BDD and Use Case Final Release

# Backend User Story BDD

## Feature: Product Search Microservice API

### Scenario 1: Search API returns results

GIVEN the client invokes the Search API with a valid keyword  
WHEN the backend processes the query  
THEN the API should return http status code 200  
AND the system should return the list of matching products

### Scenario 2: Filter API applies selected filters

GIVEN the client invokes the Filter API with valid filter parameters  
WHEN the backend applies the filters to the dataset  
THEN the API should return http status code 200  
AND the system should return only products matching the filters

### Scenario 3: Handle missing required field

GIVEN the client invokes the Search API without required parameters  
WHEN the backend validates the request  
THEN the API should return http status code 400  
AND the system should return an error message for missing input

### Scenario 4: Handle invalid data format

GIVEN the client invokes the Filter API with incorrect data type  
WHEN the backend attempts to parse the request  
THEN the API should return http status code 400  
AND the system should return an error message for invalid format

### Scenario 5: Inventory service integration

GIVEN the client invokes the Search API for a product  
WHEN the backend checks inventory status  
THEN the API should return http status code 200  
AND the system should mark products as in-stock or out-of-stock

### Scenario 6: Database not reachable

GIVEN the client invokes the Search API  
WHEN the backend cannot connect to the database  
THEN the API should return http status code 503  
AND the system should return an error message for service unavailable

# Use Case Document

## Use Case ID: █████

## Product Search and Filtering

## Level: █████

### Overview

A customer wishes to search and filter products on the e-commerce platform using backend microservice APIs.

### Preconditions

Catalog and inventory data are ingested into MySQL.  
Microservice APIs are deployed and accessible via gateway.

### Trigger

The customer initiates a product search or applies filters from the front-end interface.

### Main Success Path

1. The system receives the search query from the front end.
2. The Search API retrieves relevant products from the catalog database.
3. The Filter API applies filters (price, brand, color, rating).
4. The Inventory service updates stock availability.
5. The system returns filtered product results to the customer.
6. The system logs the search and filter events for analytics.

### Alternative Paths

* A01: If no products match, the system returns to zero results and may provide recommendations.
* A02: If the inventory service is unreachable, the system returns cached stock data with a warning.

### Error Paths

* E01: Missing required field → API returns 400 with error message.
* E02: Invalid data format → API returns 400 with error message.
* E03: No matching records found → API returns 404.
* E04: Database unavailable → API returns 503 service unavailable.

### Business Rules

All API requests must be authenticated and validated.  
Filters must map to valid product attributes.  
Inventory data must be refreshed within ≤60 seconds.

### Start State

Customer is on the e-commerce search page.

### End State

Customer receives search results that reflect real-time filters and availability.