


ÇUKUROVA UNIVERSITY ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CEN440 - E-Commerce

E-Commerce Web Application
(Furniture Sales)

By

2019556032 - Oğuzhan Güney

2019556055 - Ali Can Sarıboğa

Advisor

Asst. Prof. MEHMET SARIGÜL

2024

ABSTRACT

This report extensively describes the development process and utilized technologies of an e-commerce website. The aim of the project is to contribute to online shopping platforms, replacing traditional shopping methods, and to provide a sample solution for enhancing students' web development skills. The report elaborates on significant aspects such as the methodology of the project process, employed technologies, system architecture, code structure, and website design. Additionally, it includes a detailed evaluation of the project's outcomes, along with suggestions for future improvements. This report is designed to elucidate the successful development process of an e-commerce website using modern software development practices and tools.

CONTENTS

1. Purpose and Scope of the Project

1.1. Purpose of the Project

In today's world, the acceleration of digitalization and the widespread use of the internet have given significant momentum to the e-commerce sector. As traditional shopping methods are increasingly being replaced by online shopping, it has become essential for businesses to establish a presence on digital platforms. This project aims to develop a user-friendly and functional e-commerce website, providing a model solution for businesses looking to enter the digital market. Additionally, the development of this e-commerce site offers us students an important opportunity to enhance our web development skills and gain necessary experience for real-world applications.

1.2. Objectives of the Project

I. User-Friendly Interface: Design an intuitive and attractive user interface that allows users to easily navigate, search for products, and make purchases.

II. Product Management: Create a backend system that enables administrators to easily add, update, and manage stock for products.

III. Mobile Compatibility: Ensure the site is compatible with different devices and screen sizes, providing seamless usability on mobile devices.

IV. Performance and Speed: Enhance user experience with fast loading times and high performance.

1.3. Importance of the Project

This project is of great importance in enhancing the competitiveness of both students and businesses in the digital world. Students will make a significant step in their careers by acquiring technical and practical knowledge during the project process. Businesses can be inspired by such projects to develop their own e-commerce solutions, reaching a wider audience. Additionally, contributing to the growth of the digital economy supports the overall transformation of commerce and the business world.

2. Project Plan and Process

Planning
To successfully complete the project, the following phases were undertaken:

2.1. Requirements Analysis

At the initial stage of the project, the needs and expectations of the customers and users were analyzed. The information gathered during this phase was used to determine the requirements of the e-commerce site. The actions users would perform on the site and the features that were prioritized were identified. These requirements were documented, forming the foundational building blocks of the project.

2.2. Design:

After completing the requirements analysis, a design for the site was created. During this phase, user interface (UI) designs were developed.

2.3. Devolopment:

Following the design phase, the coding phase of the site commenced. During this phase, front-end and back-end development was carried out. Front-end development utilized technologies such as HTML, CSS, and JavaScript, while back-end development was done using C# and the Entity Framework Core. A database design was created to store and manage product, user, and order data.

2.4. Testing:

Once development was complete, comprehensive tests were conducted to check the functionality and performance of the site.

3. System Architecture and Design

In this section, the system architecture and design details used for the project will be explained. The project has been developed using the N-tier architecture. Below, the layers of the project and the technologies used are detailed:

3.1. N-Tier Architecture

The project has been designed using the N-tier architecture. Each layer has specific responsibilities and enables the partitioning of the system. The layers of the project are as follows:

1. Presentation Layer :
   - Technologies: HTML, CSS, JavaScript, Bootstrap
   - Creates the user interface and interacts with the user.
   - Runs in the browser and displays the website to the user.

2. Business Layer :
   - Technologies: C#
   - Executes the business logic and manages the application's business processes.
   - Provides an interface between the data access layer and the presentation layer.

3. Data Access Layer :
   - Technologies: C#, Entity Framework, MS SQL
   - Interacts with the database, retrieves data from and sends data to the database.
   - Performs database operations and returns them to the business layer.

4. Entity Layer :
   - Technologies: C#
   - Contains entity classes representing database tables and relationships.
   - Facilitates database operations and reduces dependency on the data access layer.

3.2. Used Technologies

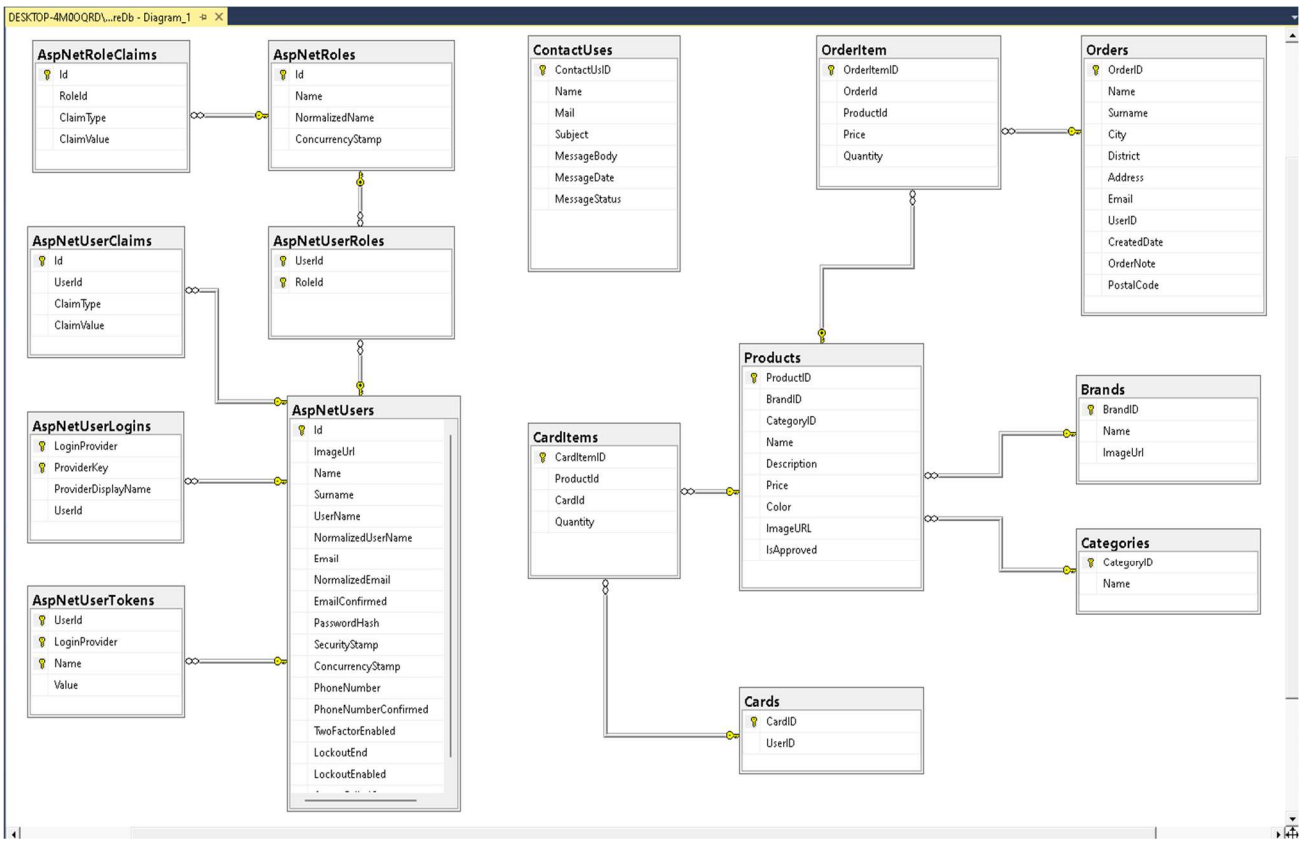
HTML and CSS: Used for designing and styling the user interface.
JavaScript: Utilized for client-side interactions and dynamic features.
Bootstrap: Employed for rapid and responsive user interface development.
C# (.NET Core Framework): Utilized for the business layer and data access layer.
Entity Framework: Used for managing database operations.
MS SQL: Utilized as the database.
Identity Library: Employed for user authentication and authorization processes.

3.3. Database Design

The database used for the project is designed using MS SQL. The following tables and relationships are utilized:

- Products: Stores product information.
- Brands: Stores product brands.
- Categories: Stores product categories.
- Cards: Stores user cart information.
- CardItems: Stores the products added to carts.
- Orders: Stores user order information.
- ContactUses: Stores user contact information.
- Identity Framework Tables: Tables used for user authentication and authorization processes.

This architecture and design provide a modular, scalable, and easily maintainable structure for the project. Each layer has specific responsibilities, ensuring that the code is more organized and manageable. Additionally, a modern and responsive design is created for the user interface, allowing users to interact easily and perform their transactions.



4. Technologies and Tools

This section details the technologies, programming languages, libraries, and development tools used in the development of the project. The project has been effectively carried out by integrating modern software development practices and tools.

4.1. Development Environment

- Visual Studio 2022: The primary development environment used during the project development. This IDE provides powerful tools for C# coding, debugging, and developing ASP.NET Core applications.
- SQL Server Management Studio (SSMS): A tool used for database management and design. SSMS facilitates interaction with MS SQL Server, allowing us to create tables, execute database queries, and manage data.

4.2. Programming Languages and Frameworks

- ASP.NET Core 8.0: A modern, performance-oriented framework used for web application development. This framework enables the creation of secure and scalable web applications.
- C#: The primary programming language used in the presentation layer, business logic, and data access layers. With its object-oriented and high-performance nature, it allows for the development of advanced application components.

4.3. Used Libraries

The libraries used in the project enable leveraging security, data access, and Object-Relational Mapping (ORM) features provided by ASP.NET Core:

- Microsoft.AspNetCore.Identity: A library that facilitates user authentication and authorization processes. This library standardizes security and identity management processes, enhancing application security.
- Microsoft.EntityFrameworkCore: Entity Framework Core is an ORM library used for database operations. It manages database queries and relationships between C# objects and database tables.
- Microsoft.EntityFrameworkCore.Design, Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Tools: These libraries provide design-time functionality of Entity Framework Core, integration with SQL Server, and necessary tools during development.
- Microsoft.VisualStudio.Web.CodeGeneration.Design: Used for code generation and automated tasks in ASP.NET Core projects.

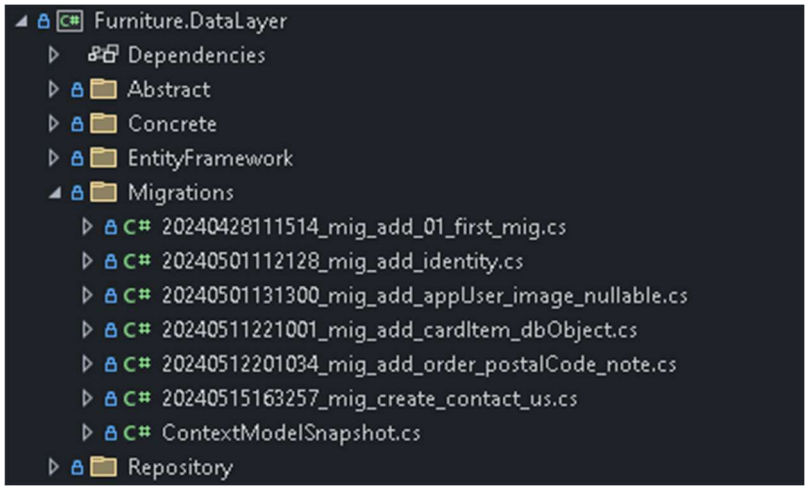
4.4. Integration of Tools and Technologies

Throughout the development process of this project, a combination of tools and technologies has been used. This integration has facilitated rapid and efficient development, testing, and deployment of the project. The combination of Visual Studio 2022 and SSMS allows for centralized management of both code writing and database management. Additionally, the use of ASP.NET Core and C# forms a robust backend structure, while Entity Framework Core simplifies and accelerates database operations.

5. Code Structure and Site Appearance

In the Furniture Web Application project; N-Tier architecture consisting of 4 layers: Presentation, Business, Entity and User Interface was used. You can find the details of this architecture in the System Architecture and Design section.

The entities used in the project are given as diagrams in the 3.1 database section. After these entities were written in .net core, the database was updated using the "First Code" method. In order for this code to be added to the database in a meaningful way, a "migration" process is performed. With this process, the database is created as empty tables.



Then, we add the repository codes made in connection with the context (database) in the project.

```
namespace Furniture.DataLayer.Repository
{
    8 references
    public class Repository<T ,TContext> : IRepository<T> where T : class where TContext : DbContext, new()
    {
        8 references
        public virtual void Create(T entity)...

        7 references
        public virtual void Delete(T entity)...

        8 references
        public virtual List<T> GetAll(Expression<Func<T, bool>> filter = null)
        {
            using (var context = new TContext())
            {
                return filter == null
                        ? context.Set<T>().ToList()
                        : context.Set<T>().Where(filter).ToList();
            }
        }

        7 references
        public virtual T GetById(int id)
        {
            using (var context = new TContext())
            {
                return context.Set<T>().Find(id);
            }
        }

        1 reference
        public virtual T GetOne(Expression<Func<T, bool>> filter)...

        8 references
        public virtual void Update(T entity)...
    }
```

With this repository, CRUD (Create-Read-Update-Delete) operations are enabled. In addition to CRUD operations, the necessary special methods have also been added to the project, and their control has been inherited in the manager using the Data Access Layer.

```
namespace Furniture.BusinessLayer.Concrete
{
    2 references
    public class CardItemManager : ICardItemService
    {
        private ICardItemDal _cardItemDal;

        0 references
        public CardItemManager(ICardItemDal cardItemDal)
        {
            _cardItemDal = cardItemDal;
        }

        2 references
        public List<CardItem> GetCardItemsByCardId(int cardId)
        {
            return _cardItemDal.GetCardItemsByCardId(cardId);
        }

        6 references
        public List<CardItem> GetCardItemsWithProducts()
        {
            return _cardItemDal.GetCardItemsWithProducts();
        }

        7 references
        public void TCreate(CardItem entity)
        {
            _cardItemDal.Create(entity);
        }

        7 references
        public void TDelete(CardItem entity)
        {
            _cardItemDal.Delete(entity);
        }

        10 references
        public List<CardItem> TGetAll()
        {
            return _cardItemDal.GetAll();
        }

        10 references
        public CardItem TGetById(int id)
        {
            return _cardItemDal.GetById(id);
        }

        2 references
        public void TUpdate(CardItem entity)
        {
            _cardItemDal.Update(entity);
        }
    }
```

After the Manager contents are added, these services are created in the constructor in the controller of the page in the User Interface and are kept ready for the relevant method to be used wherever it is.

```
namespace Furniture.UILayer.Controllers
{
    [Authorize(Roles = "Admin,Member")]
    1 reference
    public class CheckoutController : Controller
    {
        private ICardItemService _cardItemService;
        private IAppUserService _appUserService;
        private readonly UserManager<AppUser> _userManager;
        private IOrderService _orderService;
        private IOrderItemService _orderItemService;
        private ICardService _cardService;

        0 references
        public CheckoutController(ICardItemService cardItemService, IAppUserService
          appUserService, UserManager<AppUser> userManager, IOrderService orderService,
           IOrderItemService orderItemService, ICardService cardService)
        {
            _cardItemService = cardItemService;
            _appUserService = appUserService;
            _userManager = userManager;
            _orderService = orderService;
            _orderItemService = orderItemService;
            _cardService = cardService;
        }
```

It may be necessary to work with more than one entity on the controller and page. In such cases, ViewModel is used and the information in this model is used/filled by making special assignments.

```
namespace Furniture.UILayer.Models
{
    6 references
    public class CheckoutViewModel
    {
        3 references
        public List<CardItem> CardItems { get; set; }
        9 references
        public AppUser AppUser { get; set; }
        0 references
        public OrderItem OrderItem { get; set; }
        10 references
        public Order Order { get; set; }
    }
}
```

The use of the services prepared in the Constructor and the methods of those services is given as an example on the controller below.

```
0 references
public async Task<IActionResult> Index(CheckoutViewModel checkoutViewModel)
{
    var value = await _userManager.FindByNameAsync(User.Identity.Name);
    var userId = value.Id;

    var cardItems = _cardItemService.GetCardItemsWithProducts();

    if (!ModelState.IsValid)
    {
        var order = new Order()
        {
            Name = checkoutViewModel.AppUser.Name,
            Surname = checkoutViewModel.AppUser.Surname,
            City = checkoutViewModel.Order.City,
            District = checkoutViewModel.Order.District,
            Address = checkoutViewModel.Order.Address,
            PostalCode = checkoutViewModel.Order.PostalCode,
            OrderNote = checkoutViewModel.Order.OrderNote,
            Email = checkoutViewModel.AppUser.Email,
            UserID = userId.ToString(),
            CreatedDate = DateTime.Now,
        };

        _orderService.TCreate(order);
        var orderid = _orderService.GetOrderByUserId(userId).OrderID;

        foreach (var item in cardItems)
        {
            var orderItem = new OrderItem()
            {
                OrderId = orderid,
                ProductId = item.Product.ProductID,
                Quantity = item.Quantity,
                Price = item.Product.Price,
            };

            _orderItemService.TCreate(orderItem);
        }

        var cardIdNew =_cardService.GetCardByUserID(userId.ToString()).CardID;
        _cardService.ClearCard(cardIdNew.ToString());

        return RedirectToAction("Done", "Checkout");
    }

    return View(new CheckoutViewModel()
    {
        CardItems = _cardItemService.GetCardItemsWithProducts(),
        AppUser = _appUserService.TGetById(userId),
    });
}
0 references
public IActionResult Done()
{
    return View();
}
```

The use of the viewModel example mentioned above on the Web page is shown on the index page below. The submission is implemented as an example on this page. So the create/update process was implemented. Additionally, it is possible to perform both read and write operations on the same viewModel.

```
<div class="checkout-area pt-100px pb-100px">
    <div class="container">
        <form method="post">
            <div class="row">
                <div class="col-lg-7">
                    <div class="billing-info-wrap">
                        <h3>Fatura Detayları</h3>
                        <div class="row">
                            <div class="col-lg-6 col-md-6">
                                <div class="billing-info mb-20px">
                                    <label>İsim</label>
                                    <input type="text" asp-for="AppUser.Name" />
                                </div>
                            </div>
                            <div class="col-lg-6 col-md-6">
                                <div class="billing-info mb-20px">
                                    <label>Soyisim</label>
                                    <input type="text" asp-for="AppUser.Surname" />
                                </div>
                            </div>
                            <div class="col-lg-12">
                                <div class="billing-info mb-20px">
                                    <label>Adres</label>
                                    <textarea placeholder="Yeni Yüzyıl Üniversitesi, 26, Yılanlı Ayazma Sokağı, Maltepe Mahallesi, Zeytinburnu,
                                     34010, İstanbul/ Turkey" asp-for="Order.Address"></textarea>
                                </div>
                            </div>
                            @* Şehir *@
                            <div class="col-lg-12">…</div>
                            @* İlçe *@
                            <div class="col-lg-6 col-md-6">…</div>
                            @* Posta Kodu *@
                            <div class="col-lg-6 col-md-6">…</div>
                            @* Telefon *@
                            <div class="col-lg-6 col-md-6">…</div>
                            @* Email *@
                            <div class="col-lg-6 col-md-6">…</div>
                        </div>
                        <div class="checkout-account mb-30px">
                            <input class="checkout-toggle2 w-auto h-auto" type="checkbox" />
                            <label>Hesabınız Yok Mu?</label>
                        </div>
                        <div class="checkout-account-toggle open-toggle2 mb-30">
                            <button class="btn-hover login-btn"><a href="/Login/SignUp/"> Şimdi Kayıt Ol </a></button>
                        </div>
                        <div class="additional-info-wrap">
                            <h4>Eklemek İstedikleriniz</h4>
                            <div class="additional-info">
                                <label>Sipariş Notu</label>
                                <textarea placeholder="Sipariş hakkındaki notunuz veya teslimat için özel notlar." asp-for="Order.OrderNote"></
                                 textarea>
                            </div>
                        </div>
                    </div>
                </div>
```

6. Result and Evaluation

The development of this project required the effective utilization of modern technologies and software development tools. Here are the results and evaluation obtained during this project:

Technological Infrastructure:

- Powerful technological foundations such as ASP.NET Core 8.0, C#, and Entity Framework Core were utilized as the cornerstones in the development of the project. These technologies enabled the development of a fast and reliable web application.

Modular Architecture:

- The N-tier architecture made the project code more organized and manageable. Clear distinction was provided between the presentation layer, business layer, data access layer, and entity layer. This modular structure facilitated code reusability and maintenance.

Development Tools:

- Visual Studio 2022 facilitated the development process and improved the coding experience. SQL Server Management Studio (SSMS) served as a reliable tool for database design and management. These tools, used throughout the development process, enhanced productivity.

Security and Authentication:

- The Microsoft.AspNetCore.Identity library simplified user authentication and authorization processes. This ensured user security and privacy, while managing authorization processes effectively.

Performance and Scalability:

- The performance-oriented structure of ASP.NET Core ensured the application was fast and scalable. This provided the capacity to handle increased user traffic, enhancing the user experience.

Future Improvements:

- The project laid the groundwork for future enhancements. By adding more features and improving existing ones, the user experience can be further enhanced. Additionally, performance improvements and security updates can be implemented.

This project demonstrates the effective use of modern technologies and development tools. Selecting the right technology and appropriate architectural design is crucial to developing a web application that meets the needs of users, is secure, fast, and scalable. By successfully applying these principles, this project has achieved a successful outcome.

7.References

1. Microsoft. (n.d.). ASP.NET Core overview. Microsoft Docs. Retrieved May 21, 2024, from https://docs.microsoft.com/en-us/aspnet/core

2. Microsoft. (n.d.). C# guide. Microsoft Docs. Retrieved May 21, 2024, from https://docs.microsoft.com/en-us/dotnet/csharp

3. Microsoft. (n.d.). Entity Framework Core. Microsoft Docs. Retrieved May 21, 2024, from https://docs.microsoft.com/en-us/ef/core

4. Microsoft. (n.d.). Introduction to Identity on ASP.NET Core. Microsoft Docs. Retrieved May 21, 2024, from https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity

5. Microsoft. (n.d.). Visual Studio IDE. Retrieved May 21, 2024, from https://visualstudio.microsoft.com/vs/

6. Microsoft. (n.d.). SQL Server Management Studio (SSMS). Microsoft Docs. Retrieved May 21, 2024, from https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms

7. Bootstrap. (n.d.). Introduction to Bootstrap. Retrieved May 21, 2024, from https://getbootstrap.com/docs/5.3/getting-started/introduction/

8. Mozilla Developer Network (MDN). (n.d.). JavaScript guide. Retrieved May 21, 2024, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

9. Mozilla Developer Network (MDN). (n.d.). HTML basics. Retrieved May 21, 2024, from https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

10.Mozilla Developer Network (MDN). (n.d.). CSS basics. Retrieved May 21, 2024, from https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics