

# Testing plan 2024

---

JUNE 12

---

HBO-ICT

Authored by: Joeri Harreman





Contents

**Test Plan ..... 3**

**User Stories: ..... 3**

**System Tests per User Story ..... 4**

**Unit Tests per User Story ..... 4**

**Applying the V-Model ..... 5**

**Evaluation..... 6**

**Screenshots ..... 7**

---

# Test Plan

## User Stories:

### ***Viewing Enappsys Data:***

- As a user, I want to view the Enappsys data on the detailed page, so that I can see the energy prices for the given time period.

### ***Paths:***

- Happy Path: The data is successfully retrieved and displayed.
- Unhappy Path: The data cannot be retrieved due to an error.

### ***Data classification***

- As a user, I want the system to classify the day as 'Apple Pie' or 'Cheesecake' based on the energy prices, so that I can easily understand the day's energy price category.

### ***Paths:***

- Happy Path: The data is classified correctly based on the given price ranges.
- Unhappy Path: The data classification fails due to incorrect data format.

---

## System Tests per User Story

### ***User Story 1: Viewing Enappsys Data***

- **Scenario 1 (Happy Path):** The data is retrieved successfully from the external API and displayed on the detailed page.
- **Scenario 2 (Unhappy Path):** The data cannot be retrieved due to an external API error, and an error message is displayed.

### ***User Story 2: Data Classification***

- **Scenario 1 (Happy Path):** The data is correctly classified as 'Apple Pie' or 'Cheesecake' based on the price values.
- **Scenario 2 (Unhappy Path):** The data contains invalid values, leading to a failure in classification.

## Unit Tests per User Story

### ***User Story 1: Viewing Enappsys Data***

- **Functionality:** Retrieve and process Enappsys data from the external API.
- Mock the external API call and check if the data is correctly retrieved and processed.
- Ensure the response contains the correct data structure.

### ***User Story 2: Data Classification***

- **Functionality:** Classify the day based on energy prices.
- Verify the classification logic to ensure it correctly categorizes 'Apple Pie' and 'Cheesecake' days.

---

## Applying the V-Model

### ***User Story 1: Viewing Enappsys Data***

- **Requirement Gathering:** Users need to view energy prices on the detailed page.
- **System Analysis:** The system must fetch data from an external API and display it on a web page.
- **Software Design:** Design the EnappsysController to handle data fetching and processing.
- **Module Design:** Implement functions to make API calls and process the response data.
- **Coding:** Implement the getEnappsysData method in the EnappsysController.

### ***Testing Phases:***

- **Unit Testing:** Test individual functions like API calls and data processing using mock data.
- **Integration Testing:** Test the interaction between the EnappsysController and external API.
- **System Testing:** Ensure the entire system retrieves and displays data correctly on the detailed page.
- **Acceptance Testing:** Validate that the detailed page meets user requirements for viewing energy prices.

### ***User Story 2: Data Classification***

- **Requirement Gathering:** Users need the system to classify days based on energy prices.
- **System Analysis:** The system must analyze price data and determine the category of the day.
- **Software Design:** Design the classification logic within the EnappsysController.
- **Module Design:** Implement functions to classify days based on price values.
- **Coding:** Implement the classification logic in the getEnappsysData method.

### ***Testing Phases:***

- **Unit Testing:** Test the classification function with different price inputs.
- **Integration Testing:** Ensure the classification logic works correctly with the processed data.
- **System Testing:** Validate that the system classifies days correctly when integrated with real data.
- **Acceptance Testing:** Verify that the classification meets user expectations and is displayed correctly.

---

## Evaluation

**1. Describe a possible mistake/error that can be detected by your test(s):**

- **Error Detection:** The tests can detect if the external API fails to provide data or if the data structure returned by the API changes.

**2. Describe a possible mistake/error that cannot be detected by your test(s):**

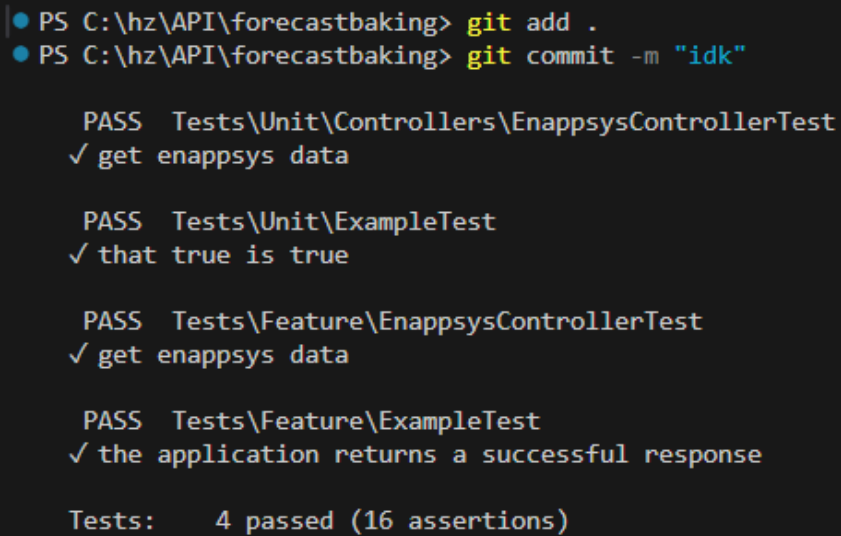
- **Undetected Errors:** The tests might not detect issues related to the actual content accuracy of the data.

**3. To what extent can you conclude that "everything works correctly"? Provide arguments!**

- **Conclusion:** The tests provide a high level of confidence that the system works as expected under normal and error conditions. They verify that data retrieval, processing, and classification work correctly, and the user interface displays the data accurately. However, the tests assume the external API provides accurate data and do not cover unexpected data anomalies or security issues related to external data sources.

---

## Screenshots



```
PS C:\hz\API\forecastbaking> git add .
PS C:\hz\API\forecastbaking> git commit -m "idk"

PASS Tests\Unit\Controllers\EnappsysControllerTest
✓ get enappsys data

PASS Tests\Unit\ExampleTest
✓ that true is true

PASS Tests\Feature\EnappsysControllerTest
✓ get enappsys data

PASS Tests\Feature\ExampleTest
✓ the application returns a successful response

Tests: 4 passed (16 assertions)
```

*Figure 1, Auto tests when committing.*