Functional Test Happy Herbivore

Introduction

This is a functional testing document for the self-service system of Happy Herbivore, a veggie/vegetarian restaurant, located in Hoog-Catharijne, Utrecht. This document provides tests that the website must pass, to make sure that all features work as expected. The aspects that will be tested:

- Adding products (including quantities) to the cart
- Filtering, based on category and diet type
- Reviewing your order, removing items and adjusting quantities
- The payment and confirmation process (payment is simulated)

Test Environment

There are a few prerequisites for setting up the testing environment.

Hardware requirements:

- PC or laptop, running Windows 10/11. Must have terminal access
- A screen with an aspect ratio of 1080px (width) x 1920px (height)

Software requirements

- Node.js, preferably the latest version (currently v22.x.x)
- Git (with access to the repository)
- An API key, for connecting to the server

Setting up the test environment

To get more info on setting up the testing environment, view the README.md file in the GitHub repository.

Test scenarios (kiosk screen)

To build the front-end for launching it as a standalone application, you will need to run the following command in the /react directory:

npm run build

When it's done building, you can launch the app by running:

npm run start

Menu and Navigation

1. Click on one of the two white buttons located at the bottom of the screen.

Expected result: the menu will open.

Actual result: after clicking on one of the buttons, a blank page showed up. The language in the localStorage was set to 'en-US' instead of 'en', which lead to a 400-error response after calling the API.

(Pass/Fail)

2. Click on one of the categories in the sidebar to change the selected category and only show items within that category.

Expected result: the orange selected category indicator should slide towards the selected category. Only products from that category will be visible.

Actual result: The orange indicator slid smoothly from the previously selected category to the newly selected category. Only products from that category showed up.

(Pass/Fail)

3. Deselect the vegan or veggie selector, located at the top-right button of the screen.

Expected result: the button will become transparent instead of green, and the X turns into a plus. Only products that fit within the diet type will be shown.

Actual result: The button became transparent instead of green, the X rotated 45 degrees and turned into a plus, and the products that didn't fit within that diet type disappeared.

(Pass/Fail)

4. Deselect both the vegan and the vegetarian options.

Expected result: both buttons will be transparent and contain a plus instead of an X. All the products will be shown.

Actual result: Both buttons became transparent and container a plus instead of an X. All products were showing.

(Pass/Fail)

Order Placement and Payment

1. Click on a product you want to add to your basket

Expected result: a popup will slide from the bottom of the screen into the center of the screen. The popup will contain the following items:

- Full title
- Description
- Image
- Chart that shows the calories of the product compared to the recommended daily intake
- Quantity selector
- Total price
- Add to cart button

Actual result: The popup animated from the bottom of the screen to the center. And contained all the items listed above.

(Pass/Fail)

2. Select a quantity using the plus and minus buttons

Expected result: the price changes when changing the quantity. You can't add less than 1 product to your basket.

Actual result: The price changed when adjusting the quantity. When pressing the minus button when the quantity is already 1, it stays at 1.

(Pass/Fail)

3. Add the product to the basket by clicking on the 'add product to basket' button

Expected result: the popup with the product details disappears. A popup will appear at the bottom of the screen, notifying the customer that the product has been added to the basket. The total price (bottom of the sidebar) will update.

Actual result: The product details popup disappeared, and a toast notification showed up at the bottom of the screen that said 'Product(s) added to basket!'. The total price at the bottom-left of the screen got updated.

(Pass/Fail)

4. View the order summary by clicking on the green button at the bottom-left of the screen

Expected result: A screen listing the items of your order will appear. Each item has buttons that allow you to change quantities. At the bottom of the screen, there's a chart that displays the total calories of your order, compared to the recommended daily intake.

Actual result: The screen listing the selected items appeared. Every item has a title, image, total price, price per piece and a quantity selector. The calorie chart shows the total calories.

(Pass/Fail)

5. Change the quantity of an item by clicking on the plus or minus button

Expected result: The quantity of the product will increase or decrease. When the quantity is zero, the product will be removed from the basket. The total price and calories will change when changing quantities.

Actual result: The quantity, price, total price and total calories changed when clicking on the quantity selector buttons. Items get removed from the cart when quantity is zero.

(Pass/Fail)

6. Proceed to the checkout page by clicking on the 'proceed to checkout' button

Expected result: a popup will appear that displays the price and instructs the user to follow the instructions on the terminal. The order process gets simulated, and the confirmation screen shows after 5 seconds. The user can cancel the payment by clicking on the cross at the top-left corner of the popup (view next test scenario).

Actual result: The popup with the total price appeared.

(Pass/Fail)

7. Cancel payment by clicking on the cross or outside of the popup window

Expected result: the simulated five-second timeout gets cancelled, and the payment popup disappears.

Actual result: The payment got cancelled, both when clicking outside of the popup, or when clicking on the cross. I could change the order after cancelling.

(Pass/Fail)

8. Click on the 'proceed to checkout' again. Let the simulated payment timeout run out.

Expected result: after the timeout has run out, it sends the order (including products) to the server and the server responds with an order id. A confirmation page will appear with the order id.

Actual results: A confirmation page appeared. The order id and a link redirecting to the idle screen appeared. After ten seconds, the user automatically got redirected to the idle screen.

(Pass/Fail)

Test Scenarios (Kiosk Controller)

Receiving Orders

1. Open the dashboard

Expected result: a page with statistics shows up. At the top, you will see the revenue and the order count from the past month. Right below, you will see a table displaying orders that aren't completed/picked up yet. At the bottom of the page there's a bar chart showing the most-ordered products.

Actual results: A loading indicator will show before the stats have been loaded. Once all stats have been loaded, they are displayed. Currently outstanding orders pop up.

(Pass/Fail)

2. Place an order from the kiosk screen and see it popping up in real-time.

Expected result: the new order will pop up under 'Recent Orders'. There will be a badge in the top-left corner that displays if the order type is 'Take out' or 'Dine in'. At the top-right corner, there will be a badge displaying the status of the entire order (for instance 'pending' or 'in progress'). You can expand the order to view the individual products. All stats will be updated.

Actual results: The new order pops up with an animation. The status in the top-right corner is 'Placed and Paid'. You can expand the order by clicking on the down-arrow. This will list all products and quantities of the order.

(Pass/Fail)

3. Mark products as 'preparing' by expanding the order and clicking on the orange button.

Expected results: if the order status is still 'pending', it will change to 'in-progress'. The orange button will be greyed out on that product.

Actual result: The order status changes from 'Placed and Paid' to 'Preparing...'. The orange prepare button is greyed out and can't be clicked again.

(Pass/Fail)

4. Mark all products as 'completed' by clicking on all the green buttons inside of the expanded order.

Expected results: the green and orange buttons will disappear on the completed product. When all items are completed, the entire order will be marked as 'Ready for Pickup'. A red button will appear in the top-right corner of the order, that says 'Mark as Picked up'.

Actual result: Both the 'prepare' and the 'complete' button disappear. The status changed to 'Ready for Pickup' and a red button appeared at the top-right corner, that says 'Mark as Picked up'.

(Pass/Fail)

5. Mark the order as picked up by clicking on the 'Mark as Picked up' button.

Expected results: The order will disappear from the list.

Actual result: The order disappeared from the list.

(Pass/Fail)

Findings

- When the language isn't set in the localStorage, it will default to 'en-US', instead of 'en'. This results in a 400 – bad request error when making the API calls because the language is invalid.
- A layout shift can happen on some very slow machines when the images in the menu are loading too slow.

Improvement proposals

- Set the language to 'en' when no language is present in the localStorage (within the i18n.tsx file).
- Set an aspect ratio on the image, so it will still take up the correct amount of space when the image is loading.