PHASE 3: Create an ASP.NET MVC Ecommerce Site to Sell Laptops

DESCRIPTION

Create an ASP.NET MVC web application to sell laptops online.

Background of the problem statement:

An existing retail company, Digital Retailers Inc, wants to take their business online and create an e-commerce portal that will enable people to purchase laptops online. They like to prototype new projects quickly using Agile methods and hence want to avoid a long requirement capture stage. They want to keep track of their codebase using Git so that multiple developers can work on it as required.

You must use the following:

- Visual Studio ASP.NET MVC web Project
- SQL Server 17 Express Edition or later
- Check in the project source into GitHub using Git tools.

Application Requirements:

- The pages must be done using Bootstrap and must be mobile-responsive.
- All data must be stored and retrieved using Data Access Layer and/or EntityFramework.

Following requirements should be met:

Some of the source code should be tracked on GitHub repositories. You need to document the tracked files that are ignored during the final push to the GitHub repository.

The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository in the document.

The step-by-step process involved in completing this task should be documented.

SUBMITTED BY:

Preetisha Srivastava

Software Engineer

Dover India

DATE OF SUBMISSION:

2nd October 2021

PROJECT OUTLINE

This is an ASP.NET MVC web application to sell laptops online. It has a Login Page for Admin, Seller and Users. The app displays all the items to the User and provides them an option to buy a particular laptop of their choice. The user can pay using a card. Backend should consist of Admin privileges to insert, update, and delete items from the page. Then finally we have a checkout page to verify the transaction and lead to a success page.

IMPLEMENTED FEATURES

1. Admin
    - Admin has privileges to do CRUD Operations on the database.
    - A Static UserName:**admin** and Password:**admin** is used in the project,
    - Admin can allocate sellers in the List created and sellers can login via the credentials given by the admin. Therefore, Admin has also the power to remove a seller from the list.
    - CRUD Operations are performed on the database, created on the local machine.
2. Seller
    - Seller can login with the credentials given by the Admin.
    - Seller can perform Create and Delete operations on the database and add his item into database
3. User
    - User is authorized and authenticated via SQL Database queries.
    - Gaurds are implemented.
    - User can view products only and checkout for the desired laptop.

PROBLEM LOGIC

- I have implemented the project using Visual Studio 2019 with ASP.net Entity Framework and MVC Patterns while using SQL DMS running on version 18.9.2.
- Used Identity Method to authorize and authenticate users via the sql db.

- Created Accounts Controller to manage with Admin, Seller and User Accounts and created respected Models and Views for them.
- Used Entity Framework Approach to automatically generate the code to perform CRUD Operations on the Database.
- Used razor Views to automatically generate views for Login, Register and many other creation of .cshtml files.
- Separate View page was created to make user page attractive using cards.
- Checkout and Success page views were generated.
- Respective dependencies were installed prior to the Code development.
- Have followed **Code-first Approach** in the submitted project and respective migrations were done followed by updating database.

EXECUTION

- After creating a new project in WebApp (Model, View and Controller) template, the Migration needs to be added. This is done by using a command "Add-migration <MigrationName>", by following Tools →NuGet Package Manager → Package Manager Console.
- Before that, Controllers must be created for this project.
- Then a DbContext.cs file is added.
- After finishing all the coding, open the Package manager Console and execute the command "update-database".
- After the build is successful, create the database in the MSSQL server, with the same name as in appsettings.json file and execute the code in visual studio using IIS Express.