

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Кейдалюк Владислав Михайлович,
здобувач групи РФ-211

КУРСОВИЙ ПРОЕКТ

Розробка та тестування веб-застосування “Книжний магазин” з
використанням технології EF Core
та різними правами доступу до системи

Спеціальність:
122 Комп’ютерні науки

Спеціалізація, освітня програма:
Комп’ютерні науки та інформаційна безпека

Керівник:
Трифорова Катерина Олексіївна,
ст. викладач

Одеса – 2023

ЗМІСТ

ВСТУП.....	4
1 ПЛАНУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	5
1.1 Проектування багаторівневої інфраструктури	5
1.2 Вибір сховища даних.....	6
1.3 Проектування бази даних.....	6
1.4 Проектування рівня доступу до даних	8
1.5 Проектування рівня бізнес-логіки	8
1.6 Збереження рядків з'єднання та інших параметрів.....	9
1.7 Проектування інтерфейсу користувача.....	9
1.8 Потреби в безпеці	12
2 СИСТЕМА ЧЛЕНСТВА ТА СИСТЕМА ПРОФІЛІВ КОРИСТУВАЧІВ	13
2.1 Проектування інфраструктури членства	13
2.2 Проектування бази даних.....	14
2.3 Реалізація конфігураційного модуля	14
2.4 Реалізація валідації даних	15
2.5 Реалізація автентифікації користувача.....	15
2.6 Реалізація авторизації користувача	15
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	16
3.1 Реалізація бази даних	16
3.2 Реалізація конфігураційного модуля	16
3.3 Реалізація рівня доступу до даних.....	18
3.4 Реалізація рівня бізнес-логіки.....	26
3.5 Реалізація інтерфейсу користувача	32
4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	46

4.1 Автоматизоване тестування.....	46
4.2 Функціональне тестування.....	47
4.3 Шаблон проектування Page Object Model.....	48
4.4 Класи сторінок.....	49
4.5 Тести та тестування.....	52
ВИСНОВКИ	56
ПЕРЕЛІК ПОСИЛАНЬ	57
Додаток А Код файлу appsettings.json та Program.cs.....	58
Додаток Б Код усіх контролерів.....	59
Додаток В Код усіх класів-моделей та моделей представлень.....	71
Додаток Г Всі файли з папки Data	75
Додаток Д Код усіх представлень	85
Додаток Е Код усіх файлів з проекту з тестами	104

ВСТУП

Дана робота виконана з дисципліни «Веб-програмування та веб-дизайн».

Необхідно було розробити повноцінний веб-додаток на ASP.Net Core з використанням шаблону MVC, користувацькою та серверною частиною на певну тематику. Була обрана тема - «Книжний магазин».

Додаток за умовою завдання мав містити базу даних з таблицями, які б містили різні типи зв'язків між собою.

Також має бути розподіл користувачів за ролями, аби забезпечити різний рівень доступу та можливості у системі відповідно до ролі. Ще має бути реалізована аутентифікація та авторизація у системі. Для даного переліку вимог добре підходить інфраструктура Identity, яку і було застосовано в застосунку.

Згідно обраної теми, задля повноцінної реалізації магазину, було вирішено додати систему оплати Stripe.

Програмне забезпечення має наступні можливості:

- система акаунтів користувачів, адміністратор має можливість створювати нових користувачів відповідних ролей а також блокувати доступ для певних користувачів за необхідністю,
- додавання, редагування, видалення категорій, книг, компаній,
- реалізовано кошик, систему оплати,
- реалізовано розділ для керування замовленнями, а саме оновлення певних даних, зміна статусу замовлення, або скасування замовлення (при скасуванні замовлення гроші будуть автоматично повернені покупцеві)

Додатково задля забезпечення стабільності роботи застосунку були написані тести для автоматизованого тестування з використанням NUnit та Selenium WebDriver.

1 ПЛАНУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Проектування багаторівневої інфраструктури

Для створення застосунку було обрано наступну багаторівневу інфраструктуру (рис. 1.1)

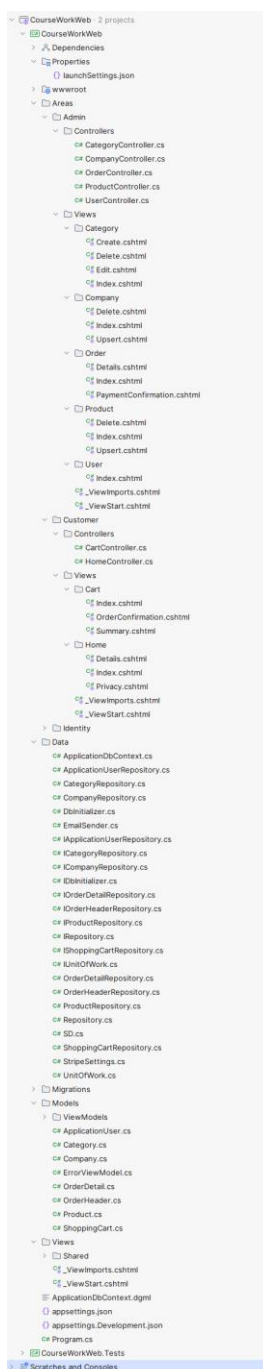


Рисунок 1.1 – Загальна структура застосунку

Тут можна побачити “Presentation Layer” – рівень представлення, бізнес логіку та загалом інфраструктуру. Було використано шаблон MVC (Model-View-

Controller).

1.2 Вибір сховища даних

У якості сховища даних було обрано MySQL Server. Вона легка у користуванні, можна взаємодіяти як через CLI так і через графічний інтерфейс (MySQL Workbench).

MySQL Server відома своєю високою надійністю та продуктивністю. Вона забезпечує ефективне зберігання та швидкий доступ до даних, що особливо важливо для веб-додатків з великою кількістю користувачів.

MySQL має велику спільноту користувачів та активний екосистем. Це означає, що можна легко знайти документацію, форуми та різноманітні ресурси для вирішення проблем та отримання порад щодо розробки.

MySQL є кросплатформеною СУБД і підтримується на різних операційних системах, таких як Windows, Linux та macOS. Це дозволяє легко переносити додаток між різними середовищами.

1.3 Проектування бази даних

Загальну структуру бази даних можна побачити на рисунку 1.2.

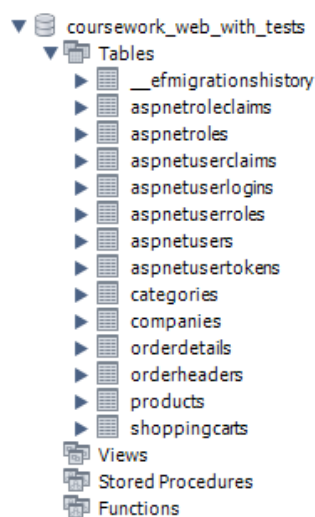


Рисунок 1.2 – Структура бази даних

У якості сховища даних було обрано СУБД MySQL.

В базі даних буде зв'язок 1 до багатьох між таблицею *aspnetusers* та *companies*.

Що стосується наступних таблиць: між *aspnetusers* та *shoppingcarts*, *products* та *shoppingcarts*, *products* та *orderdetails*, *categories* та *products*, *aspnetusers* та *orderheaders*, між усіма цими таблицями також буде зв'язок один до багатьох.

Більш детально це можна побачити на розширеній діаграмі «Сутність-Зв'язок», що показана на рисунку 1.3.

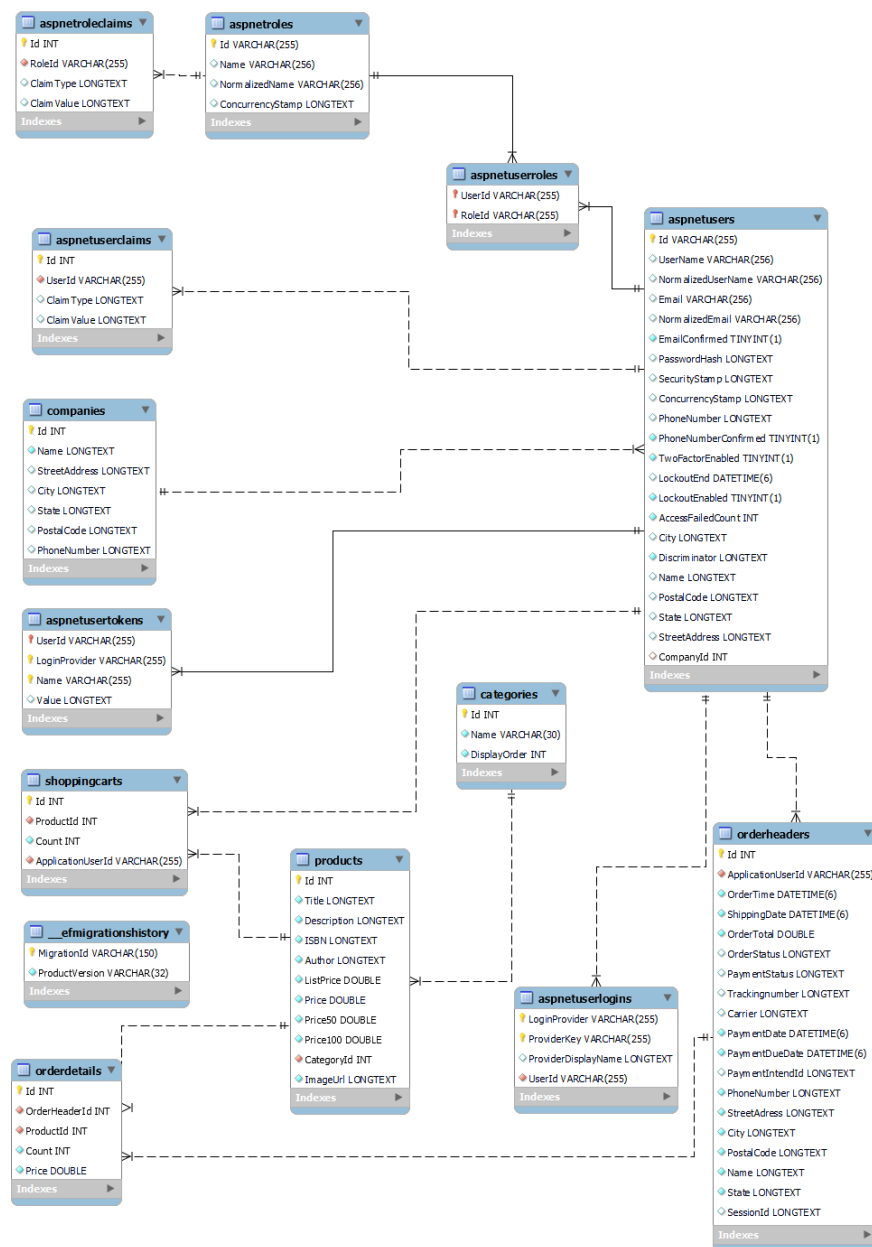


Рисунок 1.3 – EER діаграма

Дана діаграма нам чітко показує взаємозв'язки між таблицями, а також видно усі атрибути та ключі, як PK (Primary Key), так і FK (Foreign Key).

1.4 Проектування рівня доступу до даних

Розглянемо так званий DAL (Data Access Layer), для цього було використано Entity Framework.

Для аутентифікації та авторизації буде задіяно Identity.

Згідно теми проєкту, було вирішено створити 4 ролі :

- Адміністратор (Admin) – користувачі з даною роллю будуть мати максимальні права доступу, надана можливість створення нових користувачів будь-якої ролі або блокування/розблокування існуючих, додавання/редагування та видалення категорій, книг та компаній.
- Робітник (Employee) – користувачі з даною роллю мають менші права доступу. Люди з цією роллю будуть обробляти замовлення, лише до цього блоку вони мають доступ.
- Компанія (Company) – користувачі з даною роллю мають право оплатити замовлення у 30 денний період після відправлення.
- Покупець (Customer) – користувачі з даною роллю мають право на купівлю товару (книг) та миттєвої оплати замовлення, лише після оплати замовлення поступить в обробку.

1.5 Проектування рівня бізнес-логіки

Розглянемо так званий BLL (Business Logic Layer), він в нас отримує дані від DAL (Data Access Layer) та передає рівню UI (User Interface). У даному випадку, згідно шаблону Model-View-Controller, це наші моделі, а саме: *ApplicationUser*, *Category*, *Company*, *OrderDetail*, *OrderHeader*, *Product*, *ShoppingCart*.

Звісно необхідно згадати ViewModels, вони дозволяють нам передати у представлення одразу декілька об'єктів різних типів, а саме:

- OrderVM – *OrderHeader* та колекцію, що складається з *OrderDetail*,
- ProductVM – *Product* та колекцію *SelectListItem*, що складається з категорій,
- ShoppingCartVM – *OrderHeader* та колекцію, що складається з *ShoppingCart*.

1.6 Збереження рядків з'єднання та інших параметрів

Для збереження рядків з'єднання використовується файл `appsettings.json`, що є нормальною практикою. Побачити це можна на рисунку 1.4.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
    "Server=localhost;Database=coursetwork_web_with_tests;User=root;Password=123"
  },
  "Stripe": {
    "PublicKey": "pk_test_sdfws369nY46Fdfhdnfnfujeqewtgehehe",
    "Secretkey": "sk_test_sdfws369dfwproetergerwtyr33jcx"
  }
}
```

Рисунок 1.4 – Код файлу `appsettings.json`

Також туди було додано публічний та приватний ключ для взаємодії з платіжною системою (приватна інформація була замінена на фальшиву).

1.7 Проектування інтерфейсу користувача

Інтерфейс користувача розділений по зонам доступу та по відношенню до певних моделей. Розглянемо все покроково, побачити список представлень можна у загальній структурі проекту на рисунку 1.1.

Почнемо з представлень, що відносяться до категорій (папка Category):

- Index.cshtml – дане представлення надає можливість побачити список наявних категорій та кнопки редагування, видалення, або створення нової.
- Create.cshtml – надає можливість створити нову категорію.
- Edit.cshtml – надає можливість змінити та оновити дані обраної категорії.
- Delete.cshtml – надає можливість видалити обрану категорію.

Далі розглянемо представлення, що відносяться до компаній (папка Company):

- Index.cshtml – дане представлення надає можливість побачити список наявних компаній та кнопки редагування, видалення, або створення нової.
- Upsert.cshtml – надає можливість створити нову компанію, або редагувати інформацію про обрану компанію.
- Delete.cshtml – надає можливість видалити обрану компанію.

Далі розглянемо представлення, що відносяться до замовлень (папка Order):

- Index.cshtml – дане представлення надає можливість побачити список наявних замовлень та кнопку, що відкриє сторінку з деталями.
- Details.cshtml – надає можливість подивитись більш детальну інформацію про замовлення та змінювати його статус, або відхилити.
- PaymentConfirmation.cshtml – сторінка що відображується після успішної оплати.

Далі розглянемо представлення, що відносяться до товарів (папка Product):

- Index.cshtml – дане представлення надає можливість побачити список наявних товарів та кнопки редагування, видалення, або створення нового.
- Upsert.cshtml – надає можливість створити новий товар, або редагувати інформацію про обраний.
- Delete.cshtml – надає можливість видалити обраний товар.

Далі розглянемо представлення, що відносяться до користувачів (папка

User):

- *Index.cshtml* – дане представлення надає можливість побачити список наявних користувачів та заблокувати, або розблокувати обраного користувача.

Далі розглянемо представлення, що відносяться до кошиків (папка *Cart*):

- *Index.cshtml* – дане представлення надає можливість побачити список наявних замовлень у кошику та кнопки, що дозволять перейти до сторінки *Summary.cshtml*, а також змінити кількість обраних товарів, або прибрати з кошика.
- *Summary.cshtml* – надає можливість вписати коректні дані для доставки замовлення, побачити ітогову суму замовлення та перейти на сторінку оплати, після чого відбудеться переадресація на сторінку *PaymentConfirmation.cshtml*.
- *OrderConfirmation.cshtml* – сторінка що відображується після успішного створення замовлення.

Наостанок розглянемо представлення, що відносяться до головної сторінки (папка *Home*):

- *Index.cshtml* – дане представлення надає можливість побачити список наявних товарів у системі.
- *Details.cshml* – дане представлення надає можливість побачити більш детальну інформацію про товар та обрати відповідну кількість. Наявна кнопка додавання товару до кошика.

Взаємозв'язок між представленнями можна побачити на мапі сайту (рис. 1.7.1).

Якщо її розібрати більш детально, то побачимо, що без авторизації користувач має право лише на перегляд продуктів (книжок).

Для будь-яких інших дій необхідна реєстрація на сайті, або ж авторизація у вже попередньо створений акаунт.

1.8 Потреби в безпеці

Про безпеку можна сказати те, що в нас застосовується інфраструктура Identity, яка містить в собі все необхідне. Також в нас відбувається підключення за допомогою протоколу https, що забезпечує передавання паролів у захищеному вигляді і наявний SSL сертифікат.

Впровадження Identity дозволило доволі легко і швидко реалізувати різний доступ до різних сторінок та функцій сайту. У Identity вже є вбудовані системи авторизації, аутентифікації.

Також вона дозволила дуже легко реалізувати блокування користувача в системі. Треба ще сказати, що всі паролі користувачів зберігаються у базі даних в хешованому вигляді і звісно id кожного користувача унікальний.

Застосування Identity дозволило скоротити час на створення користувацького інтерфейсу, адже Identity постачає сторінки реєстрації, авторизацію і багато яких ще.

2 СИСТЕМА ЧЛЕНСТВА ТА СИСТЕМА ПРОФІЛІВ КОРИСТУВАЧІВ

2.1 Проектування інфраструктури членства

Система членства використовує Identity Framework для налаштування аутентифікації та авторизації користувачів. В файлі *Program.cs* визначено необхідні служби та налаштування (рис. 2.1).

```
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>().AddDefaultTokenProviders();

builder.Services.ConfigureApplicationCookie(options =>
{
    options.LoginPath = $"/Identity/Account/Login";
    options.LogoutPath = $"/Identity/Account/Logout";
    options.AccessDeniedPath = $"/Identity/Account/AccessDenied";
});

builder.Services.AddRazorPages();
```

Рисунок 2.1 – Налаштування Identity в Program.cs

Були використані стандартні моделі користувачів (*ApplicationUser*), які розширено для включення додаткових полів.

Система використовує ролі для групування користувачів та надання прав доступу. Додатково використовуються політики для точного контролю доступу. Приклад можна побачити на рисунку 2.2, там визначається зона дії та для якої ролі буде доступний доступ до відповідного контролера.

```
[Area("Admin")]
[Authorize(Roles = SD.Role_Admin)]
public class UserController : Controller
{
```

Рисунок 2.2 – Фрагмент коду, що вказує обмеження доступу до контролера

Процес аутентифікації користувачів здійснюється за допомогою вбудованих механізмів Identity.

2.2 Проектування бази даних

Подивитись EER діаграму можна на рисунку 1.1. Там же можна побачити таблиці, що необхідні для системи членства та профілів користувачів, а саме: *aspnetusers*, *aspnetuserroles*, *aspnetroles*, *aspnetroleclaims*.

2.3 Реалізація конфігураційного модуля

Реалізація була представлена на рисунку 2.1. Розглянемо детальніше.

- `AddIdentity<IdentityUser, IdentityRole>()`: Додає служби Identity з використанням типів IdentityUser та IdentityRole для користувачів та ролей відповідно.
- `AddEntityFrameworkStores<ApplicationDbContext>()`: Конфігурує Entity Framework як зберігальний механізм для Identity, використовуючи ApplicationDbContext.
- `AddDefaultTokenProviders()`: Додає стандартні постачальники токенів для функціоналу, такого як підтвердження електронної пошти та відновлення паролю.

Конфігурація Cookies для аутентифікації:

- `ConfigureApplicationCookie(options => { ... })`: Налаштовує параметри аутентифікаційного печива для контролю процесу входу та виходу.
- `options.LoginPath`: Вказує шлях, на який буде переадресовано користувача у випадку необхідності аутентифікації.
- `options.LogoutPath`: Вказує шлях для виходу з системи.
- `options.AccessDeniedPath`: Вказує шлях, який буде використовуватися, коли у користувача недостатньо прав для доступу до певного ресурсу.

2.4 Реалізація валідації даних

Використовуючи атрибути валідації, такі як Required забезпечується базовий рівень валідації на рівні моделі.

Використовуючи бібліотеки, такі як jQuery Validation і вбудований валідатор браузера, було реалізовано клієнтську валідацію для перевірки коректності введених даних перед відправкою на сервер. Для складніших випадків валідації, які не можна вирішити на рівні моделей, було використано механізми валідації на рівні контролерів.

2.5 Реалізація автентифікації користувача

За допомогою системи Identity було налаштовано процес автентифікації користувача, забезпечуючи можливість входу з використанням електронної пошти та паролю. Система Identity використовує куки і токени для зберігання інформації про автентифіковану сесію, токени зберігаються у таблиці *aspnetusertokens*, побачити можна на рисунку 1.3.

2.6 Реалізація авторизації користувача

Було визначено ролі (Customer, Company, Employee, Admin) та політики, які визначають права доступу користувачів. Було використано атрибути авторизації для визначення прав доступу на рівні контролерів та певних дій.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація бази даних

Реалізація бази даних була реалізована через інфраструктуру Entity Framework та механізм міграцій. Рядку під'єднання збережені в appsettings.json (рис. 1.4). Для заповнення бази даних першочерговими даними було задіяно контекст, для створення категорій, компаній та товарів (шукати в додатках). Для створення користувачів з відповідними ролями та додавання самих ролей використовується клас DbInitializer, код зображено на рисунку 3.6.3. Моделі описані в підрозділі 3.4.

3.2 Реалізація конфігураційного модуля

В даному випадку конфігураційним модулем є файл Program.cs., код знаходиться на рисунку 3.1.

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySQL(builder.Configuration.GetConnectionString("DefaultConnection")));
builder.Services.Configure<StripeSettings>(builder.Configuration.GetSection("Stripe"));
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>().AddDefaultTokenProviders();
builder.Services.ConfigureApplicationCookie(options =>
{
    options.LoginPath = $"/Identity/Account/Login";
    options.LogoutPath = $"/Identity/Account/Logout";
    options.AccessDeniedPath = $"/Identity/Account/AccessDenied";
});
builder.Services.AddRazorPages();

builder.Services.AddScoped<IDbInitializer, DbInitializer>();
builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();
builder.Services.AddScoped<IEmailSender, EmailSender>();
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
StripeConfiguration.ApiKey =
builder.Configuration.GetSection("Stripe:SecretKey").Get<string>();
app.UseRouting();
```



```

app.UseAuthentication();
app.UseAuthorization();
SeedDataBase();
app.MapRazorPages();
app.MapControllerRoute(
    name: "default",
    pattern: "{area=Customer}/{controller=Home}/{action=Index}/{id?}");
app.Run();
void SeedDataBase()
{
    using (var scope = app.Services.CreateScope())
    {
        var dbInitializer = scope.ServiceProvider.GetRequiredService<IDbInitializer>();
        dbInitializer.Initialize();
    }
}

```

Рисунок 3.1 – Код файлу Program.cs

Створення веб-застосунку: Використовується для створення екземпляра `WebApplicationBuilder`, який дозволяє налаштовувати та конфігурувати додаток.

Додавання сервісів: Додаються основні сервіси, такі як контролери, `Entity Framework` для роботи з базою даних, конфігураційні налаштування `Stripe`, система ідентифікації користувачів тощо.

Додавання областей та сервісів для роботи з базою даних: Додаються сервіси, які використовуються для роботи з базою даних.

Створення та налаштування застосунку: Створюється екземпляр застосунку.

Налаштування середовища виконання: Якщо додаток не працює в середовищі розробки, налаштовується обробник помилок та використовується `HTTPS`.

Налаштування протоколів та статичних файлів: Налаштовуються перенаправлення на `HTTPS` та використання статичних файлів.

Налаштування середовища маршрутизації та автентифікації:

Налаштовуються середовища маршрутизації, автентифікації та авторизації.

Виклик ініціалізації бази даних: Здійснюється виклик функції ініціалізації бази даних.

Мапування сторінок `Razor` та контролерів: Налаштовуються маршрути для сторінок `Razor` та контролерів.

Запуск застосунку: Додаток починає виконуватись.

Функція ініціалізації бази даних: Викликається ініціалізація бази даних через

створення області служб та отримання необхідного сервісу.

3.3 Реалізація рівня доступу до даних

Створена зручна структура (рис.3.2) для того, щоб розуміти області доступу та відповідальності. Код усіх контролерів надано в додатках.

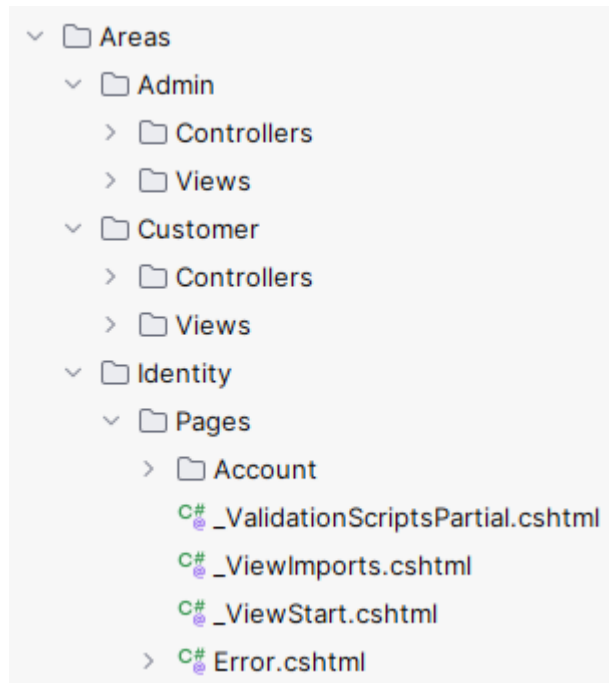


Рисунок 3.2- Структура розподілу по доступу

Розглянемо структуру проєкту більш детально. В нас є папка *Areas* , в якій буде розділення по доступу.

В папці *Admin* будуть міститись контролери та представлення лише тих сторінок та функцій, що доступні користувачеві з роллю адміністратора.

Далі в нас є папка *Customer*, що відповідно відповідає за область можливостей покупця.

Можна побачити папку *Identity*, якраз в ній містяться усі представлення та необхідні функції для реєстрації, авторизації і багато чого ще.

Далі в нас йде папка *Data* . В ній зберігаються моделі, контекст, інтерфейси репозиторіїв та самі репозиторії. Тобто було використано шаблон Repository.

Також там є інтерфейс та клас *UnitOfWork* («Робоча одиниця»), зображено на рисунку 3.3.

Шаблон "Робоча одиниця" використовується для управління транзакціями і гарантує, що кілька операцій розглядаються як одна логічна одиниця.

Він забезпечує спосіб групування операцій бази даних разом, гарантуючи, що вони будуть виконані успішно або не виконані як єдине ціле.

Ключовим принципом шаблону "Робоча одиниця" є підтримка узгодженості та цілісності даних шляхом скоординованого внесення або відкату змін.

Розглянемо код інтерфейсу робочої одиниці (рис. 3.3).

```
public interface IUnitOfWork
{
    ICategoryRepository Category { get; }
    IProductRepository Product { get; }
    ICompanyRepository Company { get; }
    IShoppingCartRepository ShoppingCart { get; }
    IApplicationUserRepository ApplicationUser { get; }
    IOrderDetailRepository OrderDetail { get; }
    IOrderHeaderRepository OrderHeader { get; }

    void Save();
}
```

Рисунок 3.3 – код інтерфейсу UnitOfWork

Як ми бачимо він в собі містить усі інтерфейси інших репозиторіїв для взаємодії з ними. Метод *Save* викликається для збереження всіх змін в базі даних.

Такий підхід сприяє зменшенню кількості звернень до бази даних та забезпеченню атомарності змін в контексті бази даних.

Тепер розглянемо код класу робочої одиниці на рисунку 3.4.

Цей клас працює з репозиторіями різних об'єктів домену (наприклад, категорії, товари, користувачі тощо) в межах однієї транзакції.

Коли викликається метод *Save*, всі зміни, здійснені через репозиторії, будуть збережені в базі даних.

Що стосується додаткових класів, то тут необхідно виділити клас *SD* (скорочення від *StaticData*) та *DbInitializer*.

```

    public class UnitOfWork : IUnitOfWork
    {
        private ApplicationDbContext _context;
        public ICategoryRepository Category { get; private set; }
        public ICompanyRepository Company { get; private set; }
        public IProductRepository Product { get; private set; }
        public IShoppingCartRepository ShoppingCart { get; private set; }
        public IApplicationUserRepository ApplicationUser { get; private set; }

        public IOrderHeaderRepository OrderHeader { get; private set; }
        public IOrderDetailRepository OrderDetail { get; private set; }

        public UnitOfWork(ApplicationDbContext context)
        {
            _context = context;
            ApplicationUser = new ApplicationUserRepository(_context);
            ShoppingCart = new ShoppingCartRepository(_context);
            Category = new CategoryRepository(_context);
            Product = new ProductRepository(_context);
            Company = new CompanyRepository(_context);
            OrderHeader = new OrderHeaderRepository(_context);
            OrderDetail = new OrderDetailRepository(_context);
        }
        public void Save()
        {
            _context.SaveChanges();
        }
    }
}

```

Рисунок 3.4 – код класу робочої одиниці

Клас *SD* містить в собі константи для ролей користувачів та різних статусів, пов'язаних з процесом обробки замовлень та оплати.

Клас *DbInitializer* використовується для ініціалізації бази даних під час запуску додатка. Код знаходиться в додатках. Також є інтерфейс *IDbInitializer*, що містить лише оголошення одного методу – *Initialize*.

Розглянемо його основні елементи:

Конструктор приймає *UserManager<IdentityUser>*, *RoleManager<IdentityRole>* та *ApplicationDbContext*. Ці параметри використовуються для взаємодії з користувачами, ролями та базою даних.

Метод *Initialize* викликається для виконання ініціалізації бази даних. У блоку *try* перевіряється, чи є невикористані міграції. Якщо є, викликається *_db.Database.Migrate()* для автоматичного виконання міграцій та оновлення схеми бази даних.

Далі перевіряється наявність ролей. Якщо ролі відсутні, вони створюються.

Створюються чотири користувачі з різними ролями: адміністратор, клієнт, компанія та співробітник. Кожному користувачеві призначається відповідна роль.

Папка *Migrations* - це папка, що створюється автоматично при створенні міграцій.

Папка *Models* містить всередині себе відповідні класи об'єктів. Також всередині є папка *ViewModels*, що містить в собі моделі подання, які містять лише ті дані (представлені властивостями), які необхідно використовувати у поданні.

Спочатку розглянемо папку що відповідає за доступ для покупців (рис.3.5). Розглянемо більш детально контролер *CartController*.

Даний контролер відповідає за обробку запитів, пов'язаних з корзиною покупок в області "*Customer*".

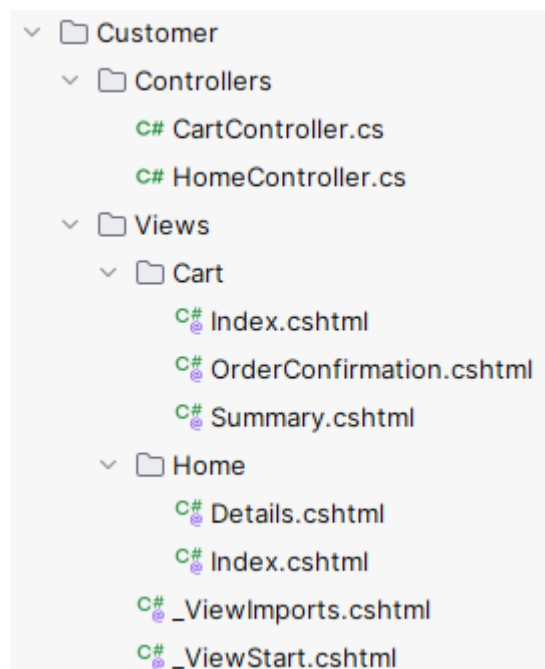


Рисунок 3.5 – Зона покупця, яка доступна користувачам з будь-якою роллю

[Area("Customer")] - вказує, що контролер належить області "Customer".

[Authorize] - вимагає, щоб користувач був автентифікованим для доступу до дій контролера. Тобто доступ до цього блоку може отримати лише зареєстрований у системі користувач.

_unitOfWork - використовується для взаємодії з репозиторіями та іншими

компонентами робочої одиниці.

[BindProperty] *public ShoppingCartVM ShoppingCartVM { get; set; }* - забезпечує автоматичний зв'язок властивостей моделі *ShoppingCartVM* з введеннями з форм на сторінках перегляду.

Метод дії *Index*:

Виводить сторінку корзини покупок та виводить інформацію про товари у корзині.

Метод дії *Summary*:

Виводить сторінку зі списком товарів у корзині та інформацією для здійснення покупки.

Метод дії *SummaryPOST*:

Обробляє POST-запит із сторінки *Summary* для оформлення замовлення та перенаправляє на сторінку підтвердження замовлення.

Метод дії *OrderConfirmation*:

Виводить сторінку підтвердження замовлення з ідентифікатором *id*.

Методи дії для зміни кількості товарів у корзині (*Plus*, *Minus*, *Remove*):

Виконують операції зі збільшенням, зменшенням та видаленням товарів у корзині.

Допоміжний метод *GetPriceBasedOnQuantity*:

Визначає ціну товару в залежності від кількості товару у корзині.

Тепер розглянемо папку що відповідає за доступ для адміністраторів та робітників, хоч і деякі функції доступні і для звичайних покупців (рис. 3.6).

Розглянемо більш детально контролер *CategoryController*.

[Area("Admin")] - вказує, що контролер належить області *Admin*.

[Authorize(Roles = SD.Role_Admin)] - обмежує доступ до контролера лише для користувачів з роллю *Admin*.

Поля та конструктор:

_unitOfWork - використовується для взаємодії з репозиторіями та іншими компонентами робочої одиниці.

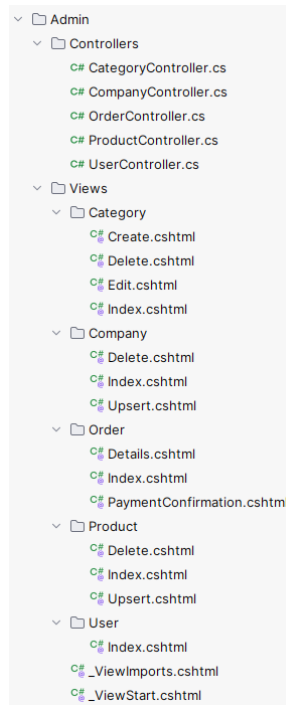


Рисунок 3.6— Зона адміністратора

Метод дії *Index*:

Виводить сторінку зі списком всіх категорій.

Методи дії *Create*:

Create() - виводить сторінку для створення нової категорії.

Create(Category category) - обробляє POST-запит для створення нової категорії та зберігає її в базу даних.

Методи дії *Edit*:

Edit(int? id) - виводить сторінку редагування категорії за вказаним ідентифікатором.

Edit(Category category) - обробляє POST-запит для збереження змін у редагованій категорії.

Методи дії *Delete*:

Delete(int? id) - виводить сторінку підтвердження видалення категорії.

DeletePOST(int? id) - обробляє POST-запит для видалення категорії та оновлює базу даних.

Розглянемо більш детально контролер *CompanyController*.

[Area("Admin")] - вказує, що контролер належить області *Admin*

[Authorize(Roles = SD.Role_Admin)] - обмежує доступ до контролера лише для користувачів з роллю *Admin*.

Поля та конструктор:

_unitOfWork - використовується для взаємодії з репозиторіями та іншими компонентами робочої одиниці.

Метод дії *Index*:

Виводить сторінку зі списком всіх компаній.

Методи дії *Upsert*:

Upsert(int? Id) - виводить сторінку для створення нової компанії або редагування існуючої, залежно від наявності ідентифікатора.

Upsert(Company company) - обробляє POST-запит для створення нової компанії або редагування існуючої та зберігає її в базу даних.

Метод дії *Delete*:

Delete(int? Id) - виводить сторінку підтвердження видалення компанії за вказаним ідентифікатором.

DeletePOST(int? Id) - обробляє POST-запит для видалення компанії та оновлює базу даних.

Розглянемо більш детально контролер *OrderController*.

[Area("Admin")] - вказує, що контролер належить області *Admin*.

[Authorize] - обмежує доступ до контролера лише для авторизованих користувачів.

Поля та конструктор:

_unitOfWork - використовується для взаємодії з репозиторіями та іншими компонентами робочої одиниці.

orderVM - об'єкт *ViewModel* для передачі даних між контролером та представленням.

Метод дії *Details*:

Details(int? orderId) - виводить сторінку деталей замовлення за вказаним ідентифікатором.

Метод дії *Index*:

Index - виводить список всіх замовлень. Для адміністратора або співробітника відображаються всі замовлення, в іншому випадку - тільки замовлення поточного користувача.

Методи дії для оновлення статусу замовлення (*UpdateOrderDetail*, *StartProcessing*, *ShipOrder*, *CancelOrder*):

Ці дії відповідають за оновлення деталей замовлення, початок обробки замовлення, відправку замовлення та його скасування.

Методи дій *Details_PAY_NOW* та *PaymentConfirmation*:

Details_PAY_NOW - готує дані для оплати через Stripe та перенаправляє на сторінку оплати.

PaymentConfirmation - обробляє підтвердження оплати через Stripe та оновлює статус замовлення.

Розглянемо більш детально контролер *ProductController*.

[Area("Admin")] - вказує, що контролер належить області *Admin*.

[Authorize(Roles = SD.Role_Admin)] - обмежує доступ до контролера лише для користувачів з роллю *Admin*.

Поля та конструктор:

_unitOfWork - використовується для взаємодії з репозиторіями та іншими компонентами робочої одиниці.

_webHostEnvironment - надає доступ до інформації про середовище веб-хостингу.

Метод дії *Index*:

Index - виводить список всіх продуктів з підключеною інформацією про категорію.

Методи дії *Upsert*:

Upsert(int? id) - виводить форму для створення нового продукту або редагування існуючого за ідентифікатором.

Upsert(ProductVM productVM, IFormFile? file) - обробляє дані форми для створення або редагування продукту, включаючи завантаження зображення.

Методи дії *Delete*:

Delete(int? id) - виводить підтвердження для видалення продукту за ідентифікатором.

DeletePOST(int? id) - видалляє продукт після підтвердження.

Розглянемо більш детально контролер *UserController*.

[Area("Admin")] - вказує, що контролер належить області *Admin*.

[Authorize(Roles = SD.Role_Admin)] - обмежує доступ до контролера лише для користувачів з роллю *Admin*.

Поля та конструктор:

_context - контекст бази даних для взаємодії з таблицею користувачів та інших таблиць.

Метод дії *Index*:

Index - виводить список користувачів та їхніх ролей.

Метод дії *LockUnlock*:

LockUnlock(string id) - блокує або розблоковує користувача за його ідентифікатором.

Використовує *LockoutEnd* для встановлення часу блокування користувача. Якщо *LockoutEnd* встановлено на значення в майбутньому, то користувач заблокований.

3.4 Реалізація рівня бізнес-логіки

Розглянемо класи-моделі, що містяться в застосунку (рис. 3.7). Увесь код наданий в додатках. Клас *ApplicationUser* представляє користувача з необхідними полями, такими як:

- ім'я
- адреса
- місто
- поштовий індекс
- id компанії (таким чином буде зв'язок «багато користувачів може відноситись до 1 компанії»).

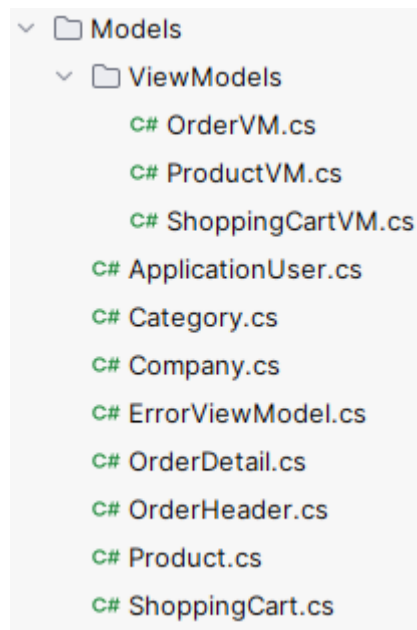


Рисунок 3.7 – Список моделей

Клас *ApplicationUser* представляє користувача з необхідними полями, такими як:

- ім'я
- адреса
- місто
- поштовий індекс
- id компанії (таким чином буде зв'язок «багато користувачів може відноситись до 1 компанії»)

Як можна було помітити тут нема поля *Id*, бо ми наслідуємо наш клас від класу *Identity*, а саме “*IdentityUser*”.

Клас *Category* представляє категорію книг з необхідними полями, такими як:

- id
- назва (поставимо обмеження на довжину у 30 символів)
- DisplayOrder (з допустимими межами від 1 до 100)
- Клас *Company* представляє компанію з необхідними полями, такими як:
- id
- ім'я

- адреса
- місто
- поштовий індекс
- номер телефону

Клас *OrderHeader* представляє основну інформацію про замовлення з необхідними полями, такими як:

- id
- id користувача (тим самим ми забезпечуємо зв'язок «багато замовлень може бути у одного покупця»)
- дата замовлення
- дата відправлення
- загальна ціна замовлення
- статус замовлення
- статус оплати
- трек-номер
- перевізник
- дата оплати
- дата, до якої необхідно виконати оплату (для компаній)
- id сесії (стосується платіжної системи)
- ідентифікатор призначення платежу (стосується платіжної системи)
- номер телефону
- адреса
- місто
- країна
- поштовий індекс
- ім'я

Клас *OrderDetail* представляє додаткову інформацію про замовлення з необхідними полями, такими як:

- id

- OrderHeaderId (таким чином буде зв'язок «багато OrderDetail до 1 OrderHeader »)
- ProductId (таким чином буде зв'язок «багато OrderDetail до 1 продукту (книги)»)
- Кількість
- Ціна

Клас *Product* представляє продукт (переважно книгу) з необхідними полями, такими як:

- id
- назва книги
- опис
- ISBN
- автор
- стара ціна
- ціна при замовленні в межах від 1 до 50 одиниць (межа ціни від 1 до 1000)
- ціна при замовленні в межах від 50 до 100 одиниць (межа ціни від 1 до 1000)
- ціна при замовленні в межах від 100 одиниць (межа ціни від 1 до 1000)
- id категорії (таким чином буде зв'язок «багато книг до 1 категорії »)
- посилання на зображення

Клас *ShoppingCart* представляє кошик з необхідними полями, такими як:

- id
- id продукту (таким чином буде зв'язок «у багатьох кошиках може знаходитись 1 продукт»)
- id користувача (в результаті ми організували зв'язок багато кошиків може бути в 1 користувача)

Клас *OrderVM* має в собі OrderHeader та перерахування OrderDetail, тому при передачі даної моделі у представлення ми зможемо отримати доступ до 2-х класів одразу.

Клас *ProductVM* має в собі *Product* та перерахування категорій , тому при передачі даної моделі у представлення ми зможемо отримати доступ до 2-х класів одразу.

Клас *ShoppingVM* має в собі *OrderHeader* та перерахування кошиків , тому при передачі даної моделі у представлення ми зможемо отримати доступ до 2-х класів одразу.

Розглянемо папку *Data* детальніше, а саме файли, що виділені червоним на рисунку 3.8. Код кожного з класів можна знайти у додатках.

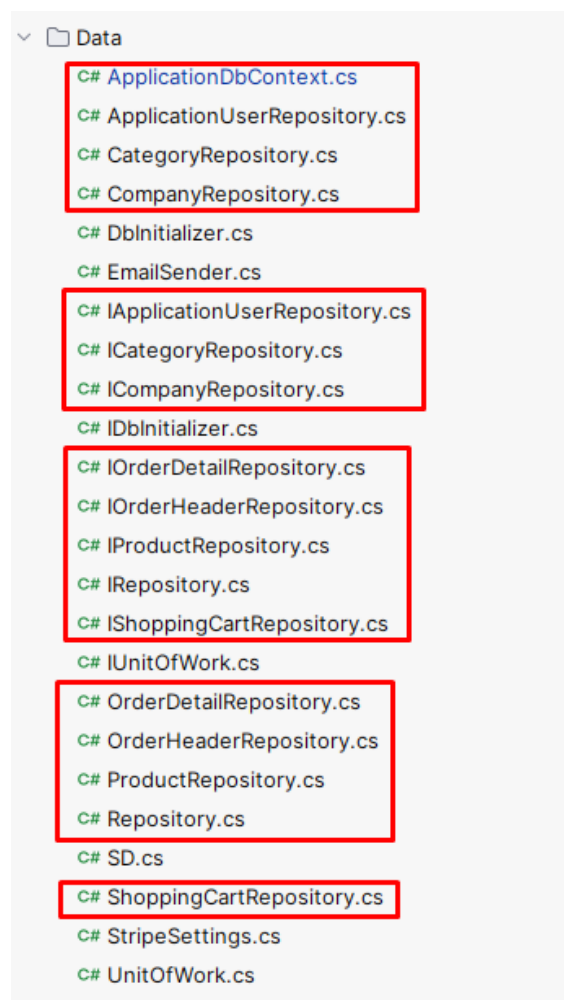


Рисунок 3.8 – папка *Data* з виділеними файлами, що зроблять можливим реалізувати шаблон «Репозиторій»

Клас *ApplicationDbContext* - це контекст бази даних Entity Framework, який використовується для взаємодії з базою даних у програмі.

Кожна властивість представляє DbSet для конкретного класу сутності в базі даних.

Метод *OnModelCreating*:

Цей метод викликається при створенні моделі бази даних. Викликає метод базового класу для виконання за замовчуванням і потім дозволяє налаштувати додаткові параметри моделі.

В даному випадку використовується для додавання початкових даних (Seed Data) до таблиць бази даних, таких як категорії, компанії та продукти.

Інтерфейс *IRepository* – узагальнений інтерфейс, який встановлює контракт для репозиторію, який може працювати з об'єктами сутностей типу *T*. Даний інтерфейс включає основні операції CRUD, окрім оновлення, воно буде оголошено вже в конкретних інтерфейсах.

Метод *GetAll*:

Повертає колекцію всіх сутностей типу *T*. Параметр *filter* дозволяє вказати умову для фільтрації сутностей.

Параметр *includeProperties* дозволяє заздалегідь завантажувати пов'язані сутності.

Метод *Get*:

Повертає одну сутність типу *T* на основі заданої умови (*filter*).Параметр *includeProperties* дозволяє заздалегідь завантажувати пов'язані сутності.

Параметр *tracked* визначає, чи слід відстежувати сутність контекстом Entity Framework.

Метод *Add*:

Додає нову сутність типу *T*.

Метод *Remove*:

Видаляє сутність типу *T*.

Метод *RemoveRange*:

Видаляє колекцію сутностей типу *T*.

Клас *Repository* реалізує інтерфейс *IRepository*.

В цілому можливо було обійтися використанням узагальненого репозиторію,

Розглянемо як виглядає головна сторінка без авторизації (рис.2.6.1).

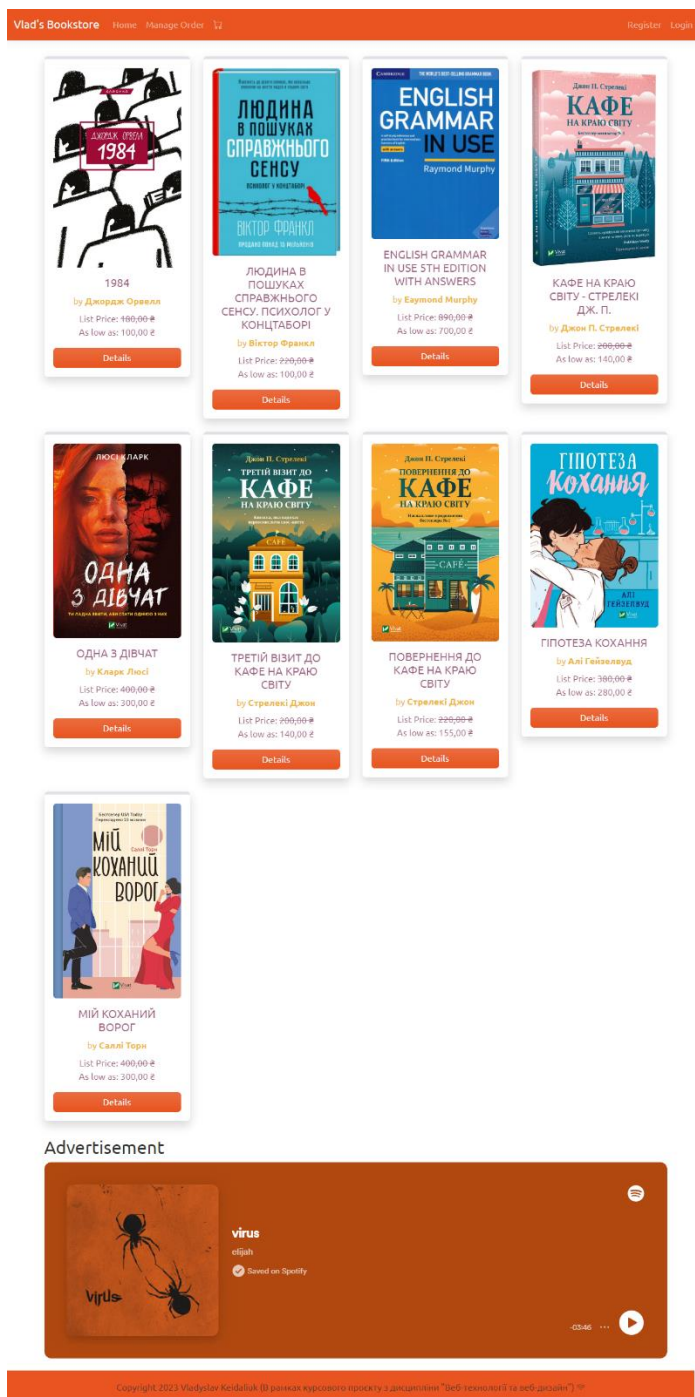


Рисунок 3.9 – Вигляд головної сторінки без авторизації

Можна побачити список книг, блок з рекламою.

Далі подивимось як виглядає сторінка реєстрації (рис.3.10), а також авторизації (рис.3.11).

Register
Create a new account.

Email

The Email field is required.

Name

The Name field is required.

PhoneNumber

Password

The Password field is required.

Confirm password

StreetAddress

City

State

PostalCode

Register

Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.10 – Вигляд сторінки реєстрації

Log in
Use a local account to log in.

Email

The Email field is required.

Password

The Password field is required.

Log in

Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.11 – Вигляд сторінки авторизації

Якщо підсумувати, то на сторінці реєстрації наявні всі необхідні поля для заповнення, а поле Ім'я, пошта та пароль є обов'язковими, тобто наявна валідація даних. Теж саме стосується і сторінки авторизації. Червоний текст зв'являється, коли користувач щось ввів і видалив, або некоректні дані вводить.

Далі розберемо сторінку списку замовлень (рис.3.12– рис.3.13)

Vlad's Bookstore

Home

Manage Order

🔍

Hello customer@gmail.com!

Logout

Order List

ID	Name	Phone Number	Email	Status	Total	
1	Basic Customer	0963951645	customer@gmail.com	Shipped	2630	✕ Details
2	Basic Customer	0963951645	customer@gmail.com	Cancelled	760	✕ Details
3	Basic Customer	0963951645	customer@gmail.com	Approved	150	✕ Details
4	Basic Customer	0963951645	customer@gmail.com	Pending	0	✕ Details
5	Basic Customer	0963951645	customer@gmail.com	Pending	760	✕ Details
6	Basic Customer	0963951645	customer@gmail.com	Pending	760	✕ Details
7	Basic Customer	0963951645	customer@gmail.com	Cancelled	760	✕ Details
8	Basic Customer	0963951645	customer@gmail.com	Pending	1060	✕ Details
9	Basic Customer	0963951645	customer@gmail.com	Approved	1210	✕ Details

Copyright 2023 Vladyslav Kaidaluk (в рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.12 – Вигляд сторінки списку замовлень, якщо зайти під акаунтом покупця, або компанії

Vlad's Bookstore

Home

Manage Order

Content Management

Hello admin@gmail.com

Logout

Order List

ID	Name	Phone Number	Email	Status	Total	
1	Basic Customer	0963951645	customer@gmail.com	Shipped	2630	<div>✕ Details</div>
2	Basic Customer	0963951645	customer@gmail.com	Cancelled	760	<div>✕ Details</div>
3	Basic Customer	0963951645	customer@gmail.com	Approved	150	<div>✕ Details</div>
4	Basic Customer	0963951645	customer@gmail.com	Pending	0	<div>✕ Details</div>
5	Basic Customer	0963951645	customer@gmail.com	Pending	760	<div>✕ Details</div>
6	Basic Customer	0963951645	customer@gmail.com	Pending	760	<div>✕ Details</div>
7	Basic Customer	0963951645	customer@gmail.com	Cancelled	760	<div>✕ Details</div>
8	Basic Customer	0963951645	customer@gmail.com	Pending	1060	<div>✕ Details</div>
9	Basic Customer	0963951645	customer@gmail.com	Approved	1210	<div>✕ Details</div>
10	Standart Employee	0954629678	employee@gmail.com	Pending	150	<div>✕ Details</div>
11	Ily Company	04863128321	company@gmail.com	Approved	760	<div>✕ Details</div>

Copyright 2023 Vladyslav Kaidaluk (в рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.13 – Вигляд сторінки списку замовлень, якщо зайти під акаунтом адміністратора, або робітника

Підсумовуючи можна сказати, що дана сторінка виводить основну інформацію про те, хто зробив замовлення, на яку суму та який статус замовлення.

Далі розберемо сторінку керування замовленням (рис. 3.14 – рис. 2.6.12).

Vlad's Bookstore Home Manage Order Content Management ⌵ Hello admin@gmail.com! Logout

Order Summary Back to Orders

PickUp Details:

Name: My Company

Phone: 04863128321

Address: Lvivska 54

City: Kharkiv

State: Ukraine

Zip Code: 66984

Email: company@gmail.com

Order Date: 10.12.2023 00:00:00

Carrier:

Tracking Number:

Shipping Date: 01.01.0001

Session ID:

Payment Intent ID:

Payment Due Date: 01.01.0001

Payment Status: ApprovedForDelayedPayment

Update Order Details

Order Summary

Order Status - Approved

English Grammar in Use 5th Edition with Answers

Price : 760,00 ₴

Quantity : 1

TOTAL 760,00 ₴

Start Processing

Cancel Order

Copyright 2023 Vladyslav Keldaluk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.14 – Вигляд сторінки керування замовлення здійснений робітником якоїсь компанії, статус *Approved*

Натискаємо на кнопку *StartProcessing*. Стан замовлення змінився на *Processing*. Побачити це можна на рисунку 3.15.

Vlad's Bookstore Home Manage Order Content Management ⌵ Hello admin@gmail.com! Logout

Order Details Updated Successfully

Order Summary Back to Orders

PickUp Details:

Name: My Company

Phone: 04863128321

Address: Lvivska 54

City: Kharkiv

State: Ukraine

Zip Code: 66984

Email: company@gmail.com

Order Date: 10.12.2023 00:00:00

Carrier:

Tracking Number:

Shipping Date: 01.01.0001

Session ID:

Payment Intent ID:

Payment Due Date: 01.01.0001

Payment Status: ApprovedForDelayedPayment

Update Order Details

Order Summary

Order Status - Processing

English Grammar in Use 5th Edition with Answers

Price : 760,00 ₴

Quantity : 1

TOTAL 760,00 ₴

Ship Order

Cancel Order

Copyright 2023 Vladyslav Keldaluk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.15 – Вигляд сторінки керування замовлення здійснений робітником якоїсь компанії, статус *Processing*

Далі нам необхідно натиснути кнопку *ShipOrder*, але система не дозволить цього зробити, поки не будуть заповнені поля *Carrier* (перевізник) та *Tracking Number* (трек номер, за яким можна відстежувати процес доставки в перевізника).

Заповнюємо ці поля та натискаємо на кнопку *ShipOrder*. Стан замовлення змінився на *Shipped*. Також в нас з'явилась кнопка *Pay Now*. Так як замовлення зроблено компанією, то в неї є 30 днів на оплату товару після його відправки. Побачити це можна на рисунку 3.16.

The screenshot displays the 'Order Summary' page. At the top, there is a navigation bar with 'Vlad's Bookstore' and links for 'Home', 'Manage Order', and 'Content Management'. A green notification banner at the top right states 'Order Shipped Successfully.' with a 'Logout' link. The main content area is divided into two columns. The left column, titled 'PickUp Details:', contains a form with fields for Name, Phone, Address, City, State, Zip Code, Email, Order Date, Carrier, Tracking Number, Shipping Date, Session ID, Payment Intent ID, Payment Due Date, and Payment Status. The right column, titled 'Order Summary', shows the 'Order Status - Shipped' and a list of items: 'English Grammar in Use 5th Edition with Answers' with a price of 760,00 ₺ and a quantity of 1. A 'TOTAL' row shows 760,00 ₺, and a green 'Pay Now' button is at the bottom. A footer bar at the bottom contains the copyright notice: 'Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")'.

Рисунок 3.16 – Вигляд сторінки керування замовленням здійснений робітником якоїсь компанії, статус *Shipped*

Натискаємо на кнопку *Pay Now*. В нас відбувається переадресація на платіжну систему Stripe.

Так як в нас аккаунт в платіжній системі в режимі тестування, то введемо тестові дані, що рекомендовані (номер картки : 4242 4242 4242 4242, будь-яка дата

видачі та буд-який код CVV/CVC). Заповнюємо дані для оплати (рис. 3.17)

Після оплати ми побачимо сторінку про успішну оплату та який номер в нашого замовлення (рис. 3.18)

Натискаємо на кнопку Back to Order Details та бачимо, що в нас заповнились поля Session ID та Payment Intent ID (рис. 3.19), вони надаються платіжною системою. Також важливим моментом є те, що ці поля не є редагованими, як і пошта, дата замовлення, дата відправлення та дата оплати.

Рисунок 3.17 – Заповнення даних для оплати замовлення

Рисунок 3.18 – Вигляд сторінки про успішну оплату

Vlad's Bookstore
Home
Manage Order
Content Management
Hello admin@gmail.com! Logout

Order Summary

Back to Orders

PickUp Details:

Name

My Company

Phone

04863128321

Address

Lvivska 54

City

Kharkiv

State

Ukraine

Zip Code

66984

Email

company@gmail.com

Order Date

10.12.2023 00:00:00

Carrier

Nova Poshta

Tracking Number

30603846903

Shipping Date

11.12.2023

Session ID

cs_test_a1xdBs5t0gf71Ok5QcJec2wTY

Payment Intent ID

pi_30MAmyL9izDJMcDB0mQTjTzp

Payment Date

11.12.2023

Payment Status

Approved

Update Order Details

Order Summary

Order Status - Shipped

English Grammar in Use 5th Edition with Answers

760

Price : 760,00 ₪

Quantity : 1

TOTAL

760,00 ₪

Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.19 – Вигляд сторінки керування замовленням здійснений робітником якоїсь компанії після оплати

Тепер перейдемо в акаунт платіжної системи і бачимо, що платіж дійсно був успішно виконаний, Session ID співпадали, побачити це можна на рисунку 3.20.

	Amount		Description	Customer	Date
<input type="checkbox"/>	₪760.00	UAH	Succeeded ✓	pi_30MAmyL9izDJMcDB0mQTjTzp mycompany@gmail.com	Dec 11, 2:48 PM

Рисунок 3.20 – Підтвердження успішної оплати в платіжній системі

Далі розберемо сторінку кошика (рис. 3.21) та ітогової сторінки оформлення замовлення (рис.3.22).

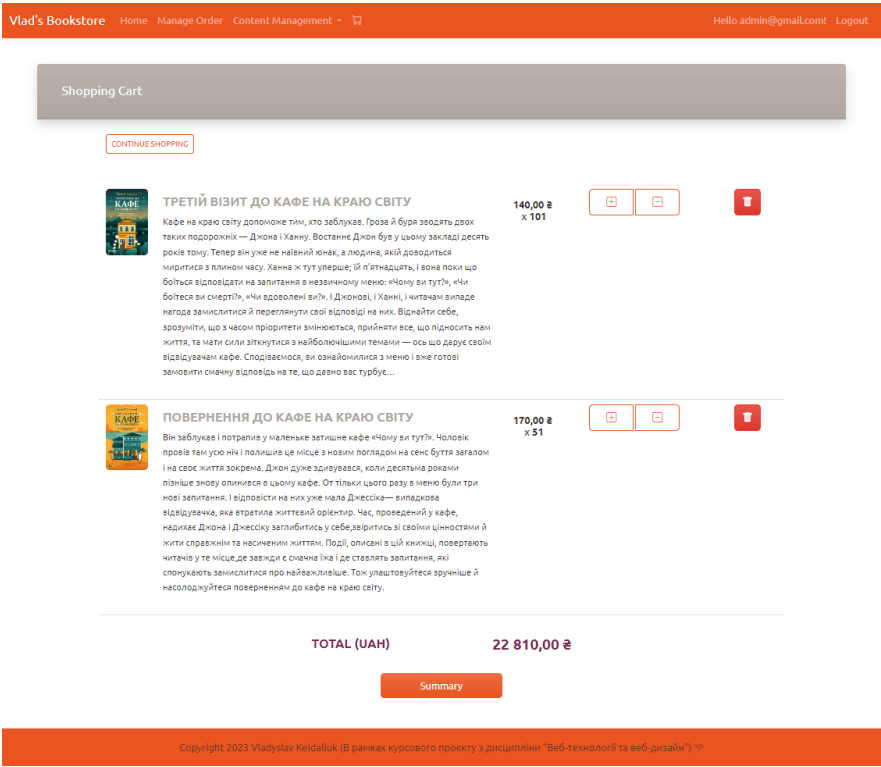


Рисунок 3.21 – Вигляд сторінки кошика з товарами всередині

Як ми бачимо в нас є зображення товару, назва, опис, кількість а також ціна. Важливо сказати, що при різній кількості товарів буде різна ціна.

Далі можна побачити, що в нас є кнопка видалення товару з кошика, а також загальна сума замовлення. Ще можна побачити кнопку для переходу на ітогову сторінку оформлення замовлення.

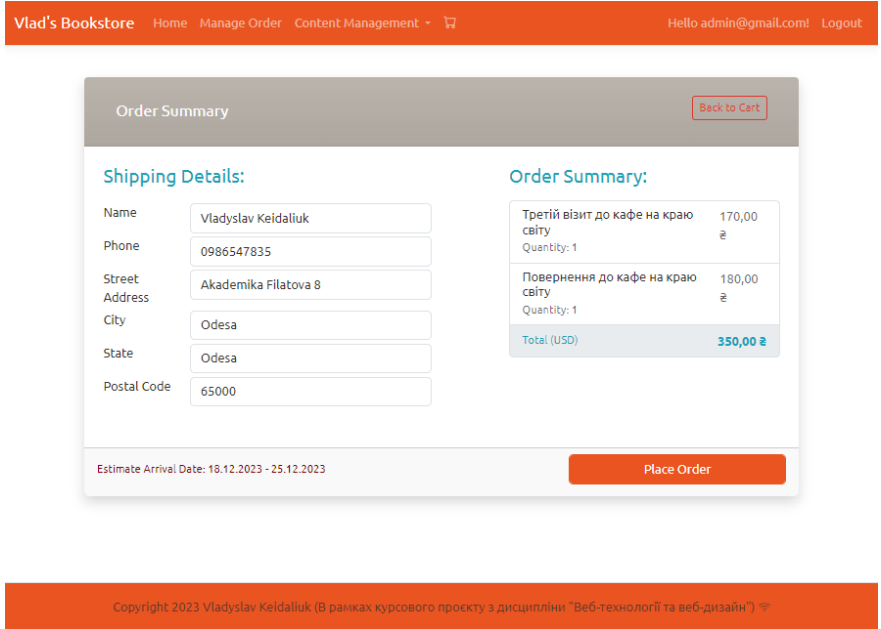


Рисунок 3.22 – Вигляд ітогової сторінки оформлення замовлення

На даній сторінці (рис. 3.22) ми бачимо, що дані автоматично заповнюються тими даними, що користувач увів при реєстрації, за необхідністю їх можна змінити.

Дана сторінка складається з двох блоків – інформація про доставку та інформація про замовлення.

Також можна побачити, що в нас пишеться орієнтовний термін доставки замовлення.

При натисканні на кнопку Place Order буде переадресування на сторінку платіжної системи та після оплати буде переадресація на сторінку, де буде сказано, що замовлення успішно оформлене та номер замовлення. Все це вже було показано на рисунках 3.16 – 3.20.

Далі розберемо сторінку товару на прикладі книги «Повернення до кафе на краю світу» (рис. 3.23).

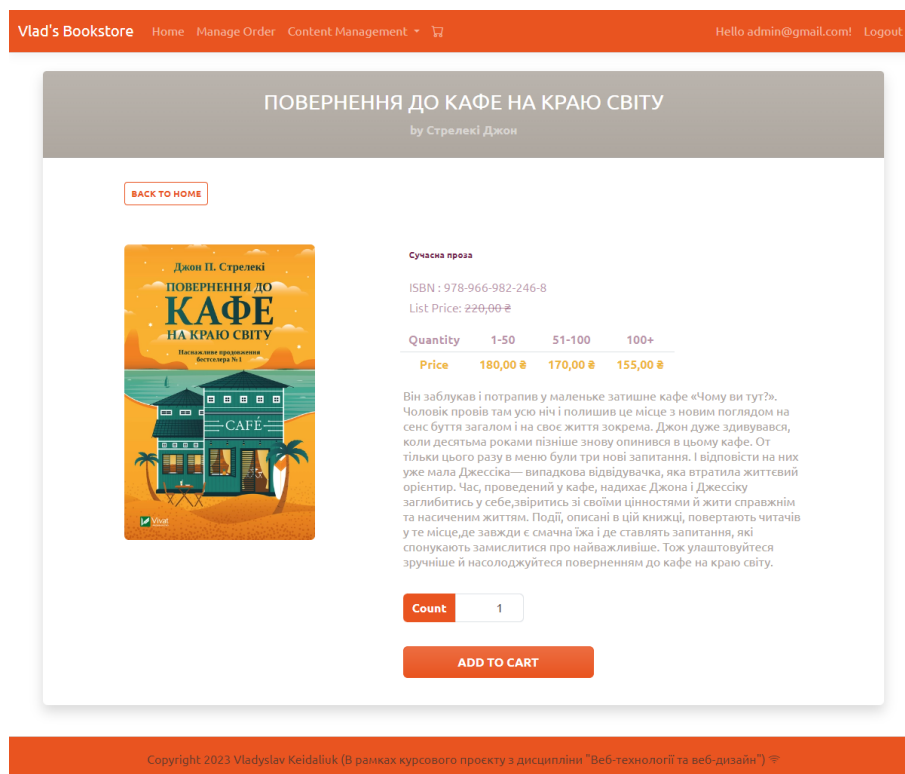


Рисунок 3.23 – Вигляд сторінки товару (книги)

Як ми бачимо тут наявна назва книги, автор, зображення лицьової частини книжки.

Далі можна побачити категорію, до якої відноситься книга. Нижче видно таблицю кількість-ціна. Ще можна обрати кількість товару та додати до кошика.

Що стосується профіля користувача, то ці розглядати не будемо, адже вони поставляються Identity.

Далі розберемо сторінки відображення списку категорій, продуктів та компаній (рис. 3.24 – рис. 3.26)

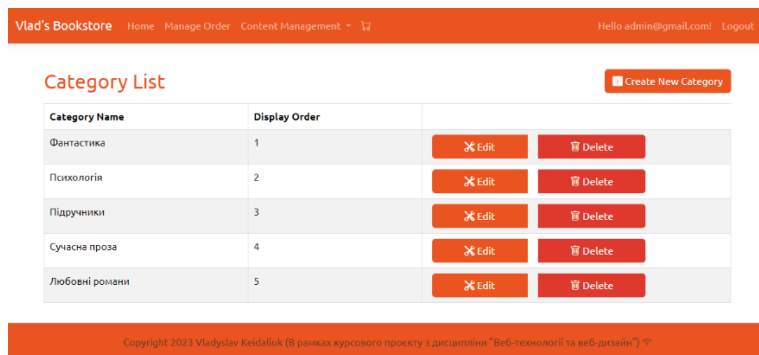


Рисунок 3.24 – Вигляд сторінки відображення списку категорій

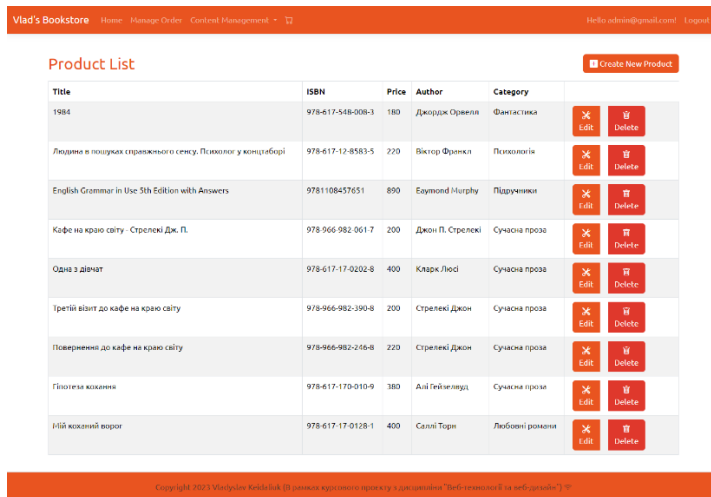


Рисунок 3.25 – Вигляд сторінки відображення списку товарів

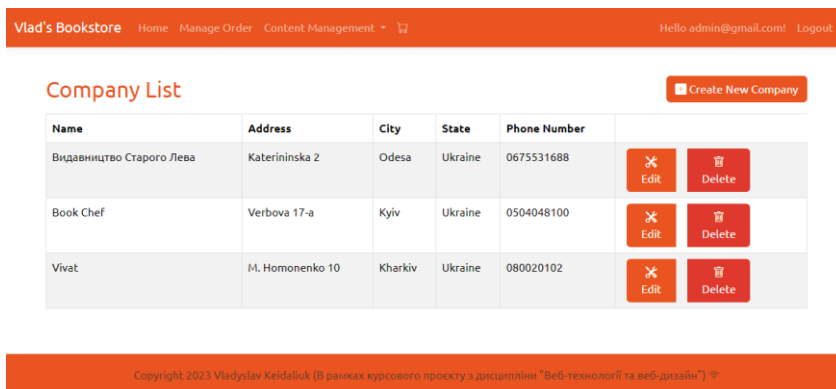


Рисунок 3.26 – Вигляд сторінки відображення списку компаній

На рисунку 3.24 ми можемо побачити назву категорії та номер порядку відображення.

На рисунку 3.25 бачимо назви книг, ISBN код, ціну, автора та категорію, до якої відноситься книга.

На рисунку 3.26 можна побачити назв компаній, їх адреси, місто, країну та номер телефону.

Так як сторінки створення, редагування та видалення приблизно однакового вигляду, то розглянемо сторінку створення (рис.3.27), редагування (рис.3.28) та видалення (рис.3.29) на прикладі категорії.

Vlad's Bookstore Home Manage Order Content Management • Hello admin@gmail.com Logout

Create Category

Category Name

Display Order

Create Back to List

Copyright 2023 Vladyslav Keldaluk (В рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.27 – Вигляд сторінки додавання нової категорії

Vlad's Bookstore Home Manage Order Content Management • Hello admin@gmail.com Logout

Edit Category

Category Name
Любовні романи

Display Order
5

Update Back to List

Copyright 2023 Vladyslav Keldaluk (В рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.28 – Вигляд сторінки редагування категорії

Vlad's Bookstore Home Manage Order Content Management • Hello admin@gmail.com Logout

Delete Category

Category Name
Фантастика

Display Order
1

Delete Back to List

Copyright 2023 Vladyslav Keldaluk (В рамках курсового проекту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.29 – Вигляд сторінки видалення категорії

Далі розберемо сторінку реєстрації нового користувача (рис. 3.30).

Як ми бачимо, тут є поля для заповнення пошти, ім'я, номеру телефона, пароля, підтвердження пароля, адреси, міста, країни та поштового індексу.

Також якщо натиснути на *Select Role* то буде випадаючий список (рис. 3.31), з якого можна обрати який рівень доступу буде в даного користувача.

Vlad's Bookstore Home Manage Order Content Management Hello admin@gmail.com! Logout

Register

Create a new account.

Email

Name

PhoneNumber

Password

Confirm password

StreetAddress

City

State

PostalCode

-Select Role-

Register

Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн")

Рисунок 3.30 – Вигляд сторінки реєстрації нового користувача

-Select Role-

-Select Role-

Company

Customer

Employee

Admin

Рисунок 3.31 – Випадаючий список для вибору ролі

Наостанок розберемо сторінку списку користувачів (рис. 3.32).



Vlad's Bookstore Home Manage Order Content Management  Hello admin@gmail.com! Logout				
User List				
Name	Email	Phone	Role	Status
Basic Customer	customer@gmail.com	0963951645	Customer	Locked
NewUser	newuser838840@gmail.com	0956543621	Customer	Unlocked
Standart Employee	employee@gmail.com	0954629678	Employee	Unlocked
NewUser	newuser626487@gmail.com	0956543621	Customer	Unlocked
My Company	company@gmail.com	04863128321	Company	Unlocked
User0612	user0613442@gmail.com	0965267653	Customer	Unlocked
Vladyslav Keidaliuk	admin@gmail.com	0986547835	Admin	Unlocked
User0612	user0612@gmail.com	0965267653	Customer	Unlocked
NewUser	newuser452023@gmail.com	0956543621	Customer	Unlocked
Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн") 				

Рисунок 3.32 – Вигляд сторінки списку користувачів

На даній сторінці (рис. 3.32) адміністратор може побачити список всіх користувачів системи та їх дані, такі як:

- ім'я
- пошта
- номер телефону
- роль
- статус

Статус – звичайний стан (розблокований), або заблокований. Якщо заблокувати користувача, то він втратить доступ до системи назавжди, але адміністратор завжди зможе розблокувати певного користувача.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Автоматизоване тестування

Тестування програмного забезпечення може зробити програмний продукт більш привабливим для кінцевих користувачів, адже це :

- покращує зручність використання та користувацький досвід,
- допомагає надати користувачам результати, яких вони хочуть від продукту,
- підвищує загальну стабільність застосунку.

Необхідно надати ключові переваги автоматизованого тестування.

Перш за все, це швидкість.

Звичайно, комп'ютери можуть виконувати деякі дії набагато швидше, ніж людина.

Друга перевага - це надійність.

Звичайно, ми розуміємо, що автоматизовані тести та будь-яке інше програмне забезпечення може містити помилки, може бути якимось чином несправним. Але комп'ютер ніколи не робить людських помилок. Він ніколи не забуде заповнити якісь поля, натиснути якісь кнопки і так далі.

Наступне - це потужність.

Комп'ютери можуть виконувати такі дії, які абсолютно неможливі для людини, особливо якщо ми говоримо про низькорівневі операції або якесь високопродуктивне навантаження і так далі.

Наступна перевага - це статистика.

Комп'ютери дуже добре працюють зі статистикою. Комп'ютери можуть збирати деякі дані, обчислювати деякі дані, робити гарні графіки, таблиці чи будь-що інше, що нам подобається. Тому збирати статистику і представляти її у зручному вигляді - це гарна ідея. Це добре для автоматизації тестування.

І як було сказано вище, низькорівневі дії, низькорівневі операції, такі як виклик деякої динамічно зв'язаної бібліотеки з використанням деякого двійкового протоколу, виконання деяких речей всередині операційної системи, виконання

деяких низькорівневих або файлових операцій і так далі. Людині дуже, дуже важко і складно виконувати такі дії. І це абсолютно просто для будь-якого інструменту автоматизації тестування.

Саме тому для тестування свого застосунку я обрав автоматизоване тестування. Для цього було обрано NUnit – фреймворк для юніт-тестування для всіх .Net мов.

А також було використано Selenium - це безкоштовний фреймворк автоматизованого тестування з відкритим вихідним кодом, який використовується для перевірки веб-додатків у різних браузерах і на різних платформах. Було задіяно Selenium Webdriver.

4.2 Функціональне тестування

Функціональне тестування - це тип тестування програмного забезпечення, який перевіряє функціональність програмної системи або застосунку. Воно зосереджене на тому, щоб переконатися, що система поводить себе відповідно до визначених функціональних вимог і відповідає запланованим бізнес-потребам.

Метою функціонального тестування є перевірка функцій, можливостей та взаємодії системи з різними компонентами. Воно включає в себе тестування вхідних і вихідних даних програмного забезпечення, маніпуляції з даними, взаємодію з користувачем і реакцію системи на різні сценарії та умови. Функціональне тестування стосується лише перевірки того, чи працює система за призначенням.

В контексті виконаного проекту, було створено тестові випадки, необхідні вхідні дані.

Було визначено, що в застосунку необхідно протестувати:

- Реакцію систему на вхід з правильними вхідними даними (пошта та пароль користувача)
- Реакцію на вхід з неправильними вхідними даними
- Зони доступу (покупець ніяким чином не має потрапити на сторінки, що

доступні адміністратору і що зона адміністратора доступна для самого адміністратора)

- Роботу блокування та розблокування користувача
- Роботу функцій створення, зчитування, редагування та видалення категорій, компаній, товарів
- Можливість створення нового користувача через панель адміністратора
- Правильність додавання товарів до кошика

4.3 Шаблон проектування Page Object Model

Мною була обрана наступна структура проекту, що показана на рисунку 4.1.

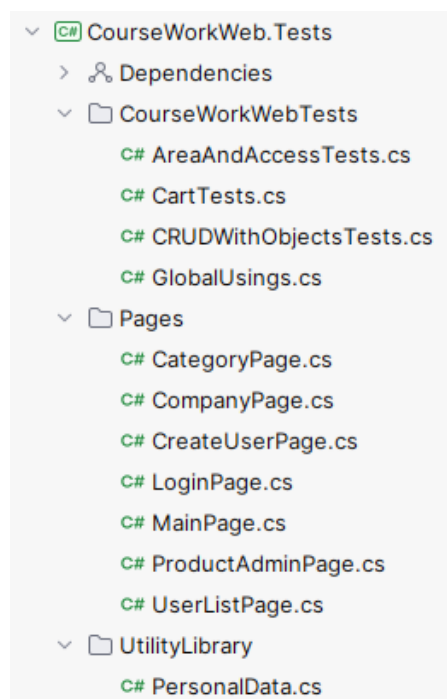


Рисунок 4.1 – Структура проекту тестів

Дана структура відповідає «*Page Object Model*» - це шаблон проектування для створення сховища об'єктів для елементів інтерфейсу, доступних для тестування.

Він в основному використовується в Selenium для автоматизації наших тестів. Ідея реалізації «*Page Object Model*» полягає у створенні центрального сховища для управління веб-сторінкою.

Даний шаблон було обрано через низку переваг:

Зручність супроводу:

У моделі об'єктів сторінок існують окремі об'єкти сторінок для декількох веб-сторінок. Тому при зміні функціоналу, він продовжить працювати без особливого впливу на тестові скрипти. Це робить наші тестові скрипти легко підтримуваними і пропонує нам чистіший код, який легко зрозуміти.

Можливість багаторазового використання:

Основна перевага використання об'єктної моделі сторінки в Selenium - це отримання бажаного сценарію за меншу кількість коду. Використовуючи *Page Object Model*, можна використовувати одні й ті самі об'єкти для різних цілей. Таким чином, це економить час і ресурси.

4.4 Класи сторінок

Класи, що представляють відповідні сторінки можна побачити в папці *Pages* на рисунку 4.1.

Розберемо детальніше кожен клас, почнемо з *CategoryPage*.

Локатори елементів (*By*):

- *NewCategoryButton*, *CategoryNameInput*, *DisplayOrderInput*, *CreateButton*, *DeleteButton*, *EditCategoryNameInput*, *EditDisplayOrderInput*, *UpdateButton* - це CSS селектори для знаходження відповідних елементів на веб-сторінці.

Методи:

- *NewCategoryButtonClick()*: Натискання на кнопку для створення нової категорії.
- *NewCategoryCreate(string name, string displayOrder)*: Введення даних та натискання кнопки для створення нової категорії. Після цього перевіряється наявність нової категорії в таблиці.
- *EditCategory(string name, string newName, string displayOrder)*: Редагування вказаної категорії з новими даними та оновлення.

Перевіряється успішність редагування.

- *DeleteCategory(string name)*: Видалення вказаної категорії. Перевіряється успішність видалення.

Використання `Thread.Sleep()`:

Було використано `Thread.Sleep()` для пауз між окремими кроками. Це було використано для того, щоб дати час веб-сторінці на відповідь і для відображення всіх об'єктів.

Далі розглянемо *CompanyPage*.

Локатори елементів (By):

- *NewProductButton*, *NameInput*, *PhoneNumberInput*, *StreetAddressInput*, *CityInput*, *StateInput*, *PostalCodeInput*, *CreateButton*, *DeleteButton*, *UpdateButton* - це CSS селектори для знаходження відповідних елементів на веб-сторінці.

Методи:

- *NewCompanyButtonClick()*: Натискання на кнопку для створення нової компанії.
- *NewCompanyCreate(...)*: Введення даних та натискання кнопки для створення нової компанії. Після цього перевіряється наявність нової компанії в таблиці.
- *EditCompanyData(...)*: Редагування даних вказаної компанії з новими даними та оновлення. Перевіряється успішність редагування.
- *DeleteCompany(...)*: Видалення вказаної компанії. Перевіряється успішність видалення.

Тепер розглянемо *CreateUserPage*.

Локатори елементів (By):

- *NewProductButton*, *EmailInput*, *NameInput*, *PhoneNumberInput*, *PasswordInput*, *ConfirmPasswordInput*, *StreetAddressInput*, *CityInput*, *StateInput*, *PostalCodeInput*, *RegisterButton* - це CSS селектори для знаходження відповідних елементів на веб-сторінці.

Методи:

- *NewCompanyButtonClick()*: Натискання на кнопку для створення нового користувача.
- *NewCustomerCreate(...)*: Введення даних та натискання кнопки для реєстрації нового користувача.
- *CheckUserExist(...)*: Перевірка наявності користувача в таблиці. Повертає true, якщо користувач існує.

Далі розглянемо *CreateUserPage*.

Локатори елементів (By):

- *EmailTextBox*, *PasswordTextBox*, *LogInButton*, *LoginForm*, *LockedOut* - це CSS селектори та XPath для знаходження відповідних елементів на веб-сторінці.

Методи:

- *Login(...)*: Введення даних електронної пошти та пароля, натискання кнопки входу та перевірка успішності входу.

Далі розглянемо *MainPage*.

Методи для взаємодії з елементами сторінки:

Методи служать для взаємодії з основними елементами, такими як кнопки і посилання. Наприклад, *LoginButtonClick()*, *RegisterButtonClick()*, *CartButtonClick()*, тощо. Кожен з цих методів використовує об'єкт *WebDriverWait* для очікування та знаходження відповідного елемента на сторінці та подальшого виконання дії, такої як клік.

Додаткові функції:

Крім основних методів, є деякі додаткові функції, такі як *ScrollDown()*, *CheckThatBookExistInsideCart()*, які використовуються для прокрутки сторінки та перевірки, чи є певний товар в кошику.

Далі розглянемо *ProductAdminPage*.

Локатори елементів на сторінці:

У класі оголошено локатори елементів, такі як *NewProductButton*, *TitleInput*, *DescriptionInput*, тощо. Кожен локатор вказує на конкретний елемент сторінки за допомогою CSS-селекторів.

Методи для взаємодії зі сторінкою:

Даний клас має методи для виконання різних дій на сторінці адміністрування продуктів, такі як *NewProductButtonClick()*, *NewProductCreate()*, *EditTitleInProduct()*.

Наостанок розглянемо *UserListPage*.

Метод *LockUnlockUserWithEmail*: Цей метод призначений для блокування або розблокування користувача за його електронною поштою.

Метод отримує рядок із електронною поштою, шукає відповідний рядок у таблиці користувачів і взаємодіє з відповідним елементом, щоб виконати дію блокування / розблокування.

4.5 Тести та тестування

Самі тести можна побачити в папці *CourseWorkWebTests* на рисунку 4.1.

Розглянемо файл *AreaAndAccessTests*.

Налаштування (*Setup* та *TearDown*) та ініціалізація:

- В методі *Setup* створюється екземпляр веб-драйвера (*ChromeDriver*), налаштовується для максимізації вікна та ініціалізуються екземпляри сторінок (*MainPage*, *LoginPage*, *UserListPage*).
- У методі *TearDown* викликається закриття та завершення роботи драйвера.

Тести авторизації:

- *LogInWithValidDataReturnTrue*: Перевіряє, чи можна авторизуватися з різними ролевими обліковими записами. Застосовується параметризація за допомогою атрибуту *TestCaseSource*.
- *LogInWithInvalidDataReturnFalse*: Перевіряє, чи неможлива авторизація з невірними обліковими даними.

Тести доступу до областей:

- *AccessDeniedForCustomerToAdminAreaReturnFalse*: Перевіряє, чи заборонено доступ користувачам з роллю «*Customer*» до адміністративної

області.

- *AccessAllowedForAdminToAdminAreaReturnTrue*: Перевіряє, чи дозволяється доступ адміністратору до адміністративної області.

Тести блокування та розблокування користувачів:

- *LockCustomerByAdminReturnFalse*: Перевіряє, чи адміністратор може заблокувати користувача та чи йому буде відмовлено в доступі після цього.
- *UnlockCustomerByAdminReturnTrue*: Перевіряє, чи адміністратор може розблокувати користувача та чи йому буде надано доступ після цього.

Далі розглянемо файл *CartTests*.

Налаштування (*Setup* та *TearDown*) та ініціалізація:

- В методі *Setup* створюється екземпляр веб-драйвера (*ChromeDriver*), налаштовується для максимізації вікна та ініціалізуються екземпляри сторінок (*MainPage*, *LoginPage*).
- У методі *TearDown* викликається закриття та завершення роботи драйвера.

Тест додавання 5 книг до кошика (*Add5BookToCart*):

Авторизується користувач. Додає до кошика 5 книг шляхом прокрутки сторінки, натискання кнопки "*Details*", а потім "*Add to Cart*". Після цього виходить з облікового запису користувача.

Тест перевірки, що 5 книг додані в кошик (*CheckThat5BookToCartAddedReturnTrue*):

Авторизується користувач. Перевіряє, чи кожна з п'яти книг із заданими назвами дійсно присутня в кошику. Після цього виходить з облікового запису користувача.

Далі розглянемо файл *CRUDWithObjectsTests*.

Налаштування (*SetUp* та *TearDown*) та ініціалізація:

- В методі *SetUp* створюється екземпляр веб-драйвера (*ChromeDriver*), налаштовується для максимізації вікна та ініціалізуються екземпляри сторінок (*MainPage*, *LoginPage*, *CategoryPage*, *ProductAdminPage*,

CompanyPage, CreateUserPage).

- У методі `TearDown` викликається закриття та завершення роботи драйвера.

Тести для CRUD-операцій з категоріями:

- Створення нової категорії (*CreateNewCategoryTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління категоріями. Створює нову категорію і перевіряє успішність створення.
- Редагування категорії (*EditCategoriesTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління категоріями. Редагує існуючу категорію та перевіряє успішність редагування.
- Видалення категорії (*DeleteCategoriesTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління категоріями. Видаляє категорію та перевіряє успішність видалення.

Тести для CRUD-операцій з продуктами:

- Створення нового продукту (*CreateProductTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління продуктами. Створює новий продукт та перевіряє успішність створення.
- Редагування назви продукту (*EditProductTitleTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління продуктами. Редагує існуючу назву продукту та перевіряє успішність редагування.

Тести для CRUD-операцій з компаніями:

- Створення нової компанії (*CreateCompanyTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління компаніями. Створює нову компанію та перевіряє успішність створення.
- Редагування даних компанії (*EditCompanyDataTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління компаніями. Редагує існуючі дані компанії та перевіряє успішність редагування.
- Видалення компанії (*DeleteCompanyTestReturnTrue*): Логіниться адміністратор. Переходить до розділу управління компаніями. Видаляє компанію та перевіряє успішність видалення.

Запустимо створені тести на виконання. Результат виконання можна побачити на рисунку 4.2.

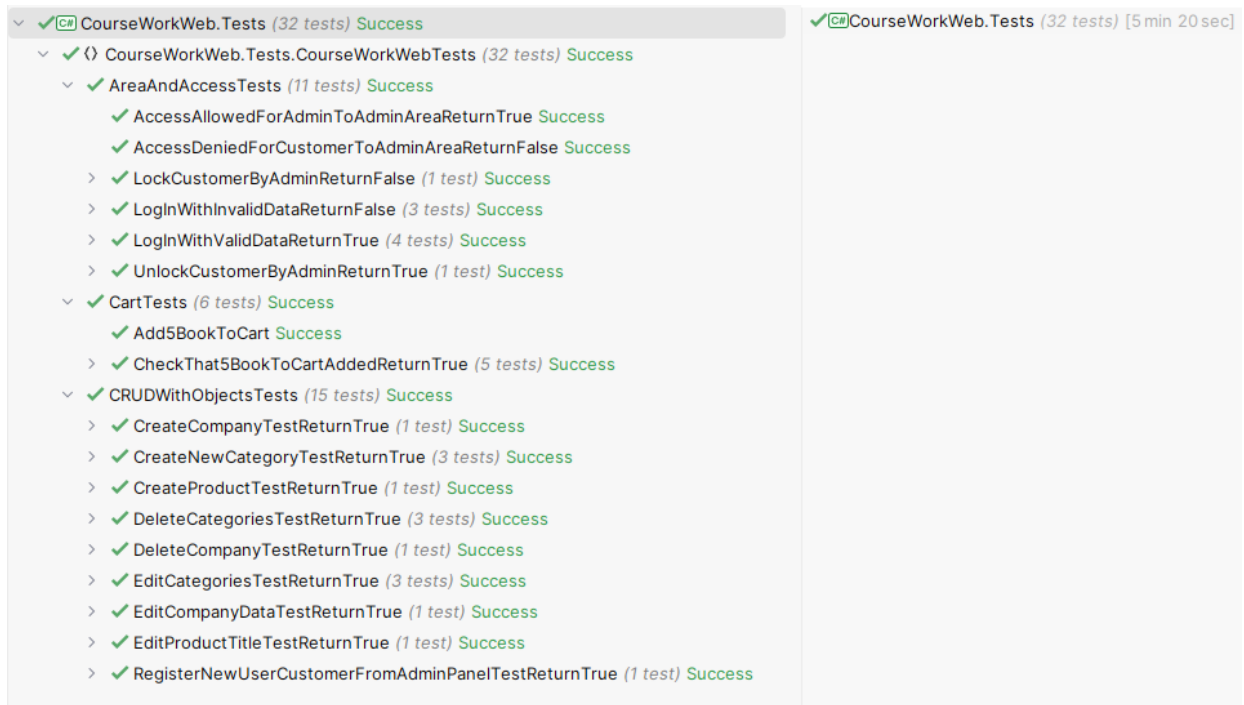


Рисунок 4.2 – Результат виконання тестів

Як бачимо, усі тести успішно пройдені. На даній підставі можна прийти до висновку, що більшість функціоналу працює відповідно до очікування.

У кожному тесті використовується метод `Assert`, щоб перевірити очікуваний результат. Також використовується анотація `Order` для вказівки порядку виконання тестів.

ВИСНОВКИ

У ході виконання курсової роботи був розроблений повноцінний веб-додаток на платформі ASP.Net Core з використанням шаблону MVC, Unit of Work (робоча одиниця) та ViewModel (модель представлення).

Обрана тема - «Книжний магазин», дозволила реалізувати різноманітний функціонал та використати різні аспекти веб-програмування.

В процесі створення застосунку було використано інфраструктуру Identity для реалізації системи аутентифікації та авторизації. Це надало можливість розподілу користувачів за ролями, забезпечивши різний рівень доступу та можливостей у системі.

Для бізнес-рівню було використано інфраструктуру Entity Framework, що дозволило облегшити оперування даними, а завдяки механізму міграцій не довелося вручну прописувати запити на створення таблиць.

Окрім того, була впроваджена система оплати Stripe, що дозволяє користувачам безпечно та зручно здійснювати оплату товарів.

Важливим етапом було написання автоматизованих тестів за допомогою NUnit та Selenium WebDriver для забезпечення стабільності роботи застосунку та вчасного виявлення можливих проблем.

При написанні тестів було дотримано шаблон проектування Page-Object-Model, що дозволяє скоротити час на написання тестів, та дозволяє легко масштабувати тести.

Отже, результатом виконаної роботи є функціональний та безпечний веб-застосунок, а саме інтернет-магазин, який відповідає вимогам проектування та безпеки. Реалізований функціонал дозволяє користувачам зручно та ефективно взаємодіяти з книжним магазином, а адміністраторам та робітникам - ефективно управляти контентом та замовленнями відповідно.

Увесь код застосунку розміщено в додатках, приватна інформація замінена на фальшиву.

ПЕРЕЛІК ПОСИЛАНЬ

1. Brind M. ASP.NET Core Razor Pages in action / Mike Brind. – [Б. м.] : Manning, 2022. – 400 с.
2. Collin M. Mastering Selenium WebDriver 3.0: Boost the performance and reliability of your automated checks by mastering Selenium WebDriver, 2-е видання / Mark Collin. – [Б. м.] : Packt Publishing, 2018. – 376 с.
3. Crispin L. Agile testing: a practical guide for testers and agile teams / Lisa Crispin. – Upper Saddle River, NJ : Addison-Wesley, 2009. – 576 с.
4. Freeman A. Pro ASP.NET Core Identity: under the hood with authentication and authorization in ASP.NET core 5 and 6 applications / Adam Freeman. – [Б. м.] : Apress, 2021. – 746 с.
5. Martin R. Clean code: a handbook of agile software craftsmanship / Robert Martin. – [Б. м.] : Pearson, 2008. – 464 с.
6. Price M. J. C# 11 and .NET 7 - modern cross-platform development fundamentals: start building websites and services with ASP. NET core 7, blazor, and EF core 7, 7-е видання / Mark J. Price. – [Б. м.] : Packt Publishing, Limited, 2022.
7. Smart J. F. BDD in Action: Behavior-driven development for the whole software lifecycle / John Ferguson Smart. – [Б. м.] : Manning Publications, 2014. – 384 с.
8. Smith J. Entity framework core in action / Jon Smith. – [Б. м.] : Manning Publications, 2018. – 520 с.

Додаток А

Код файлу appsettings.json та Program.cs

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Database=your_db;User=root;Password=your_password"
  },
  "Stripe": {
    "PublicKey": "pk_test_544kj3h56j4h6465j464k5hM0Xnq6jBknbeqqJtZq151CF6aRy3kqEYdFgVDMhrj5tNnLmAXu00MxoPgQLn",
    "Secretkey": "sk_test_544kj3h56j4h6465j464k5h 7bxJoviNSIkCAsY9cnkud8HaVTqskhoYGuUKEmt8cE5N9DdmjV8C1DviWhuCU00b1KypbDA"
  }
}
```

Program.cs

```
using CourseWorkWeb.Data;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Stripe;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySQL(builder.Configuration.GetConnectionString("DefaultConnection")));
builder.Services.Configure<StripeSettings>(builder.Configuration.GetSection("Stripe"));
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>().AddDefaultTokenProviders();
builder.Services.ConfigureApplicationCookie(options =>
{
    options.LoginPath = $"/Identity/Account/Login";
    options.LogoutPath = $"/Identity/Account/Logout";
    options.AccessDeniedPath = $"/Identity/Account/AccessDenied";
});
builder.Services.AddRazorPages();

builder.Services.AddScoped<IDbInitializer, DbInitializer>();
builder.Services.AddScoped<IUnitOfWork, UnitOfWork>();
builder.Services.AddScoped<IEmailSender, EmailSender>();
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
StripeConfiguration.ApiKey = builder.Configuration.GetSection("Stripe:SecretKey").Get<string>();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
SeedDataBase();
app.MapRazorPages();
app.MapControllerRoute(
    name: "default",
    pattern: "{area=Customer}/{controller=Home}/{action=Index}/{id?}");
app.Run();
void SeedDataBase()
{
    using (var scope = app.Services.CreateScope())
    {
        var dbInitializer = scope.ServiceProvider.GetRequiredService<IDbInitializer>();
        dbInitializer.Initialize();
    }
}
```

Додаток Б

Код усіх контролерів

CategoryController.cs

```

using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace CourseWorkWeb.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = SD.Role_Admin)]
    public class CategoryController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;

        public CategoryController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public IActionResult Index()
        {
            List<Category> CategoryList = _unitOfWork.Category.GetAll().ToList();

            return View(CategoryList);
        }

        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public IActionResult Create(Category category)
        {
            if (category.Name == category.DisplayOrder.ToString())
            {
                ModelState.AddModelError("Name", "The Display Order cannot exactly match the Name.");
            }

            if (ModelState.IsValid)
            {
                _unitOfWork.Category.Add(category);
                _unitOfWork.Save();
                TempData["success"] = "Category created successfully";
                return RedirectToAction("Index");
            }

            return View();
        }

        public IActionResult Edit(int? id)
        {
            if (id == null || id == 0)
            {
                return NotFound();
            }

            Category? categoryFromDb = _unitOfWork.Category.Get(u => u.Id == id);

            if (categoryFromDb == null)
            {
                return NotFound();
            }

            return View(categoryFromDb);
        }

        [HttpPost]
        public IActionResult Edit(Category category)
        {
            if (ModelState.IsValid)
            {
                _unitOfWork.Category.Update(category);
                _unitOfWork.Save();
                TempData["success"] = "Category updated successfully";
                return RedirectToAction("Index");
            }

            return View();
        }
    }
}

```

```

    }

    public IActionResult Delete(int? id)
    {
        if (id == null || id == 0)
        {
            return NotFound();
        }

        Category? categoryFromDb = _unitOfWork.Category.Get(u => u.Id == id);

        if (categoryFromDb == null)
        {
            return NotFound();
        }

        return View(categoryFromDb);
    }

    [HttpPost, ActionName("Delete")]
    public IActionResult DeletePOST(int? id)
    {
        Category? category = _unitOfWork.Category.Get(u => u.Id == id);
        if (category == null)
        {
            return NotFound();
        }

        _unitOfWork.Category.Remove(category);
        _unitOfWork.Save();
        TempData["success"] = "Category deleted successfully";
        return RedirectToAction("Index");
    }
}
}

```

CompanyController.cs

```

using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using CourseWorkWeb.Models.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace CourseWorkWeb.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = SD.Role_Admin)]
    public class CompanyController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;

        public CompanyController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public IActionResult Index()
        {
            List<Company> CompanyList = _unitOfWork.Company.GetAll().ToList();

            return View(CompanyList);
        }

        public IActionResult Upsert(int? Id)
        {
            if (Id == null || Id == 0)
            {
                //create
                return View(new Company());
            }
            else
            {
                //update
                Company company = _unitOfWork.Company.Get(u => u.Id == Id);
                return View(company);
            }
        }

        [HttpPost]
        public IActionResult Upsert(Company company)
        {
            if (ModelState.IsValid)

```

```

        {
            if (company.Id == 0)
            {
                _unitOfWork.Company.Add(company);
            }
            else
            {
                _unitOfWork.Company.Update(company);
            }

            _unitOfWork.Save();
            TempData["success"] = "Company created successfully";
            return RedirectToAction("Index");
        }
        else
        {
            return View(company);
        }
    }

    public IActionResult Delete(int? Id)
    {
        if (Id == null || Id == 0)
        {
            return NotFound();
        }

        Company? companyFromDb = _unitOfWork.Company.Get(u => u.Id == Id);

        if (companyFromDb == null)
        {
            return NotFound();
        }

        return View(companyFromDb);
    }

    [HttpPost, ActionName("Delete")]
    public IActionResult DeletePOST(int? Id)
    {
        Company? company = _unitOfWork.Company.Get(u => u.Id == Id);
        if (company == null)
        {
            return NotFound();
        }

        _unitOfWork.Company.Remove(company);
        _unitOfWork.Save();
        TempData["success"] = "Company deleted successfully";
        return RedirectToAction("Index");
    }
}

```

OrderController.cs

```

using System.Security.Claims;
using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using CourseWorkWeb.Models.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Stripe;
using Stripe.Checkout;
using Stripe.Climate;

namespace CourseWorkWeb.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize]
    public class OrderController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;
        [BindProperty] public OrderVM orderVM { get; set; }

        public OrderController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public IActionResult Details(int? orderId)
        {

```

```

        orderVM = new OrderVM();
        orderVM.OrderHeader = _unitOfWork.OrderHeader
            .Get(u => u.Id == orderId, includeProperties: "ApplicationUser");
        orderVM.OrderDetail = _unitOfWork.OrderDetail
            .GetAll(u => u.OrderHeaderId == orderId, includeProperties: "Product");
        return View(orderVM);
    }

    public IActionResult Index()
    {
        List<OrderHeader> orderHeaders;
        if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
        {
            orderHeaders = _unitOfWork.OrderHeader.GetAll(includeProperties: "ApplicationUser").ToList();
        }
        else
        {
            var claimsIdentity = (ClaimsIdentity)User.Identity;
            var userId = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier).Value;
            orderHeaders = _unitOfWork.OrderHeader
                .GetAll(u => u.ApplicationUserId == userId, includeProperties: "ApplicationUser").ToList();
        }

        List<OrderVM> orderVms = new List<OrderVM>();
        foreach (var orderHeader in orderHeaders)
        {
            OrderVM orderVM = new OrderVM();
            orderVM.OrderHeader = orderHeader;
            orderVM.OrderDetail = _unitOfWork.OrderDetail
                .GetAll(u => u.OrderHeaderId == orderHeader.Id, includeProperties: "Product");
            orderVms.Add(orderVM);
        }

        return View(orderVms);
    }

    [HttpPost]
    [Authorize(Roles = SD.Role_Admin + "," + SD.Role_Employee)]
    public IActionResult UpdateOrderDetail()
    {
        var orderHeaderFromDb = _unitOfWork.OrderHeader.Get(u => u.Id == orderVM.OrderHeader.Id);

        orderHeaderFromDb.Name = orderVM.OrderHeader.Name;
        orderHeaderFromDb.PhoneNumber = orderVM.OrderHeader.PhoneNumber;
        orderHeaderFromDb.StreetAddress = orderVM.OrderHeader.StreetAddress;
        orderHeaderFromDb.City = orderVM.OrderHeader.City;
        orderHeaderFromDb.State = orderVM.OrderHeader.State;
        orderHeaderFromDb.PostalCode = orderVM.OrderHeader.PostalCode;

        if (!string.IsNullOrEmpty(orderVM.OrderHeader.Carrier))
        {
            orderHeaderFromDb.Carrier = orderVM.OrderHeader.Carrier;
        }

        if (!string.IsNullOrEmpty(orderVM.OrderHeader.Trackingnumber))
        {
            orderHeaderFromDb.Trackingnumber = orderVM.OrderHeader.Trackingnumber;
        }

        _unitOfWork.OrderHeader.Update(orderHeaderFromDb);
        _unitOfWork.Save();

        TempData["Success"] = "Order Details Updated Successfully.";

        return RedirectToAction(nameof(Details), new { orderId = orderHeaderFromDb.Id });
    }

    [HttpPost]
    [Authorize(Roles = SD.Role_Admin + "," + SD.Role_Employee)]
    public IActionResult StartProcessing()
    {
        _unitOfWork.OrderHeader.UpdateStatus(orderVM.OrderHeader.Id, SD.StatusInProgress);
        _unitOfWork.Save();
        TempData["Success"] = "Order Details Updated Successfully.";
        return RedirectToAction(nameof(Details), new { orderId = orderVM.OrderHeader.Id });
    }

    [HttpPost]
    [Authorize(Roles = SD.Role_Admin + "," + SD.Role_Employee)]
    public IActionResult ShipOrder()
    {
        var orderHeader = _unitOfWork.OrderHeader.Get(u => u.Id == orderVM.OrderHeader.Id);
        orderHeader.Trackingnumber = orderVM.OrderHeader.Trackingnumber;
        orderHeader.Carrier = orderVM.OrderHeader.Carrier;
        orderHeader.OrderStatus = SD.StatusShipped;
        orderHeader.ShippingDate = DateTime.Now;
        if (orderHeader.PaymentStatus == SD.PaymentStatusDelayedPayment)
    }

```

```

        {
            orderHeader.PaymentDueDate = DateTime.Now.AddDays(30);
        }
        _unitOfWork.OrderHeader.Update(orderHeader);
        _unitOfWork.Save();
        TempData["Success"] = "Order Shipped Successfully.";
        return RedirectToAction(nameof(Details), new { orderId = orderVM.OrderHeader.Id });
    }

    [HttpPost]
    [Authorize(Roles = SD.Role_Admin + "," + SD.Role_Employee)]
    public IActionResult CancelOrder()
    {
        var orderHeader = _unitOfWork.OrderHeader.Get(u => u.Id == orderVM.OrderHeader.Id);

        if (orderHeader.PaymentStatus == SD.PaymentStatusApproved)
        {
            var options = new RefundCreateOptions
            {
                Reason = RefundReasons.RequestedByCustomer,
                PaymentIntent = orderHeader.PaymentIntentId
            };
            var service = new RefundService();
            service.Create(options);

            _unitOfWork.OrderHeader.UpdateStatus(orderHeader.Id, SD.StatusCancelled, SD.StatusRefunded);
        }
        else
        {
            _unitOfWork.OrderHeader.UpdateStatus(orderHeader.Id, SD.StatusCancelled, SD.StatusCancelled);
        }
        _unitOfWork.Save();
        TempData["Success"] = "Order Cancelled Successfully.";
        return RedirectToAction(nameof(Details), new { orderId = orderVM.OrderHeader.Id });
    }

    [ActionName("Details")]
    [HttpPost]
    public IActionResult Details_PAY_NOW()
    {
        orderVM.OrderHeader = _unitOfWork.OrderHeader
            .Get(u => u.Id == orderVM.OrderHeader.Id, includeProperties: "ApplicationUser");
        orderVM.OrderDetail = _unitOfWork.OrderDetail
            .GetAll(u => u.OrderHeaderId == orderVM.OrderHeader.Id, includeProperties: "Product");

        var domain = "https://localhost:7217/";
        var options = new Stripe.Checkout.SessionCreateOptions
        {
            SuccessUrl = domain + $"admin/order/PaymentConfirmation?orderHeaderId={orderVM.OrderHeader.Id}",
            CancelUrl = domain + $"admin/order/details?orderId={orderVM.OrderHeader.Id}",
            LineItems = new List<Stripe.Checkout.SessionLineItemOptions>(),
            Mode = "payment",
        };

        foreach (var item in orderVM.OrderDetail)
        {
            var SessionLineitem = new Stripe.Checkout.SessionLineItemOptions
            {
                PriceData = new Stripe.Checkout.SessionLineItemPriceDataOptions
                {
                    UnitAmount = (long)(item.Price * 100),
                    Currency = "uah",
                    ProductData = new SessionLineItemPriceDataProductDataOptions
                    {
                        Name = item.Product.Title
                    }
                },
                Quantity = item.Count
            };
            options.LineItems.Add(SessionLineitem);
        }
        var service = new Stripe.Checkout.SessionService();
        Session session = service.Create(options);
        _unitOfWork.OrderHeader.UpdateStripePaymentID(orderVM.OrderHeader.Id, session.Id, session.PaymentIntentId);
        _unitOfWork.Save();
        Response.Headers.Add("Location", session.Url);
        return new StatusCodeResult(303);
    }

    public IActionResult PaymentConfirmation(int orderHeaderId)
    {
        OrderHeader orderHeader =
            _unitOfWork.OrderHeader.Get(u => u.Id == orderHeaderId);
        if (orderHeader.PaymentStatus == SD.PaymentStatusDelayedPayment)
        {
            // this is an order by company

```

```

        var service = new SessionService();
        Session session = service.Get(orderHeader.SessionId);

        if (session.PaymentStatus.ToLower() == "paid")
        {
            _unitOfWork.OrderHeader.UpdateStripePaymentID(orderHeaderId,
                session.Id, session.PaymentIntentId);
            _unitOfWork.OrderHeader.UpdateStatus(orderHeaderId, orderHeader.OrderStatus,
                SD.PaymentStatusApproved);
            _unitOfWork.Save();
        }
    }
    return View(orderHeaderId);
}
}
}
}

```

ProductController.cs

```

using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using CourseWorkWeb.Models.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace CourseWorkWeb.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = SD.Role_Admin)]
    public class ProductController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;
        private readonly IWebHostEnvironment _webHostEnvironment;

        public ProductController(IUnitOfWork unitOfWork, IWebHostEnvironment webHostEnvironment)
        {
            _unitOfWork = unitOfWork;
            _webHostEnvironment = webHostEnvironment;
        }

        public IActionResult Index()
        {
            List<Product> productList = _unitOfWork.Product.GetAll(includeProperties: "Category").ToList();

            return View(productList);
        }

        public IActionResult Upsert(int? id)
        {
            ProductVM productVm = new()
            {
                CategoryList = _unitOfWork.Category
                    .GetAll().Select(u => new SelectListItem
                    {
                        Text = u.Name,
                        Value = u.Id.ToString()
                    }),
                Product = new Product()
            };
            if (id == null || id == 0)
            {
                //create
                return View(productVm);
            }
            else
            {
                //update
                productVm.Product = _unitOfWork.Product.Get(u => u.Id == id);
                return View(productVm);
            }
        }

        [HttpPost]
        public IActionResult Upsert(ProductVM productVM, IFormFile? file)
        {
            if (ModelState.IsValid)
            {
                string wwwRootPath = _webHostEnvironment.WebRootPath;
                if (file != null)
                {
                    string fileName = Guid.NewGuid().ToString() + Path.GetExtension(file.FileName);
                    string productPath = Path.Combine(wwwRootPath, @"images\product");

                    if (!string.IsNullOrEmpty(productVM.Product.ImageUrl))

```



```

        {
            //delete the old image
            var OldImagePath =
                Path.Combine(wwwRootPath, productVM.Product.ImageUrl.TrimStart('\'));

            if (System.IO.File.Exists(OldImagePath))
            {
                System.IO.File.Delete(OldImagePath);
            }
        }

        using (var fileStream = new FileStream(Path.Combine(productPath, fileName), FileMode.Create))
        {
            file.CopyTo(fileStream);
        }

        productVM.Product.ImageUrl = @"\images\product\" + fileName;
    }

    if (productVM.Product.Id == 0)
    {
        _unitOfWork.Product.Add(productVM.Product);
    }
    else
    {
        _unitOfWork.Product.Update(productVM.Product);
    }

    _unitOfWork.Save();
    TempData["success"] = "Product created successfully";
    return RedirectToAction("Index");
}

return View();
}

public IActionResult Delete(int? id)
{
    if (id == null || id == 0)
    {
        return NotFound();
    }

    Product? productFromDb = _unitOfWork.Product.Get(u => u.Id == id);

    if (productFromDb == null)
    {
        return NotFound();
    }

    return View(productFromDb);
}

[HttpPost, ActionName("Delete")]
public IActionResult DeletePOST(int? id)
{
    Product? product = _unitOfWork.Product.Get(u => u.Id == id);
    if (product == null)
    {
        return NotFound();
    }

    _unitOfWork.Product.Remove(product);
    _unitOfWork.Save();
    TempData["success"] = "Product deleted successfully";
    return RedirectToAction("Index");
}
}
}

```

UserController.cs

```

using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using CourseWorkWeb.Models.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkWeb.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = SD.Role_Admin)]
    public class UserController : Controller
    {
    }
}

```

```

{
    private readonly ApplicationDbContext _context;

    public UserController(ApplicationDbContext context, IUnitOfWork unitOfWork)
    {
        _context = context;
    }

    public IActionResult Index()
    {
        List<ApplicationUser> applicationUsers = _context.ApplicationUsers.Include(u => u.Company).ToList();

        var userRoles = _context.UserRoles.ToList();

        var roles = _context.Roles.ToList();

        foreach (var user in applicationUsers)
        {
            var roleId = userRoles.FirstOrDefault(u => u.UserId == user.Id).RoleId;
            user.Role = roles.FirstOrDefault(u => u.Id == roleId).Name;
        }

        return View(applicationUsers);
    }

    [HttpPost]
    [Authorize(Roles = SD.Role_Admin + ", " + SD.Role_Employee)]
    public IActionResult LockUnlock(string id)
    {
        var objFromDb = _context.ApplicationUsers.FirstOrDefault(u => u.Id == id);
        if (objFromDb is null)
        {
            TempData["Error"] = "Error in unlock | lock !";
        }

        if (objFromDb.LockoutEnd != null && objFromDb.LockoutEnd > DateTime.Now)
        {
            // user is currently locked and we need to unlock them
            objFromDb.LockoutEnd = DateTime.Now;
            TempData["Success"] = "Unlock user successfully!";
        }
        else
        {
            // user is currently locked and we need to unlock them
            objFromDb.LockoutEnd = DateTimeOffset.Now.AddYears(1000);
            TempData["Error"] = "User locked successfully!";
        }

        _context.SaveChanges();

        return RedirectToAction(nameof(Index));
    }
}
}

```

CartController.cs

```

using CourseWorkWeb.Data;
using CourseWorkWeb.Models;
using CourseWorkWeb.Models.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Stripe.Checkout;

namespace CourseWorkWeb.Areas.Customer.Controllers
{
    [Area("Customer")]
    [Authorize]
    public class CartController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;
        [BindProperty] public ShoppingCartVM ShoppingCartVM { get; set; }

        public CartController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public IActionResult Index()
        {
            var claimsIdentity = (ClaimsIdentity)User.Identity;
            var userId = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier).Value;

```

```

        ShoppingCartVM = new()
        {
            ShoppingCartList = _unitOfWork.ShoppingCart.GetAll(u => u.ApplicationUserId == userId,
                includeProperties: "Product"),
            OrderHeader = new()
        };

        foreach (var cart in ShoppingCartVM.ShoppingCartList)
        {
            cart.Price = GetPriceBasedOnQuantity(cart);
            ShoppingCartVM.OrderHeader.OrderTotal += (cart.Price * cart.Count);
        }

        return View(ShoppingCartVM);
    }

    public IActionResult Summary()
    {
        var claimsIdentity = (ClaimsIdentity)User.Identity;
        var userId = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier).Value;

        ShoppingCartVM = new()
        {
            ShoppingCartList = _unitOfWork.ShoppingCart.GetAll(u => u.ApplicationUserId == userId,
                includeProperties: "Product"),
            OrderHeader = new()
        };

        ShoppingCartVM.OrderHeader.ApplicationUser = _unitOfWork.ApplicationUser.Get(u => u.Id == userId);

        ShoppingCartVM.OrderHeader.Name = ShoppingCartVM.OrderHeader.ApplicationUser.Name;
        ShoppingCartVM.OrderHeader.PhoneNumber = ShoppingCartVM.OrderHeader.ApplicationUser.PhoneNumber;
        ShoppingCartVM.OrderHeader.StreetAddress = ShoppingCartVM.OrderHeader.ApplicationUser.StreetAddress;
        ShoppingCartVM.OrderHeader.City = ShoppingCartVM.OrderHeader.ApplicationUser.City;
        ShoppingCartVM.OrderHeader.State = ShoppingCartVM.OrderHeader.ApplicationUser.State;
        ShoppingCartVM.OrderHeader.PostalCode = ShoppingCartVM.OrderHeader.ApplicationUser.PostalCode;

        foreach (var cart in ShoppingCartVM.ShoppingCartList)
        {
            cart.Price = GetPriceBasedOnQuantity(cart);
            ShoppingCartVM.OrderHeader.OrderTotal += (cart.Price * cart.Count);
        }

        return View(ShoppingCartVM);
    }

    [HttpPost]
    [ActionName("Summary")]
    public IActionResult SummaryPOST()
    {
        var claimsIdentity = (ClaimsIdentity)User.Identity;
        var userId = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier).Value;

        ShoppingCartVM.ShoppingCartList = _unitOfWork.ShoppingCart.GetAll(u => u.ApplicationUserId == userId,
            includeProperties: "Product");

        ShoppingCartVM.OrderHeader.OrderTime = DateTime.Now;
        ShoppingCartVM.OrderHeader.ApplicationUserId = userId;

        ApplicationUser applicationUser = _unitOfWork.ApplicationUser.Get(u => u.Id == userId);

        foreach (var cart in ShoppingCartVM.ShoppingCartList)
        {
            cart.Price = GetPriceBasedOnQuantity(cart);
            ShoppingCartVM.OrderHeader.OrderTotal += (cart.Price * cart.Count);
        }

        if (applicationUser.CompanyId.GetValueOrDefault() == 0)
        {
            // it is a regular customer
            ShoppingCartVM.OrderHeader.PaymentStatus = SD.PaymentStatusPending;
            ShoppingCartVM.OrderHeader.OrderStatus = SD.StatusPending;
        }
        else
        {
            // it is a company user
            ShoppingCartVM.OrderHeader.PaymentStatus = SD.PaymentStatusDelayedPayment;
            ShoppingCartVM.OrderHeader.OrderStatus = SD.StatusApproved;
        }

        _unitOfWork.OrderHeader.Add(ShoppingCartVM.OrderHeader);
        _unitOfWork.Save();

        foreach (var cart in ShoppingCartVM.ShoppingCartList)
        {

```

```

        OrderDetail orderDetail = new()
        {
            ProductId = cart.ProductId,
            OrderHeaderId = ShoppingCartVM.OrderHeader.Id,
            Price = cart.Price,
            Count = cart.Count
        };
        _unitOfWork.OrderDetail.Add(orderDetail);
        _unitOfWork.Save();
    }

    if (applicationUser.CompanyId.GetValueOrDefault() == 0)
    {
        // it is a regular customer account and we need to capture payment
        // stripe logic
        var domain = "https://localhost:7217/";
        var options = new Stripe.Checkout.SessionCreateOptions
        {
            SuccessUrl = domain + $"customer/cart/OrderConfirmation?id={ShoppingCartVM.OrderHeader.Id}",
            CancelUrl = domain + "customer/cart/index",
            LineItems = new List<Stripe.Checkout.SessionLineItemOptions>(),
            Mode = "payment",
        };

        foreach (var item in ShoppingCartVM.ShoppingCartList)
        {
            var SessionLineitem = new Stripe.Checkout.SessionLineItemOptions
            {
                PriceData = new Stripe.Checkout.SessionLineItemPriceDataOptions
                {
                    UnitAmount = (long)(item.Price * 100),
                    Currency = "uah",
                    ProductData = new SessionLineItemPriceDataProductDataOptions
                    {
                        Name = item.Product.Title
                    }
                },
                Quantity = item.Count
            };
            options.LineItems.Add(SessionLineitem);
        }

        var service = new Stripe.Checkout.SessionService();
        Session session = service.Create(options);
        _unitOfWork.OrderHeader.UpdateStripePaymentID(ShoppingCartVM.OrderHeader.Id, session.Id,
            session.PaymentIntentId);
        _unitOfWork.Save();

        Response.Headers.Add("Location", session.Url);
        return new StatusCodeResult(303);
    }

    return RedirectToAction(nameof(OrderConfirmation), new { id = ShoppingCartVM.OrderHeader.Id });
}

public IActionResult OrderConfirmation(int id)
{
    OrderHeader orderHeader =
        _unitOfWork.OrderHeader.Get(u => u.Id == id, includeProperties: "ApplicationUser");

    if (orderHeader.PaymentStatus != SD.PaymentStatusDelayedPayment)
    {
        // this is an order by customer

        var service = new SessionService();
        Session session = service.Get(orderHeader.SessionId);

        if (session.PaymentStatus.ToLower() == "paid")
        {
            _unitOfWork.OrderHeader.UpdateStripePaymentID(id,
                session.Id, session.PaymentIntentId);

            _unitOfWork.OrderHeader.UpdateStatus(id, SD.StatusApproved, SD.PaymentStatusApproved);
            _unitOfWork.Save();
        }
    }

    List<ShoppingCart> shoppingCarts = _unitOfWork.ShoppingCart
        .GetAll(u => u.ApplicationUserId == orderHeader.ApplicationUserId).ToList();

    _unitOfWork.ShoppingCart.RemoveRange(shoppingCarts);
    _unitOfWork.Save();

    return View(id);
}

```

```

public IActionResult Plus(int cartId)
{
    var cartFromDb = _unitOfWork.ShoppingCart.Get(u => u.Id == cartId);
    cartFromDb.Count += 1;
    _unitOfWork.ShoppingCart.Update(cartFromDb);
    _unitOfWork.Save();
    return RedirectToAction(nameof(Index));
}

public IActionResult Minus(int cartId)
{
    var cartFromDb = _unitOfWork.ShoppingCart.Get(u => u.Id == cartId);
    if (cartFromDb.Count <= 1)
    {
        //remove from cart
        _unitOfWork.ShoppingCart.Remove(cartFromDb);
    }
    else
    {
        cartFromDb.Count -= 1;
        _unitOfWork.ShoppingCart.Update(cartFromDb);
    }

    _unitOfWork.Save();
    return RedirectToAction(nameof(Index));
}

public IActionResult Remove(int cartId)
{
    var cartFromDb = _unitOfWork.ShoppingCart.Get(u => u.Id == cartId);
    _unitOfWork.ShoppingCart.Remove(cartFromDb);
    _unitOfWork.Save();
    return RedirectToAction(nameof(Index));
}

private double GetPriceBasedOnQuantity(ShoppingCart shoppingCart)
{
    if (shoppingCart.Count <= 50)
    {
        return shoppingCart.Product.Price;
    }

    if (shoppingCart.Count <= 100)
    {
        return shoppingCart.Product.Price50;
    }

    return shoppingCart.Product.Price100;
}
}
}

```

HomeController.cs

```

using CourseWorkWeb.Models;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;
using CourseWorkWeb.Data;
using Microsoft.AspNetCore.Authorization;

namespace CourseWorkWeb.Areas.Customer.Controllers
{
    [Area("Customer")]
    public class HomeController : Controller
    {
        private readonly IUnitOfWork _unitOfWork;

        public HomeController(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public IActionResult Index()
        {
            IEnumerable<Product> productList = _unitOfWork.Product.GetAll(includeProperties: "Category");
            return View(productList);
        }

        public IActionResult Details(int productId)
        {
            ShoppingCart cart = new()

```

```

        {
            Product = _unitOfWork.Product.Get(u => u.Id == productId, includeProperties: "Category"),
            Count = 1,
            ProductId = productId
        };
        return View(cart);
    }

    [HttpPost]
    [Authorize]
    public IActionResult Details(ShoppingCart shoppingCart)
    {
        var claimsIdentity = (ClaimsIdentity)User.Identity;
        var userId = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier).Value;
        shoppingCart.ApplicationUserId = userId;

        ShoppingCart cartFromDb = _unitOfWork.ShoppingCart
            .Get(
                u => u.ApplicationUserId == userId &&
                u.ProductId == shoppingCart.ProductId);

        if (cartFromDb != null)
        {
            // shopping cart exists
            cartFromDb.Count += shoppingCart.Count;
            _unitOfWork.ShoppingCart.Update(cartFromDb);
        }
        else
        {
            // add cart record
            _unitOfWork.ShoppingCart.Add(shoppingCart);
        }

        _unitOfWork.Save();

        return RedirectToAction(nameof(Index));
    }
}

```

Додаток В

Код всіх класів-моделей та моделей представлень

ApplicationUser.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using Microsoft.Build.Framework;

namespace CourseWorkWeb.Models
{
    public class ApplicationUser : IdentityUser
    {
        [Required]
        public string Name { get; set; }

        public string? StreetAddress { get; set; }
        public string? City { get; set; }
        public string? State { get; set; }
        public string? PostalCode { get; set; }
        public int? CompanyId { get; set; }
        [ForeignKey("CompanyId")]
        [ValidateNever]
        public Company? Company { get; set; }
        [NotMapped]
        public string Role { get; set; }
    }
}
```

Category.cs

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace CourseWorkWeb.Models;

public class Category
{
    [Key] // Primary key
    public int Id { get; set; }
    [Required]
    [MaxLength(30)]
    [DisplayName("Category Name")]
    public string Name { get; set; }
    [DisplayName("Display Order")]
    [Range(1,100,ErrorMessage = "Display Order must be between 1-100")]
    public int DisplayOrder { get; set; }
}
```

Company.cs

```
using Microsoft.Build.Framework;

namespace CourseWorkWeb.Models;

public class Company
{
    public int Id { get; set; }
    [Required]
    public string Name { get; set; }
    public string? StreetAddress { get; set; }
    public string? City { get; set; }
    public string? State { get; set; }
    public string? PostalCode { get; set; }
    public string? PhoneNumber { get; set; }
}
```

OrderDetail.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.Build.Framework;

namespace CourseWorkWeb.Models;

public class OrderDetail
{
    public int Id { get; set; }
    [Required]
    public int OrderHeaderId { get; set; }
    [ForeignKey("OrderHeaderId")]
    [ValidateNever]
    public OrderHeader OrderHeader { get; set; }

    [Required]
    public int ProductId { get; set; }
    [ForeignKey("ProductId")]
    [ValidateNever]
    public Product Product { get; set; }

    public int Count { get; set; }
    public double Price { get; set; }
}
```

OrderHeader.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.Build.Framework;

namespace CourseWorkWeb.Models;

public class OrderHeader
{
    public int Id { get; set; }
    public string ApplicationUserId { get; set; }
    [ForeignKey("ApplicationUserId")]
    [ValidateNever]
    public ApplicationUser ApplicationUser { get; set; }

    public DateTime OrderTime { get; set; }
    public DateTime ShippingDate { get; set; }
    public double OrderTotal { get; set; }

    public string? OrderStatus { get; set; }
    public string? PaymentStatus { get; set; }
    public string? Trackingnumber { get; set; }
    public string? Carrier { get; set; }

    public DateTime PaymentDate { get; set; }
    public DateTime PaymentDueDate { get; set; }

    public string? SessionId { get; set; }
    public string? PaymentIntendId { get; set; }

    [Required]
    public string PhoneNumber { get; set; }
    [Required]
    public string StreetAddress { get; set; }
    [Required]
    public string City { get; set; }

    [Required]
    public string State { get; set; }
    [Required]
    public string PostalCode { get; set; }
    [Required]
    public string Name { get; set; }
}
```


Product.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

namespace CourseWorkWeb.Models;

public class Product
{
    [Key]
    public int Id { get; set; }
    [Required]
    public string Title { get; set; }
    [Required]
    public string Description { get; set; }
    [Required]
    public string ISBN { get; set; }
    [Required]
    public string Author { get; set; }
    [Required]
    [Display(Name = "List Price")]
    [Range(1, 1000)]
    public double ListPrice { get; set; }

    [Required]
    [Display(Name = "Price for 1-50")]
    [Range(1, 1000)]
    public double Price { get; set; }

    [Required]
    [Display(Name = "Price for 50+")]
    [Range(1, 1000)]
    public double Price50 { get; set; }

    [Required]
    [Display(Name = "Price for 100+")]
    [Range(1, 1000)]
    public double Price100 { get; set; }
    public int CategoryId { get; set; }
    [ForeignKey("CategoryId")]
    [ValidateNever]
    public Category Category { get; set; }
    [ValidateNever]
    public string? ImageUrl { get; set; }
}
```

ShoppingCart.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;

namespace CourseWorkWeb.Models;

public class ShoppingCart
{
    public int Id { get; set; }
    public int ProductId { get; set; }
    [ForeignKey("ProductId")]
    [ValidateNever]
    public Product Product { get; set; }
    [Range(1, 1000, ErrorMessage = "Please enter a value between 1 and 1000")]
    public int Count { get; set; }

    public string ApplicationUserId { get; set; }
    [ForeignKey("ApplicationUserId")]
    [ValidateNever]
    public ApplicationUser ApplicationUser { get; set; }

    [NotMapped]
    public double Price { get; set; }
}
```

OrderVM.cs

```
namespace CourseWorkWeb.Models.ViewModels;

public class OrderVM
{
    public OrderHeader OrderHeader { get; set; }
    public IEnumerable<OrderDetail> OrderDetail { get; set; }
}
```

ProductVM.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
using Microsoft.AspNetCore.Mvc.Rendering;

namespace CourseWorkWeb.Models.ViewModels;

public class ProductVM
{
    public Product Product { get; set; }
    [ValidateNever]
    public IEnumerable<SelectListItem> CategoryList { get; set; }
}
```

ShoppingCartVM.cs

```
namespace CourseWorkWeb.Models.ViewModels
{
    public class ShoppingCartVM
    {
        public IEnumerable<ShoppingCart> ShoppingCartList { get; set; }
        public OrderHeader OrderHeader { get; set; }
    }
}
```

Додаток Г

Всі файли з папки Data

ApplicationDbContext.cs

```

using CourseWorkWeb.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkWeb.Data;

public class ApplicationDbContext : IdentityDbContext<IdentityUser>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }

    public DbSet<Category> Categories { get; set; }
    public DbSet<Product> Products { get; set; }
    public DbSet<Company> Companies { get; set; }
    public DbSet<ShoppingCart> ShoppingCarts { get; set; }
    public DbSet<ApplicationUser> ApplicationUsers { get; set; }
    public DbSet<OrderDetail> OrderDetails { get; set; }
    public DbSet<OrderHeader> OrderHeaders { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Category>().HasData(
            new Category { Id = 1, Name = "Фантастика", DisplayOrder = 1 },
            new Category { Id = 2, Name = "Психологія", DisplayOrder = 2 },
            new Category { Id = 3, Name = "Підручники", DisplayOrder = 3 },
            new Category { Id = 4, Name = "Сучасна проза", DisplayOrder = 4 },
            new Category { Id = 5, Name = "Любовні романи", DisplayOrder = 5 }
        );

        modelBuilder.Entity<Company>().HasData(
            new Company
            {
                Id = 1,
                Name = "Видавництво Старого Лева",
                StreetAddress = "Katerininska 2",
                City = "Odesa",
                PostalCode = "65025",
                State = "Odesa",
                PhoneNumber = "0675531688"
            },
            new Company
            {
                Id = 2,
                Name = "Book Chef",
                StreetAddress = "Verbova 17-a",
                City = "Kyiv",
                PostalCode = "62654",
                State = "Kyiv",
                PhoneNumber = "0504048100"
            },
            new Company
            {
                Id = 3,
                Name = "Vivat",
                StreetAddress = "М. Homonenko 10",
                City = "Kharkiv",
                PostalCode = "66934",
                State = "Kharkiv",
                PhoneNumber = "080020102"
            }
        );

        modelBuilder.Entity<Product>().HasData(
            new Product
            {
                Id = 1,
                Title = "Кафе на краю світу - Стрелекі Дж. П.",
                Author = "Джон П. Стрелекі",
                Description =
                    "Зупиніться. Поставте на паузу щоденну метушню, вирвітьсся з її тенет і зазирніть до кафе на краю світу. Розгорніть меню й придивіться уважніше до останньої сторінки – там ви прочитаєте три запитання, від яких тікали все життя. Чому ви тут? Чи боїтеся ви смерті? Чи вдоволені ви?\n\nЦя книжка допоможе у пошуках відповідей. З нею ви запалите вогонь у душі, усвідомите, що цінне, а що – марнота, поглянете на життя під іншим кутом зору, незалежно від набутого досвіду. \nКафе

```

```

на краю світу\" – справжній видавничий феномен, бестселер поза часом, що надихнув мільйони людей у всьому світі на великі
зміни. Чарівна, легка й наснажлива, ця книжка вплине на ваш світогляд і назавжди змінить життя.",
ISBN = "978-966-982-061-7",
ListPrice = 200,
Price = 170,
Price50 = 165,
Price100 = 140,
CategoryId = 4,
ImageUrl = "https://content2.rozetka.com.ua/goods/images/big/129306820.png"
},
new Product
{
    Id = 2,
    Title = "Одна з дівчат",
    Author = "Кларк Люсі",
    Description =
        "Шість молодих жінок прибувають на грецький острів на дівич-вечір. Незабаром їхня подруга Лексі виходить
заміж за чоловіка, якого по-справжньому кохає. Залитий сонцем пляж, солодкі коктейлі, помаранчеві заходи сонця... Ці\nночі
на райському острові мають стати незабутніми.\n\nДівчата купуються під зоряним небом, насолоджуючись краєвидами та смачною
їжею, але насправді їхня дружба вдавана, а секрети обтяжливі. Зрештою одна з дівчат втрачає ореол таємничості – і все летить
шкереберть. Невже хтось із подруг хоче зруйнувати щасливе майбутнє Лексі з Едом? Але чому?\n\nПеред вами постануть питання
про жіночу дружбу, сексуальне насильство, стосунки між чоловіком та жінкою й відповідальність за власні вчинки.",
ISBN = "978-617-17-0202-8",
ListPrice = 400,
Price = 340,
Price50 = 320,
Price100 = 300,
CategoryId = 4,
ImageUrl = "https://content2.rozetka.com.ua/goods/images/big/360883197.jpg"
},
new Product
{
    Id = 3,
    Title = "Третій візит до кафе на краю світу",
    Author = "Стрелекі Джон",
    Description =
        "Кафе на краю світу допоможе тим, хто заблукав. Гроза й буря зводять двох таких подорожніх – Джона і
Ханну. Востаннє Джон був у цьому закладі десять років тому. Тепер він уже не наївний юнак, а людина, якій доводиться миритися
з плином часу. Ханна ж тут уперше; їй п'ятнадцять, і вона поки що боїться відповідати на запитання в незвичному меню: «Чому
ви тут?», «Чи боїтеся ви смерті?», «Чи вдоволені ви?». І Джонови, і Ханні, і читачам випаде нагода замислитися й переглянути
свої відповіді на них. Віднайти себе, зрозуміти, що з часом пріоритети змінюються, прийняти все, що підносить нам життя, та
мати сили зіткнутися з найболючішими темами – ось що дарує своїм відвідувачам кафе. Сподіваємося, ви ознайомилися з меню і
вже готові замовити смачну відповідь на те, що давно вас турбує...",
ISBN = "978-966-982-390-8",
ListPrice = 200,
Price = 170,
Price50 = 165,
Price100 = 140,
CategoryId = 4,
ImageUrl = "https://content1.rozetka.com.ua/goods/images/big/222409612.jpg"
},
new Product
{
    Id = 4,
    Title = "Мій коханий ворог",
    Author = "Саллі Торн",
    Description =
        "Люсі Гаттон ділить тісний робочий простір з Джошуа Темплменом, бридким типом, якого вона щиро
ненавидить. Можливо, тому що він копіює кожен її рух, сидючи за столом навпроти, а може, тому що він посіпака, який обоюдно
правила й алгоритми.\n\nДжошуа теж не в захваті від чарівної Люсі, яка має хронічну потребу в обожнюванні, тому вони ніяк не
можуть спрацюватися. Та одного дня їхній офісний конфлікт виходить на новий рівень – тепер вони удвох претендують на посаду
виконавчого директора. Якщо Люсі виграє, то стане Джошевою начальницею, а якщо програє, то звільниться. Тож вона стрімголов
кидається у бій.\n\nАле враз усе летить шкереберть. Спочатку цей мерзотник проникає в її сни, а потім у порожньому ліфті
Джошуа Темплмен її цілує.\n\nНевже за взаємною ненавистю ховалася закоханість? І чи вдасться Люсі після всього обійняти посаду
мрії?",
ISBN = "978-617-17-0128-1",
ListPrice = 400,
Price = 340,
Price50 = 325,
Price100 = 300,
CategoryId = 5,
ImageUrl = "https://content2.rozetka.com.ua/goods/images/big/352822283.jpg"
},
new Product
{
    Id = 5,
    Title = "Повернення до кафе на краю світу",
    Author = "Стрелекі Джон",
    Description =
        "Він заблукав і потрапив у маленьке затишне кафе «Чому ви тут?». Чоловік провів там усю ніч і полишив це
місце з новим поглядом на сенс буття загалом і на своє життя зокрема. Джон дуже здивувався, коли десятьма роками пізніше
знову опинився в цьому кафе. От тільки цього разу в меню були три нові запитання. І відповісти на них уже мала Джессіка –
випадкова відвідувачка, яка втратила життєвий орієнтир. Час, проведений у кафе, надихає Джона і Джессіку заглибитись у
себе, звиритись зі своїми цінностями й жити справжнім та насиченим життям. Події, описані в цій книжці, повертають читачів у
те місце, де завжди є смачна їжа і де ставлять запитання, які спонукають замислитися про найважливіше. Тож улаштовуйтеся
зручніше й насолоджуйтеся поверненням до кафе на краю світу.",
ISBN = "978-966-982-246-8",
ListPrice = 220,

```

```

Price = 180,
Price50 = 170,
Price100 = 155,
CategoryId = 4,
ImageUrl = "https://content.rozetka.com.ua/goods/images/big/168723969.jpg"
},
new Product
{
    Id = 6,
    Title = "Гіпотеза кохання",
    Author = "Алі Гейзелвуд",
    Description =
        "Олівія Сміт – біологиня, яка мріє не про романтичні стосунки, а про докторський ступінь. Але що робити, коли найліпша подруга не дає тобі спокою з отим коханням ні вдень, ні вночі? Звісно ж, знайти хлопця, який вдаватиме любчика. На цю роль годиться не абихто, а молодий і гарячий професор Карлсен: м-м-м, біологія, Стендфорд, поцілунки... Непоганий початок, погодьтеся! Олівія має чіткий план і не чекає від славного на всю кафедру Адама «Гедзя» Карлсена розуміння чи підтримки. Але після інциденту на великій науковій конференції Адам стає на її бік. І раптом дівчина помічає, що і її обранець не лише розумний, але й чуйний і врівноважений хлопець... із кубиками пресу (о боги!). Щойно маленький експеримент виходить з-під контролю, а теорія романтики розбивається на друзки, як Олівія розуміє: дослідження власного серця під мікроскопом буде складнішим за гіпотезу кохання!",
    ISBN = "978-617-170-010-9",
    ListPrice = 380,
    Price = 320,
    Price50 = 300,
    Price100 = 280,
    CategoryId = 4,
    ImageUrl = "https://content.rozetka.com.ua/goods/images/big/310134514.jpg"
},
new Product
{
    Id = 7,
    Title = "English Grammar in Use 5th Edition with Answers",
    Author = "Eaymond Murphy",
    Description =
        "English Grammar in Use - це musthave для кожного студента, вивчає англійську на будь-якому рівні (A1-C2).\n\nДопомоги вже більше 30 років знаходяться в топі продажів книг для вивчення англійської.\n\nАвтором двох перших рівнів серії є всесвітньо відомий Raymond Murphy.\n\nСерія розрахована для самонавчання студентів 9-11 класів, студентів університетів, дорослих і, в загальному, всіх, хто хоче вдосконалити свій рівень, розібратися в граматиці і раз і назавжди систематизувати свої знання. Підійде і для вчителів, які шукають додаткових завдань для урізноманітнення уроків. Книги містять чіткі доступні пояснення граматики з візуалізацією і схемами, безліч практичних вправ, структурованих за зростанням рівня складності і згрупованих за граматичним темам.",
    ISBN = "9781108457651",
    ListPrice = 890,
    Price = 760,
    Price50 = 730,
    Price100 = 700,
    CategoryId = 3,
    ImageUrl = "https://content2.rozetka.com.ua/goods/images/big/323205101.jpg"
},
new Product
{
    Id = 8,
    Title = "Людина в пошуках справжнього сенсу. Психолог у концтаборі",
    Author = "Віктор Франкл",
    Description =
        "Віктор Франкл – всесвітньо відомий психіатр, психотерапевт, філософ. 1942 року він потрапив до концтабору, де на нього чекали голод, приниження, хвороби, постійна загроза життю. Аналізуючи свою поведінку та поведінку інших в'язнів, Франкл віднайшов стратегії, що утримують людину над прірвою, захищають розум від божевілля та надають сенс життю. Свій хакливий досвід виживання він описав у книжці, яка допомогла мільйонам людей віднайти себе та змінити життя. Віктор Франкл доводить, що тільки-но людина знаходить сенс свого існування, вона отримує сили, щоб здолати будь-які випробування. Поради людини, яка зазнала нелюдських випробувань, варті того, щоб бути почутими.",
    ISBN = "978-617-12-8583-5",
    ListPrice = 220,
    Price = 150,
    Price50 = 120,
    Price100 = 100,
    CategoryId = 2,
    ImageUrl = "https://bookclub.ua/images/db/goods/k/37421_56153_k.jpg"
},
new Product
{
    Id = 9,
    Title = "1984",
    Author = "Джордж Орвелл",
    Description =
        "Джордж Орвелл (справжнє ім'я – Ерік Артур Блер) – англійський прозаїк, есеїст, журналіст і критик. Його творчість характеризується пронизливою соціальною критикою, опозиційністю до тоталітаризму та відвертою підтримкою демократичних прагнень людства. Автор та популяризатор в політичному дискурсі терміна «холодна війна».\n\n«1984» – без сумніву, найвідоміша в світі антиутопія, вперше опублікована в 1949 році. Книга оповідає про життя Вінстона Сміта, низькорангового члена Партії, який розчарований повсякденним життям та всюдишними очима Партії та її зловісного очільника – Старшого Брата. Старший Брат контролює всі аспекти життя: впровадив спрощений новоговір, намагаючись повністю задушити навіть можливість спротиву системі; криміналізував думкозлочини, щоб люди навіть помислити не сміли про бунт проти влади. Партія контролює все: що люди читають, говорять, роблять, погрожуючи відправляти неслухів до страхітливої «кімнати сто один». Орвелл надзвичайно виразно та яскраво досліджує теми контролю за засобами масової інформації, нагляду уряду, тоталітаризму та того, як диктатор може маніпулювати та контролювати історію, думки та життя таким чином, що жодній живій душі не вдасться уникнути невиспного ока Старшого Брата.",
    ISBN = "978-617-548-008-3",
    ListPrice = 180,

```

```

        Price = 150,
        Price50 = 120,
        Price100 = 100,
        CategoryId = 1,
        ImageUrl = "https://content1.rozetka.com.ua/goods/images/big/241506234.jpg"
    };
}
}
}

```

ApplicationUserRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class ApplicationUserRepository : Repository<ApplicationUser>, IApplicationUserRepository
{
    private ApplicationDbContext _context;
    public ApplicationUserRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }
}

```

CategoryRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class CategoryRepository : Repository<Category>, ICategoryRepository
{
    private ApplicationDbContext _context;
    public CategoryRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }

    public void Update(Category category)
    {
        _context.Categories.Update(category);
    }
}

```

CompanyRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class CompanyRepository : Repository<Company>, ICompanyRepository
{
    private ApplicationDbContext _context;
    public CompanyRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }

    public void Update(Company company)
    {
        _context.Companies.Update(company);
    }
}

```

DbInitializer.cs

```

using CourseWorkWeb.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkWeb.Data;

public class DbInitializer : IDbInitializer
{
    private readonly UserManager<IdentityUser> _userManager;
    private readonly RoleManager<IdentityRole> _roleManager;
    private readonly ApplicationDbContext _db;

    public DbInitializer(
        UserManager<IdentityUser> userManager,

```

```

RoleManager<IdentityRole> roleManager,
ApplicationDbContext db)
{
    _roleManager = roleManager;
    _userManager = userManager;
    _db = db;
}

public void Initialize()
{
    try
    {
        if (_db.Database.GetPendingMigrations().Count() > 0)
        {
            _db.Database.Migrate();
        }
    }
    catch (Exception ex)
    {
    }

    if (!_roleManager.RoleExistsAsync(SD.Role_Customer).GetAwaiter().GetResult())
    {
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Customer)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Employee)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Admin)).GetAwaiter().GetResult();
        _roleManager.CreateAsync(new IdentityRole(SD.Role_Company)).GetAwaiter().GetResult();

        _userManager.CreateAsync(new ApplicationUser
        {
            UserName = "admin@gmail.com",
            Email = "admin@gmail.com",
            Name = "Vladyslav Keidaliuk",
            PhoneNumber = "0986547835",
            StreetAddress = "Akademika Filatova 8",
            State = "Ukraine",
            PostalCode = "65000",
            City = "Odesa"
        }, "Password_123").GetAwaiter().GetResult();

        _userManager.CreateAsync(new ApplicationUser
        {
            UserName = "customer@gmail.com",
            Email = "customer@gmail.com",
            Name = "Basic Customer",
            PhoneNumber = "0963951645",
            StreetAddress = "Central Street 9",
            State = "Ukraine",
            PostalCode = "63651",
            City = "Kyiv"
        }, "Password_123").GetAwaiter().GetResult();

        _userManager.CreateAsync(new ApplicationUser
        {
            UserName = "company@gmail.com",
            Email = "company@gmail.com",
            Name = "My Company",
            PhoneNumber = "04863128321",
            StreetAddress = "Lvivska 54",
            State = "Ukraine",
            PostalCode = "66984",
            City = "Kharkiv",
            CompanyId = 3
        }, "Password_123").GetAwaiter().GetResult();

        _userManager.CreateAsync(new ApplicationUser
        {
            UserName = "employee@gmail.com",
            Email = "employee@gmail.com",
            Name = "Standart Employee",
            PhoneNumber = "0954629678",
            StreetAddress = "Last street 24",
            State = "Ukraine",
            PostalCode = "69264",
            City = "Mykolaiv"
        }, "Password_123").GetAwaiter().GetResult();

        ApplicationUser admin = _db.ApplicationUsers.FirstOrDefault(u => u.Email == "admin@gmail.com");
        _userManager.AddToRoleAsync(admin, SD.Role_Admin).GetAwaiter().GetResult();

        ApplicationUser customer = _db.ApplicationUsers.FirstOrDefault(u => u.Email == "customer@gmail.com");
        _userManager.AddToRoleAsync(customer, SD.Role_Customer).GetAwaiter().GetResult();

        ApplicationUser company = _db.ApplicationUsers.FirstOrDefault(u => u.Email == "company@gmail.com");
        _userManager.AddToRoleAsync(company, SD.Role_Company).GetAwaiter().GetResult();

        ApplicationUser employee = _db.ApplicationUsers.FirstOrDefault(u => u.Email == "employee@gmail.com");
        _userManager.AddToRoleAsync(employee, SD.Role_Employee).GetAwaiter().GetResult();
    }
}

```

```

        return;
    }
}

```

EmailSender.cs

```

using Microsoft.AspNetCore.Identity.UI.Services;

namespace CourseWorkWeb.Data
{
    public class EmailSender : IEmailSender
    {
        public Task SendEmailAsync(string email, string subject, string htmlMessage)
        {
            // Logic to send Email
            return Task.CompletedTask;
        }
    }
}

```

IApplicationUserRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface IApplicationUserRepository : IRepository<ApplicationUser>
{
}

```

ICategoryRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface ICategoryRepository : IRepository<Category>
{
    void Update(Category category);
}

```

ICompanyRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface ICompanyRepository : IRepository<Company>
{
    void Update(Company company);
}

```

IDbInitializer.cs

```

namespace CourseWorkWeb.Data;

public interface IDbInitializer
{
    void Initialize();
}

```

IOrderDetailRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface IOrderDetailRepository : IRepository<OrderDetail>
{
    void Update(OrderDetail orderDetail);
}

```


IOrderHeaderRepository.cs

```
using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface IOrderHeaderRepository : IRepository<OrderHeader>
{
    void Update(OrderHeader orderHeader);
    void UpdateStatus(int id, string orderStatus, string? paymentStatus = null);
    void UpdateStripePaymentID(int id, string sessionId, string? paymentIntendId);
}
```

IProductRepository.cs

```
using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface IProductRepository : IRepository<Product>
{
    void Update(Product product);
}
```

IRepository.cs

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

namespace CourseWorkWeb.Data;

public interface IRepository<T> where T : class
{
    IEnumerable<T> GetAll(Expression<Func<T, bool>>? filter = null, string? includeProperties = null);
    T Get(Expression<Func<T, bool>> filter, string? includeProperties = null, bool tracked = false);
    void Add(T entity);
    void Remove(T entity);
    void RemoveRange(IEnumerable<T> entity);
}
```

IShoppingCartRepository.cs

```
using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public interface IShoppingCartRepository : IRepository<ShoppingCart>
{
    void Update(ShoppingCart shoppingCart);
}
```

IUnitOfWork.cs

```
namespace CourseWorkWeb.Data;

public interface IUnitOfWork
{
    ICategoryRepository Category { get; }
    IProductRepository Product { get; }
    ICompanyRepository Company { get; }
    IShoppingCartRepository ShoppingCart { get; }
    IApplicationUserRepository ApplicationUser { get; }
    IOrderDetailRepository OrderDetail { get; }
    IOrderHeaderRepository OrderHeader { get; }

    void Save();
}
```

OrderDetailRepository.cs

```
using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class OrderDetailRepository : Repository<OrderDetail>, IOrderDetailRepository
{
    private ApplicationDbContext _context;

    public OrderDetailRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }
}
```

```

    public void Update(OrderDetail orderDetail)
    {
        _context.OrderDetails.Update(orderDetail);
    }
}

```

OrderHeaderRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class OrderHeaderRepository : Repository<OrderHeader>, IOrderHeaderRepository
{
    private ApplicationDbContext _context;

    public OrderHeaderRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }

    public void Update(OrderHeader orderHeader)
    {
        _context.OrderHeaders.Update(orderHeader);
    }

    public void UpdateStatus(int id, string orderStatus, string? paymentStatus = null)
    {
        var orderFromDb = _context.OrderHeaders.FirstOrDefault(u => u.Id == id);
        if (orderFromDb != null)
        {
            orderFromDb.OrderStatus = orderStatus;
            if (!string.IsNullOrEmpty(paymentStatus))
            {
                orderFromDb.PaymentStatus = paymentStatus;
            }
        }
    }

    public void UpdateStripePaymentID(int id, string sessionId, string? paymentIntendId)
    {
        var orderFromDb = _context.OrderHeaders.FirstOrDefault(u => u.Id == id);
        if (!string.IsNullOrEmpty(sessionId))
        {
            orderFromDb.SessionId = sessionId;
        }

        if (!string.IsNullOrEmpty(paymentIntendId))
        {
            orderFromDb.PaymentIntendId = paymentIntendId;
            orderFromDb.PaymentDate = DateTime.Now;
        }
    }
}

```

ProductRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class ProductRepository : Repository<Product>, IProductRepository
{
    private ApplicationDbContext _context;

    public ProductRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }

    public void Update(Product product)
    {
        _context.Products.Update(product);
    }
}

```

Repository.cs

```

using System.Linq.Expressions;
using CourseWorkWeb.Data;
using Microsoft.EntityFrameworkCore;

namespace CourseWorkWeb.Data;

public class Repository<T> : IRepository<T> where T : class
{
    private readonly ApplicationDbContext _context;
    internal DbSet<T> dbSet;

    public Repository(ApplicationDbContext context)
    {
        _context = context;
        this.dbSet = _context.Set<T>();
        _context.Products
            .Include(p => p.Category)
            .Include(p => p.CategoryId);
    }

    public void Add(T entity)
    {
        dbSet.Add(entity);
    }

    public void Remove(T entity)
    {
        dbSet.Remove(entity);
    }

    public void RemoveRange(IEnumerable<T> entity)
    {
        dbSet.RemoveRange(entity);
    }

    public T Get(Expression<Func<T, bool>> filter, string? includeProperties = null, bool tracked = false)
    {
        IQueryable<T> query;

        if (tracked)
        {
            query = dbSet;
        }
        else
        {
            query = dbSet.AsNoTracking();
        }

        query = query.Where(filter);
        if (!string.IsNullOrEmpty(includeProperties))
        {
            foreach (var includeProp in includeProperties
                .Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))
            {
                query = query.Include(includeProp);
            }
        }

        return query.FirstOrDefault();
    }

    public IEnumerable<T> GetAll(Expression<Func<T, bool>>? filter, string? includeProperties = null)
    {
        IQueryable<T> query = dbSet;
        if (filter != null)
        {
            query = query.Where(filter);
        }

        if (!string.IsNullOrEmpty(includeProperties))
        {
            foreach (var includeProp in includeProperties
                .Split(new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries))
            {
                query = query.Include(includeProp);
            }
        }

        return query.ToList();
    }
}

```

SD.cs

```

namespace CourseWorkWeb.Data
{
    public class SD
    {
        public const string Role_Customer = "Customer";
        public const string Role_Company = "Company";
        public const string Role_Admin = "Admin";
        public const string Role_Employee = "Employee";

        public const string StatusPending = "Pending";
        public const string StatusApproved = "Approved";
        public const string StatusInProgress = "Processing";
        public const string StatusShipped = "Shipped";
        public const string StatusCancelled = "Cancelled";
        public const string StatusRefunded = "Refunded";

        public const string PaymentStatusPending = "Pending";
        public const string PaymentStatusApproved = "Approved";
        public const string PaymentStatusDelayedPayment = "ApprovedForDelayedPayment";
    }
}

```

ShoppingCartRepository.cs

```

using CourseWorkWeb.Models;

namespace CourseWorkWeb.Data;

public class ShoppingCartRepository : Repository<ShoppingCart>, IShoppingCartRepository
{
    private ApplicationDbContext _context;

    public ShoppingCartRepository(ApplicationDbContext context) : base(context)
    {
        _context = context;
    }

    public void Update(ShoppingCart shoppingCart)
    {
        _context.ShoppingCarts.Update(shoppingCart);
    }
}

```

StripeSettings.cs

```

namespace CourseWorkWeb.Data;

public class StripeSettings
{
    public string PublicKey { get; set; }
    public string SecretKey { get; set; }
}

```

UnitOfWork.cs

```

namespace CourseWorkWeb.Data;

public class UnitOfWork : IUnitOfWork
{
    private ApplicationDbContext _context;
    public ICategoryRepository Category { get; private set; }
    public ICompanyRepository Company { get; private set; }
    public IProductRepository Product { get; private set; }
    public IShoppingCartRepository ShoppingCart { get; private set; }
    public IApplicationUserRepository ApplicationUser { get; private set; }

    public IOrderHeaderRepository OrderHeader { get; private set; }
    public IOrderDetailRepository OrderDetail { get; private set; }

    public UnitOfWork(ApplicationDbContext context)
    {
        _context = context;
        ApplicationUser = new ApplicationUserRepository(_context);
        ShoppingCart = new ShoppingCartRepository(_context);
        Category = new CategoryRepository(_context);
        Product = new ProductRepository(_context);
        Company = new CompanyRepository(_context);
        OrderHeader = new OrderHeaderRepository(_context);
        OrderDetail = new OrderDetailRepository(_context);
    }

    public void Save()
    {
        _context.SaveChanges();
    }
}

```

Додаток Д

Код усіх представлень

Category\Create.cshtml

```
@model Category

<div class="card shadow border-0 mt-4">
  <div class="card-header bg-secondary bg-gradient m1-0 py-3">
    <div class="row">
      <div class="col-12 text-center">
        <h2 class="text-white py-2">Create Category</h2>
      </div>
    </div>
  </div>
  <div class="card-body p-4">
    <form method="post" class="row">
      <div class="border p-3">
        <div class="form-floating py-2 col-12">
          <input asp-for="Name" class="form-control border-0 shadow" />
          <label asp-for="Name" class="ms-2"></label>
          <span asp-validation-for="Name" class="text-danger"></span>
        </div>
        <div class="form-floating py-2 col-12">
          <input asp-for="DisplayOrder" class="form-control border-0 shadow" />
          <label asp-for="DisplayOrder" class="ms-2"></label>
          <span asp-validation-for="DisplayOrder" class="text-danger"></span>
        </div>
        @* <div asp-validation-summary="All"></div> *@

        <div class="row">
          <div class="col-6 col-md-3">
            <button type="submit" class="btn btn-primary form-control">Create</button>
          </div>
          <div class="col-6 col-md-3">
            <a asp-controller="Category" asp-action="Index" class="btn btn-outline-primary border form-control">
              <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
            </a>
          </div>
        </div>
      </div>
    </form>
  </div>
</div>

@section Scripts
{
  @{
    <partial name="_ValidationScriptsPartial"/>
  }
}
```

Category\Delete.cshtml

```
@model Category

<form method="post">
  <input asp-for="Id" hidden />
  <div class="border p-3 mt-4">
    <div class="row pb-2">
      <h2 class="text-primary">Delete Category</h2>
    </div>
    <div class="mb-3 row p-1">
      <label asp-for="Name" class="p-0"></label>
      <input asp-for="Name" disabled class="form-control" />
    </div>
    <div class="mb-3 row p-1">
      <label asp-for="DisplayOrder" class="p-0"></label>
      <input asp-for="DisplayOrder" disabled class="form-control" />
    </div>
    @* <div asp-validation-summary="All"></div> *@

    <div class="row">
      <div class="col-6 col-md-3">
        <button type="submit" class="btn btn-danger form-control">Delete</button>
      </div>
    </div>
  </div>
</form>
```

```

        </div>
        <div class="col-6 col-md-3">
            <a asp-controller="Category" asp-action="Index" class="btn btn-outline-secondary form-control">
                <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
            </a>
        </div>
    </div>
</form>

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}

```

Category\Edit.cshtml

```

@model Category

<form method="post">
    <input asp-for="Id" hidden />
    <div class="border p-3 mt-4">
        <div class="row pb-2">
            <h2 class="text-primary">Edit Category</h2>
            <hr/>
        </div>
        <div class="mb-3 row p-1">
            <label asp-for="Name" class="p-0"></label>
            <input asp-for="Name" class="form-control"/>
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>
        <div class="mb-3 row p-1">
            <label asp-for="DisplayOrder" class="p-0"></label>
            <input asp-for="DisplayOrder" class="form-control"/>
            <span asp-validation-for="DisplayOrder" class="text-danger"></span>
        </div>
        @* <div asp-validation-summary="All"></div> *@

        <div class="row">
            <div class="col-6 col-md-3">
                <button type="submit" class="btn btn-primary form-control">Update</button>
            </div>
            <div class="col-6 col-md-3">
                <a asp-controller="Category" asp-action="Index" class="btn btn-outline-secondary form-control">
                    <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
                </a>
            </div>
        </div>
    </div>
</form>

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}

```

Category\Index.cshtml

```

@model List<Category>

<div class="container">
    <div class="row pt-4 pb-2">
        <div class="col-6">
            <h2 class="text-primary">
                Category List
            </h2>
        </div>
        <div class="col-6 text-end">
            <a asp-controller="Category" asp-action="Create" class="btn btn-primary">
                <i class="bi bi-plus-square-fill"></i> Create New Category
            </a>
        </div>
    </div>

    <table class="table table-bordered table-striped">
        <thead>
            <tr>

```

```

        <th>
            Category Name
        </th>
        <th>
            Display Order
        </th>

    </tr>
</thead>
<tbody>
@foreach (var category in Model.OrderBy(u => u.DisplayOrder))
{
    <tr>
        <td>@category.Name</td>
        <td>@category.DisplayOrder</td>
        <td>
            <div class="w-75 btn-group" role="group">
                <a asp-controller="Category" asp-action="Edit" asp-route-id="@category.Id" class="btn btn-primary mx-2">
                    <i class="bi bi-tools"></i> Edit
                </a>
                <a asp-controller="Category" asp-action="Delete" asp-route-id="@category.Id" class="btn btn-danger mx-2">
                    <i class="bi bi-trash"></i> Delete
                </a>
            </div>
        </td>
    </tr>
}
</tbody>
</table>
</div>

```

Company\Delete.cshtml

```

@model Company

<form method="post">
    <input asp-for="Id" hidden />
    <div class="border p-3 my-4">
        <div class="row pb-2">
            <h2 class="text-primary">Delete Company</h2>
            <hr />
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="Name" disabled class="form-control border-0 shadow" />
            <label asp-for="Name" class="ms-2"></label>
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="PhoneNumber" disabled class="form-control border-0 shadow" />
            <label asp-for="PhoneNumber" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="StreetAddress" disabled class="form-control border-0 shadow" />
            <label asp-for="StreetAddress" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="City" disabled class="form-control border-0 shadow" />
            <label asp-for="City" class="ms-2"></label>
            <span asp-validation-for="City" class="text-danger"></span>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="State" disabled class="form-control border-0 shadow" />
            <label asp-for="State" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="PostalCode" disabled class="form-control border-0 shadow" />
            <label asp-for="PostalCode" class="ms-2"></label>
        </div>
        <div class="row">
            <div class="col-6 col-md-3">
                <button type="submit" class="btn btn-danger form-control">Delete</button>
            </div>
            <div class="col-6 col-md-3">
                <a asp-controller="Company" asp-action="Index" class="btn btn-outline-secondary form-control">
                    <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
                </a>
            </div>
        </div>
    </div>
</form>

```

```

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}

```

Company\Index.cshtml

```

@model List<Company>

<div class="container">
    <div class="row pt-4 pb-2">
        <div class="col-6">
            <h2 class="text-primary">
                Company List
            </h2>
        </div>
        <div class="col-6 text-end">
            <a asp-controller="Company" asp-action="Upsert" class="btn btn-primary">
                <i class="bi bi-plus-square-fill"></i> Create New Company
            </a>
        </div>
    </div>

    <table class="table table-bordered table-striped">
        <thead>
            <tr>
                <th>Name</th>
                <th>Address</th>
                <th>City</th>
                <th>State</th>
                <th>Phone Number</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var company in Model)
            {
                <tr>
                    <td>@company.Name</td>
                    <td>@company.StreetAddress</td>
                    <td>@company.City</td>
                    <td>@company.State</td>
                    <td>@company.PhoneNumber</td>
                    <td>
                        <div class="w-75 btn-group" role="group">
                            <a asp-controller="Company" asp-action="Upsert" asp-route-id="@company.Id" class="btn btn-primary mx-2">
                                <i class="bi bi-tools"></i> Edit
                            </a>
                            <a asp-controller="Company" asp-action="Delete" asp-route-id="@company.Id" class="btn btn-danger mx-2">
                                <i class="bi bi-trash"></i> Delete
                            </a>
                        </div>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>

```

Company\Upsert.cshtml

```

@model Company

<div class="card shadow border-0 my-4">
    <div class="card-header bg-secondary bg-gradient m1-0 py-3">
        <div class="row">
            <div class="col-12 text-center">
                <h2 class="text-white py-2">@(Model.Id!=0?"Update":"Create") Company</h2>
            </div>
        </div>
    </div>
    <div class="card-body p-4">
        <form method="post" class="row" enctype="multipart/form-data">
            <input asp-for="Id" hidden />
            <div class="row">
                <div class="col-12">
                    <div class="border p-3">
                        <div class="form-floating py-2 col-12">
                            <input asp-for="Name" class="form-control border-0 shadow" />
                            <label asp-for="Name" class="ms-2"></label>

```



```

        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="PhoneNumber" class="form-control border-0 shadow" />
        <label asp-for="PhoneNumber" class="ms-2"></label>
        <span asp-validation-for="PhoneNumber" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="StreetAddress" class="form-control border-0 shadow" />
        <label asp-for="StreetAddress" class="ms-2"></label>
        <span asp-validation-for="StreetAddress" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="City" class="form-control border-0 shadow" />
        <label asp-for="City" class="ms-2"></label>
        <span asp-validation-for="City" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="State" class="form-control border-0 shadow" />
        <label asp-for="State" class="ms-2"></label>
        <span asp-validation-for="State" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="PostalCode" class="form-control border-0 shadow" />
        <label asp-for="PostalCode" class="ms-2"></label>
        <span asp-validation-for="PostalCode" class="text-danger"></span>
    </div>
    <div asp-validation-summary="All"></div>

    <div class="row">
        <div class="col-6 col-md-3">
            @if (Model.Id != 0)
            {
                <button type="submit" class="btn btn-primary form-control">Update</button>
            }
            else
            {
                <button type="submit" class="btn btn-primary form-control">Create</button>
            }
        </div>
        <div class="col-6 col-md-3">
            <a asp-controller="Company" asp-action="Index" class="btn btn-outline-primary border form-
control">
                <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
            </a>
        </div>
    </div>

</div>
</div>
</div>

</form>
</div>
</div>

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial"/>
    }
}

```

Order\Details.cshtml

```

@using CourseWorkWeb.Data
@model OrderVM
<form method="post">
    <input asp-for="OrderHeader.Id" hidden/>
    <br/>
    <div class="container">
        <div class="card">
            <div class="card-header bg-dark text-light ml-0">
                <div class="container row">
                    <div class="col-12 d-none d-md-block col-md-6 pb-1">
                        <i class="fas fa-shopping-cart"></i> &nbsp; Order Summary
                    </div>
                    <div class="col-12 col-md-4 offset-md-2 text-right">
                        <a asp-action="Index" class="btn btn-outline-info form-control btn-sm">Back to Orders</a>
                    </div>
                </div>
            </div>
        </div>
        <div class="card-body">

```

```

<div class="container rounded p-2">
  <div class="row">
    <div class="col-12 col-lg-6 pb-4">
      <div class="row">
        <h4 class="d-flex justify-content-between align-items-center mb-3">
          <span class="text-primary">PickUp Details:</span>
        </h4>
      </div>
      <div class="row my-1">
        <div class="col-3">Name</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.Name" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.Name"></span>
          }
          else
          {
            <input asp-for="OrderHeader.Name" readonly type="text" class="form-control"/>
          }
        </div>
      </div>
      <div class="row my-1">
        <div class="col-3">Phone</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.PhoneNumber" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.PhoneNumber"></span>
          }
          else
          {
            <input asp-for="OrderHeader.PhoneNumber" readonly type="text" class="form-control"/>
          }
        </div>
      </div>
      <div class="row my-1">
        <div class="col-3">Address</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.StreetAddress" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.StreetAddress"></span>
          }
          else
          {
            <input asp-for="OrderHeader.StreetAddress" readonly type="text" class="form-control"/>
          }
        </div>
      </div>
      <div class="row my-1">
        <div class="col-3">City</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.City" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.City"></span>
          }
          else
          {
            <input asp-for="OrderHeader.City" readonly type="text" class="form-control"/>
          }
        </div>
      </div>
      <div class="row my-1">
        <div class="col-3">State</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.State" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.State"></span>
          }
          else
          {
            <input asp-for="OrderHeader.State" readonly type="text" class="form-control"/>
          }
        </div>
      </div>
      <div class="row my-1">
        <div class="col-3">Zip Code</div>
        <div class="col-9">
          @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
          {
            <input asp-for="OrderHeader.PostalCode" type="text" class="form-control"/>
            <span asp-validation-for="OrderHeader.PostalCode"></span>
          }
          else
          {
            <input asp-for="OrderHeader.PostalCode" type="text" class="form-control"/>
          }
        </div>
      </div>
    </div>
  </div>

```

```

        <input asp-for="OrderHeader.PostalCode" readonly type="text" class="form-control"/>
    }
</div>
</div>
<div class="row my-1">
    <div class="col-3">Email</div>
    <div class="col-9">
        <input asp-for="OrderHeader.ApplicationUser.Email" readonly type="text" class="form-
control"/>
    </div>
</div>
<div class="row my-1">
    <div class="col-3">Order Date</div>
    <div class="col-9">
        <input asp-for="@Model.OrderHeader.OrderTime.Date" readonly type="text" class="form-
control"/>
    </div>
</div>
<div class="row my-1">
    <div class="col-3">Carrier</div>
    <div class="col-9">
        @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
        {
            <input asp-for="OrderHeader.Carrier" id="carrier" type="text" class="form-control" />
        }
        else
        {
            <input asp-for="OrderHeader.Carrier" readonly type="text" class="form-control"/>
        }
    </div>
</div>
<div class="row my-1">
    <div class="col-3">Tracking Number</div>
    <div class="col-9">
        @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
        {
            <input asp-for="OrderHeader.Trackingnumber" id="trackingNumber" type="text"
class="form-control" />
        }
        else
        {
            <input asp-for="OrderHeader.Trackingnumber" readonly type="text" class="form-
control"/>
        }
    </div>
</div>
<div class="row my-1">
    <div class="col-3">Shipping Date</div>
    <div class="col-9">
        <input value="@Model.OrderHeader.ShippingDate.ToShortDateString()" readonly type="text"
class="form-control"/>
    </div>
</div>
@if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
{
    <div class="row my-1">
        <div class="col-3">Session ID</div>
        <div class="col-9">
            <input asp-for="OrderHeader.SessionId" readonly type="text" class="form-control"/>
        </div>
    </div>
    <div class="row my-1">
        <div class="col-3">Payment Intent ID</div>
        <div class="col-9">
            <input asp-for="OrderHeader.PaymentIntendId" readonly type="text" class="form-
control"/>
        </div>
    </div>
}

<div class="row my-1">
    @if (Model.OrderHeader.SessionId == null)
    {
        <div class="col-3">Payment Due Date</div>
        <div class="col-9">
            <input value="@Model.OrderHeader.PaymentDueDate.ToShortDateString()" readonly
type="text" class="form-control"/>
        </div>
    }
    else
    {
        <div class="col-3">Payment Date</div>
        <div class="col-9">
            <input value="@Model.OrderHeader.PaymentDate.ToShortDateString()" readonly
type="text" class="form-control"/>
        </div>
    }
}
</div>

```

```

        <div class="row my-1">
            <div class="col-3">Payment Status</div>
            <div class="col-9">
                <input asp-for="OrderHeader.PaymentStatus" readonly type="text" class="form-control"/>
            </div>
        </div>
        @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
        {
            <button type="submit" asp-action="UpdateOrderDetail" class="btn btn-warning form-control my-1">Update Order Details</button>
        }
    </div>
    <div class="col-12 col-lg-5 offset-lg-1">
        <h4 class="d-flex justify-content-between align-items-center mb-3">
            <span class="text-primary">Order Summary</span>
        </h4>
        <label class="btn btn-outline-primary form-control my-2">Order Status - @Model.OrderHeader.OrderStatus</label>

        <ul class="list-group mb-3">
            @foreach (var detail in Model.OrderDetail)
            {
                <li class="list-group-item d-flex justify-content-between p-2">
                    <div class="row container">
                        <div class="col-8">
                            <h6 class="my-0 text-primary">@detail.Product.Title</h6>
                            <small class="text-muted">Price : @detail.Price.ToString("c")</small><br/>
                            <small class="text-muted">Quantity : @detail.Count</small>
                        </div>
                        <div class="col-4 text-end">
                            <p class="text-success">@((detail.Count * detail.Price).ToString())</p>
                        </div>
                    </div>
                </li>
            }
            <li class="list-group-item bg-primary">
                <div class="row container">
                    <div class="col-6">
                        <h5 class="text-white">TOTAL </h5>
                    </div>
                    <div class="col-6 text-end">
                        <h5 class="text-white">@Model.OrderHeader.OrderTotal.ToString("c")</h5>
                    </div>
                </div>
            </li>
        </ul>
        @if (Model.OrderHeader.PaymentStatus == SD.PaymentStatusDelayedPayment
            && Model.OrderHeader.OrderStatus == SD.StatusShipped)
        {
            <button type="submit" class="btn btn-success form-control my-1">Pay Now</button>
        }
        @if (User.IsInRole(SD.Role_Admin) || User.IsInRole(SD.Role_Employee))
        {
            @if (Model.OrderHeader.OrderStatus == SD.StatusApproved)
            {
                <button type="submit" asp-action="StartProcessing" class="btn btn-primary form-control my-1">Start Processing</button>
            }
            @if (Model.OrderHeader.OrderStatus == SD.StatusInProgress)
            {
                <button type="submit" asp-action="ShipOrder" onclick="return validateInput()" class="btn btn-primary form-control my-1">Ship Order</button>
            }
            @if (Model.OrderHeader.OrderStatus != SD.StatusRefunded &&
                Model.OrderHeader.OrderStatus != SD.StatusCancelled &&
                Model.OrderHeader.OrderStatus != SD.StatusShipped)
            {
                <button asp-action="CancelOrder" type="submit" class="btn btn-danger form-control my-1">Cancel Order</button>
            }
        }
    </div>
</div>
</div>
</div>
</div>
</div>
</form>

@section Scripts
{
    <partial name="_ValidationScriptsPartial"/>
    <script>
        function validateInput(){
            if (document.getElementById("trackingNumber").value=="") {
                alert("Please enter tracking number!");
                return false;
            }
        }
    </script>
}

```

```

    }
    if (document.getElementById("carrier").value=="") {
        alert("Please enter carrier!");
        return false;
    }
    return true;
}
</script>
}

```

Order\Index.cshtml

```

@model List<OrderVM>

<div class="container">
    <div class="row pt-4 pb-2">
        <div class="col-6">
            <h2 class="text-primary">
                Order List
            </h2>
        </div>
    </div>

    <div>

        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Phone Number</th>
                    <th>Email</th>
                    <th>Status</th>
                    <th>Total</th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                @foreach (var order in Model)
                {
                    <tr>
                        <td>@order.OrderHeader.Id</td>
                        <td>@order.OrderHeader.Name</td>
                        <td>@order.OrderHeader.PhoneNumber</td>
                        <td>@order.OrderHeader.ApplicationUser.Email</td>
                        <td>@order.OrderHeader.OrderStatus</td>
                        <td>@order.OrderHeader.OrderTotal</td>
                        <td>
                            <div class="w-75 btn-group" role="group">
                                <a asp-area="Admin" asp-controller="Order" asp-action="Details" asp-route-
orderId="@order.OrderHeader.Id" class="btn btn-primary mx-2">
                                    <i class="bi bi-tools"></i>Details
                                </a>
                            </div>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>

```

Order\PaymentConfirmation.cshtml

```

@model int

<div class="container row pt-4">
    <div class="col-12 text-center">
        <h1 class="text-primary text-center">Payment Successful!</h1>
        Order number : @Model <br /><br />
        
    </div>
    <div class="col-12 text-center" style="color: maroon">
        <br />
        Payment has been applied succesfully!
        <a asp-area="Admin"
            asp-controller="Order"
            asp-action="Details"
            asp-route-orderId="@Model"
            class="btn btn-primary">Back to Order Details</a>
    </div>
</div>

```

Product\Delete.cshtml

```

@model Product

<form method="post">
    <input asp-for="Id" hidden />
    <div class="border p-3 my-4">
        <div class="row pb-2">
            <h2 class="text-primary">Delete Product</h2>
            <hr />
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="Title" disabled class="form-control border-0 shadow" />
            <label asp-for="Title" class="ms-2"></label>
            <span asp-validation-for="Title" class="text-danger"></span>
        </div>

        <div class="form-floating py-2 col-12">
            <input asp-for="Description" disabled class="form-control border-0 shadow" />
            <label asp-for="Description" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="ISBN" disabled class="form-control border-0 shadow" />
            <label asp-for="ISBN" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="Author" disabled class="form-control border-0 shadow" />
            <label asp-for="Author" class="ms-2"></label>
            <span asp-validation-for="Author" class="text-danger"></span>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="ListPrice" disabled class="form-control border-0 shadow" />
            <label asp-for="ListPrice" class="ms-2"></label>
        </div>

        <div class="form-floating py-2 col-12">
            <input asp-for="Price" disabled class="form-control border-0 shadow" />
            <label asp-for="Price" class="ms-2"></label>
        </div>

        <div class="form-floating py-2 col-12">
            <input asp-for="Price50" disabled class="form-control border-0 shadow" />
            <label asp-for="Price50" class="ms-2"></label>
        </div>
        <div class="form-floating py-2 col-12">
            <input asp-for="Price100" disabled class="form-control border-0 shadow" />
            <label asp-for="Price100" class="ms-2"></label>
        </div>
        @* <div asp-validation-summary="All"></div> *@

        <div class="row">
            <div class="col-6 col-md-3">
                <button type="submit" class="btn btn-danger form-control">Delete</button>
            </div>
            <div class="col-6 col-md-3">
                <a asp-controller="Product" asp-action="Index" class="btn btn-outline-secondary form-control">
                    <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
                </a>
            </div>
        </div>
    </div>
</form>

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}

```

Product\Index.cshtml

```

@model List<Product>

<div class="container">
    <div class="row pt-4 pb-2">
        <div class="col-6">
            <h2 class="text-primary">
                Product List
            </h2>
        </div>
        <div class="col-6 text-end">
            <a asp-controller="Product" asp-action="Upsert" class="btn btn-primary">
                <i class="bi bi-plus-square-fill"></i> Create New Product
            </a>
        </div>
    </div>
</div>

```

```

<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th>Title</th>
      <th>ISBN</th>
      <th>Price</th>
      <th>Author</th>
      <th>Category</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var category in Model)
    {
      <tr>
        <td>@category.Title</td>
        <td>@category.ISBN</td>
        <td>@category.ListPrice</td>
        <td>@category.Author</td>
        <td>@category.Category.Name</td>
        <td>
          <div class="w-75 btn-group" role="group">
            <a asp-controller="Product" asp-action="Upsert" asp-route-id="@category.Id" class="btn btn-primary
mx-2">
              <i class="bi bi-tools"></i> Edit
            </a>
            <a asp-controller="Product" asp-action="Delete" asp-route-id="@category.Id" class="btn btn-danger mx-
2">
              <i class="bi bi-trash"></i> Delete
            </a>
          </div>
        </td>
      </tr>
    }
  </tbody>
</table>
</div>

```

Product\Upsert.cshtml

```

@model ProductVM

<div class="card shadow border-0 my-4">
  <div class="card-header bg-secondary bg-gradient ml-0 py-3">
    <div class="row">
      <div class="col-12 text-center">
        <h2 class="text-white py-2">@(Model.Product.Id!=0?"Update":"Create") Product</h2>
      </div>
    </div>
  </div>
  <div class="card-body p-4">
    <form method="post" class="row" enctype="multipart/form-data">
      <input asp-for="Product.Id" hidden />
      <input asp-for="Product.ImageUrl" hidden />
      <div class="row">
        <div class="col-10">
          <div class="border p-3">
            <div class="form-floating py-2 col-12">
              <input asp-for="Product.Title" class="form-control border-0 shadow" />
              <label asp-for="Product.Title" class="ms-2"></label>
              <span asp-validation-for="Product.Title" class="text-danger"></span>
            </div>
            <div class="form-floating py-2 col-12">
              <textarea asp-for="Product.Description" class="form-control border-0 shadow"></textarea>
              <label asp-for="Product.Description" class="ms-2"></label>
              <span asp-validation-for="Product.Description" class="text-danger"></span>
            </div>
            <div class="form-floating py-2 col-12">
              <input asp-for="Product.ISBN" class="form-control border-0 shadow" />
              <label asp-for="Product.ISBN" class="ms-2"></label>
              <span asp-validation-for="Product.ISBN" class="text-danger"></span>
            </div>
            <div class="form-floating py-2 col-12">
              <input asp-for="Product.Author" class="form-control border-0 shadow" />
              <label asp-for="Product.Author" class="ms-2"></label>
              <span asp-validation-for="Product.Author" class="text-danger"></span>
            </div>
            <div class="form-floating py-2 col-12">
              <input asp-for="Product.ListPrice" class="form-control border-0 shadow" />
              <label asp-for="Product.ListPrice" class="ms-2"></label>
              <span asp-validation-for="Product.ListPrice" class="text-danger"></span>
            </div>
          </div>
        </div>
      </div>
    </form>
  </div>
</div>

```

```

        <input asp-for="Product.Price" class="form-control border-0 shadow" />
        <label asp-for="Product.Price" class="ms-2"></label>
        <span asp-validation-for="Product.Price" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="Product.Price50" class="form-control border-0 shadow" />
        <label asp-for="Product.Price50" class="ms-2"></label>
        <span asp-validation-for="Product.Price50" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input asp-for="Product.Price100" class="form-control border-0 shadow" />
        <label asp-for="Product.Price100" class="ms-2"></label>
        <span asp-validation-for="Product.Price100" class="text-danger"></span>
    </div>

    <div class="form-floating py-2 col-12">
        <select asp-for="@Model.Product.CategoryId" asp-items="@Model.CategoryList" class="form-select
border-0 shadow">
            <option disabled selected>--Select Category--</option>
        </select>
        <label asp-for="@Model.Product.CategoryId" class="ms-2"></label>
        <span asp-validation-for="@Model.Product.CategoryId" class="text-danger"></span>
    </div>
    <div class="form-floating py-2 col-12">
        <input type="file" name="file" class="form-control border-0 shadow" />
        <label asp-for="Product.ImageUrl" class="ms-2"></label>
    </div>
    <div asp-validation-summary="All"></div>

    <div class="row">
        <div class="col-6 col-md-3">
            @if (Model.Product.Id != 0)
            {
                <button type="submit" class="btn btn-primary form-control">Update</button>
            }
            else
            {
                <button type="submit" class="btn btn-primary form-control">Create</button>
            }
        </div>
        <div class="col-6 col-md-3">
            <a asp-controller="Product" asp-action="Index" class="btn btn-outline-primary border form-
control">
                <i class="bi bi-card-list"></i> Back to List <i class="bi bi-card-list"></i>
            </a>
        </div>
    </div>
    <div class="col-2">
        
    </div>
</div>
</form>
</div>
</div>

@section Scripts
{
    @{
        <partial name="_ValidationScriptsPartial"/>
    }
}

```

User\Index.cshtml

```
@model List<ApplicationUser>
```

```

<div class="container">
    <div class="row pt-4 pb-2">
        <div class="col-6">
            <h2 class="text-primary">
                User List
            </h2>
        </div>
    </div>

    <table class="table table-bordered table-striped">
        <thead>
            <tr>
                <th>Name</th>
            </tr>
        </thead>
    </table>

```



```

        <th>Email</th>
        <th>Phone</th>
        <th>Role</th>
        <th>Status</th>
    </tr>
</thead>
<tbody>
@foreach (var user in Model)
{
    <tr>
        <td>@user.Name</td>
        <td>@user.Email</td>
        <td>@user.PhoneNumber</td>
        <td>@user.Role</td>
        <td>
            <form method="post">
                @if (user.LockoutEnd > DateTime.Now)
                {
                    <div>
                        <button asp-action="LockUnlock" type="submit" asp-route-id="@user.Id" class="btn btn-
danger w-100 py-2">Locked</button>
                    </div>
                }
                else
                {
                    <div>
                        <button asp-action="LockUnlock" type="submit" asp-route-id="@user.Id" class="btn btn-
success border-0 bg-gradient w-100 py-2">Unlocked</button>
                    </div>
                }
            </form>
        </td>
    </tr>
}
</tbody>
</table>
</div>

```

Cart/Index.cshtml

```

@model ShoppingCartVM

<form method="post">
    <br />
    <div class="card shadow border-0">
        <div class="card-header bg-secondary bg-gradient text-light ml-0 py-4">
            <div class="row px-4">
                <div class="col-6">
                    <h5 class="pt-2 text-white">
                        Shopping Cart
                    </h5>
                </div>
            </div>
        </div>
    </div>
    <div class="card-body my-4">
        <div class="row">
            </div>
            <div class="row mb-3 pb-3">
                <div class="col-md-2 offset-md-1">
                    <a asp-area="Customer" asp-controller="Home" asp-action="Index" class="btn btn-outline-primary text-uppercase
mb-5 btn-sm"><small>Continue Shopping</small></a>
                </div>
                <div class="col-md-10 offset-md-1">

                    @foreach (var item in Model.ShoppingCartList)
                    {
                        <div class="row border-bottom pb-3">
                            <div class="d-none d-lg-block col-lg-1 text-center py-2">
                                
                            </div>
                            <div class="col-12 col-lg-6 pt-md-3">
                                <h5 class="text-uppercase text-secondary"><strong>@item.Product.Title</strong></h5>
                                <p><small>@item.Product.Description</small></p>
                            </div>
                            <div class="col-12 col-lg-5 text-center row">
                                <div class="col-3 text-md-right pt-2 pt-md-4">
                                    <h6 class="fw-semibold">
                                        @item.Price.ToString("c")
                                    <span class="text-muted">&nbsp;x&nbsp;&nbsp;x&nbsp;&nbsp;x</span>@item.Count
                                </h6>
                            </div>
                            <div class="col-6 col-sm-4 col-lg-6 pt-2">
                                <div class="w-75 btn-group" role="group">
                                    <a asp-action="plus" asp-route-cartId="@item.Id" class="btn btn-outline-primary bg-

```



```

        <span class="text-info">Shipping Details:</span>
      </h4>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>Name</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.Name" class="form-control" />
        <span asp-validation-for="OrderHeader.Name" class="text-dander"></span>
      </div>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>Phone</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.PhoneNumber" class="form-control" />
        <span asp-validation-for="OrderHeader.PhoneNumber" class="text-dander"></span>
      </div>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>Street Address</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.StreetAddress" class="form-control" />
        <span asp-validation-for="OrderHeader.StreetAddress" class="text-dander"></span>
      </div>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>City</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.City" class="form-control" />
        <span asp-validation-for="OrderHeader.City" class="text-dander"></span>
      </div>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>State</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.State" class="form-control" />
        <span asp-validation-for="OrderHeader.State" class="text-dander"></span>
      </div>
    </div>
    <div class="row my-1">
      <div class="col-3">
        <label>Postal Code</label>
      </div>
      <div class="col-9">
        <input asp-for="OrderHeader.PostalCode" class="form-control" />
        <span asp-validation-for="OrderHeader.PostalCode" class="text-dander"></span>
      </div>
    </div>
    <div>
      <div class="col-12 col-lg-5 offset-lg-1">
        <h4 class="d-flex justify-content-between align-items-center mb-3">
          <span class="text-info">Order Summary:</span>
        </h4>
        <ul class="list-group mb-3">
          @foreach (var details in Model.ShoppingCartList)
          {
            <li class="list-group-item d-flex justify-content-between">
              <div>
                <h6 class="my-0">@details.Product.Title</h6>
                <small class="text-muted">Quantity: @details.Count</small>
              </div>
              <span class="text-muted">@((details.Price*details.Count).ToString("c"))</span>
            </li>
          }
          <li class="list-group-item d-flex justify-content-between bg-light">
            <small class="text-info">Total (USD)</small>
            <strong class="text-info">@Model.OrderHeader.OrderTotal.ToString("c")</strong>
          </li>
        </ul>
      </div>
    </div>
  </div>
</div>
<div class="card-footer">
  <div class="row">
    <div class="col-12 col-md-8 pt-2">
      <p style="color:maroon; font-size:14px;">

```

```

        Estimate Arrival Date:
        @DateTime.Now.AddDays(7).ToShortDateString() - @DateTime.Now.AddDays(14).ToShortDateString()
    </p>
</div>
<div class="col-12 col-md-4">
    <button type="submit" value="Place Order" class="btn btn-primary form-control">Place Order</button>
</div>
</div>
</div>
</div>
</div>
</form>

```

Home\Details.cshtml

```

@model ShoppingCart

<form method="post">
    <input hidden asp-for="ProductId"/>
    <div class="card shadow border-0 mt-4 mb-4">
        <div class="card-header bg-secondary bg-gradient text-light py-4">
            <div class="row">
                <div class="col-12 text-center">
                    <h3 class="text-white text-uppercase">@Model.Product.Title</h3>
                    <p class="text-white-50 fw-semibold mb-0">by @Model.Product.Author</p>
                </div>
            </div>
        </div>
        <div class="card-body">
            <div class="py-3">
                <div class="row">
                    <div class="col-6 col-md-2 offset-lg-1 pb-1">
                        <a asp-action="Index" class="btn btn-outline-primary bg-gradient mb-5 fw-semibold btn-sm text-
uppercase">
                            <small>Back to home</small>
                        </a>
                    </div>
                </div>
                <div class="row">
                    <div class="col-12 col-lg-3 offset-lg-1 text-center mb-3">
                        
                    </div>
                    <div class="col-12 col-lg-6 offset-lg-1">
                        <div class="col-12 col-md-6 pb-4">
                            <span class="badge text-dark">@Model.Product.Category.Name</span>
                        </div>
                        <div class="row ps-2">
                            <h6 class="text-dark text-opacity-50 ">ISBN : @Model.Product.ISBN</h6>
                        </div>
                        <div class="row ps-2">
                            <h6 class="text-dark text-opacity-50 pb-2">
                                List Price:
                                <span class="text-decoration-line-through">
                                    @Model.Product.ListPrice.ToString("c")
                                </span>
                            </h6>
                        </div>
                        <div class="row text-center ps-2">
                            <div class="p-1 col-3 col-lg-2 bg-white border-bottom">
                                <div class="text-dark text-opacity-50 fw-semibold">Quantity</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white border-bottom">
                                <div class="text-dark text-opacity-50 fw-semibold">1-50</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white border-bottom">
                                <div class="text-dark text-opacity-50 fw-semibold">51-100</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white border-bottom">
                                <div class="text-dark text-opacity-50 fw-semibold">100+</div>
                            </div>
                        </div>
                        <div class="row text-center ps-2">
                            <div class="p-1 col-3 col-lg-2 bg-white text-warning fw-bold">
                                <div>Price</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white text-warning fw-bold">
                                <div>@Model.Product.Price.ToString("c")</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white text-warning fw-bold">
                                <div>@Model.Product.Price50.ToString("c")</div>
                            </div>
                            <div class="p-1 col-3 col-lg-2 bg-white text-warning fw-bold">
                                <div>@Model.Product.Price100.ToString("c")</div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</form>

```

```

        <p class="text-secondary lh-sm">@Model.Product.Description</p>
    </div>
    <div class="row pl-2 mb-3">
        <div class="col-md-4">
            <div class="input-group mb-3">
                <span class="input-group-text bg-primary text-white border-0 fw-semibold"
                    id="inputGroup-sizing-default">
                    Count
                </span>
                <input asp-for="Count" type="number" value="1" class="form-control text-end"
                    aria-label="Sizing example input" aria-describedby="inputGroup-sizing-default"/>
            </div>
        </div>
    </div>
    <div class="row">
        <div class="col-12 col-md-6 pb-1">
            <button type="submit"
                class="btn btn-primary bg-gradient w-100 py-2 text-uppercase fw-semibold">
                Add to Cart
            </button>
        </div>
    </div>
</div>
</div>
</div>
</div>
</form>

```

Home\Index.cshtml

```

@model IEnumerable<Product>

<div class="row pb-3">
    @foreach (var product in Model)
    {
        <div class="col-lg-3 col-sm-6">
            <div class="row p-2">
                <div class="col-12 p-1">
                    <div class="card border-0 p-3 shadow border-top border-5 rounded">
                        
                        <div class="card-body pb-0">
                            <div class="pl-1">
                                <p class="card-title h5 text-dark opacity-75 text-uppercase text-center">@product.Title</p>
                                <p class="card-title text-warning text-center">by <b>@product.Author</b></p>
                            </div>
                            <div class="pl-1">
                                <p class="text-dark opacity-75 text-center mb-0">
                                    List Price:
                                    <span class="text-decoration-line-through">
                                        @product.ListPrice.ToString("c")
                                    </span>
                                </p>
                            </div>
                            <div class="pl-1">
                                <p class="text-dark opacity-75 text-center">
                                    As low as:
                                    <span>
                                        @product.Price100.ToString("c")
                                    </span>
                                </p>
                            </div>
                        </div>
                    </div>
                    <div>
                        <a asp-action="Details"
                            asp-route-productId="@product.Id"
                            class="btn btn-primary bg-gradient border-0 form-control">
                            Details
                        </a>
                    </div>
                </div>
            </div>
        </div>
    }
</div>
<div>
    <h2>Advertisement</h2>
    <iframe style="border-radius:12px" src="https://open.spotify.com/embed/track/56cIg3GhiWW50Lp87pQ8MA?utm_source=generator"
        width="100%" height="352" frameborder="0" allowfullscreen=""
        allow="autoplay; clipboard-write; encrypted-media; fullscreen; picture-in-picture" loading="lazy">
    </iframe>
</div>

```

Shared_Layout.cshtml

```

@using CourseWorkWeb.Data
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"]Vlad's Bookstore</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="~/CourseWorkWeb.styles.css" asp-append-version="true" />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/css/toastr.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-primary border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Vlad's Bookstore</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav me-auto">
                        <li class="nav-item">
                            <a class="nav-link" " asp-area="Customer" asp-controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" " asp-area="Admin" asp-controller="Order" asp-action="Index">Manage Order</a>
                        </li>
                        @if (User.IsInRole(SD.Role_Admin))
                        {
                            <li class="nav-item dropdown">
                                <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
                                    Content Management
                                </a>
                                <ul class="dropdown-menu">
                                    <li class="nav-item">
                                        <a class="dropdown-item" " asp-area="Admin" asp-controller="Category" asp-
action="Index">Category</a>
                                    </li>
                                    <li><hr class="dropdown-divider"></li>
                                    <li class="nav-item">
                                        <a class="dropdown-item" " asp-area="Admin" asp-controller="Product" asp-
action="Index">Product</a>
                                    </li>
                                    <li class="nav-item">
                                        <a class="dropdown-item" " asp-area="Admin" asp-controller="Company" asp-
action="Index">Company</a>
                                    </li>
                                    <li class="nav-item">
                                        <a class="dropdown-item" " asp-area="Identity" asp-page="/Account/Register">Create
User</a>
                                    </li>
                                    <li class="nav-item">
                                        <a class="dropdown-item" " asp-area="Admin" asp-controller="User" asp-
action="Index">Manage User</a>
                                    </li>
                                    <li><hr class="dropdown-divider"></li>
                                </ul>
                            </li>
                        }
                        <li class="nav-item">
                            <a class="nav-link" " asp-area="Customer" asp-controller="Cart" asp-action="Index">
                                <i class="bi bi-cart"></i>
                            </a>
                        </li>
                    </ul>
                    <partial name="_LoginPartial"/>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            <partial name="_Notification" />
            @RenderBody()
        </main>
    </div>
    <footer class="border-top footer bg-primary text-muted">

```

```
<div class="text-center">
  Copyright 2023 Vladyslav Keidaliuk (В рамках курсового проєкту з дисципліни "Веб-технології та веб-дизайн") <i
class="bi bi-wifi"></i>
  @* <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a> *@
</div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Додаток Е

Код усіх файлів з проекту з тестами

AreaAndAccessTests.cs

```

using CourseWorkWeb.Tests.Pages;
using CourseWorkWeb.Tests.UtilityLibrary;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace CourseWorkWeb.Tests.CourseWorkWebTests
{
    public class AreaAndAccessTests
    {
        private IWebDriver driver;
        private MainPage mainPage;
        private LoginPage loginPage;
        private UserListPage userListPage;

        private static IEnumerable<object[]> ValidCredentialsTestData()
        {
            yield return new object[] { PersonalData.AdminUsername, PersonalData.AdminPassword };
            yield return new object[] { PersonalData.CompanyUsername, PersonalData.CompanyPassword };
            yield return new object[] { PersonalData.CustomerUsername, PersonalData.CustomerPassword };
            yield return new object[] { PersonalData.EmployeeUsername, PersonalData.EmployeePassword };
        }

        private static IEnumerable<object[]> InvalidCredentialsTestData()
        {
            yield return new object[] { PersonalData.AdminUsername, "testpass1" };
            yield return new object[] { "Userandom@gmail.com", PersonalData.CompanyPassword };
            yield return new object[] { "wronguser", PersonalData.CustomerPassword };
        }

        [SetUp]
        public void Setup()
        {
            driver = new ChromeDriver();
            driver.Manage().Window.Maximize();
            mainPage = new MainPage(driver);
            loginPage = new LoginPage(driver);
            userListPage = new UserListPage(driver);
        }

        [Test]
        [TestCaseSource(nameof(ValidCredentialsTestData))]
        public void LogInWithValidDataReturnTrue(string email, string password)
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool status = loginPage.Login(email, password);
            mainPage.LogOutButtonClick();
            Assert.That(status, Is.EqualTo(true));
        }

        [Test]
        [TestCaseSource(nameof(InvalidCredentialsTestData))]
        public void LogInWithInvalidDataReturnFalse(string email, string password)
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool status = loginPage.Login(email, password);
            Assert.That(status, Is.EqualTo(false));
        }

        [Test]
        public void AccessDeniedForCustomerToAdminAreaReturnFalse()
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool status = loginPage.Login(PersonalData.CustomerUsername, PersonalData.CustomerPassword);
            bool access = true;
            if (status)
            {
                driver.Navigate().GoToUrl("https://localhost:7217/Admin/User");
                if (driver.FindElement(By.CssSelector("body > div > main > header > h1")).Displayed)
                {
                    access = false;
                }
            }

            Assert.That(access, Is.EqualTo(false));
        }
    }
}

```



```

[Test]
public void AccessAllowedForAdminToAdminAreaReturnTrue()
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool status = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool access = false;
    if (status)
    {
        driver.Navigate().GoToUrl("https://localhost:7217/Admin/User");
        try
        {
            driver.FindElement(By.CssSelector("body > div > main > header > h1"));
        }
        catch (Exception e)
        {
            access = true;
        }
    }

    Assert.That(access, Is.EqualTo(true));
}

[Test, Order(1)]
[TestCase("employee@gmail.com")]
public void LockCustomerByAdminReturnFalse(string email)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool statusAfterBlock = true;
    bool status = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool access = false;
    if (status)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementManageUserButtonClick();
        userListPage.LockUnlockUserWithEmail(email);
        mainPage.LogOutButtonClick();
        mainPage.LoginButtonClick();
        statusAfterBlock = loginPage.Login(PersonalData.EmployeeUsername, PersonalData.EmployeePassword);
    }
    Assert.That(statusAfterBlock, Is.EqualTo(false));
}

[Test, Order(2)]
[TestCase("employee@gmail.com")]
public void UnlockCustomerByAdminReturnTrue(string email)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool statusAfterBlock = false;
    bool status = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool access = false;
    if (status)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementManageUserButtonClick();
        userListPage.LockUnlockUserWithEmail(email);
        mainPage.LogOutButtonClick();
        mainPage.LoginButtonClick();
        statusAfterBlock = loginPage.Login(PersonalData.EmployeeUsername, PersonalData.EmployeePassword);
    }
    Assert.That(statusAfterBlock, Is.EqualTo(true));
}

[TearDown]
public void TearDown()
{
    driver.Close();
    driver.Quit();
}
}
}

```

CartTests.cs

```

using CourseWorkWeb.Tests.Pages;
using CourseWorkWeb.Tests.UtilityLibrary;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace CourseWorkWeb.Tests.CourseWorkWebTests
{
    public class CartTests
    {
        private IWebDriver driver;
        private MainPage mainPage;
        private LoginPage loginPage;

        [SetUp]
        public void Setup()
        {
            driver = new ChromeDriver();
            driver.Manage().Window.Maximize();
            mainPage = new MainPage(driver);
            loginPage = new LoginPage(driver);
        }

        [Test, Order(1)]
        public void Add5BookToCart()
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool status = loginPage.Login(PersonalData.CustomerUsername, PersonalData.CustomerPassword);
            if (status)
            {
                for (int i = 1; i < 6; i++)
                {
                    mainPage.ScrollDown(500);
                    mainPage.DetailsButtonNClick(i);
                    mainPage.AddToCartButtonClick();
                }

                status = true;
            }

            mainPage.LogOutButtonClick();
            Assert.That(status, Is.EqualTo(true));
        }

        [Test, Order(2)]
        [TestCase("1984")]
        [TestCase("ЛЮДИНА В ПОШУКАХ СПРАВЖНЬОГО СЕНСУ. ПСИХОЛОГ У КОНЦТАБОРІ")]
        [TestCase("ENGLISH GRAMMAR IN USE 5TH EDITION WITH ANSWERS")]
        [TestCase("КАФЕ НА КРАЮ СВІТУ - СТРЕЛЕКИ ДЖ. П.")]
        [TestCase("ОДНА З ДІВЧАТ")]
        public void CheckThat5BookToCartAddedReturnTrue(string title)
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool bookExist = false;
            bool status = loginPage.Login(PersonalData.CustomerUsername, PersonalData.CustomerPassword);
            if (status)
            {
                mainPage.CartButtonClick();
                bookExist = mainPage.CheckThatBookExistInsideCart(title);
            }

            mainPage.LogOutButtonClick();
            Assert.That(bookExist, Is.EqualTo(true));
        }

        [TearDown]
        public void TearDown()
        {
            driver.Close();
            driver.Quit();
        }
    }
}

```

CRUDWithObjectsTests.cs

```

using CourseWorkWeb.Tests.Pages;
using CourseWorkWeb.Tests.UtilityLibrary;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace CourseWorkWeb.Tests.CourseWorkWebTests
{
    public class CRUDWithObjectsTests
    {
        private IWebDriver driver;
        private MainPage mainPage;
        private LoginPage loginPage;
        private CategoryPage categoryPage;
        private ProductAdminPage productAdminPage;
        private CompanyPage companyPage;
        private CreateUserPage createUserPage;

        [SetUp]
        public void Setup()
        {
            driver = new ChromeDriver();
            driver.Manage().Window.Maximize();
            mainPage = new MainPage(driver);
            loginPage = new LoginPage(driver);
            categoryPage = new CategoryPage(driver);
            productAdminPage = new ProductAdminPage(driver);
            companyPage = new CompanyPage(driver);
            createUserPage = new CreateUserPage(driver);
        }

        [Test, Order(1)]
        [TestCase("My Category", "8")]
        [TestCase("My Category1", "9")]
        [TestCase("My Category2", "10")]
        public void CreateNewCategoryTestReturnTrue(string name, string displayOrder)
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
            bool CreateStatus = false;
            if (LoginStatus)
            {
                mainPage.ContentManagementDropdownClick();
                mainPage.ContentManagementCategoryButtonClick();
                categoryPage.NewCategoryButtonClick();
                CreateStatus = categoryPage.NewCategoryCreate(name, displayOrder);
            }

            Assert.That(CreateStatus, Is.EqualTo(true));
        }

        [Test, Order(2)]
        [TestCase("My Category", "Polytech", "25")]
        [TestCase("My Category1", "IIBRT", "50")]
        [TestCase("My Category2", "ICS", "27")]
        public void EditCategoriesTestReturnTrue(string name, string newName, string displayOrder)
        {
            driver.Url = "https://localhost:7217/";
            mainPage.LoginButtonClick();
            bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
            bool EditStatus = false;
            if (LoginStatus)
            {
                mainPage.ContentManagementDropdownClick();
                mainPage.ContentManagementCategoryButtonClick();
                EditStatus = categoryPage.EditCategory(name, newName, displayOrder);
            }

            Assert.That(EditStatus, Is.EqualTo(true));
        }

        [Test, Order(3)]
        [TestCase(
            "Метро 2033",
            "Метро 2033 – культовий роман-антиутопія",
            "978-966-10-6112-4",
            "Глуховський Д.",
            "500",
            "450",
            "430",
            "400"
        )]
        public void CreateProductTestReturnTrue(
            string title,

```

```

        string description,
        string ISBN,
        string author,
        string listPrice,
        string price,
        string price50,
        string price100
    )
    {
        driver.Url = "https://localhost:7217/";
        mainPage.LoginButtonClick();
        bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
        bool CreateStatus = false;
        if (LoginStatus)
        {
            mainPage.ContentManagementDropdownClick();
            mainPage.ContentManagementProductButtonClick();
            productAdminPage.NewProductButtonClick();
            CreateStatus = productAdminPage
                .NewProductCreate(title, description, ISBN, author, listPrice, price, price50, price100);
        }

        Assert.That(CreateStatus, Is.EqualTo(true));
    }

[Test, Order(4)]
[TestCase("Merpo 2033", "Vladyslav Book")]
public void EditProductTitleTestReturnTrue(string name, string newName)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool EditStatus = false;
    if (LoginStatus)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementProductButtonClick();
        EditStatus = productAdminPage.EditTitleInProduct(name, newName);
    }

    Assert.That(EditStatus, Is.EqualTo(true));
}

[Test, Order(5)]
[TestCase("Polytech")]
[TestCase("IIBRT")]
[TestCase("ICS")]
public void DeleteCategoriesTestReturnTrue(string name)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool DeleteStatus = false;
    if (LoginStatus)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementCategoryButtonClick();
        DeleteStatus = categoryPage.DeleteCategory(name);
    }

    Assert.That(DeleteStatus, Is.EqualTo(true));
}

[Test, Order(6)]
[TestCase("Polytech",
    "0486953248",
    "Shevchenko 1",
    "Odesa",
    "Ukraine",
    "65000"
)]
public void CreateCompanyTestReturnTrue(
    string name,
    string phoneNumber,
    string streetAddress,
    string city,
    string state,
    string postalCode
)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool CreateStatus = false;
    if (LoginStatus)

```

```

        {
            mainPage.ContentManagementDropdownClick();
            mainPage.ContentManagementCompanyButtonClick();
            companyPage.NewCompanyButtonClick();
            CreateStatus = companyPage
                .NewCompanyCreate(name, phoneNumber, streetAddress, city, state, postalCode);
        }

        Assert.That(CreateStatus, Is.EqualTo(true));
    }

[Test, Order(7)]
[TestCase(
    "Polytech",
    "EPAM",
    "080096513",
    "Main Street 3",
    "Kyiv",
    "Ukraine",
    "63654"
)]
public void EditCompanyDataTestReturnTrue(
    string name,
    string newName,
    string phoneNumber,
    string streetAddress,
    string city,
    string state,
    string postalCode)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool EditStatus = false;
    if (LoginStatus)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementCompanyButtonClick();
        EditStatus =
            companyPage.EditCompanyData(name, newName, phoneNumber, streetAddress, city, state, postalCode);
    }

    Assert.That(EditStatus, Is.EqualTo(true));
}

[Test, Order(8)]
[TestCase("EPAM")]
public void DeleteCompanyTestReturnTrue(string name)
{
    driver.Url = "https://localhost:7217/";
    mainPage.LoginButtonClick();
    bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
    bool DeleteStatus = false;
    if (LoginStatus)
    {
        mainPage.ContentManagementDropdownClick();
        mainPage.ContentManagementCompanyButtonClick();
        DeleteStatus = companyPage.DeleteCompany(name);
    }

    Assert.That(DeleteStatus, Is.EqualTo(true));
}

[Test]
[TestCase(
    "NewUser",
    "0956543621",
    "Testpassword_123",
    "Testpassword_123",
    "Loopback Street 5",
    "Rivne",
    "Ukraine",
    "63465"
)]
public void RegisterNewUserCustomerFromAdminPanelTestReturnTrue(
    string name,
    string phoneNumber,
    string password,
    string confirmPassword,
    string streetAddress,
    string city,
    string state,
    string postalCode)
{
    Random random = new Random();
    string email = $"newuser{random.Next(100, 1000000)}@gmail.com";

```

```

        driver.Url = "https://localhost:7217/";
        mainPage.LoginButtonClick();
        bool LoginStatus = loginPage.Login(PersonalData.AdminUsername, PersonalData.AdminPassword);
        bool CreateStatus = false;
        if (LoginStatus)
        {
            mainPage.ContentManagementDropdownClick();
            mainPage.ContentManagementCreateUserButtonClick();
            createUserPage
                .NewCustomerCreate(
                    email,
                    name,
                    phoneNumber,
                    password,
                    confirmPassword,
                    streetAddress,
                    city,
                    state,
                    postalCode);
            mainPage.ContentManagementDropdownClick();
            mainPage.ContentManagementManageUserButtonClick();
            CreateStatus = createUserPage.CheckUserExist(email);
        }

        Assert.That(CreateStatus, Is.EqualTo(true));
    }

    [TearDown]
    public void TearDown()
    {
        driver.Close();
        driver.Quit();
    }
}
}

```

CategoryPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class CategoryPage : MainPage
{
    private readonly
        By NewCategoryButton =
            By.CssSelector(
                "body > div > main > div > div > div.col-6.text-end > a");

    private readonly
        By CategoryNameInput =
            By.CssSelector(
                "#Name");

    private readonly
        By DisplayOrderInput =
            By.CssSelector(
                "#DisplayOrder");

    private readonly
        By CreateButton =
            By.CssSelector("body > div > main > div > div.card-body.p-4 > form > div > div.row > div:nth-child(1) > button");

    private readonly
        By DeleteButton =
            By.CssSelector("body > div > main > form > div > div:nth-child(4) > div:nth-child(1) > button");

    private readonly
        By EditCategoryNameInput =
            By.CssSelector("#Name");

    private readonly
        By EditDisplayOrderInput =
            By.CssSelector("#DisplayOrder");

    private readonly
        By UpdateButton =
            By.CssSelector("body > div > main > form > div > div:nth-child(4) > div:nth-child(1) > button");

    public CategoryPage(IWebDriver driver) : base(driver)
    {
    }
}

```

```

public void NewCategoryButtonClick()
{
    _waiter.Until(d => d.FindElement(NewCategoryButton)).Click();
}

public bool NewCategoryCreate(string name, string displayOrder)
{
    bool status = false;
    _waiter.Until(d => d.FindElement(CategoryNameInput)).SendKeys(name);
    _waiter.Until(d => d.FindElement(DisplayOrderInput)).SendKeys(displayOrder);
    _waiter.Until(d => d.FindElement(CreateButton)).Click();

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        IWebElement categoryNameCell = row.FindElement(By.XPath("./td[1]"));

        string categoryName = categoryNameCell.Text;

        if (categoryName.Equals(name, StringComparison.OrdinalIgnoreCase))
        {
            status = true;
        }
    }
    return status;
}

public bool EditCategory(string name, string newName, string displayOrder)
{
    bool status = false;

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        Thread.Sleep(1000);
        IWebElement categoryNameCell = row.FindElement(By.XPath("./td[1]"));

        string categoryName = categoryNameCell.Text;

        if (categoryName.Equals(name, StringComparison.OrdinalIgnoreCase))
        {
            IWebElement Cell = row.FindElement(By.XPath("./td[3]/div/a[1]"));
            Cell.Click();
            Thread.Sleep(1000);
            _waiter.Until(d => d.FindElement(EditCategoryNameInput)).Clear();
            _waiter.Until(d => d.FindElement(EditCategoryNameInput)).SendKeys(newName);
            _waiter.Until(d => d.FindElement(EditDisplayOrderInput)).Clear();
            _waiter.Until(d => d.FindElement(EditDisplayOrderInput)).SendKeys(displayOrder);

            _waiter.Until(d => d.FindElement(UpdateButton)).Click();
            return true;
        }
    }
    return status;
}

public bool DeleteCategory(string name)
{
    bool status = false;

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        Thread.Sleep(1000);
        IWebElement categoryNameCell = row.FindElement(By.XPath("./td[1]"));

        string categoryName = categoryNameCell.Text;

        if (categoryName.Equals(name, StringComparison.OrdinalIgnoreCase))
        {
            IWebElement Cell = row.FindElement(By.XPath("./td[3]/div/a[2]"));
            Cell.Click();
            Thread.Sleep(1000);
            _waiter.Until(d => d.FindElement(DeleteButton)).Click();
            return true;
        }
    }
    return status;
}
}

```

CompanyPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class CompanyPage : MainPage
{
    private readonly
        By NewProductButton =
            By.CssSelector(
                "body > div > main > div > div > div.col-6.text-end > a");

    private readonly
        By NameInput =
            By.CssSelector(
                "#Name");
    private readonly
        By PhoneNumberInput =
            By.CssSelector(
                "#PhoneNumber");
    private readonly
        By StreetAddressInput =
            By.CssSelector(
                "#StreetAddress");
    private readonly
        By CityInput =
            By.CssSelector(
                "#City");
    private readonly
        By StateInput =
            By.CssSelector(
                "#State");
    private readonly
        By PostalCodeInput =
            By.CssSelector(
                "#PostalCode");

    private readonly
        By CreateButton =
            By.CssSelector("body > div > main > div > div.card-body.p-4 > form > div > div > div > div.row > div:nth-child(1)
> button");

    private readonly
        By DeleteButton =
            By.CssSelector("body > div > main > form > div > div:nth-child(8) > div:nth-child(1) > button");

    private readonly
        By UpdateButton =
            By.CssSelector("body > div > main > div > div.card-body.p-4 > form > div > div > div > div.row > div:nth-child(1)
> button");

    public CompanyPage(IWebDriver driver) : base(driver)
    {
    }

    public void NewCompanyButtonClick()
    {
        _waiter.Until(d => d.FindElement(NewProductButton)).Click();
    }

    public bool NewCompanyCreate(
        string name,
        string phoneNumber,
        string streetAddress,
        string city,
        string state,
        string postalCode
    )
    {
        bool status = false;

        _waiter.Until(d => d.FindElement(NameInput)).SendKeys(name);
        _waiter.Until(d => d.FindElement(PhoneNumberInput)).SendKeys(phoneNumber);
        _waiter.Until(d => d.FindElement(StreetAddressInput)).SendKeys(streetAddress);
        _waiter.Until(d => d.FindElement(CityInput)).SendKeys(city);
        _waiter.Until(d => d.FindElement(StateInput)).SendKeys(state);
        _waiter.Until(d => d.FindElement(PostalCodeInput)).SendKeys(postalCode);

        _waiter.Until(d => d.FindElement(CreateButton)).Click();

        Thread.Sleep(2000);
        IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    }
}

```



```

foreach (IWebElement row in rows)
{
    IWebElement companyNameCell = row.FindElement(By.XPath("./td[1]"));

    string companyName = companyNameCell.Text;

    if (companyName.Equals(name, StringComparison.OrdinalIgnoreCase))
    {
        status = true;
    }
}
return status;
}

public bool EditCompanyData(
    string name,
    string newName,
    string phoneNumber,
    string streetAddress,
    string city,
    string state,
    string postalCode)
{
    bool status = false;

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        Thread.Sleep(1000);
        IWebElement productNameCell = row.FindElement(By.XPath("./td[1]"));
        string productName = productNameCell.Text;
        if (productName.Equals(name, StringComparison.OrdinalIgnoreCase))
        {
            IWebElement Cell = row.FindElement(By.XPath("./td[6]/div/a[1]"));
            Cell.Click();
            Thread.Sleep(1000);
            _waiter.Until(d => d.FindElement(NameInput)).Clear();
            _waiter.Until(d => d.FindElement(NameInput)).SendKeys(newName);

            _waiter.Until(d => d.FindElement(PhoneNumberInput)).Clear();
            _waiter.Until(d => d.FindElement(PhoneNumberInput)).SendKeys(phoneNumber);

            _waiter.Until(d => d.FindElement(StreetAddressInput)).Clear();
            _waiter.Until(d => d.FindElement(StreetAddressInput)).SendKeys(streetAddress);

            _waiter.Until(d => d.FindElement(CityInput)).Clear();
            _waiter.Until(d => d.FindElement(CityInput)).SendKeys(city);

            _waiter.Until(d => d.FindElement(StateInput)).Clear();
            _waiter.Until(d => d.FindElement(StateInput)).SendKeys(state);

            _waiter.Until(d => d.FindElement(UpdateButton)).Click();
            return true;
        }
    }
    return status;
}

public bool DeleteCompany(string name)
{
    bool status = false;

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        Thread.Sleep(1000);
        IWebElement categoryNameCell = row.FindElement(By.XPath("./td[1]"));

        string categoryName = categoryNameCell.Text;

        if (categoryName.Equals(name, StringComparison.OrdinalIgnoreCase))
        {
            IWebElement Cell = row.FindElement(By.XPath("./td[6]/div/a[2]"));
            Cell.Click();
            Thread.Sleep(1000);
            _waiter.Until(d => d.FindElement(DeleteButton)).Click();
            return true;
        }
    }
    return status;
}
}

```

CreateUserPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class CreateUserPage : MainPage
{
    private readonly
        By NewProductButton =
            By.CssSelector(
                "body > div > main > div > div > div.col-6.text-end > a");

    private readonly
        By EmailInput =
            By.CssSelector("#Input_Email");

    private readonly
        By NameInput =
            By.CssSelector(
                "#Input_Name");

    private readonly
        By PhoneNumberInput =
            By.CssSelector(
                "#Input_PhoneNumber");

    private readonly
        By PasswordInput =
            By.CssSelector("#Input_Password");

    private readonly
        By ConfirmPasswordInput =
            By.CssSelector("#Input_ConfirmPassword");

    private readonly
        By StreetAddressInput =
            By.CssSelector(
                "#Input_StreetAddress");

    private readonly
        By CityInput =
            By.CssSelector(
                "#Input_City");

    private readonly
        By StateInput =
            By.CssSelector(
                "#Input_State");

    private readonly
        By PostalCodeInput =
            By.CssSelector(
                "#Input_PostalCode");

    private readonly
        By RegisterButton =
            By.CssSelector(
                "#registerSubmit");

    public CreateUserPage(IWebDriver driver) : base(driver)
    {
    }

    public void NewCompanyButtonClick()
    {
        _waiter.Until(d => d.FindElement(NewProductButton)).Click();
    }

    public void NewCustomerCreate(
        string email,
        string name,
        string phoneNumber,
        string password,
        string confirmPassword,
        string streetAddress,
        string city,
        string state,
        string postalCode
    )
    {
        _waiter.Until(d => d.FindElement(EmailInput)).SendKeys(email);
        _waiter.Until(d => d.FindElement(NameInput)).SendKeys(name);
        _waiter.Until(d => d.FindElement(PhoneNumberInput)).SendKeys(phoneNumber);
        _waiter.Until(d => d.FindElement>PasswordInput)).SendKeys(password);
    }

```

```

        _waiter.Until(d => d.FindElement(ConfirmPasswordInput)).SendKeys(confirmPassword);
        _waiter.Until(d => d.FindElement(StreetAddressInput)).SendKeys(streetAddress);
        _waiter.Until(d => d.FindElement(CityInput)).SendKeys(city);
        _waiter.Until(d => d.FindElement(StateInput)).SendKeys(state);
        _waiter.Until(d => d.FindElement(PostalCodeInput)).SendKeys(postalCode);

        _waiter.Until(d => d.FindElement(RegisterButton)).Click();
    }

    public bool CheckUserExist(string email)
    {
        bool status = false;

        Thread.Sleep(2000);
        IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
        foreach (IWebElement row in rows)
        {
            IWebElement userEmailCell = row.FindElement(By.XPath("./td[2]"));

            string userEmail = userEmailCell.Text;

            if (userEmail.Equals(email, StringComparison.OrdinalIgnoreCase))
            {
                status = true;
            }
        }

        return status;
    }
}

```

LoginPage.cs

```

using CourseWorkWeb.Tests.UtilityLibrary;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class LoginPage : MainPage
{
    private readonly By EmailTextBox = By.CssSelector(
        "#Input_Email");

    private readonly By PasswordTextBox = By.CssSelector(
        "#Input_Password");

    private readonly
        By LogInButton =
        By.CssSelector("#login-submit");

    private readonly
        By LoginForm = By.XPath("#account");

    private readonly
        By LockedOut = By.CssSelector("body > div > main > header > h1");

    public LoginPage(IWebDriver _driver) : base(_driver)
    {
    }

    public bool Login(string email, string password)
    {
        try
        {
            _waiter.Until(d => d.FindElement(EmailTextBox)).SendKeys(email);
            _waiter.Until(d => d.FindElement>PasswordTextBox)).SendKeys(password);
            _waiter.Until(d => d.FindElement(LogInButton)).Click();

            try
            {
                string username = _waiter.Until(d => d.FindElement(MainPage.UsernameInsideSystemAfterLogin)).Text;
                if (username.Equals($"Hello {email}!"))
                {
                    return true;
                }
            }
            else
            {
                return false;
            }
        }
        catch (Exception e)
        {
            string lockedOut = _waiter.Until(d => d.FindElement(LockedOut)).Text;

```

```

        if (lockedOut.Equals("Locked Out"))
        {
            return false;
        }
    }

    }
    catch (Exception e)
    {
        return false;
    }

    return false;
}
}

```

MainPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class MainPage
{
    protected readonly IWebDriver _driver;
    protected readonly WebDriverWait _waiter;

    protected readonly
        By LoginButton =
            By.CssSelector(
                "#login");

    protected static readonly
        By RegisterButton =
            By.CssSelector(
                "#register");

    protected static readonly
        By CartButton =
            By.CssSelector("body > header > nav > div > div > ul.navbar-nav.me-auto > li:nth-child(3) > a");

    protected static readonly
        By HomeButton =
            By.CssSelector("body > header > nav > div > div > ul.navbar-nav.me-auto > li:nth-child(1) > a");

    protected static readonly
        By ManageOrderButton =
            By.CssSelector("body > header > nav > div > div > ul.navbar-nav.me-auto > li:nth-child(2) > a");

    protected static readonly
        By ContentManagementDropdown =
            By.CssSelector("body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > a");

    protected static readonly
        By ContentManagementCategoryButton =
            By.CssSelector(
                "body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > ul > li:nth-child(1) > a");

    protected static readonly
        By ContentManagementProductButton =
            By.CssSelector(
                "body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > ul > li:nth-child(3) > a");

    protected static readonly
        By ContentManagementCompanyButton =
            By.CssSelector(
                "body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > ul > li:nth-child(4) > a");

    protected static readonly
        By ContentManagementCreateUserButton =
            By.CssSelector(
                "body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > ul > li:nth-child(5) > a");

    protected static readonly
        By ContentManagementManageUserButton =
            By.CssSelector(
                "body > header > nav > div > div > ul.navbar-nav.me-auto > li.nav-item.dropdown > ul > li:nth-child(6) > a");

    protected static readonly
        By AddToCartButton =
            By.CssSelector(
                "div:nth-child(8) > div > button");

    protected static readonly By UsernameInsideSystemAfterLogin = By.CssSelector("#manage");
}

```

```

protected static By LogOutButton = By.CssSelector("#logout");

public MainPage(IWebDriver webDriver)
{
    _driver = webDriver;
    _waiter = new WebDriverWait(_driver, TimeSpan.FromSeconds(8));
}

public void LoginButtonClick()
{
    _waiter.Until(d => d.FindElement(LoginButton)).Click();
}

public void RegisterButtonClick()
{
    _waiter.Until(d => d.FindElement(RegisterButton)).Click();
}

public void CartButtonClick()
{
    _waiter.Until(d => d.FindElement(CartButton)).Click();
}

public void HomeButtonClick()
{
    _waiter.Until(d => d.FindElement(HomeButton)).Click();
}

public void ManageOrderClick()
{
    _waiter.Until(d => d.FindElement(ManageOrderButton)).Click();
}

public void ContentManagementDropdownClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementDropdown)).Click();
}

public void ContentManagementCategoryButtonClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementCategoryButton)).Click();
}

public void ContentManagementProductButtonClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementProductButton)).Click();
}

public void ContentManagementCompanyButtonClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementCompanyButton)).Click();
}

public void ContentManagementCreateUserButtonClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementCreateUserButton)).Click();
}

public void ContentManagementManageUserButtonClick()
{
    _waiter.Until(d => d.FindElement(ContentManagementManageUserButton)).Click();
}

public void LogOutButtonClick()
{
    _waiter.Until(d => d.FindElement(LogOutButton)).Click();
}

public void DetailsButtonNClick(int n)
{
    _waiter.Until(d => d.FindElement(By.
        CssSelector($"body > div > main > div.row.pb-3 > div:nth-child({n}) > div > div > div > div:nth-child(3) > a"))
        .Click();
    Thread.Sleep(500);
}

public void AddToCartButtonClick()
{
    _waiter.Until(d => d.FindElement(AddToCartButton)).Click();
    Thread.Sleep(500);
}

public void ScrollDown(int pixels)
{
    IJavaScriptExecutor js = (IJavaScriptExecutor)_driver;
    js.ExecuteScript($"window.scrollTo(0, {pixels});");
    Thread.Sleep(500);
}

```

```

public bool CheckThatBookExistInsideCart(string title)
{
    _waiter.Until(d => d.FindElement(CartButton)).Click();
    for (int i = 1; i < 6; i++)
    {
        string titleatCart = _waiter.Until(d => d.FindElement(By.
            CssSelector($"body > div > main > form > div.card-body.my-4 > div.row.mb-3.pb-3 > div.col-md-10.offset-
md-1 > div:nth-child({i}) > div.col-12.col-lg-6.pt-md-3 > h5 > strong")))
            .Text);
        if (title.Equals(titleatCart))
        {
            return true;
        }
    }
    return false;
}
}

```

ProductAdminPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class ProductAdminPage : MainPage
{
    private readonly
        By NewProductButton =
            By.CssSelector(
                "body > div > main > div > div > div.col-6.text-end > a");

    private readonly
        By TitleInput =
            By.CssSelector(
                "#Product_Title");

    private readonly
        By DescriptionInput =
            By.CssSelector(
                "#Product_Description");

    private readonly
        By ISBNInput =
            By.CssSelector(
                "#Product_ISBN");

    private readonly
        By AuthorInput =
            By.CssSelector(
                "#Product_Author");

    private readonly
        By ListPriceInput =
            By.CssSelector(
                "#Product_ListPrice");

    private readonly
        By PriceInput =
            By.CssSelector(
                "#Product_Price");

    private readonly
        By Price50Input =
            By.CssSelector(
                "#Product_Price50");

    private readonly
        By Price100Input =
            By.CssSelector(
                "#Product_Price100");

    private readonly
        By CategoryIdSelector =
            By.CssSelector(
                "#Product_CategoryId");

    private readonly
        By CategoryIdSelectFiction =
            By.CssSelector(
                "#Product_CategoryId > option:nth-child(8)");
}

```

```

private readonly
    By CreateButton =
        By.CssSelector(
            "body > div > main > div > div.card-body.p-4 > form > div > div.col-10 > div > div.row > div:nth-child(1) >
button");

private readonly
    By DeleteButton =
        By.CssSelector("body > div > main > form > div > div:nth-child(10) > div:nth-child(1) > button");

private readonly
    By UpdateButton =
        By.CssSelector(
            "body > div > main > div > div.card-body.p-4 > form > div > div.col-10 > div > div.row > div:nth-child(1) >
button");

public ProductAdminPage(IWebDriver driver) : base(driver)
{
}

public void NewProductButtonClick()
{
    _waiter.Until(d => d.FindElement(NewProductButton)).Click();
}

public bool NewProductCreate(
    string title,
    string description,
    string ISBN,
    string author,
    string listPrice,
    string price,
    string price50,
    string price100
)
{
    bool status = false;

    _waiter.Until(d => d.FindElement(TitleInput)).SendKeys(title);
    _waiter.Until(d => d.FindElement(DescriptionInput)).SendKeys(description);
    _waiter.Until(d => d.FindElement(ISBNInput)).SendKeys(ISBN);
    _waiter.Until(d => d.FindElement(AuthorInput)).SendKeys(author);

    _waiter.Until(d => d.FindElement(ListPriceInput)).Clear();
    _waiter.Until(d => d.FindElement(ListPriceInput)).SendKeys(listPrice);

    _waiter.Until(d => d.FindElement(PriceInput)).Clear();
    _waiter.Until(d => d.FindElement(PriceInput)).SendKeys(price);

    _waiter.Until(d => d.FindElement(Price50Input)).Clear();
    _waiter.Until(d => d.FindElement(Price50Input)).SendKeys(price50);

    _waiter.Until(d => d.FindElement(Price100Input)).Clear();
    _waiter.Until(d => d.FindElement(Price100Input)).SendKeys(price100);

    _waiter.Until(d => d.FindElement(CategoryIdSelector)).Click();
    Thread.Sleep(500);
    _waiter.Until(d => d.FindElement(CategoryIdSelectFiction)).Click();
    Thread.Sleep(500);

    _waiter.Until(d => d.FindElement(CreateButton)).Click();

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)
    {
        IWebElement categoryNameCell = row.FindElement(By.XPath("./td[1]"));

        string categoryName = categoryNameCell.Text;

        if (categoryName.Equals(title, StringComparison.OrdinalIgnoreCase))
        {
            status = true;
        }
    }

    return status;
}

public bool EditTitleInProduct(string name, string newName)
{
    bool status = false;

    Thread.Sleep(2000);
    IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
    foreach (IWebElement row in rows)

```

```

        {
            Thread.Sleep(1000);
            IWebElement productNameCell = row.FindElement(By.XPath("./td[1]"));
            string productName = productNameCell.Text;
            if (productName.Equals(name, StringComparison.OrdinalIgnoreCase))
            {
                IWebElement Cell = row.FindElement(By.XPath("./td[6]/div/a[1]"));
                Cell.Click();
                Thread.Sleep(1000);
                _waiter.Until(d => d.FindElement(TitleInput)).Clear();
                _waiter.Until(d => d.FindElement(TitleInput)).SendKeys(newName);

                _waiter.Until(d => d.FindElement(UpdateButton)).Click();
                return true;
            }
        }

        return status;
    }
}

```

UserListPage.cs

```

using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

namespace CourseWorkWeb.Tests.Pages;

public class UserListPage : MainPage
{
    public UserListPage(IWebDriver driver) : base(driver)
    {
    }

    public bool LockUnlockUserWithEmail(string email)
    {
        bool status = false;
        Thread.Sleep(2000);
        IList<IWebElement> rows = _driver.FindElements(By.CssSelector(".table tbody tr"));
        foreach (IWebElement row in rows)
        {
            Thread.Sleep(1000);
            IWebElement emailCell = row.FindElement(By.XPath("./td[2]"));

            string getEmail = emailCell.Text;

            if (getEmail.Equals(email, StringComparison.OrdinalIgnoreCase))
            {
                IWebElement Cell = row.FindElement(By.XPath("./td[5]"));
                Cell.Click();
                Thread.Sleep(5000);
                return true;
            }
        }
        return status;
    }
}

```

PersonalData.cs

```

namespace CourseWorkWeb.Tests.UtilityLibrary;

public class PersonalData
{
    public static readonly string AdminUsername = "admin@gmail.com";
    public static readonly string CustomerUsername = "customer@gmail.com";
    public static readonly string CompanyUsername = "company@gmail.com";
    public static readonly string EmployeeUsername = "employee@gmail.com";

    public static readonly string AdminPassword = "Password_123";
    public static readonly string CustomerPassword = "Password_123";
    public static readonly string CompanyPassword = "Password_123";
    public static readonly string EmployeePassword = "Password_123";
}

```