## **Functional Test**

**Purpose:**

* Verify that the core functionalities of the web application work correctly according to the specifications.
* Check that all required functions are present and functioning properly.

**Document Content:**

1. **Introduction:**
   * Test if all items are displayed correctly without fail
   * Check if switching between categories is smooth
   * Check info card’s smoothness and consistency on showing the product’s correct information
2. **Test Environment:**
   * Portrait rotated monitor that fits the provided aspect ratio (9:16), using browser developer tools to simulate the provided aspect ratio
   * Test if the items that has been added at the products page won’t desync in the cart page
   * Verify the deletion and duplication functionality of the items within in the cart page
3. **Test Scenarios:**
   * Adding & Removing functionality testing
     + Steps
       1. Select eat in
       2. Select 2 random items from each category
       3. Remove 3 items
       4. Duplicate 3 items
     + Expected results:  
       When adding it’ll show the items correctly  
       When duplicating a item it’ll correctly show the duplicated item at the bottom
     + Actual results:  
       It went how we predicted  
       There were no problems nor bugs that has surfaced
     + Status:  
       Checked
4. **Findings:**
   * Summary of the test results.

Everything went according to plan, we tested, and everything was good.

* + List of any errors or deviations from the expected results.

No Errors or deviations.

## **Document 2: Compatibility Test**

**Purpose:**

* Verify that the web application works correctly in different browsers, operating systems, and devices.
* Ensure a consistent user experience across different platforms.

**Document Content:**

1. **Introduction:**
   * Description of the compatibility test and its purpose.
   * Overview of the tested platforms.
2. **Test Environment:**
   * Chrome
   * Windows,linux
   * Kiosk
   * We used 1080x1920
3. **Test Scenarios:**
   * We used the 1080x1920 interface, and a screen side ways to test. We also went trough the use friendliness guidelines and made something for little people, so they can reach anything.

Going trough a fake order to see how a real customer would order something and if everything works correctly, behaves correctly.

* + Examples of test scenarios:
    - A little person walks up to the kiosk and cant reach, but he sees the option for a smaller person, and clicks the button for it. He can now reach everything.
    - We tested in a bigger screen in 1080x1920 to see if it works for a real kiosk. That way we knew all the front end looked good.

1. **Findings:**
   * Summary of the compatibility test results.

* Little person can reach
* 1080x1920 on big screen is good and works everything works.
* All animation work after testing.
* Info needed to be displayed was displayed.

## **Document 3: Performance Test**

**Purpose:**

* Measure the performance of the web application under different loads.
* Identify performance bottlenecks and optimization opportunities.

**Document Content:**

1. **Introduction:**
   * Description of the performance test and its purpose.
     + Loading time upon loading the products
   * Overview of the tested performance aspects.
     + Improved loading by caching the api data
2. **Test Environment:**
   * Description of the server configuration.
     + Server has PHP loaded
     + Phpmyadmin is installed
     + Cross platform is enabled
   * Details about the performance testing tools used (JMeter, LoadRunner, etc.).
     + Hyper-v
     + Task manager
     + Dev tools (Performance local metrics)
3. **Test Scenarios:**
   * Description of the performance tests performed, including:
     + Load time tests

First init: 1200 ms   
Other inits: 100 - 300 ms

* + - Stress tests
      1. Used Postman to put pressure on the server
    - Scalability tests
  + Describe how many requests per second that are simulated, and for how long.
    - When simulating a real scenario with 2 kiosks, 2 per every 10 minutes
    - This also includes the stress test, due to the api caching it’ll still request every 10 minutes

1. **Findings:**
   * Because we stored everything by caching, everything loaded really fast after being loaded 1 time.
   * You can see this with the videos, and testing we showed.