Gas station system

By:

Zeyad Amr 42110107

Mahmoud Mohamed 42110489

Moaaz Hossam 42110042

AbdelRahman Osama 42110221

Mahmoud Rafaat 42110309

Mustafa Mahmoud 42110114

Report

The provided code appears to be part of a gasoline and car service pricing system. The system allows users to log in, record sales transactions, search for sales records, modify and delete existing records, and calculate the total price based on various services and fuel consumption. In this report, I will provide an overview of the code and its functionality, discuss potential improvements, and suggest possible future enhancements.

The code consists of multiple event handlers for different buttons and data manipulation functions. Let's break down the key components and functionality of each section:

1. Login Functionality:

- The `btnLogin_Click` event handler is responsible for handling the login process. It checks if the entered username and password match the predefined admin credentials ("admin" and "admin" respectively). If the credentials are correct, it opens a new form (`Form2`) and hides the current form.

2. Sales Recording:

- The `btnSale_Click` event handler is triggered when the user clicks on the "Sale" button. It opens a new form (`Form3`) for recording sales transactions.

3. Loading and Searching Sales Records:

- The `LoadSales` function retrieves all sales records from the database table `salesTable` and returns them as a DataTable.
- The `btnSearch_Click` event handler searches for sales records based on a specified date. It retrieves records from the database table where the date matches the entered search criteria and displays the results in a DataGridView.

4. Modifying and Deleting Sales Records:

- The `btnDelete_Click` event handler deletes the selected sales record from the database table.
- The `btnModify_Click` event handler updates the selected sales record with new values for fuel cost, car service cost, car wash cost, and date.

5. Saving Sales Records:

- The `btnSave_Click` event handler saves a new sales record to the database table. It takes input values for fuel cost, car service cost, car wash cost, and date from the corresponding text boxes and inserts them into the table.

The system provides users with accurate pricing information for their selected options.

Code Analysis:

1. Event Handler and Initialization:

- The code begins by defining an event handler for the "btnTotalPrice" button click event.
- The handler initializes variables such as `result1`, `result2`, `result3`, `liter`, and `total` for storing intermediate and final results.

2. Maintenance Options:

- The code checks various maintenance options using a series of conditional statements.
- If a maintenance option is selected (e.g., "Motor" checkbox), the corresponding cost is added to `result1`.
- Each maintenance option is evaluated individually to calculate the cumulative cost of selected services.

3. Fuel Type and Quantity:

- The code checks the selected fuel type and multiplies it by the input liter value to calculate the fuel cost (`result2`).
- Conditional statements are used to determine the appropriate fuel price based on the selected radio button.
- The cost is calculated by multiplying the fuel price by the input liter value.

4. Car Wash Options:

- The code checks the selected car wash option using conditional statements.
- If a car wash option is selected (e.g., "radioButton4"), the corresponding cost is added to `result3`.
- The cumulative cost of the selected car wash service is calculated based on the chosen option.

5. Total Price Calculation:

- The code adds `result1` (maintenance cost) and `result2` (fuel cost) to obtain the total cost (`total`).
- The calculated costs (`result1`, `result2`, `result3`), and the total price (`total`) are displayed in a message box.

6. Input Validation:

- The code validates the input liter value by attempting to parse it as an integer.
- If the input is invalid, an error message is displayed, and the text box is cleared.

The code provided offers a basic functionality for managing sales records and calculating prices.

1. Input Validation: The code should include validation checks to ensure that the user enters valid input. For example, it should verify that the entered date is in the correct format, numeric values are entered for fuel consumption, and appropriate selections are made for car services.

2. Code Separation: The code currently lacks proper separation of concerns. It would be beneficial to refactor the code into separate classes or modules to improve maintainability

and code organization. This would help isolate the logic for database operations, UI interactions, and business calculations.

3. Use of Prepared Statements: The code currently uses string concatenation to construct SQL queries. This approach can be susceptible to SQL injection attacks. It is recommended to use parameterized queries or prepared statements to ensure safer database operations.

4. Error Handling: The code lacks proper error handling mechanisms. It should include trycatch blocks to handle exceptions and provide meaningful error messages to the user in case of failures.

5. Code Documentation: It is essential to include comments and documentation to explain the purpose and functionality of the code. This would make it easier for other developers to understand and maintain the code in the future.

Import button:

The code connects to a SQL Server database and allows users to manage and track fuel imports. Here's a brief report on the code:

The "imports" class is a partial class derived from the Form class. It contains various event handlers and methods to perform different operations on the database. The SQL Server connection string is defined as a static string variable named "sql".

The form's Load event handler retrieves data from the "importsTable" in the database using the "LoadImports" method. The method establishes a connection, executes a SELECT query, and fills a DataTable with the retrieved data. The DataTable is then set as the data source for a DataGridView, which displays the data on the form.

When a row is selected in the DataGridView, the SelectionChanged event handler updates the text boxes on the form with the corresponding fuel prices and date from the selected row.

The form includes various buttons to perform different actions. The btnDelete button deletes the selected row from the database using a DELETE query. The btnSearch button searches for imports by date using a SELECT query with a LIKE operator. The btnRefresh button reloads the data from the database by calling the LoadImports method again.

The btnModify button updates the selected row in the database with the modified fuel prices and date. It uses an UPDATE query with parameterized values to prevent SQL injection.

The btnTotal button calculates the total price based on the fuel type and quantity entered in the text boxes. It multiplies the quantity by the respective fuel price and displays the calculated values and the total in a MessageBox.

The btnSave button inserts a new row into the database with the entered fuel prices and date using an INSERT query. After successful insertion, a MessageBox is displayed, and the text boxes are cleared.

Overall, the code provides a functional gas station imports module with basic CRUD (Create, Read, Update, Delete) operations on the database. It allows users to view, add, modify, and delete fuel import records. Additionally, it includes features like searching, refreshing, and calculating the total price. The code could be further improved by adding error handling and input validation to enhance robustness and user experience.

In summary, the provided code is a simple but effective implementation of a gas station imports module, providing essential functionalities for managing fuel import records.

In conclusion, the provided code demonstrates a basic gasoline and car service pricing system with functionalities for recording sales transactions, calculating prices, and managing sales records. However, there is room for improvement in terms of input validation, code organization, error handling, and code documentation. Furthermore, there are several potential enhancements that can be implemented to expand the system's capabilities and provide a better user experience.