



**Module Code & Title**

CS6004NI Application Development .NET

Bislerium Caffee– Desktop Application

**Assessment Weightage & Type**

30% Coursework

**Student Details**

Name: Sita Ram Thing

London Met Id: 22015892

College Id: NP01MA4S220003

Islington College, Kathmandu

11 January 2024

**Git Hub:** [Coffee-Book-DotNet-Application-Development/ at main · devsitaram/Coffee-Book-DotNet-Application-Development (github.com)](https://github.com/devsitaram/Coffee-Book-DotNet-Application-Development/tree/main)

Table of Contents

[**2.** **Background** 2](#_Toc155866497)

[**3.** **User Manual (Instructions to run the program)** 3](#_Toc155866498)

[**3.1.** **Admin Role** 3](#_Toc155866499)

[3.1.1. Admin Login 3](#_Toc155866500)

[3.1.2. Admin change password. 4](#_Toc155866501)

[3.1.3. Add-Ins Coffee and Flavor 5](#_Toc155866502)

[3.1.4. Coffee Order 6](#_Toc155866503)

[3.1.5. View Order History 6](#_Toc155866504)

[**3.2.** **Staff Role** 7](#_Toc155866505)

[3.2.1. Staff Login 7](#_Toc155866506)

[3.2.2. Staff Change Password 8](#_Toc155866507)

[3.2.3. Staff Coffee Order 9](#_Toc155866508)

[3.2.4. Staff View History 10](#_Toc155866509)

[**4.** **Export Data** 11](#_Toc155866510)

[**4.1. Export as JSON Format:** 11](#_Toc155866511)

[**4.2. Export as CSV Format:** 12](#_Toc155866512)

[**4.3. Export as Excel Format:** 12](#_Toc155866513)

[**4.4. Export as PDF Format:** 13](#_Toc155866514)

[**5.** **Software architecture diagram** 15](#_Toc155866515)

[**6.** **Data structure** 16](#_Toc155866516)

[**6.1.** **Shorting (Coffee name, price, and customer number)** 16](#_Toc155866517)

[**6.2.** **Searching** 18](#_Toc155866518)

[**7.** **Details Description of Classes** 22](#_Toc155866519)

[**7.1.** **Enums (Role)** 22](#_Toc155866520)

[**7.2.** **Model classes** 22](#_Toc155866521)

[**7.3.** **Services classes** 24](#_Toc155866522)

[7.3.1. CoffeeAddInServices 24](#_Toc155866523)

[7.3.2. CoffeeServices 24](#_Toc155866524)

[7.3.3. CustomerServices 24](#_Toc155866525)

[7.3.4. OrderServices 25](#_Toc155866526)

[7.3.5. UserServices 25](#_Toc155866527)

[7.3.6. Utils 26](#_Toc155866528)

[7.3.7. Data Export Services (CSV, Excel, JSON, PDF) 26](#_Toc155866529)

[**8.** **Conclusion** 27](#_Toc155866530)

[**References** 27](#_Toc155866531)

List of Tables:

[Table 3: Model table where a short description 23](#_Toc155866584)

**List of Figures:**

[Figure 1: Admin login page 4](#_Toc155866670)

[Figure 2: Admin changes the password. 5](#_Toc155866671)

[Figure 3: Admin add the new coffee 5](#_Toc155866672)

[Figure 4: Admin view the coffee order 6](#_Toc155866673)

[Figure 5: Admin report the top 5 purchased coffee and add-in flavour 7](#_Toc155866674)

[Figure 6: Staff Login page 8](#_Toc155866675)

[Figure 7: Staff Password Change 9](#_Toc155866676)

[Figure 8: View the Coffee and Add-Ins flavour page 9](#_Toc155866677)

[Figure 9: Staff Coffee order page 10](#_Toc155866678)

[Figure 10: Staff view history 11](#_Toc155866679)

[Figure 11: Data export into JSON file 12](#_Toc155866680)

[Figure 12: Data export in CSV file 12](#_Toc155866681)

[Figure 13: Data Export in Excel file 13](#_Toc155866682)

[Figure 14: PDF data export 14](#_Toc155866683)

[Figure 15: Software architectures of Bislrium cafe 15](#_Toc155866684)

[Figure 16: LINQ method to sorting data by ascending and descending 17](#_Toc155866685)

[Figure 17: Sorting by coffee price ascending order 18](#_Toc155866686)

[Figure 18: Searching data by LINQ’s method 20](#_Toc155866687)

[Figure 19: Searching the data. 21](#_Toc155866688)

1. **Introduction**

The Bislerium Café Point of Sale (POS) System represents a C# Desktop application, simplifying order-taking and report generation for the staff. This software equips the admin with versatile functionalities, enabling the addition of new coffee variants or flavours, along with the flexibility to adjust the pricing or eliminate items as necessary. Additionally, the Admin holds the authority to modify staff passwords if required, ensuring security protocols remain robust. For the staff, this system facilitates coffee ordering and customization based on individual customer preferences. They input unique Customer IDs to unlock discounts or even offer a complimentary coffee in adherence to the system's guidelines, enhancing customer experience and loyalty.

Developed within Visual Studio 2022 and primarily utilizing C# as its coding language, the Bislerium Café Point of Sale System embodies a comprehensive digital solution. Its primary goal lies in furnishing the Café with an efficient inventory management tool and a streamlined operational framework. Beyond the mentioned functionalities, this system incorporates an intuitive user interface for seamless navigation, enhancing staff productivity and customer service. Furthermore, it's designed to integrate with existing databases and systems, ensuring a smooth transition and compatibility with the Café's current infrastructure.

# **Background**

The system designed for Bislerium Café is a multifaceted solution aimed at streamlining their operational efficiency. Beyond simplifying order-taking processes, it incorporates an intuitive interface that facilitates seamless communication between the staff and customers. This digital infrastructure centralises order management and integrates real-time inventory monitoring. Offering insights into the current availability of coffee blends and flavours empowers the staff to make informed recommendations and promptly update customers on stock status, ensuring a smoother service experience.

Moreover, the system's comprehensive reporting functionalities extend beyond just order summaries. It compiles detailed transaction logs, providing invaluable data for strategic decision-making. These reports offer insights into popular menu items, peak hours, and customer preferences, aiding in optimizing inventory levels and tailoring marketing strategies. The database architecture not only stores transaction records but also employs robust encryption protocols to safeguard sensitive customer information, ensuring data privacy and compliance with regulatory standards.

In its commitment to security, this system employs a multifaceted authentication framework. Access controls and user privileges are meticulously configured, ensuring that only authorized personnel can access specific levels of data. Multi-factor authentication mechanisms add an extra layer of protection, safeguarding the integrity of the database against unauthorized intrusions. Regular security audits and updates further fortify the system, providing ongoing protection against emerging threats and vulnerabilities, establishing it as a robust and resilient digital infrastructure for Bislerium Café.

# **User Manual (Instructions to run the program)**

To launch the Bislerium Cafe C# desktop application (Cafe Management System), follow these steps:

* Ensure that your computer has Microsoft Visual Studio 2022, C#, and .NET MAUI Blazor installed.
* Open Visual Studio 2022 and choose "Open a project or solution" from the main menu.
* Navigate to the location where the project files are stored and open the "BisleriumCafe.sln" file.
* Click "Start" to initiate the application.
* On the login screen, enter the administrator's username and password, which are set as "admin" for both, then click "Log In."
* Upon the initial login, users will be prompted to update their initial password to a new one.
* Once logged in, the main window of the program will be displayed, showcasing the current inventory, checked-out items, and various other features as described below. This includes functionalities specific to cafe management, allowing users to efficiently handle cafe-related tasks.

## **Admin Role**

### Admin Login

To access the Cafe Management system, the admin must log in with the username and password provided by the admin. Where the admin can access the default username and password for login.

A screenshot of a login box

Description automatically generated

Figure 1: Admin login page

### Admin change password.

The system has after admin login valid the navigate for change in password options.

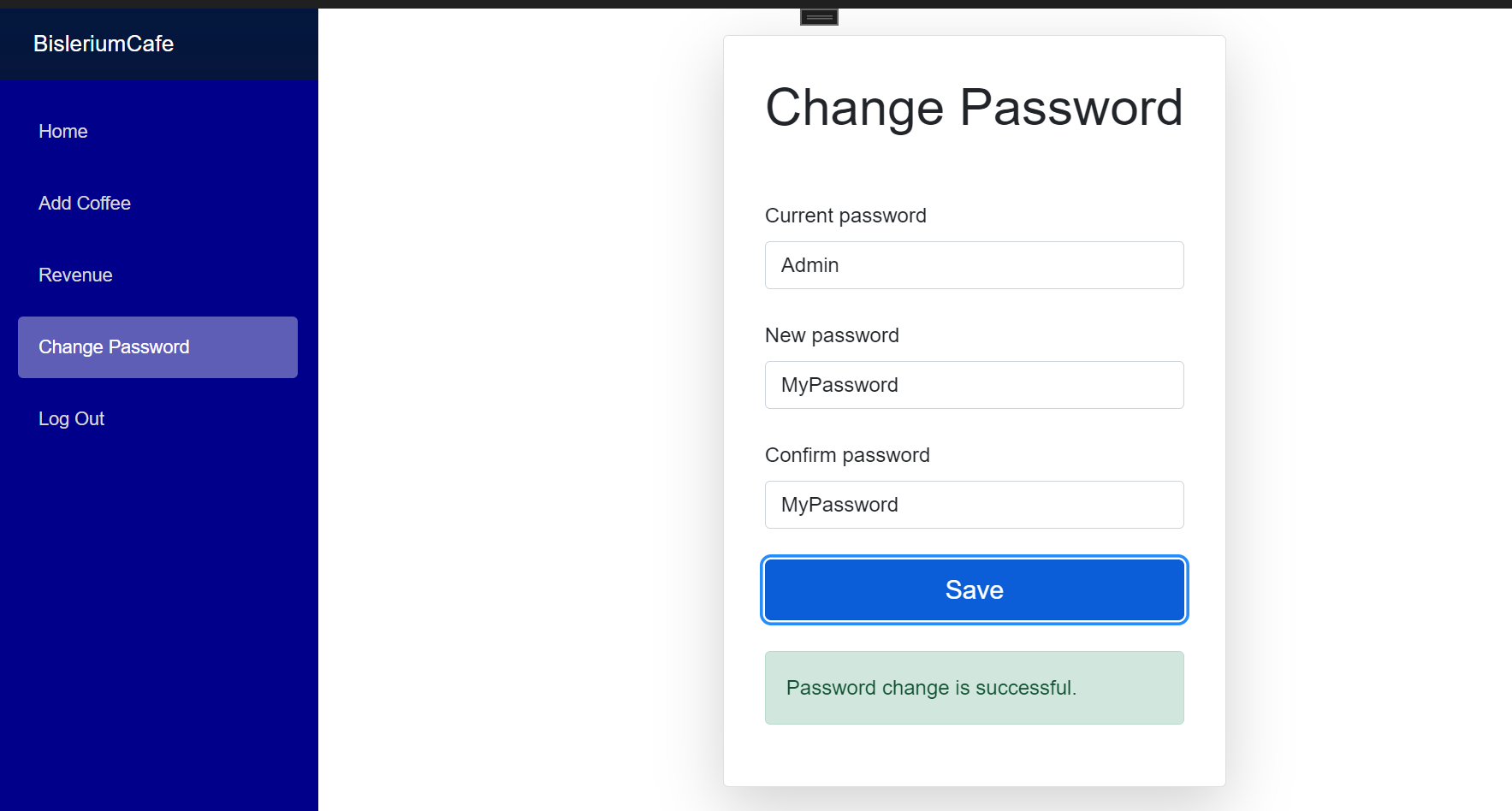


Figure 2: Admin changes the password.

### Add-Ins Coffee and Flavor

The admin can add the new coffee and price and also add the add-in coffee flavours.

A screenshot of a computer

Description automatically generated

Figure 3: Admin add the new coffee

### Coffee Order

The admin has only permission for coffee orders or sales where the system fully allows to admin the coffee and add-in flavour edit, update, or delete options.

**A screenshot of a computer

Description automatically generated**

Figure 4: Admin view the coffee order

### View Order History

The admin can view the order history where they can filter, sort, search, data export and report.

A screenshot of a computer

Description automatically generated

Figure 5: Admin report the top 5 purchased coffee and add-in flavour

## **Staff Role**

### Staff Login

To access the Cafe Management system, the admin must log in with the username and password provided by the admin the staff have a default register password, but it does not show the password for staff. Where the staff can enter the password and then access the default username and password to log in to the system.

A screenshot of a login screen

Description automatically generated

Figure 6: Staff Login page

### Staff Change Password

The system has after staff login valid the navigate for change in password options for staff.

A screen shot of a computer screen

Description automatically generated

Figure 7: Staff Password Change

### Staff Coffee Order

The staff have only permission for coffee orders or sell not allow the coffee and add-ins flavor edit, update, or delete options.

A screenshot of a computer

Description automatically generated

Figure 8: View the Coffee and Add-Ins flavour page

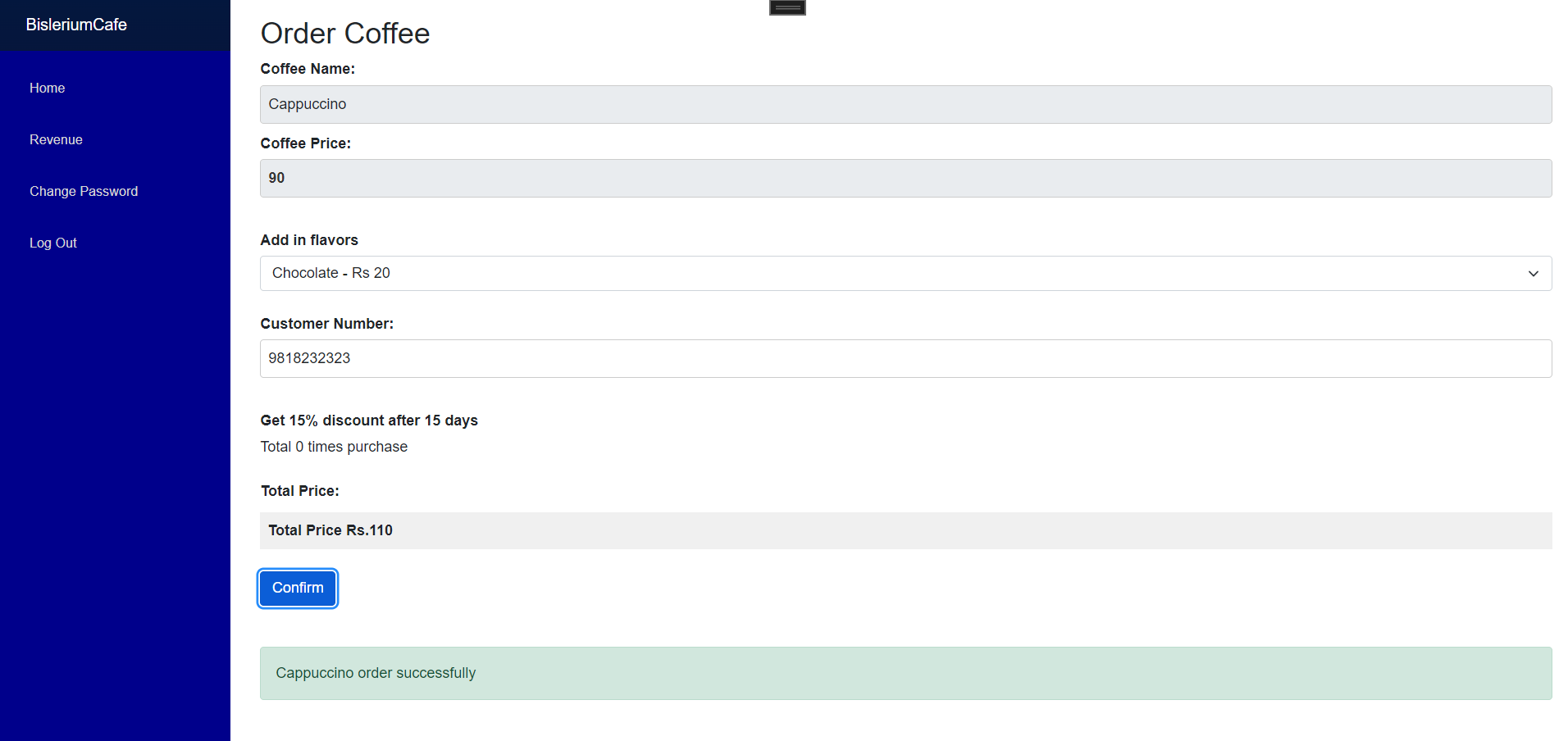


Figure 9: Staff Coffee order page

### Staff View History

The staff have also viewed the history where searching, sorting, and data export of the report form.

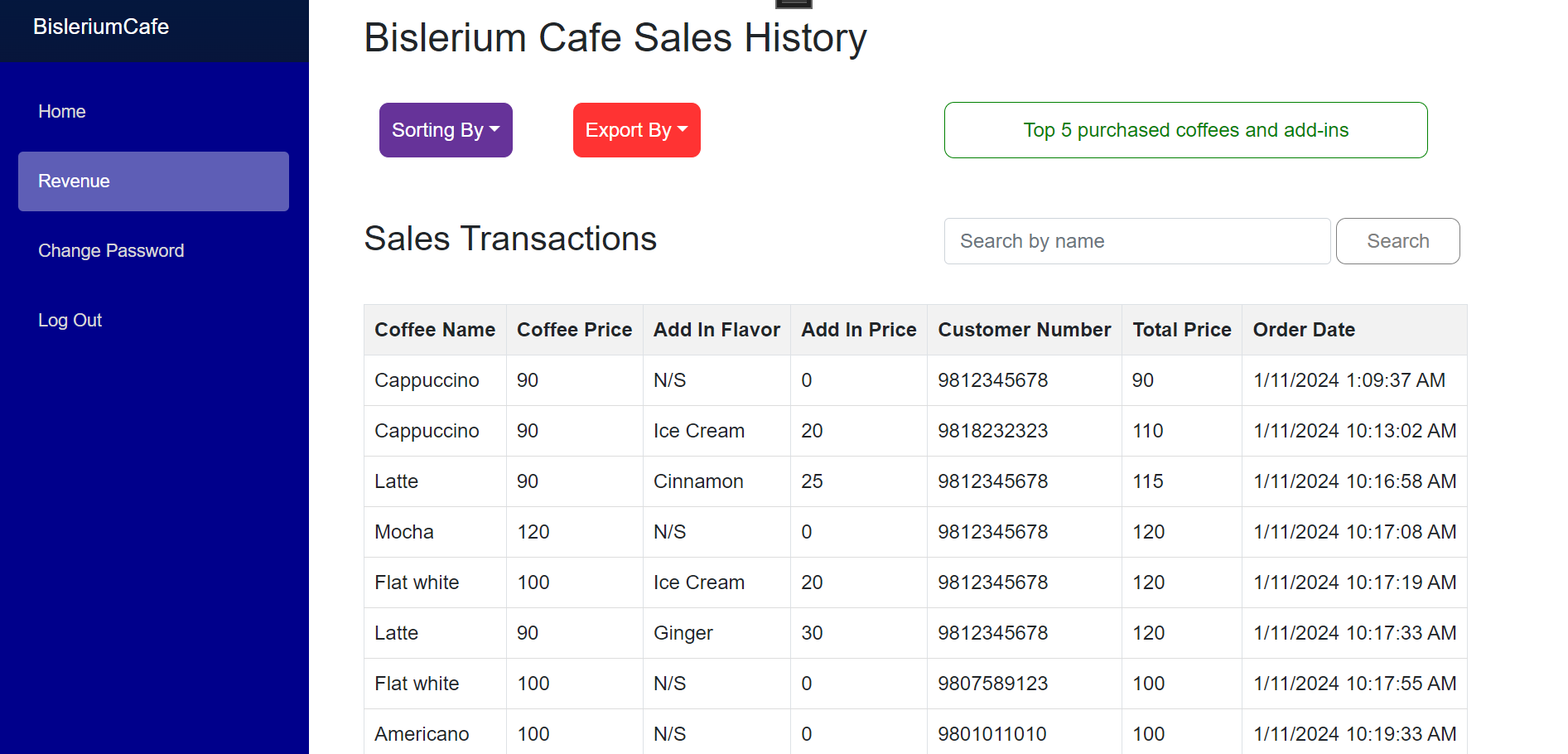
****

Figure 10: Staff view history

# **Export Data**

The system has order history data and can export different data formats like JSON, CSV, Excel, and PDF and that is given below:

## **4.1.** **Export as JSON Format:**

The café's system compiles and exports its coffee-selling transaction data into a JSON format, enabling structured representation for easy sharing or storage. (Syncfusion, 2024)

**A screen shot of a computer

Description automatically generated**

Figure 11: Data export into JSON file

## **4.2. Export as CSV Format:**

Another export option involves converting the transaction data into CSV format, suitable for spreadsheet applications, simplifying data manipulation and analysis. To export data to a Comma Delimited (CSV) file for an Excel application, use the Response object rather than the File object. (Washington, 2024)

**A computer screen shot of a program

Description automatically generated**

Figure 12: Data export in CSV file

## **4.3. Export as Excel Format:**

The system facilitates the export of transaction data in Excel format, ensuring compatibility with Excel spreadsheets for detailed and organized data representation. Information stored in ADO.NET objects like DataTable, DataColumn, and DataView can be efficiently transferred to Excel spreadsheets. This transfer can include various options such as recognizing column types or cell value types as column headers, incorporating hyperlinks, and handling large datasets swiftly, typically accomplished within a matter of seconds. (Manohar, 2024)

**A screenshot of a computer

Description automatically generated**

Figure 13: Data Export in Excel file

## **4.4. Export as PDF Format:**

Additionally, the café's system provides functionality to export the transaction data in PDF format, creating a portable and universally viewable document for convenient sharing and archiving purposes. (QuestPDF, 2024)



Figure 14: PDF data export

# **Software architecture diagram**

A software architecture diagram serves as a visual representation depicting the arrangement and structure of the components within a computer system. Its purpose is to enhance comprehension of the system's layout and organization, commonly utilized by engineers as a means of articulating their concepts. Analyzing a software architecture diagram provides a comprehensive overview of a system, showcasing its key components and their interrelationships. This visual representation proves particularly valuable in the design or modification of the architecture for both new and existing systems. Additionally, the software architecture diagram can illustrate data flow and system workflows, offering insights into the intricacies of information processing within the system. In essence, a software architecture diagram is an invaluable tool, streamlining the assessment and design processes involved in shaping the architecture of a computer system.

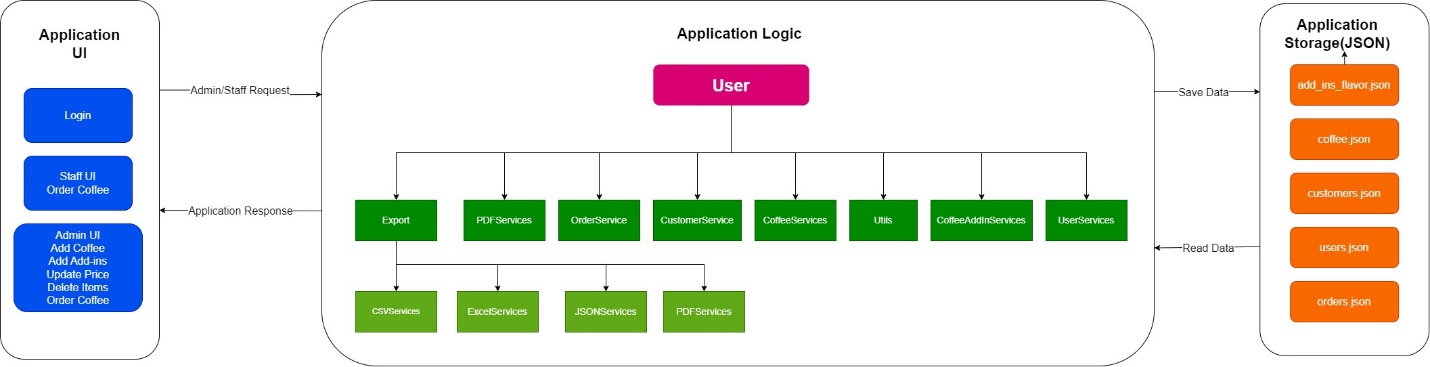


Figure 15: Software architectures of Bislrium cafe

# **Data structure**

Language-integrated query (LINQ) represents a collection of technologies that involve the integration of query capabilities directly into the C# language. In conventional practices, queries on data are typically articulated as plain strings, lacking compile-time type checking or support from IntelliSense. (Microsoft, 2024)

## **Shorting (Coffee name, price, and customer number)**

This SortColumn function manages the sorting of a list of CoffeeOrder objects based on specific columns indicated by the columnName parameter. It begins by evaluating if the currently sorted column matches the provided column name and adjusting the sorting direction accordingly. Upon a new column request, it resets the sorting direction to ascending and updates the sorted column variable. Using a switch statement, it discerns the desired column to sort by and employs LINQ's OrderBy or OrderByDescending methods based on the sorting direction. Following the sorting operation, it triggers a UI update to reflect the sorted data in the user interface, facilitating dynamic and column-dependent sorting for the listOfCoffeeOrder. The sorting steps are given below:

Step 1: Data source (list of data).

Step 2: Applying filters (Where clause).

Step 3: Sorting (OrderBy/OrderByDescending).

Step 4: Selecting specific properties (Select clause).

Step 5: Aggregating data (GroupBy, Sum, Average).

Step 6: Result or output.

Step 7: End.

A diagram of a software processing process

Description automatically generated

Figure 16: LINQ method to sorting data by ascending and descending

**A screenshot of a computer

Description automatically generated**

Figure 17: Sorting by coffee price ascending order

## **Searching**

The CoffeeSearch method is crafted to enhance the user's ability to find specific information within a list of CoffeeOrder objects. It begins by validating the provided search term, ensuring it is not null or composed solely of white spaces. If a valid search term exists, the method employs LINQ's Where method to filter the list of coffee orders, focusing on the CoffeeName property. Notably, this comparison is case-insensitive, broadening the scope of potential matches. The filtered results are then stored in the filteredCoffeeOrders list. Conversely, if the search term is empty or invalid, the method gracefully displays the entire original list by creating a new list, ensuring a seamless user experience regardless of the search input. This approach emphasizes flexibility and responsiveness in presenting relevant data to the user. The search types of staples are given below:

Step 1: Start

Step 1: Input the search value.

Step 2: Applying filters.

Step 3: Sorting (OrderBy/OrderByDescending).

Step 4: Selecting specific properties (Select clause).

Step 5: Aggregating data (GroupBy, Sum, Average).

Step 6: Result or output.

Step 7: End

A diagram of a search engine

Description automatically generated

Figure 18: Searching data by LINQ’s method

A screenshot of a computer

Description automatically generated

Figure 19: Searching the data.

# **Details Description of Classes**

## **Enums (Role)**

The provided code snippet is a declaration of a public enumeration named "Role" in a programming language. It defines two enumerator constants: "Staff" and "Admin," representing possible roles within a system or application. This enum can be used to manage and differentiate between different user roles, such as distinguishing staff members from administrators.

## **Model classes**

Model classes in software development typically define the structure and behaviour of data entities. Let's briefly explain each model class:

|  |  |  |
| --- | --- | --- |
| S.N | Model classes | Descriptions |
| 1 | Coffee | Represents information about a type of coffee.  Properties:   * Id: Unique identifier for the coffee. * CoffeeName: Name of the coffee. * CoffeePrice: Price of the coffee.   CoffeeAddIn: |
| 2 | CoffeeOrder | Represents an order placed by a customer.  3Properties:   * Id: Unique identifier for the order. * CoffeeName: Name of the coffee in the order. * CoffeePrice: Price of the coffee in the order. * AddFlavorName: Name of the additional flavour added to the coffee. * AddFlavorPrice: Price of the additional flavour. * CustomerNumber: Customer identifier for the order. * TotalPrice: Total price of the order. * OrderDate: Date and time when the order was placed. |
| 3 | Customer | Represents information about a customer.  Properties:   * CustomerNumber: Unique identifier for the customer (required). * Frequency: Frequency of customer visits (default value is 1). |
| 4 | User | Represents a user in the application.  Properties:   * PasswordHash: Hashed password of the user.   Role: Role of the user (assuming it's an enumeration).   * HasInitialPassword: Indicates whether the user has an initial password (default is true). * CreatedAt: Date and time when the user was created. |

Table 1: Model table where a short description

## **Services classes**

The service class contains various functions for adding, updating, deleting, reading, and writing data for the Bislerium Cafe management application. The description of the service class is given below:

### CoffeeAddInServices

The services add-ins class includes methods for retrieving all available coffee add-ins from a JSON file, adding a new coffee add-in flavour if it does not already exist, updating the price of an existing coffee add-in, seeding initial add-in data, saving the list of add-ins to a JSON file, and deleting a coffee add-in. The methods handle exceptions, such as file operations and data validation, and utilize the Utils class for file path operations. The class demonstrates basic CRUD operations for managing coffee add-ins and is designed for use within a cafe or similar application. Improvements could include enhanced error handling, logging, and further abstraction for better maintainability.

### CoffeeServices

The CoffeeServices class manages coffee-related operations within the Bislerium Cafe management application. The class provides methods for creating a new coffee entry, updating the price of an existing coffee, retrieving a list of all available coffees, seeding initial coffee data, getting a coffee by its name, deleting a coffee by name, and saving the coffee data to a JSON file. The CreateCoffee method ensures the uniqueness of coffee names before adding a new entry, updating the price if the coffee already exists. The UpdateCoffee method modifies the price of an existing coffee. The class also includes methods for retrieving, deleting, and persisting coffee data, with exception handling to address potential errors during these operations.

### CustomerServices

The CustomerService class provides functionalities for managing customer data in the context of a Bislerium Cafe management application. It includes methods to create a new customer, retrieve all customers, get a customer by their unique number, update the frequency of customer visits, determine the discount for a customer based on visit frequency, and save the customer data to a JSON file. The CreateCustomer method checks if a customer with the given number already exists, updating the frequency if so, or creating a new customer otherwise. The UpdateCustomer method increments the visit frequency for an existing customer. The class handles data retrieval and persistence, ensuring the integrity of customer information. Exception handling is implemented to address potential errors during these operations.

### OrderServices

The OrderService class in the Bislerium Cafe management application handles order-related operations. It provides methods to retrieve all existing orders, create new orders, and save orders to a JSON file. The GetAllOrders method retrieves a list of orders from the specified file path, and the CreateNewOrder method creates a new order with details such as coffee name, price, additional flavour, customer number, and total price. It performs validations on input parameters and updates the customer frequency using the CustomerService class. The SaveAllOrders method serializes and saves the updated list of orders to the JSON file, with exception handling to address potential errors during these operations. Overall, the class facilitates order management within the application, ensuring data integrity and providing feedback on the success or failure of order creation.

### UserServices

The UserServices class manages user-related operations in a system, including login, password management, and user creation. It contains constants for the login role and a default password. The class provides methods to retrieve all users from a JSON file, create new users with hashed passwords, seed initial users (admin and staff), perform user login authentication, and change user passwords. The password change method ensures password consistency and updates the password hash. The class handles exceptions, such as file operations and authentication errors, and utilizes the Utils class for file path operations. The methods are designed for managing user authentication and password-related tasks securely. However, improvements could include enhanced error handling, logging, and further abstraction for better maintainability and security.

### Utils

The Utils class provides utility methods for handling file paths, hashing secrets, and verifying hashed passwords. It includes methods to retrieve file paths for various data storage, such as users, customers, orders, revenue transactions, add-in flavours, and coffee. The class uses the special folder "MyDocuments" to create a directory named "Bislerium-Cafe-Data" for storing application data. The HashSecret method generates a hashed representation of a given input using the PBKDF2 algorithm with SHA-256, incorporating a random salt, iteration count, and key size. The VerifyHash method checks the validity of a password against a hashed string. The class employs secure practices for password hashing and demonstrates a modular approach for file path management in a cafe-related application. However, potential improvements could include additional error handling and documentation for better clarity.

### Data Export Services (CSV, Excel, JSON, PDF)

The JSONServices, PDFServices, CSVServices, and ExcelServices classes are utility services designed to facilitate data export functionality within a system. Each class specializes in exporting data to a specific format: JSON, PDF, CSV, and Excel, respectively. These services provide methods for converting internal data structures into their respective formats, allowing users to export and save information from the system in a structured and readable manner. The JSON service likely serializes data to JSON format, the PDF service generates PDF documents, the CSV service creates Comma-Separated Values files, and the Excel service produces Excel spreadsheets. These classes contribute to the system's flexibility by enabling users to export data in multiple formats, supporting diverse use cases for data analysis, reporting, or external integration.

# **Conclusion**

The development of this desktop application, utilizing C#, Visual Studio 2022, and dot.NET MAUI Blazor, has equipped me with a robust foundation for crafting sophisticated Point of Sale (POS) Systems. Engaging in this project has offered hands-on experience, allowing me to practically apply the knowledge acquired during the creation of the system. One noteworthy challenge I confronted involved implementing the Login Authorization process and incorporating a sorting algorithm. The intricacies of Login Authorization demanded meticulous attention to security measures, ensuring the safeguarding of sensitive data. This experience provided valuable insights into data security and user authentication, contributing significantly to my learning journey.

In addition to navigating security challenges, this project has deepened my understanding of data structures and various algorithms. Selecting the appropriate data structures and algorithms for specific tasks has enriched my knowledge and underscored the significance of these choices in enhancing system efficiency. Beyond the technical aspects, the project has been a comprehensive learning experience, enhancing both practical and theoretical knowledge. This multifaceted approach has not only improved my skills and confidence as a C# programmer but has also broadened my understanding of the entire process involved in designing and developing desktop applications.

Overall, this project serves as a pivotal learning milestone, providing a holistic understanding of desktop application development. The knowledge gained encompasses technical proficiency, problem-solving skills, and a nuanced comprehension of system security. Moving forward, I am eager to apply the insights garnered from this project to future endeavours, confident that the lessons learned will significantly contribute to the success of upcoming projects and my continuous growth as a developer.

# **References**

Manohar, J. (2024, 01 11). *Excel Data export core.NET*. Retrieved from Confusion: https://www.syncfusion.com/blogs/post/export-data-to-excel-csharp.aspx

Microsoft. (2024, 01 10). *Language Integrated Query*. Retrieved from .NET: https://learn.microsoft.com/en-us/dotnet/csharp/linq/

QuestPDF. (2024, 01 11). *QuestPDF*. Retrieved from QuestPDF: https://www.questpdf.com/introduction.html

Syncfusion. (2024, 01 11). *JSON Data*. Retrieved from Syncfusion: https://support.syncfusion.com/kb/article/8322/using-json-export-to-export-a-lot-of-data-from-aspnet-web-forms-pivotgrid

Washington, M. (2024, 01 11). *Exporting CSV Files in ASP.NET Core: A Guide*. Retrieved from CoppyProgramming: https://copyprogramming.com/howto/how-to-export-csv-file-from-asp-net-core