Unit Test Plan

Module #:	Application pour restaurant local.:
Tester: Amassana	Test Manager: Haranga

Module Overview

The Basket Management module allows users to add items to their basket, remove items, and calculate the total price of the items in their basket. It is essential for managing user selections and computing the final cost before checkout.

Module Inputs

- Item to be added (name, price)
- Item name to be removed

Module Outputs

- Updated list of items in the basket
- Total price of items in the basket

Logic Flow

```
graph TD
  A[Start] --> B[Add Item]
  A --> C[Remove Item]
  B --> D[Update Basket]
  C --> D
  D --> E[Calculate Total Price]
  E --> F[End]
```

Test Data

- Add a single item to the basket
- Add multiple items to the basket
- Remove an item from the basket
- Remove an item that doesn't exist in the basket
- Calculate total price with no items
- Calculate total price with multiple items

Positive Test Cases:

1. **Add Item to Basket**
   - **Test:** Add an item {name: "Burger", price: 10} to an empty basket.
   - **Expected Outcome:** Basket contains one item, "Burger" with a price of 10.
2. **Add Multiple Items to Basket**

- **Test:** Add items {name: "Burger", price: 10} and {name: "Fries", price: 5}.
- **Expected Outcome:** Basket contains two items: "Burger" and "Fries".

3. **Remove Item from Basket**
   - **Test:** Remove "Burger" from the basket containing {name: "Burger", price: 10} and {name: "Fries", price: 5}.
   - **Expected Outcome:** Basket contains one item, "Fries".
4. **Calculate Total Price with Multiple Items**
   - **Test:** Calculate total price for items {name: "Burger", price: 10} and {name: "Fries", price: 5}.
   - **Expected Outcome:** Total price is 15.

Number each test case. Indicate the test to be performed and expected outcome

Negative Test Cases:

- **Remove Non-Existent Item**
  - **Test:** Remove "Pizza" from a basket containing {name: "Burger", price: 10} and {name: "Fries", price: 5}.
  - **Expected Outcome:** Basket remains unchanged with "Burger" and "Fries".
- **Calculate Total Price with No Items**
  - **Test:** Calculate total price for an empty basket.
  - **Expected Outcome:** Total price is 0.

Listin valid data selections

Interface Modules

Identify interfacing modules indicating the nature of the interface:

- **Output Data:** Updated basket items and total price
- **Data Input:** Item details (name and price)
- **Internal Program Interface:** Basket state management within the app
- **External Program Interface:** None

Test Tools

Identify software used for unit testing.
Identify names and locations of software for future regression testing.

- **Unit Testing Framework:** Vi-test
- **Future Regression Testing Software:** Vi-test (same as unit testing), located in the project repository under the `tests` directory.

Archive Plan
Specify location of archived data. Define procedures required to obtain access to this data. **Location of Archived Data:** • **Location:** `project-root/tests/archive` • **Access Procedure:** Access granted to team members through repository permissions. Data can be retrieved via the version control system.
Updates
Identify how unit test plan will be updated. **Procedure:** The unit test plan will be updated with each sprint. Changes will be reviewed by the Test Manager (Haranga) and approved by the team. Updated plans will be documented and stored in the repository under `project-root/tests/plans`.