# Goal

The goal of this test plan is to provide stakeholders with an upfront view of the scope of testing, risks and limitations, a list of comprehensive test cases, and feedback for potential changes to make the API more consistent.

# Scope of testing

## In Scope

### Functional Testing

* + GET /product: Retrieve a list of all products.
  + GET /product/{id}: Retrieve a specific product by id.
  + POST /product: Create a new product with valid inputs.
  + PUT /product/{id}: Update an existing product.
  + DELETE /product/{id}: Delete a specific product by id.

### Data Validation

Validate the structure and correctness of input and output data:

* + Ensure that name, description, and price in products adhere to the schema.

### Schema Validation

* + Ensure that responses adhere to the API specification, including data types, patterns, and constraints (ex. price must not be negative, name should match the ^[A-Z]+$ regex, description must be 30 characters or less).

### Boundary Testing

* + Maximum and minimum lengths for description.
  + Price constraints (ex, 0 for price is valid, -1 is invalid).

### Error Handling

Validate how the API handles incorrect requests, including:

* + Sending malformed or missing data.
  + Deleting or updating non-existent products (404 errors).
  + Sending invalid data formats (e.g., price: -10 or name: "hat" in lowercase).

### Status Codes

Verify correct status codes are returned for each action:

200 OK for successful operations.

201 Created for successful creation of products/orders.

400 for invalid input.

404 Not Found for non-existent resources.

## Out of Scope

### User Interface Testing

* + There is no UI associated in this API challenge.

### Performance Testing

* + This test plan is majorly functional testing focused.

### Database Testing

* + There is no database to validate against in this API challenge.

### Security Testing

* + There is no authentication or other connected systems associated with this API challenge.

# Risks and Limitations

## Risks

| Risk | Likelihood | Mitigation | Impact |
| --- | --- | --- | --- |
| Time Limitation | High | Prioritize Testing | High |
| Missing Test Cases | Low | Test Case Review | High |
| Environment Downtime | Low | Notify DEV team | Medium |

## Limitations

* The Order API creation is in progress.
  + We are unable to validate our tests against working endpoints.
* There is no UI to view the data.
  + We can only see the data returned by the API. We do not know what it looks like to the end user.
* Security related details have been left out.
  + Therefore, we have not included authentication methods in our test plan.

# Test Cases

Endpoint specific test cases can be found in each ../e2e/product-api spec file of the repository. More detailed test case related comments can be found in the methods under ../support/.

# Results/Feedback and Considerations

The Order API would benefit from placing meaningful messages between status codes. For example, GET /product/{id} returns 404 Product not found while DELETE /product/{id} returns 404 Visitor Not Found. `Visitor` doesn’t make sense in the context of product. Both messages can be Product Not Found.

Another suggestion is to ensure error status codes and messages are present for each call. For example, GET /product/{id}returns 404 Product not found while PUT /product/{id} does not have a message for the error state.

My third recommendation would be to make status code message consistent across the board. For example, most calls return 200 Successful operation while PUT /product/{id}returns 200 Successfully updated product. It should be made to match the other calls.

Lastly, I would recommend using descriptive messages for every status code. For example, POST /product returns 201 Created. POST /product should return something along the lines of, 201 Successfully created the product.