Practical project: Shopping app

420-D05-SU
C#: Programming in a graphical environment

AEC Programming, Networks and Security
AEC Programming and Web Technologies
AEC Video Game Programming

Evaluation weight:	20% of final grade
Due date:	Morning of the final exam

Submission specifications

- Project name: ShoppingApp + name 1 + name 2 + name 3 + name 4
- Your submission must include all of your source code (**.cs**)
- Your submission should include the Visual Studio project file (**.csproj**)
- Your submission should include a documentation text file (a readme file), describing the application's features, including required features implemented, optional features implemented, as well as any other relevant information, including any remaining bugs or problems with your code
- If your project uses a database for persistence, you **must** provide an SQL script or other method of reconstructing your database tables and any initial data they should contain
- For each class, you must write a comment indicating which teammate is the author, or if multiple teammates work on the same class, you must indicate which teammates worked on which methods
- Important: Delete any hidden **.vs** folder, and any **Debug**/**Release**/**obj**/**bin** folders
- Finally, compress your entire project folder into a **.zip** file, including the required documentation
- Submit this **.zip** file on Moodle using the project submission link

Description

Develop a WPF desktop application in C#, using XAML for UI design, which allows customer users and admin users to log in.

Each customer has their own personal shopping cart. A customer should be able to view products in the store's inventory, to add and remove products from their cart, to empty their cart, to change product quantities in their cart, to view their cart's totals, and to check out.

An admin user should be able to view and update product data, to view store inventory data, to add a new product to the inventory, and to order more stock for a product.

Required features

- Log in
    - User must enter username and password
    - Authenticate the username and password
        - Is there a user account with that username?
        - If so, is the password correct?
        - If so, the user successfully logs in
        - Otherwise, display an appropriate error message
    - There should be two different types of users, with different authorization levels: (Authorization = permissions for accessing data and performing actions):
        - Admin
        - Customer
    - Create at least one user account of each type, and provide the username/password for each to me in a readme file submitted together with your project

- Log out
    - When a user is logged in, they can log out
    - After logging out, a user needs to log in again in order to access the system's data/functionality

- Customer user features
    - View all products in store inventory
        - Should include at least name, category, description, (discount?), price
    - Add product to cart
    - Remove product from cart
    - Empty cart (remove all products)
    - Change product quantities in cart
    - View cart totals (subtotal, discount?, tax, total)
    - Check out (purchase all products in cart)

- Admin user features
    - View and update product data of any product
    - View store inventory data, such as:
        - View all products in store inventory
            - Total units sold for each product
            - Total units available (unsold) for each product
            - Total dollars in sales for each product
        - Total dollars in sales for the store
    - Add a new product to inventory
    - Increase the total units available for a product (order more stock)

Optional features

- Data persistence
  - Save the store’s products and inventory data in a database/file, and load this data
  - Save the users’ usernames and password-hashes in a database/file, and load this data whenever a user tries to log in
    - Important: Never store passwords in plain text
    - Best to use a randomly generated salt with the password when generating the hash, and store the salt and hash along with the user name and other user data
  - Save orders in a database/file whenever a customer purchases some products by checking out
- Search inventory for products
  - Any user can browse for products by category (view all and only products in a certain category)
  - Any user can search for products by name or some other data (view all and only products with a matching name, etc.)
- Product discounts
  - Create a system of product discounts
  - This could range between:
    - Simple percentage discount on certain products, and
    - Percentage discount on certain products, with minimum required quantity, maximum allowed quantity, a start date and expiry date, etc.
  - Admin users have the power to create, view, update, and delete discounts
  - Customer users can view applicable discounts when viewing products
- Order history
  - A customer user can view their personal order history
  - An admin user can view the order history of all customers
- Sign up
  - A user can sign up as a customer, creating a new customer user account
  - Validate that the username is not already in use
  - Store the password hash (and the salt, if you use a salt)
- Change password
  - After logging in, any user (customer/admin) can change their password
  - The user must correctly enter their old password to apply the change
- Other features
  - If you wish to add some other additional features not listed here, you are free to do so, but only if you first propose your idea and get approval from the professor

Teams of 3 must complete all required features, plus 1–2 optional features.

Teams of 4 must complete all required features, plus 3–4 optional features.

Regardless of team size, every team must declare their chosen optional features to the professor and get approval. The total amount of work must be proportional to the number of team members. In other words, all students in the class are responsible for completing the same amount of individual work.

Evaluation

This is a team project, and it must be produced by the entire team working together as a group. All work must be original, and must not be copied from any external sources, including any students in other teams or classes. All members in a team are expected to write a roughly equal amount of the code.

The project will be graded according to the following criteria:

Functionality

- Does the program do what it is supposed to do, and meet all requirements?

Input validation

- The application should correctly validate all user input
- The application should handle any errors that can occur during input, including displaying user-friendly error messages to the user

User experience and visual presentation

- All text displayed to the screen should be well-arranged and written in proper English
- The user interface should be user-friendly: it should be easy to use and behave predictably
- The user interface should show the right information to the user at the right time

Structure

- Each method should perform a single simple task (usually a few lines of code)
- If a method is longer than a few lines of code, it should probably be split up into multiple methods
- Each class should only contain fields/properties/methods that are closely related to each other
- If a class contains members that are not conceptually or logically related to each other, then the class should probably be split up into multiple classes
- Unchanging values should be stored in constants, rather than hard-coded as literal values

Format and conventions

- The code should contain relevant comments
- The code should be properly indented
- The code should respect naming conventions
  - The names of variables should be written in **camelCase**
  - The names of files, classes, methods, and properties should be written in **UpperCamelCase** (Pascal case)
  - The names of constants should be written in **UPPER_CASE**