

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | Milestone 3: Final Plan | | |  |
|  |  | | |  |
|  | |  |  | |
|  | | John Keen johnkeenishere@gmail.com |  | |
|  | | January 21, 2024—Project Name:Laptop Catalog—CST-451: Senior Project 1—Professor Brandon Bass—Document Revision Number: 0 |  | |

# Abstract

This document outlines the design and architecture of a laptop catalog web application. It includes design specifications for an e-commerce website. The web application will be capable of creating, reading, updating and deleting catalog information and making catalog information available by means of a REST API. It will also be capable of processing and retaining records of customer transactions used to complete a purchase of inventory in the laptop catalog. The web application has an MVC architecture.

|  |  |  |
| --- | --- | --- |
|  |  |  |

# Table Of Contents

[Cover Page i](#_Toc156555474)

[Abstract ii](#_Toc156555475)

[Table Of Contents iii](#_Toc156555476)

[Design Planning Summary 1](#_Toc156555477)

[Final Technical Requirements 2](#_Toc156555478)

[Overview of Design Concepts 3](#_Toc156555479)

[Detailed Solution Architecture 24](#_Toc156555480)

# Design Planning Summary

A laptop manufacturing company has the need to publish information about its products so that retailers can leverage the information in order to sell laptops by the manufacturer. My suggested solution is a laptop catalog that is updated using CRUD functionality in a laptop management section of a web application. The catalog will be available to retailers by means of a REST API. Since the creation of the original proposal for this web application, the scope of the project has been expanded to include the management of roles for users and an e-commerce section of the web application where customers can purchase laptops directly from the laptop manufacturer. My proposed solution for managing authentication and roles is to leverage the ASP.NET Core Identity Framework offered by Microsoft. A custom identity can be created and the registration process can be modified to include the fields required to create an account. A laptop catalog that is available by means of a REST API will ensure that retailers can update their inventory and that product information regarding the latest laptops offered by the retailer does not become stale. This is where the tail of the project is trimmed. This project has the potential to grow. Future versions of the project could include authentication for the retailer’s REST API so that the API could be used by the retailer to place orders. The web application could have a section of the site only available to people with a retailer role so that retailers can manually place orders by the pallet. While these ideas would provide a value to the laptop manufacturer, they are currently outside the scope of the project.

# Final Technical Requirements

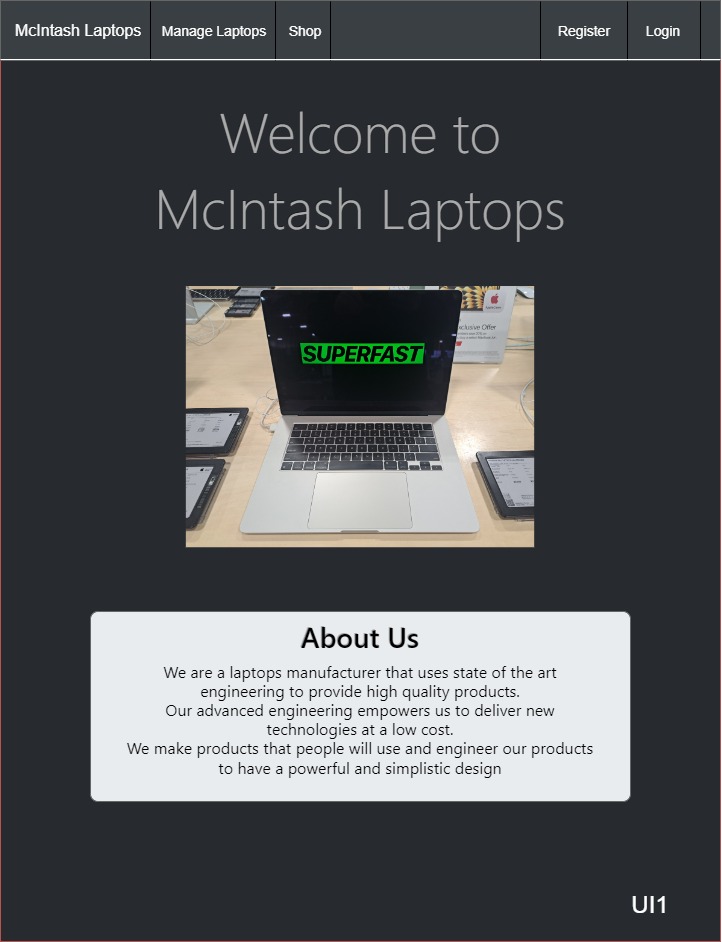
|  |
| --- |
| Final Technical Requirements |
| 1 – Architecture  The web application will need an architecture to help prevent the code from having a puddle of mud architecture. |
| 2 – Style  The web application will need a way to quickly and easily style the application. |
| 3 – Asynchronous Functions  The web application will need a way to asynchronously post data to a controller’s action and then asynchronously update part of the page with a partial view. This will help reduce the frequency of full-page refreshes. |
| 4 – Payment Gateway  The application will need a payment gateway so that payments can be processed and credit card information is not stored in the manufacturers database. |
| 5 – User Authentication  The web application will need a means of authentication. The password should be salted and encrypted. The system should use roles to minimize the web applications attack surface. |
| 6 – Encryption (At Rest)  The last four of the customers credit card number should be encrypted at rest. The web application should decrypt the information for the user if this data needs to be accessed. This will help prevent the data from being used for phishing in the event that the database security is compromised. |
| 7 – Encryption (In Transit)  When a customer provides their vital information in order to make a payment, the customer needs their information secured between the web application and the user. This will help prevent data from being exploited in the event of a man in the middle attack. |

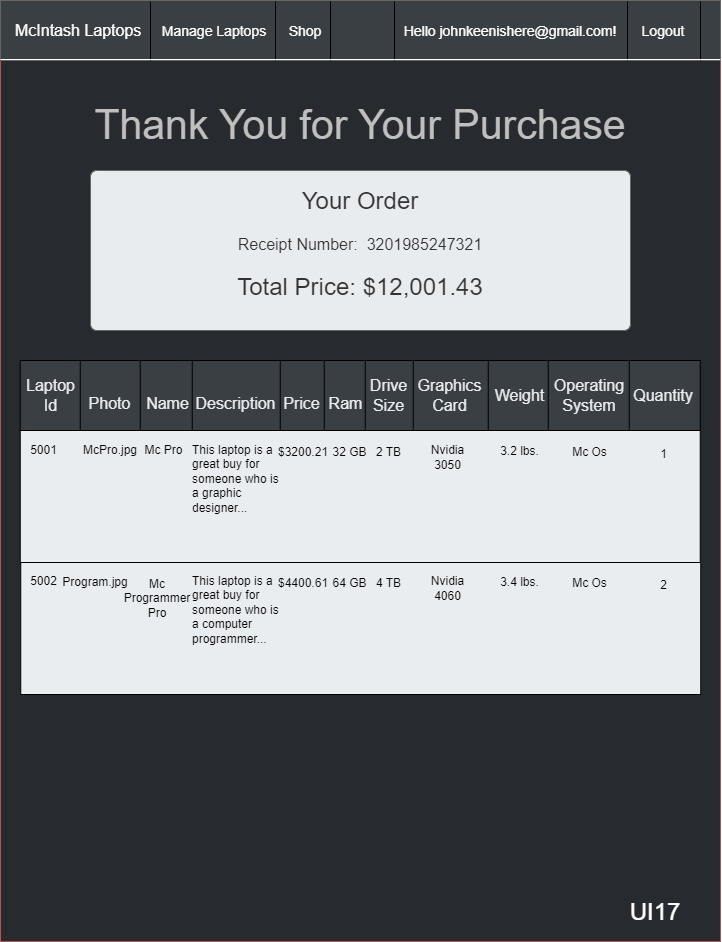
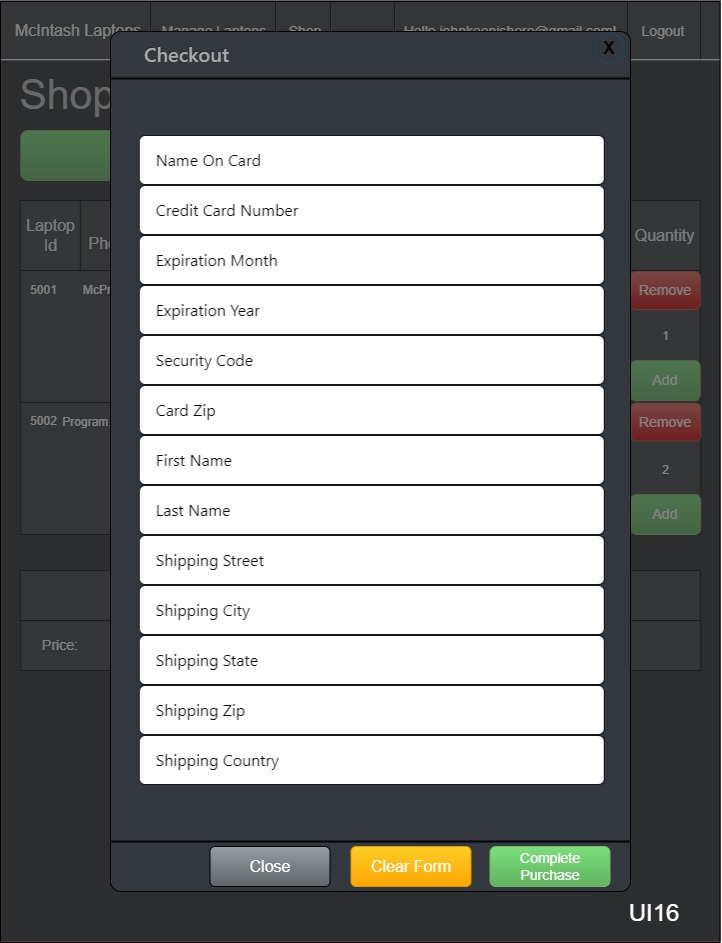
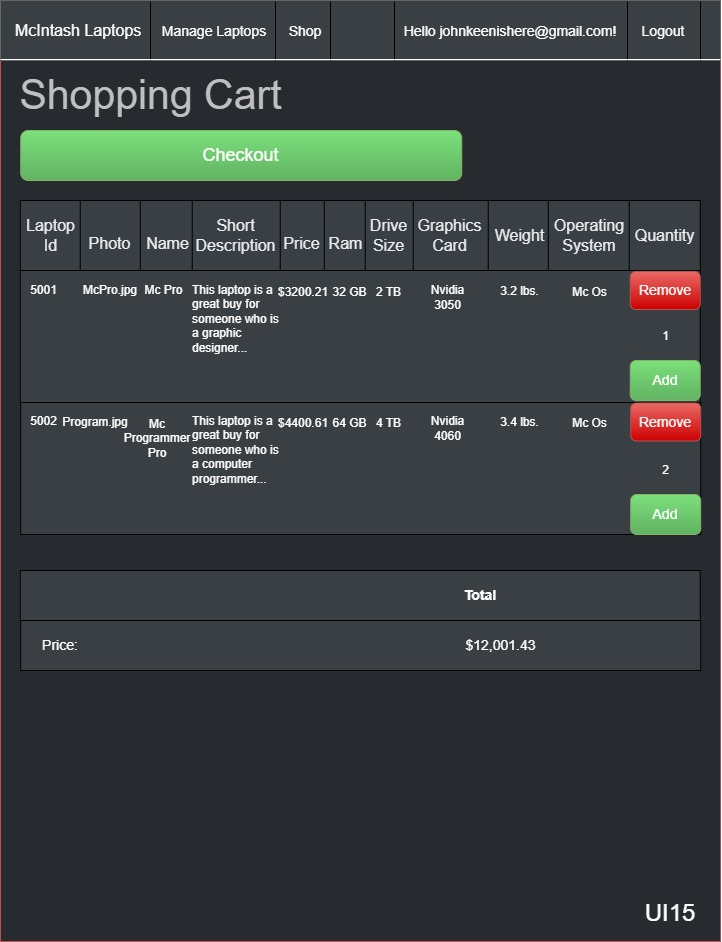
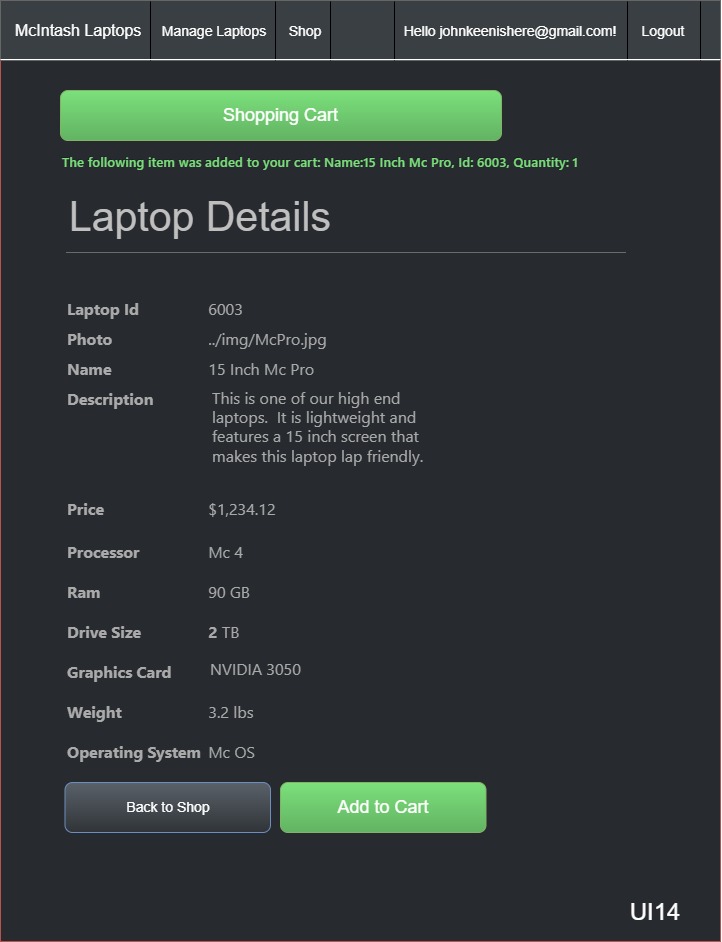
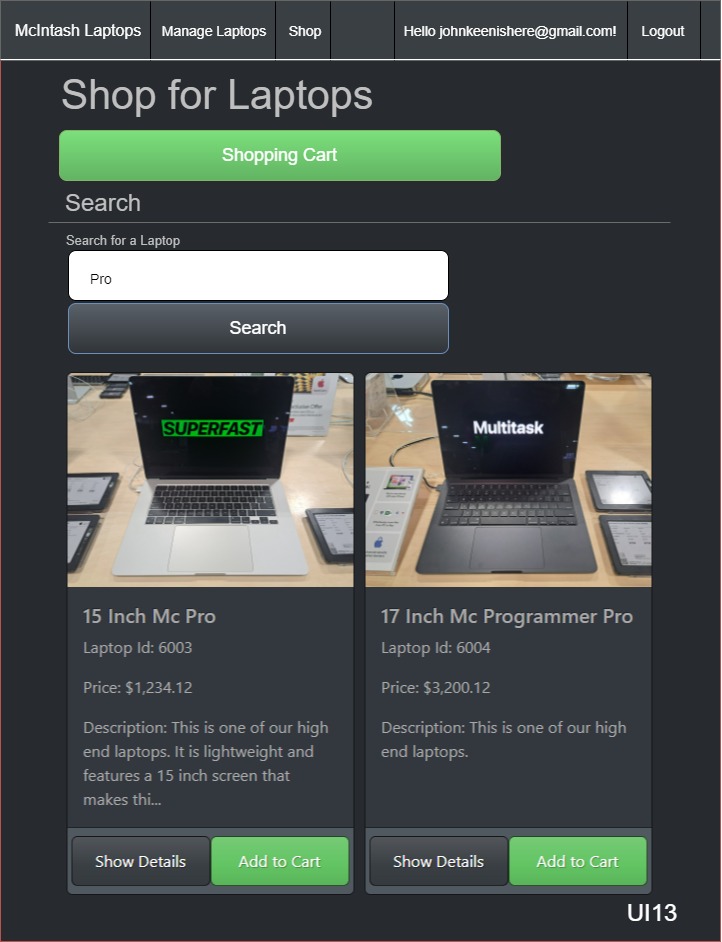
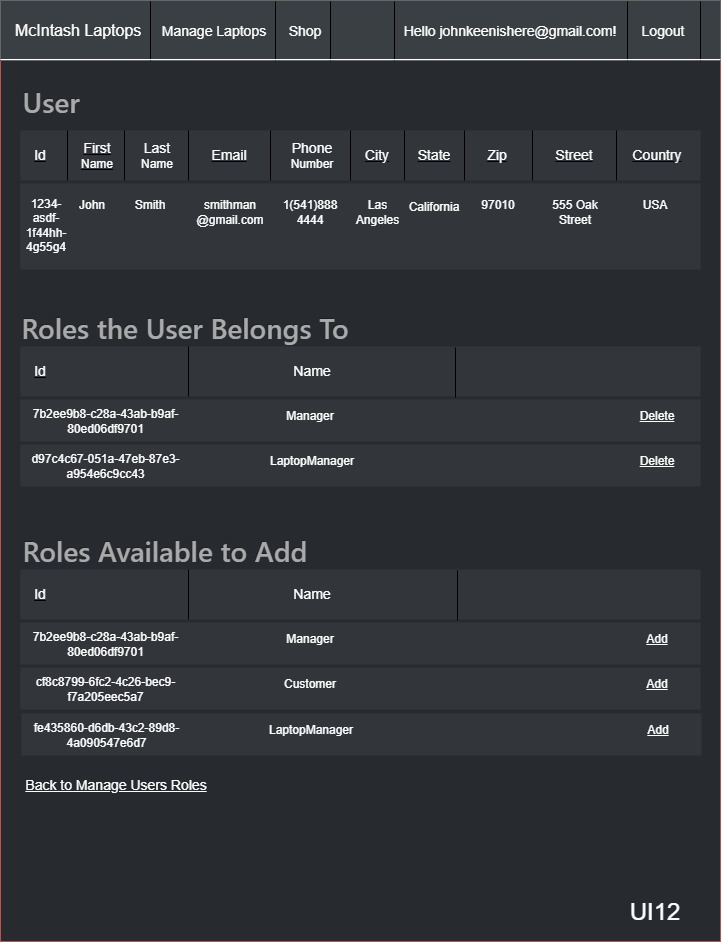
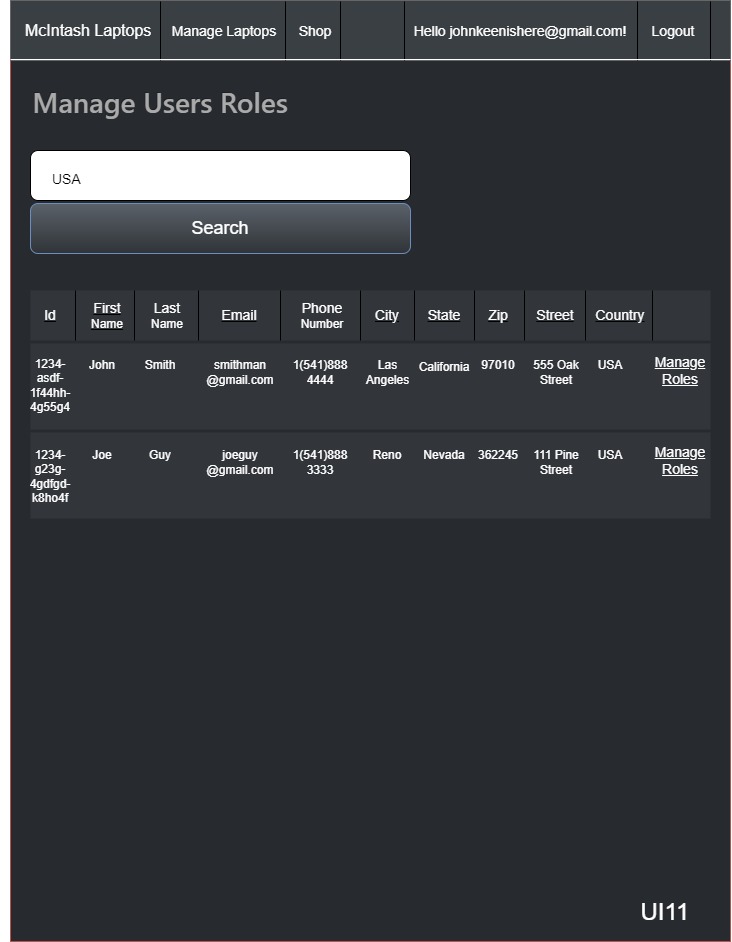
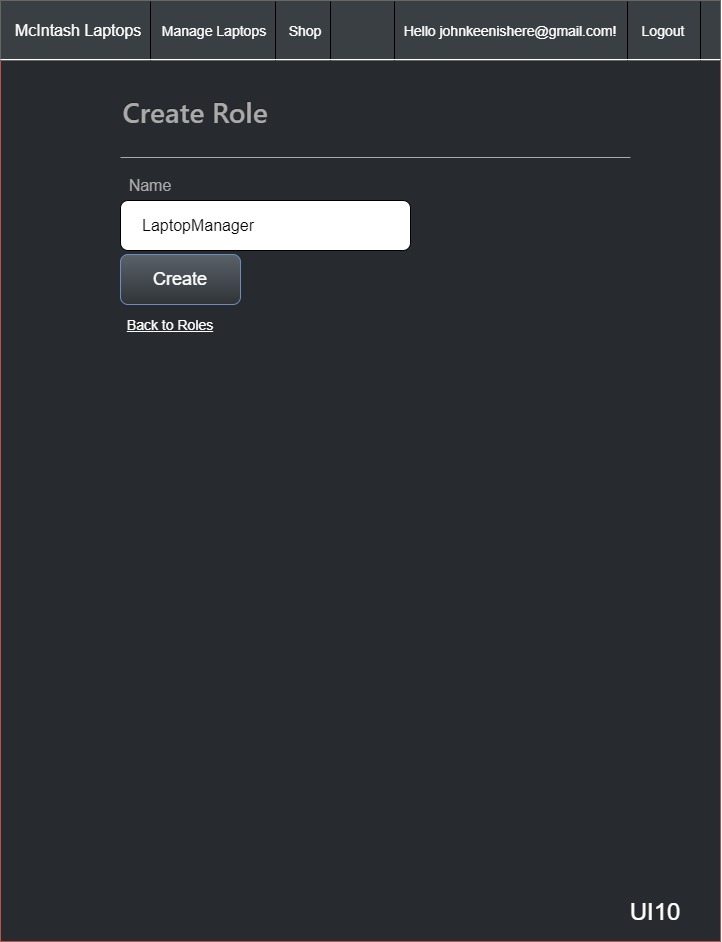
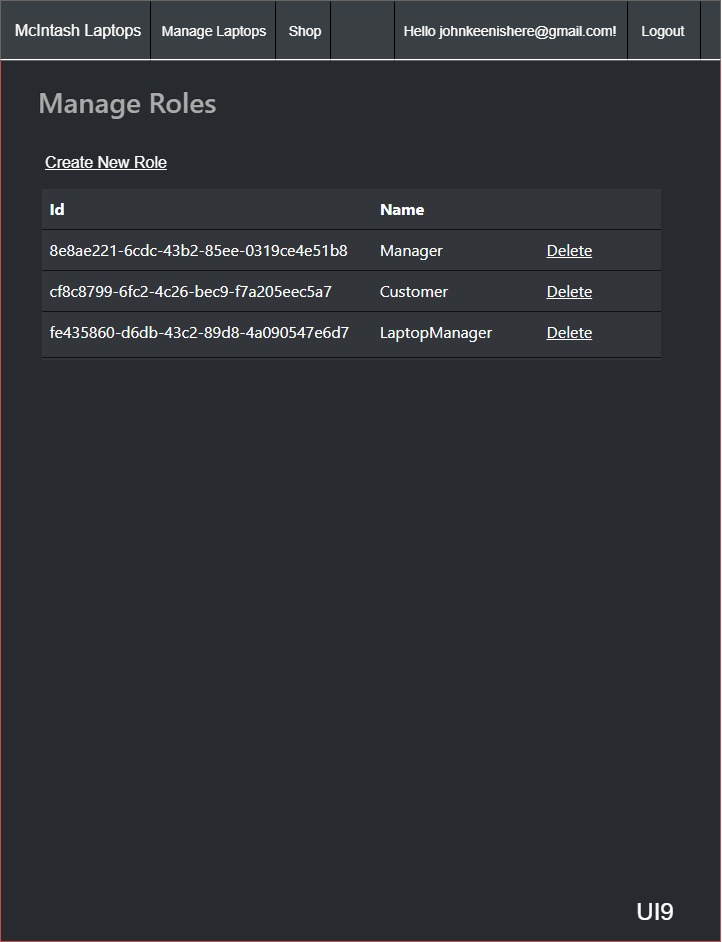
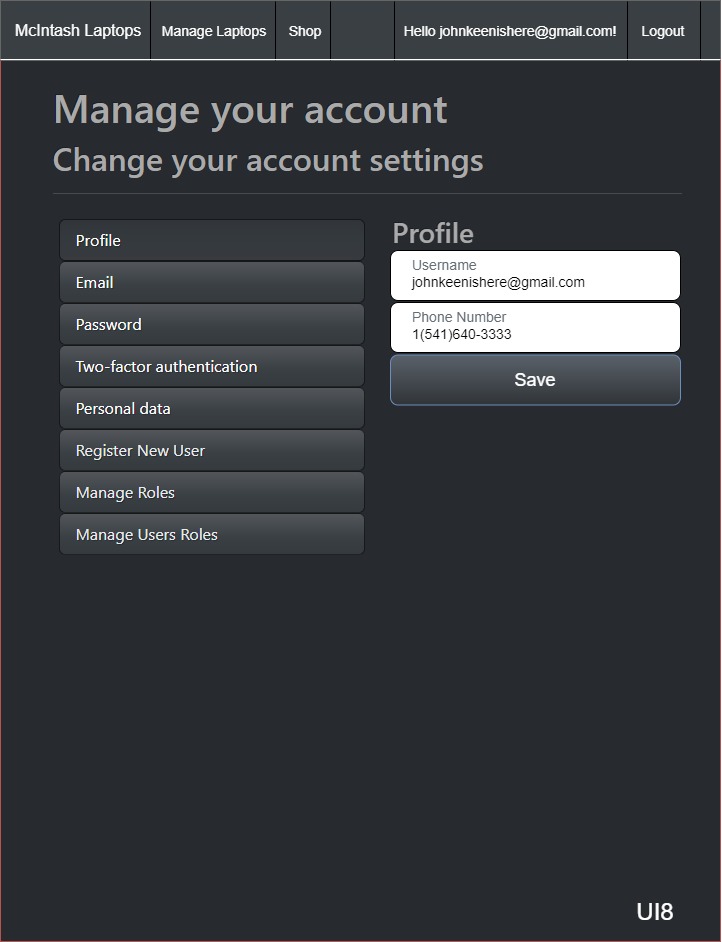
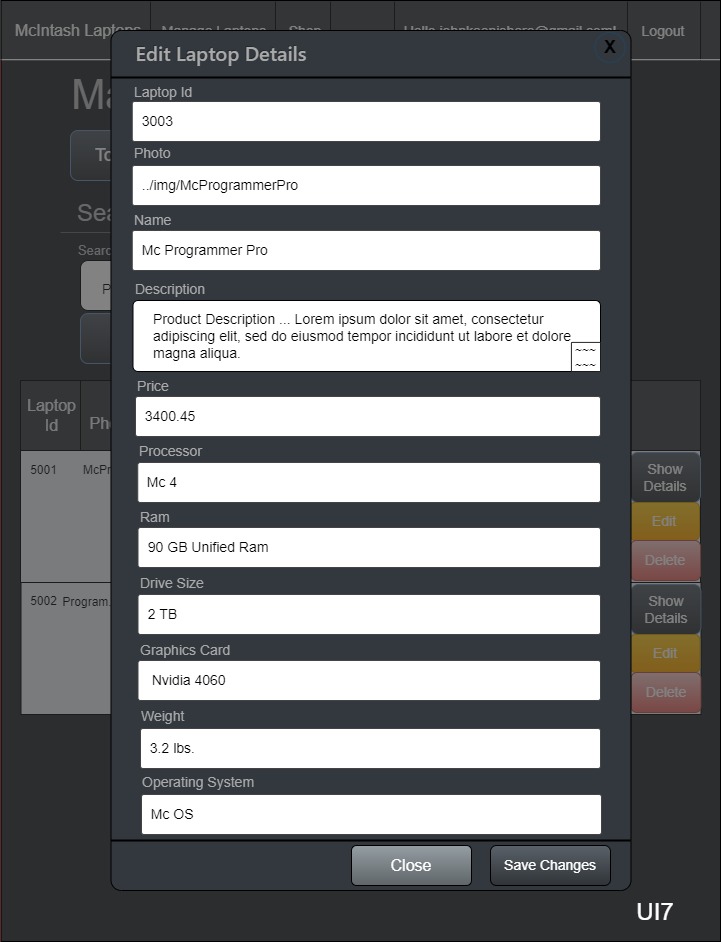
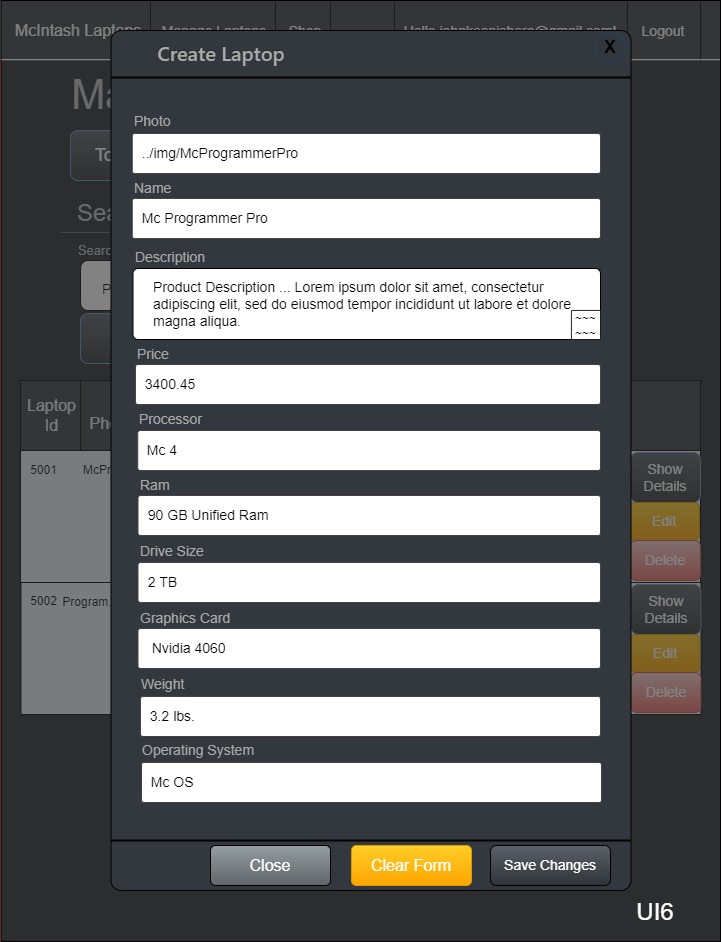
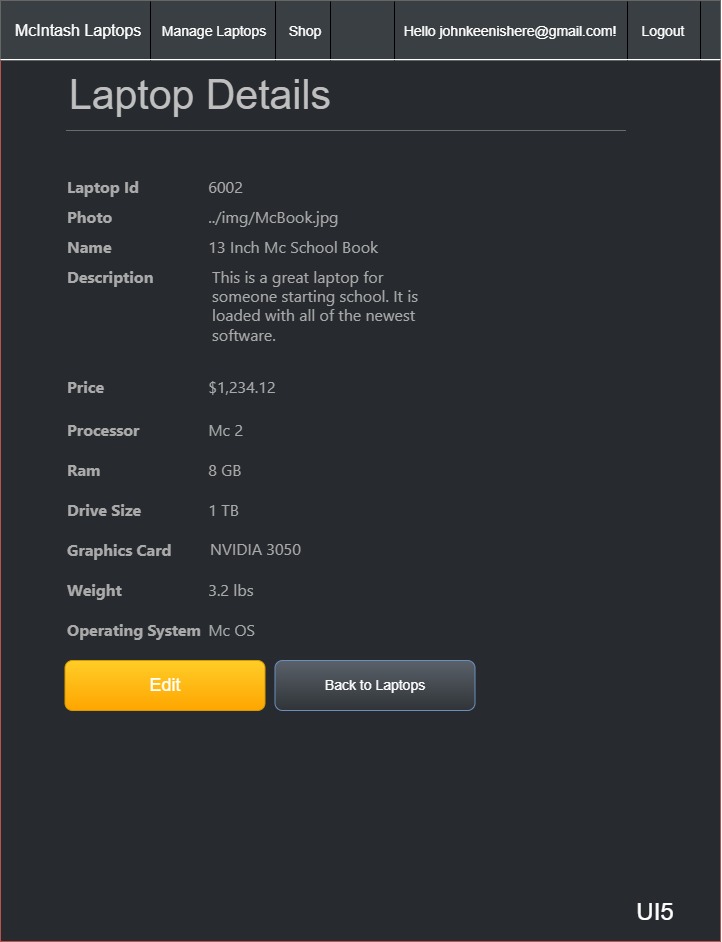
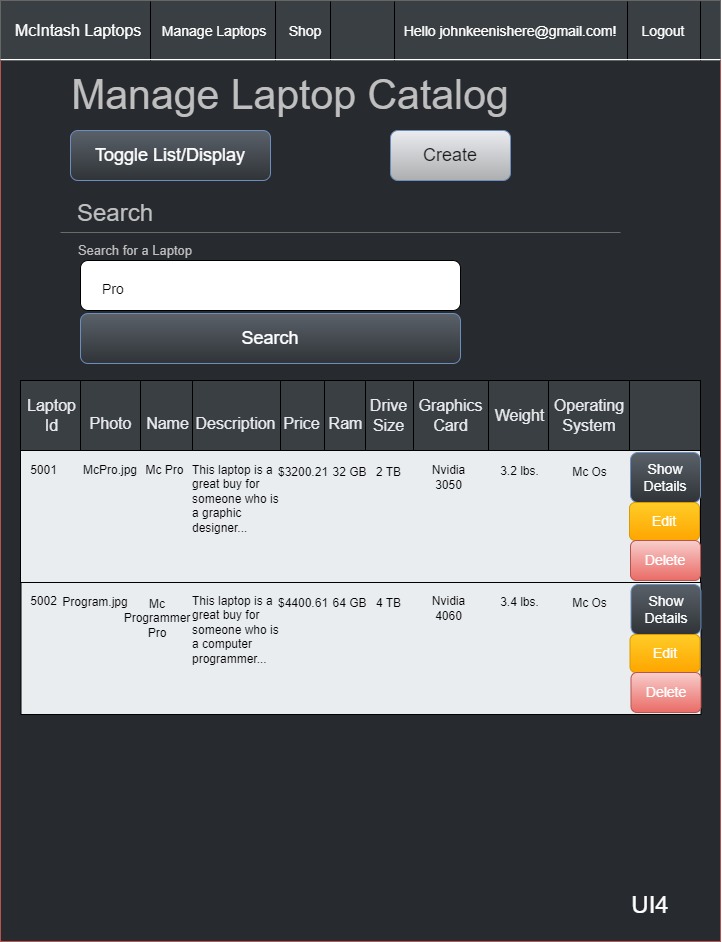
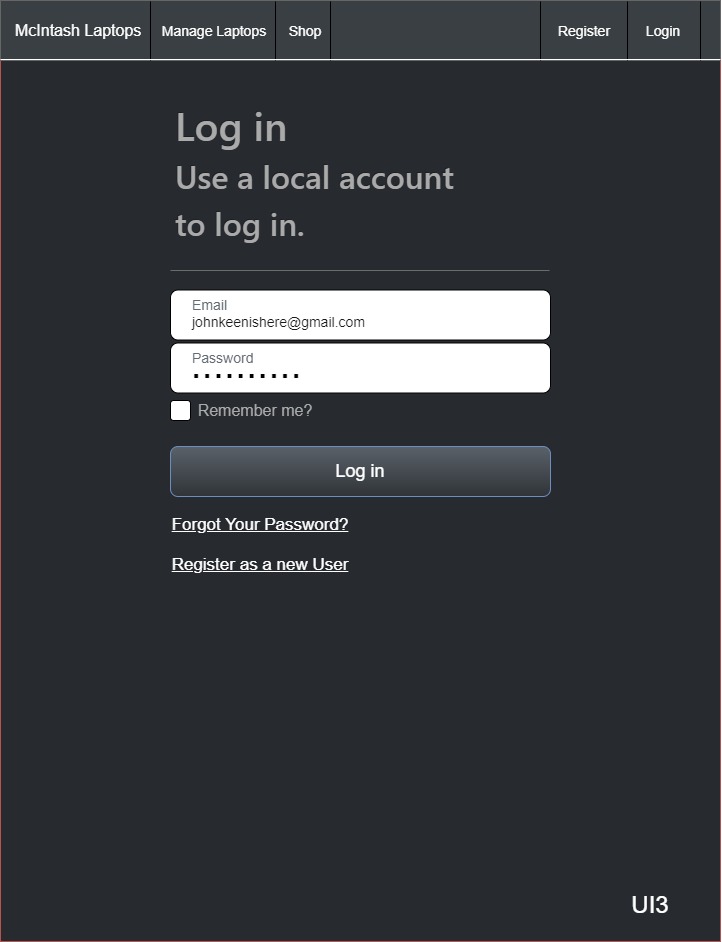
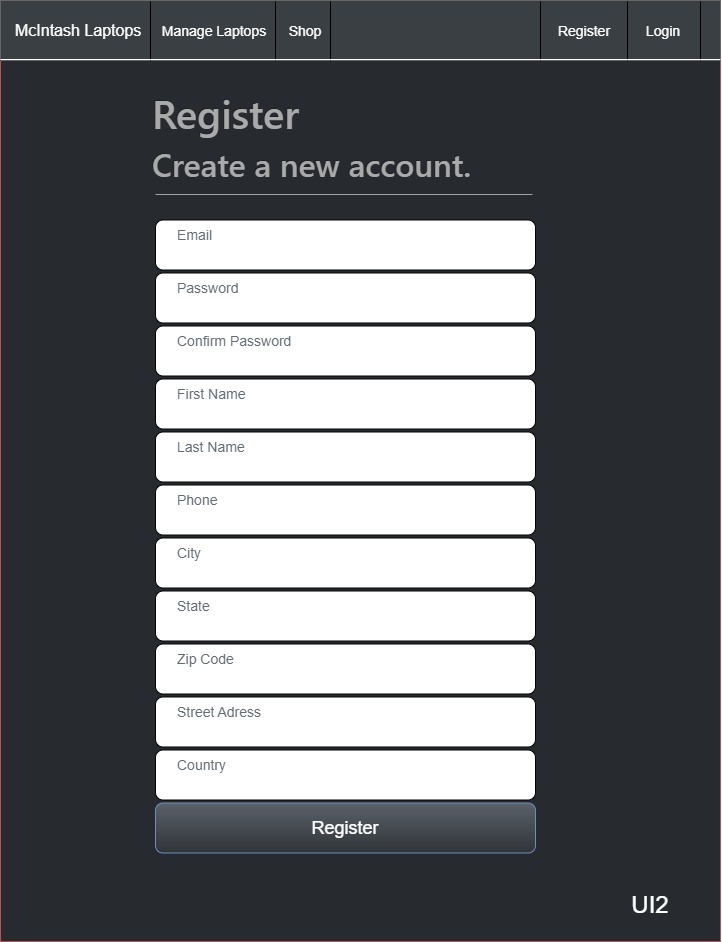
|  |
| --- |
| Hardware and Software Technologies |
| 1 - ASP.NET Core MVC |
| 2 – Bootstrap |
| 3 – jQuery |
| 4 – Stripe |
| 5 - Identity Framework |
| 6 - Microsoft’s IDataProtectionProvider |
| 7 – TSL |

# Overview of Design Concepts

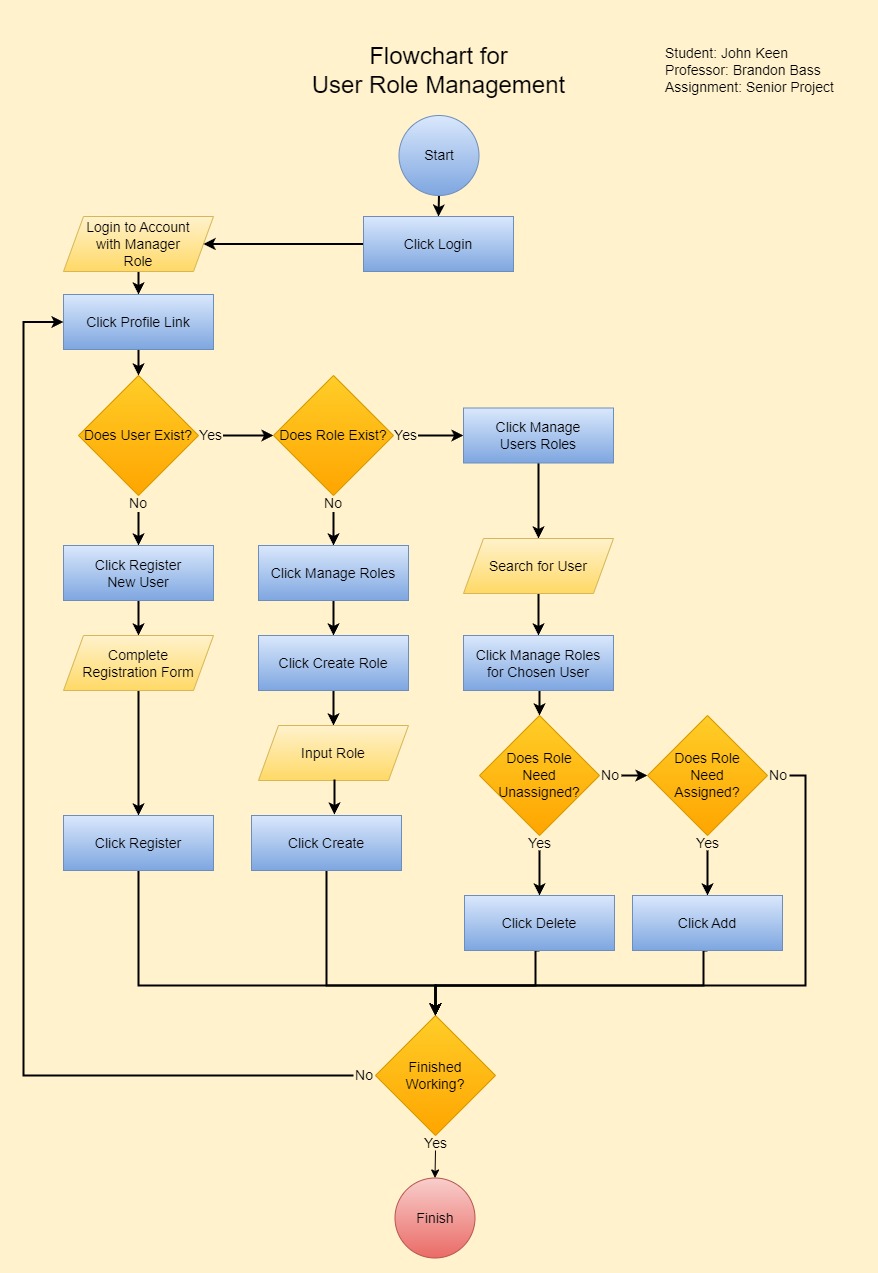
The original need of McIntash Laptops was to provide retailers with catalog information while simultaneously protecting access to the CRUD functionality allotted to users with a designated laptop manager role. An ASP.NET Core MVC web application for managing laptop records and an API for retailers provides a clear separation of concerns in regard to workflow. If the manufacturer chooses to require authentication for REST API access, then they will further decrease their attack surface but even without authentication there is no way the retailer can access the functionality of the site responsible for creating, updating or deleting laptop records through the REST API. The web application responsible for CRUD functionality leverages the ASP.NET Core Identity Framework that allows entire controllers to only be accessible if the user is assigned the required role. This provided an opportunity to leverage the architecture of the laptop management web application for the use of providing e-commerce services to potential customers. Logic in razor pages is utilized in order to only display certain links when the user possess the authorized role. This further decreased the attack surface of the web application because customers are unaware that they are not authorized to access certain pages of the site. This functionality could be useful if McIntash Laptops chooses to build more functionality into the same application. An example would be if McIntash Laptops wanted to provide retailers with a way to order laptops by the pallet. A retailer role could be created and the controller responsible for placing orders would be protected until accessed by users with the authorized role. This is currently beyond the scope of this project but it demonstrates that more functionality could be securely built into this web application which serves as a full solution to the manufacturer’s needs.

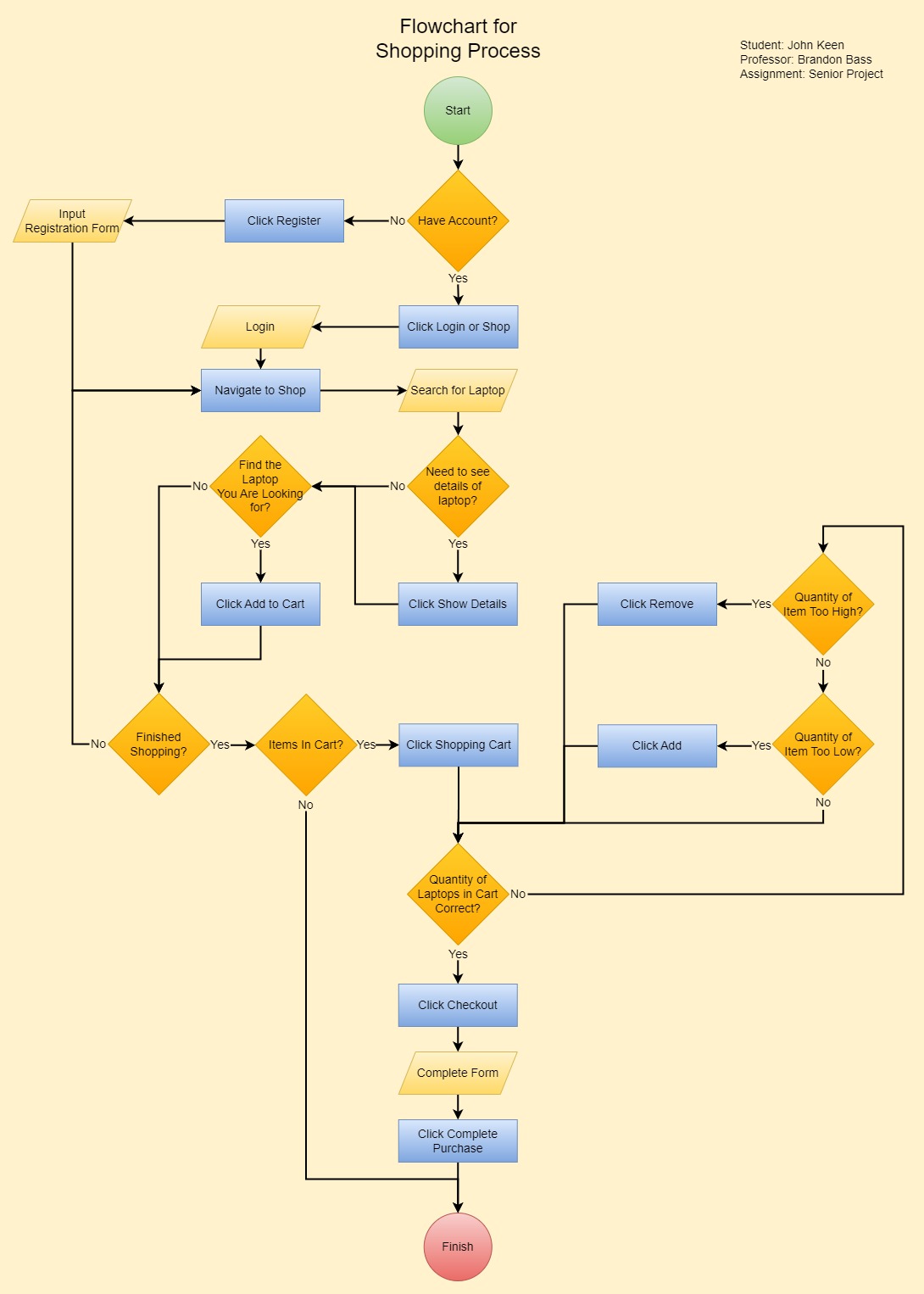
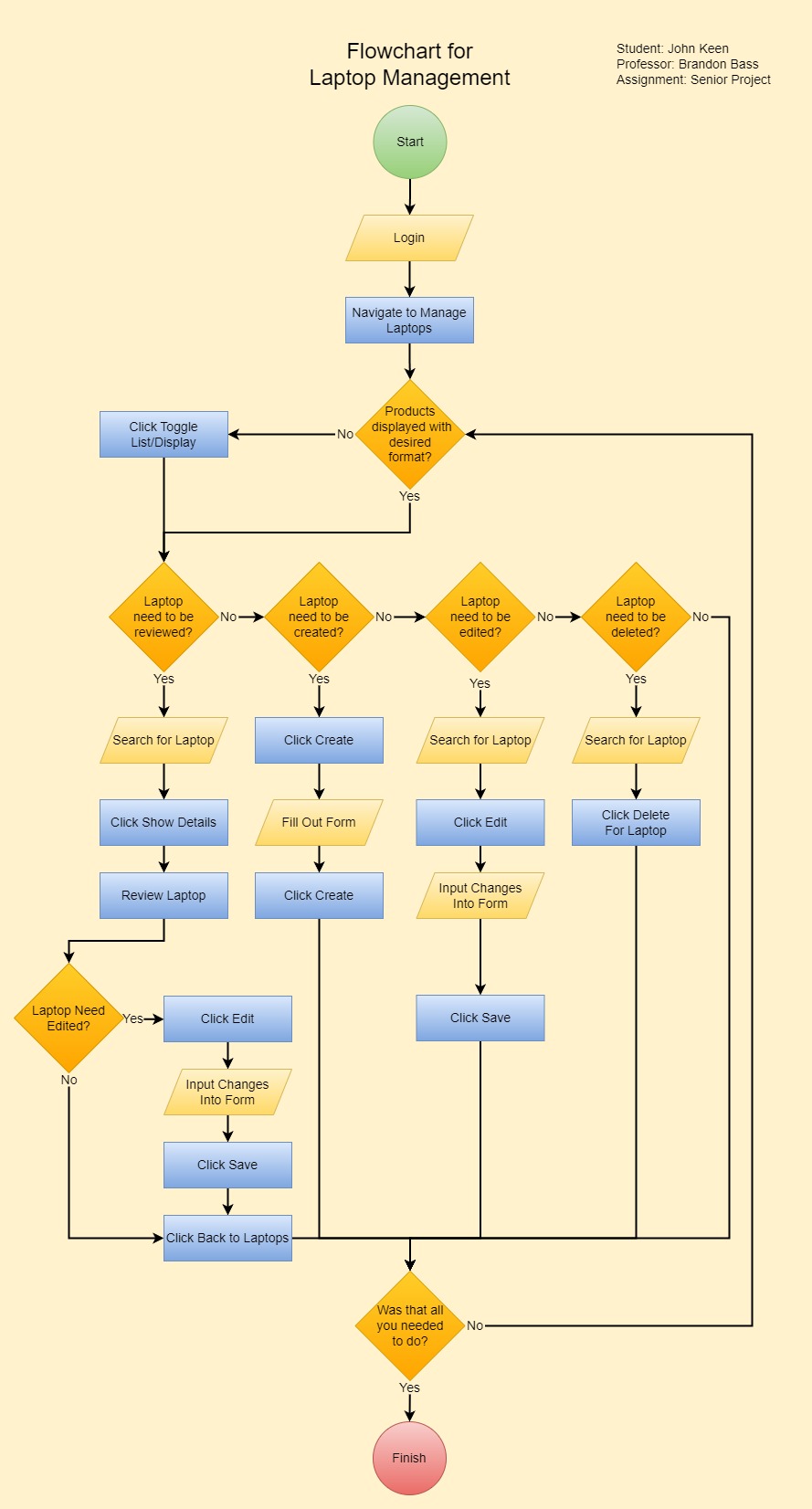
## User Interface Screens (Wireframes)





## Flowcharts





# Detailed Solution Architecture

This laptop catalog design aims at being a comprehensive solution to both the retailers needs to access product information and a customers need to complete a transaction. The design of my project fits into the overall project structure for this project because the REST API allows retailers to freely access product information. The project scope has been modified to allow customers to make complete transactions. This design aligns with the broader objective of building an end-to-end solution for laptop management. The design of the software allows for the development of new features. ASP.NET Core MVC can be used to create new pages or features while taking advantage of the current architecture.

## Security

The web application uses ASP.NET Core Identity Framework for authentication. This framework uses hashing functions to encrypt passwords at rest. It also uses random salting to ensure that a user’s credentials are less likely to decrypted by means of a rainbow table in the event that the user’s credentials are compromised. The web application will use TLS to ensure that the user’s vital information is encrypted in transit. A payment gateway will be used to process credit card information and only the last four numbers of the credit card number will be stored on the manufacturers site. The last four digits of the credit card number will be encrypted at rest using Microsoft’s IDataProtectionProvider. Information provided by users of the web application will be passed to the database as parameters. This will help protect existing data in the database from SQL Injection attacks.

## Explanation of Classes in UML

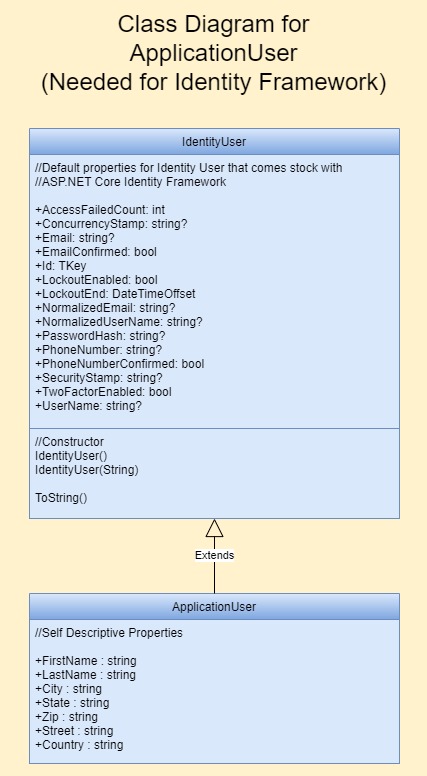
A class called ApplicationUser was created to ensure that properties such as first and last name are incorporated into the user account. The ApplicationUser requires a full address at the time of registration. Its base class is the IdentityUser which comes with the Identity Framework for authentication purposes.

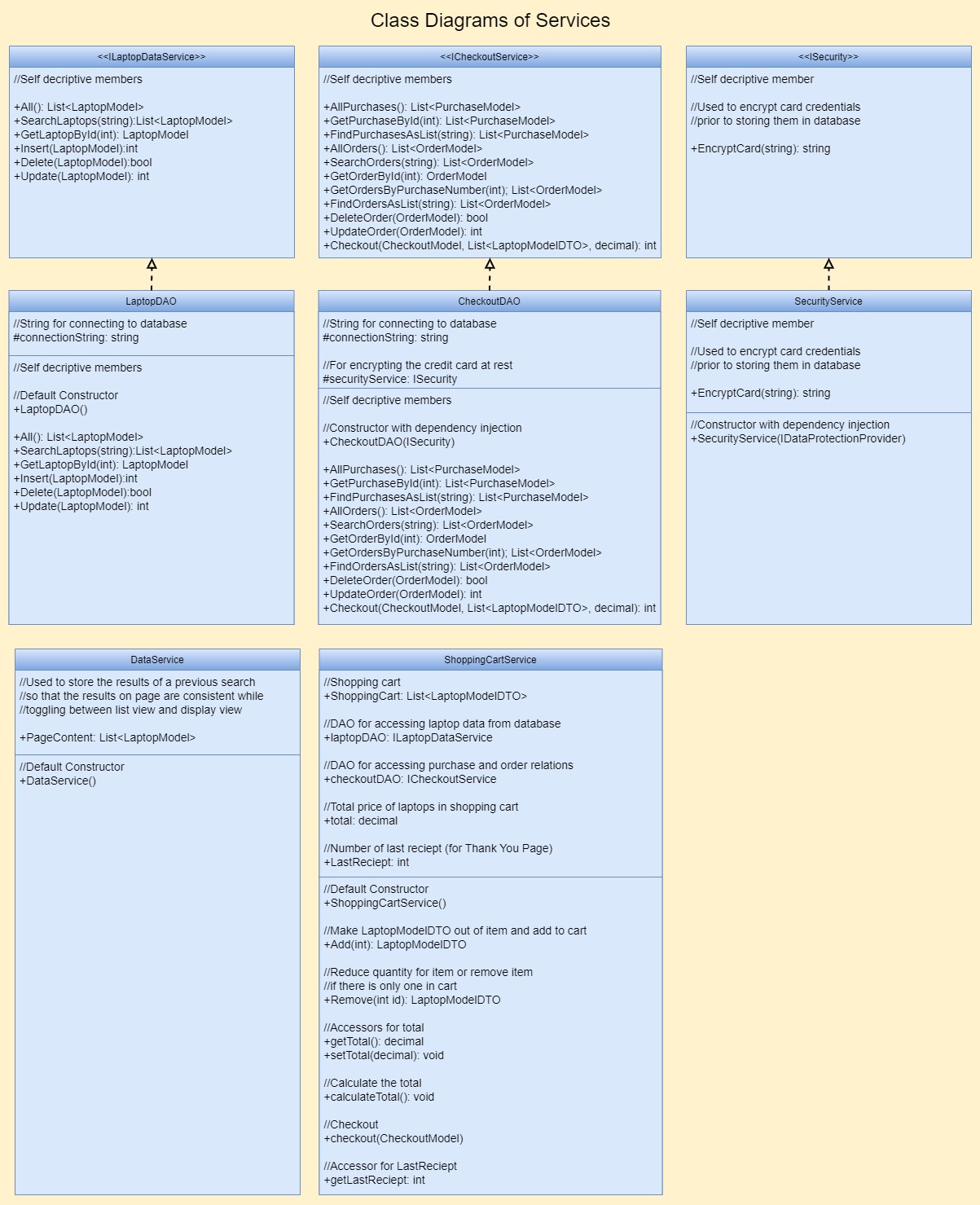
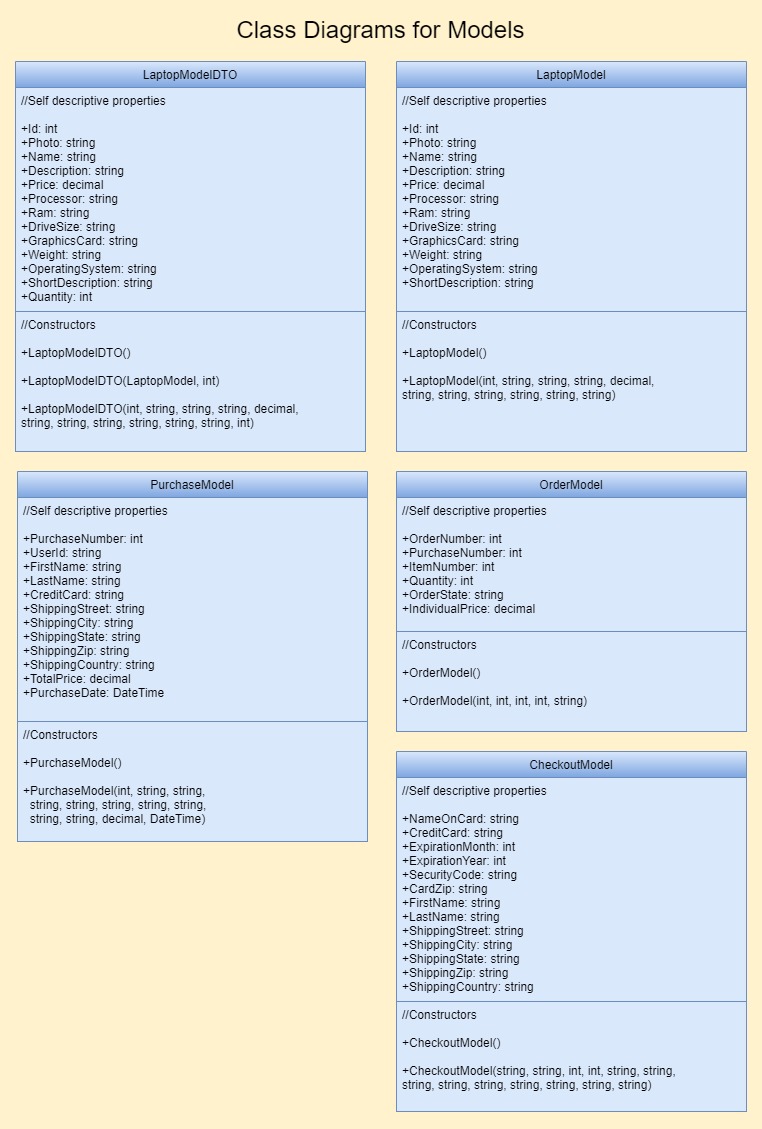
A LaptopModel is a model used for creating, reading, updating or deleting laptops. The LaptopModelDTO is essentially the same as a LaptopModel but it includes the Quantity property. The LaptopModelDTO is used for the shopping cart so that more of one item can be purchased at a time. The PurchaseModel will be used to keep a receipt of the customers purchase information such as the last four of the credit card number, the total price, the shipping address, the date of purchase, the shipping name and the UserId. The OrderModel will be used to keep track of individual items on the receipt, the quantity of the item that was ordered, and the state of the order. The order item will have its own primary key and have the PurchaseNumber as a foreign key so that the purchase information can be found in the database for the order.

So far, the security service is simply for encrypting credit card information. The LaptopDAO will be used for accessing laptop records from the catalog. The ShoppingCartService will be used in the ShopController to hold the shopping cart and provide methods that manipulate the quantity of each item in the cart or items in the cart itself. The CheckoutDAO will be used in the ShoppingCartService to help process a customer transaction. The DataService is used in the LaptopController where laptop managers have CRUD functionality. Its sole purpose is to store the results of a search. This way the results can be passed as a model to a partial view while toggling between “list” view and “display” view. This prevents redundant database queries and ensures that user can easily find what they are looking for while switching between a table and a grid of cards.

The HomeContoller will show the welcome page for the entire web application. Everyone will have access to the welcome page and it will be shown upon arrival and after logging out. The LaptopAPIController is the REST API. It has the endpoints for returning all laptop records, searching for a laptop or returning an individual record for a laptop based on the id. The LaptopController is used for the laptop management section. This is where the user gains the ability to use the CRUD functionality of the web application. It is only available to users with a role of “LaptopManager.” The UserRolesController is used for creating and deleting roles and assigning or unassigning roles to users. This controller is only available to users with a “Manager” role. Users must be logged in to access the ShopController. They do not need a role in order to access the controller but they must have an account. The ShopController is used for the e-commerce section of the website. It has the functionality to add or remove items to the cart or even checkout.

## UML

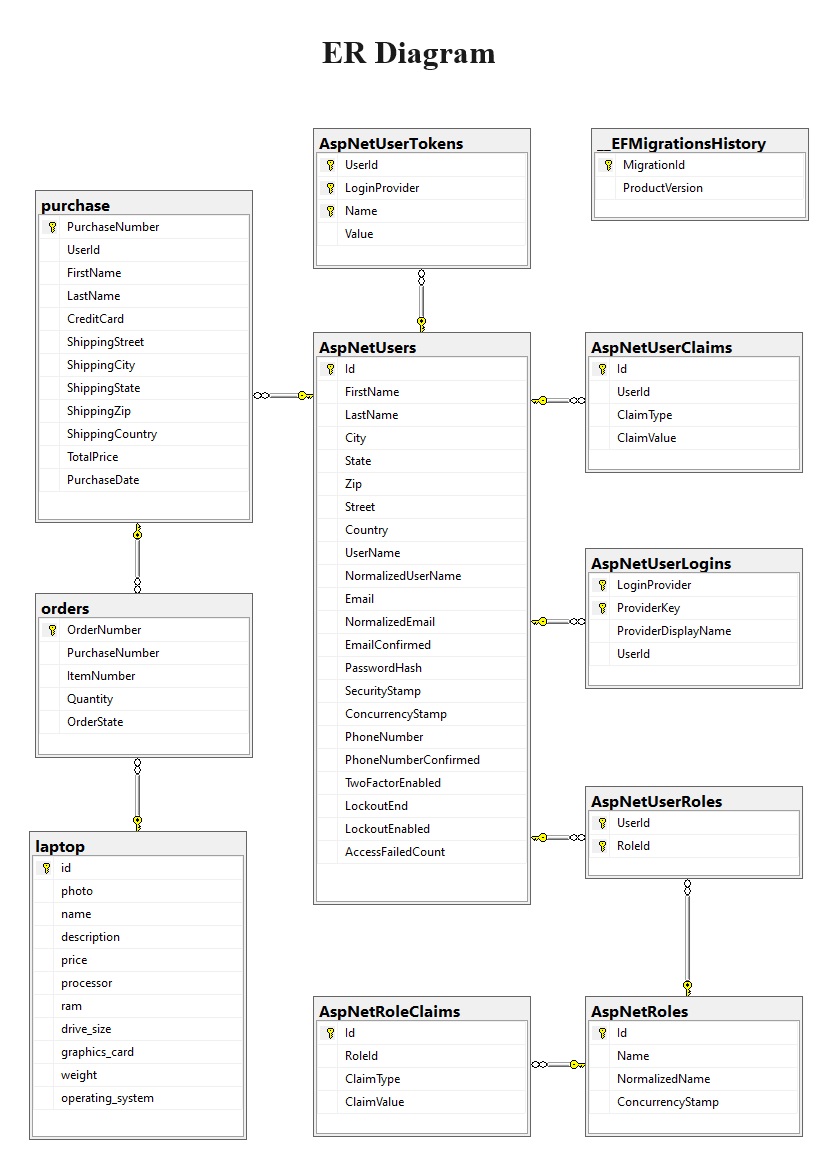






##### ER Diagram Explained

Most of the classes in the ER Diagram are created by the Identity Framework. I modify the AspNetUsers relation by creating and implementing the use of ApplicationUser. I create a migration to update the database with the new information and then update the database. The \_EFMigrationsHistory exists for this purpose. The user data in AspNetUsers is created by the use of the registration form. The AspNetUsersRoles is a junction table with a composite primary key of the UserId and RoleId. This ensures that a role can only be assigned to a user once. I do not modify this table directly but data is inserted or deleted from this table by using the UserManager that is injected into the UserRolesController. Information in the AspNetRoles table where the roles are stored is inserted or removed by means of a RoleManager that is also injected into the UserRolesController. The PurchaseNumber of the purchase relation is a primary key and a foreign key in the orders relation. The id field of the laptop relation is also a primary key and a foreign key in the orders relation as the field ItemNumber. This ensures that each order must belong to a purchase and have an item that is in the database. ON DELETE CASCADE will be used in the foreign key for PurchaseNumber so that all orders corresponding to the purchase are deleted when the purchase is deleted from the database.



## Collaboration Diagrams

