Work Reflection – Chris's Card Collective

Date: 2024-02-15

Project Overview

For my Web Application Programming course, I was tasked with developing a web application using the ASP.NET Razor Pages framework, with SQLite as the backend database engine. In this project, I learned how to implement backend functionality, primarily CRUD operations, and integrate them into my web application. For this project, we had to follow predetermined requirements, but the site's theme and topic were open-ended. I chose to develop an online marketplace where users could create accounts. As a test case, I decided to put the project theme around users creating an account and selling Magic: The Gathering cards.

Key Decisions & Actions Taken

Since this was my first project involving back-end architecture, our instructor required us to use SQLite. Its lightweight and flexible nature made it easy to integrate into our web application. The project was built using the Razor Page framework, which provided automatic model generation and SQL command creation for managing database tables especially with my relatively simple SQLite server.

During my first year at NSCC, I primarily developed using Visual Studio Code. However, in my second year, I transitioned to Visual Studio 2022 Community Edition, which granted me free access to Microsoft's developer tools. This included the .NET framework, advanced debugging tools, and the Package Manager Console, significantly improving my ability to build, test, and deploy both my application and database.

I developed a basic administrative system that allowed administrative users to modify, delete, and add new card listings. This feature was accessible through the user login page in my web application, enabling database entries in SQLite to be updated seamlessly.

Skills Applied & Developed

In my web application, I learned how to create and utilize cookies, primarily for cookie-based authentication. I used BCrypt to hash user passwords, storing the hashed values in my SQLite database. When a user attempted to log in, my application sent a fetch request to my backend server, which compared the stored hash with the hashed input to verify credentials. Upon successful authentication, an authentication cookie was generated to maintain the user's session. This cookie expired once the user closed the page, requiring them to log in again. Additionally, I implemented a 24-hour persistent cookie to track the user's shopping cart. This allowed users to view and manage their

selected items at checkout, even if they navigated away from the site before completing their purchase.

My web application's front end was developed using Bootstrap, primarily for its ability to create a polished and responsive UI efficiently. While the project was graded mainly on its back-end functionality, including cookie-based authentication, it was still expected to have a clean and professional UX/UI design. Bootstrap's grid system, form components, and utility classes allowed me to structure the interface quickly while being responsive to different screen sizes. Since I was using the Razor Page framework, I had to learn Razor syntax to integrate C# logic into my HTML views. I explored various ASP.NET attributes that enhanced form validation and user feedback—such as using `asp-validation-for` to display error messages when a user entered incorrect credentials. Razor Pages also required each `.cshtml` file to have a corresponding `.cs` file to handle server-side logic. Through this, I learned how to create data models, bind form inputs, and generate SQL queries dynamically to interact with my SQLite database.

Project Outcomes

- Functional Admin user page that can create, delete, update, and read database entries with proper cookie verification
- A simple yet clear marketplace app prototype with a home, checkout, confirmation, and other pages

Areas of Improvement

- Expand upon the different types of accounts so that site users can also create an account without being an administrator
- Improve the home pages UI design for displaying cards
- Expand upon the details page to list all available cards for sale and allow users to see the sellers they are buying from and their other listings

Conclusion

This project provided valuable hands-on experience in developing a full-stack web application using ASP.NET Razor Pages and SQLite. Through implementing CRUD functionality, managing user authentication, and structuring a marketplace-style platform, I deepened my understanding of web development. While the current prototype successfully supports admin-based account management and database operations there is room for improvement. Enhancing user roles, refining the UI, and improving the marketplace’s functionality, will help make the application more user-friendly.