Coding exercise:

Write a WebAPI with only 1 method called “ProcessPayment” that receives a request like this

- CreditCardNumber (mandatory, string, it should be a valid CCN)
- CardHolder: (mandatory, string)
- ExpirationDate (mandatory, DateTime, it cannot be in the past)
- SecurityCode (optional, string, 3 digits)
- Amount (mandatoy decimal, positive amount)

Write a Payment domain/entity with the same properties as the request and a second entity to store the payment state (pending, processed, failed). Use Entity framework code first approach, write entity configurations and generate the migrations.

The response of this method should be 1 of the followings based on

- Payment is processed: 200 OK
- The request is invalid: 400 bad request
- Any error: 500 internal server error
- The request should be validated before processed.

The payment could be processed using different payment providers (external services) called:

- IExpensivePaymentGateway or
- ICheapPaymentGateway.

The payment gateway that should be used to process each payment follows the next set of business rules:

a) If the amount to be paid is less than £20, use ICheapPaymentGateway.
b) If the amount to be paid is £21-500, use IExpensivePaymentGateway if available. Otherwise, retry only once with ICheapPaymentGateway.
c) If the amount is > £500, try only PremiumPaymentService and retry up to 3 times in case payment does not get processed
d) Store/update the payment and payment state entities created previously once the processing is completed.

Bonus:

- Use repository/unit of work patterns
- Use eager loading for all entities

Recommendations :

- The classes should be written in such a way that they are easy to test.
- Write enough unit tests to cover the developed code (at least 1).
- Use SOLID principles.
- Use any IOC you think is best.
- Use Entity Framework.

Optional:

- Use AutoMapper or other mapping tools of your choice
- Use .Net Core to develop the WebAPI and Entity Framework Core.