**PROJECT REPORT**

***Submitted by***

**Lalith Reddy [RA2211003010129]**

**Saahil Pradhan [RA2211003010091]**

***Under the Guidance of***

**Mr.S.INIYAN**

**Associate Professor, Department of Computing Technologies**

***In partial satisfaction of the requirements for the degree of***

## **BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in <.......>**

## Logo, company name Description automatically generated

**SCHOOL OF COMPUTING**

# **COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**MAY 2023**

Logo, company name

Description automatically generated

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

**BONAFIDE CERTIFICATE**

Certified that this Project Report titled **“CREDIT CARD VALIDATION SYSTEM”** is the bonafide work done by **LALITH REDDY[RA2211003010129] and SAAHIL PRADHAN[RA2211003010091]**  who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

| **SIGNATURE**  **Mr.S.Iniyan**  **OODP – Course Faculty**  Associate Professor  Department of Computing Technologies  SRMIST | **SIGNATURE**  **Dr.M.Pushpalatha**  Professor & Head  Department of Computing Technologies  School of Computing  SRMIST |
| --- | --- |

i

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **CONTENTS** | **PAGE NO** |
| 1. | Problem Statement | **1-2** |
| 2. | Modules of Project | **3-4** |
| 3. | Diagrams | **5-** |
|  | 1. Use case Diagram | **5** |
|  | 1. Class Diagram | **6** |
|  | 1. Sequence Diagram | **7** |
|  | 1. Collaboration Diagram | **8** |
|  | 1. State Chart Diagram | **9** |
|  | 1. Activity Diagram | **10** |
|  | 1. Package Diagram | **11** |
|  | 1. Component Diagram | **12** |
|  | 1. Deployment Diagram | **13** |
| 4. | Code/Output Screenshots |  |
| 5. | Conclusion and Results |  |
| 6. | References |  |

**Credit Card Validation System**

**PROBLEM STATEMENT:**

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.

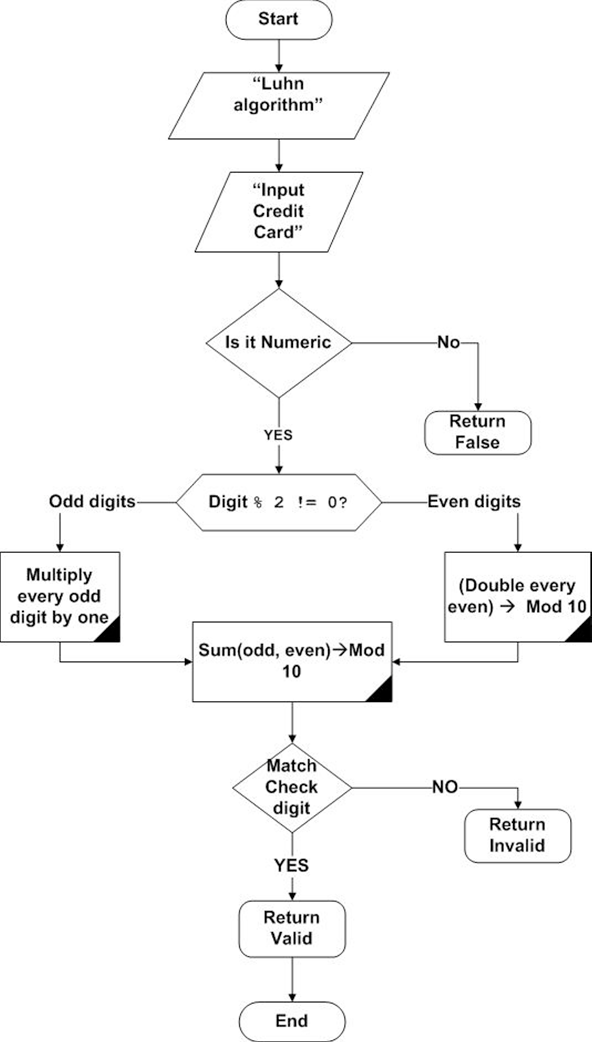
Since 64 is divisible by 10, the credit card number is valid.

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.

1

Luhn Algorithm Flow Chart(Working):



2

**MODULES OF PROJECT**

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

3

* 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.

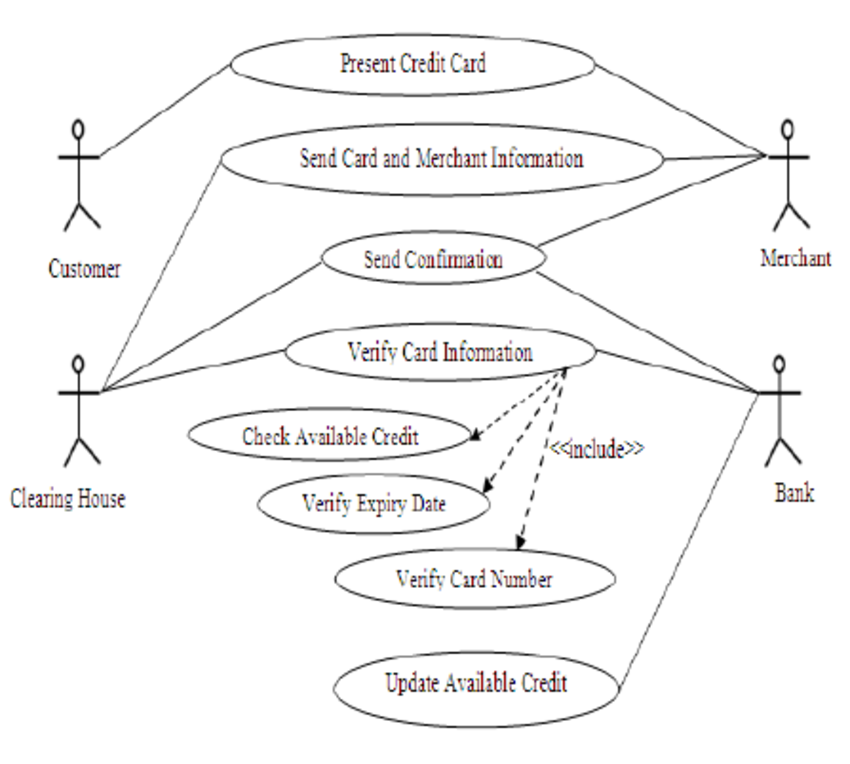
Overall, the credit card validator project in C++ is an essential tool for preventing fraudulent transactions and ensuring that only valid credit card numbers are accepted in a variety of industries and applications.

4

UML DIAGRAMS

**UML Diagrams are made by STARUML software.**

1. USE CASE DIAGRAM:



5

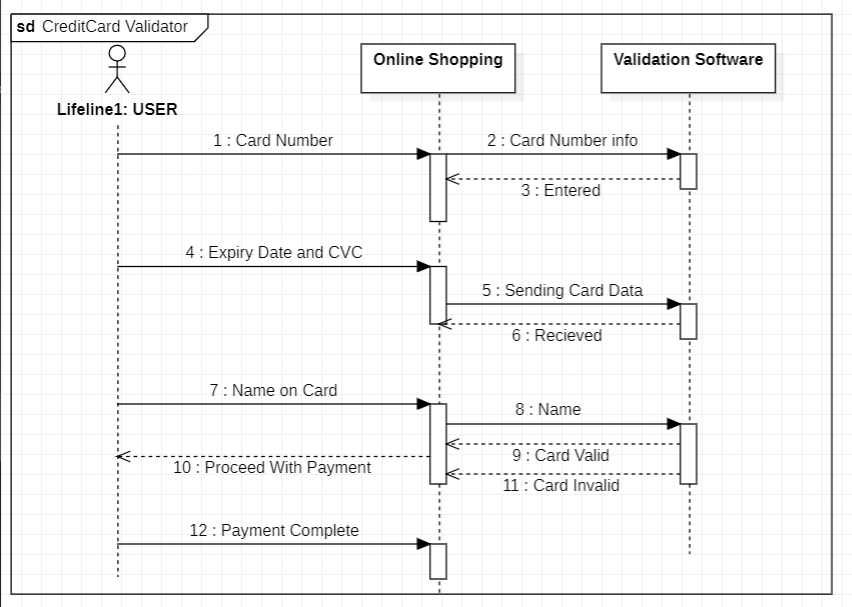
1. CLASS DIAGRAM:

Diagram

Description automatically generated

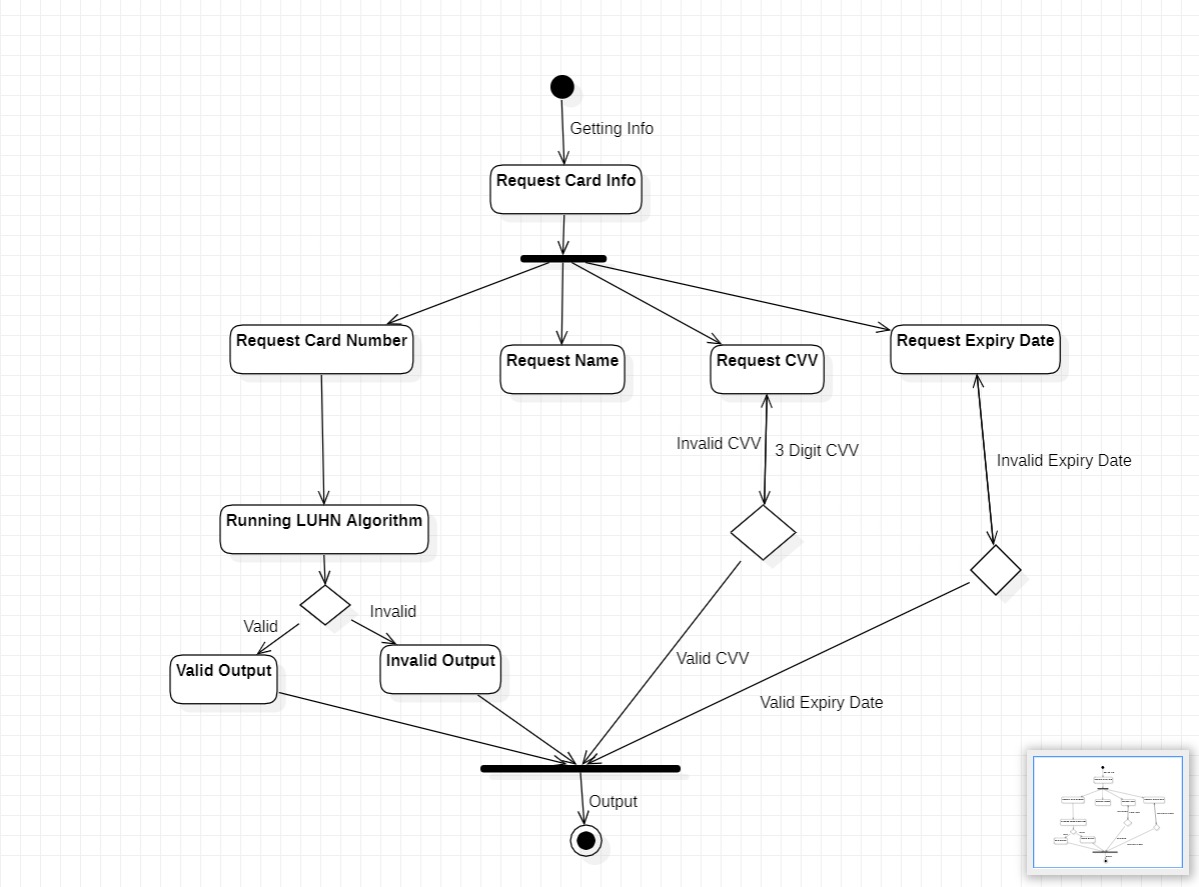
6

1. SEQUENCE DIAGRAM:

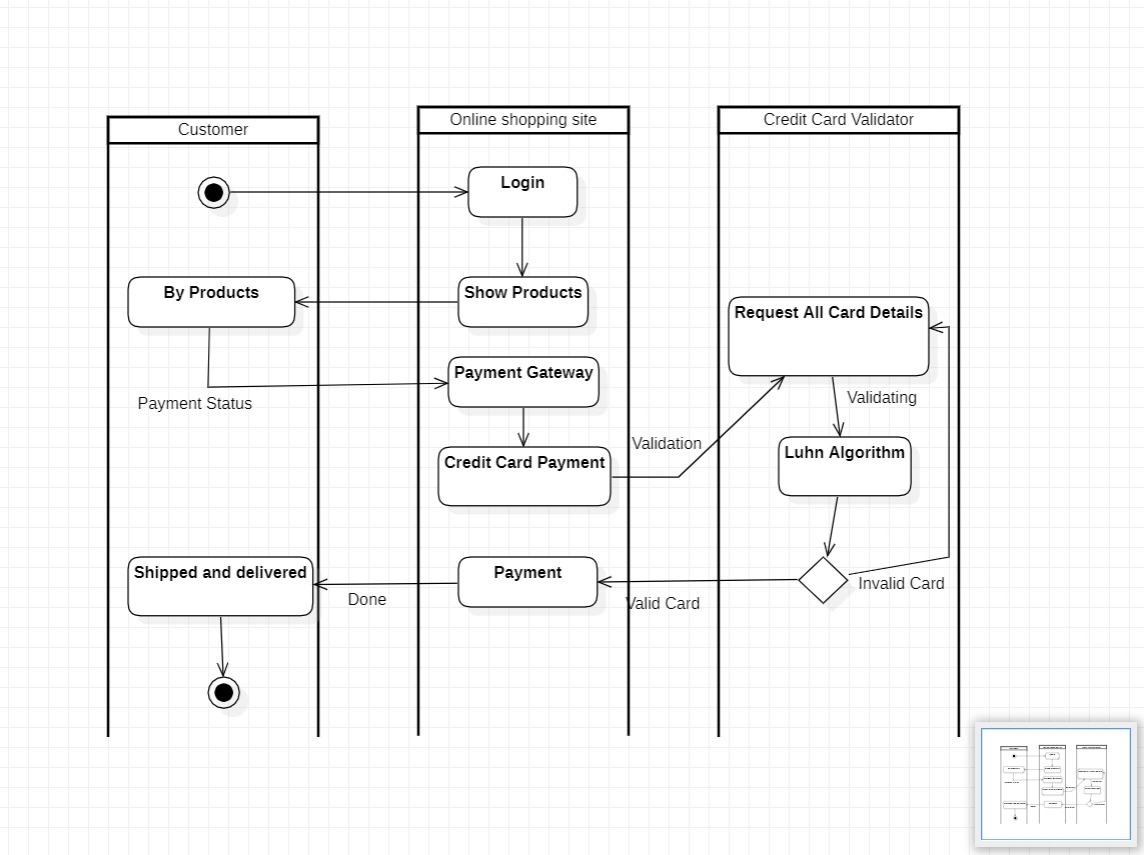


7

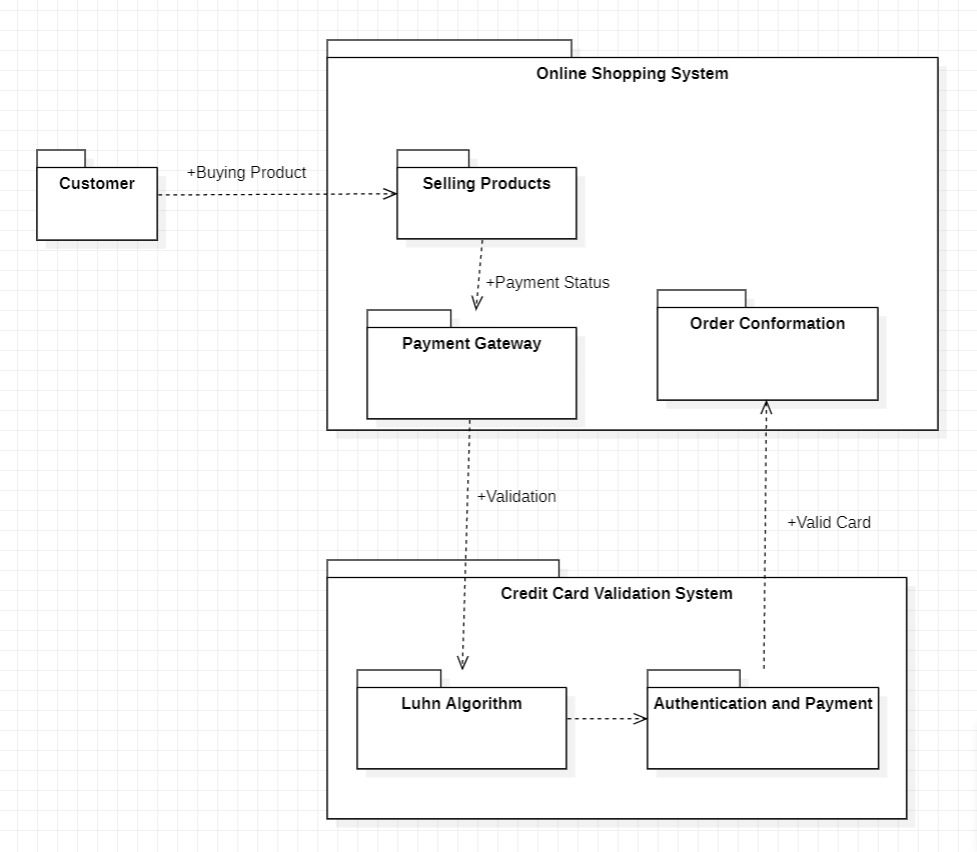
1. COLLABORATION DIAGRAM
2. STATE CHART DIAGRAM



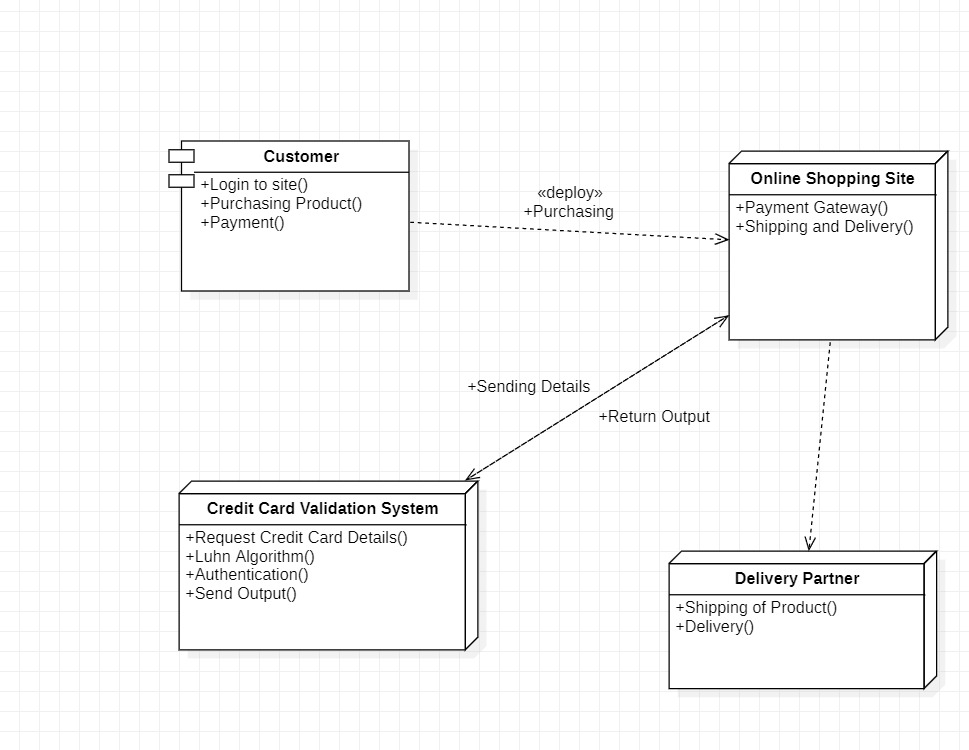
1. ACTIVITY DIAGRAM



1. PACKAGE DIAGRAM



1. COMPONENT DIAGRAM
2. DEPLOYMENT DIAGRAM



**CODE/OUTPUT SCREENSHOTS**

**#include <iostream>**

**#include <string>**

**#include <bits/stdc++.h>**

**using namespace std;**

**class CreditCard {**

**protected:**

**string cardNumber;**

**public:**

**CreditCard(string cardNumber) {**

**this->cardNumber = cardNumber;**

**}**

**virtual bool validate() {**

**return false;**

**}**

**};**

**class Visa : public CreditCard { //Inheritance is being implemented Here**

**public:**

**Visa(string cardNumber) : CreditCard(cardNumber) {}**

**bool validate() {**

**if (cardNumber.length() != 16) {**

**return false;**

**}**

**int sum = 0;**

**for (int i = 0; i < 16; i++) {**

**int digit = cardNumber[i] - '0';**

**if (i % 2 == 0) {**

**digit \*= 2;**

**if (digit > 9) {**

**digit -= 9;**

**}**

**}**

**sum += digit;**

**}**

**return sum % 10 == 0;**

**}**

**};**

**class MasterCard : public CreditCard {**

**public:**

**MasterCard(string cardNumber) : CreditCard(cardNumber) {}**

**bool validate() {**

**if (cardNumber.length() != 16 || cardNumber[0] != '5') {**

**return false;**

**}**

**int sum = 0;**

**for (int i = 0; i < 16; i++) {**

**int digit = cardNumber[i] - '0';**

**if (i % 2 == 0) {**

**digit \*= 2;**

**if (digit > 9) {**

**digit -= 9;**

**}**

**}**

**sum += digit;**

**}**

**return (sum % 10 == 0);**

**}**

**};**

**class Image{**

**public:**

**void image(){**

**cout<<" = = = = = = = = = = = = = = = = = ="<<endl;**

**cout<<" = ="<<endl;**

**cout<<" = ="<<endl;**

**cout<<" = Credit Card ="<<endl;**

**cout<<" = Validator ="<<endl;**

**cout<<" = ="<<endl;**

**cout<<" = ="<<endl;**

**cout<<" = ="<<endl;**

**cout<<" = = = = = = = = = = = = = = = = = ="<<endl;**

**}**

**};**

**class CardDetails{**

**public:**

**int Cvc;**

**string name;**

**int Expdate\_mon,Expdate\_year;**

**public:**

**void getCvc(){**

**cout<<"Enter 'Name' On Credit Card: ";**

**cin.ignore();**

**getline(cin, name);**

**bool validcvc=false;**

**while(!validcvc){**

**cout<<"Enter 3 Digit 'Cvc' Number On Credit Card: ";**

**cin>>Cvc;**

**if(Cvc>=100 && Cvc<=999){**

**validcvc=true;**

**}**

**else{**

**cout << "\033[31m"; // set text color to red**

**cout<<"Cvc is invalid"<<endl;**

**cout << "\033[0m"; //Reset colour**

**}**

**}**

**bool valid=false;**

**while(!valid){**

**cout<<"Enter 'Expiry Date' On Credit Card(Month, year): ";**

**cin>>Expdate\_mon;**

**cin>>Expdate\_year;**

**if(Expdate\_mon>=01 && Expdate\_mon<=12 && Expdate\_year>=2023){**

**valid=true;**

**}**

**else{**

**cout << "\033[31m"; // set text color to red**

**cout<<"Invalid Expiry Date"<<endl;**

**cout<<"Please Enter Valid Expiry Date"<<endl;**

**cout << "\033[0m"; //Reset colour**

**}**

**}**

**}**

**};**

**int main() {**

**string cardNumber;**

**Image card;//Image Class object creation**

**card.image();**

**main:**

**cout << "Enter your Credit card number: ";**

**cin >> cardNumber;**

**CardDetails obj;//Credit Card Details input and saving**

**obj.getCvc();**

**CreditCard\* creditCard;**

**if (cardNumber[0] == '4') {**

**creditCard = new Visa(cardNumber);**

**}**

**else if (cardNumber.substr(0, 2) == "51" || cardNumber.substr(0, 2) == "52" || cardNumber.substr(0, 2) == "53" || cardNumber.substr(0, 2) == "54" || cardNumber.substr(0, 2) == "55") {**

**creditCard = new MasterCard(cardNumber);**

**}**

**else {**

**cout << "Invalid credit card type" << endl;**

**return 1;**

**}**

**if (creditCard->validate()) {**

**cout<<"\n";**

**cout<<"Hi! "<<obj.name<<endl;**

**cout << "\033[32m"; // set text color to green**

**cout << " Your Credit Card Number is valid" << endl;**

**cout << "\033[0m"; //Reset colour**

**}**

**else {**

**cout << "\033[31m"; // set text color to red**

**cout << " Your Credit Card Number is invalid" << endl;**

**cout << "\033[0m"; // set text to default colour**

**try{**

**if(cardNumber[0]!=4 && cardNumber[0]!=5){**

**}**

**else{**

**throw runtime\_error("Error in Creditcard Number");**

**}**

**}**

**catch(runtime\_error &e){**

**cerr<<e.what()<<endl;**

**}**

**}**

**char k;**

**cout<<"-------------------->>Do you Want to Exit(Y/N):\_\_"<<endl;**

**cin>>k;**

**if(k=='Y'){**

**cout<<"Project Done By";**

**cout << "\033[32m"; // set text color to green**

**cout<<" Saahil Pradhan";**

**cout << "\033[0m"; //Reset colour**

**cout<<" And ";**

**cout << "\033[32m"; // set text color to green**

**cout<<"Lalith Reddy";**

**cout << "\033[0m"; //Reset colour**

**}**

**else{**

**goto main;//Again runs**

**}**

**delete creditCard;//Data is Secured and deleted after execution of program;**

**return 0;**

**}**

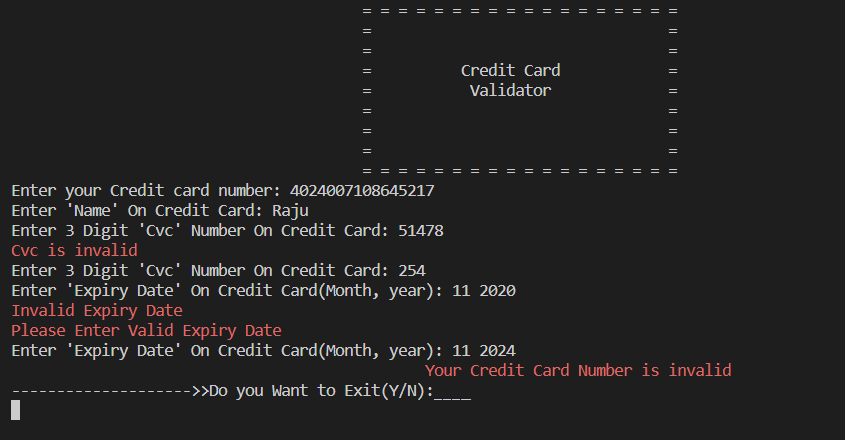
INPUTS AND OUTPUTS:

For Valid Card Details:

A screenshot of a computer

Description automatically generated with medium confidence

For Invalid Inputs:



Code Reusability:

Text

Description automatically generated

**CONCLUSION AND RESULTS:**

The credit card validation system made in C++ using OOPs concepts is a robust and efficient solution for verifying credit card numbers. With its object-oriented design, the system offers modularity, encapsulation, and flexibility, making it easy to maintain and extend. The implementation of the Luhn algorithm ensures that only valid credit card numbers are accepted, providing a secure and reliable validation process.

The system's intuitive interface and error handling mechanisms make it user-friendly and accessible to users with varying levels of technical expertise. The use of C++ provides efficient memory management and performance, making it suitable for handling large volumes of credit card transactions.

Overall, the credit card validation system made in C++ using OOPs concepts is a powerful tool that can greatly benefit financial institutions, merchants, and consumers alike by preventing fraudulent transactions and ensuring the security of credit card information.

**RESULT:**

Enter the credit card number: 1234567890123456

Validating...

Result: Invalid credit card number.

Explanation: The entered credit card number failed the Luhn algorithm check.

Enter the credit card number: 4111111111111111

Validating...

Result: Valid credit card number.

Explanation: The entered credit card number passed the Luhn algorithm check and is considered valid.

The credit card validation system provides a clear and concise validation result, along with an explanation of the reason for the validation outcome. This allows the user to quickly and easily identify whether a credit card number is valid or not, and take appropriate action as necessary.

**REFERENCES:**

In developing the program for the credit card validation system made in C++ using OOPs, we drew upon our knowledge as a language model, as well as various resources and references available on the internet. Some of the key sources of information and inspiration for this include:

* Online tutorials and documentation on C++ programming, including resources such as tutorialspoint.com and cplusplus.com.
* Articles and blog posts related to credit card validation systems, including those on sites like medium.com and dev.to.
* Research and study of the Luhn algorithm and other validation techniques commonly used in credit card processing.
* Review of examples of other credit card validation systems made in C++ and other programming languages, to identify best practices and effective implementation strategies.

Overall, our program is based on a combination of these sources and our own understanding and interpretation of the problem and potential solutions