CheckOut.com API GATEWAY DESIGN DOCUMENTATION

Name: Chukwuemeka Okobi

Email: Chukwuemeka.okobi@outlook.com

Project Description

This project was built on .Net 6 using Asp.net core 6, C-Sharp programming language.

IDE: Visual Studio 2022.

Project Solutions contains the following projects.

- **Bank Simulator Library:** A simple library used to simulate the required bank functions such as card validation, payment to merchant and debit of client accounts
- **Shared Library:** This contains models, validators and extensions used by other projects
- **Entities Library:** This contains for the Gate way database entities
- **Services Library:** this contains the business logic for the Gateway functions and sample data.
- **WebApi:** the web API project which has the asp.net core API functions.
- **Test:** Containing Test

Assumptions Made

- Both shopper and Merchants use the same bank
- Card Payment requires OTP authorization to verify and process payment
- The Payment Gateway is assumed to be separate from the bank having its own database
- Merchant Requests are Authenticated using Merchant unique API Key.
- All API Keys and references are represented using GUIDs.
- The Bank Simulator does not account for currency conversion.

How to Run code

- Visual studio 2019 or 2022 and .net 6 SDK is required.
- Open the project solution using the Visual studio IDE and build the solution
- Resolve any dependencies require and install required nuget packages.
- Run the WebApi project, the swagger API page should show up on your browser.

API Documentation

Status Codes for Payment:

- “00”: successful
- “02”: pending
- “99”: failed or failed request or transaction.

Api behaviour

- Request model Validation, an invalid model returns a 400 bad request
- A failed transaction or payment return a 400 bad request response with response data
- A successful transaction or payment returns a 200 Ok response with response data.

Making Payments

Step 1: Initiate payment:

- Endpoint:- "/payments/initiate"
- Method: POST
- Request Information:

Param	value description	Required	Expected
ApiKey	Merchant unique api key	yes	in Header
cardNumber	Shopper card number	yes	payload
cvv	shopper cvv	yes	payload
expiryYear	card expiry year	yes	payload
currency	values: GBP, USD, EURO	yes	payload
amount	actual amount	yes	payload

- **Sample Request**

```
–H , "ApiKey : merchant unique api key"
Payload:{
          "cardNumber": "57247735457",
          "cvv": "567",
          "expiryYear": 2023,
          "expiryMonth": 12,
          "currency": "GBP",
          "amount": 150.45
        }
```

Sample Response

```
{
  "data": {
    "reference": "generated reference number",
    "metaData": "payment initiated, awaiting otp",
    "status": "pending"
  },
  "statusCode": "02",
  "message": "request successful"
}
```

Step 2: Validate OTP, Step 1 required

- Endpoint:- "/payments/otp"
- Method:- POST
- Request Information

Param	value description	Required	Expected
ApiKey	Merchant unique api key	yes	in Header
reference	reference from Initiate payment	yes	payload

otp	otp value	yes	payload

- Sample Request

```
-H , “ApiKey : merchant unique api key”
 Payload:{
   "reference": "f0de21da-9312-4b83-af66-814ebf890e08",
   "otp": "08352"
 }
```

- Sample Response

```
{
  "data": {
    "reference": "generated reference number",
    "metaData": "payment successful",
    "status": "successful"
  },
  "statusCode": "00",
  "message": "request successful"
}
```

Retrieve Previous Payments:

- Endpoint: “/Payments”
- Method: GET
- Query information

Param	value description	Required	Expected
ApiKey	Merchant unique api key	yes	in Header
from	Date time from	no	query
to	Date time to	no	query
status	transaction status (pending, failed, successful)	no	query
currency	GBP, USD, EURO	no	query

- Sample Request

```
-H , “ApiKey : merchant unique api key”?from=2022-06-01&to=2022-06-04&status=failed&currency=GBP
```

- Sample Response

```
{
  "data": [
          {
            "reference": "e0b0511a-0697-4147-b35f-21bb216f6d3a",
            "time": "2022-06-01T12:13:25.0147359+02:00",
            "amount": 156.49,
            "currency": "GBP",
            "status": "pending",
            "statusCode": "02",
            "comment": "payment initiated, awaiting otp",
            "cardNumber": "57*********",
```

```
            "cvv": "***",
            "cardExpiry": "*/****"
          },
    ],
    "statusCode": "00",
    "message": "success"
}
```


Testing API

Initiate Payment

- Merchant ApiKey – "8d8689a1-0157-4b4d-a468-4ba925e07a17"
- Valid Payment


```
Payload:{
  "cardNumber": "57247735457",
  "cvv": "567",
  "expiryYear": 2023,
  "expiryMonth": 12,
  "currency": "GBP",
  "amount": 150.45
}
```


- Invalid Card


```
Payload: {
  "cardNumber": "57247735457",
  "cvv": "566",
  "expiryYear": 2023,
  "expiryMonth": 12,
  "currency": "GBP",
  "amount": 28.12
}
```


- Expired Card


```
Payload: {
  "cardNumber": "52893673695",
  "cvv": "093",
  "expiryYear": 2022,
  "expiryMonth": 4,
  "currency": "GBP",
  "amount": 49.49
}
```


- InActive Card


```
Payload: {
  "cardNumber": "52952772755",
  "cvv": "026",
  "expiryYear": 2022,
  "expiryMonth": 9,
  "currency": "GBP",
  "amount": 219.79
}
```


- Insufficient Funds


```
Payload: {
  "cardNumber": "5424725289",
  "cvv": "825",
```

```
  "expiryYear": 2022,
  "expiryMonth": 7,
  "currency": "GBP",
  "amount": 982.99
}
```

Validate OTP

- Merchant ApiKey – "1fa0ce55-f16a-4bea-88c1-83f93fad640b"
- Valid OTP

  Payload: {
  ```
  "reference": "f0de21da-9312-4b83-af66-814ebf890e08",
  "otp": "08352"
  }
  ```

- Invalid OTP

  Payload: {
  ```
  "reference": "5747a3ca-3d20-4dc0-b5b9-4e5fee121fe0",
  "otp": "986104"
  }
  ```

Areas for Improvement

- Unique Merchant API Keys can be generated using a more secure and meaningful way rather than just using GUIDs,
- Bank Simulator and Gateway responses should have more details comments or meta data to give or information on each transaction.
- The system might benefit from using a KeyStore to generate unique payment references
- Merchants can be allowed to provide their own unique Payment references.
- OTP verification should have a time limit.
- Include currency conversion in the Bank Simulator.

Cloud Technologies

- **AWS RDS for the database**: SQL seems appropriate to store the transaction data as this data is more structured and defined. And using AWS manage RDS , Aurora or mysql or postgres is appropriate as AWS helps with most of its management.

- **AWS Elastic Beanstalk:** this will help in the hosting of the application and help with its elasticity to manage large and small requests to the servers, as opposed to using an EC2 instance this requires some manual management, configuring autoscaling and running load balancers.