**Credit Card Validation System**

The Problem statement for a credit card validator project in C++ is as follows:

Your task is to write a program that validates credit card numbers using the Luhn algorithm. The program should prompt the user to enter a credit card number and then validate whether it is a valid credit card number or not.

The Luhn algorithm works as follows:

Starting from the rightmost digit, double every second digit (i.e., the digit at index 1, 3, 5, etc.) and subtract 9 if the result is greater than 9.

Add up all the digits.

If the total sum is divisible by 10, then the credit card number is valid.

For example, consider the credit card number 4388576018402626. Applying the Luhn algorithm, we get:

Double every second digit: 4, 32=6, 8, 82=16 (subtract 9), 5, 72=14 (subtract 9), 6, 02=0, 12=2, 8, 42=8, 02=0, 22=4, 6.

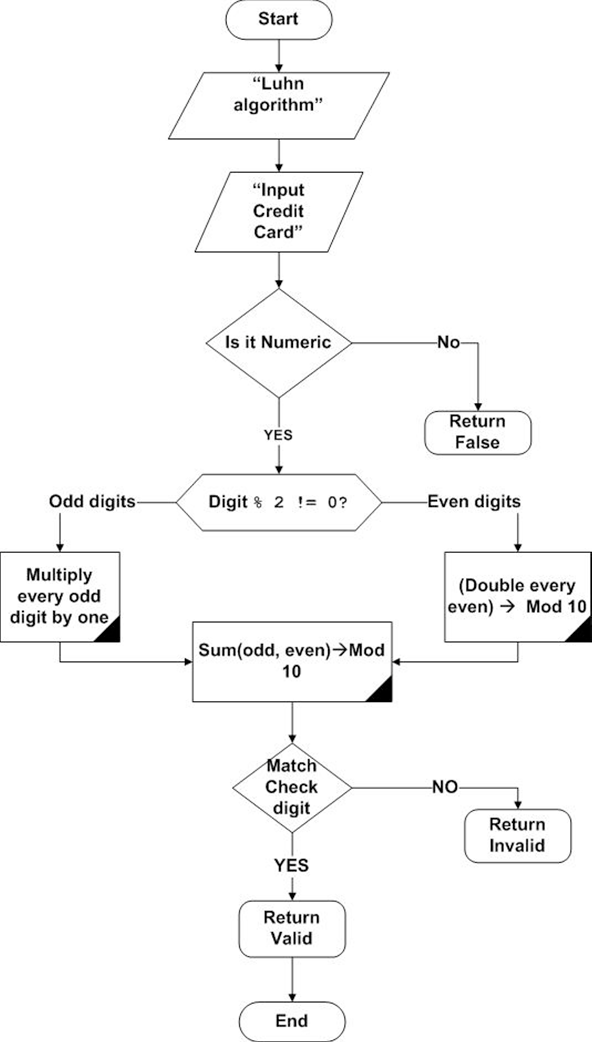
Add up all the digits: 4+6+8+7+5+5+6+0+2+8+8+0+4+6 = 64.

Since 64 is divisible by 10, the credit card number is valid.

Your program should implement this algorithm to validate credit card numbers. Additionally, it should handle invalid input (i.e., non-numeric characters or input that is too short or too long for a credit card number) and display appropriate error messages to the user.

Once the credit card number has been validated, the program should output a message indicating whether the credit card number is valid or invalid.

Luhn Algorithm Flow Chart(Working):



There are several advantages of implementing a credit card validation project:

Security: Credit card validation ensures that the credit card number entered by the user is valid and meets certain criteria, such as the Luhn algorithm. This helps to prevent fraud and ensure that only legitimate transactions are processed.

Accuracy: Credit card validation ensures that the credit card number entered by the user is accurate and free of errors. This can help to reduce the likelihood of transaction failures due to incorrect credit card information.

Customer trust: By implementing credit card validation, you can demonstrate to your customers that you take security seriously and are committed to protecting their sensitive financial information. This can help to build trust and confidence in your brand, leading to increased customer loyalty and repeat business.

Efficiency: By automating the credit card validation process, you can reduce the time and effort required to manually validate credit card numbers. This can help to improve efficiency and streamline your payment processing workflows.

Compliance: Credit card validation is often required by regulatory bodies, such as the Payment Card Industry Data Security Standard (PCI DSS). By implementing credit card validation, you can ensure that you are in compliance with these regulations and avoid potential fines or penalties.

The Concepts that we have used in Mini Project from the lessons taught:

* Inheritance
* Classes and objects
* Access Specifiers
* String operations
* goto keyword
* Types of Inheritance
* Constructors
* Ternary operators
* Exceptional Handling (Runtime error)
* Functions
* Arrays
* Pointers for objects

The credit card validator project has various real-world applications, including:

E-commerce: Online shopping websites often use credit card validators to prevent fraudulent transactions by verifying the validity of the credit card number entered by the user before processing the payment.

Point-of-sale systems: Point-of-sale systems used in retail stores and other businesses can use credit card validators to ensure that only valid credit card numbers are accepted for payment.

Financial institutions: Banks and other financial institutions use credit card validators to verify the credit card numbers entered by customers when they apply for credit cards or make payments.

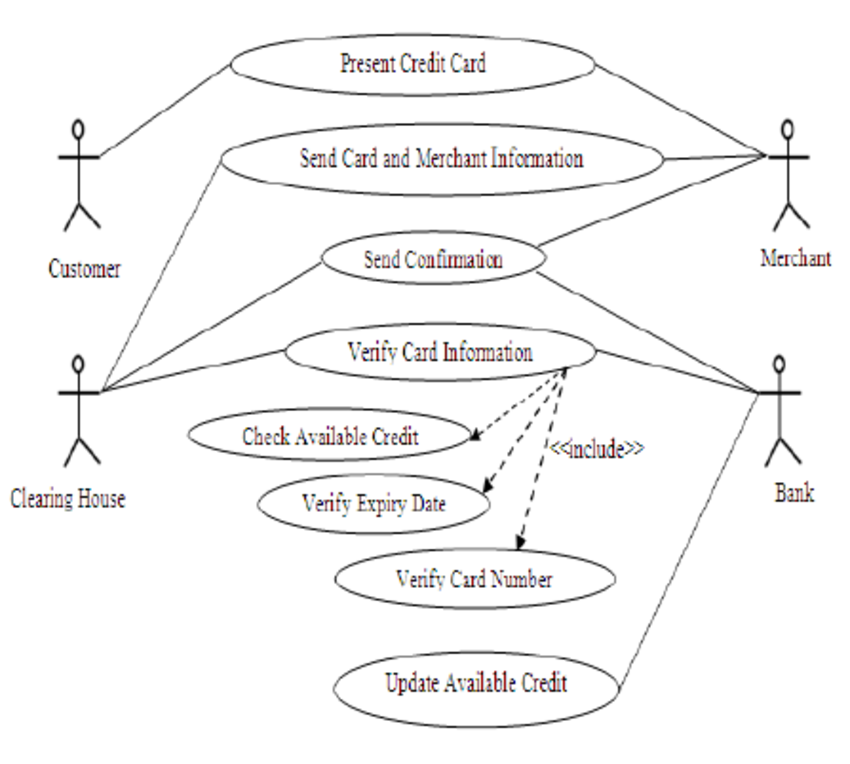
Credit card companies: Credit card companies themselves can use credit card validators to check the validity of the credit card numbers submitted by merchants for payment processing.

Overall, the credit card validator project in C++ is an essential tool for preventing fraudulent transactions and ensuring that only valid credit card numbers are accepted in a variety of industries and applications.

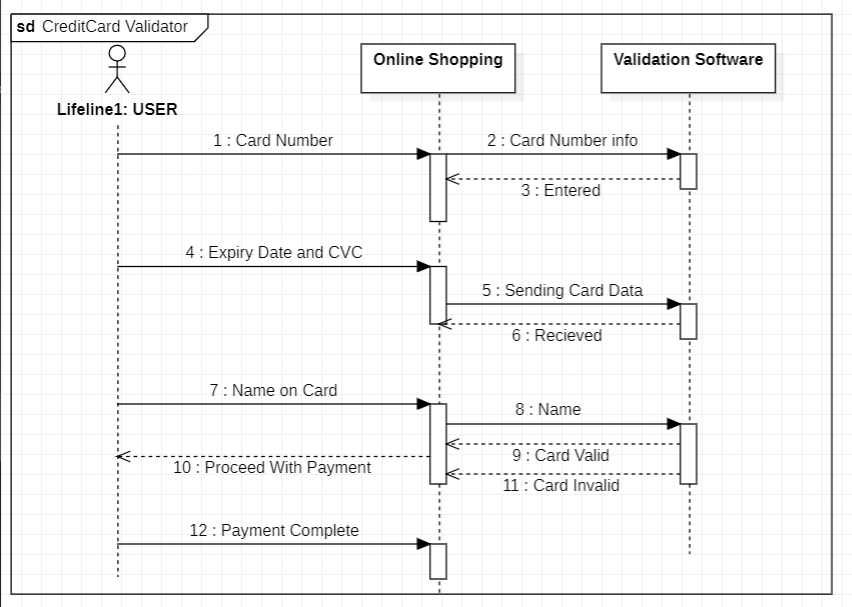
Diagram

Description automatically generatedClass Diagram Of Credit Card Validator:

Use case Diagram:



**Sequence Diagram Of Credit Card Validation:**



Use Case Diagrams are made by STARUML software.

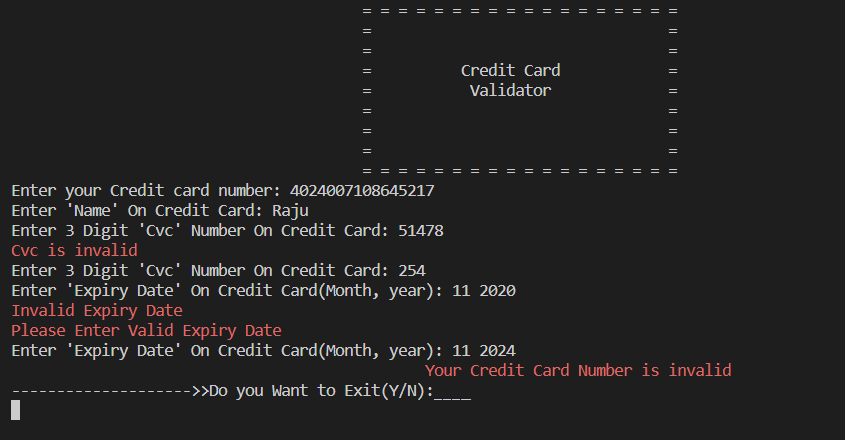
INPUTS AND OUTPUTS:

For Valid Card Details:

A screenshot of a computer

Description automatically generated with medium confidence

For Invalid Inputs:



Code Reusability:

Text

Description automatically generated

DONE By:

Saahil Pradhan And Lalith Reddy.