Картина, която съдържа текст, графична колекция

Описанието е генерирано автоматичноТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „Софтуерни и интернет технологии“

СЕМЕСТРИАЛНА ДОМАШНА РАБОТА

по дисциплината „ТСП”

на тема: „Онлайн магазин”

|  |  |
| --- | --- |
| **Изготвил:**  Адам Аяш | **Изготвил:**  Павел Бойчев |
| **Специалност**: СИТ | **Специалност**: СИТ |
| **Група: 3** | **Група: 3** |
| **Факултетен номер:** 21621553  **Проверил:** доц. В. Божикова | **Факултетен номер:** 21621538 |

Съдържание

[1.Задание 2](#_Toc167652172)

[2.Условие на проекта 3](#_Toc167652173)

[3.Изисквания 3](#_Toc167652174)

[3.1.Функционални изисквания**:** 3](#_Toc167652175)

[3.2.Не-Функционални изисквания**:** 3](#_Toc167652176)

[4.Реализация на проекта 4](#_Toc167652177)

[4.1.Проектиране**:** 4](#_Toc167652178)

[4.2.Кратко описание : 5](#_Toc167652179)

[4.3 Програмна част 10](#_Toc167652180)

[4.3.1 Sign Up Controller 10](#_Toc167652181)

[4.3.2 Login Page Controller 11](#_Toc167652182)

[4.3.3 Shopping Cart Controller 12](#_Toc167652183)

[4.3.4 Order Controller 13](#_Toc167652184)

[4.3.5 Order Class 15](#_Toc167652185)

[4.3.6 DbInitializer Class 17](#_Toc167652186)

# 

# 1.Задание

# 2.Условие на проекта

Да се разработи онлайн магазин за продажба на компютри и компютърни компоненти.

# 3.Изисквания

## 3.1.Функционални изисквания**:**

* Софтуерът трябва да позволява избиране от различните категории продукти и тяхното добавяне в кошница достъпна само за текущата клиентска сесия
* Трябва да има възможност за влизане и регистрация
* Системата съхранява данни за:
  + Продукти
  + Потребители
  + Категории
  + Поръчки

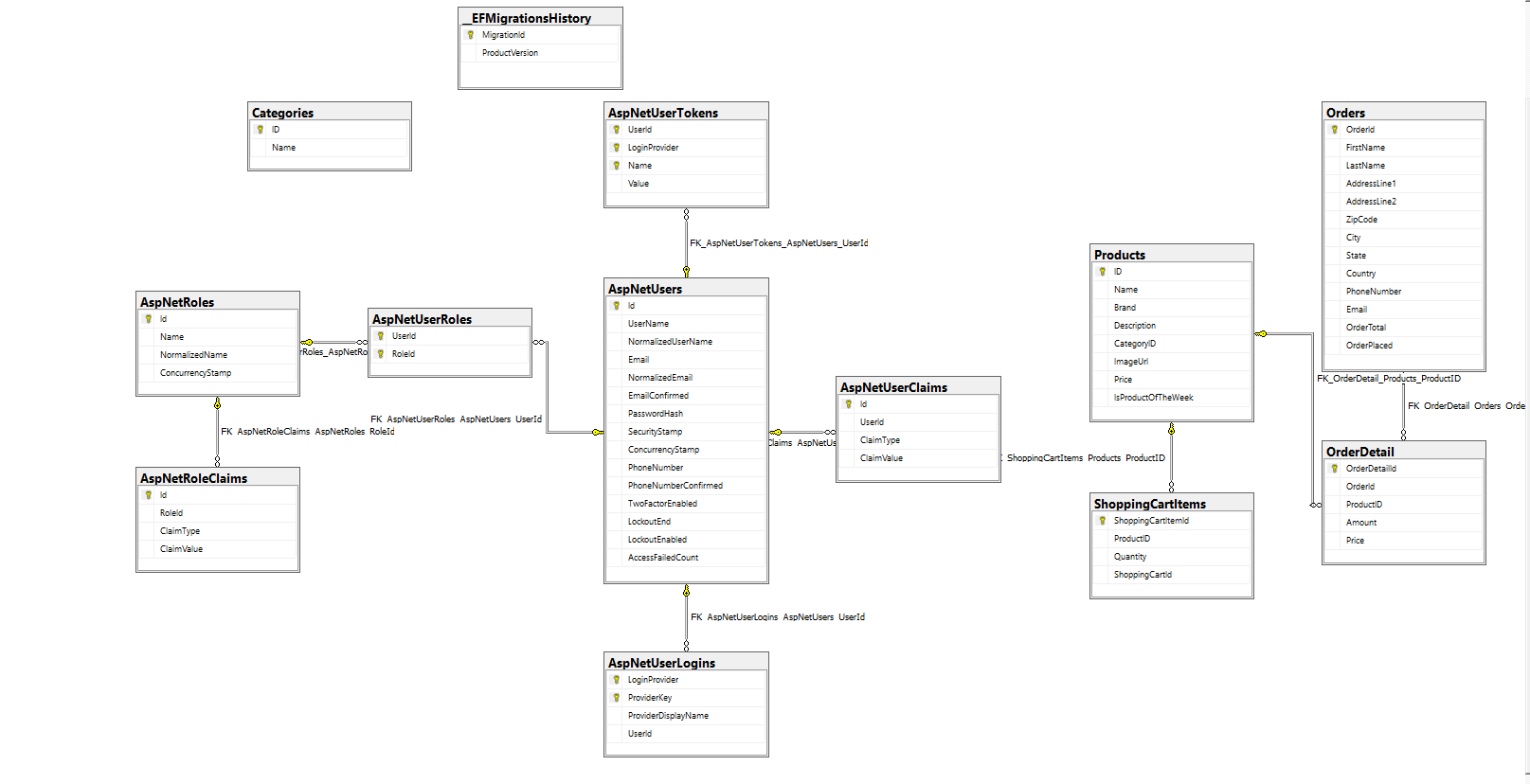
## 3.2.Не-Функционални изисквания**:**

* Използват се различни форми , за регистрация на нов потребител (въвеждат се потребителско име и парола) , и за влизане в съществуващ профил (чрез въвеждане на коректни данни за потребителско име и парола), като има налична валидация и потребителят трябва да съобрази сложността на паролата си. Поддържа се и запомняне на профила

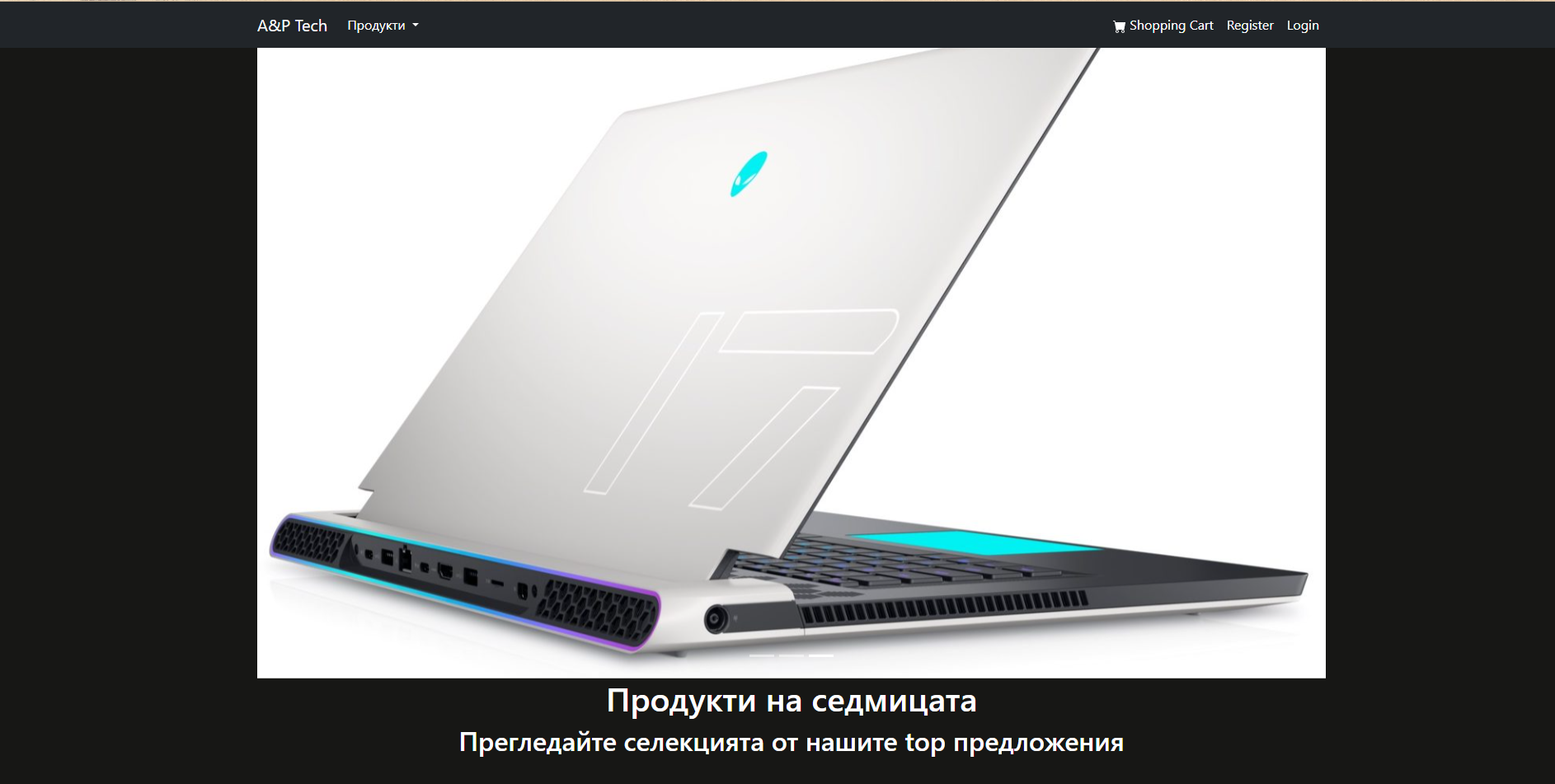
# 4.Реализация на проекта

## 4.1.Проектиране**:**

Софтуерът поддържа бази от данни, които съдържат следните таблици:

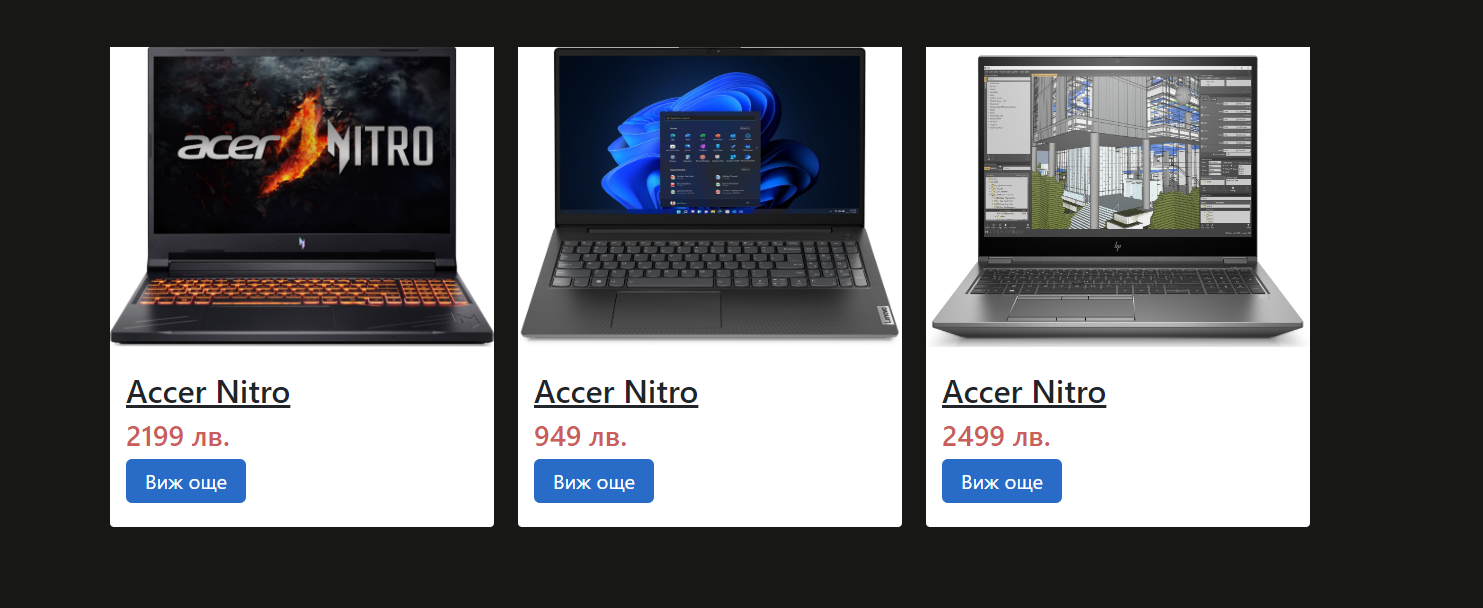
****

4.2.Кратко описание :4.2.1.Landing page

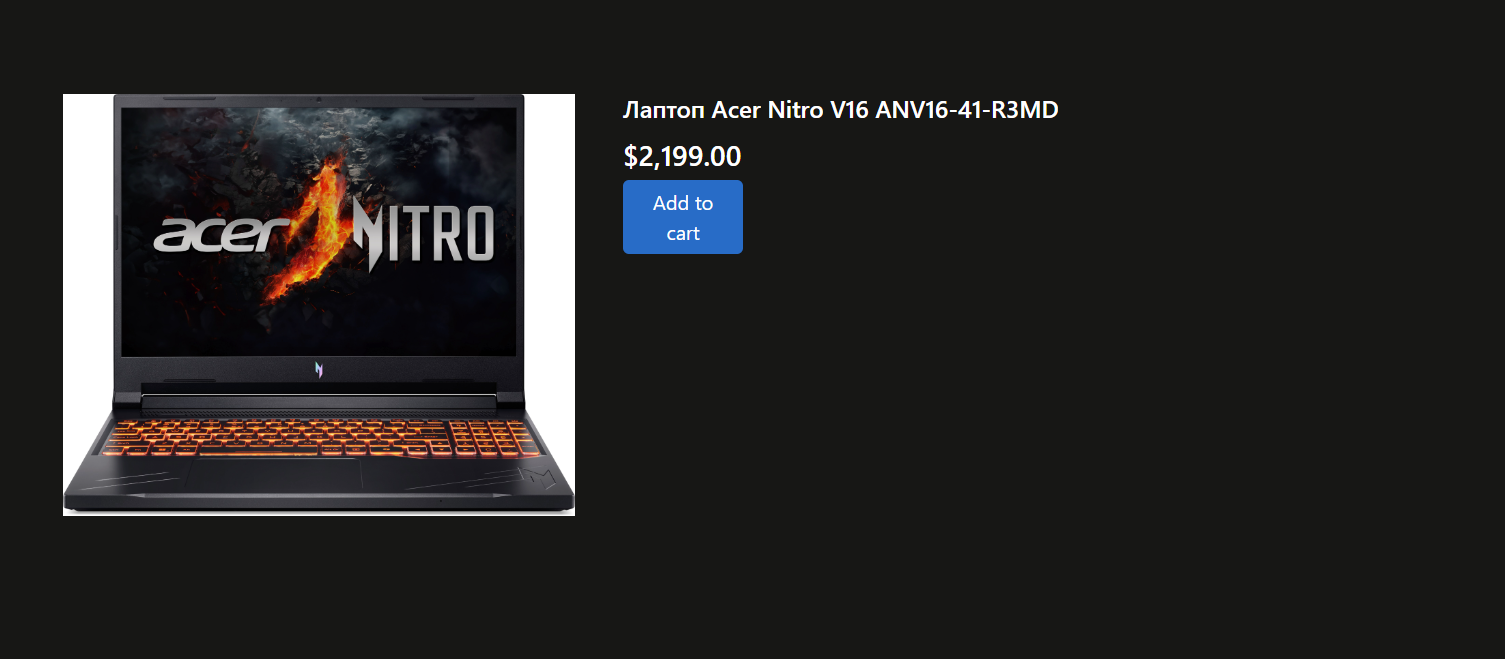


При първоначално влизане в сайта потребителят попада в така нареченият home-pаge. От тук той може свободно да разглежда наличните продукти на страницата посредством падащото меню, изброяващо различните категории продукти.

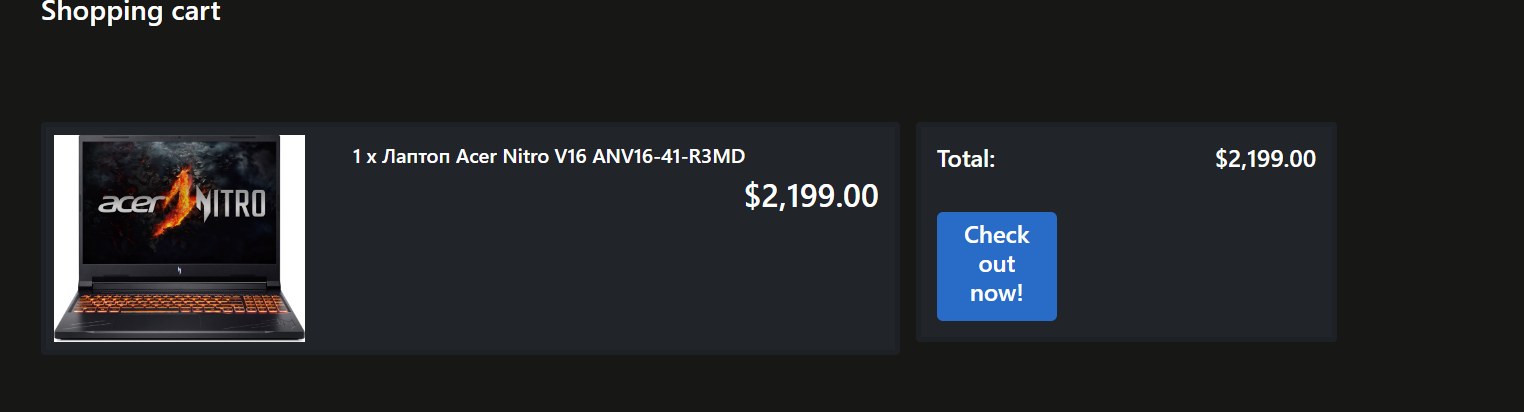
При избиране на дадена категория на потребителят ще се изведе списък с всички налични продукти от тази категория



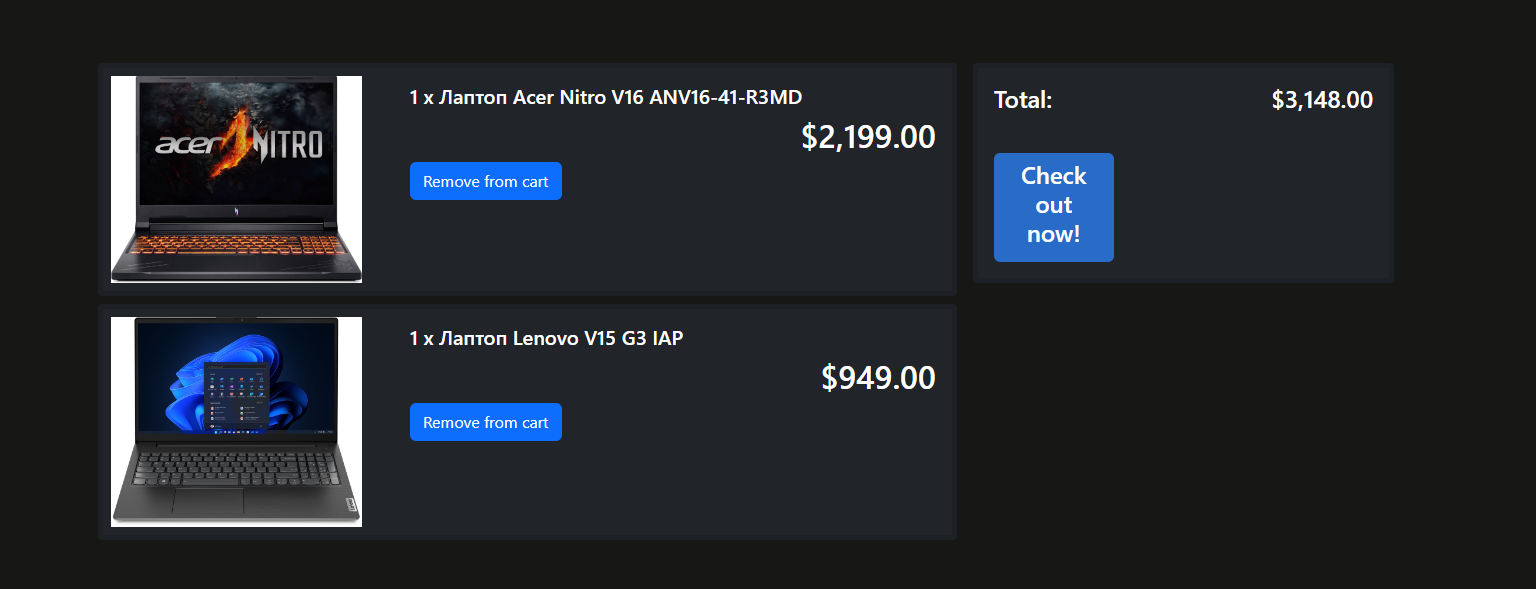
При разглеждане на продуктите потребителят има възможност на натисне бутона виж още за да му се предостави по детайлна информация за продукта.



След това потребителят може да добави продукта в количката посредством бутона Add to cart.



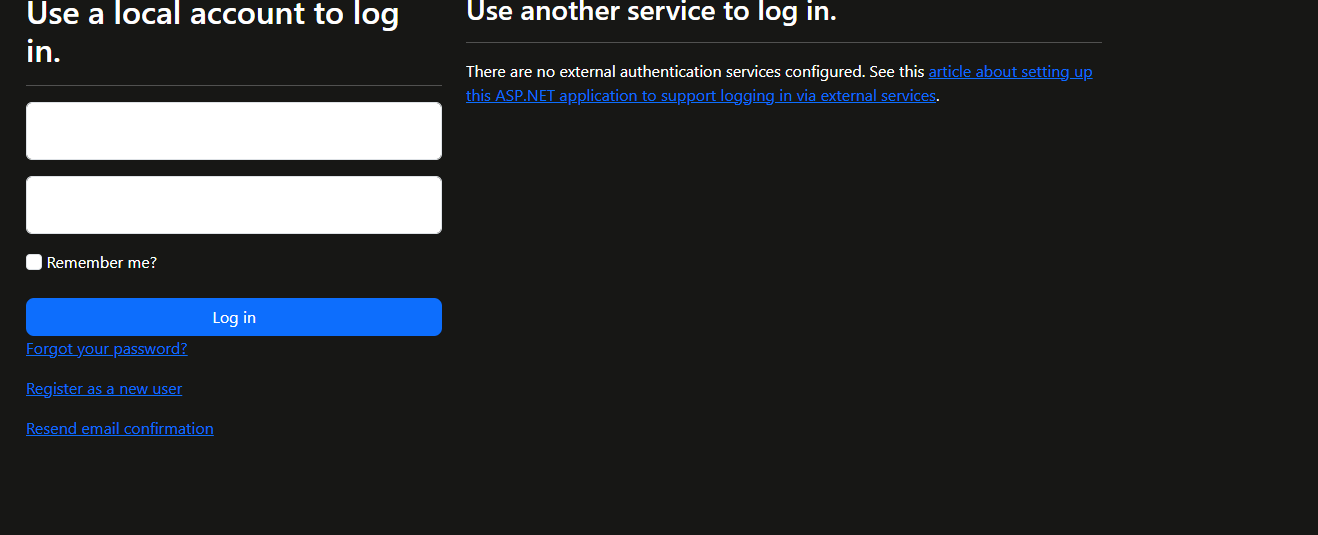
Потребителят може да разгледа добавените продукти в количката



Ако желае може да премахне продукт от количката посредством бутон Remove from cart

Като общата стойност на продуктите се сумира и се изобразява на екран

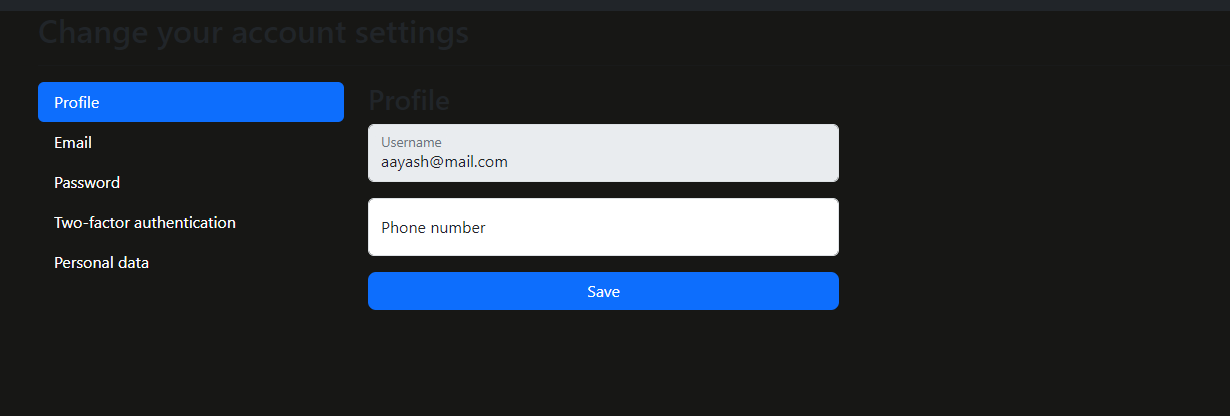
Но за да завърши поръчката си потребителят трябва да бъде логнат в системата в противен случай ще изведе грешка. И ще отведе потребителят до екран за влизане или регистрация.



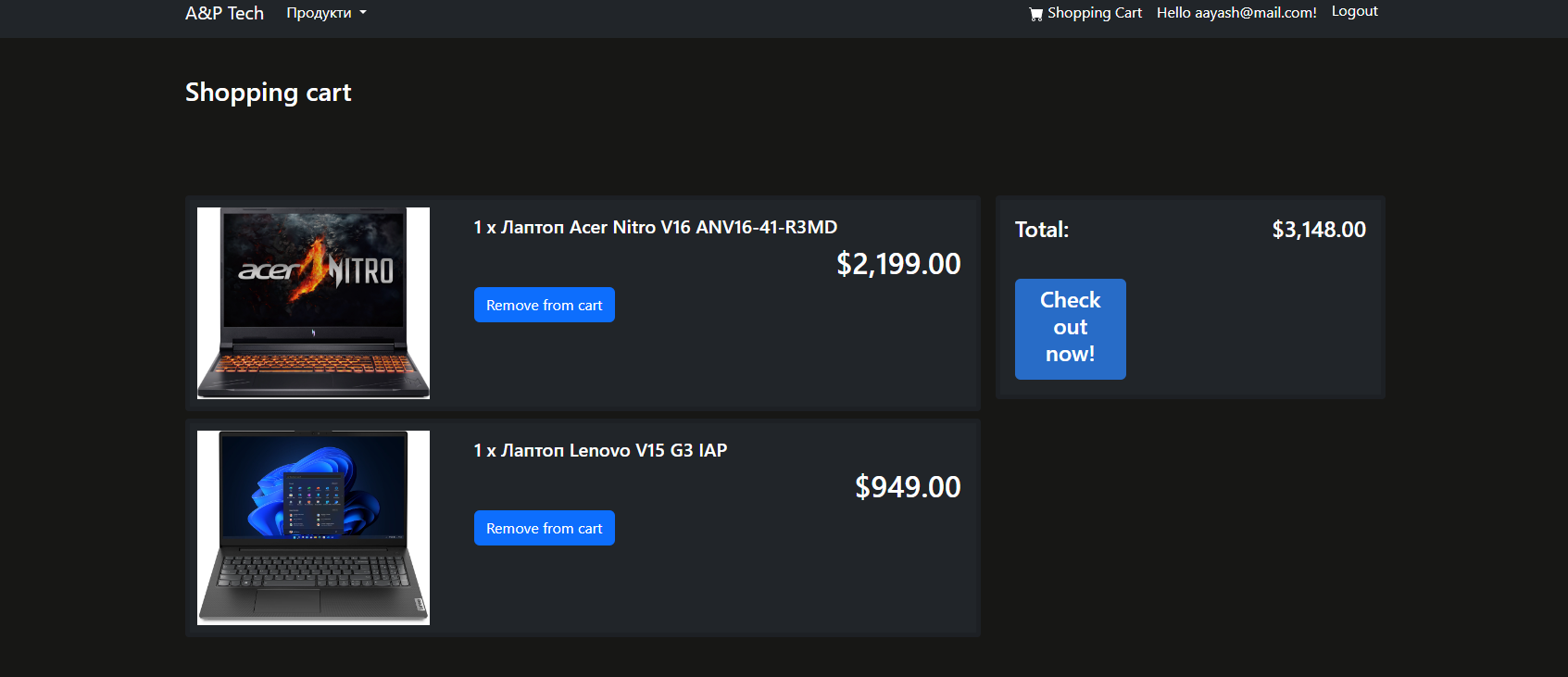
След успешно влизане или регистрация потребителят ще бъде отведен в началната страница



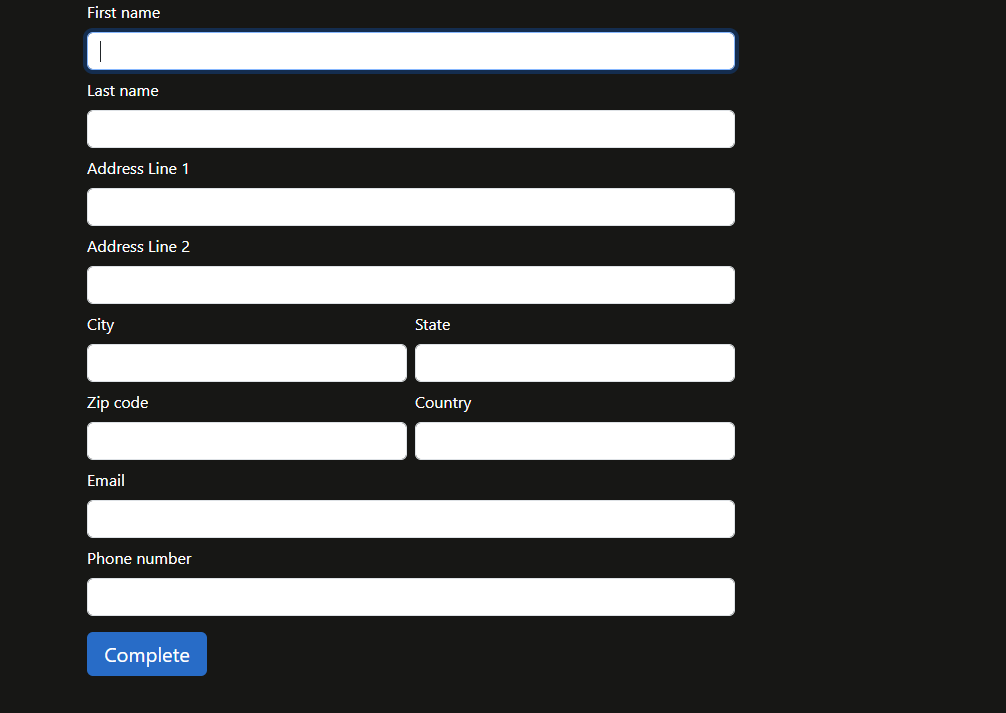
И потребителското му име ще бъде изведено на екран, както и допълнителен бутон за отписване от профила.

При натискане на профила на потребителят той ще бъде отведен в профилната си страница, където ще може да актуализира своите данни и да добави нови. 

След регистрацията или влизането на потребителя неговата клиентска сесия не се губи и продуктите в количката се запазват.



Потребителят вече има право да завърши поръчката си. При натискане на бутон Check out, потребителят ще трябва да попълни данни за доставка и контакти.



## 4.3 Програмна част

### 4.3.1 Sign Up Controller

public class SignUpController : Controller

{

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Register(string username, string password, string email, string address, string name)

{

if (IsUsernameAvailable(username))

{

return Content($"User registered: Username - {username}, Password - {password}, Email - {email}, Address - {address}, Name - {name}");

}

else

{

ModelState.AddModelError(string.Empty, "Username is already taken.");

return View("Index");

}

}

private bool IsUsernameAvailable(string username)

{

return username != "admin";

}

}

}

Това е контролера за регистрация. Потребителят трябва да попълни дадените полета, за да може да прави поръчки от сайта, като всяко поле трябва да е попълнено, за да е регистрацията успешна. При въвеждане на заето потребителско име излиза грешка и потребителят трябва да въведе ново валидно такова. Има метод, който проверява дали потребителското име е свободно, като „admin“ винаги е заето.

### 4.3.2 Login Page Controller

public class LoginPageController : Controller

{

public IActionResult LogIn()

{

return View();

}

[HttpPost]

public IActionResult LogIn(string username, string password)

{

if (IsValidUser(username, password))

{

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError(string.Empty, "Invalid username or password.");

return View();

}

}

private bool IsValidUser(string username, string password)

{

return (username == "admin" && password == "password");

}

}

}

Контролера за влизане в сайта. След като потребителя се регистрира , трябва да потвърди своите име и парола, за да влезе успешно в акаунта си. Отново се проверява дали потребителското име и паролата са валидни. Ако са валидни потребителя е пренасочен към началната страница, а ако са невалидни се връща към формуляра за влизане с показана грешка.

### 4.3.3 Shopping Cart Controller

public class ShoppingCartController : Controller

{

private readonly IProductRepository productRepository;

private readonly IShoppingCart shoppingCart;

public ShoppingCartController(IProductRepository productRepository, IShoppingCart shoppingCart)

{

this.productRepository = productRepository;

this.shoppingCart = shoppingCart;

}

public ViewResult Index()

{

var items = shoppingCart.GetShoppingCartItems();

shoppingCart.ShoppingCartItems = items;

var shoppingCartViewModel = new ShoppingCartViewModel(shoppingCart, shoppingCart.GetShoppingCartTotal());

return View(shoppingCartViewModel);

}

public RedirectToActionResult AddToShoppingCart(int prodcutId)

{

var selectedPie = productRepository.GetAllProduct().FirstOrDefault(p => p.ID == prodcutId);

if (selectedPie != null)

{

shoppingCart.AddToCart(selectedPie);

}

return RedirectToAction("Index");

}

public RedirectToActionResult RemoveFromShoppingCart(int id)

{

var selectedProduct = productRepository.GetAllProduct().FirstOrDefault(p => p.ID == id);

if (selectedProduct != null)

{

shoppingCart.RemoveFromCart(selectedProduct);

}

return RedirectToAction("Index");

}

}

}

Този контролер обработва заявките с количката за пазаруване. Има метод, който връща изглед с данните за количката за пазаруване. Създава модел на изглед „ShoppingCartViewModel“ с текущата количка и общата сума на продуктите в нея. Метода „AddToShoppingCart“ извлича продукта от хранилището по даденото му „productid“, ако продукта е намерен, той се добавя в количката и потребителя се пренасочва към страницата с количката. Метода „RemoveFromShoppingCart“ извлича продукта от хранилището по даденото му „id“, ако продукта е намерен, той се премахва от количката и потребителя се пренасочва към страницата с количката.

### 4.3.4 Order Controller

public class OrderController : Controller

{

private readonly IOrderRepository \_orderRepository;

private readonly IShoppingCart \_shoppingCart;

public OrderController(IOrderRepository orderRepository, IShoppingCart shoppingCart)

{

\_orderRepository = orderRepository;

\_shoppingCart = shoppingCart;

}

public IActionResult Checkout()

{

return View();

}

[HttpPost]

public IActionResult Checkout(Order order)

{

var items = \_shoppingCart.GetShoppingCartItems();

\_shoppingCart.ShoppingCartItems = items;

if (\_shoppingCart.ShoppingCartItems.Count == 0)

{

ModelState.AddModelError("", "Your cart is empty, add some products first");

}

if (ModelState.IsValid)

{

\_orderRepository.CreateOrder(order);

\_shoppingCart.ClearCart();

return RedirectToAction("CheckoutComplete");

}

return View(order);

}

public IActionResult CheckoutComplete()

{

ViewBag.CheckoutCompleteMessage = "Thanks for your order. You'll soon enjoy your new tech!";

return View();

}

}

}

Това е контролер, който обработва заявките, свързани с поръчките. Конструкторът приема два параметъра, два интерфейса, като единия е за достъп до хранилището за поръчките, а другия е за работа с количката за пазаруване. Има друг метод, който проверява дали количката е празна, ако е празна, излиза грешка. Ако всичко е валидно се създава поръчката, количката се изчиства и потребителят е пренасочен към „CheckoutComplete“, който връща изглед със съобщение за успешно завършена поръчка.

### 4.3.5 Order Class

public class Order

{

[BindNever]

public int OrderId { get; set; }

public List<OrderDetail>? OrderDetails { get; set; }

[Required(ErrorMessage = "Please enter your first name")]

[Display(Name = "First name")]

[StringLength(50)]

public string FirstName { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter your last name")]

[Display(Name = "Last name")]

[StringLength(50)]

public string LastName { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter your address")]

[StringLength(100)]

[Display(Name = "Address Line 1")]

public string AddressLine1 { get; set; } = string.Empty;

[Display(Name = "Address Line 2")]

public string? AddressLine2 { get; set; }

[Required(ErrorMessage = "Please enter your zip code")]

[Display(Name = "Zip code")]

[StringLength(10, MinimumLength = 4)]

public string ZipCode { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter your city")]

[StringLength(50)]

public string City { get; set; } = string.Empty;

[StringLength(10)]

public string? State { get; set; }

[Required(ErrorMessage = "Please enter your country")]

[StringLength(50)]

public string Country { get; set; } = string.Empty;

[Required(ErrorMessage = "Please enter your phone number")]

[StringLength(25)]

[DataType(DataType.PhoneNumber)]

[Display(Name = "Phone number")]

public string PhoneNumber { get; set; } = string.Empty;

[Required]

[StringLength(50)]

[DataType(DataType.EmailAddress)]

[RegularExpression(@"(?:[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*|""(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])\*"")@(?:(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]\*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])",

ErrorMessage = "The email address is not entered in a correct format")]

public string Email { get; set; } = string.Empty;

[BindNever]

public decimal OrderTotal { get; set; }

[BindNever]

public DateTime OrderPlaced { get; set; }

}

}

Този клас съдържа информация за поръчка, включително данни за клиента и детайли за поръчаните продукти. Полетата са валидирани с помощта на анотации за данни, които указват задължителността, максималната дължина и други правила за валидиране.

### 4.3.6 DbInitializer Class

public class DbInitializer

{

//-------------------------

//Constants:

//-------------------------

//-------------------------

//Members:

//-------------------------

//-------------------------

//Properties:

//-------------------------

//-------------------------

//Constructor/Destructor:

//-------------------------

//-------------------------

//Methods:

//-------------------------

public static void Seed(IApplicationBuilder applicationBuilder)

{

TechStoreDbContext context = applicationBuilder.ApplicationServices.CreateScope().ServiceProvider.GetRequiredService<TechStoreDbContext>();

if (!context.Categories.Any())

{

context.AddRange

(

new Category() { Name = Messages.MSG\_COMPUTER\_CATEGORY },

new Category() { Name = "Телефони" },

new Category() { Name = "Процесори" },

new Category() { Name = "ВидеоКарти" }

);

}

if (!context.Categories.Any())

{

context.AddRange

(

new Product() { CategoryID = 1,Name= "Лаптоп Acer Nitro V16 ANV16-41-R3MD", ImageUrl = "https://ardes.bg/uploads/original/acer-nitro-v16-anv16-41-529298.jpg", Price=2199d },

new Product() { CategoryID = 1, Name = "Лаптоп Lenovo V15 G3 IAP", ImageUrl = "https://ardes.bg/uploads/original/lenovo-v15-g3-432234.jpg", Price = 949d },

new Product() { CategoryID = 1, Name = "Лаптоп HP ZBook Fury 15 G8", ImageUrl = "https://ardes.bg/uploads/original/hp-zbook-fury-15-g7-300512.jpg", Price = 2499d }

);

}

context.SaveChanges();

}

//-------------------------

//Overrides:

//-------------------------

}

}

Този клас съдържа статичен метод за зареждане (seed) на базата данни с начални данни. Това е полезно, когато създаваме нова база данни или при инициализиране на приложението. Методът „seed“ добавя категории и продукти, ако те все още не съществуват в базата данни. Този подход осигурява наличие на основни данни, които могат да бъдат използвани за тестови или демонстрационни цели.