ASM2 1670 Chau Van Phuc GCC2001 23

Cung cấp điện (Trường Đại học Bách khoa - Đại học Quốc gia Thành phố Hồ Chí Minh)

ASSIGNMENT 2 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Business**		
Unit number and title	**Unit 30: Application Development**		
Submission date	8/3/2023	**Date Received 1st submission**	
Re-submission Date	10/3/2023	**Date Received 2nd submission**	
Student Name	Chau Van Phuc	**Student ID**	GCC200123
Class	GCC0902	**Assessor name**	Luong Hoang Huong
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	

Grading grid

P4	P5	P6	M3	M4	M5	D2	D3

❒ Summative Feedback:	❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:

Signature & Date:

Assignment Brief 2 (RQF)

Higher National Certificate/Diploma in Computing

Student Name/ID Number:	
Unit Number and Title:	**Unit 30: Application Development**
Academic Year:	**2021 – 2022**
Unit Assessor:	**Hoang Nhu Vinh**
Assignment Title:	**Application development with design diagrams and code**
Issue Date:	**01 April 2021**
Submission Date:	
Internal Verifier Name:	
Date:	

Submission Format:

Format:

- An individual report document in PDF

Submission

- Students are compulsory to submit the assignment in due date and in a way requested by the Tutor.
- The form of submission will be a soft copy posted on http://cms.greenwich.edu.vn/.
- Remember to convert the word file into PDF file before the submission on CMS.

Note:

- The individual Assignment *must* be your own work, and not copied by or from another student.
- If you use ideas, quotes or data (such as diagrams) from books, journals or other sources, you must reference your sources, using the Harvard style.
- Make sure that you understand and follow the guidelines to avoid plagiarism. Failure to comply this requirement will result in a failed assignment.

Unit Learning Outcomes:

LO3 Work individually and as part of a team to plan and produce a functional business application with support documentation

LO4 Evaluate the performance of a business application against its Software Design Document and initial requirements

Assignment Brief and Guidance:

Assignment scenario (continued from Assignment 1) Your team has finished the analysis and design for the system. Next task is development of the system.

Tasks:

After the presentation about your design (from Assignment 1), you need to create a formal questionnaire that effectively reviews your business application, problem definition statement, proposed solution and development strategy. This formal questionnaire should be answered by your colleagues. For any new insights, ideas or potential improvements to your system you need to evaluate and justify the reasons why you have chosen to include (or not to include) them as part of this business application. Based on the feedback of your colleagues, amend the design if needed.

Next task is to develop the business application based on the design, chosen technologies and methodology. When the application is fully built and tested, you need to review its performance against the Software Requirement Specificationn, analyze the factors that influence its performance and use them to undertake a critical review of the design, development and testing stages of your application. Conclude your review by reflectively discussing your previously identified risks. You should evaluate the strengths and weaknesses of your business application and fully justify opportunities for improvement and further development.

To conclude, your report document should include:

- Peer review section (questionnaire and answers, your reflection on the feedback)
- Development section (how you develop and test the application, what is the result)
- Review section (review, analyse and critical evaluate your application)

Your team needs to prepare a demo based on this report for the final demonstration.
The working application must also be demonstrated.

Learning Outcomes and Assessment Criteria (Assignment 2):			
Learning Outcome	Pass	Merit	Distinction
LO3	**P4** Create a formal questionnaire that effectively reviews your business application, problem definition statement, proposed solution and development strategy. Use this questionnaire as part of a peer-review and document any feedback given. **P5** Develop a functional business application based on a specified business problem.	**M3** Interpret your peer-review feedback and identify opportunities not previously considered. **M4** Develop a functional business application based on a specific Software Design Document with supportive evidence of using the preferred tools, techniques and methodologies.	**D2** Evaluate any new insights, ideas or potential improvements to your system and justify the reasons why you have chosen to include (or not to include) them as part of this business application.
LO4	**P6** Review the performance of your business application against the Problem Definition Statement and initial requirements.	**M5** Analyse the factors that influence the performance of a business application and use them to undertake a critical review of the design, development and testing stages of your application. Conclude your review by reflectively discussing your previously identified risks.	**D3** Critically evaluate the strengths and weaknesses of your business application and fully justify opportunities for improvement and further development.

Assignment 2

Assignment 2

I. Introduction

I will show the initial directory structure of the program in the application development section. Next, the next goal of this section will include explanations and source code examples. Third, I will take a screenshot of the user interface of the finished product. Then I'll demonstrate with screenshots how to source code using GitHub or GitLab. An image showing how to deploy the application using IIS or Azure will be shown in the final stage.

In the app review section will first give an assessment of the application's performance. Next, the application will be evaluated to see if it meets all the requirements or if further improvement is needed. I will also study the factors that affect the performance of the program. Finally, it evaluates the original directory structure of the program in the next application development. The next goal of this section will include explanations and source code examples. Third, I will take a screenshot of the user interface of the finished product. Then, I will demonstrate with screenshots how to manage the source code using GitHub or GitLab. An image showing how to deploy the application using IIS or Azure will be shown in the final stage.

The app review area will first give an assessment of the app's performance. Next, the application will be evaluated to see if it meets all the requirements or if further improvement is needed. I will also study the factors that affect the performance of the program. Then evaluate the advantages and disadvantages of the application.

II. Formal questionnaire to review the business application, problem definition statement, proposed solution, and development strategy

In this section, the jobs that users can use are based on their account roles. Here is a list of jobs sorted by role:

- Administrator role: Login, log out, register for Owner account, manage Owner and User accounts, reset Owner and User accounts, manage category browsing.
- Owner: Log in, log out, manage books(CRUD), request category, manage orders.
-Customers: Login, log out, register, account management, cart management, book preview, book lookup, see About page.
- Guest: book preview, book lookup, see About page, register.

Questionnaire about the FPT bookstore application:

No./Function	Question	Date	Answer	Date
1. Log-in	Whether or not users may log in using accounts from other websites, such as Facebook or Google. Because, as far as I'm aware, the system does not currently have such a function. Will that feature ever be upgraded on your system?	20/2/2023	Currently, we are unable to add such functionality to the system, but we will work to do so in the future	20/2/2023
	If your system has a feature to save account information, do you use user cookies?	20/2/2023	To make logging in the next time quicker, our system can save the user's account. However, that doesn't mean we'll get cookies from the user. Since we only save the account you have registered for and do not use your cookies, you may use our system with confidence	20/2/2023
2. Register	Although your system has a function for account authentication, in my opinion it is not very good. Will your system ever be updated to send authentication by email?	21/3/2023	To ensure the safety of the users of our system, we will work to implement such function in the near future.	21/3/2023

3. Edit information	Can the administrator make changes to the data that other users have submitted?	21/3/2023	Any information regarding other users may be changed by our administrators, including the deletion of their accounts. The user must give us permission to intervene; otherwise, we do not have the legal right to do so.	21/3/2023
4. Searching	Can I find a book by looking up the author or publisher as well as the book's description if I'm looking for one but don't know the title?	21/3/2023	It goes without saying that our system will utilize the information provided by the user to search the system for pertinent book goods	21/3/2023
5. Role	Are functions like customer accounts usable by accounts with higher permissions, and vice versa, are services like customer accounts usable by accounts with higher permissions?	22/3/2023	The decentralization of accounts is supported by our technology. Each account will also have access to more sophisticated features. The account of the administrator has the greatest permissions, followed by the account of the business owner and finally the account of the	22/3/2023

			client. Higherpermissioned accounts can utilize the features of lowerpermissioned accounts, while lowerpermissioned accounts cannot use the features of senior accounts.	

III. Application development

1. Entity Relationship Diagram (ERD)

The diagram below displays entity relationship of the FPTBook web-based application:

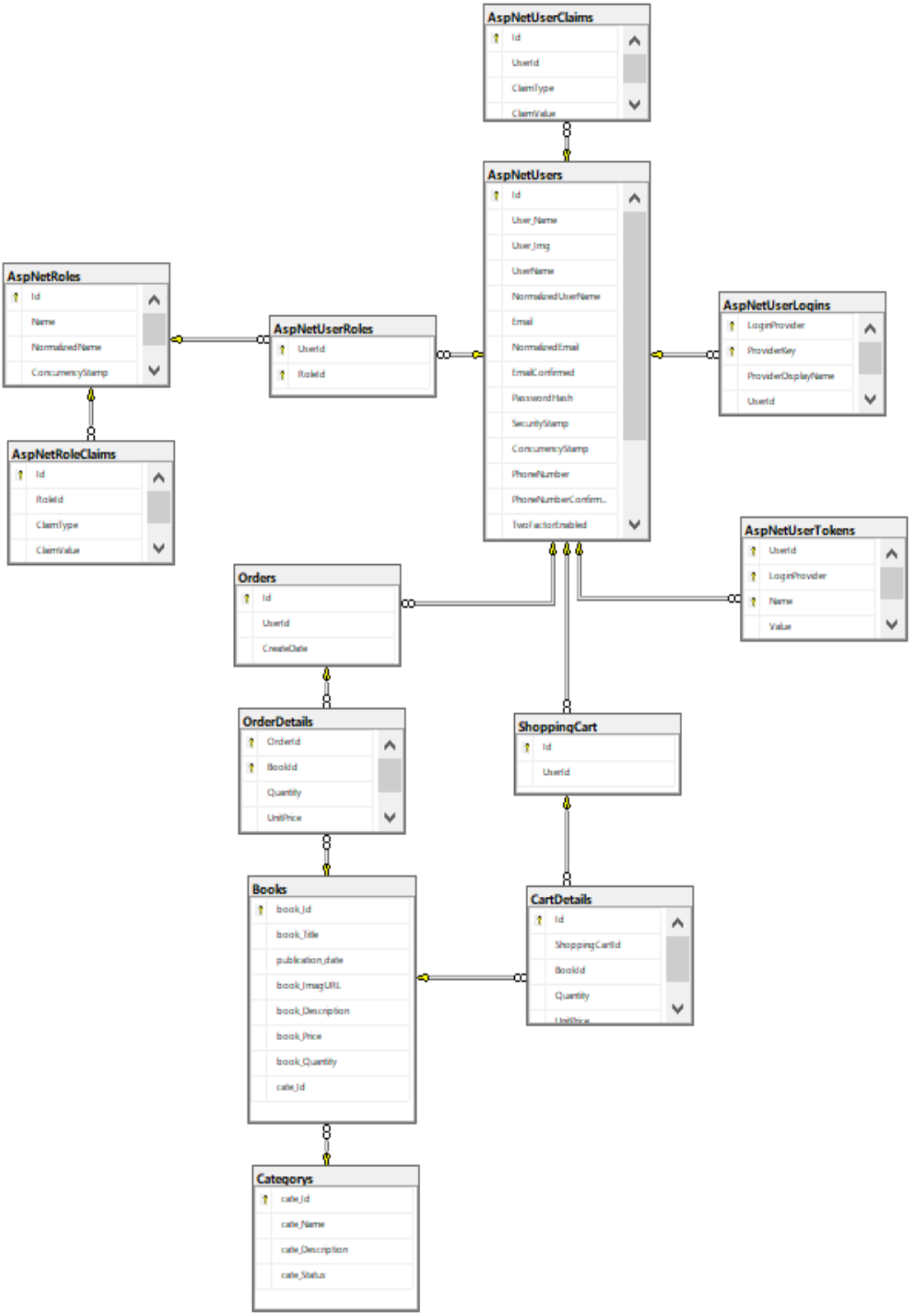


Figure 1 Entity Relationship Diagram (ERD)

2. Develop a functional business application

There are some tools, techniques and methodologies that I use to develop the FPT bookstore application:

❖ Develop tools

Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS. Evolution of Visual Studio: The first version of VS(Visual Studio) was released in 1997, named as Visual Studio 97 having version number 5.0. The latest version of Visual Studio is 15.0 which was released on March 7, 2017. It is also termed as Visual Studio 2017. The supported .Net Framework Versions in latest Visual Studio is 3.5 to 4.7. Java was supported in old versions of Visual Studio but in the latest version doesn't provide any support for Java language. (geeksforgeeks, 2023)

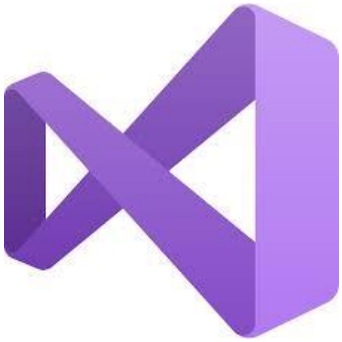


Figure 2 Visual Studio

Data is a collection of facts and figures and we have humungous data available to the users via the internet and other sources. To manipulate the data, Structured Query Language (SQL) in short has been introduced years ago. There are different versions of SQL available in the market provided by different organizations. In this article, we shall see the version of SQL provided by Microsoft.

- Microsoft SQL Server or MS SQL Server for short is the query language provided for data definition and manipulation.
- SQL Server is a Relational Database Management Systems which was developed and marketed by the Microsoft company.
- SQL and SQL servers are built as two layers where the SQL server is on the top for interacting with the relational databases.
- MS SQL Server also has T-SQL or Transact-SQL and the main focus of T-SQL is to handle the transactions.
- As it is a Microsoft's developed system, it worked only on Microsoft's environment until it was made available on Linux platforms in the year 2016.

Figure 3 Microsoft SQL Server

- **Technique**

The term "HTML" stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is an abbreviation of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text document within the tag which defines the structure of web pages. HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM). (geeksforgeeks, 2023)

Figure 4 HTML

CSS (Cascading Style Sheets)is used to apply styles to web pages. Cascading Style Sheets are fondly referred to as CSS. It is used to make web pages presentable. The reason for using this is to simplify the process of making web pages presentable. It allows you to apply styles on web pages. More importantly, it enables you to do this independently of the HTML that makes up each web page. (geeksforgeeks, 2023)

Figure 5 CSS

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. Nowadays, the websites are perfect for all browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers – Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project. (geeksforgeeks, 2023)

Figure 6 Bootstrap

ASP.NET Core is a free open source, a general-purpose development platform for developing modern cloud-based software applications on Windows, Linux, and macOS operating systems. It operates across several platforms and has been revamped to make .NET fast, scalable, and modern. .NET Core is one of Microsoft's big contributions and released under the MIT License. It offers the following features: (geeksforgeeks, 2023)

- Cross-Platform
- Open Source

- High Performance
- Multiple environments and development mode etc.

Figure 7 ASP.NET Core

- **Methodologies**

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs). Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps. (geeksforgeeks, 2023)

MVC was created by Trygve Reenskaug. The main goal of this design pattern was to solve the problem of users controlling a large and complex data set by splitting a large application into specific sections that all have their own purpose.

Features of MVC :
It provides a clear separation of business logic, Ul logic, and input logic.
It offers full control over your HTML and URLs which makes it easy to design web application architecture.
It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs.
It supports Test Driven Development (TDD).
Components of MVC :

The MVC framework includes the following 3 components:

- Controller
  The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn't

have to worry about handling data logic, it just tells the model what to do. It process all the business logic and incoming requests, manipulate data using the Model component and interact with the View to render the final output.

- Model

  The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller's request because the controller can't interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

- View

  The View component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller. It only interacts with the controller.

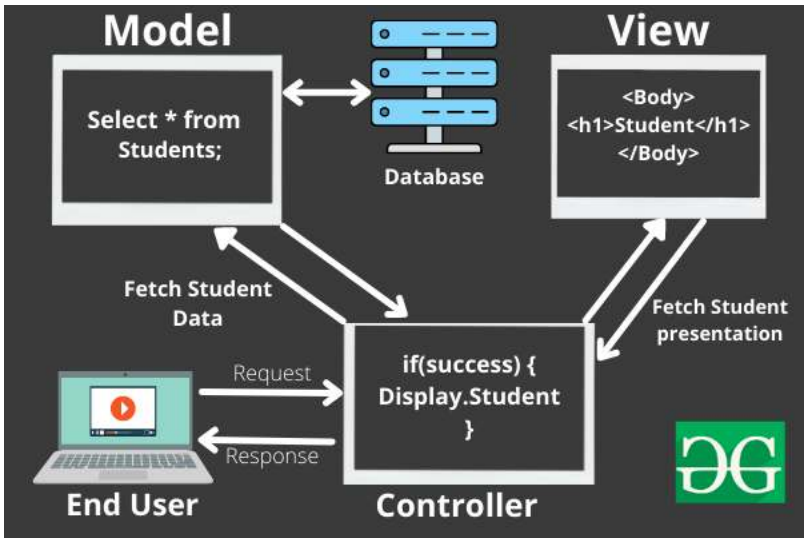


Figure 8Model-View-Controller (MVC) framework

3. Folder structure of the application

Our website FPT Book project is built on the ASP.NET Core MVC structure paradigm, which is comprised of Models, Views, and Controllers:

Overall structure of the project: This folder contains the whole code source for the FPTBook project.

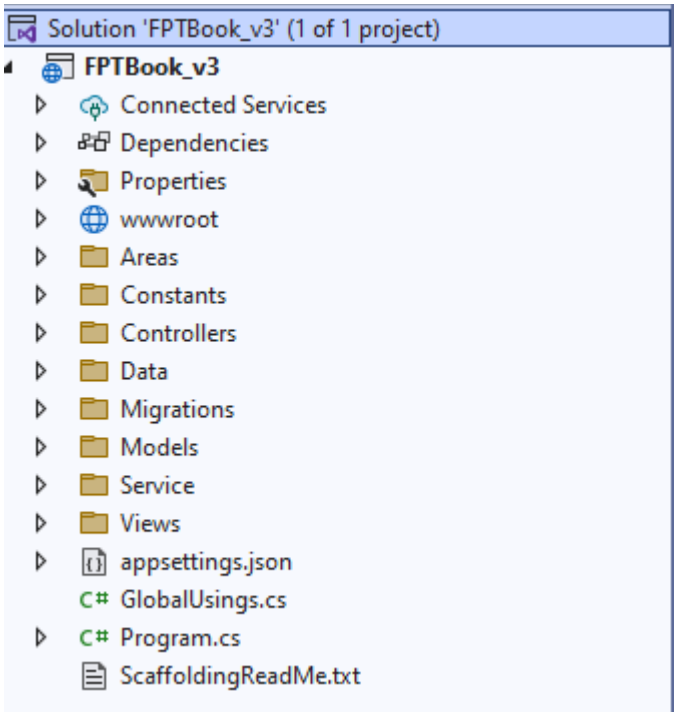


Figure 9 General FPTBook project

As seen in the figure below, application data files are saved in the Areas/Identity folder: The database's Data folder will house all of the account's data. The output for the registration page screen, navigation, and account authentication are all included in this Pages folder.

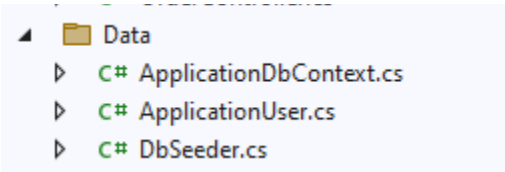


Figure 10 Data folder

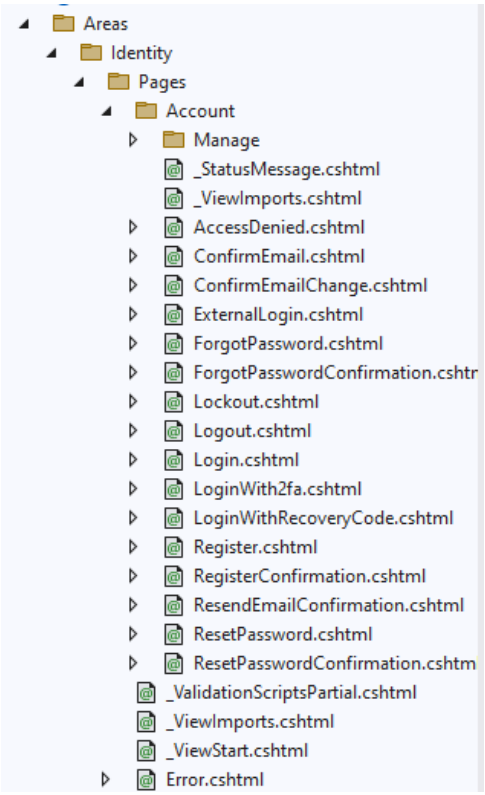


Figure 11 Areas/Identity folder

The Controllers folder contains directories that describe operations like adding, modifying, removing, and reading particular model information:

- Controllers folder: This folder is used to store the controllers of use cases such as Admin, Book, Cart, Home, Order, Category

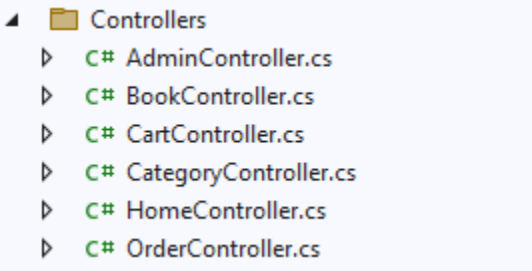


Figure 12 Controllers folder

- Constants folders: This directory is used to declare model of Constants

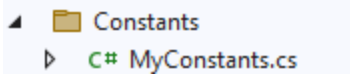


Figure 13 Constants folders

Models folders: This folder is used to declare the Category, Book, BookDisplayModel, Order, CartDetail, ShoppingCart, OrderDetail,ErrorViewModel, Owner.

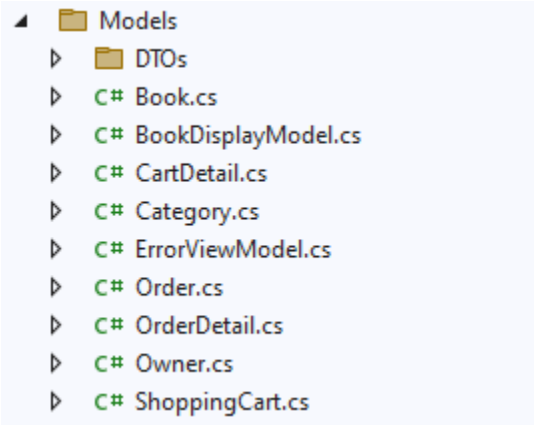


Figure 14 Models folders

Views folders: This folder is used to save the interfaces of use cases such as Admin, Book, Cart, Category, Home, Order, Shared

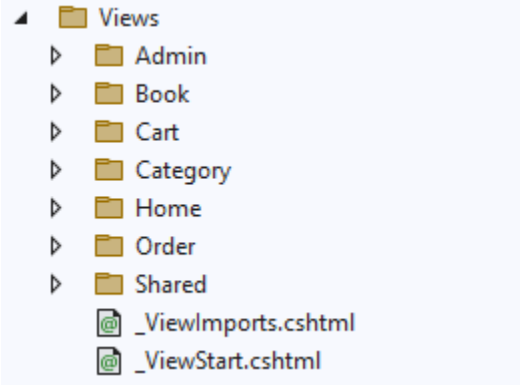


Figure 15 Views folders

In the Views folder, it will contain folders to clearly show the interfaces of each specific model:

- Admin folders: The interfaces of the Admin as: Index, RegisterOwner, RequestCategory, ShowOwner, ShowUser.

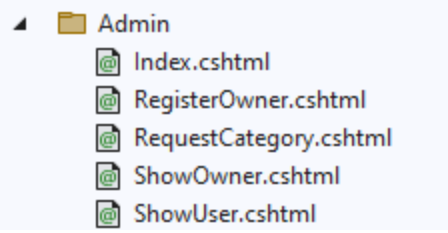


Figure 16 Admin folders

- Book folders: The interfaces of the Book as: Create, Edit, Delete

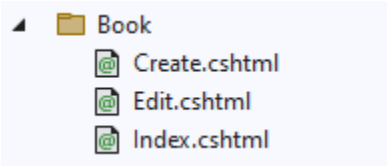


Figure 17 Book folders

- Cart folders: The interfaces of the Cart as : GetUserCart

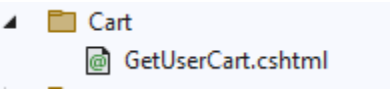


Figure 18 Cart folders

- Category folders: The interfaces of the Category as: Create, Edit, Delete

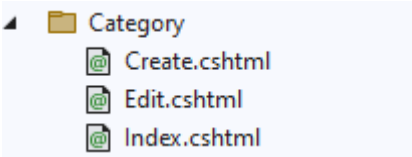


Figure 19 Category folders

- Home folders: The interfaces of the Home as : BookDetail, Index, ShowBook

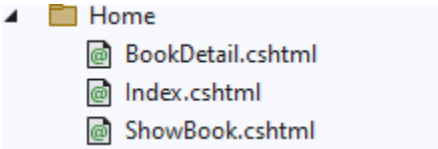


Figure 20 Home folders

- Order folders: The interfaces of the Order as : GetOrder, OrderDetail

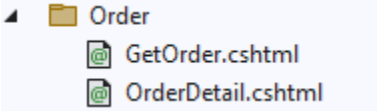


Figure 21 Order folders

- Shared folder: In this folder, it will save the navigation interface after login as well as the partial login page of the FPTBook project

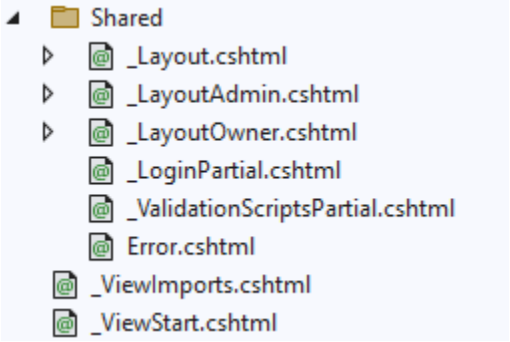


Figure 22 Shared folder

wwwroot folder: This is the folder to store the source css, fonts, js, libraries as well as images (in uploads folder) to serve the website.

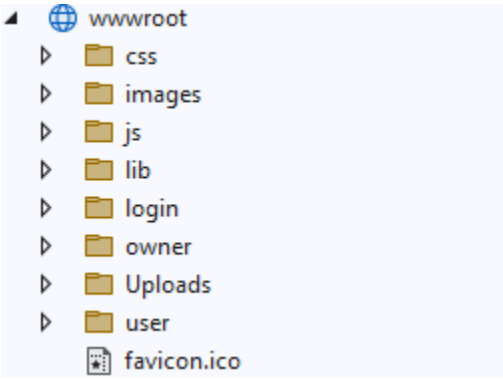


Figure 23 wwwroot folder

4. Code source samples of the application with an explanation

To get the website up and running using the ASP.NET Core framework, we used the "dotnet new mvc -o FPTBook_v3" command to create a project called FPTBook_v3. The "dotnet dev-certs https –trust" declaration is then used to trust the HTTPS development certificate.

Then we create the necessary models for the project like Category, Book, BookDisplayModel, Order, CartDetail, ShoppingCart, OrderDetail,ErrorViewModel, Owner.

❖ Model

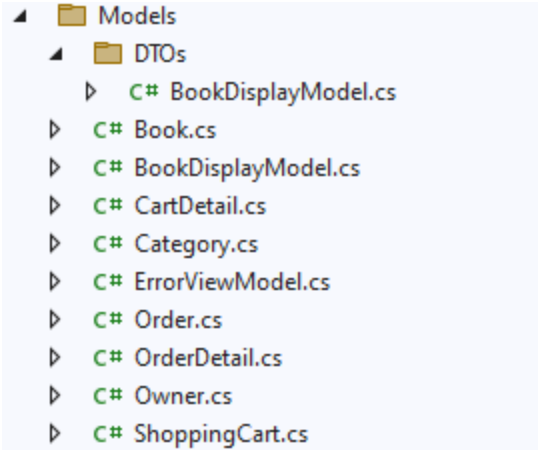


Figure 24 Models folder:

The table blow shows code source of files in Models folder:

File	Code source

Book	`namespace FPTBook_v3.Models`
	`{`
	`  public class Book`
	`  {`
	`  [Key]`
	`  [DatabaseGenerated(DatabaseGeneratedOption.Identity)]`
	`  public int book_Id { get; set; }`
	`  [Required]`
	`  public string book_Title { get; set; }`
	`  [Required]`
	`  public DateTime publication_date { get; set; }`
	`  public string? book_ImagURL { get; set; }`
	`  [Required]`
	`  public string book_Description { get; set; }`
	`  [Required]`
	`  public double book_Price { get; set; }`
	`  [Required]`
	`  public int book_Quantity { get; set; }`
	`  public int cate_Id { get; set; }`
	
	`  [ForeignKey("cate_Id")]`
	`  public virtual Category? category { get; set; }`
	
	`  public virtual ICollection<OrderDetail>? OrderDetails { get; set; }`
	
	`  [NotMapped]`
	`  public IFormFile? book_Img { get; set; }`
	`  }`
	`}`

BookDisplayModel	`namespace FPTBook_v3.Models`
	`{`
	`  public class BookDisplayModel`
	`  {`

	`public IEnumerable<Book> Books { get; set; }`
	`public IEnumerable<Category> Categorys { get; set; }`
	`public string STerm { get; set; } = "";`
	`public int GenreId { get; set; } = 0;`
	
	`public static implicit operator BookDisplayModel(DTOs.BookDisplayModel v)`
	`{`
	`throw new NotImplementedException();`
	`}`
	`}`
	`}`

CartDetail	`namespace FPTBook_v3.Models`
	`{`
	`public class CartDetail`
	`{`
	`[Key]`
	`public int Id { get; set; }`
	`[Required]`
	`public int ShoppingCartId { get; set; }`
	`[Required]`
	`public int BookId { get; set; }`
	`[Required]`
	`public int Quantity { get; set; }`
	`[Required]`
	`public double UnitPrice { get; set; }`
	
	`[ForeignKey("BookId")]`
	`public virtual Book? Book { get; set; }`
	
	`[ForeignKey("ShoppingCartId")]`
	`public ShoppingCart ShoppingCart { get; set; }`
	`}`
	`}`

Category	`namespace FPTBook_v3.Models`
	`{`
	`public class Category`
	`{`
	`[Key]`
	`[DatabaseGenerated(DatabaseGeneratedOption.Identity)]`
	`[Required]`
	`public int cate_Id { get; set; }`
	`[Required]`
	`[Display(Name = "cate_Name")]`
	`public string cate_Name { get; set; }`
	`[Required]`
	`[Display(Name = "cate_Description")]`
	`public string cate_Description { get; set; }`
	`public string? cate_Status { get; set; }`
	
	`public virtual ICollection<Book>? Books { get; set; }`
	`}`
	`}`

ErrorViewModel	`namespace FPTBook_v3.Models`
	`{`
	`public class ErrorViewModel`

	`{`
	`public string? RequestId { get; set; }`
	
	`public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);`
	`}`
	`}`

Order	`namespace FPTBook_v3.Models`
	`{`
	`public class Order`
	`{`
	`[Key]`
	`public int Id { get; set; }`
	`public string UserId { get; set; }`
	`[Required, Display(Name = "Create Date")]`
	`public DateTime CreateDate { get; set; } = DateTime.UtcNow;`
	
	`public List<OrderDetail> OrderDetail { get; set; }`
	
	`[ForeignKey("UserId")]`
	`public virtual ApplicationUser? ApplicationUsers { get; set; }`
	`}`
	`}`

OrderDetail	`namespace FPTBook_v3.Models`
	`{`
	`public class OrderDetail`
	`{`
	
	`[Required]`
	`public int OrderId { get; set; }`
	`[Required]`
	`public int BookId { get; set; }`
	`[Required]`
	`public int Quantity { get; set; }`
	`[Required]`
	`public double UnitPrice { get; set; }`
	`public Order Order { get; set; }`
	`public Book Book { get; set; }`
	`}`
	`}`

Owner	`namespace FPTBook_v3.Models`
	`{`
	`public class Owner`
	`{`
	`[Required, Display(Name = "Name")]`
	`public string Name { get; set; }`
	
	`[Required,Display(Name = "Email")]`
	`public string Email { get; set; }`
	
	`[StringLength(10, ErrorMessage = "Phone number must have 10 digits",`
	`MinimumLength = 10)]`
	`public string Phone { get; set; }`
	
	`[Required, Display(Name = "Image")]`
	`public string? Image { get; set; }`
	
	`[StringLength(100, ErrorMessage = "The {0} must be at least {2} and`
	`at max {1} characters long.", MinimumLength = 6)]`

	`[DataType(DataType.Password)]`
	`[Display(Name = "Password")]`
	`public string Password { get; set; }`
	
	`[DataType(DataType.Password)]`
	`[Display(Name = "Confirm password")]`
	`[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]`
	`public string ConfirmPassword { get; set; }`
	
	`public IFormFile? Img { get; set; }`
	`}`
	`}`

ShoppingCart	`namespace FPTBook_v3.Models`
	`{`
	`public class ShoppingCart`
	`{`
	`[Key]`
	`[DatabaseGenerated(DatabaseGeneratedOption.Identity)]`
	`public int Id { get; set; }`
	`[Required]`
	`public string UserId { get; set; }`
	
	`public ICollection<CartDetail> CartDetails { get; set; }`
	
	`[ForeignKey("UserId")]`
	`public virtual ApplicationUser? ApplicationUsers { get; set; }`
	`}`
	`}`

❖ **Views**

After creating models, we create files in the Views folder.

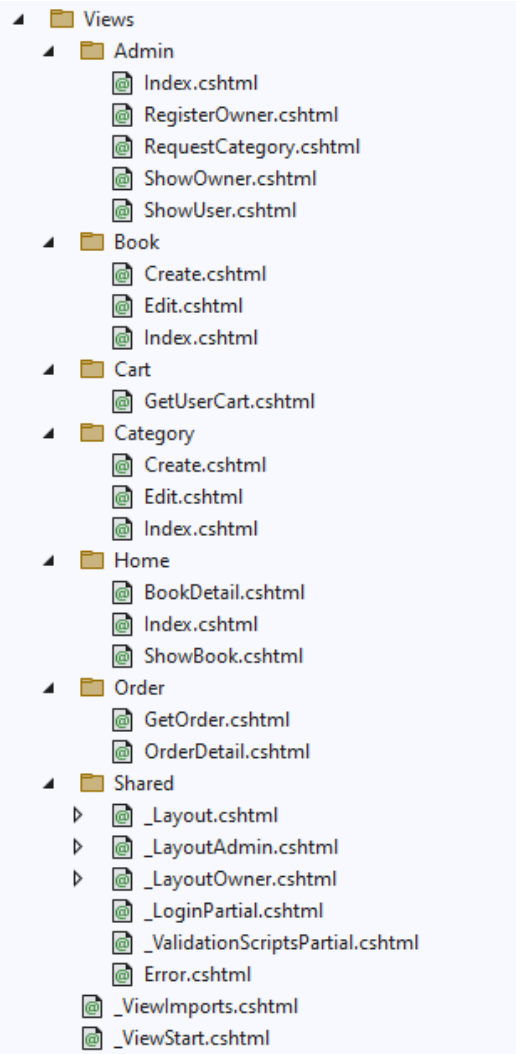


Figure 25 Views folder

❖ **Controllers**

Next, we develop file controllers to conduct the operations of each model, such as create, delete, edit, view, and so on. in the Controllers folder.

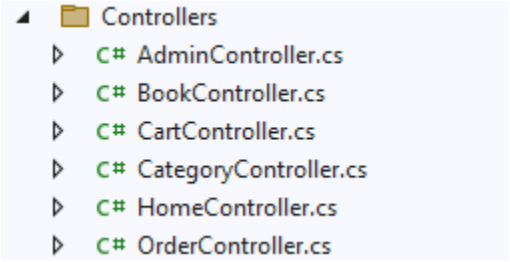


Figure 26 Controller folder

The table blow shows code source of files in Controllers folder:

File	Code source
AdminController	

```
namespace FPTBook_v3.Controllers
{
    [Authorize(Roles = "Admin")]
    public class AdminController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly ApplicationDbContext _db;
        private readonly IFileService _fileService;
        public AdminController(ApplicationDbContext db,
            UserManager<ApplicationUser> userManager,
            IFileService fileService)
        {
            _db = db;
            _userManager = userManager;
            _fileService = fileService;
        }
        public IActionResult Index()
        {
            return View();
        }

        [Route("Admin/ShowUser")]
        public async Task<IActionResult> ShowUser()
        {
            var user = await (from users in _db.Users
                              join UserRole in _db.UserRoles
                              on users.Id equals UserRole.UserId
                              join role in _db.Roles
                              on UserRole.RoleId equals role.Id
                              where role.Name == "User"
                              select users).ToListAsync();
            return View(user);
        }

        [Route("Admin/ShowOwner")]
        public async Task<IActionResult> ShowOwner()
        {
            var owner = await (from users in _db.Users
                              join UserRole in _db.UserRoles
                              on users.Id equals UserRole.UserId
                              join role in _db.Roles
                              on UserRole.RoleId equals role.Id
                              where role.Name == "Owner"
                              select users).ToListAsync();
            return View(owner);
        }


        [Route("Admin/RequestCategory")]
        public IActionResult RequestCategory()
        {
            var category = _db.Categorys.Where(c => c.cate_Status ==
"processing").ToList();
            return View(category);
        }

        [Route("Admin/RequestCategory/Approval")]
```

```
        public IActionResult Approval(int id)
        {
            Category category = _db.Categorys.Find(id);
            if (category == null)
            {
                return RedirectToAction("RequestCategory");
            }
            else
            {
                category.cate_Status = "processed";
                _db.Categorys.Update(category);
                _db.SaveChanges();
                return RedirectToAction("RequestCategory");
            }

        }


        [Route("Admin/RequestCategory/Reject")]
        public IActionResult Reject(int id)
        {
            Category category = _db.Categorys.Find(id);
            if (category == null)
            {
                return RedirectToAction("RequestCategory");
            }
            else
            {
                _db.Categorys.Remove(category);
                _db.SaveChanges();
                return RedirectToAction("RequestCategory");
            }
        }

        [Route("Admin/RegisterOwner")]
        public async Task<IActionResult> RegisterOwner()
        {
            return View();
        }

        [HttpPost]
        [Route("Admin/RegisterOwner")]
        public async Task<IActionResult> RegisterOwner(Owner owners)
        {
            if (ModelState.IsValid)
            {
                var owner = new ApplicationUser
                {
                    UserName = owners.Email,
                    User_Name = owners.Name,
                    Email = owners.Email,
                    PhoneNumber = owners.Phone,


                };

                if (owners.Img != null)
                {
```

```
                    var resultt = _fileService.SaveImage(owners.Img);
                    if (resultt.Item1 == 1)
                    {
                        var oldImage = owner.User_Img;
                        owner.User_Img = resultt.Item2;


                        if (oldImage == null)
                        {

                        }
                        else
                        {
                            var                delete                =
_fileService.Delete(oldImage);
                        }

                    }
                }

                var  result  =  await  _userManager.CreateAsync(owner,
owners.Password);
                if (result.Succeeded)
                {
                    await          _userManager.AddToRoleAsync(owner,
Role.Owner.ToString());
                    return RedirectToAction("ShowOwner");
                }
                else
                {
                    TempData["Fail"] = "RegisterOwner Fail!";
                    return RedirectToAction("RegisterOwner");
                }
            }
            return RedirectToAction("RegisterOwner");
        }
    }
}
```

BookController

```
namespace FPTBook_v3.Controllers
{

    [Authorize(Roles = "Owner")]
    public class BookController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly ILogger<BookController> _logger;

        public     BookController(ILogger<BookController>     logger,
ApplicationDbContext db)
        {
            _logger = logger;
            _db = db;
        }

        [Route("/Owner/Book")]
        public async Task<IActionResult>  Index()
        {
            IEnumerable<Book> books = await GetBooks();
```

```
            IEnumerable<Category>       categorys       =       await
_db.Categorys.ToListAsync(); ;
            Models.BookDisplayModel       bookModel       =       new
Models.BookDisplayModel
            {
                Books = books,
                Categorys = categorys
            };
            return View(bookModel);
        }


        [Route("/Owner/Book/Create")]
        public IActionResult Create()
        {
            ViewData["Cate_Id"] = new SelectList(_db.Categorys.Where(c
=> c.cate_Status == "processed").ToList(), "cate_Id", "cate_Name");
            return View();
        }

        [HttpPost]
        [Route("/Owner/Book/Create")]
        public IActionResult Create(Book book)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    string uniqueFileName = UploadFile(book);
                    book.book_ImagURL = uniqueFileName;
                    if (book.book_Quantity < 0 || book.book_Price < 0)
                    {
                        TempData["Fail"] = "Quantity and Price must be
greater than 0";
                        ViewData["Cate_Id"]           =           new
SelectList(_db.Categorys.Where(c       =>       c.cate_Status       ==
"processed").ToList(), "cate_Id", "cate_Name");
                        return View(book);
                    }
                    _db.Books.Add(book);
                    _db.SaveChanges();
                    return RedirectToAction("Index");
                }
                catch (Exception)
                {
                    ViewData["Cate_Id"]             =             new
SelectList(_db.Categorys.Where(c       =>       c.cate_Status       ==
"processed").ToList(), "cate_Id", "cate_Name");
                    TempData["Error"] = "";
                    return View(book);
                }

            }
            return View(book);
        }


        [Route("/Owner/Book/Edit/{id:}")]
```

```
        public IActionResult Edit(int id)
        {
            Book book = _db.Books.Find(id);
            if (book == null)
            {
                return RedirectToAction("Index");
            }
            ViewData["Cate_Id"]   =   new   SelectList(_db.Categorys,
"cate_Id", "cate_Name");
            return View(book);
        }

        [HttpPost]
        [Route("/Owner/Book/Edit/{id:}")]
        public IActionResult Edit(int id, Book book,string img)
        {
            book.book_Id = id;
            if (ModelState.IsValid)
            {
                try
                {
                    if (book.book_Img == null)
                    {
                        book.book_ImagURL = img;
                        if (book.book_Quantity < 0 || book.book_Price
< 0)
                        {
                            TempData["Fail"]  =  "Quantity  and  Price
must be greater than 0";
                            ViewData["Cate_Id"]         =         new
SelectList(_db.Categorys.Where(c       =>       c.cate_Status       ==
"processed").ToList(), "cate_Id", "cate_Name");
                            return View(book);
                        }
                        _db.Books.Update(book);
                        _db.SaveChanges();
                    }
                    else
                    {
                        string uniqueFileName = UploadFile(book);
                        book.book_ImagURL = uniqueFileName;

                        if (book.book_Quantity < 0 || book.book_Price
< 0)
                        {
                            TempData["Fail"]  =  "Quantity  and  Price
must be greater than 0";
                            ViewData["Cate_Id"]         =         new
SelectList(_db.Categorys.Where(c       =>       c.cate_Status       ==
"processed").ToList(), "cate_Id", "cate_Name");
                            return View(book);
                        }
                        _db.Books.Update(book);
                        _db.SaveChanges();

                        img = Path.Combine("wwwroot", "uploads", img);
                        FileInfo infor = new FileInfo(img);
                        if (infor != null)
```

```
                        {
                            System.IO.File.Delete(img);
                            infor.Delete();
                        }
                    }

                    return RedirectToAction("Index");
                }
                catch (Exception)
                {
                    TempData["Error"] = "";
                    ViewData["Cate_Id"]             =             new
SelectList(_db.Categorys.Where(c       =>       c.cate_Status       ==
"processed").ToList(), "cate_Id", "cate_Name");
                    return View(book);
                }

            }

            return View(book);
        }




        [Route("/Owner/Book/Delete/{id:}")]
        public ActionResult Delete(int id, string img)
        {
            Book book = _db.Books.Find(id);
            if (book == null)
            {
                return RedirectToAction("Index");
            }
            else
            {
                img = Path.Combine("wwwroot", "uploads", img);
                FileInfo infor = new FileInfo(img);
                if (infor != null)
                {
                    System.IO.File.Delete(img);
                    infor.Delete();
                }
                _db.Books.Remove(book);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
        }

        public string UploadFile(Book book)
        {
            string uniqueFileName = null;

            if (book.book_Img != null)
            {
                var ext = Path.GetExtension(book.book_Img.FileName);
                var allowedExtensions = new string[] { ".jpg", ".png",
".jpeg" };
                if (!allowedExtensions.Contains(ext))
```

```
                {
                    string msg = string.Format("Only {0} extensions
are allowed", string.Join(",", allowedExtensions));
                    throw new Exception(msg);
                }

                string uploadsFoder = Path.Combine("wwwroot",
"uploads");
                uniqueFileName = Guid.NewGuid().ToString() +
book.book_Img.FileName;
                string filePath = Path.Combine(uploadsFoder,
uniqueFileName);
                using (var fileStream = new FileStream(filePath,
FileMode.Create))
                {
                    book.book_Img.CopyTo(fileStream);
                }
            }
            return uniqueFileName;
        }



public async Task<IEnumerable<Book>> GetBooks()
{
IEnumerable<Book> books = await (from book in _db.Books
join genre in _db.Categorys
on book.cate_Id equals genre.cate_Id
select new Book
{
book_Id = book.book_Id,
book_ImagURL = book.book_ImagURL,
category = book.category,
book_Title = book.book_Title,
cate_Id = book.cate_Id,
book_Price = book.book_Price,
book_Quantity = book.book_Quantity,
publication_date = book.publication_date,
book_Description = book.book_Description
}
).ToListAsync();
return books;

        }


    }
}
```

CartController

```
namespace FPTBook_v3.Controllers
{

    [Authorize(Roles = "User")]
    public class CartController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly IHttpContextAccessor _httpContextAccessor;
```

```
        public        CartController(ApplicationDbContext        db,
IHttpContextAccessor httpContextAccessor,
            UserManager<ApplicationUser> userManager)
        {
            _db = db;
            _userManager = userManager;
            _httpContextAccessor = httpContextAccessor;
        }

        [Route("User/Cart/AddItem")]
        public async Task<IActionResult> AddItem(int bookId, int qty =
1, int redirect = 0)
        {
            var cartCount = await AddItemCart(bookId, qty);

            if (redirect == 0)
                return Ok(cartCount);
            return RedirectToAction("GetUserCart");
        }

        [Route("User/Cart/RemoveItem")]
        public async Task<IActionResult> RemoveItem(int bookId)
        {
            var cartCount = await RemoveCartItem(bookId);
            return RedirectToAction("GetUserCart");
        }

        [Route("User/Cart/GetUserCart")]
        public async Task<IActionResult> GetUserCart()
        {
            var cart = await GetCartItem();
            return View(cart);
        }

        [Route("User/Cart/GetTotalItemInCart")]
        public async Task<IActionResult> GetTotalItemInCart()
        {
            int cartItem = await GetCartItemCount();
            return Ok(cartItem);
        }

        [Route("User/Cart/Checkout")]
        public async Task<IActionResult> Checkout()
        {
            var isCheckedOut = await DoCheckout();
            if (!isCheckedOut)
            {
                TempData["Quantity"] = "The number of products is not
enough!";
                return Redirect("~/User/Cart/GetUserCart");
            }
            else
            {
                TempData["Success"] = "Order Success";
                return Redirect("~/User/Cart/GetUserCart");
            }
```

```
        }


        public async Task<int> AddItemCart(int bookId, int qty)
        {
            string userId = GetUserId();
            try
            {
                if (string.IsNullOrEmpty(userId))
                    throw new Exception("user is not logged-in");
                var cart = await GetCart(userId);
                if (cart is null)
                {
                    cart = new ShoppingCart
                    {
                        UserId = userId
                    };
                    _db.ShoppingCarts.Add(cart);
                }
                _db.SaveChanges();

                var cartItem = _db.CartDetails
                                .FirstOrDefault(a                    =>
a.ShoppingCartId == cart.Id && a.BookId == bookId);
                if (cartItem is not null)
                {
                        _db.SaveChanges();
                        cartItem.Quantity += qty;
                }
                else
                {

                        var book = _db.Books.Find(bookId);
                        cartItem = new CartDetail
                        {
                            BookId = bookId,
                            ShoppingCartId = cart.Id,
                            Quantity = qty,
                            UnitPrice = book.book_Price   // it is a
new line after update


                        };
                        _db.CartDetails.Add(cartItem);

                }
                _db.SaveChanges();
            }
            catch (Exception ex)
            {
            }
            var cartItemCount = await GetCartItemCount(userId);
            return cartItemCount;
        }


        public async Task<int> RemoveCartItem(int bookId)
        {
```

```
            //using var transaction = _db.Database.BeginTransaction();
            string userId = GetUserId();
            try
            {
                if (string.IsNullOrEmpty(userId))
                    throw new Exception("user is not logged-in");
                var cart = await GetCart(userId);
                if (cart is null)
                    throw new Exception("Invalid cart");
                // cart detail section
                var cartItem = _db.CartDetails
                                .FirstOrDefault(a                    =>
a.ShoppingCartId == cart.Id && a.BookId == bookId);
                if (cartItem is null)
                    throw new Exception("Not items in cart");
                else if (cartItem.Quantity == 1)
                    _db.CartDetails.Remove(cartItem);
                else
                    cartItem.Quantity = cartItem.Quantity - 1;
                _db.SaveChanges();
            }
            catch (Exception ex)
            {

            }
            var cartItemCount = await GetCartItemCount(userId);
            return cartItemCount;
        }

        public async Task<ShoppingCart> GetCartItem()
        {
            var userId = GetUserId();
            if (userId == null)
                throw new Exception("Invalid userid");
            var shoppingCart = await _db.ShoppingCarts
                                .Include(a => a.CartDetails)
                                .ThenInclude(a => a.Book)
                                .ThenInclude(a => a.category)
                                .Where(a     =>     a.UserId     ==
userId).FirstOrDefaultAsync();
            return shoppingCart;

        }
        public async Task<ShoppingCart> GetCart(string userId)
        {
            var  cart  =  await  _db.ShoppingCarts.FirstOrDefaultAsync(x
=> x.UserId == userId);
            return cart;
        }

        public async Task<int> GetCartItemCount(string userId = "")
        {
            if (!string.IsNullOrEmpty(userId))
            {
                userId = GetUserId();
            }
            var data = await (from cart in _db.ShoppingCarts
                              join cartDetail in _db.CartDetails
```

```
                                on            cart.Id            equals
cartDetail.ShoppingCartId
                                select new { cartDetail.Id }
                            ).ToListAsync();
            return data.Count;
        }

        public async Task<bool> DoCheckout()
        {
            try
            {
                // logic
                // move data from cartDetail to order and order detail
then we will remove cart detail
                var userId = GetUserId();
                if (string.IsNullOrEmpty(userId))
                    throw new Exception("User is not logged-in");
                var cart = await GetCart(userId);
                if (cart is null)
                    throw new Exception("Invalid cart");
                var cartDetail = _db.CartDetails
                                        .Where(a  =>  a.ShoppingCartId  ==
cart.Id).ToList();
                if (cartDetail.Count == 0)
                    throw new Exception("Cart is empty");
                var order = new Order
                {
                    UserId = userId,
                    CreateDate = DateTime.UtcNow,
                };
                _db.Orders.Add(order);
                _db.SaveChanges();
                foreach (var item in cartDetail)
                {

                    var orderDetail = new OrderDetail
                    {
                        BookId = item.BookId,
                        OrderId = order.Id,
                        Quantity = item.Quantity,
                        UnitPrice = item.UnitPrice
                    };
                    _db.OrderDetails.Add(orderDetail);


                    var  quantity  =  _db.Books.FirstOrDefault(a  =>
a.book_Id == item.BookId);

                        if (quantity.book_Quantity < item.Quantity)
                        {

                            return false;
                        }
                        else
                        {
                            quantity.book_Quantity                  =
quantity.book_Quantity - item.Quantity;
                            _db.Update(quantity);
```

	`_db.SaveChanges();`
	
	`}`
	
	`}`
	`_db.CartDetails.RemoveRange(cartDetail);`
	`_db.SaveChanges();`
	
	`return true;`
	`}`
	`catch (Exception)`
	`{`
	
	`return false;`
	`}`
	`}`
	
	`private string GetUserId()`
	`{`
	`var principal = _httpContextAccessor.HttpContext.User;`
	`string userId = _userManager.GetUserId(principal);`
	`return userId;`
	`}`
	`}`
	`}`

CategoryController	`namespace FPTBook_v3.Models`
	`{`
	
	`[Authorize(Roles = "Owner")]`
	
	`public class CategoryController : Controller`
	`{`
	`private readonly ApplicationDbContext _db;`
	
	`public CategoryController(ApplicationDbContext db)`
	`{`
	`_db = db;`
	`}`
	
	`[Route("/Owner/Category")]`
	`public IActionResult Index()`
	`{`
	`IEnumerable<Category> ds = _db.Categorys.Where(c => c.cate_Status == "processed").ToList();`
	
	`return View(ds);`
	`}`
	
	`[Route("/Owner/Category/Create")]`
	`public IActionResult Create()`
	`{`
	`return View();`
	`}`
	
	`[HttpPost]`
	`[Route("/Owner/Category/Create")]`

```
        public IActionResult Create(Category category)
        {

            if (ModelState.IsValid)
            {
                category.cate_Status = "Processing";
                _db.Categorys.Add(category);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(category);
        }

        [Route("/Owner/Category/Edit/{id:}")]
        public IActionResult Edit(int id)
        {
            Category category = _db.Categorys.Find(id);
            if (category == null)
            {
                return RedirectToAction("Index");
            }
            return View(category);
        }

        [HttpPost]
        [Route("/Owner/Category/Edit/{id:}")]
        public IActionResult Edit(int id, Category category)
        {
            if (ModelState.IsValid)
            {
                category.cate_Id = id;
                category.cate_Status = "processed";
                _db.Categorys.Update(category);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(category);
        }


        [Route("/Owner/Category/Delete/{id:}")]
        public ActionResult Delete(int id)
        {
            Category category = _db.Categorys.Find(id);
            if (category == null)
            {
                return RedirectToAction("Index");
            }
            else
            {
                _db.Categorys.Remove(category);
                _db.SaveChanges();
                return RedirectToAction("Index");
            }
        }
    }
}
```

HomeController

```
namespace FPTBook_v3.Controllers
```

```
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly ApplicationDbContext _db;

        public     HomeController(ILogger<HomeController>     logger,
ApplicationDbContext db)
        {
            _db = db;
            _logger = logger;
        }

        [Route("Home/ShowBook")]
        public async Task<IActionResult> ShowBook(string sterm = "",
int genreId = 0)
        {

            IEnumerable<Book> books = await GetBooks(sterm, genreId);
            IEnumerable<Category>       categorys       =       await
_db.Categorys.Where(x => x.cate_Status == "processed").ToListAsync();
;
            Models.BookDisplayModel       bookModel       =       new
Models.BookDisplayModel
            {
                Books = books,
                Categorys = categorys,
            };
            return View(bookModel);
        }


        [Route("/Book/Detail/{id:}")]
        public async Task<IActionResult> BookDetail(int id)
        {
            if (id == null || _db.Books == null)
            {
                return NotFound();
            }
            else
            {
                var      book      =      _db.Books.Include(x      =>
x.category).FirstOrDefault(b => b.book_Id == id);
                if (book == null)
                {
                    return NotFound();
                }
                else
                {
                    return View(book);
                }
            }

        }

        public async Task<IActionResult> Index(string sterm = "", int
genreId = 0)
        {
```

```
            IEnumerable<Book>   books   =   await   IndexGetBook(sterm,
genreId);
            IEnumerable<Category>       categorys       =       await
_db.Categorys.Where(x => x.cate_Status == "processed").ToListAsync();
;
            Models.BookDisplayModel       bookModel       =       new
Models.BookDisplayModel
            {
                Books = books,
                Categorys = categorys,
            };
            return View(bookModel);
        }

        [ResponseCache(Duration      =      0,      Location      =
ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return    View(new    ErrorViewModel    {    RequestId    =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }

        public async Task<IEnumerable<Category>> Category()
        {
            return await _db.Categorys.ToListAsync();
        }


        public async Task<IEnumerable<Book>> IndexGetBook(string sTerm
= "", int genreId = 0)
        {

            IEnumerable<Book> books = await (from book in _db.Books
                                                 join       genre       in
_db.Categorys
                                                 on   book.cate_Id   equals
genre.cate_Id
                                                 where
string.IsNullOrWhiteSpace(sTerm)     ||     (book     !=     null     &&
book.book_Title.ToLower().StartsWith(sTerm))
                                                 select
book).ToListAsync();

            if (genreId != 0 && sTerm != null)
            {

                books = await (from book in _db.Books
                               join genre in _db.Categorys
                               on book.cate_Id equals genre.cate_Id
                               where   genre.cate_Id   ==   genreId   &&
book.book_Title == sTerm
                               select book).ToListAsync();
                /*books    =    books.Where(a    =>    a.book_Id    ==
genreId).ToList();*/
            }
            else if (genreId != 0 && sTerm == null)
            {
                books = await (from book in _db.Books
```

```
                                join genre in _db.Categorys
                                on book.cate_Id equals genre.cate_Id
                                where genre.cate_Id == genreId
                                select book).ToListAsync();
            }
            return books;

        }


        public async Task<IEnumerable<Book>> GetBooks(string sTerm =
"", int genreId = 0)
        {

            IEnumerable<Book> books = await (from book in _db.Books
                                             join genre in
_db.Categorys
                                             on book.cate_Id equals
genre.cate_Id
                                             where
string.IsNullOrWhiteSpace(sTerm) || (book != null &&
book.book_Title.ToLower().StartsWith(sTerm))
                                             select
book).ToListAsync();

            if (genreId != 0 && sTerm != null)
            {

                books = await (from book in _db.Books
                                join genre in _db.Categorys
                                on book.cate_Id equals genre.cate_Id
                                where genre.cate_Id == genreId &&
book.book_Title == sTerm
                                select book).ToListAsync();
                /*books = books.Where(a => a.book_Id ==
genreId).ToList();*/
            }
            else if (genreId != 0 && sTerm == null)
            {
                books = await (from book in _db.Books
                                join genre in _db.Categorys
                                on book.cate_Id equals genre.cate_Id
                                where genre.cate_Id == genreId
                                select book).ToListAsync();
            }
            return books;

        }


    }
}
```

OrderController

```
namespace FPTBook_v3.Controllers
{
    public class OrderController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly IHttpContextAccessor _httpContextAccessor;
```

```
        private readonly UserManager<ApplicationUser> _userManager;


        public OrderController(ApplicationDbContext db,
            UserManager<ApplicationUser> userManager,
             IHttpContextAccessor httpContextAccessor)
        {
            _db = db;
            _httpContextAccessor = httpContextAccessor;
            _userManager = userManager;
        }

        [Authorize(Roles = "User")]
        [Route("/User/UserOrders")]
        public async Task<IEnumerable<Order>> UserOrders()
        {
            var userId = GetUserId();
            if (string.IsNullOrEmpty(userId))
                throw new Exception("User is not logged-in");
            var orders = await _db.Orders
                            .Include(x => x.OrderDetail)
                            .ThenInclude(x => x.Book)
                            .ThenInclude(x => x.category)
                            .Where(a => a.UserId == userId)
                            .ToListAsync();
            return orders;
        }

        private string GetUserId()
        {
            var principal = _httpContextAccessor.HttpContext.User;
            string userId = _userManager.GetUserId(principal);
            return userId;
        }

        [Authorize(Roles = "User")]
        [Route("/User/UserOrders/OrderDetail")]
        public async Task<IActionResult> OrderDetail()
        {
            var orders = await UserOrders();
            return View(orders);
        }

        [Authorize(Roles = "Owner")]
        [Route("Owner/GetOrder")]
        public async Task<IActionResult> GetOrder()
        {
            var orders = await _db.Orders
                            .Include(x => x.ApplicationUsers)
                            .Include(x => x.OrderDetail)
                            .ThenInclude(x => x.Book)
                            .ThenInclude(x => x.category)

                            .ToListAsync();
            return View(orders);
        }
    }
}
```

❖ Connect database

The next step is to connect the data to the database; to do this, we connect to SQL Server using the following statement in the appsettings.json file:

```
"ConnectionStrings": {
  "DefaultConnection": "Server = WIN-58SOEU8QDHO\\SQLEXPRESS;Database=FPTBook_v3;uid=sa;pwd=Phuc@09012002;TrustServerCertificate=true;"
},
```

Figure 27 appsettings.json file

To connect to the database, we used the following .NET CLI instructions to add NuGet packages: “dotnet tool uninstall --global dotnet-aspnet-codegenerator dotnet tool install --global dotnet-aspnetcodegenerator dotnet tool uninstall --global dotnet-ef dotnet tool install --global dotnet-ef dotnet add package Microsoft.EntityFrameworkCore.Design dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design dotnet add package Microsoft.EntityFrameworkCore.SqlServer”

Then, use the following command to make it a global tool: “dotnet tool install --global dotnet-ef” Finally, use the following command to make a migration and update the database:
“dotnet ef migrations add InitialCreate1 dotnet ef database update”

The FPTBook database is seen in the Microsoft SQL Server Management Studio:

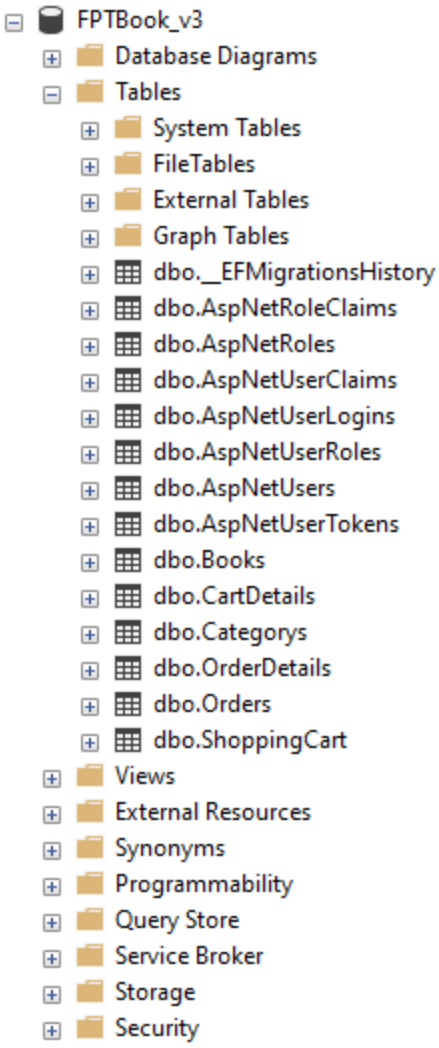


Figure 28 Database of FPTBook project

5. Final screenshots of the application

There are all pages of my website and new frameworks (if they have changed):

- **Sign-in:** Login: For a better look at the Login page, the title has been removed. The navigation pages have been removed.

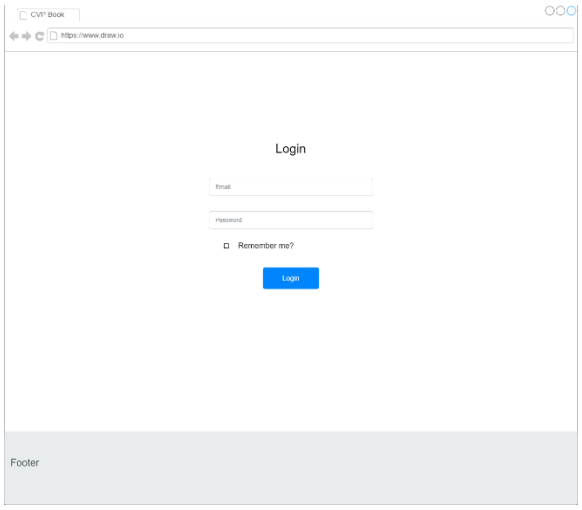


Figure 29 login framework

Figure 30 Login Page

- **Register**: The information fields have been moved to the middle of the page, the input column has been enlarged, and the info column has been changed to a vertical column. The home address has also been deleted.

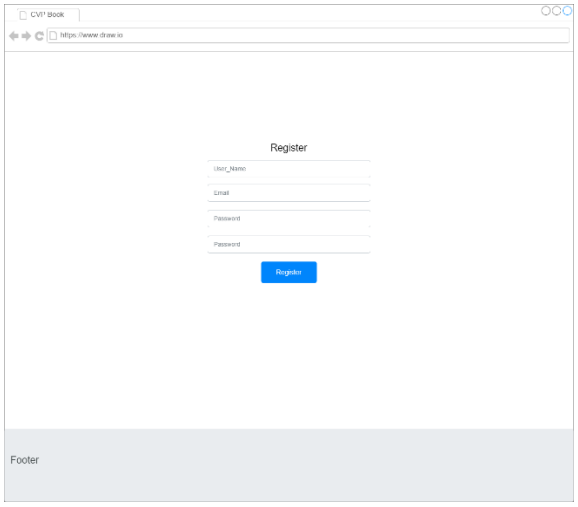


Figure 31 Register framework

Figure 32 Register Page

- **Home**: The navigation elements above have been moved to the right corner of the site, and the product and tool displays have been removed. When the user checks in, the system displays an item responsible for controlling the behavior of users such as store owners, administrators, and consumers. Customer accounts have only basic control, but more privileged accounts, such as store owners and administrators, have access to the inner workings of the system.

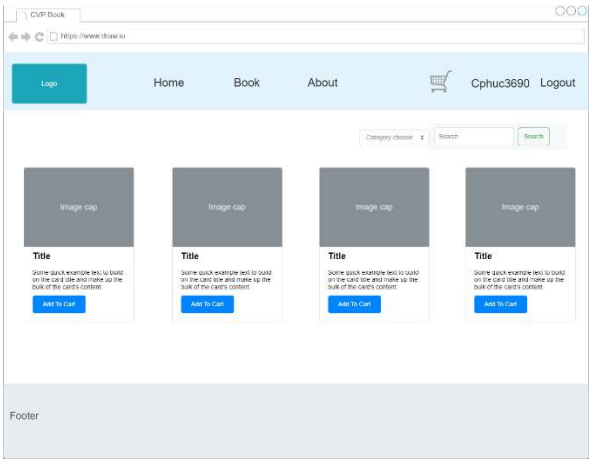


Figure 33 Home framework

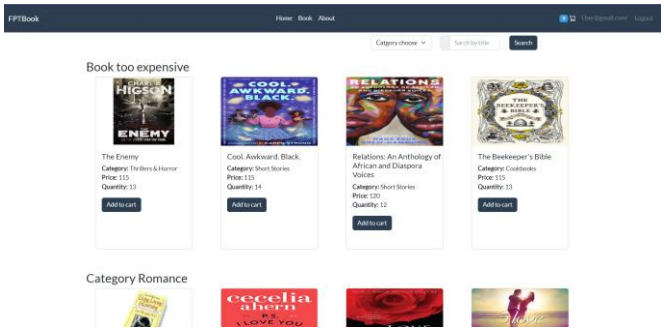


Figure 34 Home Page

- **Products**: On our last site, we moved certain items from the main page to a separate product page, which is a new page added to the system and used to display products for client. The system will display the products connected to the document they are looking for when the user swipes left or right to browse the products on this page or search for the item they are looking for. A small item that appears when the user hovers over a product in the product screen; when they click on it, the system will display the information of the product.

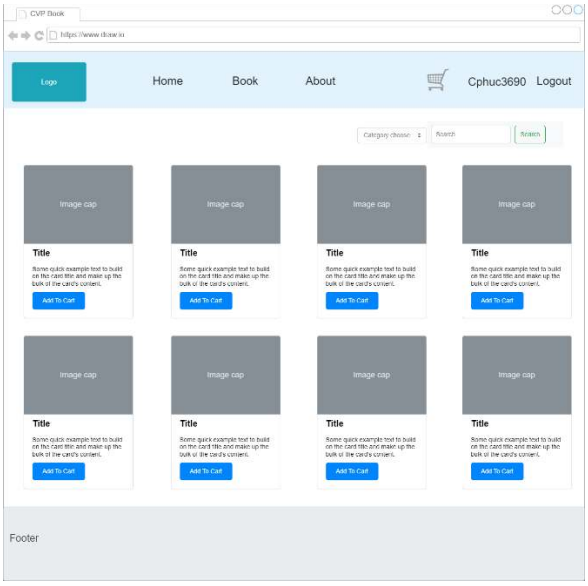


Figure 35 Book framework

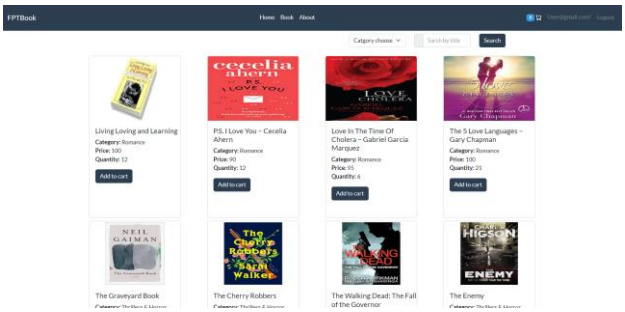


Figure 36 Home Page

- **Details**: To the completed product, we made various alterations and information additions. First, the price was moved up and down, followed by the description. The second component we've established is the author section, which will display after the description but before the category. The third newly added component will be included under the category section and is the publisher. Fourth, instead of displaying the quantity as before, we modified the quantity section to a box form so that the user could enter the quantity.

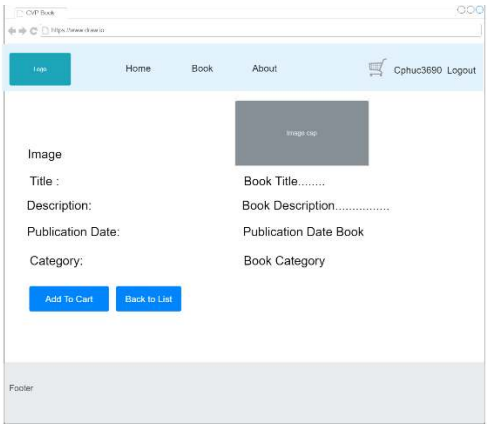


Figure 37 Book Detail framework

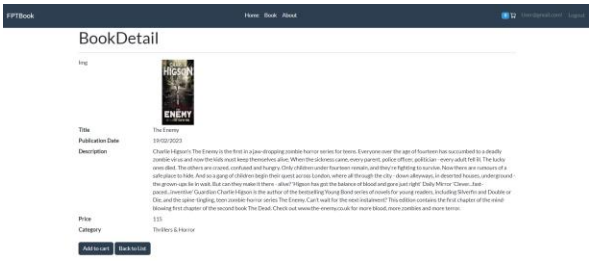


Figure 38 Book Detail Page

- **Cart**: An up or down button to update the cart has been added and some unnecessary elements have been removed from the cart page. The item's title, price, and quantity will all be displayed on the cart page. The system will update the new order and determine the new price depending on the number of items the customer has selected, so when the quantity of a product changes, the customer presses the increase (decrease) button.

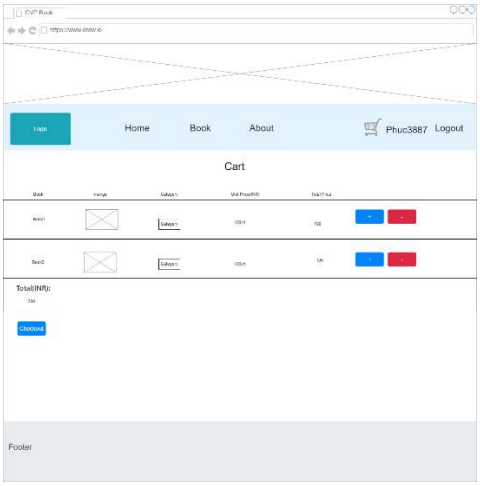


Figure 39 Cart framework

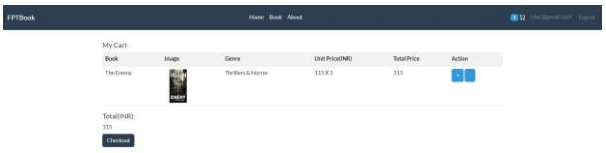


Figure 40 Cart Page

- **Mange account**: The end product's user interface was altered. Particularly, information like Username, Full name, etc. has been moved from its prior placement on the left to the top of the information box.

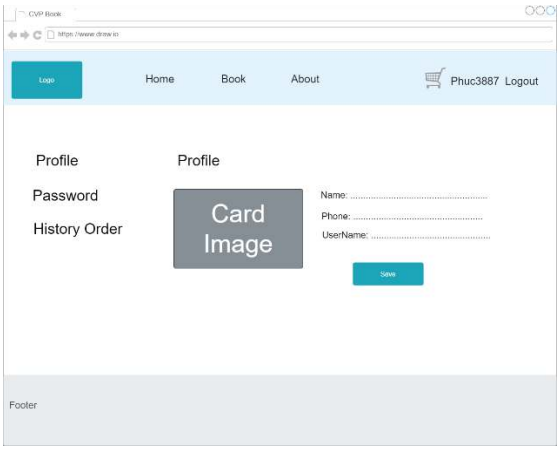


Figure 41 Mange account framework

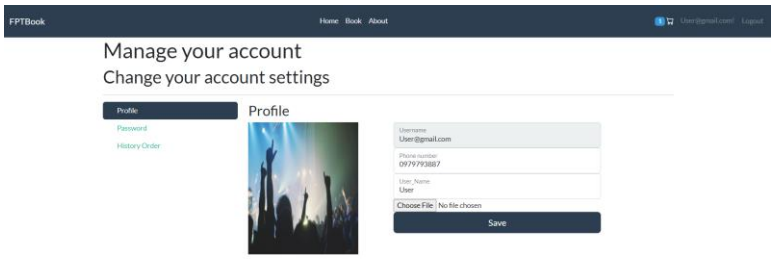


Figure 42 Mange account Page

6. Screenshots of using GitHub or GitLab to manage the source code

The project's source code is hosted on GitHub.

An online site called GitHub makes it simple to publish your code (or projects) Online storage for our personal git repository is provided by GitHub. In essence, it enables you to work with a team of people. Both its own features and all of Git's features are supported.

One way to think of GitHub is as a social networking site where engineers may showcase their work. It might be any undertaking including the creation or redesign of a website, the implementation of an operating system like Linux or Android, etc (geeksforgeeks, 2023)

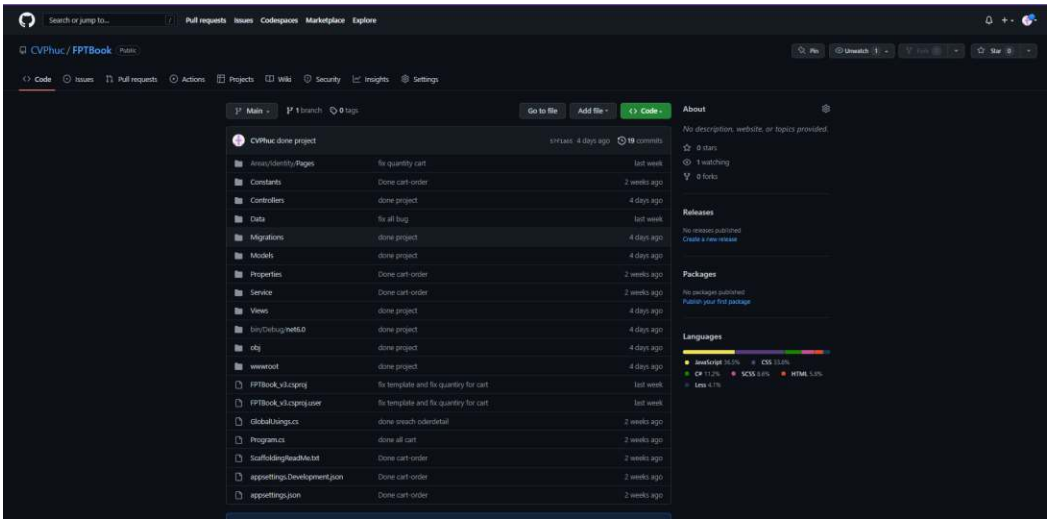


Figure 43 GitHub Code

Then we used the command to clone the archive to the computer.

```
C:\Users\Administrator> git clone https://github.com/CVPhuc/FPTBook
```

Figure 44 Clone code

After completing a function or changing the code, if you want to upload it to github, use the following commands.

- Git add.
- Git commit -m "done project"
- Git push -u origin Main

```
PS E:\Greenwich\1670\FPTBook_v3\FPTBook_v3> git add .
PS E:\Greenwich\1670\FPTBook_v3\FPTBook_v3> git commit -m "done project"
[Main b6eb948] done project
 30 files changed, 157 insertions(+), 264 deletions(-)
 delete mode 100644 Models/ChangePassword.cs
 delete mode 100644 Views/Admin/EditOwner.cshtml
 delete mode 100644 Views/Admin/EditUser.cshtml
 delete mode 100644 Views/Home/Privacy.cshtml
 create mode 100644 wwwroot/Uploads/4330725f-5426-4aa0-a51a-588b602cf0dc.jpg
 delete mode 100644 wwwroot/Uploads/514d030d-ea09-4ebf-af7b-eabaae522fb0A Literary Vampire Novel.jpg
 create mode 100644 wwwroot/Uploads/9f28e7d8-9b3f-4d61-8501-02093a97e8c6Living_Loving_and_Learning.jpg
PS E:\Greenwich\1670\FPTBook_v3\FPTBook_v3> git push -u origin Main
Enumerating objects: 75, done.
Counting objects: 100% (75/75), done.
Delta compression using up to 8 threads
Compressing objects: 100% (34/34), done.
Writing objects: 100% (39/39), 377.63 KiB | 4.02 MiB/s, done.
Total 39 (delta 27), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (27/27), completed with 26 local objects.
To https://github.com/CVPhuc/FPTBook.git
   57f1a65..b6eb948  Main -> Main
branch 'Main' set up to track 'origin/Main'.
PS E:\Greenwich\1670\FPTBook_v3\FPTBook_v3>
```

Figure 45 Upload it to Github

Link Code : https://github.com/CVPhuc/FPTBook

7. Screenshots of using IIS or Azure for the application deployment

IIS is the application we choose to deploy for y the project.

Internet Information Services (IIS) 7 and later provide a request-processing architecture which includes:

- The Windows Process Activation Service (WAS), which enables sites to use protocols other than HTTP and HTTPS.
- A Web server engine that can be customized by adding or removing modules.
- Integrated request-processing pipelines from IIS and ASP.NET.

IIS contains several components that perform important functions for the application and Web server roles in Windows Server® 2008 (IIS 7.0) and Windows Server 2008 R2 (IIS 7.5). Each component has responsibilities, such as listening for requests made to the server, managing processes, and reading configuration files. These components include protocol listeners, such as HTTP.sys, and services, such as World Wide Web Publishing Service (WWW service) and Windows Process Activation Service (WAS).

Figure 46 IIS

First, right-click on Sites and click Add Website to add a website.

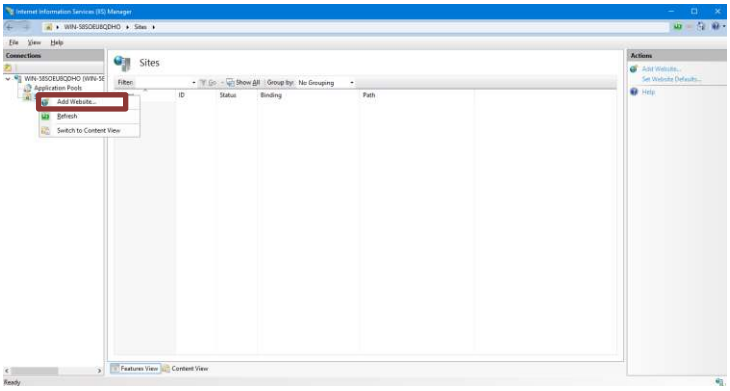


Figure 47 Interface of IIS

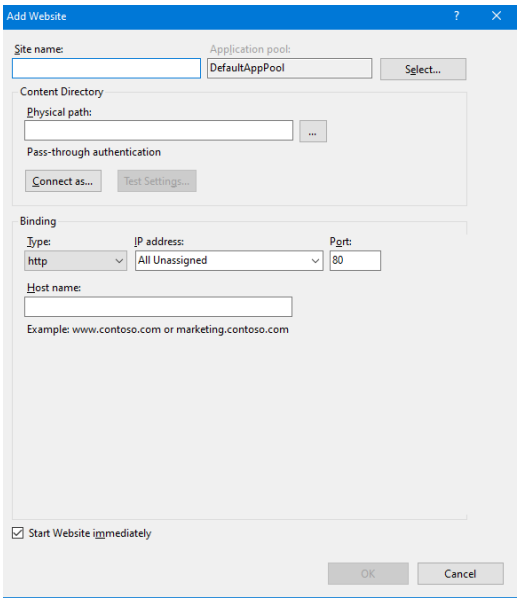


Figure 48 Add Website page

To File Publish, provide the site name and physical path. Select the relevant IP address in Binding and click OK.

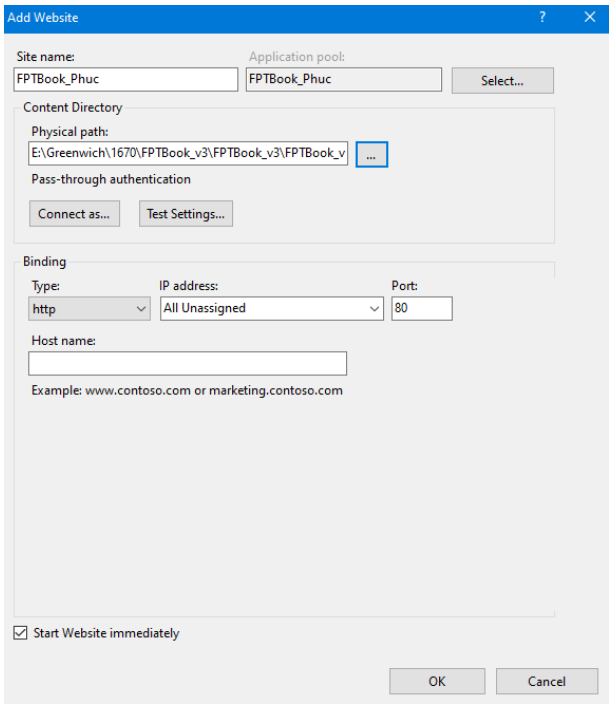


Figure 49 Fill in the information on the website

Then, to view the just launched website, click the Browse Website option in the right sidebar.

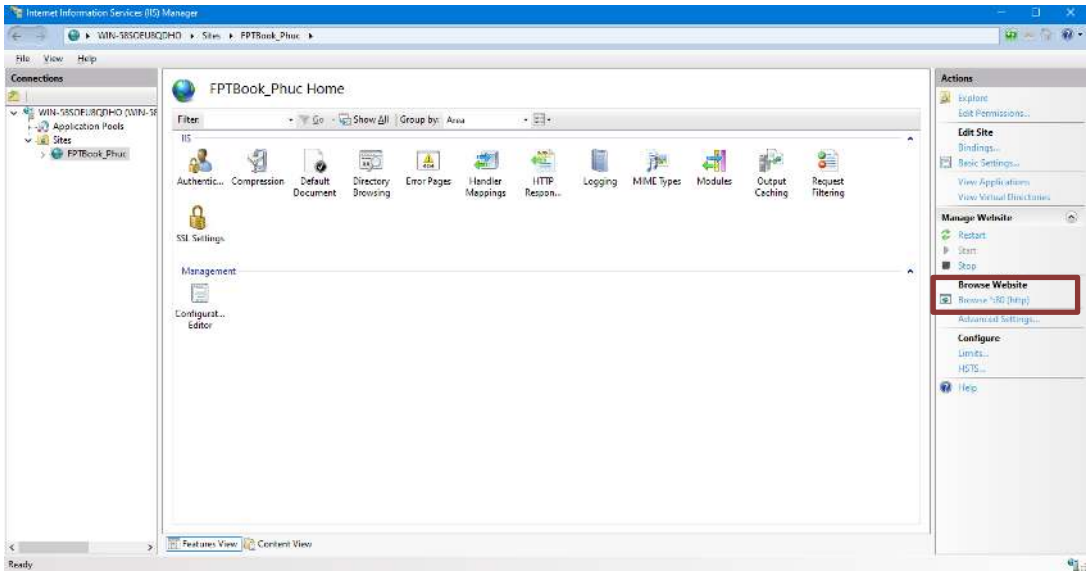


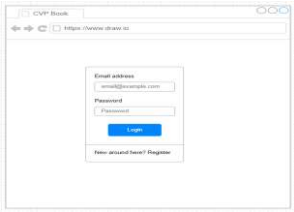
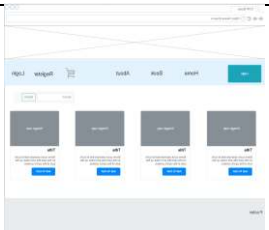
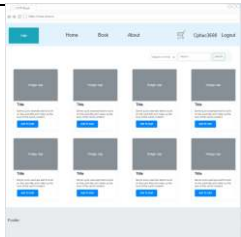
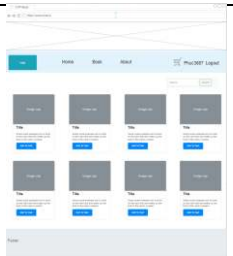
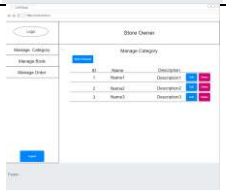
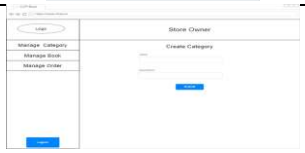
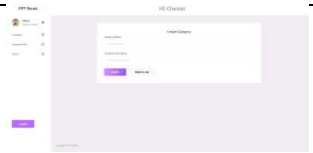
Figure 50 Go to the website

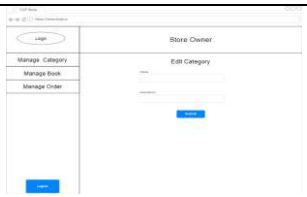
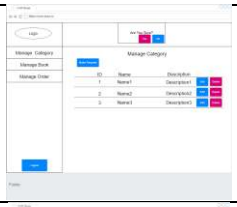
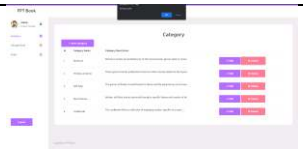
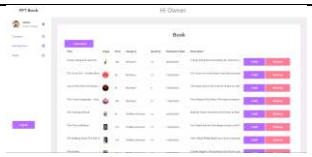
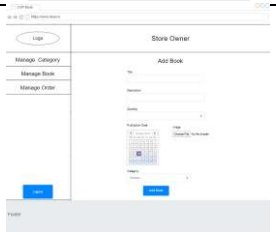
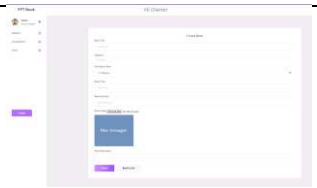
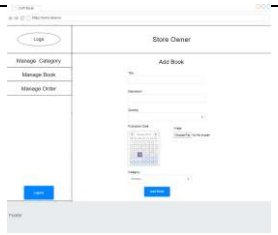
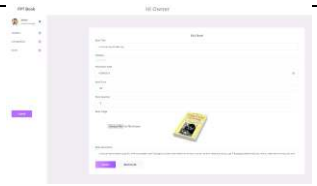
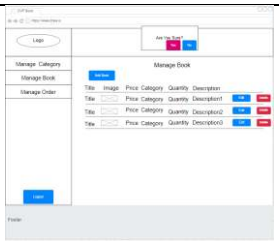
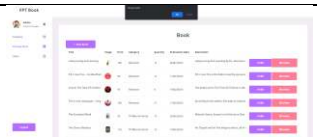
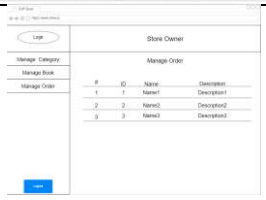
IV. Application evaluation

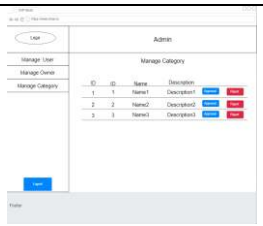
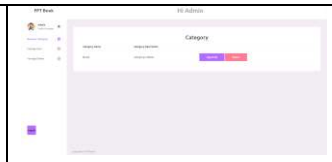
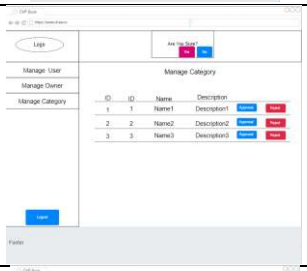
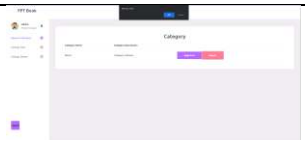
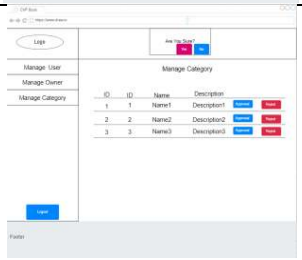
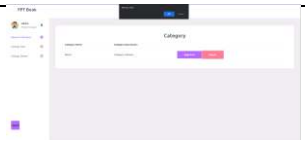
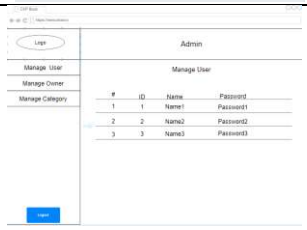
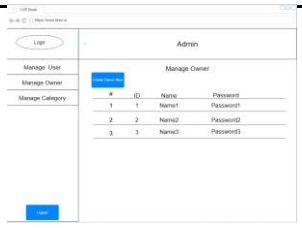
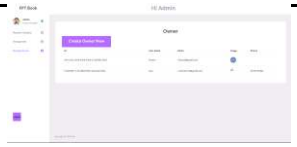
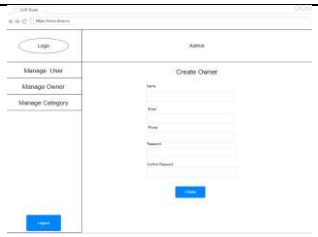
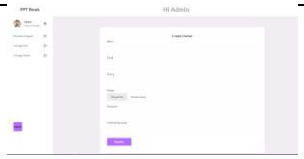
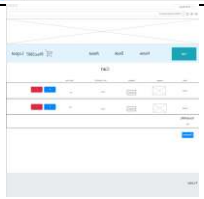
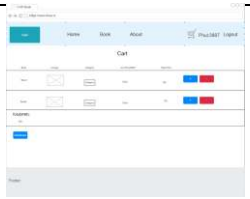
1. Review the performance of the application

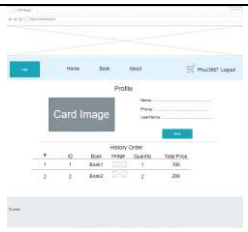
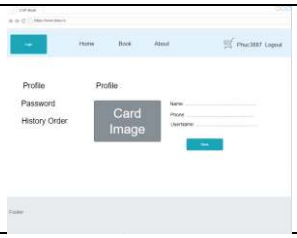
For many different reasons, users have modified their projects. Some of their changes have produced a more complete product than what was originally envisioned. Project management has been made better by the introduction of certain features. In addition to improving the project's convenience, the design also offers a more user-friendly interface for the website.

The following table contrasts Initial Requirements before to change, following change, and following project completion:

No	Use case	Describe	Mock-up	Updated mock-up	Result	Complete (%)	Status
1	Log in	Log in page of the website.				100	**Finish**
2	Register	Register page of the website**.**				100	**Finish**
3	Home	Home page of the website.				100	**Finish**
4	Book	Book page of the website.		No		100	**Finish**
5	Category (Owner)	Category page of the website.		No		100	**Finish**
6	Create Category (Owner)	Create Category		No		100	**Finish**

		page of the website.					
7	Edit Category (Owner)	Edit Category page of the website.		No		100	**Finish**
8	Delete Category (Owner)	Delete Category page of the website.		No		100	**Finish**
9	Book (Owner)	Book (Owner) page of the website.		No		100	**Finish**
10	Create Book (Owner)	Create Book (Owner) page of the website.		No		100	**Finish**
11	Edit Book (Owner)	Edit Book (Owner) page of the website.		No		100	**Finish**
12	Delete Book (Owner)	Delete Book (Owner) page of the website.		No		100	**Finish**
13	Order (Owner)	Order (Owner) page of the website.		No		100	**Finish**

14	Request Category (Admin)	Request Category (Admin) page of the website.		No		100	**Finish**
15	Reject Request Category (Admin)	Reject Request Category (Admin) page of the website.		No		100	**Finish**
16	Approval Request Category (Admin)	Approval Request Category (Admin) page of the website.		No		100	**Finish**
17	Manage User (Admin)	Manage User (Admin) page of the website.		No		100	**Finish**
18	Manage Owner (Admin)	Manage Owner (Admin) page of the website.		No		100	**Finish**
19	Create Owner (Admin)	Create Owner (Admin) page of the website.		No		100	**Finish**
20	Cart (User)	Cart (User) page of the website.				100	**Finish**

21	Manage Account (User)	Manage Account (User) page of the website.				100	**Finish**

2. **Conclude whether the application adapts all requirements, or it needs to be improved later**

The system's primary operations remain steady. It has login and logout functionality, CRUD, role management, the ability for the admin to accept or reject the shop owner's request for a new book catalog. Based on data about each customer, company owner, and administrator, we want to introduce more services in the future, such as a more pleasurable user experience and online payments.

3. **Analyze the factors that influence the performance of the application**

The software's operation was assessed in light of a wide range of scenarios and potential outcomes that may occur during user use. To encrypt data as effectively as possible while eliminating unintentional mistakes, the developer has solved a number of concerns. A customer, for instance, won't be able to use or access the admin capabilities since they don't have the privileges listed in the required paperwork. However, the developer lessens the possibility that users would inadvertently use capabilities for which they are not permitted by showing and hiding functionality based on a user's role.

4. **Evaluate the strengths and weaknesses of the application**

❖ **The website's advantages**:

- The source code is straightforward to maintain, debug, and operate: The MVC architectural pattern and the ASP.NET framework enable each source code module to function differently. Functional modularization improves the organization, readability, and flexibility of the code, reducing the likelihood of errors and other issues. Independent components simplify application design, administration, operation, and maintenance as well as program processing. The MVC methodology also produces a standardized project model that makes accessing the application easier..

- Utilize the tools, procedures, and techniques listed below to develop a useful business application based on a particular Software Design Document: The system's primary objective is to be fully operational. It has CRUD, role management, a login and logout feature, and an

admin who can approve or disapprove the shop owner's request for a new book catalog. Additionally, it offers the option of exporting data to Excel and includes a chart that lists details about each consumer, business owner, and administrator. The classdiagram's entities were implemented in the code. Additionally, each object will contain attributes and methods, as indicated in the class diagram. The system's creator introduced an image option for business owners and administrators to make it more realistic.

- Administrator duties should include the ability to add to, edit, delete from, and view lists, per the requirements specification. These crucial elements are included in the author's solution because the developer satisfactorily met the paper's requirements. The information is sent to the database system whenever a user adds, updates, or deletes something. The data will be updated in response to the user's request. Functions are tested to ensure they are complete, efficient, and error-free..

❖ **The website's disadvantages:**

- **Security**:

+ User information gathering: Crawl data may be used to compile user information. The only data that can be collected using this method is readily available online data. However, there is a sizable amount of personal information, including phone numbers and email addresses. In contrast, the system only encrypts passwords. If a crawler sends the server a lot of requests in an effort to get data, the server might experience issues..

+ Reputational harm: Most e-commerce websites will experience long-term losses due to reputational harm. If their clients and suppliers lose faith in them, particularly smaller online retailers run the risk of having to shut down. A data breach response plan is more important than ever.

- **The user interface**:

+ As was previously said, when there are too many users, it is challenging for the administrator to set a user restriction. Therefore, pagination is necessary. Pagination is a method for connecting several landing pages with relevant content. This makes it easier for the administrator to update and enhance the UI/UX of the program.

+ Book page category-ordered: On our website, customers may immediately discover the book.

- **Lack of functionality**:

+ View users who are logged in and out of the system: Since this is a management application, the administrator will find it very beneficial to be able to manage users who are logged in and out of the system. Additionally, statistics on how many users access the system online as opposed to offline will help the team improve it fast and easily. However, this feature was not used since the team was unable to complete it.

+ The picture error: Administrators may occasionally submit wrong or too big photographs; to enhance user experience, the error will be displayed. The information will be shown if the folder has no photographs.

The website's difficulties:

- The first project for the development team presents a number of challenges because it is unfamiliar to them. Members frequently argue when pulling and submitting source code because of incompatible libraries and hardware. We encountered a problem and had to change our strategy when I concurrently uploaded the updated source code to GitHub. We encountered this problem after learning how to fork on GitHub.

-I have consented to remove a few features, but we will work to eventually add all features to the system. We also found some very serious bugs that took us days to fix, as well as some features that we initially intended to add to the system but that, once they were added, resulted in conflicts and unfixable errors.

References

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-visual-studio/

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/html5-introduction/

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/css/

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/bootstrap/

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/differences-between-net-core-and-net-framework/?ref=gcse

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/mvc-framework-introduction/?ref=gcse

geeksforgeeks, 2023. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-to-github/