

LAB 1

DEVELOP ASP.NET CORE WEB APP (P1)

❖ CONTENT

- Create an ASP.NET Core MVC project
- Setup database connection & implement Authentication (login/logout, register)
- Create tables with Model
- Implement CRUD features with Scaffolding technique
- Customize web layout (template) & identity pages
- Set default role for user registration
- Seed (populate) initial data to database
- Setup Authorization (role-based access) for different features

❖ INSTRUCTION

1. Open Visual Studio and create new ASP.NET Core MVC web app

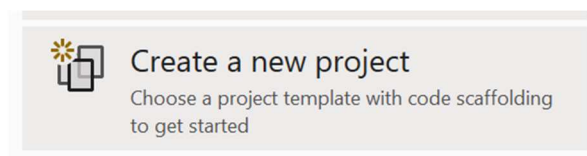


Figure 1 - Create new project in Visual Studio

2. Config ASP.NET Core project

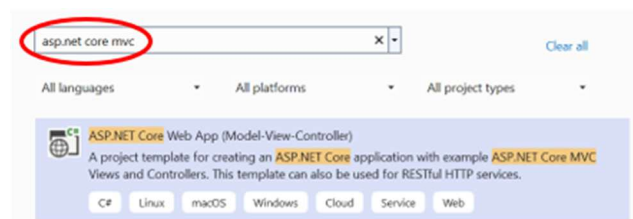


Figure 2 - Search & select ASP.NET Core Web App (MVC)

Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows

Project name
lab1

Location
D:\workspace\Teaching\netcore-course\codes\web\

Solution name ⓘ
web

☐ Place solution and project in the same directory

Figure 3 – Set project location, solution & project name

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS

Framework ⓘ
.NET 8.0 (Long Term Support)

Authentication type ⓘ
Individual Accounts

☒ Configure for HTTPS ⓘ

☐ Enable Docker ⓘ

Docker OS ⓘ
Linux

☐ Do not use top-level statements ⓘ

Figure 4 – Setup automatic database connection & Authentication

3. Add model to automatically create table

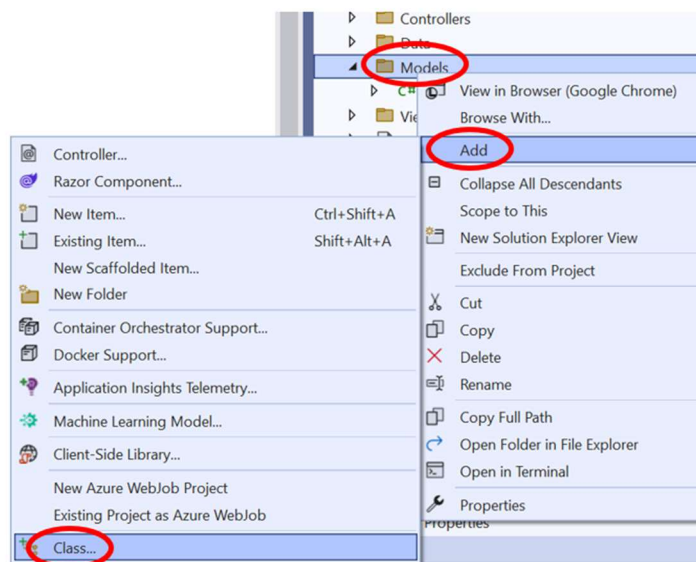


Figure 5 - Add new Model (class)

```
public class Laptop
{
    public int Id { get; set; }
    public string Brand { get; set; }
    public string Model { get; set; }
    public int Quantity { get; set; }
    public decimal Price { get; set; }
    public string Image { get; set; }
}
```

Figure 6 – Models/Laptop.cs

4. Generate Controller with Views for CRUD features with Scaffolding technique

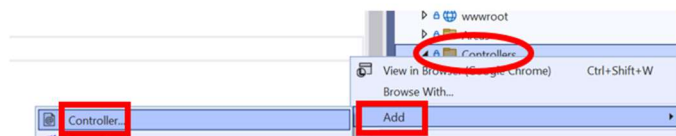


Figure 7 - Add new controller (class)

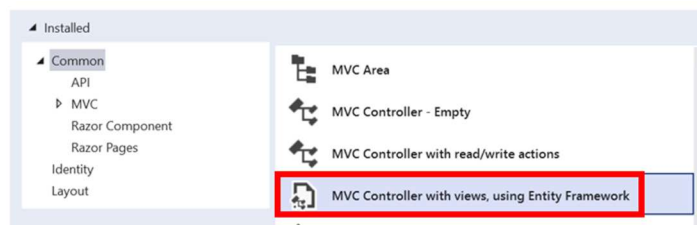


Figure 8 - Select MVC Controller with views

Add MVC Controller with views, using Entity Framework

Model class: Laptop (lab1.Models)

DbContext class: ApplicationDbContext (lab1.Data)

Database provider: Configured from the selected DbContext

Views

☒ Generate views

☒ Reference script libraries

☒ Use a layout page

Controller name: LaptopsController

Add Cancel

Figure 9 - Select Model class & DbContext class

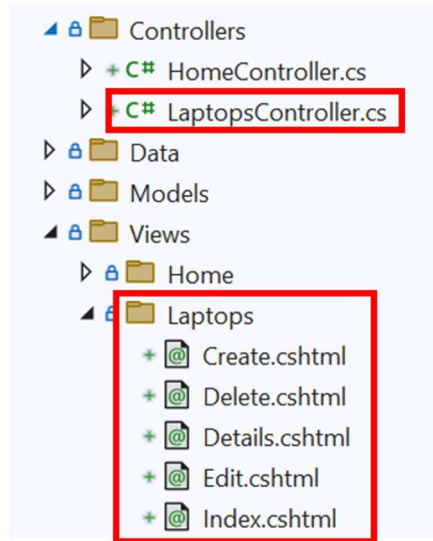


Figure 10 - Controller & Views have been generated for selected Model

5. Override Identity pages to customize view of Authentication (Login, Register,...)

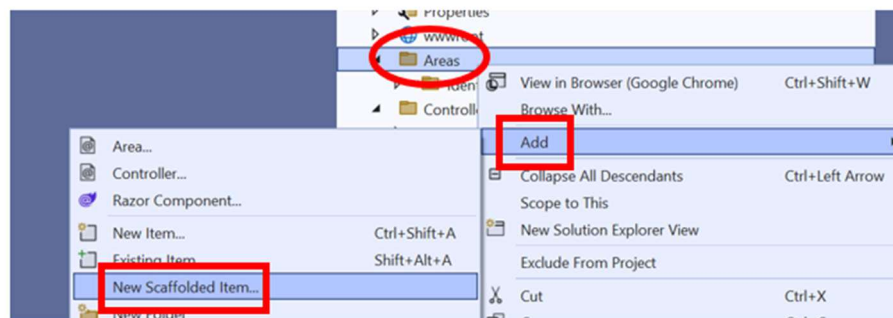


Figure 11 - Add new Scaffolded Item

Add New Scaffolded Item

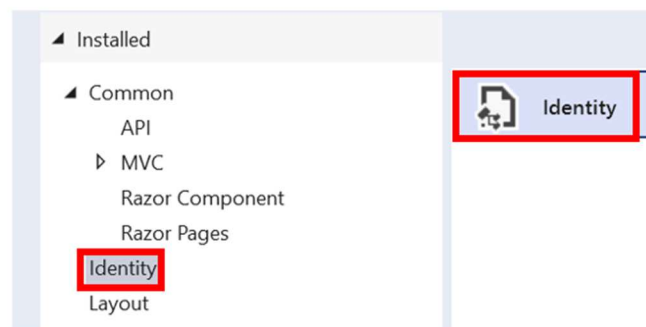


Figure 12 - Select option **Identity**

Add Identity

Select an existing layout page, or specify a new one:
/Areas/Identity/Pages/Account/Manage/_Layout.cshtml
(If you specify a file, it is not in a Razor _viewstart file)

☒ Override all files

Choose files to override

<input checked="" type="checkbox"/> Account/StatusMessage	<input checked="" type="checkbox"/> Account/AccessDenied	<input checked="" type="checkbox"/> Account/ConfirmEmail
<input checked="" type="checkbox"/> Account/ConfirmEmailChange	<input checked="" type="checkbox"/> Account/ExternalLogin	<input checked="" type="checkbox"/> Account/ForgotPassword
<input checked="" type="checkbox"/> Account/ForgotPasswordConfirmation	<input checked="" type="checkbox"/> Account/Lockout	<input checked="" type="checkbox"/> Account/Login
<input checked="" type="checkbox"/> Account/LoginWith2fa	<input checked="" type="checkbox"/> Account/LoginWithRecoveryCode	<input checked="" type="checkbox"/> Account/Logout
<input checked="" type="checkbox"/> Account/Manage/Layout	<input checked="" type="checkbox"/> Account/Manage/ManageNav	<input checked="" type="checkbox"/> Account/Manage/StatusMessage
<input checked="" type="checkbox"/> Account/Manage/ChangePassword	<input checked="" type="checkbox"/> Account/Manage/DeletePersonalData	<input checked="" type="checkbox"/> Account/Manage/Disable2fa
<input checked="" type="checkbox"/> Account/Manage/DownloadPersonalData	<input checked="" type="checkbox"/> Account/Manage/Email	<input checked="" type="checkbox"/> Account/Manage/EnableAuthenticator
<input checked="" type="checkbox"/> Account/Manage/ExternalLogins	<input checked="" type="checkbox"/> Account/Manage/GenerateRecoveryCodes	<input checked="" type="checkbox"/> Account/Manage/Index
<input checked="" type="checkbox"/> Account/Manage/PersonalData	<input checked="" type="checkbox"/> Account/Manage/ResetAuthenticator	<input checked="" type="checkbox"/> Account/Manage/SetPassword
<input checked="" type="checkbox"/> Account/Manage/ShowRecoveryCodes	<input checked="" type="checkbox"/> Account/Manage/TwoFactorAuthentication	<input checked="" type="checkbox"/> Account/Register
<input checked="" type="checkbox"/> Account/RegisterConfirmation	<input checked="" type="checkbox"/> Account/ResendEmailConfirmation	<input checked="" type="checkbox"/> Account/ResetPassword
<input checked="" type="checkbox"/> Account/ResetPasswordConfirmation		

DbContext class:

Database provider:

User class:

Figure 13 - Override all files & select DbContext class

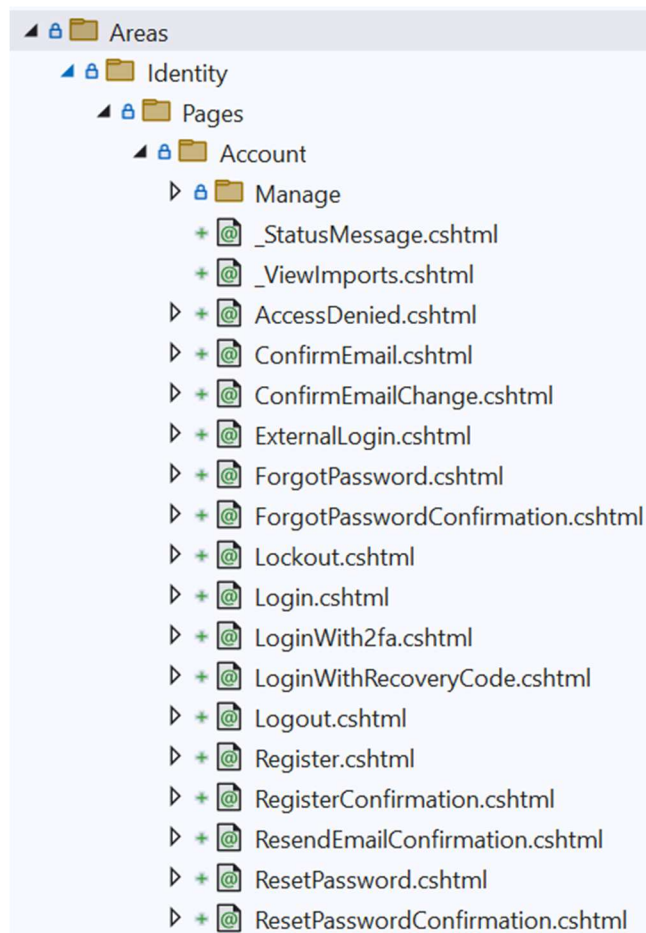


Figure 14 - All Identity pages were displayed to be overridden

6. Set default role for user registration (optional)

```
var callbackUrl = Url.Page(
    "/Account/ConfirmEmail",
    pageHandler: null,
    values: new { area = "Identity", userId = userId, code = code, returnUrl
    = returnUrl },
    protocol: Request.Scheme);
```

```
//Set default role "User" for new user registration
var role = "User";
await _userManager.AddToRoleAsync(user, role);
```

```
await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
    $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode
    (callbackUrl)}'>clicking here</a>.");
```

Figure 15 - Areas/Identity/Pages/Account/Register.cshtml.cs

7. Customize web layout & other web pages

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Laptops" asp-action="Index">Laptop</a>
    </li>
  </ul>
  <partial name="_LoginPartial" />
</div>
```

Figure 16 - Views/Shared/_Layout.cshtml

```
<div class="text-center">
  
</div>
```

Figure 17 - Views/Home/Index.cshtml

```
<td>
  @Html.DisplayFor(modelItem => item.Price)
</td>
<td>
  
</td>
```

Figure 18 - Views/Laptops/Index.cshtml

8. Seed initial data to database

```
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);

    //Seed data for User & Role
    SeedUserRole(builder);

    //Seed data for table Laptop
    SeedLaptop(builder);
}
```

Figure 19 - Data/ApplicationDbContext.cs (1)

```
private void SeedUserRole(ModelBuilder builder)
{
    //A) Setup IdentityUser
    //1. Create accounts
    var adminAccount = new IdentityUser
    {
        Id = "user1",
        UserName = "admin@gmail.com",
        Email = "admin@gmail.com",
        NormalizedUserName = "ADMIN@GMAIL.COM",
        NormalizedEmail = "ADMIN@GMAIL.COM",
        EmailConfirmed = true
    };

    var userAccount = new IdentityUser
    {
        Id = "user2",
        UserName = "user@gmail.com",
        Email = "user@gmail.com",
        NormalizedUserName = "USER@GMAIL.COM",
        NormalizedEmail = "USER@GMAIL.COM",
        EmailConfirmed = true
    };

    //2. Declare hasher for password encryption
    var hasher = new PasswordHasher<IdentityUser>();

    //3. Setup password for accounts
    adminAccount.PasswordHash = hasher.HashPassword(adminAccount, "123456");
    userAccount.PasswordHash = hasher.HashPassword(userAccount, "123456");

    //4. Add accounts to database
    builder.Entity<IdentityUser>().HasData(adminAccount, userAccount);
}
```

Figure 20 - Data/ApplicationDbContext.cs (2)


```

//B) Setup IdentityRole
builder.Entity<IdentityRole>().HasData(
    new IdentityRole
    {
        Id = "role1",
        Name = "Administrator",
        NormalizedName = "ADMINISTRATOR"
    },
    new IdentityRole
    {
        Id = "role2",
        Name = "User",
        NormalizedName = "USER"
    }
);

//C) Setup IdentityUserRole
builder.Entity<IdentityUserRole<string>>().HasData(
    new IdentityUserRole<string>
    {
        UserId = "user1",
        RoleId = "role1"
    },
    new IdentityUserRole<string>
    {
        UserId = "user2",
        RoleId = "role2"
    }
);
}

```

Figure 21 - *Data/ApplicationDbContext.cs (3)*

```

private void SeedLaptop(ModelBuilder builder)
{
    builder.Entity<Laptop>().HasData(
        new Laptop
        {
            Id = 1,
            Brand = "Apple",
            Model = "Macbook Pro M2",
            Quantity = 10,
            Price = 2345,
            Image = "https://bizweb.dktcdn.net/100/444/581/products/macbook-m1-vs-intel-1536x1268-6c00654d-ad87-4caf-8b88-aa6c34048199.png?v=1656134590567"
        },
        new Laptop
        {
            Id = 2,
            Brand = "Dell",
            Model = "XPS 15",
            Quantity = 15,
            Price = 1999,
            Image = "https://thegioiso365.vn/wp-content/uploads/2023/04/Dell-xps-9530-3.png"
        },
        new Laptop
        {
            Id = 3,
            Brand = "LG",
            Model = "Gram 17",
            Quantity = 22,
            Price = 2024,
            Image = "https://product.hstatic.net/1000333506/product/pc-gram-17z90q-b-gallery-02_dd780c6249ec430b84f82ed466fffd6e.jpg"
        }
    );
}

```

Figure 22 - *Data/ApplicationDbContext.cs (4)*


```
builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>();
builder.Services.AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.AddControllersWithViews();
```

Figure 23 - Program.cs

9. Setup role-based access for different features

- Restrict user access in backend (controller)

```
//Setup role for the whole controller
[Authorize(Roles = "Administrator")]
public class LaptopsController : Controller
```

Figure 24 - Controllers/LaptopsController.cs (1)

```
//Setup multiple roles for an action
[Authorize(Roles = "Administrator, User")]
public async Task<IActionResult> Index()
```

Figure 25 - Controllers/LaptopsController.cs (2)

```
//Setup single role for an action
[Authorize(Roles = "Administrator")]
public IActionResult Create()
```

Figure 26 - Controllers/LaptopsController.cs (3)

- Restrict user access in frontend (view)

```
<p>
    @if (User.Identity.IsAuthenticated && User.IsInRole("Administrator"))
    {
        <a asp-action="Create">Create New</a>
    }
</p>
```

Figure 27 - Views/Laptops/Index.cshtml

10.Migrate data to database

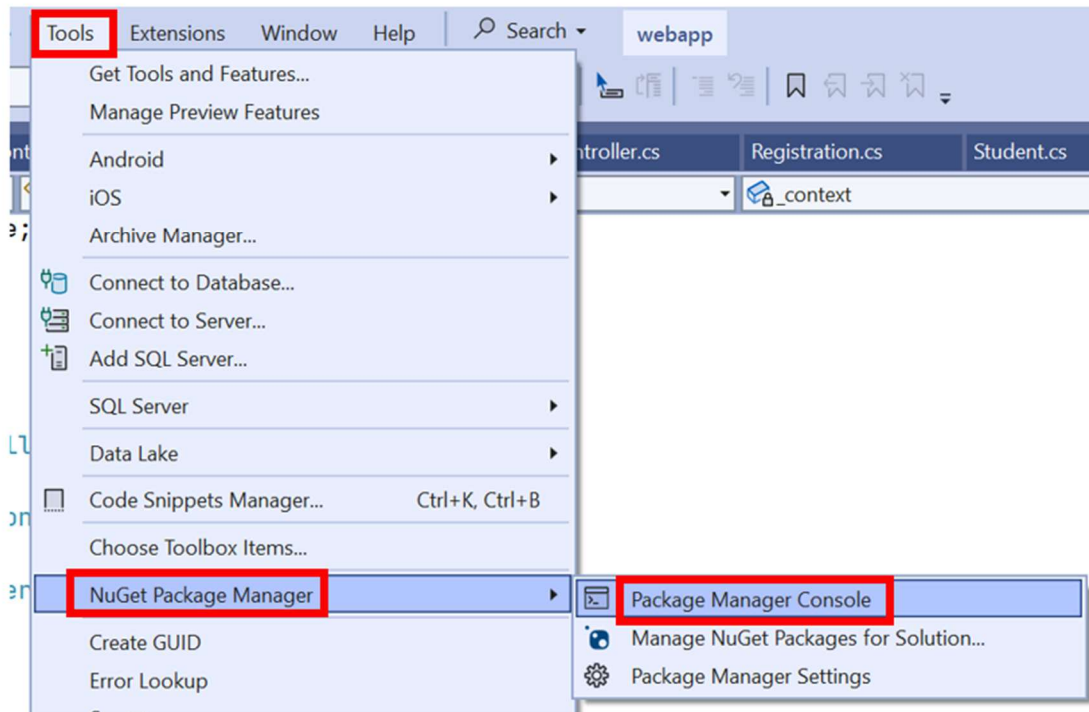


Figure 28 - Open *Package Manager Console (PMC)*

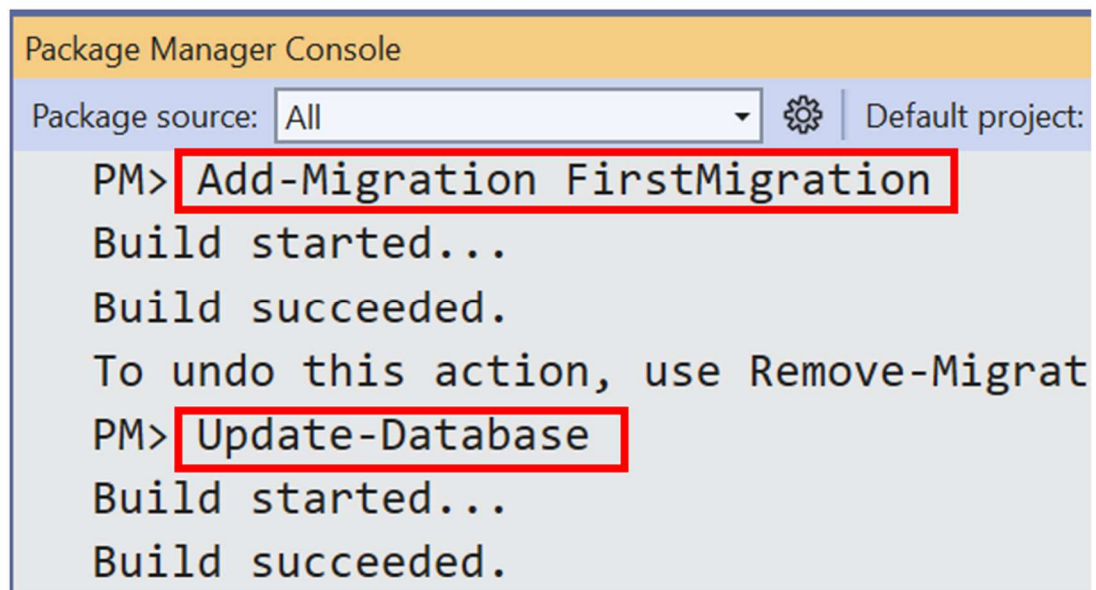


Figure 29 - Add migration & update database using PMC

11.Sample final result

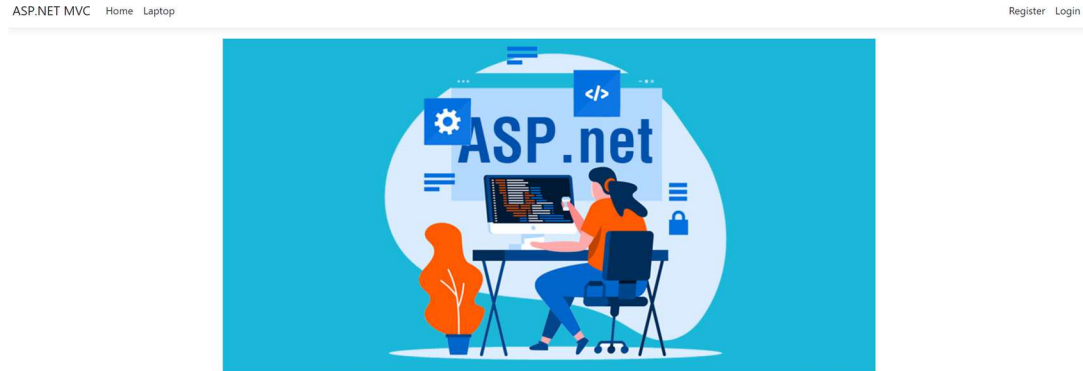


Figure 30 - Homepage

Log in

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

Figure 31 - Login page

Register

Register

Figure 32 - Register page

Index

[Create New](#)




Brand	Model	Quantity	Price	Image	
Apple	Macbook Pro M2	10	2345.00		Edit Details Delete
Dell	XPS 15	15	1999.00		Edit Details Delete
LG	Gram 17	22	2024.00		Edit Details Delete

Figure 33 - View laptop list

Create

Laptop

Brand

Model

Quantity

Price

Image

Create

[Back to List](#)

Figure 34 - Create new laptop

Edit

Laptop

Brand

Model

Quantity

Price

Image

Save

[Back to List](#)

Figure 35 - Edit existing laptop

Delete

Are you sure you want to delete this?

Laptop

Brand	Dell
Model	XPS 15
Quantity	15
Price	1999.00
Image	https://thegioiso365.vn/wp-content/uploads/2023/04/Dell-xps-9530-3.png

[Delete](#) | [Back to List](#)

Figure 36 - Delete existing laptop

Access denied

You do not have access to this resource.

Figure 37 - Access denied page