## Task 2: [REFACTORING]

Assumption/Decision Made

- Assumption/Decision:

* Creating 2 new Services ”Insurance Service & ProductDataService”.
* Remove Redundant Code.
* Modify product API calls.
* Use HttpClientFacctory instead of httpClient.
* Use Polly.

- Reason:

* Apply Separation of Concern where each Service will be responsible for specific task.
* Increase API performance by optimizing calls.
* Remove /product\_types API call and call /product\_types/{id} instead as there is no need to get all the product types and loop through them when you can get the producttype details directly.
* Use HttPClientFactory to enhance resource management and for provides better Configurability and testing.
* Use Polly retry mechanism to ensure the application is self-healing and resilient in case of failures & transient issues.

- PR : [Task-2 refacor: Refactor CalculateInsurance API by Mohamed-Mohamed-fahmy · Pull Request #2 · Mohamed-Mohamed-fahmy/CoolBlue-Task (github.com)](https://github.com/Mohamed-Mohamed-fahmy/CoolBlue-Task/pull/2/files)

## Task 3: [FEATURE 1]

Assumption/Decision Made

- Assumption/Decision:

* Create a new API for Calculating Order insurance.

- Reason:

* To calculating Insurance for multiple ProductIds in an order.

-PR : [Task-3 feat: Implement CalculateOrderInsurance API by Mohamed-Mohamed-fahmy · Pull Request #3 · Mohamed-Mohamed-fahmy/CoolBlue-Task (github.com)](https://github.com/Mohamed-Mohamed-fahmy/CoolBlue-Task/pull/3/files)

## Task 4: [FEATURE 2]

Assumption/Decision Made

- Assumption/Decision:

* Add a Boolean to check whether there are any Digital Cameras in the order or not.

- Reason:

* Insurance cost is added once if there is one or more digital cameras.

-PR : [Task-4 feat: Add 500$ if order has one or more digital Camera by Mohamed-Mohamed-fahmy · Pull Request #4 · Mohamed-Mohamed-fahmy/CoolBlue-Task (github.com)](https://github.com/Mohamed-Mohamed-fahmy/CoolBlue-Task/pull/4/files)

## Task 5: [FEATURE 3]

Assumption/Decision Made

- Assumption/Decision:

* Create new Service & Controller.
* Use SQLite database.
* Implement GlobalExceptionHandler Middleware.
* Add ‘Version’ Column to ‘ProductTypeSurcharge’ Table.

- Reason:

* Create new Service & Controller for Uploading Surcharge.
* Use SQLite as a lightweight, efficient database solution.
* Implement GlobalExceptionHandler Middleware to catch and handle exceptions across the application.
* Add ‘Version’ Column to handle Concurrency Conflicts as new endpoint is going to be used simultaneously by multiple users.

-PR : [Feature/task 5 by Mohamed-Mohamed-fahmy · Pull Request #5 · Mohamed-Mohamed-fahmy/CoolBlue-Task (github.com)](https://github.com/Mohamed-Mohamed-fahmy/CoolBlue-Task/pull/5)