# Coffee Company Point of Sale System INTRODUCTION

1) Accept drink orders

2) Identify customers by phone numbers. Add new customers who are not in the customer list.

3) Orders can be for named customers in the list or anonymous customers who just want to pay and go

3) Track and award Reward points

4) Accept credit card payment or Reward points

5) Load menu and prices from a JSON data file.

6) Record customers and orders in a JSON data file

7) Produce a CSV file of sales information

==================================

## Architecture

1) C# and Winforms

2) JSON for data storage

3) Use the Decorator pattern to customize drinks, such as "add vanilla" or "no foam".

==================================

## Screens

1) Main screen

- Large buttons to go to Order Drink (for anonymous customers who do not wish to participant in Rewards), Customer List, and Management Tools

2) Customer List screen

- Sorted by last name, then first name, then phone number. The \*anonymous\* customer should always be at the top of the list.

- Includes number of reward points

- Includes button to add a new customer

- Each customer has an Order Drink button

- Include a button to return to the Main screen.

3) Add Customer Screen

- Enter First Name, Last Name, Phone Number. All fields are required.

- Verify phone number is unique and name is at least 10 characters total.

- After adding a new customer, go directly to the Order Drink button.

- Cancelling adding a new customer goes back to the Main screen.

4) Order Drink Screen

- If coming directly from Main screen | Order Drink, the Customer is \*anonymous\*.

- Customer can order one or more drinks.

- Each drink can be customized. Customizations affect the price.

- This screen will have two panes:

+ Left side: drink creator

+ Right side: drinks added to the order, plus subtotal, tax, and total.

- Once there is at least one order, activate the button to go to Payment screen.

- Cancelling the order goes back to the Main screen.

4) Payment Screen

- 10 Reward points are worth $1.

- Customer can pay with a credit card or reward points

+ If credit card, capture (and validate) number and expiration date.

+ Reward points can only be used if the customer has at least as many reward points as is necessary to pay for the entire order.

- If customer pays with a credit card, award the customer 10 Reward points for every $1 spent.

+ Awards can be fractional (round down to nearest integer). For example, a $3.75 drink will generate 3.75 \* 10 = 37.5 points = rounded down to 37 points.

- Anonymous customers do not receive reward points.

- Go to Receipt screen after successful payment.

- Cancelling the order goes back to the Main screen.

5) Receipt Screen

- Shows the drink(s) the customer ordered (including customizations), prices and quantities, subtotal, tax, and total.

- Shows payment method - last 4 digits of credit card or number of Reward points redeemed

- Shows remaining Reward points \*if\* customer is not \*anonymous\*.

- From the Receipt, return to the Main screen.

6) Management Screen

- Include a button to generate a CSV sales report.

+ When clicked, generate a CSV file of all sales and open the file in Excel.

- Include a button to return to the Main screen.

==================================

## Configuration Data

1) On start, load the configuration data from a JSON file.

2) Data includes:

- Drink menu and customizations for each drink

- Tax rate  
 - Reward points per dollar

==================================

## Customer Data

1) The first record in the Customer data file is the \*Anonymous\* customer. This customer does not receive Rewards points.

+ First Name = Anonymous

+ Last Name = Anonymous

+ Phone = 000-000-0000

2) Record structure

+ Customer ID - GUID/string

+ First Name - string

+ Last Name - string

+ Phone - string

+ Reward Points - integer

==================================

## Sales Data

1) Customer ID - GUID/string

2) Date / Time - C# DateTime

3) Tax - C# decimal

4) Subtotal - C# decimal

5) Total - C# decimal

6) Payment method - Credit Card or Rewards

7) List of drinks

- Name - string

- All customizations in a single comma-separated string

- Total price

==================================

## Data Storage

1) There will be three JSON files:

- Configuration Data  
 *This will be in* ***appsettings.json****. The base project I provided already has the beginning of this working and ready for any extension you need.*

- Customer List and Sales Data (stored under the customer record) and the Drink Menu and customizations allowed per drink (stored under each Drink Menu item).  
 *This will be handled by the* ***IStorageService*** *I provided in the base project. That service will read and write Customer and Sales data to one file.*

- the Drink Menu and customizations allowed per drink (stored under each Drink Menu item).  
 *This will be handled by the* ***IStorageService*** *I provided in the base project. That service will read Drink Menu data from another file. You will not be updating the Drink Menu via code. Rather, you will write out the drink data in the menu’s associated JSON file and read it into memory when the application starts.*

2) After any data entry event, re-write the Customer and Sales Data file with the latest information.

3) Load the Configuration data and Drink Menu into memory when the application starts. This data will not be modified by the application.

==================================

## User Interface

1) The user interface will be built using C# Winforms.

2) All prices will be formatted as US currency with two decimal places.

3) All input fields will be validated for proper data (for example, name is required, credit card cannot be expired, credit card must be numeric, etc.)

==================================

## Required Open Source Nuget Packages

1) Newtonsoft JSON

+ For reading and writing JSON data

2) CSVHelper

+ For generating CSV data file

3) CreditCardValidator

+ For verifying credit card numbers (without talking to an actual bank)

==================================

## Code Sharing and Documentation Repository

1) Each team needs a Github repository for code and documentation

2) You will fork the base project from my Git repository. The base project has one screen and the beginning implementation for reading and writing JSON.