Development of a Banking System in ASP.NET Core MVC

A Journey through Database Connections and CRUD Operations

January 5, 2025

Amal Bashir

Introduction

In the digital age, banking systems have evolved to become more than just a place to store money. They are now comprehensive platforms that facilitate transactions, manage accounts, and provide financial services to users. This report details my journey in developing a banking system using ASP.NET Core MVC, focusing on creating a robust application with essential features such as user authentication, database connectivity, and CRUD operations.

Project Objectives

The primary goal of this project was to develop a secure and efficient banking system that allows users to manage their accounts, perform transactions, and interact with administrators. Key objectives included:

* Implementing a user-friendly interface for both customers and administrators.
* Ensuring secure user authentication and authorization.
* Facilitating CRUD operations for account management and transactions.
* Creating a responsive design that works across various devices.

Development Journey

* **Choosing ASP.NET Core MVC**

I chose ASP.NET Core MVC for its powerful features and flexibility. The MVC architecture separates the application into three interconnected components—Model, View, and Controller. This separation allows for easier management and scaling of the application, making it a suitable choice for building a banking system.

* **Setting Up the Project**

The initial setup involved using Visual Studio as the Integrated Development Environment (IDE) and SQL Server for database management. I created a new ASP.NET Core MVC project, ensuring that the necessary dependencies were included in the project file.

* **Designing the Database**

I designed the database schema to accommodate the various functionalities of the banking system. The primary entities included:

* Users: Storing user information, including usernames, passwords, and roles (customer, admin).
* Accounts: Managing user accounts, including balance and account type.
* Transactions: Recording each transaction, including the amount, type (debit/credit), and date.

Key Features Implemented

* **Admin Layout**

The admin dashboard was designed to provide an overview of the system's activities. It included sections for managing users, viewing transaction histories, and generating reports. The layout was intuitive, allowing administrators to navigate easily between different functionalities.

* **User Registration and Login**

I implemented a secure user registration and login system. The registration process included verifying user identity through email confirmation. Passwords were encrypted using hashing algorithms to enhance security. The login process utilized authentication cookies to maintain session integrity.

* **CRUD Operations**

The core functionality of the banking system revolved around CRUD operations.

* **Create:** Users could create new accounts and initiate transactions.
* **Read:** Users could view their account details and transaction histories.
* **Update:** Users could update their profile information and account settings.
* **Delete:** Users could request account deletion, which was handled securely by the admin.

These operations were implemented using Entity Framework Core, which facilitated seamless interaction with the database.

Challenges Faced

Throughout the development process, I encountered several challenges:

* **Data Security:** Ensuring that user data remained secure required thorough planning and implementation of encryption techniques.
* **Error Handling:** Managing errors gracefully was crucial for maintaining a good user experience. I implemented robust error handling to provide users with clear feedback.
* **Responsive Design:** Making the application responsive across devices was challenging, but using CSS frameworks like Bootstrap helped in achieving a mobile-friendly layout.

Each challenge provided valuable lessons and strengthened my problem-solving skills.

Conclusion

The development of the banking system in ASP.NET Core MVC was a rewarding experience that enhanced my understanding of web development and database management. I successfully implemented key features such as user authentication, a functional admin layout, and CRUD operations. Looking ahead, I plan to incorporate additional features, such as real-time transaction notifications and advanced reporting tools, to further enhance the system’s capabilities.

References

* Microsoft Documentation on ASP.NET Core
* Entity Framework Core Documentation
* Online tutorials and courses on MVC architecture