Test Challenge - Python

1. Requirements

This challenge must accomplish the next goals:

1.1. Create a rest API using Python (use 3.9 or 3.10 version). The use of Flask is prohibited; you can use other microframeworks or pure Python for this.
1.2. Document the API collection using Swagger
1.3. Patterns use (ie. repository pattern, cqrs,etc) make clear what you are choosing and why.
1.4. Apply SOLID and Clean Code principles.
1.5. The solution should be implemented by using TDD (Test Driven Development) Tests are mandatory.
1.6. Each request should use good validation patterns and HTTP Status Code per response.
1.7. You should need to structure the project using N-layers.
1.8. A README.md file should be included with the contain description (patterns, architecture, setup and startup steps, etc) consider include any information that could help
1.9. Import the project code to a public repository on Github

2. Challenge

Consider the use of a web system that manages different product inventory. You are asked to create a API service that support the next features:

2.1. Insert(POST), Update(PUT) and GetbyId(GET) for the products.
Maintain a log file (plaintext) of the elapsed time used per request

2.2. Maintain a cache (with a minimum of 5 min) for a dictionary that contains the product status, here a table with the values of this dictionary:

Status(key)	**StatusName(value)**
1	Active
0	Inactive

You can use a standard Cache, Lazy Cache or any kind of cache that you considered fits well.

2.3. Record the product information locally using any kind of local storage for the data persistence. The mandatory fields/attributes to use are:

- ProductId
- Name
- Status // 0 or 1
- Stock,
- Description
- Price

You are free to add additional fields for the products

2.4. The GetById method should return a product response with the next fields:

Fields	**Observations**
ProductId	
Name	
StatusName	This field will be obtained from the cache created on 2.2 based on "Status" field

Stock	
Description	
Price	
Discount	Discount Percentage [0-100] obtained by using an external/third party service based on the ProductId. This service could be https://mockapi.io/ or others that help you to obtain random values.
FinalPrice	Price * (100 - Discount) / 100
…	Any additional fields that you could consider relevant

3. Recommendations

Be creative
Comment your code
Have fun