

# Credit Card Validator

**Business Requirement Document**  
for Software Artist Assessment



**Nethra Guruge**

v1.1

2024-02-02

## Document Info

---

Author	Nethra Guruge
Date	2018-08-20
Version	1
Revision	2

## Revision History

---

Author	Date	Reason	Revision
Nethra Guruge	2022-03-07	Credit card numbers updated.	1
Amali Bandara Premaratne	2024-02-02	Reference added.	2

## Table of Contents

<b>1 Background</b>	3
1.1 Purpose	3
1.2 Objectives	3
<b>2 Functional Requirements</b>	4
2.1 Overview	4
2.2 Requirement Scope	4
2.2.1 Requirement ID #1 – UI	4
2.2.2 Requirement ID #2 – RESTful service for validation	4
2.2.3 Requirement Id #3 – Unit Testing	5
2.3 Assumptions	5
<b>3 Non-functional Requirements</b>	6
3.1 Performance Requirements	6
3.2 Safety Requirements	6
3.3 Security Requirements	6
3.4 Software Quality Attributes	6
<b>4 General</b>	7
4.1 Definitions	7
4.2 Glossary	7
<b>5 Sign Off</b>	8
5.1 Technical	8
5.2 Stakeholder	8

## 1 Background

---

### 1.1 Purpose

Sample Codes (Pvt) Ltd is a leading a leading online trading company. This document emphasis the requirement of a solution to validate credit cards entered by their users before submitting to their IPG service. The document keeps all the key functions that should be included regarding the requirement.

Currently, Sample Codes (Pvt) Ltd.'s IPG supports the following card providers.

- AmEx
- Visa
- Mastercard
- Discovery

This is a sample BRD given to asses you as a part of the onboarding process.

### 1.2 Objectives

Main objective is to measure your approach, thinking, collaboration, execution of a given task along with the coding skills.

- How you understand the requirement
- How you communicate, resolves conflicts and get help when needed.
- How you prepare and plan a task.
- Obviously, how good at coding.
- Use of **OOP** concepts.
- Implementation of SOLID principals & necessary **Design Patterns**.\*
- Production ready code.
- Full test coverage. \*
- Industry standards & best practices.\*
- Timely delivery
- Negotiations & justifications

*\* Optional for junior artists.*

## 2 Functional Requirements

---

### 2.1 Overview

#	Title
1	Decent UI to capture credit card number.
2	.Net RESTful web service for validation.
3	Unit test to cover the logics implemented.

### 2.2 Requirement Scope

#### 2.2.1 Requirement ID #1 – UI

- This must be a web application.
- Languages should be – HTML, CSS, JavaScript
- Frameworks – Bootstrap, jQuery, Vanilla JavaScript(AKA CoreJS)
- Refrain from using ASP.Net MVC Framework.

#### 2.2.2 Requirement ID #2 – RESTful service for validation

- .Net Framework (v4.8.x) is recommended. (But you can use .NET 8 as well)
- Use ASP.NET Web API
- Basic required UI validations should be identified and implemented.
- Validating the card number is suffice.
- The card validator should be able to validate the card type based on the starting digits and length of a card number.
  - Followings are the parameters that can be used.

Provider	Starting From	Length
AmEx	34 or 37	15
Visa	4	16
Mastercard	22 or 51 - 55	16
Discover	6011	16

- All of the above-mentioned providers generate numbers according to the **Luhn Algorithm**. Reference - [https://en.wikipedia.org/wiki/Luhn\\_algorithm](https://en.wikipedia.org/wiki/Luhn_algorithm)

### 2.2.3 Requirement Id #3 – Unit Testing

- Validation logic should be unit tested.
- 100% of a code coverage expected. But not mandatory.
- Use MS Test Framework.

## 2.3 Assumptions

- Assumption 01 – Card number validation is enough
- Assumptions 02 – Database is already developed. No need of building data access, just store your data in a class.

CONFIDENTIAL

## 3 Non-functional Requirements

---

Here we specify some nonfunctional constraints that the application should satisfy to be more concrete and stable.

### 3.1 Performance Requirements

#### Performance:

Checking the fact that the system must perform as every user expects. So, in every action-response of the system, there are no immediate delays. In the case of opening pages, popping error messages, and saving the settings or sessions there is a delay much below 2 seconds, in the case of searching inventory, sorting results, and computing prices there are no delays and the operation is performed in less than 3 seconds for opening, sorting, computing > 95% of the times. \*

Also, when connecting to the server the delay is based on the distance between the 2 systems and the configuration between them so there is a high probability that there will be or not a successful connection in less than 20 seconds.

*\* Working with a decent(4mbps+) internet connection assumed.*

### 3.2 Safety Requirements

#### Consistency:

Checking the fact that all clients must be attachable to one server, so there would be appropriate control of the test statistics and information.

Also, in case of a potential loss of connection between the client and the server, the client's transaction state should be rolled back. And client should restart the transaction from the application again.

### 3.3 Security Requirements

This program uses object-oriented mechanisms to protect its data passed using methods. Also, HTTP connections should be encrypted via 1024bit SSL or higher certificates. In The backend, sensitive data should be encrypted using 256-bit AES encryption.

### 3.4 Software Quality Attributes

#### Availability:

Checking that the system always has something to function and always pops up error messages in case of component failure. In that case, the error messages appear when something goes wrong to prevail over availability problems.

**Usability:**

Checking that the system is easy to handle and navigates in the most expected way with no delays. In that case, the system program reacts accordingly and transverses quickly between its states.

**Functionality:**

Checking that the system provides the right tools for editing inventory, creating transactions, and retrieving data reports. Functionality should be according to the “Minimal Clicks” approach and UI should be modern, responsive, and user-friendly.

## 4 General

---

### 4.1 Definitions

- **The Luhn algorithm** or Luhn formula, also known as the "modulus 10" or "mod 10" algorithm, named after its creator, IBM scientist Hans Peter Luhn, is a simple check digit formula used to validate a variety of identification numbers.
- **REST (representational state transfer)** is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web.

### 4.2 Glossary


<u>Abbreviation</u>	<u>Description</u>
UI	User Interface
REST	Representational State Transfer
OOP	Object Oriented Programming
SOLID	SOLID Principles



## 5 Sign Off

---

### 5.1 Technical

<b>Signature</b>	
<b>Name</b>	Pathum Bandara Premaratne
<b>Designation</b>	Chief Tech Artist
<b>Company</b>	DevArt Software Co.
<b>Contact Number</b>	+94 769 006 949
<b>Date</b>	2022-03-08

### 5.2 Stakeholder

<b>Signature</b>	
<b>Name</b>	Amali Bandara Premaratne
<b>Designation</b>	Chief Executive Artist
<b>Company</b>	DevArt Software Co.
<b>Contact Number</b>	+94 769 006 947
<b>Date</b>	2022-03-10