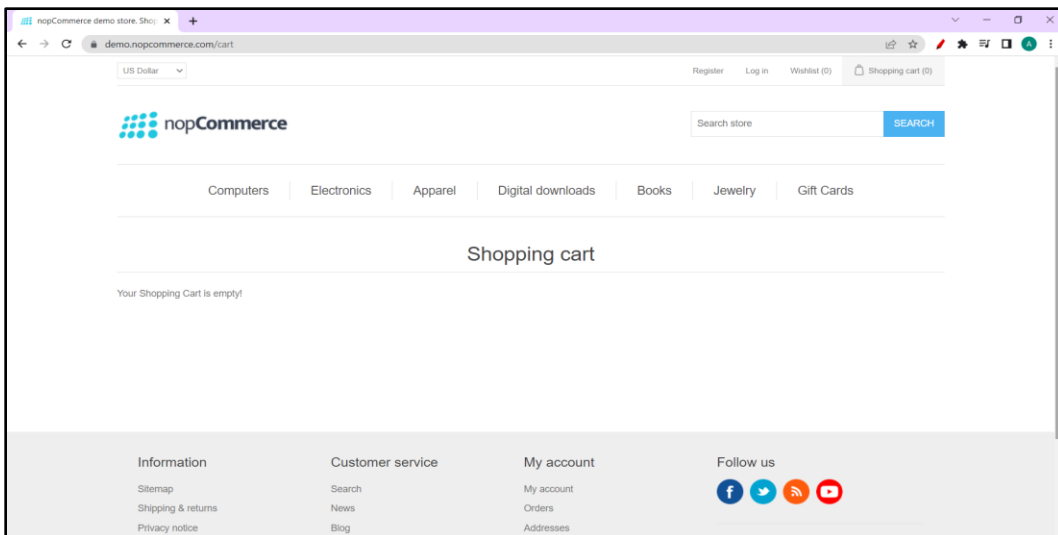


Figure 19- Shopping Cart page of NopCommerce Website

The figures 16 to 19, show the store-front side view of the NopCommerce Website. Particularly the Login Page, Shopping Cart, Register form and the home page.

3.4 Project Development:

3.4.1 MVC Lifecycle:

MVC (Model-View-Controller) is a popular design pattern used in software development. The lifecycle of an MVC application can be broken down into the following steps:

1. **User Interaction:** In an MVC application, the user can communicate with the View component. The View component receives user input like as mouse clicks, keyboard inputs, and touch events and displays the user interface, which contains items like forms, buttons, and menus.
2. **Handling requests:** When a user interacts with the View component, the Controller component receives a request from the View component. Typically, the request includes details of the user's action, such as the data entered into a form or the button clicked. The request is received by the Controller, who then uses the application's business logic to decide what should happen next.
3. **Model Interaction:** After deciding on the best course of action, the Controller engages with the Model component. The data and business logic of the application are represented by the Model. Depending on the user's action and the needs of the program, the Controller could obtain data from the Model or modify data there. For instance, the Controller might evaluate data entered by the user into a form before saving it to the Model.
4. **View Update:** After updating the Model, the Controller passes control back to the View component. The View component then modifies how the application's data is displayed in response to changes made to the Model. For instance, the View might show the updated data in a table if the user input data into a table or chart.

5. **User Feedback:** The updated View is presented to the user, and the user may interact with the application again. The cycle begins again with the user interacting with the View component.

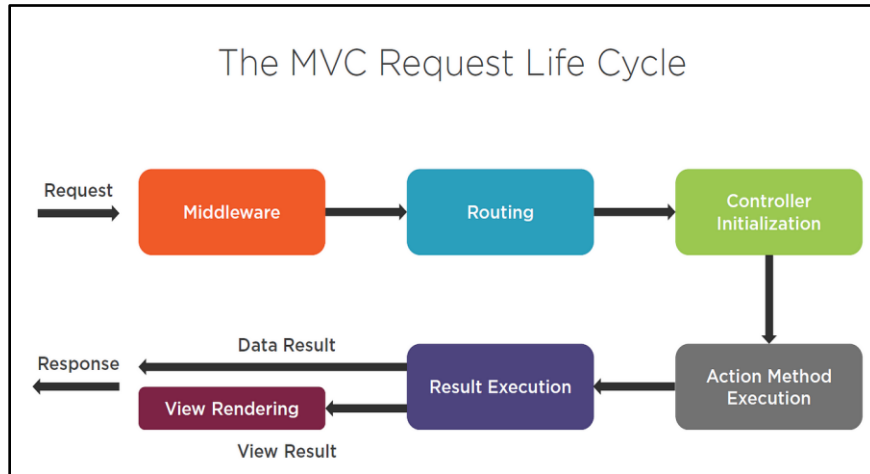


Figure 20- MVC Request Life Cycle [8]

Throughout the MVC lifecycle, the three main components of the pattern work together to create a responsive and maintainable application. Here's a closer look at each component in figure 21:

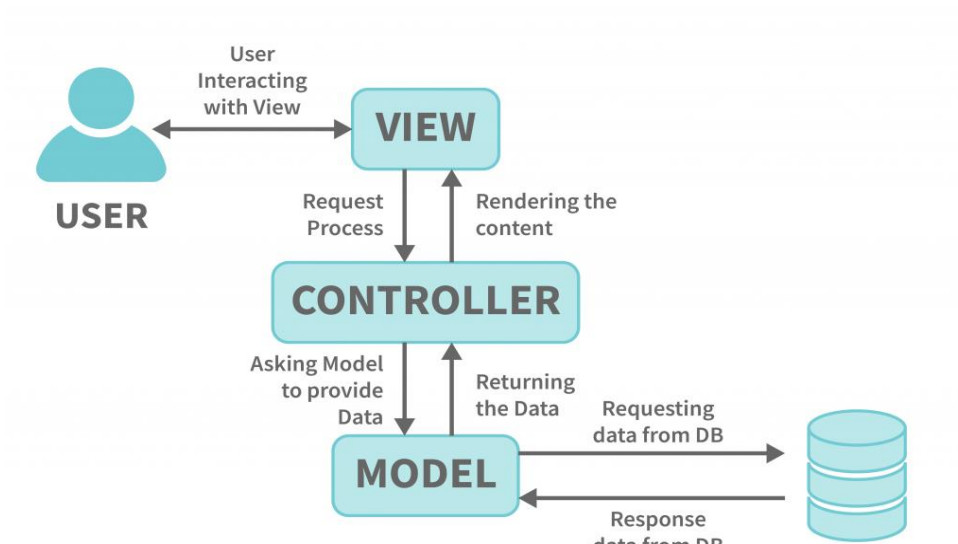


Figure 21- Working of MVC [9]

1. **Model:** The data and business logic of the application are represented by the Model component. It controls data storage, retrieval, and change while defining the data's structure and behavior. The user or the View component are not in direct communication with the Model. Instead, it gives the Controller a way to access and alter data.
2. **View:** The View component shows the user interface and makes the data from the application visible. User input is taken into account as it sends requests to the Controller component. The View concentrates on display and user interaction rather than application logic.
3. **Controller:** This component manages user input and chooses the best course of action. In order to retrieve or update data, it communicates with the Model component. Control is then transferred back to the View component, which updates the user interface. The Controller is in charge of organizing the interaction between the Model and View components and houses the business logic for the application. MVC offers a flexible and modular architecture for creating complex applications by decoupling concerns in this manner.

3.4.2 Services in NopCommerce

In NopCommerce, a service is a class that implements a specific functionality of the application. Services are typically used to perform operations such as data access, business logic processing, and communication with external systems. There are two main types of services in NopCommerce:

1. **Application Services:** The main features of the program, such as managing products, orders, customers, and other entities, are provided by these services.
 1. AddressService: Offers tools for managing addresses.
 2. BlogService: Offers tools for managing blog entries and comments.

3. The CategoryService offers tools for managing product categories.
4. CheckoutService: Offers tools for controlling the checkout procedure.
5. CustomerAttributeService: Offers tools for managing unique customer attributes.
6. Customer Service: Offers tools for managing customers.
7. LocalizationService: Offers tools for controlling localization preferences.
8. ManufacturerService: Offers tools for controlling the manufacturers of goods.
9. PictureService: Provides functionality to manage product images.
10. PictureService: Offers tools for organising product photographs.
11. ProductService: Offers tools for managing products.
12. ProductTagService: Offers tools for controlling product tags.
13. RewardPointsService: Offers tools for customers to manage their reward points.
14. SettingService: Offers tools for controlling application settings.

2. Infrastructure Services: These services offer low-level features like email sending, caching, and logging that assist application services.

1. CacheService: Offers capabilities for managing caching.
2. ContentManagementService: Offers tools for controlling content.
3. EmailSenderService: Offers tools for controlling email sending.
4. EncryptionService: Offers tools for controlling encryption.
5. ILoggerService: Offers logging management capabilities.

3.4.3 Repositories in NopCommerce

A repository is a class in NopCommerce that offers a data access layer for working with the database. In order to hide the specifics of the underlying database technology and to retrieve and manipulate data from the database, repositories are typically used.

A contemporary, open-source, cross-platform object-relational mapping (ORM) framework called Entity Framework Core enables programmers to interact with databases using .NET objects. With support for numerous database providers, such as SQL Server, MySQL, PostgreSQL, SQLite, and more, it is a compact and adaptable version of Entity Framework.

Entity Framework Core has a wealth of features and a straightforward API that make it simple to interact with databases. It is made to be flexible and simple to use. It has the following characteristics-

1. **Code-First Approach:** Using Entity Framework Core, you may create the database schema from scratch using your object-oriented C# or VB.NET code as a starting point.
2. **Migrations:** The robust migration system in Entity Framework Core enables you to control changes to your database structure over time.
3. **LINQ Support:** Language Integrated Query (LINQ) is a feature of Entity Framework Core that enables you to write queries on your data in either C# or VB.NET syntax.
4. **Transactions:** Transaction support in Entity Framework Core enables you to carry out multiple database operations in a single transaction.
5. **Change Tracking:** The change tracking mechanism in Entity Framework Core automatically identifies data changes and prepares the SQL statements required to persist those changes to the database.
6. **Lazy Loading:** Lazy loading is supported by Entity Framework Core, allowing you to defer the loading of related data until you need it.

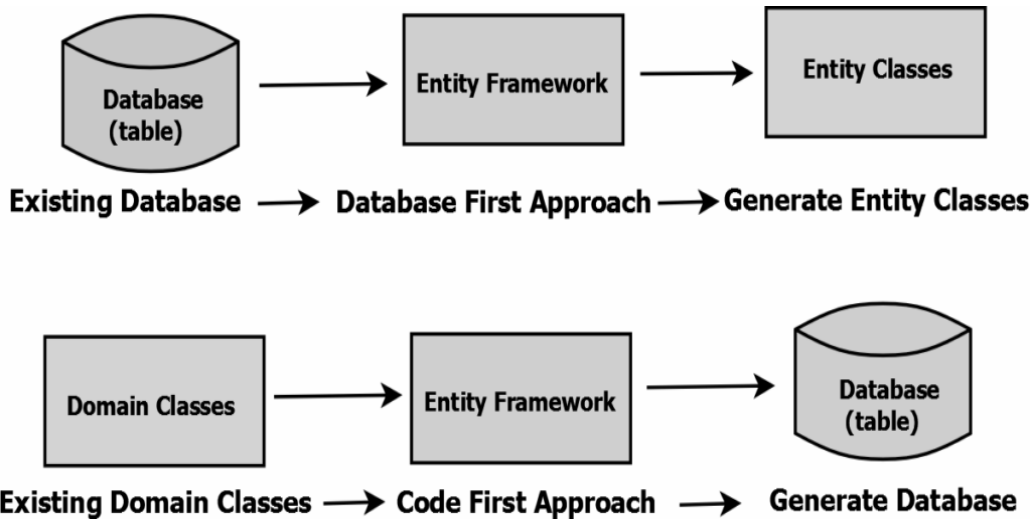


Figure 22- Approaches of Entity Core Framework

Figure 22 shows the 2 approaches for Entity Framework Core. Database first approach means to generate classes based on the database. Whereas, Code first approach means to generate the database based on classes. It works by mapping your model classes to database tables, and providing a simple API for querying and manipulating data in those tables. Annotations and a fluent API can be used to specify how a model class in Entity Framework Core should be mapped to a database table when the class is defined. For instance, you may give the name of the table using the [Table] element and each column's name and data type using the [Column] attribute.


```

[Table("Customers")]
public class Customer
{
    [Key]
    public int Id { get; set; }

    [Column("FirstName")]
    public string FirstName { get; set; }

    [Column("LastName")]
    public string LastName { get; set; }

    [Column("Email")]
    public string Email { get; set; }
}

```

Figure 23- Code Snippet showing the Customer Class

Figure 23 of this example, the [Table] property instructs the database to map this class to a table called "Customers" for this example's class. The [Key] attribute indicates that the Id property should serve as the table's primary key, while the [Column] attribute indicates the names and data types of each column.

Using the DbContext class, you can communicate with the database after defining model classes. The DbContext presents a session with the database and offers a number of methods for data manipulation and querying.

Figure 24 provides an illustration of how the DbContext can be used to fetch a list of customers from the database:

```

using (var context = new MyDbContext())
{
    var customers = context.Customers.ToList();
}

```

Figure 24- Code Snippet to retrieve the list of customers from database

3.4.4 Razor Pages

ASP.NET Core's Razor Pages feature enables programmers to create dynamic web pages by combining C# code and HTML content. Razor Pages adheres to the Model-View-Controller (MVC) design but takes a more straightforward approach that places an emphasis on code separation and user friendliness. Each web page in Razor Pages is represented by a Razor Page file (.cshtml), which also contains the C# code that creates the dynamic content. The Razor syntax makes it simple to create dynamic content and manage user interactions by allowing you to integrate C# code right into your HTML layout.

```
@page
@model ProductListModel

<h1>Products</h1>

<ul>
    @foreach (var product in Model.Products)
    {
        <li>@product.Name - @product.Price</li>
    }
</ul>
```

Figure 25- C# code in razor page to display product-name and product-price

Figure 25 serves as an illustration of this. The "@page" directive identifies this file as a Razor Page, and the "@model" directive identifies the kind of model this page employs. A foreach loop that iterates across the model's list of products is used to create an unordered list and a heading in the HTML markup.

3.4.5 Partial Views

When a common user interface element is required on several web application pages, a partial view is developed. A standard view with the file extension .cshtml

that may be used numerous times within an application is what is known as a partial view. To divide a web page into distinct sections like the header, footer, and menu, Layout occasionally uses partial views. Other examples include remarks made on blogs, shipping and payment information on statements from online shops, etc.

3.4.6 Layout

The majority of online apps share a uniform layout, giving users a consistent user experience as they move between pages. Common user interface components like the app header, menu items, and footer are often included in the layout [19].

Many pages in an app frequently make use of scripts and stylesheets, two common HTML building blocks as shown in figure 26.

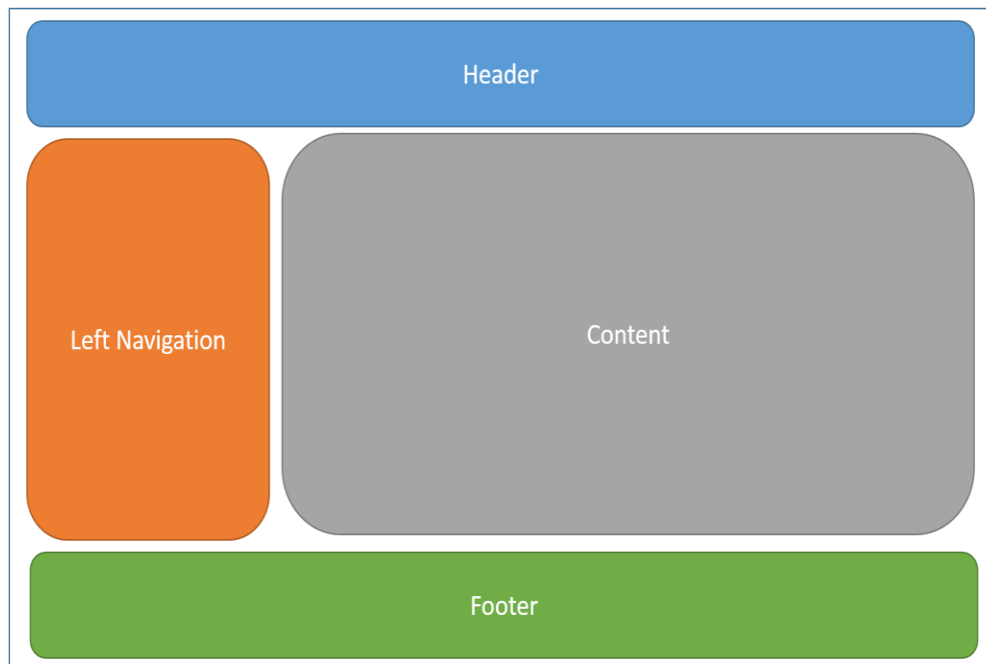


Figure 26- Common Layout used in Web Applications [19]

Any view used by the app may then refer to the layout file that contains all these shared elements. Layouts help views have less redundant code.

By convention, an ASP.NET Core app's default layout is called ‘_Layout.cshtml’.

3.4.7 Database Implementation

The database used by NopCommerce for data storage is Microsoft SQL Server. The database is made up of numerous tables for storing different kinds of data, including customer details, product details, order details, shipping details, and more. NopCommerce connects to the database using Entity Framework Core, an open-source ORM (Object-Relational Mapping) framework. Entity Framework Core maps NopCommerce objects to the proper database tables and executes SQL queries to read or write data. A scalable and efficient database schema has been developed. It follows the best practises for database design, such as the need for data integrity and normalisation. By adding or removing tables, indexes, or foreign key constraints, NopCommerce developers can easily change the database's organisational structure. NopCommerce also provides a database migration tool that helps programmers to construct or modify the database schema in an organised manner. The migration tool enables version control of the database schema, making it simple for developers to reverse changes or apply them to other settings. The following is a list of some of the important tables in the NopCommerce database:

1. Address: Stores shipping and billing addresses of customers.
2. Category: Contains information about the categories of products offered in the store.
3. Customer: Contains information about registered customers such as name, email, and password.
4. Manufacturer: Contains information about product manufacturers.
5. Order: Stores order details such as customer, product, order date, and order status.

6. Product: Contains information about the products available in the store such as name, description, price, and quantity in stock.
7. Product_Category_Mapping: Represents the many-to-many relationship between products and categories.
8. Product_Manufacturer_Mapping: Represents the many-to-many relationship between products and manufacturers.

3.4.8 Dependency Injection

Object-oriented programming frequently employs the design pattern known as Dependency Injection to encourage loose coupling between components and to make the code easier to maintain and test. Simply put, dependency injection is a method of giving a class the objects or services it needs at the time of creation rather than having the class establish those dependencies on its own. Here is a straightforward example to show how Dependency Injection works:

Think about a class called "CustomerService" that must email a customer once they place an order. We may use Dependency Injection to provide an instance of the email service to the "CustomerService" class rather than generating a new instance of the email service inside the "CustomerService" class.

Figure 27 shows an interface for the email service:

```
public interface IEmailService
{
    void SendEmail(string to, string subject, string body);
}
```

Figure 27- Code Snippet showing the IEmailService Interface

Next, the email service class implements the IEmailService Interface as shown in Figure 28:

```
public class EmailService : IEmailService
{
    public void SendEmail(string to, string subject, string body)
    {
        // Code to send email
    }
}
```

Figure 28- Implementation of IEmailService interface

Finally, the `CustomerService` class is modified to accept an instance of the email service through its constructor as shown in figure 29:

```
public class CustomerService
{
    private readonly IEmailService _emailService;

    public CustomerService(IEmailService emailService)
    {
        _emailService = emailService;
    }

    public void PlaceOrder(Order order)
    {
        // Code to place order

        _emailService.SendEmail(order.Customer.Email, "Order Confirmation",
    }
}
```

Figure 29- Code Snippet for CustomerService class

Now, instance of the `CustomerService` class is created, an instance of the email service is provided to its constructor as shown in figure 30:

```
IEmailService emailService = new EmailService();  
CustomerService customerService = new CustomerService(emailService);
```

Figure 30- Code Snippet for providing an instance of email service to its constructor

In this approach, the 'CustomerService' class is separated from the specifics of how the email service is implemented, increasing its flexibility and making testing simpler. Future usage of a different email provider can be accommodated by simply providing an alternative implementation of the 'IEmailService' interface to the 'CustomerService' class, without modifying the class's functionality.

3.4.9 Task Implementation

For my training project, I had to customize NopCommerce to extend its functionality. The platform enables quick display or concealing of field names or properties on the client side with only one click from the admin side. The action method, models, or services that are relevant must be changed on both the admin side and the front side of the store, but, if the property or field you require is missing on the admin side. For instance, my first assignment was to expand the register form's middle name field. In the "CustomerFormFields" model on the admin side, this is absent.

Admin Side:

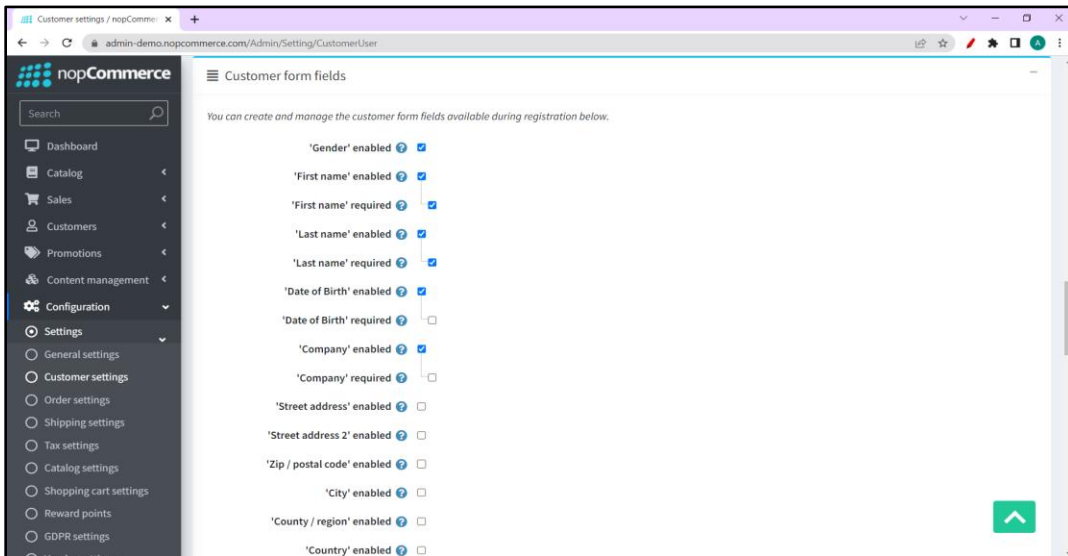


Figure 31- Customer form fields (admin view)

Figure 31 shows the admin view of the customer form fields. If the fields are enabled here, the view on the store-front is automatically updated to show the field that is enabled.

Customer-Facing Store Front:

The screenshot shows a web browser window with the title 'Register'. The browser's address bar contains the URL 'demo.nopcommerce.com/register?returnUrl=%2F'. The page content is organized into three main sections: 'Your Personal Details', 'Company Details', and 'Options'. The 'Your Personal Details' section includes a form with the following elements: a 'Gender' field with radio buttons for 'Male' and 'Female'; 'First name' and 'Last name' text input fields; a 'Date of birth' field with three dropdown menus for 'Day', 'Month', and 'Year'; and an 'Email' text input field. The 'Company Details' section contains a single 'Company name' text input field. The 'Options' section is currently empty.

Figure 32- Register page of the store-front side (default view)

Figure 32 shows the register page of the store-front view. All the fields in the customer-form fields of admin view that are enabled, will be displayed in this form.

1. The command to show the registration page is sent to the "Customer" controller.
2. Calling the "PrepareRegisterModel" method of the "CustomerModelFactory" produces a "RegisterModel".
3. The "RegisterModel" provides boolean attributes to specify which fields are made available for viewing as well as properties to store the data entered by the user.
4. In order to add the "Middle Name" field, a new property called "MiddleName" is added to the "RegisterModel".

5. When the "PrepareRegisterModel" method is called, the properties of the "RegisterModel" and the "CustomerSettings" model are now mapped.
6. The "RegisterValidator.cs" class has been changed to include validations that make sure the user-provided middle name doesn't have any extra digits.
7. The "RegisterModel" is passed as a parameter to the "Register.cshtml" Razor view, which is in responsibility of displaying the user's registration page.
8. The "Register.cshtml" view now adds a new input field for the "Middle Name" attribute of the registration form.
9. Once the user submits the form, data is collected from each field, including the new "Middle Name" field, and given to the appropriate action method to finish the registration request.

Overall:

1. On the admin side, the "CustomerSettings" model must be enabled to add a new field, and the "RegisterModel" on the customer-facing store front must have a new property added.
2. When the user submits the registration form after the new field has been enabled and added, the data can be automatically stored in the database.
3. To make the necessary adjustments, it is crucial to monitor the process and procedures being used on both the admin and customer-facing store front sides.

My next task was to store the middle name data in the customers' database and display it on the admin side table.

The screenshot shows the nopCommerce admin interface with the 'Customers' table. The table has the following columns: Email, Name, Customer roles, Company name, Active, and Edit. The data rows are as follows:

Email	Name	Customer roles	Company name	Active	Edit
<input type="checkbox"/> YjdxT@gmail.com	Mohit Yadav	Registered	Zento	✓	
<input type="checkbox"/> Fnhvi@gmail.com	Mohit Yadav	Registered	Zento	✓	
<input type="checkbox"/> kEEUq@gmail.com	Krishna Reddy	Registered	Y.PG	✓	
<input type="checkbox"/> kiyjcyhj676008@gmail.com	Virat Kohli	Registered	Indian Cricket Team	✓	
<input type="checkbox"/> victoria_victoria@nopCommerce.com	Victoria Terces	Registered		✓	
<input type="checkbox"/> brenda_lindgren@nopCommerce.com	Brenda Lindgren	Registered		✓	
<input type="checkbox"/> james_pan@nopCommerce.com	James Pan	Registered		✓	
<input type="checkbox"/> arthur_holmes@nopCommerce.com	Arthur Holmes	Registered		✓	
<input type="checkbox"/> steve_gates@nopCommerce.com	Steve Gates	Registered		✓	
<input type="checkbox"/> admin@yourStore.com	John Smith	Administrators, Forum Moderators, Registered		✓	

Figure 33- Customers Table (default admin view)

To achieve this, I had to analyze the workflow and identify which controller and action method are being called when the user submits the data. Figure 33 shows the default admin view of the customers table that stores the customer details entered during registration. In this case, the HTTP "POST" verb is used to submit the form data, and the action method responsible for processing this data is called.

I had to use a function from the proper model factory, which in turn calls the customer service, to store the middle name information in the database. The data is subsequently stored in the database by the customer service using the repository approach. The task of connecting to the SQL server and putting the data in the right table falls on the data provider. To be more precise, the customer service class is in charge of overseeing customer-related tasks like adding new clients, updating current clients, erasing clients, and retrieving client-related information. The data access functionality is abstracted from the service layer by the repository layer, with which the service class communicates.

To communicate with the database, the repository class invokes the relevant data provider methods. The "Customer" table, which contains the customer's basic information such name, email, and password, is one of numerous tables in the database that store the customer-related data. The "Address" field keeps track of the billing and delivery addresses for the client, and the "Customer Role" table keeps track of any responsibilities that have been given to them, like customer or administrator. The custom characteristics connected to the customer, such middle name in this instance, are kept in the "CustomerAttributeValue" table.

In conclusion, utilizing the proper service, repository, and data provider classes to communicate with the SQL server and store the data in the proper table is necessary for saving the middle name data in the database. As part of this training project, I was also given a number of additional tasks to complete. For example, I had to change the product details page so that the manufacturer's name and a link to the product are displayed beneath the product's name. I also had to add validations to ensure that users couldn't add products from different companies in the same order.

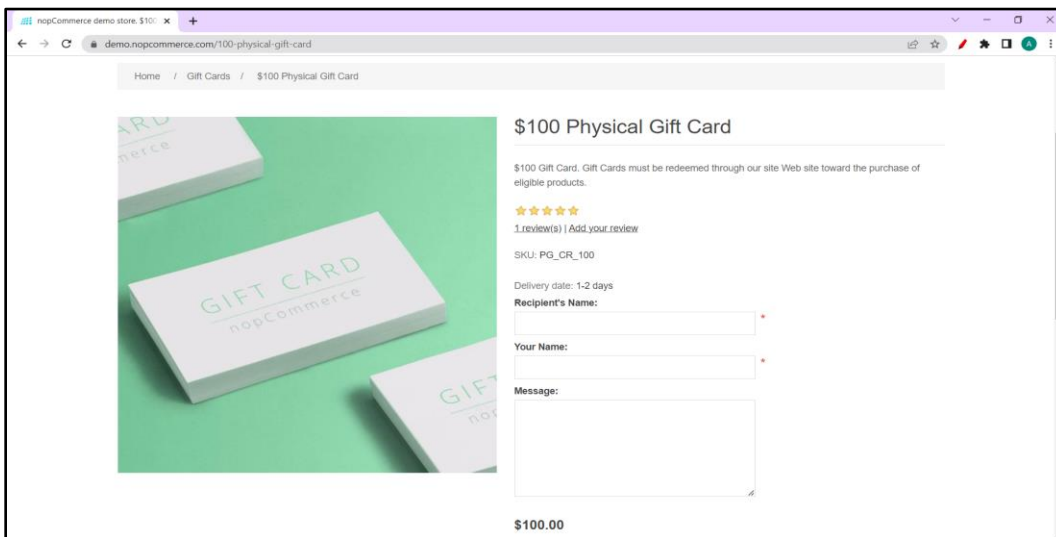


Figure 34- Default Product Details view for the Gift Card (without the Link)

Figure 34 shows the default product details page without the link of the product

CHAPTER-4

RESULTS

4.1 Results

Figure 35 shows the admin view for the customer user page. This customer user page has used various partial views for the different sections- Common, Customer form fields, Account, Password and Security, profile, and Address Form fields. On enabling the properties in these fields, the corresponding store-front side view is updated.

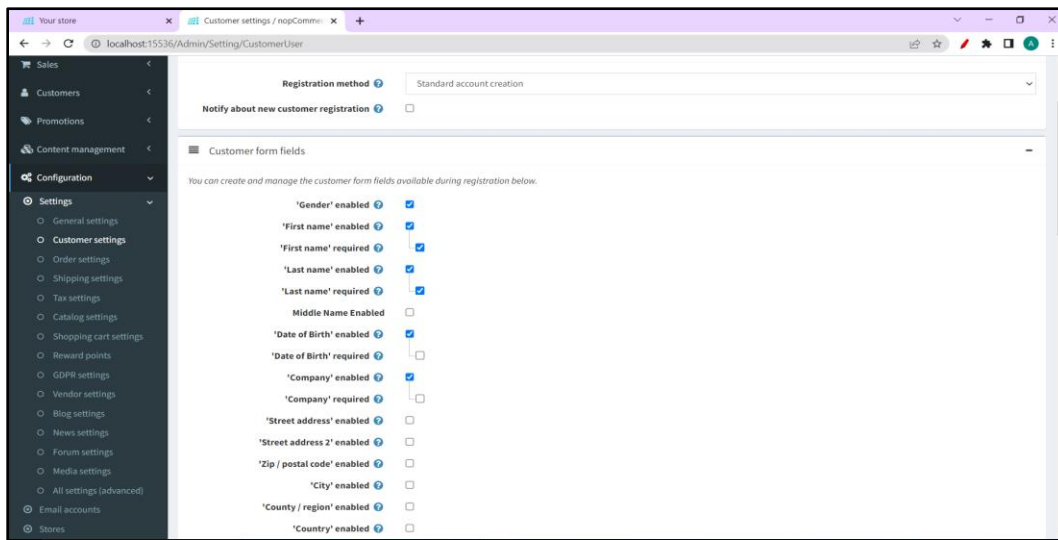


Figure 35- Admin view for customer form fields (with middle name property)

In case the property isn't present in the default admin view, the steps mentioned in the implementation are followed. The whole modification process depends on the understanding of the MVC lifecycle and the execution workflow of NopCommerce. New properties have to be added by modifying the code in the admin side and also on the store front side. Here, a new property "Middle Name" is added by modifying the "CustomerUserSettingsModel" and "CustomerModel" on the admin side and "RegisterModel" on the store-front side. For adding validations on the store-front side, "RegisterValidator.cs" class can be modified to add the suitable validations

for the corresponding form fields. The changes will be automatically reflected on the store-front side.

The following 36 to 44 show the results that were obtained –

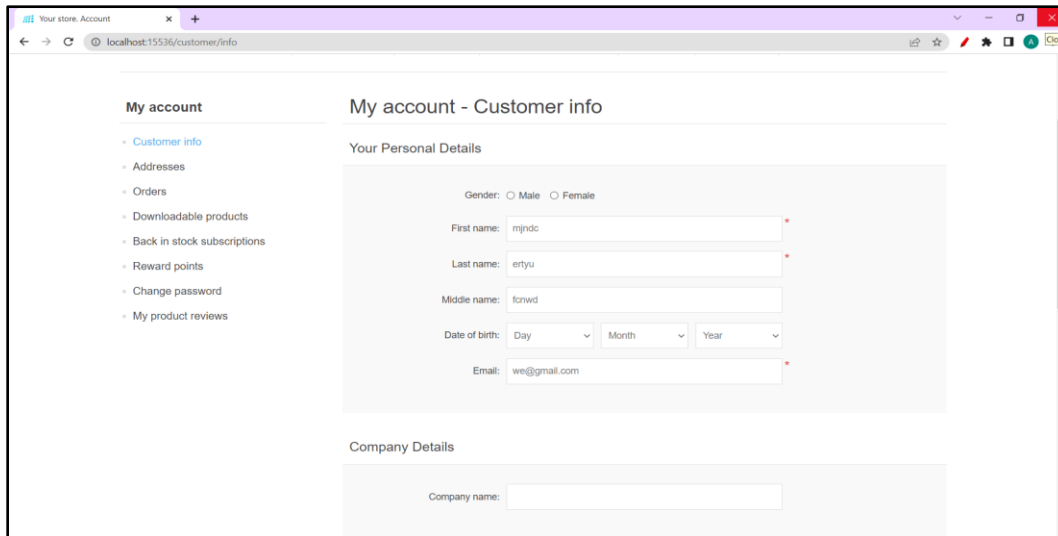


Figure 36- Store-front side view for the account info (with middle name field)

Figure 36 shows the Account details for the registered customer. In the default “AccountInfo” page, the “MiddleName” field isn’t present. Here, after the changes being made to the corresponding models, controller’s action methods, services, “MiddleName” field has been added. When the Account button is pressed on the navigation bar, a http get request is sent to the server and the respective action method of the respective controller that handles the request, returns the view required. The execution is as follows-

1. The request goes to the action method in the controller. All the public methods in the controller class are known as action methods.
2. The action method is responsible for returning the appropriate action result. In this case, firstly the “Info” action method is called. It calls the function

in the “CustomerModelFactory” by the name “PrepareCustomerInfoModel”.

3. In this function, mapping is done between the “CustomerInfoModel” and the “CustomerSettingsModel” so as to know which fields are enabled to be shown on the store-front view side.
4. Also, for the properties responsible for containing the user data in this model, they are rendered by the data with the help of the “GetAttribute” function of the “GenericAttributeService”.
5. This service is responsible for fetching the required attribute values from the database and returning it to the required model property with the help of Entity Framework Core.
6. The process of fetching data from the database is as follows- Request is sent to the respective service, service calls the respective repository, the request is further sent to the data provider, which is further responsible for making the connection with the database and returning the data.

The image shows a web browser window with a single tab titled "Your store: Register". The address bar contains "localhost:15536/register?returnUrl=%2F". The page content is titled "Register" and is divided into two main sections: "Your Personal Details" and "Company Details".

Your Personal Details

Gender: Male Female

First name:

Last name:

Middle name:

Date of birth: Day Month Year

Email:

Company Details

Company name:

Figure 37- Register page on store-front side (with middle name property)

Figure 37 shows the register page on the store-front side, inclusive of the “MiddleName” property. This property isn’t initially present in the default view of the Register Page and neither on the admin side Customer Form Fields. When the user clicks the “Submit” button, the Register action method with the Http POST verb, in the “Customer” Controller is called. In this method, first it checks whether the user is already registered or not. If yes, then the following error is shown else the user is registered successfully. In case the user is already registered, no new registration is possible and so the user is registered as a guest customer by “CustomerService” and the following view in figure 38 is displayed.

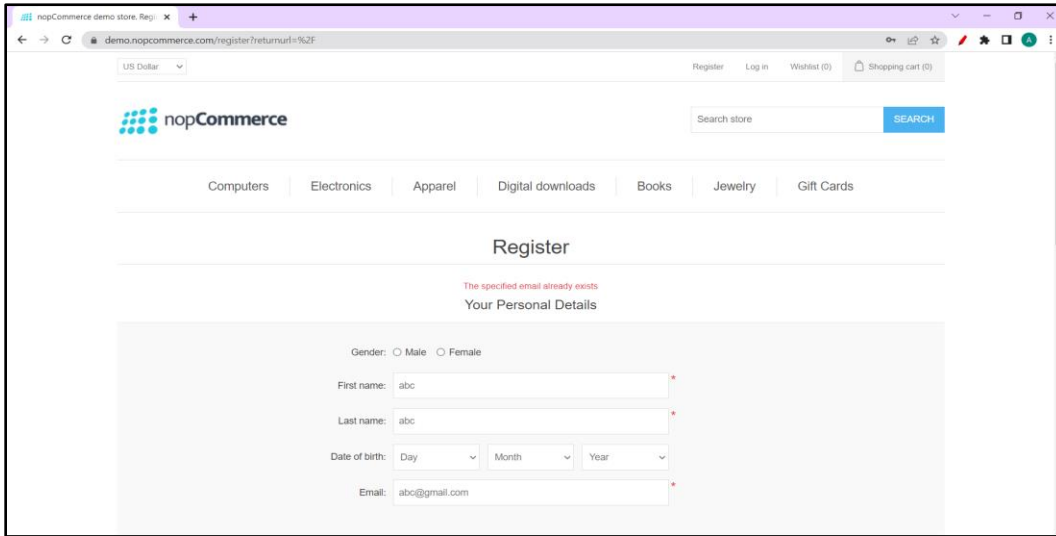


Figure 38- Store Front View showing the Register page for already registered customer

If the customer isn't registered before, the form details are checked for the errors. In case of errors, Errors are shown which implies the Model State isn't valid. On the other hand, if the ModelState is valid, a new Customer Registration Request is sent which contains the email, password, store id details, username, etc of the customer which are specific to that customer. If the registration request result is successful, then the entered values, which are stored in the register model, are stored in the database with the help of the "SaveAttribute" function of the "AttributeService". The form values are also used to create other properties like the address of the customer and are stored in the database.

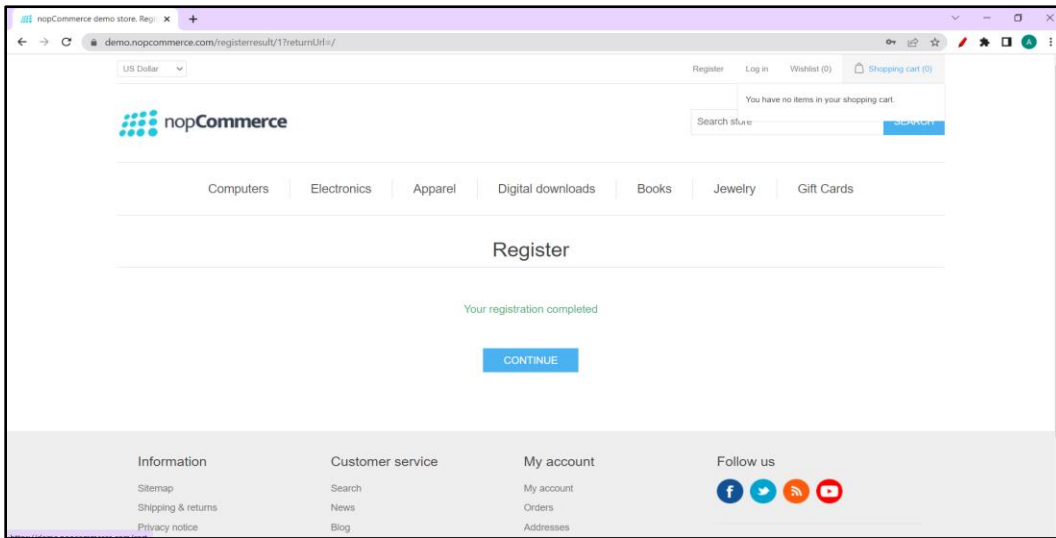


Figure 39- Register page View on Successful Registration

Figure 39 shows the register page on successful registration of the customer.

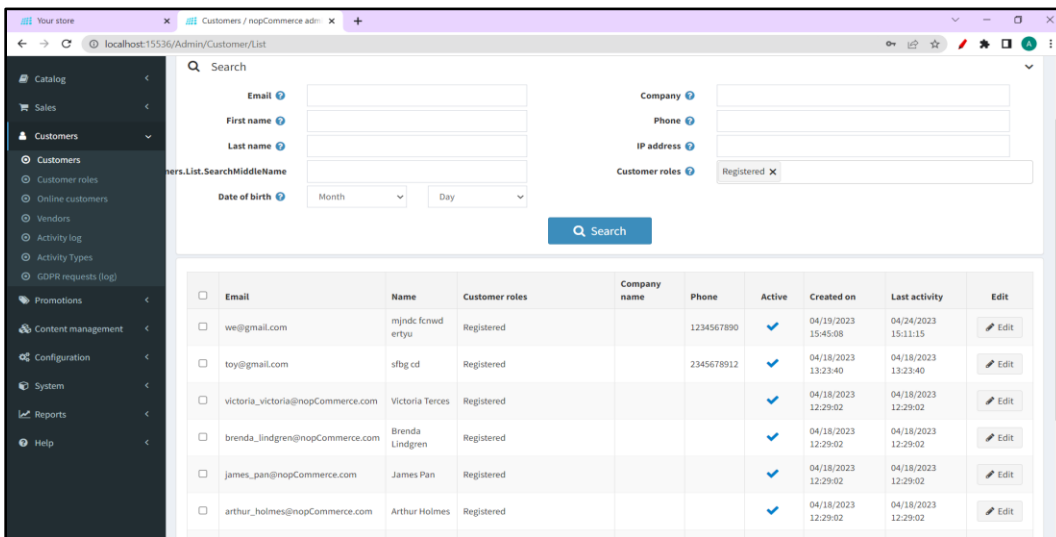


Figure 40- Customers data table (with middle name property appended in the Name column)

Figure 40 shows the customers data that is fetched from the database following the same execution flow i.e., Controllers → Action Method → Factory → Model Preparation → Service → Repository → Data Provider → Model rendered with

data → View returned with Model. For the middle name, it has been appended in the “Full Name” column as shown by the first row i.e., the most recent registration.

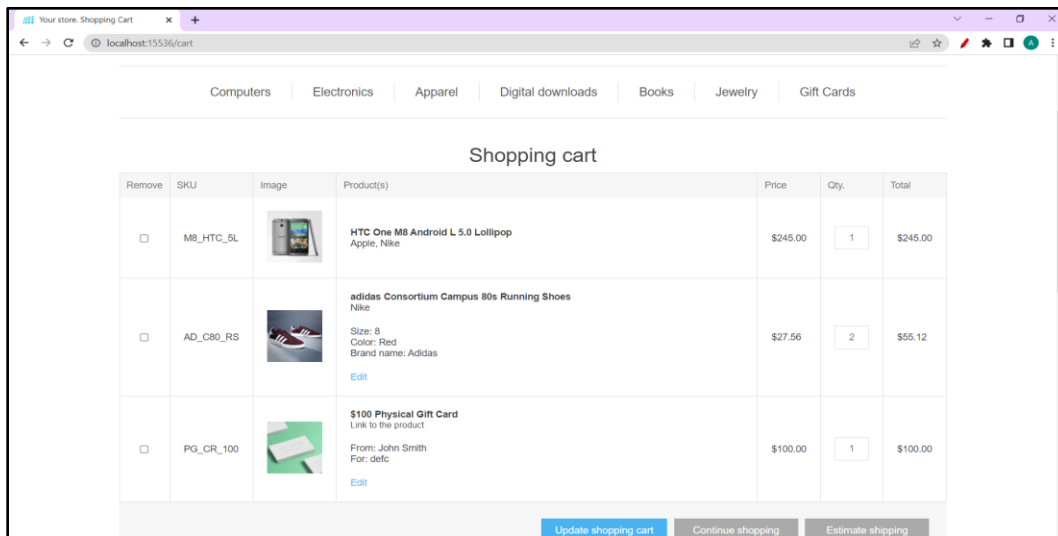


Figure 41- Shopping cart (with the manufacturer and link of the product (if given))

Figure 41 shows the Shopping Cart with the manufacturers under the product name and the link to the product (if given). The products are added at the admin side. Under the catalog→ Products shows the products in the store. To modify the details of the product, the edit button in the edit column is clicked for the specific product. In case the field to be added like “Link” isn’t present on the “EditProductDetails” Page, the execution flow has to be tracked and the corresponding view and models have to be modified.

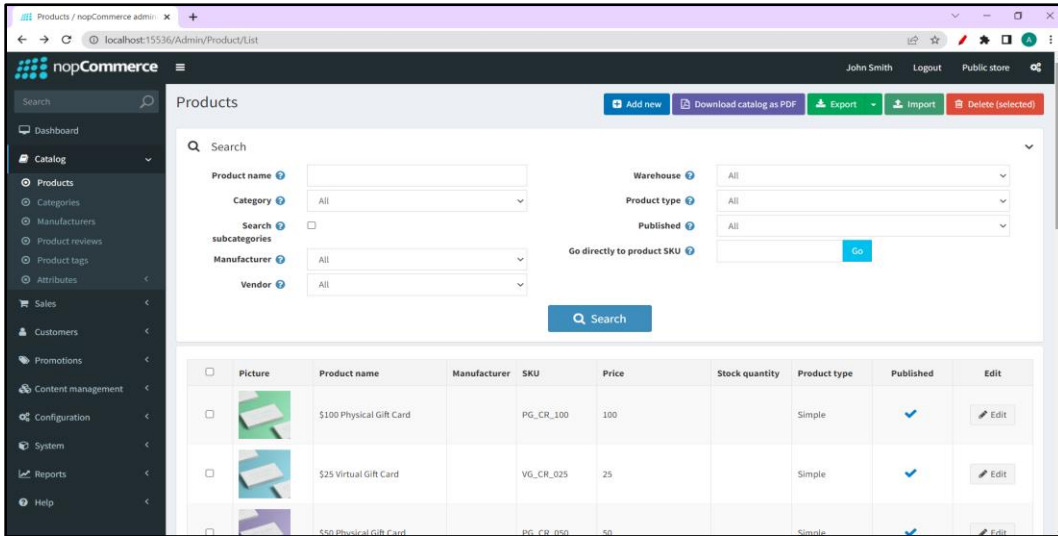


Figure 42- Admin side view of the Products Catalog page

Figure 42 shows the admin view of the Products Catalog Page which stores the Product details in the store.

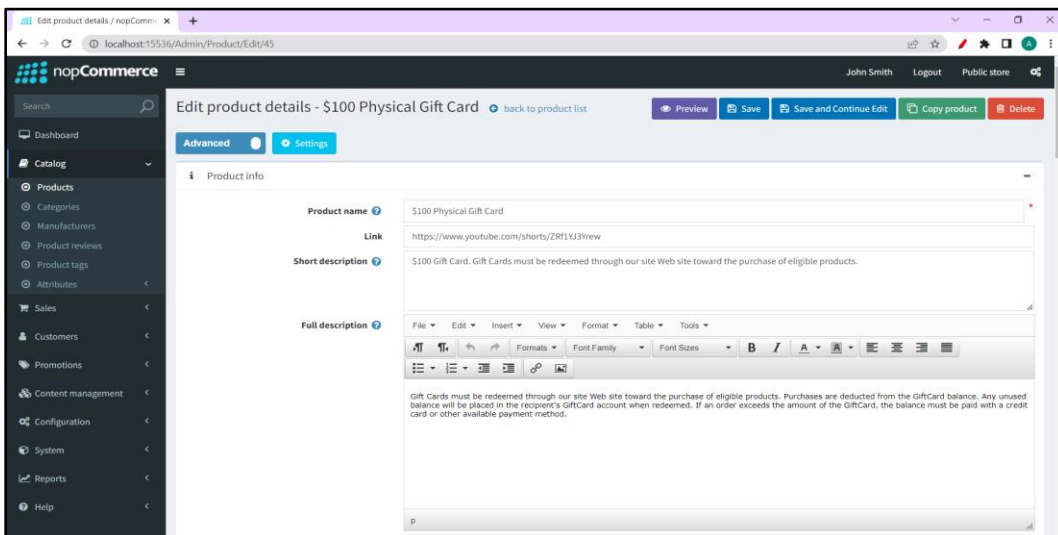


Figure 43- Edit Product Details page (admin view)

Figure 43 shows the “EditProductDetails” page (admin view) with the Link property added in the “ProductInfo” Model via code modification. The same is stored in the database.

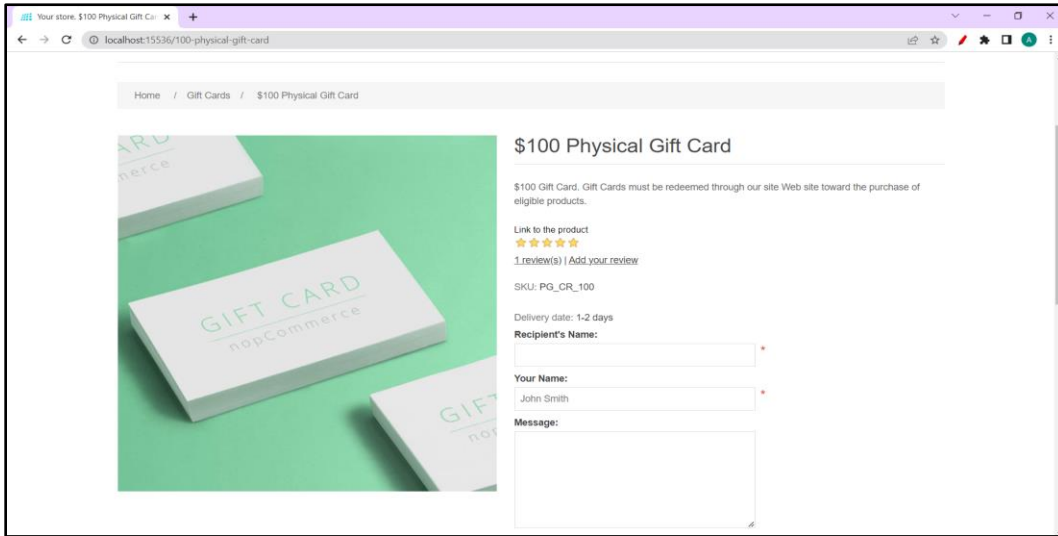


Figure 44 - Product Details page for the Gift Card with the Link of the product

Figure 44 shows the “ProductDetailsPage” for a particular product. The “ProductDetails” model has been modified to store the link. On clicking on the product, the link property is fetched from the database and stored in the model. The model is rendered to the view.

CHAPTER-5

CONCLUSIONS

5.1 Conclusion

The primary goal of the assigned project was to develop a thorough understanding of server-side programming for the building of websites. Through this project, I was able to learn in-depth information about the .NET framework, NopCommerce, how it functions, and the various parts and services used when creating an application. My learning experience was made much more interesting and rewarding by getting to use the technology firsthand and work through problems from the real world.

It was a completely new field for me, so I initially ran into some difficulties during the project's first phase. But with time, I was able to spot and fix these issues, which allowed me to finally begin the project. My mentor's ongoing advice and support also had a significant impact on enhancing my skills.

The working environment was quite encouraging, and it was impressive to see how some of the developers were able to independently complete their tasks and carry out experiments to advance their expertise. It was also exciting to be a part of a cooperative IT platform where many developers were actively contributing and assisting one another to meet the short- and long-term objectives of the business. Overall, this internship was the first step in developing my career. To elaborate, the project gave me a thorough understanding of the significance of server-side programming in the development of dynamic and interactive websites. The project also demonstrated the value of customization and adaptability in web development.

The project also offered insightful information about database implementation, particularly the data transfer between admin and client. I gained knowledge of how to effectively manage the data and make changes as needed by studying the code files and database structure. I also learnt how to ensure that the changes were

seamlessly reflected in the admin and client interfaces. My ability to solve problems and work cooperatively in a team has improved as a result of this experience, which will surely aid me in my future endeavors in web development.

5.2 Future Scope

Since its beginnings, NopCommerce has become a well-known e-commerce platform. To stay relevant, it is always being updated and given new features. NopCommerce has a bright future ahead of it, with plenty of room for expansion and development.

1. **Mobile Commerce:** NopCommerce can continue to improve its mobile commerce capabilities as more customers opt to shop online utilizing their mobile devices to offer a smooth shopping experience across all devices.
2. **Artificial Intelligence (AI) and Machine Learning (ML):** Businesses can personalize user experiences, enhance product recommendations, and enhance marketing efforts by integrating AI and ML into NopCommerce.
3. **Social Commerce:** E-commerce companies are relying more and more on social networking sites like Facebook, Instagram, and Twitter. NopCommerce can continue to develop its social commerce features, enabling companies to sell goods on social media.
4. **Augmented Reality (AR) and Virtual Reality (VR):** Customers can see products in a virtual setting by integrating AR and VR technology into NopCommerce, which can enhance their browsing experience and increase their likelihood of completing a purchase.
5. **Business needs:** NopCommerce modular architecture and robust administration interface make it simple and quick to make changes to the

website with only a few clicks. The admin, client, and database are then immediately updated to reflect these changes, making it simple for developers to maintain websites and keep them up to date in accordance with client needs.

REFERENCES

- [1]<https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>
- [2]<https://learn.microsoft.com/en-us/dotnet/csharp/>
- [3]https://en.wikipedia.org/wiki/Visual_Studio
- [4]https://en.wikipedia.org/wiki/Visual_Studio
- [5]<https://geeksarray.com/blog/aspnet-core-application-and-kestrel-web-server-settings>
- [6]<https://github.com/nopSolutions/nopCommerce>
- [7]<https://github.com/nopSolutions/nopCommerce>
- [8]<https://feyyazacet.medium.com/asp-net-core-mvc-request-life-cycle-ff2588a2af8f>
- [9]<https://www.interviewbit.com/blog/mvc-architecture/>
- [10]<https://medium.com/@ama.thanu/what-is-an-api-how-does-it-work-f4ea552d741f>
- [11]<https://www.semrush.com/blog/http-status-codes/>
- [12] <https://www.ibm.com/topics/api>
- [13] <https://towardsdatascience.com/what-is-an-api-and-how-does-it-work-1dccd7a8219e>
- [14] <https://dotnettutorials.net/lesson/routing-in-asp-net-core-web-api/>
- [15] <https://www.c-sharpcorner.com/article/razor-view-engine-in-asp-net-mvc-5/>
- [16] <https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core>
- [17] <https://www.c-sharpcorner.com/article/cors-in-dotnet-core/>
- [18]<https://www.abtosoftware.com/blog/why-use-asp-net-mvc-framework-for-web-application-development>
- [19]<https://learn.microsoft.com/enus/aspnet/core/mvc/views/layout?view=aspnetcore-7.0>

PLAGIARISM REPORT

ORIGINALITY REPORT

8%

SIMILARITY INDEX

4%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

dotnettutorials.net

Internet Source

1%

2

Submitted to National School of Business
Management NSBM, Sri Lanka

Student Paper

1%

3

John Ciliberti. "ASP.NET Core Recipes",
Springer Science and Business Media LLC,
2017

Publication

<1%

4

Adam Freeman. "Pro ASP.NET Core MVC 2",
Springer Science and Business Media LLC,
2017

Publication

<1%

5

Holger Schwichtenberg. "Modern Data Access
with Entity Framework Core", Springer Science
and Business Media LLC, 2018

Publication

<1%

6

C# 6.0 and the .NET 4.6 Framework, 2015.

Publication

<1%
