**Building a Payment Gateway API**

Goals and objective:

* Build a .Net API using .Net Core.
* Process a payment through the gateway API and receive a success or failure state.
* Retrieve the details of a previously made payment.

Assumptions:

1. Merchant has a token to access API.
2. Data received from merchant is identical to API data model.
3. Bank always validate the payment if information received is correct.

User Stories (High level):

1. As a merchant, I want to be able to authenticate with the API provided.
2. As a merchant, I want to be able to send a payment request to the API and receive a status code and a message indicating payment status.
3. As a merchant, I want to be able to retrieve previous payment requests using my merchant Id.

Acceptance criteria:

1. Authentication should be successful if:
   1. Client id, client secret and scope are valid.
   2. Token received is valid and not expired.
2. Payment should be successful if:
   1. Card information is correct.
   2. Concerned bank validated the payment. (Virtual)
3. Previous payment should appear in a list if:
   1. Given MID is correct.
   2. Has previous records.

Technical setup:

Code editors: Visual studio 2019 with .Net web API development kit. Alternatively, Visual studio code with .Net extension.

Test environment: IIS Express (Chrome browser), Postman v7.13.0.

.Net version: 3.1.100.

Database: MS SQL

Git for windows.

Git hub extension for Visual studio 2019.

Data structure:

|  |  |
| --- | --- |
| Name | Type |
| Id | Guid |
| Payment method | String |
| Card number | String |
| Card type | String |
| Expiry date | String |
| CVV | String |
| MID | String |
| Total amount | Decimal |
| Currency | String |
| Payment state | String |
| Created date | DateTime |

Function structure:

**Get method:**

GetpaymentDetailsMid (String mid) => {

1. Get all payment details.
2. Loop and remove all payment details not containing MID.
3. Decrypt card info and replace card number with ‘\*’.

Return payment detail list + status code.

}

**Post method:**

PostPaymentDetail (PaymentDetail paymentDetail) => {

1. Perform validation on data by sending to bank validator.
2. If data is valid:
   1. Encrypt card info.
   2. Save transaction to database.
   3. Return Status code + success message.
3. If date is invalid:
   1. Return Status code + failure message.

}

Sources:

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1>

<https://www.c-sharpcorner.com/article/aes-encryption-in-c-sharp/>

<https://stackoverflow.com/questions/31000039/mask-all-digits-except-first-6-and-last-4-digits-of-a-string-length-varies>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-3.1>

<https://code-maze.com/unit-testing-with-xunit-in-asp-net-core-mvc/>

<https://www.c-sharpcorner.com/article/crud-operations-unit-testing-in-asp-net-core-web-api-with-xunit/>

<https://stackoverflow.com/questions/43424095/how-to-unit-test-with-ilogger-in-asp-net-core>

<https://stackoverflow.com/questions/54219742/mocking-ef-core-dbcontext-and-dbset>

<https://medium.com/srp-labs/building-practical-web-applications-with-react-django-git-and-vs-code-setup-a87aecfc8b9e>

<http://www.notyourdadsit.com/blog/2018/4/3/cheatsheet-setup-github-on-visual-studio-code>