PsuedoCode (P1)

Technology Stack
C# (console app)
xUnit
ADO.NET

A. DATABASE

Brainstorm ideas for a shopping cart. Find area of commonality between all shopping carts, basic identity that is generally shared by all.

1.Create a new database for the GOTStoreDB within Azure.

2.Add tables that correspond to the database
(make sure that tables include at minimum the Customer / User, Product, Location, and Store)

3. Create a PK for each table and think of ways that tables can relate to each other. For example: The final cart should be able to easily follow the user id, product id, and qty from a specific location, otherwise the shopping cart will never be fully functional.

4. Use appropriate FK to create relations between tables.

5. Write database table code in Visual Studio Code

Name the DB first draft, modified, and final with FK in case I need to go back later and make changes.

6. FK's should be chosen to follow UserType > UserAuth > Product,Stock,Locations > Cart > and Customer Order Details

7. Add Values for Products with name, description, and qty based on location. May be more difficult that it appears.

8. Make all FK's an int for uniformity

B. Connecting Azure Database through C# Console App

1.Create the C# console app named Game of Thrones Sword Store

2.Use SQL connection code to copy the link to the database and make sure that the password in (code link) is updated.

3.Check tutorial from class to make sure that the SQL connection is implemented.

4.Make sure that Nuget Package Mgr or appropriate Nuget package is installed.

5. Server=tcp:timchialastriserver.database.windows.net,1433;Initial Catalog=GOTSwordDb;Persist Security Info=False;User ID=TimChialastriRevature;Password[???Inserthere];MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;";

C. Begin C# console app

1. Create classes for each table of the database

2. Use prop tab tab and enter data for the get:set

3. Classes should include UserType, UserAuth, Product, Stock, Locations, Cart, Cart Items, Customer Orders, Customer Order Details

4. Create an application class that can be used for the main section and will implement the application to run

5. Research the best time to use for loop, do while, and switch case options and where they are most effective for a shopping cart.

6. Create a menu using the switch case.

7. Make sure that as long as user does not select quit the application will continue to proceed with options.

while (option != "0")

8.Welcome user and ask if user would like to login or register. (prompt "success" message if valid)

9.Give user option to view products and purchase or quit at anytime
(Add switch case wrapped in while loop so that app continues)

*10. Use sql command to identify userid with database*****

11. User GetInt32 is used to get the value of the specified column as a 32-bit signed integer. No conversions are performed; therefore, the data retrieved must already be a 32-bit signed integer.

12. Give user ability to register: username, full name, password and confirm. Use if statement to prompt user if pw does not match.

13. Use try catch statement to check for errors during registration

14. Once user is registered allow use to place orders (does not need any verification assume that the user had paid per Mark)

15. Count items and calculate total.

16. create method that allows user to cancel a product

17. make sure connection is calculating item qty and order history based off of userID

18. Make sure that user is allowed to exit program at anytime and that connection is logging all entries.

Addtl Notes may need:
ExecuteReader()
Sends the CommandText to the Connection and builds a SqlDataReader.

ExecuteReader(CommandBehavior)
Sends the CommandText to the Connection, and builds a SqlDataReader using one of the CommandBehavior values.

D. Add Xunit Testing and Logging

1. go back and review process and details of unit testing and logging from class materials.
Implement an Xunit unit testing suite.
30% minimum in code coverage.
Logging
Logging is required on most actions, such as when a new user is created, an order is completed, etc.
You must log all actions to a .xml file.
You must provide a method in the logging class that allows a customer to print the logs to the console.
The PrintLogs() method in your logging class should return a List<string> that is then printed to the console.