Introduction:

Welcome to the documentation of the Hi-Tech Order Management System (OMS) project. In this document, we will explore the design, implementation, and functionality of the Hi-Tech OMS, a Windows Forms Application developed using Microsoft Visual Studio 2022, C#, and SQL Server 2022/2019.

Rony Raug was responsible for developing Modules 1 and 3, while Gabriel focused his expertise on Modules 2 and 4. Additionally, both collaborated on the creation and maintenance of the database, ensuring a comprehensive and integrated approach to the project.

Business Rules:

1. Book Records: Each book record in the system includes essential fields such as ISBN, Title, UnitPrice, YearPublished, and Quantity On Hand (QOH). Books are categorized for easy classification and retrieval.

2. Authors: Multiple authors may contribute to a single book, and each author can be associated with multiple books. Author records contain unique identifiers, first name, last name, and email address.

3. Publishers: Hi-Tech receives computer science books from various publishers, including Premier Press, Wrox, Murach, Prentice Hall, and others.

4. Customers: Hi-Tech's customers are educational institutions, such as colleges and universities in Quebec. Customer information includes name, street address, city, postal code, phone number, fax number, and credit limit.

5. Order Processing: Order clerks are responsible for accepting customer orders via phone, fax, or email. Payments are processed through direct withdrawal from the customer's bank account, as per contractual agreements.

Project Objectives:

- Design and implement a user-friendly Windows Forms Application for efficient order management.

- Ensure adherence to Hi-Tech's business rules and operational requirements.

- Enable order clerks to handle customer orders seamlessly across multiple channels.

- Facilitate secure payment processing and order fulfillment in accordance with contractual terms.

Throughout this documentation, we will delve into the architecture, features, and usage of the Hi-Tech OMS, providing insights into its design rationale and implementation details. Let's embark on this journey to create a robust and efficient solution that meets the needs of Hi-Tech Distribution Inc. and enhances its service delivery to educational institutions in Quebec.

Database diagram



To establish the connection with the database, the following code was employed:

A screen shot of a computer

Description automatically generated

"To access the module, the user needs to log in, which will be verified in the database for existence, returning the user's role to display the accessible module."

A screenshot of a login box

Description automatically generated

**MODULO 1**

The following is the initial segment of the project.

This system enables essential user and employee information management operations, including adding, updating, deleting, and searching/listing user and employee data. It's built using ADO.NET Connected Mode for efficient database programming, ensuring data integrity and operational effectiveness.

Operations:

1. Add/Update/Delete User Information:

- Allow users to add new user information to the system.

- Enable users to update existing user information with relevant modifications.

- Provide functionality to delete user information from the system when necessary.

2. Search/List User Information:

- Implement search functionality to allow users to search for specific user information based on criteria such as name, ID, etc.

- Enable users to list all user information available in the system.

3. Add/Update/Delete Employee Information:

- Allow authorized personnel to add new employee information to the system.

- Provide functionality to update existing employee information with any necessary changes.

- Enable authorized personnel to delete employee information from the system as needed.

4. Search/List Employee Information:

- Implement search functionality to allow users to search for specific employee information based on criteria such as name, ID, department, etc.

- Enable users to list all employee information available in the system.

Technical Requirements:

Database Programming with ADO.NET in Connected Mode:

- Utilize ADO.NET to establish a connection to the database.

- Perform CRUD (Create, Read, Update, Delete) operations on the database using connected mode programming.

- Utilize SQL queries and commands to interact with the database and retrieve/store information as required by the application.

A screenshot of a computer

Description automatically generated

In the Employee module, users have the capability to save, update, search, and delete employee records, as well as display a comprehensive list of all employees. These functionalities provide efficient management of employee information, enhancing organizational operations and productivity.

To save the employee after validation of all fields, we employ the following code.

A computer screen shot of a program code

Description automatically generated

For update an employee, the user of system can list all employees, to verify which employee ID will be update, or can use the option to search.

A screenshot of a computer

Description automatically generated

For list all employees, was used the following code:

A computer screen shot of a program code

Description automatically generated

If user prefer to search a specific field, before update, he can choose one type that options is listed and all employees that meet that requirement will show in the list.

A screenshot of a computer

Description automatically generated

For search option, the code was:

A computer screen shot of a program code

Description automatically generated

After user has all information for update an employee, he can fill al fields and press Update Employee. The code for update is:

A computer screen shot of a program code

Description automatically generated

If an employee is deleted, instead remove the register from database, the status of employee will be changed.

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

For USER management, the user can add/update/search/delete an user.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

For this part of program, the logic of codes is the same as used in employee. Only change the table that is used.

When the user finishes his work, he can do a logout to return the login form.

**MODULO 2**

**Introduction**

This code represents the implementation of a customer management form in a Windows Forms application using C#. The form allows users to add, update, list, search, and delete customers from a database.

Key Features

Add Customer: Users can add new customers by providing information such as name, email, phone number, address, province, and postal code.

Update Customer: Users can update the information of an existing customer.

List Customers: Existing customers can be listed for viewing.

Search Customer: Users can search for customers by ID or name.

Delete Customer: Users can delete an existing customer from the database.

Implementation Details

A screenshot of a computer

Description automatically generated

Main Methods

ClearCustomerFields(): Clears the fields of the customer form.

DisplayCustomerInfo(List<Customers1> customerList, ListView listView): Displays customer information in a ListView.

buttonSaveC\_Click\_1(object sender, EventArgs e): Handles the click event on the "Save Customer" button, validating and saving customer information to the database.

buttonUpdateCust\_Click\_1(object sender, EventArgs e): Handles the click event on the "Update Customer" button, validating and updating customer information in the database.

buttonListAllC\_Click\_1(object sender, EventArgs e): Handles the click event on the "List All Customers" button, displaying all customers in the form.

buttonSearchCust\_Click(object sender, EventArgs e): Handles the click event on the "Search Customer" button, searching for customers by ID or name.

buttonCustDelete\_Click\_1(object sender, EventArgs e): Handles the click event on the "Delete Customer" button, deleting a customer from the database.

comboBoxSearchC\_SelectedIndexChanged(object sender, EventArgs e): Handles the selection change event of the search ComboBox, displaying the corresponding search field.

Input Validations

The code performs various input validations to ensure that the data provided by users is valid, including validation of name, email, phone number, address, province, postal code, and status.

Database Integration

The code uses the Customers1 class to interact with the database, performing operations such as saving, updating, listing, searching, and deleting customers.

Final Remarks

This document provides an overview of the Customer Form code, detailing its key features, important methods, and database integration. It is important to maintain consistency of information and follow best practices when modifying or expanding this code.

**MODULO 3**

**Operations:**

1. **Add Book Information:** Users can add new books to the system, providing details such as ISBN, title, price, quantity, publisher, category, and author information. This operation enriches the database with comprehensive book records, ensuring accurate inventory management.
2. **Update Book Information:** Existing book details, including attributes like title, price, quantity, publisher, category, and author information, can be updated seamlessly. This flexibility empowers users to maintain accurate and up-to-date book records.
3. **Delete Book Information:** The system facilitates the removal of books from the database, ensuring data integrity and efficient resource management. Upon deletion, associated information such as author, publisher, and category details are also appropriately handled.
4. **Search Book Information:** Users can search for specific books using various criteria, such as ISBN, title, author, publisher, or category. This search functionality enhances user experience by facilitating quick and precise retrieval of relevant book records.
5. **List Book Information:** The system offers comprehensive listing functionality, enabling users to view all available books within the database. This feature provides users with a comprehensive overview of the book catalog, fostering informed decision-making.

**Technical Requirements:**

Incorporating the Entity Framework into the project's architecture enables seamless interaction with the underlying database. Additionally, the project extends its data management capabilities to include author, publisher, and category entities. By leveraging Entity Framework's capabilities, developers can efficiently perform CRUD operations on these entities, ensuring optimal data management and system performance. This approach enhances code maintainability and scalability, ultimately contributing to the project's overall success.

A screenshot of a computer

Description automatically generated

For save un author the code is:

A computer screen shot of text

Description automatically generated

For update, the user can search first.

A screenshot of a computer

Description automatically generated

The logic for all options is the same.

A computer screen shot of a program

Description automatically generated

User can list all authors, and the code that will be performed is:

A computer screen shot of a program

Description automatically generated

After user has all information, he can update an author.

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

And the user can delete an author too.

A screenshot of a computer

Description automatically generated

A computer screen shot of text

Description automatically generated

For publisher, the logic of code is the same.

A screenshot of a computer

Description automatically generated

For save a book, the user can add more than one author per book.

A screenshot of a computer

Description automatically generated

When user click in save button, this will save the information in tables: Books and AuthorsBooks.

After all validations in the fields, it will execute the following code:

A computer screen shot of a program

Description automatically generated

And user can update, search and delete a book, using the same logic that was used for author and publisher.

**MODULO 4**

**A screenshot of a computer

Description automatically generated**

# Documentation for OrderClerksForm Code

## Introduction

This code implements a form for order clerks in a Windows Forms application using C#. The form allows order clerks to add, update, cancel, search, and list orders and order lines in a database.

## Key Features

1. \*\*Add Order:\*\* Order clerks can add new orders by providing order details such as ID, date, status, type, employee ID, and customer ID.

2. \*\*Update Order:\*\* Order clerks can update existing orders.

3. \*\*Cancel Order:\*\* Order clerks can cancel existing orders.

4. \*\*Search Order:\*\* Order clerks can search for orders by ID, customer ID, employee ID, or status.

5. \*\*List Orders:\*\* Order clerks can list all orders in the database.

6. \*\*Add Order Line:\*\* Order clerks can add order lines to existing orders, specifying the order ID, ISBN, and quantity.

7. \*\*List Order Lines:\*\* Order clerks can list all order lines in the database.

## Implementation Details

### Main Methods

- `GetISBNs()`: Retrieves a list of ISBNs from the database.

- `GetOrdersId()`: Retrieves a list of order IDs from the database.

- `GetEmployeeIds()`: Retrieves a list of employee IDs from the database.

- `buttonAddO\_Click\_1(object sender, EventArgs e)`: Handles the click event on the "Add Order" button, validating and adding a new order to the database.

- `buttonUpdateO\_Click(object sender, EventArgs e)`: Handles the click event on the "Update Order" button, updating an existing order in the database.

- `buttonCancel\_Click(object sender, EventArgs e)`: Handles the click event on the "Cancel Order" button, canceling an existing order in the database.

- `buttonSearchO\_Click(object sender, EventArgs e)`: Handles the click event on the "Search Order" button, searching for orders based on user-defined criteria.

- `buttonListAllO\_Click(object sender, EventArgs e)`: Handles the click event on the "List All Orders" button, listing all orders in the database.

- `GetCustomerIds()`: Retrieves a list of customer IDs from the database.

- `buttonAddOrderLine\_Click(object sender, EventArgs e)`: Handles the click event on the "Add Order Line" button, validating and adding a new order line to the database.

- `ReduceBookQuantity(long isbn, int quantityOrdered)`: Reduces the available quantity of a book in the inventory after adding an order line.

- `IsQuantityAvailable(long isbn, int quantityRequested)`: Checks if the requested quantity of a book is available in the inventory.

- `buttonListAllOrderLine\_Click(object sender, EventArgs e)`: Handles the click event on the "List All Order Lines" button, listing all order lines in the database.

### Input Validations

- The code performs input validations to ensure that all required fields are filled before adding or updating orders and order lines.

### Database Integration

- The code interacts with the database using Entity Framework to perform operations such as adding, updating, canceling, searching, and listing orders and order lines.

## Final Remarks

This document provides an overview of the `OrderClerksForm` code, detailing its key features, important methods, and database integration. It is essential to maintain consistency in data handling and adhere to best practices when modifying or expanding this code.