  **ASSIGNMENT 2 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Business** | | |
| **Unit number and title** | **Unit 30: Application Development** | | |
| **Submission date** | 24/12/2022 | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Hoang Minh Tan | **Student ID** | GCC19009 |
| **Class** | 1670 - TUTOR | **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** | TAN |

**Grading grid**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **P4** | **P5** | **P6** | **M3** | **M4** | **M5** | **D2** | **D3** |
|  |  |  |  |  |  |  |  |

1

|  |  |  |
| --- | --- | --- |
| **Summative Feedback:**  **Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Internal Verifier’s Comments:** | | |
| **Signature & Date:** | | |

|  |
| --- |
| **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. |

**Table of Contents**

[Table of tables 7](#_Toc122803888)

[Assignment 2- Application Development 7](#_Toc122803889)

[Introduction 7](#_Toc122803890)

[Chapter 1 – Peer Review and Feedback Analysis (P4-M3-D2) 7](#_Toc122803891)

[I. Formal questionnaire that effectively reviews the business application, problem definition statement, proposed solution and development strategy. 7](#_Toc122803892)

[Chapter 2 – Application Development (P5 – M4) 10](#_Toc122803893)

[II. Entity Relationship Diagram (ERD) 10](#_Toc122803894)

[III. Develop a functional business application 10](#_Toc122803895)

[3.1. Preferred tools, techniques and methodologies. 10](#_Toc122803896)

[3.1.1. Develop tool 10](#_Toc122803897)

[3.1.2. Technique 10](#_Toc122803898)

[3.1.3. Methodologies 11](#_Toc122803899)

[3.2. Folder structure of the application 12](#_Toc122803900)

[3.3. Source code samples of the application with explanation 12](#_Toc122803901)

[3.3.1. Models 12](#_Toc122803902)

[4.3.2 Views 21](#_Toc122803903)

[4.3.3 Controllers 23](#_Toc122803904)

[IV. Screenshots of using GitHub 63](#_Toc122803905)

[Chapter 3 – Application Evaluation (P6 – M5) 65](#_Toc122803906)

[V. Comparation initial requirements 65](#_Toc122803907)

[References: 76](#_Toc122803908)

[Figure 1: ERD of FPT bookstore application 10](#_Toc122803711)

[Figure 2: MVC model 11](#_Toc122803712)

[Figure 3: Folder structure of application 12](#_Toc122803713)

[Figure 4: Models folder 12](#_Toc122803714)

[Figure 5: View folder of application 23](#_Toc122803715)

[Figure 6: Controller folder 24](#_Toc122803716)

[Figure 7: GitHub repository 63](#_Toc122803717)

[Table 1: Questionnaire about FPT bookstore application 7](#_Toc122803862)

[Table 2: review and comparation with initial 65](#_Toc122803863)

# Table of tables

Table 1- Questionnaire about FPT bookstore application ............................................................................. 9

Table 2- review and comparation with initial .............................................................................................. 60

# Assignment 2- Application Development

# Introduction

This is the continuous of the assignment 1 as a last report of subject application development. In this report I will discuss in 3 main contents has split into 3 chapter.

In chapter 1- create peer review section (questionnaire and answers, your reflection on the feedback).

In chapter 2- Development section (how to develop and test the application, what is the result). In chapter 3- Review section (review, analyze and critical evaluate my application).

# Chapter 1 – Peer Review and Feedback Analysis (P4-M3-D2)

# Formal questionnaire that effectively reviews the business application, problem definition statement, proposed solution and development strategy.

Table 1: Questionnaire about FPT bookstore application

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No/Function** | **Question** | **Creater** | **Date** | **Answer** | **Answerer** | **Date** |
| 1. Lay out | I didn't see a requirement for the website's color, so I went with blue, and white. Is that okay? | Hoang Minh Tan | 24/12/2022 | Yes, it’s no problem. | Nguyen Quang Vinh | 24/12/2022 |
| 2. Log-in | Did the remember me feature exist on the login page? used to keep track of the login and password for checking. | Hoang Minh Tan | 24/12/2022 | This feature is unnecessary since it prevents someone from stealing a user's PC and gaining access to the system. | Nguyen Quang Vinh | 24/12/2022 |
| Is the website's password-forget feature at the login form necessary? | Hoang Minh Tan | 23/12/2022 | Even while we do not now require the feature "forget password," having it would be preferable. If a user forgets their password, they can ask the administrator to reset it. | Nguyen Quang Vinh | 24/12/2022 |
| Is the website need the function log-in with external account like | Hoang Minh Tan | 23/12/2022 | In this case, we still not need this function, but in the future when the website is more stable. | Nguyen Quang Vinh | 24/12/2022 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | facebook, google or not? |  |  |  |  |  |
| 3. Register | Is the user register have the function confirm password. | Hoang Minh Tan | 23/12/2022 | Yes, it’s need to confirm with the user their password. | Nguyen Quang Vinh | 24/12/2022 |
| 4. Edit information | Is the admin can edit the user’s information? | Hoang Minh Tan | 23/12/2022 | Of course, we need the function that the admin can edit their information and user’s information. | Nguyen Quang Vinh | 24/12/2022 |
| 5. Searching | I'm not sure what you want to look for in the index's search field. I made the decision to seek for books using their names. Do you require anything? | Hoang Minh Tan | 23/12/2022 | It's OK, but I'd like to see additional search features in the upcoming edition, like the ability to search by author or category. | Nguyen Quang Vinh | 24/12/2022 |
| 6. User role | I recognize the need for users to update their data. Has it been understood correctly? | Hoang Minh Tan | 23/12/2022 | The user can update their data, yes. | Nguyen Quang Vinh | 24/12/2022 |
| I am aware that adding books to carts, removing books from cards, and confirming orders are all essential before purchasing books. Has it been understood correctly? | Hoang Minh Tan | 23/12/2022 | Yes, the user must sign in before they may purchase a book. | Nguyen Quang Vinh | 24/12/2022 |
| 7. Admin role | I am aware that the admin account may switch between the admin layout and the user layout. Is it not? | Hoang Minh Tan | 23/12/2022 | Yes, it is required for the administrator to observe whether the book they added has any issues without having to change their account. | Nguyen Quang Vinh | 24/12/2022 |
| I understand that the the admin can add/edit/ delete admin account or user account. Isn’t it? | Hoang Minh Tan | 23/12/2022 | Yes, it is. The admin can do it anyway. | Nguyen Quang Vinh | 24/12/2022 |
| Is there have the confirm alert when | Hoang Minh Tan | 23/12/2022 | Of course, yes. It’s necessary to confirm | Nguyen Quang Vinh | 24/12/2022 |
|  | the admin delete an item out of a list.  Isn’t it? |  |  | and avoid to detele wrong. |  |  |

# Chapter 2 – Application Development (P5 – M4)

# Entity Relationship Diagram (ERD)



Figure 1: ERD of FPT bookstore application

# Develop a functional business application

## Preferred tools, techniques and methodologies.

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

### Develop tool

*An integrated development environment (IDE), such as Visual Studio 2022, supports many different facets of software development. You may edit, debug, develop, and publish an app using the Visual Studio IDE as your creative launchpad. Compilers, code completion tools, graphical designers, and many other capabilities are included in Visual Studio in addition to the conventional editor and debugger that are offered by the majority of IDEs to improve the software development process*. (Microsoft, 2022)

*Developed and sold by Microsoft, SQL Server is a relational database management system (RDBMS). The main job of the SQL Server, a database server, is to store and retrieve data utilized by other programs*. (SQLservertutorial, 2022)

### Technique

* *HTML5/CSS3 - CSS3 is just the newest "release" of CSS, including more sophisticated functionality than previous iterations. Similar to how HTML5 is just HTML and CSS3 is just CSS. HTML5 and CSS3 both developed into marketing buzzwords that go well beyond the technologies they relate to.* (byte, 2021)
* *The most widely used HTML, CSS, and JavaScript framework for creating responsive, mobile-first websites is Bootstrap*. (byte, 2021)
* *With tools and modules tailored to the development of online applications, ASP.NET expands the capabilities of the.NET developer platform*. (Microsoft, 2022)

### Methodologies

A typical software design paradigm for implementing user interfaces, data, and controlling logic is MVC (Model-View-Controller). It stresses a separation between the business logic and appearance of the product. This "separation of issues" allows for a better labor-force distribution and better upkeep.

MVC is the foundation for several additional design patterns, including MVVM (Model-View-View model), MVP (Model View Presenter), and MVW.(Model-View-Whatever).

The three parts of the MVC software-design pattern can be described as follows:

* **Model:** Manages data and business logic.
* **View:** Handles layout and display.
* **Controller:** Routes commands to the model and view parts.

(Glossary, 2022)

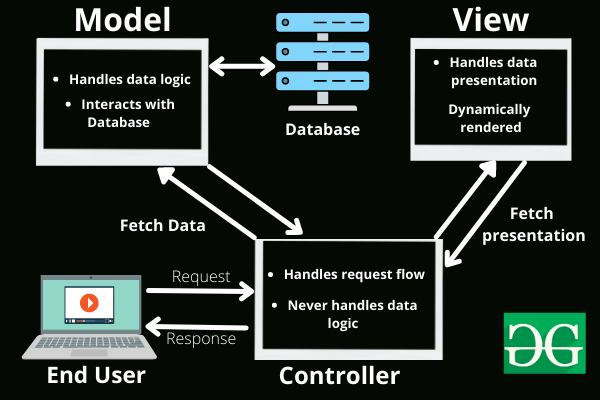


Figure 2: MVC model

## Folder structure of the application

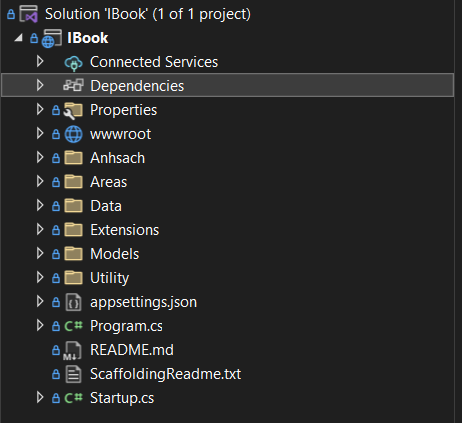


Figure 3: Folder structure of application

## Source code samples of the application with explanation

### Models

Base on the ERD above, here is some models of application

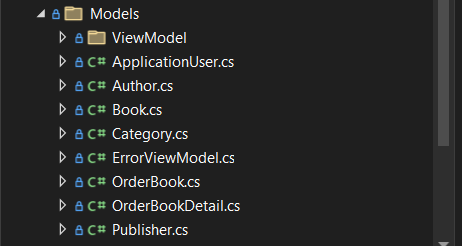


Figure 4: Models folder

|  |
| --- |
| **ApplicationUser.cs** |
| using Microsoft.AspNetCore.Identity;  using System;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  using System.Linq;  using System.Threading.Tasks;  namespace IBook.Models  {  public class ApplicationUser : IdentityUser  {  [Display(Name =("Staff"))]  public string Name { get; set; }  [NotMapped]  public bool IsSupperAdmin { get; set; }  }  } |

|  |
| --- |
| **Author.cs** |
| using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace IBook.Models  {  public class Author  {  [Display(Name = "Author ID")]  public int ID { get; set; }  [Required(ErrorMessage = "This value cannot be left blank")]  [Display(Name = "Name Author")]  public string Name { get; set; }  [Display(Name = "Story")]  public string History { get; set; }  public virtual ICollection<Book> Books { get; set; }  }  } |

|  |
| --- |
| **Book.cs** |
| using System;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace IBook.Models  {  public class Book  {  public int BookID { get; set; }  [Display(Name = "Book name")]  public string Title { get; set; }  [Display(Name = "Summary")]  public string Summary { get; set; }  [Display(Name = "Quantity")]  public int Quantities { get; set; }  [DataType(DataType.Date)]  [Display(Name = "Publication date")]  public DateTime CreateDate { get; set; }  [DataType(DataType.Date)]  [Display(Name = "Reprint date")]  public DateTime ModifiedDate { get; set; }  [Display(Name = "Number of pages")]  public int NumberOfPages { get; set; }  [Display(Name = "Price")]  public Int64 Price { get; set; }  [Display(Name = "Cover image")]  public string ImageUrl { get; set; }  [Display(Name = "Permission to sell")]  public Boolean IsPurchase { get; set; }  [Display(Name = "Author")]  public int AuthorID { get; set; }  [Display(Name = "Category")]  public int CategoryID { get; set; }  [Display(Name = "Publisher")]  public int PublisherID { get; set; }  [Display(Name = "Author")]  public virtual Author Author { get; set; }  [Display(Name = "Category")]  public virtual Category Category { get; set; }  public virtual ICollection<OrderBookDetail> OrderBookDetails { get; set; }  [Display(Name = "Author")]  public virtual Publisher Publisher { get; set; }  }  } |

|  |
| --- |
| **Category.cs** |
| using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  namespace IBook.Models  {  public class Category  {  [Display(Name = "CategoryID")]  public int ID { get; set; }  [Required(ErrorMessage = "This value cannot be left blank")]  [Display(Name = "NameID")]  public string Name { get; set; }  [Display(Name = "Description")]  public string Description { get; set; }  public virtual ICollection<Book> Books { get; set; }  }  } |

|  |
| --- |
| **OrderBook.cs** |
| using System;  using System.Collections.Generic;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  using System.Linq;  using System.Threading.Tasks;  namespace IBook.Models  {  public class OrderBook  {  [Display(Name = "Code orders")]  public int OrderID { get; set; }  [Display(Name = "Delivery date")]  public DateTime AppointmentDate { get; set; }  [NotMapped]  [Display(Name = "Delivery time")]  public DateTime AppointmentTime { get; set; }  [Display(Name = "Customer name")]  public string CustomerName { get; set; }  [Display(Name = "Delivery address")]  public string CustomerAddress { get; set; }  [Display(Name = "Contact phone number")]  public string CustomerPhone { get; set; }  [Display(Name = "Confirm")]  public Boolean isConfirmed { get; set; }  public virtual ICollection<OrderBookDetail> OrderBookDetails { get; set; }  }  } |

|  |
| --- |
| **OrderBookDetail.cs** |
| using System;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace IBook.Models  {  public class OrderBookDetail  {  [Display(Name = "Order Detail ID")]  [ForeignKey("OrderBook")]  public int OrderBookID { get; set; }  [Display(Name = "Book code")]  [ForeignKey("Book")]  public int BookID { get; set; }  [Display(Name = "Book title")]  public virtual Book Book { get; set; }  [Display(Name = "Code orders")]  public virtual OrderBook OrderBook { get; set; }  }  } |

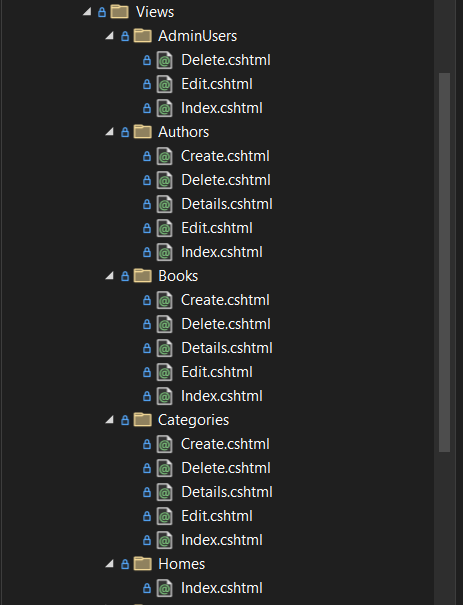
|  |
| --- |
| **Publisher.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using System.ComponentModel.DataAnnotations;  using System.ComponentModel.DataAnnotations.Schema;  namespace IBook.Models  {  public class Publisher  {  [Display(Name = "Publisher code")]  public int ID { get; set; }  [Required(ErrorMessage = "This value cannot be left blank")]  [Display(Name = "Publisher code")]  public string Name { get; set; }  [Display(Name = "Describe")]  public string Description { get; set; }  public virtual ICollection<Book> Books { get; set; }  }  } |

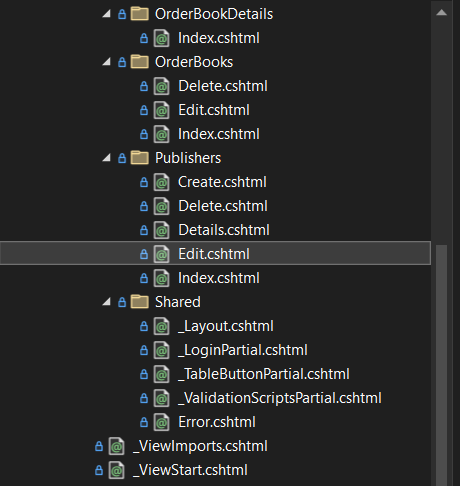
|  |
| --- |
| **ErrorViewModel.cs** |
| using System;  namespace IBook.Models  {  public class ErrorViewModel  {  public string RequestId { get; set; }  public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);  }  } |

|  |
| --- |
| BookViewModel |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace IBook.Models.ViewModel  {  public class BookViewModel  {  public Book book { get; set; }  public IEnumerable<Category> categories { get; set; }  public IEnumerable<Author> authors { get; set; }  }  } |

|  |
| --- |
| ShoppingCartViewModel |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  namespace IBook.Models.ViewModel  {  public class ShoppingCartViewModel  {  public List<Book> books { get; set; }  public OrderBook orderBook { get; set; }  }  } |

### 4.3.2 Views





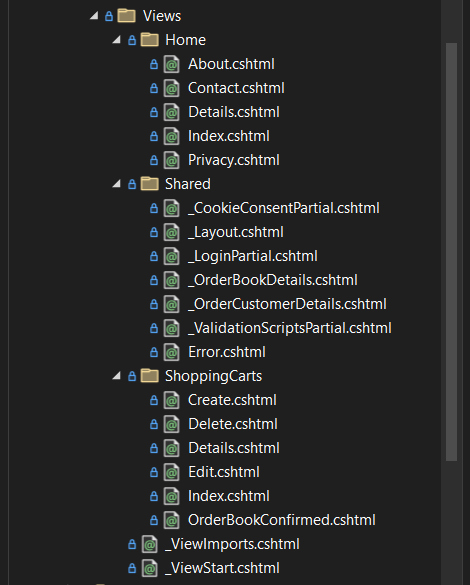


Figure 5: View folder of application

### 4.3.3 Controllers

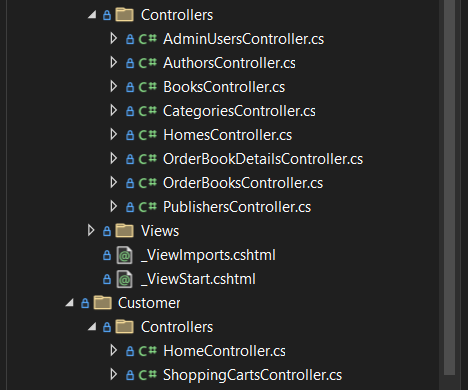


Figure 6: Controller folder

|  |  |  |  |
| --- | --- | --- | --- |
| **HomeController.cs** | | | |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to call the view of homepage to display. |
| Index | Post | Receive book’s name at search box and find the book of book user search for to display. |
| Contact | HttpGET | Used to call the help page to display. |
| Bookdetail | HttpGET | Get the id’s book, then find it in list book and call the detail page to display that book. |
| Cart | HttpGET | Use to call the cart page to display. |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using IBook.Models;  using IBook.Data;  using Microsoft.EntityFrameworkCore;  using IBook.Extensions;  using ReflectionIT.Mvc.Paging;  namespace IBook.Controllers  {  [Area("Customer")]  public class HomeController : Controller  {  private readonly ApplicationDbContext \_db;  public HomeController(ApplicationDbContext db)  {  \_db = db;  }  public async Task<IActionResult> Index(int page = 1, string findstring = null)  {  var Query = \_db.Books.AsNoTracking().OrderBy(x => x.BookID);  if(!string.IsNullOrEmpty(findstring))  {  Query = Query.Where(x => x.Title.Contains(findstring)).OrderBy(x => x.BookID);  }  var bookList = await PagingList<Book>.CreateAsync(Query, 6, page);  return View(bookList);  }  public IActionResult About()  {  ViewData["Message"] = "Your application description page.";  return View();  }  public IActionResult Contact()  {  ViewData["Message"] = "Your contact page.";  return View();  }  public IActionResult Privacy()  {  return View();  }  [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]  public IActionResult Error()  {  return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });  }  public async Task<IActionResult> Details(int id)  {  var book = await \_db.Books  .Include(b => b.Author)  .Include(b => b.Category)  .Include(b=>b.Publisher)  .FirstOrDefaultAsync(m => m.BookID == id);  return View(book);  }  [HttpPost, ActionName("Details")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DetailsPost(int id)  {  List<int> lstShoppingCart = HttpContext.Session.Get<List<int>>("ssShoppingCart");  if(lstShoppingCart==null)  {  lstShoppingCart = new List<int>();  }  int flag = 0;  foreach(int item in lstShoppingCart)  {  if (item == id)  flag++;  }  if(flag == 0)  lstShoppingCart.Add(id);  HttpContext.Session.Set("ssShoppingCart", lstShoppingCart);  return RedirectToAction("Index", "Home", new { area = "Customer" });  }  public IActionResult Remove(int id)  {  List<int> lstShoppingCart = HttpContext.Session.Get<List<int>>("ssShoppingCart");  if(lstShoppingCart.Count > 0)  {  if(lstShoppingCart.Contains(id))  {  lstShoppingCart.Remove(id);  }  }  HttpContext.Session.Set("ssShoppingCart", lstShoppingCart);  return RedirectToAction(nameof(Index));  }  }  } | | | |

**View of Home controller:**

|  |  |
| --- | --- |
| **View of Home controller** | |
| Bookdetail |  |
| Home |  |
| OrderBooks |  |
| Contact |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AdminUsersController.cs** | | | | |
| **Explanation** | | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Use to call the admin homepage after log-in success and calculate the total book, order record and account stored in database to display in statistic admin home page. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using IBook.Data;  using IBook.Models;  using IBook.Utility;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace IBook.Areas.Admin.Controllers  {  [Authorize(Roles = SD.SupperAdminEndUser)]  [Area("Admin")]  public class AdminUsersController : Controller  {  private readonly ApplicationDbContext \_context;  public AdminUsersController(ApplicationDbContext context)  {  \_context = context;  }  public IActionResult Index()  {  return View(\_context.ApplicationUser.ToList());  }  public async Task<IActionResult> Edit(string id)  {  if (id == null || id.Trim().Length == 0)  {  return NotFound();  }  var userFromDb = await \_context.ApplicationUser.FindAsync(id);  if (userFromDb == null)  {  return NotFound();  }  return View(userFromDb);  }  [HttpPost]  [ValidateAntiForgeryToken]  public IActionResult Edit(string id, ApplicationUser applicationUser)  {  if (id != applicationUser.Id)  {  return NotFound();  }  if (ModelState.IsValid)  {  ApplicationUser userFromDb = \_context.ApplicationUser.Where(b => b.Id == id).FirstOrDefault();  userFromDb.Name = applicationUser.Name;  userFromDb.PhoneNumber = applicationUser.PhoneNumber;  \_context.SaveChanges();  return RedirectToAction(nameof(Index));  }  return View(applicationUser);  }  public async Task<IActionResult> Delete(string id)  {  if (id == null || id.Trim().Length == 0)  {  return NotFound();  }  var userFromDb = await \_context.ApplicationUser.FindAsync(id);  if (userFromDb == null)  {  return NotFound();  }  return View(userFromDb);  }  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public IActionResult DeletePOST(string id, ApplicationUser applicationUser)  {  ApplicationUser userFromDb = \_context.ApplicationUser.Where(b => b.Id == id).FirstOrDefault();  userFromDb.LockoutEnd = DateTime.Now.AddYears(1000);  \_context.SaveChanges();  return RedirectToAction(nameof(Index));  }  }  } | | | | |
| **View of Admin controller** | | | | |
| Index |  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderBooksController.cs** | | | |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session user before access to the cart, then call the view of cart to display if user have login success. |
| Add | HttpGET | This action as a function use to add a book to the cart and count all the item in cart and save it into the session count to display in the cart button in the nav-bar. |
| DeleteItem | HttpGET | This is a function use to delete item in cart and update the session cart. |
| MakeOrder | HttpGET | Used to call the view confirm order to display. |
| MakeOrder | HttpPOST | Get the information of user has fill in the confirm form into the database after validation, then call the index page when make a success order. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using IBook.Data;  using IBook.Models;  using Microsoft.AspNetCore.Authorization;  using IBook.Utility;  namespace IBook.Areas.Admin.Controllers  {  [Authorize]  [Area("Admin")]  public class OrderBooksController : Controller  {  private readonly ApplicationDbContext \_context;  public OrderBooksController(ApplicationDbContext context)  {  \_context = context;  }  // GET: Admin/OrderBooks  public async Task<IActionResult> Index()  {  return View(await \_context.OrderBooks.ToListAsync());  }  // GET: Admin/OrderBooks/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var orderBook = await \_context.OrderBooks.FindAsync(id);  if (orderBook == null)  {  return NotFound();  }  return View(orderBook);  }  // POST: Admin/OrderBooks/Edit/5  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("OrderID,OrderDate,CustomerName,CustomerAddress,CustomerPhone,isConfirmed")] OrderBook orderBook)  {  if (id != orderBook.OrderID)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(orderBook);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!OrderBookExists(orderBook.OrderID))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  return View(orderBook);  }  // GET: Admin/OrderBooks/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var orderBook = await \_context.OrderBooks  .FirstOrDefaultAsync(m => m.OrderID == id);  if (orderBook == null)  {  return NotFound();  }  return View(orderBook);  }  // POST: Admin/OrderBooks/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var orderBook = await \_context.OrderBooks.FindAsync(id);  \_context.OrderBooks.Remove(orderBook);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool OrderBookExists(int id)  {  return \_context.OrderBooks.Any(e => e.OrderID == id);  }  }  } | | | |

|  |  |
| --- | --- |
| **View of Cart controller** | |
| Index |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **CategoryController.cs** | | | |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session of admin is valid or not, then call the view of category to display. |
| AddCategory | HttpGET | Used to call the view of add new category to display. |
| AddCategory | HttpPOST | Add a new category into the database if it information is valid. |
| EditCategory | HttpGET | Used to call the view of edit category with the id category have contain in the database to display. |
| EditCategory | HttpPOST | Get the new information save to the database if it’s valid. |
| DeleteCategory | HttpGET | Used to delete a category out of database. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using IBook.Data;  using IBook.Models;  using Microsoft.AspNetCore.Authorization;  using IBook.Utility;  namespace IBook.Areas.Admin.Controllers  {  [Authorize(Roles = SD.SupperAdminEndUser)]  [Area("Admin")]  public class CategoriesController : Controller  {  private readonly ApplicationDbContext \_context;  public CategoriesController(ApplicationDbContext context)  {  \_context = context;  }  // GET: Admin/Categories  public async Task<IActionResult> Index()  {  return View(await \_context.Categories.ToListAsync());  }  // GET: Admin/Categories/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var category = await \_context.Categories  .FirstOrDefaultAsync(m => m.ID == id);  if (category == null)  {  return NotFound();  }  return View(category);  }  // GET: Admin/Categories/Create  public IActionResult Create()  {  return View();  }  // POST: Admin/Categories/Create  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create([Bind("ID,Name,Description")] Category category)  {  if (ModelState.IsValid)  {  \_context.Add(category);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  return View(category);  }  // GET: Admin/Categories/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var category = await \_context.Categories.FindAsync(id);  if (category == null)  {  return NotFound();  }  return View(category);  }  // POST: Admin/Categories/Edit/5  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("ID,Name,Description")] Category category)  {  if (id != category.ID)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(category);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!CategoryExists(category.ID))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  return View(category);  }  // GET: Admin/Categories/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var category = await \_context.Categories  .FirstOrDefaultAsync(m => m.ID == id);  if (category == null)  {  return NotFound();  }  return View(category);  }  // POST: Admin/Categories/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var category = await \_context.Categories.FindAsync(id);  \_context.Categories.Remove(category);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool CategoryExists(int id)  {  return \_context.Categories.Any(e => e.ID == id);  }  }  } | | | |

|  |  |
| --- | --- |
| ViewCategoriesController | |
| Index |  |

|  |  |
| --- | --- |
| EditCategory |  |
| DeleteCategory |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **BookController.cs** | |  |  |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session of admin is valid or not, then call the view of book to display or call the log-in page again. |
| AddBook | HttpGET | Used to call the view of add new book to display. |
| AddBook | HttpPOST | Add a new book into the database if the information have filled in add new book is valid. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | EditBook | HttpGET | Used to call the view of edit book with the book’s id have contain in the database to display. |
| EditBook | HttpPOST | Get the new information have change in the edit form save to the database if it’s valid. |
| DeleteBook | HttpGET | Used to delete a book item out of database with valid id. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using IBook.Data;  using IBook.Models;  using Microsoft.AspNetCore.Hosting.Internal;  using System.IO;  using IBook.Utility;  using Microsoft.AspNetCore.Authorization;  using ReflectionIT.Mvc.Paging;  namespace IBook.Areas.Admin.Controllers  {  [Authorize(Roles = SD.SupperAdminEndUser)]  [Area("Admin")]  public class BooksController : Controller  {  private readonly ApplicationDbContext \_context;  private readonly HostingEnvironment \_hostingEnvironment;  public BooksController(ApplicationDbContext context, HostingEnvironment hostingEnvironment)  {  \_context = context;  \_hostingEnvironment = hostingEnvironment;  }  // GET: Admin/Books  public async Task<IActionResult> Index(int page = 1, string findstring = null)  {  var applicationDbContext = \_context.Books.AsNoTracking().Include(b => b.Author).Include(b => b.Category).Include(b => b.Publisher).OrderBy(x => x.BookID);  if(!string.IsNullOrEmpty(findstring))  {  applicationDbContext = applicationDbContext.Where(x => x.Title.Contains(findstring)).OrderBy(s => s.BookID);  }  var bookList = await PagingList<Book>.CreateAsync(applicationDbContext, 6, page);  return View(bookList);  }  // GET: Admin/Books/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var book = await \_context.Books  .Include(b => b.Author)  .Include(b => b.Category)  .Include(b => b.Publisher)  .FirstOrDefaultAsync(m => m.BookID == id);  if (book == null)  {  return NotFound();  }  return View(book);  }  // GET: Admin/Books/Create  public IActionResult Create()  {  ViewData["AuthorID"] = new SelectList(\_context.Authors, "ID", "Name");  ViewData["CategoryID"] = new SelectList(\_context.Categories, "ID", "Name");  ViewData["PublisherID"] = new SelectList(\_context.Publishers, "ID", "Name");  return View();  }  // POST: Admin/Books/Create  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create([Bind("BookID,Title,Summary,Quantities,CreateDate,ModifiedDate,NumberOfPages,Price,ImageUrl,IsPurchase,AuthorID,CategoryID,PublisherID")] Book book)  {  if (ModelState.IsValid)  {  //Image being saved  string webRootPath = \_hostingEnvironment.WebRootPath;  var files = HttpContext.Request.Form.Files;  if (files.Count != 0)  {  //Image has been uploaded  var uploads = Path.Combine(webRootPath, SD.ImageFolder);  var extension = Path.GetExtension(files[0].FileName);  using (var filestream = new FileStream(Path.Combine(uploads, book.Title + extension), FileMode.Create))  {  files[0].CopyTo(filestream);  }  book.ImageUrl = @"\" + SD.ImageFolder + @"\" + book.Title + extension;  }  else  {  //when user does not upload image  var uploads = Path.Combine(webRootPath, SD.ImageFolder + @"\" + SD.DefaultBookImage);  System.IO.File.Copy(uploads, webRootPath + @"\" + SD.ImageFolder + @"\" + book.Title + ".png");  book.ImageUrl = @"\" + SD.ImageFolder + @"\" + book.Title + ".png";  }  \_context.Add(book);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  ViewData["AuthorID"] = new SelectList(\_context.Authors, "ID", "Name", book.AuthorID);  ViewData["CategoryID"] = new SelectList(\_context.Categories, "ID", "Name", book.CategoryID);  ViewData["PublisherID"] = new SelectList(\_context.Publishers, "ID", "Name", book.PublisherID);  return View(book);  }  // GET: Admin/Books/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var book = await \_context.Books.FindAsync(id);  if (book == null)  {  return NotFound();  }  ViewData["AuthorID"] = new SelectList(\_context.Authors, "ID", "Name", book.AuthorID);  ViewData["CategoryID"] = new SelectList(\_context.Categories, "ID", "Name", book.CategoryID);  ViewData["PublisherID"] = new SelectList(\_context.Publishers, "ID", "Name", book.PublisherID);  return View(book);  }  // POST: Admin/Books/Edit/5  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("BookID,Title,Summary,Quantities,CreateDate,ModifiedDate,NumberOfPages,Price,ImageUrl,IsPurchase,AuthorID,CategoryID,PublisherID")] Book book)  {  if (id != book.BookID)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  string webRootPath = \_hostingEnvironment.WebRootPath;  var files = HttpContext.Request.Form.Files;  if (files.Count != 0)  {  //Image has been uploaded  var uploads = Path.Combine(webRootPath, SD.ImageFolder);  var extension = Path.GetExtension(files[0].FileName);  using (var filestream = new FileStream(Path.Combine(uploads, book.Title + extension), FileMode.Create))  {  files[0].CopyTo(filestream);  }  book.ImageUrl = @"\" + SD.ImageFolder + @"\" + book.Title + extension;  }  else  {  //when user does not upload image  var uploads = Path.Combine(webRootPath, SD.ImageFolder + @"\" + SD.DefaultBookImage);  System.IO.File.Copy(uploads, webRootPath + @"\" + SD.ImageFolder + @"\" + book.Title + ".png");  book.ImageUrl = @"\" + SD.ImageFolder + @"\" + book.Title + ".png";  }  \_context.Update(book);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!BookExists(book.BookID))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  ViewData["AuthorID"] = new SelectList(\_context.Authors, "ID", "Name", book.AuthorID);  ViewData["CategoryID"] = new SelectList(\_context.Categories, "ID", "Name", book.CategoryID);  ViewData["PublisherID"] = new SelectList(\_context.Publishers, "ID", "Name", book.PublisherID);  return View(book);  }  // GET: Admin/Books/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var book = await \_context.Books  .Include(b => b.Author)  .Include(b => b.Category)  .Include(b=>b.Publisher)  .FirstOrDefaultAsync(m => m.BookID == id);  if (book == null)  {  return NotFound();  }  return View(book);  }  // POST: Admin/Books/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var book = await \_context.Books.FindAsync(id);  \_context.Books.Remove(book);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool BookExists(int id)  {  return \_context.Books.Any(e => e.BookID == id);  }  }  } | | | |

|  |  |
| --- | --- |
| **View of Manage Book controller** | |
| Index |  |

|  |  |
| --- | --- |
| AddBook |  |

|  |  |
| --- | --- |
| EditBook |  |
| DeleteBook |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **AdminUsersController.cs** | | | |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session of admin is valid or not, then call the view of account to display. |
| RegisterAdmin | HttpGET | Used to call the view of add a new account to display. |
| RegisterAdmin | HttpPOST | Add a new account into the database if it information have filled in the register form is valid. |
| EditAccount | HttpGET | Used to call the view of edit account with the username of account have contain in the database to display. |
| EditAccount | HttpPOST | Get the new information have change save to the database if it’s valid. |
| DeleteAccount | HttpGET | Used to delete an account out of database. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using IBook.Data;  using IBook.Models;  using IBook.Utility;  using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace IBook.Areas.Admin.Controllers  {  [Authorize(Roles = SD.SupperAdminEndUser)]  [Area("Admin")]  public class AdminUsersController : Controller  {  private readonly ApplicationDbContext \_context;  public AdminUsersController(ApplicationDbContext context)  {  \_context = context;  }  public IActionResult Index()  {  return View(\_context.ApplicationUser.ToList());  }  public async Task<IActionResult> Edit(string id)  {  if (id == null || id.Trim().Length == 0)  {  return NotFound();  }  var userFromDb = await \_context.ApplicationUser.FindAsync(id);  if (userFromDb == null)  {  return NotFound();  }  return View(userFromDb);  }  [HttpPost]  [ValidateAntiForgeryToken]  public IActionResult Edit(string id, ApplicationUser applicationUser)  {  if (id != applicationUser.Id)  {  return NotFound();  }  if (ModelState.IsValid)  {  ApplicationUser userFromDb = \_context.ApplicationUser.Where(b => b.Id == id).FirstOrDefault();  userFromDb.Name = applicationUser.Name;  userFromDb.PhoneNumber = applicationUser.PhoneNumber;  \_context.SaveChanges();  return RedirectToAction(nameof(Index));  }  return View(applicationUser);  }  public async Task<IActionResult> Delete(string id)  {  if (id == null || id.Trim().Length == 0)  {  return NotFound();  }  var userFromDb = await \_context.ApplicationUser.FindAsync(id);  if (userFromDb == null)  {  return NotFound();  }  return View(userFromDb);  }  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public IActionResult DeletePOST(string id, ApplicationUser applicationUser)  {  ApplicationUser userFromDb = \_context.ApplicationUser.Where(b => b.Id == id).FirstOrDefault();  userFromDb.LockoutEnd = DateTime.Now.AddYears(1000);  \_context.SaveChanges();  return RedirectToAction(nameof(Index));  }  }  } | | | |

|  |  |
| --- | --- |
| **View of Admin Users Controller** | |
| Index |  |
| RegisterAdmin |  |
| EditAccount |  |
| DeleteAccount |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **AuthorController.cs** | |  |  |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session of admin is valid or not, then call the view of author to display. |
| AddAuthor | HttpGET | Used to call the view of add a new author |

|  |  |  |  |
| --- | --- | --- | --- |
|  | AddAuthor | HttpPOST | Add a new author’s information into the database if it is a valid information have filled in the create form. |
| EditAuthor | HttpGET | Used to call the view of edit author with the author’s id have contain in the database to display. |
| EditAuthor | HttpPOST | Get the new information save to the database if it’s valid information. |
| DeleteAuthor | HttpGET | Used to delete an author out of database. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using IBook.Data;  using IBook.Models;  using Microsoft.AspNetCore.Authorization;  using IBook.Utility;  namespace IBook.Areas.Admin.Controllers  {  [Authorize(Roles = SD.SupperAdminEndUser)]  [Area("Admin")]  public class AuthorsController : Controller  {  private readonly ApplicationDbContext \_context;  public AuthorsController(ApplicationDbContext context)  {  \_context = context;  }  // GET: Admin/Authors  public async Task<IActionResult> Index()  {  return View(await \_context.Authors.ToListAsync());  }  // GET: Admin/Authors/Details/5  public async Task<IActionResult> Details(int? id)  {  if (id == null)  {  return NotFound();  }  var author = await \_context.Authors  .FirstOrDefaultAsync(m => m.ID == id);  if (author == null)  {  return NotFound();  }  return View(author);  }  // GET: Admin/Authors/Create  public IActionResult Create()  {  return View();  }  // POST: Admin/Authors/Create  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Create([Bind("ID,Name,History")] Author author)  {  if (ModelState.IsValid)  {  \_context.Add(author);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  return View(author);  }  // GET: Admin/Authors/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var author = await \_context.Authors.FindAsync(id);  if (author == null)  {  return NotFound();  }  return View(author);  }  // POST: Admin/Authors/Edit/5  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("ID,Name,History")] Author author)  {  if (id != author.ID)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(author);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!AuthorExists(author.ID))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  return View(author);  }  // GET: Admin/Authors/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var author = await \_context.Authors  .FirstOrDefaultAsync(m => m.ID == id);  if (author == null)  {  return NotFound();  }  return View(author);  }  // POST: Admin/Authors/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var author = await \_context.Authors.FindAsync(id);  \_context.Authors.Remove(author);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool AuthorExists(int id)  {  return \_context.Authors.Any(e => e.ID == id);  }  }  } | | | |

|  |  |
| --- | --- |
| **View of Author controller** | |
| Index |  |
| AddAuthor |  |
| EditAuthor |  |
| DeleteAuthor |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderBooksController.cs** | | | |
| **Explanation** | **ActionResult** | **Method** | **Describe** |
| Index | HttpGET | Used to check the session of admin is valid or not, then call the view of item order to display. |
| DeleteOrder | HttpGET | Used to delete an order record out of database. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Rendering;  using Microsoft.EntityFrameworkCore;  using IBook.Data;  using IBook.Models;  using Microsoft.AspNetCore.Authorization;  using IBook.Utility;  namespace IBook.Areas.Admin.Controllers  {  [Authorize]  [Area("Admin")]  public class OrderBooksController : Controller  {  private readonly ApplicationDbContext \_context;  public OrderBooksController(ApplicationDbContext context)  {  \_context = context;  }  // GET: Admin/OrderBooks  public async Task<IActionResult> Index()  {  return View(await \_context.OrderBooks.ToListAsync());  }  // GET: Admin/OrderBooks/Edit/5  public async Task<IActionResult> Edit(int? id)  {  if (id == null)  {  return NotFound();  }  var orderBook = await \_context.OrderBooks.FindAsync(id);  if (orderBook == null)  {  return NotFound();  }  return View(orderBook);  }  // POST: Admin/OrderBooks/Edit/5  // To protect from overposting attacks, please enable the specific properties you want to bind to, for  // more details see http://go.microsoft.com/fwlink/?LinkId=317598.  [HttpPost]  [ValidateAntiForgeryToken]  public async Task<IActionResult> Edit(int id, [Bind("OrderID,OrderDate,CustomerName,CustomerAddress,CustomerPhone,isConfirmed")] OrderBook orderBook)  {  if (id != orderBook.OrderID)  {  return NotFound();  }  if (ModelState.IsValid)  {  try  {  \_context.Update(orderBook);  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException)  {  if (!OrderBookExists(orderBook.OrderID))  {  return NotFound();  }  else  {  throw;  }  }  return RedirectToAction(nameof(Index));  }  return View(orderBook);  }  // GET: Admin/OrderBooks/Delete/5  public async Task<IActionResult> Delete(int? id)  {  if (id == null)  {  return NotFound();  }  var orderBook = await \_context.OrderBooks  .FirstOrDefaultAsync(m => m.OrderID == id);  if (orderBook == null)  {  return NotFound();  }  return View(orderBook);  }  // POST: Admin/OrderBooks/Delete/5  [HttpPost, ActionName("Delete")]  [ValidateAntiForgeryToken]  public async Task<IActionResult> DeleteConfirmed(int id)  {  var orderBook = await \_context.OrderBooks.FindAsync(id);  \_context.OrderBooks.Remove(orderBook);  await \_context.SaveChangesAsync();  return RedirectToAction(nameof(Index));  }  private bool OrderBookExists(int id)  {  return \_context.OrderBooks.Any(e => e.OrderID == id);  }  }  } | | | |
| **View of order books controller** | | | |
| Index |  | | |
| DeleteOrder |  | | |

# Screenshots of using GitHub

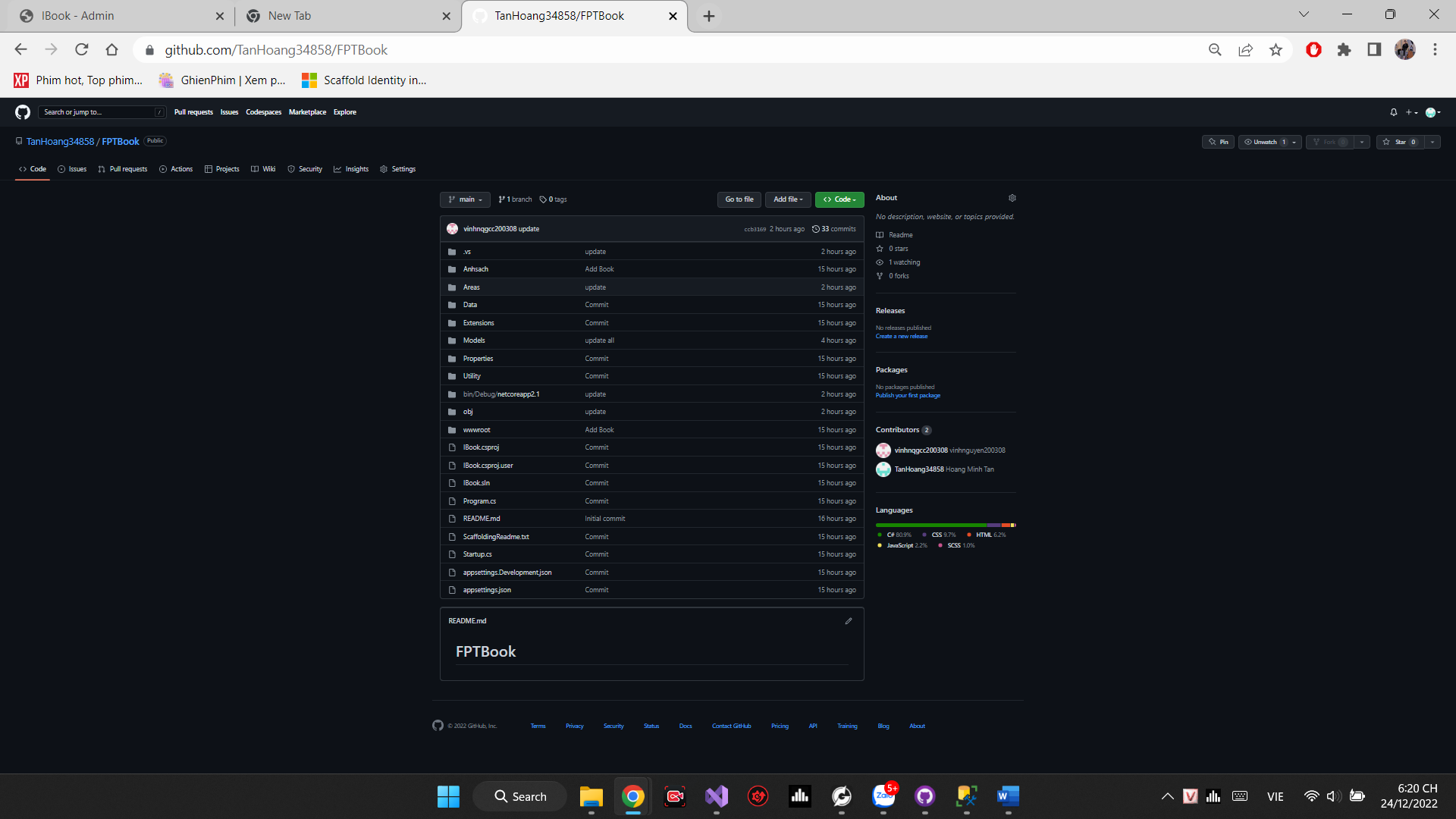


Figure 7: GitHub repository

Link GitHub: <https://github.com/TanHoang34858/FPTBook.git>

* **Account Admin** User

Email: admin@gmail.com

Password:

Admin123\*

* **Account User:** User

Email:

123@gmail.com

Password:

Tan123\*

# Chapter 3 – Application Evaluation (P6 – M5)

# Comparation initial requirements

Table 2: review and comparation with initial

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Use case | Describe | Mockup | Result | Complete(%) | Status |
| 1. | Home | Home page of web site |  |  | 100% | Finish |
| 2. | Log-in | Log-in to access and  use function of website. |  |  | 100% | Change Signin to Login |
| 3. | Register | Register new account to use website |  |  | 100% | Finish |
| 4. | Edit information | Edit user’s informatio  n |  |  | 100% | Finish |
| 5. | User page | The home page of user after login |  |  | 100% | Finish |
| 6. | Book detail | Book detail page use to view all information of book |  |  | 100% | Finish |
| 7. | Cart | Cart page display all book have add to cart |  |  | 100% | Finish |
| 8. | Order | Order page to confirm  Before make a payment. |  |  | 100% | Finish |
| 9. | Contact | Help page, display all the help  topic during using web. |  |  | 100% | Change Help to Contact |
| 10. | Admin home | Home page of admin  after log-in successful. |  |  | 100% | Change Store owner to Admin |
| 11. | Book managerment | Display all the book of store, and relative information. |  |  | 100% | Finish |
| 12. | Add book | Add new book |  |  | 100% | Finish |
| 13. | Edit book | Edit information of book |  |  | 100% | Finish |
| 14. | Category management | Display all category of store, and relative information |  |  | 100% | Finish |
| 15. | Add category | Add new category |  |  | 100% | Finish |
| 16. | Edit category | Edit information of category |  |  | 100% | Finish |
| 17. | Oder management | Display all order of store, and relative information |  |  | 100% | Change Order Recording to Order Management |
| 18. | Author management | Display all author of store, and relative information |  |  | 100% | Finish |
| 19. | Add author | Add new author |  |  | 100% | Finish |
| 20. | Edit author | Edit information of author |  |  | 100% | Finish |
| 21. | User management | Display all account have register |  |  | 100% | Change Customer to User management |
| 22. | Add user | Add new user |  |  | 100% | Change Customer to User management |
| 23. | Edit user | Edit user information |  |  | 100% | Change Customer to User management |
| 24. | Delete | Confirm delete befor delete sumit |  |  | 100% | Change Customer to User management |

# References:

Microsoft, 2022. *Microsoft.* [Online]

Available at: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022 [Accessed 3 3 2022].

byte, C., 2021. *Coding byte.* [Online]

Available at: https://www.codingbytes.com/difference-between-html-and-html5-and-between-css-andcss3/

[Accessed 3 3 2022].

Microsoft, 2022. *Microsoft.* [Online]

Available at: https://dotnet.microsoft.com/en-us/apps/aspnet [Accessed 3 3 2022].

Glossary, D., 2022. *MDN Web Docs Glossary.* [Online]

Available at: https://developer.mozilla.org/en-US/docs/Glossary/MVC [Accessed 3 3 2022].

SQLservertutorial, 2022. *SQLservertutorial.* [Online]

Available at: https://www.sqlservertutorial.net [Accessed 3 3 2022].