CODE-LOGIK
Crafting Cutting-Edge Software

THE WILLOW TREE OMS

CS4233 CAPSTONE PROJECT 2024

TESTING

PREPARED BY:

Mark Sarasua, Jr.
Application Development

Code-Logik
21081 Insight Avenue, Tulsa, OK, 74135
918-555-2240 | 918-555-2645
mark.sarasua@code-logik.com

PREPARED FOR:

Maria Zabala
Owner

The Willow Tree
600 N. Grand Ave., Tahlequah, OK, 74464
918-456-5511 | 800-722-9614
maria.zabala@thewillowtree.com

PROJECT OVERVIEW

PROJECT NAME: The Willow Tree OMS
PROJECT REFERENCE: CS4233 CAPSTONE PROJECT 2024
DATE: 29 JAN 2024 – 29 APR 2024

OBJECTIVE: Redesign The Willow Tree's Order Management System (OMS) to create a secure, stable, high-performance application capable of keeping pace with restaurant business demands while providing the customer with the best possible dining experience.

PROBLEM STATEMENT: The current OMS software performance is impacted by circumstances extending beyond The Willow Tree's scope of business; the hosted source code is also vulnerable to random and targeted security threats and requires a redesign to protect the integrity of The Willow Tree's day-to-day operations.

RETRIEVE MENU ITEMS

The Retrieve Menu Items Iteration focused development on the Database, MenuItem, and MenuItems classes. A simple JSON database file consisting of five generic menu items (Figure 1) was used with test-specific code (Figure 2) to test correctness.

```
{
  "Name": "Item 1",
  "Image": "none",
  "Description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
  "Category": 1,
  "Price": 5.99,
  "Period": [
    true,
    true,
    true
  ]
},
```

Figure 1. JSON Database Menu Item 1 of 5.

```
static void Main(string[] args)
{
    Console.WriteLine("\n\nRetrieve Menu Items Testing Tool\n\n");
    MenuItems menu_items = new MenuItems();
    List<MenuItem> menu_items_list = menu_items.Items;
    foreach (MenuItem item in menu_items_list)
    {
        Console.WriteLine
        (
            item.Name + " " +
            item.Image + " " +
            item.Description + " " +
            item.Category + " " +
            item.Price + " " +
            $"{item.Period[0]}, {item.Period[1]}, {item.Period[2]}"
        );
    }
    Console.WriteLine("\n\n");
    Console.ReadKey(true);
}
```

Figure 2. Retrieve Menu Items Test Code.

In the test code, an instance of the MenuItems class was created, and a List<> of MenuItem was used to store the five generic menu items in the JSON database. A foreach loop was used to iterate through the List<> of MenuItem. The expected console output was a line-by-line display of each of the five menu items in the JSON database.

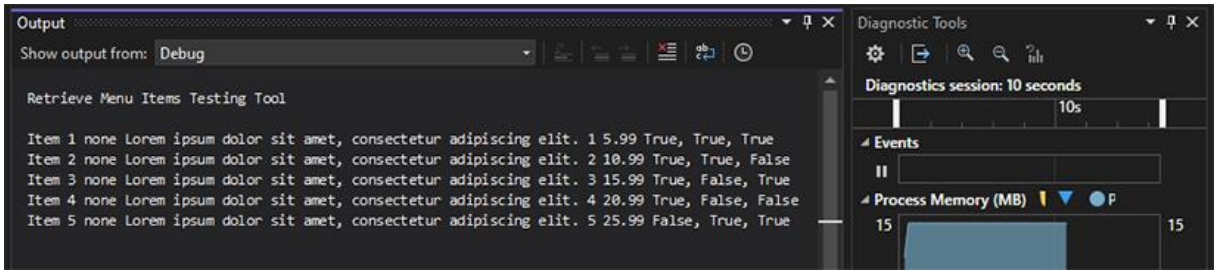

Figure 3. Retrieve Menu Items Test Console Output.

Since the console output was what was expected (Figure 3), the Retrieve Menu Items Test is considered a success. Therefore, the Retrieve Menu Items Iteration will be closed, and the View Menu Iteration will be opened.

VIEW MENU

The View Menu Iteration focused development specifically on the Menu class. The JSON database was updated to consist of six menu items (Figure 4) and was used with test-specific code (Figure 5) to test correctness.

```
{
  "Name": "Coffee",
  "Image": "none",
  "Description": "Black coffee",
  "Category": 4,
  "Price": 5.99,
  "Period": [
    true,
    true,
    true
  ]
},
{
  "Name": "Cheesecake",
  "Image": "none",
  "Description": "Strawberry Cheesecake",
  "Category": 5,
  "Price": 10.99,
  "Period": [
    true,
    true,
    true
  ]
},
```

Figure 4. JSON Database Menu Items 1 and 2 of 6.

```
Menu menu = new Menu();
List<List<MenuItem>> current_menu = new List<List<MenuItem>>();

Console.WriteLine("Breakfast Menu\n");
current_menu = menu.current_menu(PERIOD.Breakfast);
foreach (List<MenuItem> category in current_menu)
{
    foreach (MenuItem item in category)
    {
        Console.WriteLine
        (
            $"  {Convert.ToInt32(item.Period[0])}  " +
            $"  {Convert.ToInt32(item.Period[1])}  " +
            $"  {Convert.ToInt32(item.Period[2])}  " +
            $"  {item.Name}  " +
            $"  {item.Description}  " +
            $"  {item.Price}  "
        );
    }
}
Console.WriteLine("\n");
```

Figure 5. View Menu Test Code (Partial).

In the test code, an instance of the Menu class was created, and a List<> of MenuItem List<> was used to store the six menu items in the JSON database. A series of nested foreach loops were used sequentially to iterate through the List<> of MenuItem List<> to test PERIOD functionality. The expected console output was a multi-period display of the six menu items in the JSON database.

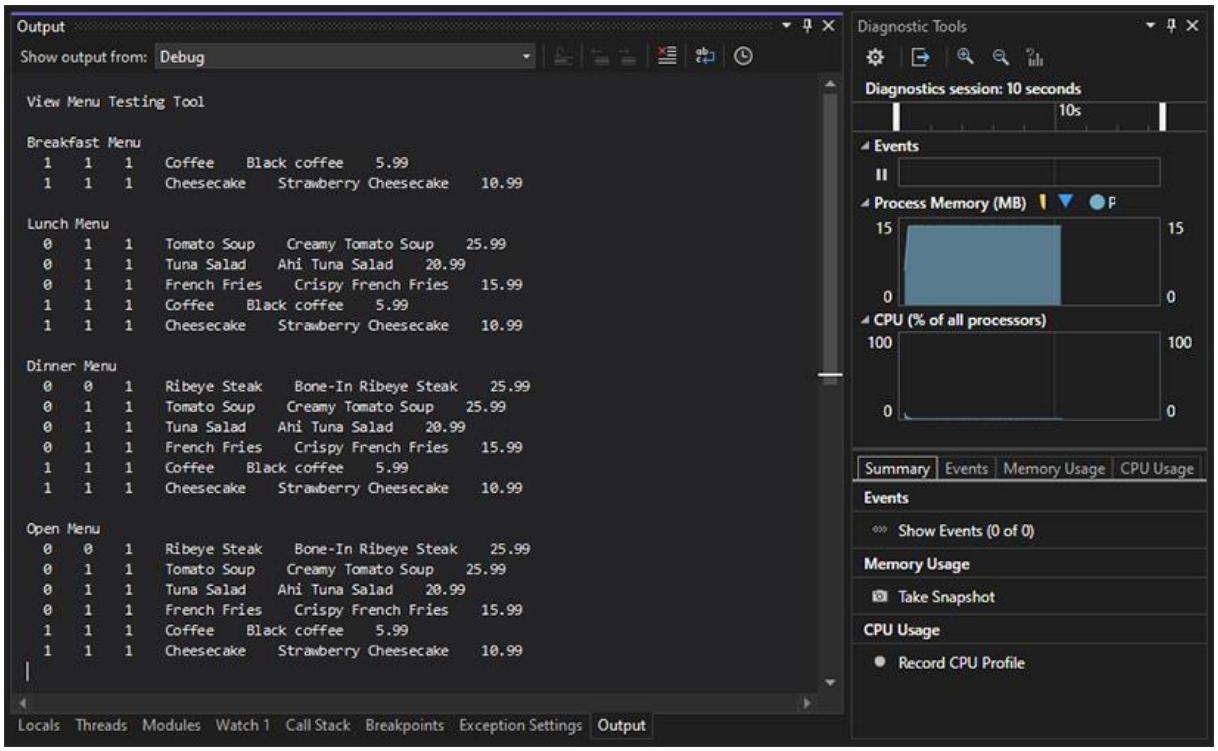

Figure 6. View Menu Test Console Output.

Since the console output was what was expected (Figure 6), the View Menu Test is considered a success. Therefore, the View Menu Iteration will be closed, and the Customer User Interface Iteration will be opened.

CUSTOMER USER INTERFACE

The Customer User Interface Iteration focused on developing the Graphical User Interface (GUI) XAML, as well as the CustomCommands, EventLogic, KMSAPI, MainWindow, OrderLogic, PSPAPI classes, and partial classes. This included modifying the Database, MenuItem, and Menu classes so that the GUI would function correctly. The JSON database was updated to consist of thirty-two menu items (Figure 7) and was used to test correctness.

```
{
  "CATEGORY": 1,
  "NAME": "Entree 1",
  "PRICE": 15.99,
  "IMAGE": "temp.png",
  "PERIOD": [ true, true, true ]
},
...
{
  "CATEGORY": 2,
  "NAME": "Side 1",
  "PRICE": 3.29,
  "IMAGE": "temp.png",
  "PERIOD": [ true, true, true ]
},
...
{
  "CATEGORY": 3,
  "NAME": "Dessert 1",
  "PRICE": 8.69,
  "IMAGE": "temp.png",
  "PERIOD": [ true, true, true ]
},
...
{
  "CATEGORY": 4,
  "NAME": "Drink 1",
  "PRICE": 5.49,
  "IMAGE": "temp.png",
  "PERIOD": [ true, true, true ]
},
```

Figure 7. JSON Database Menu Items 1, 9, 17, and 25 of 32.

The Customer User Interface Test did not require additional test-specific code to test for correctness because User Acceptance Testing (UAT) was needed to ensure proper functionality.

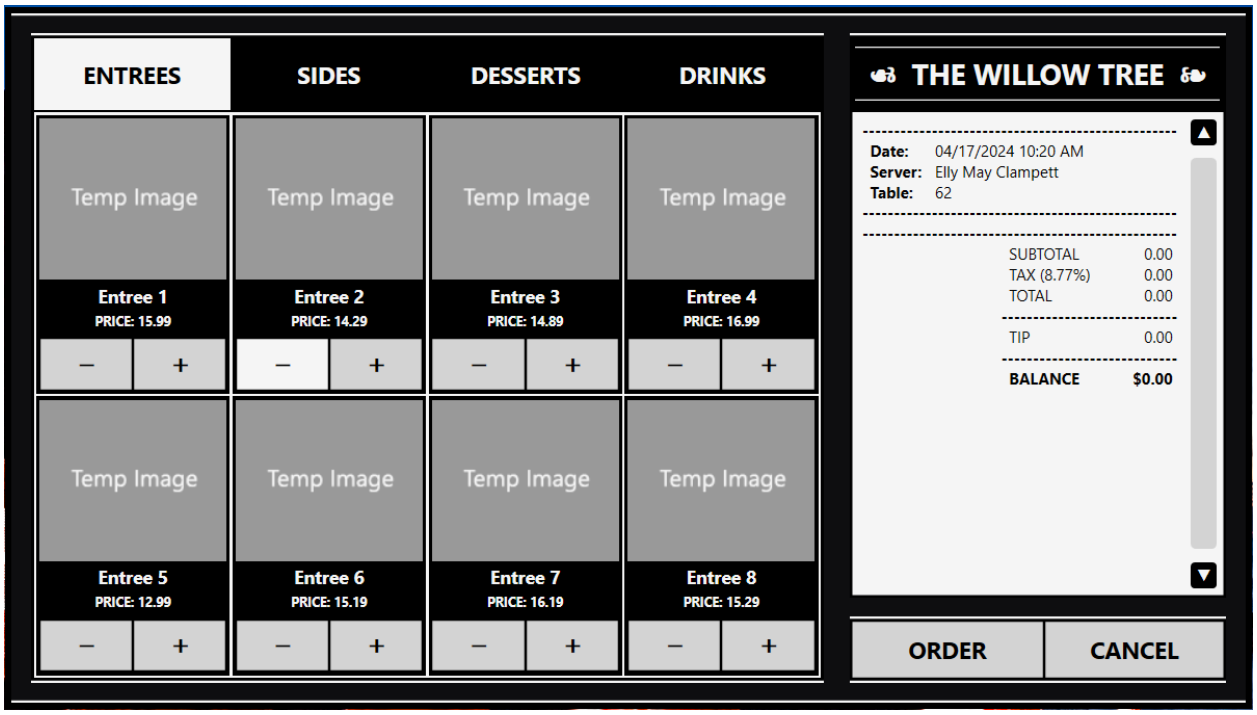

Figure 8. Pre-Order GUI State.

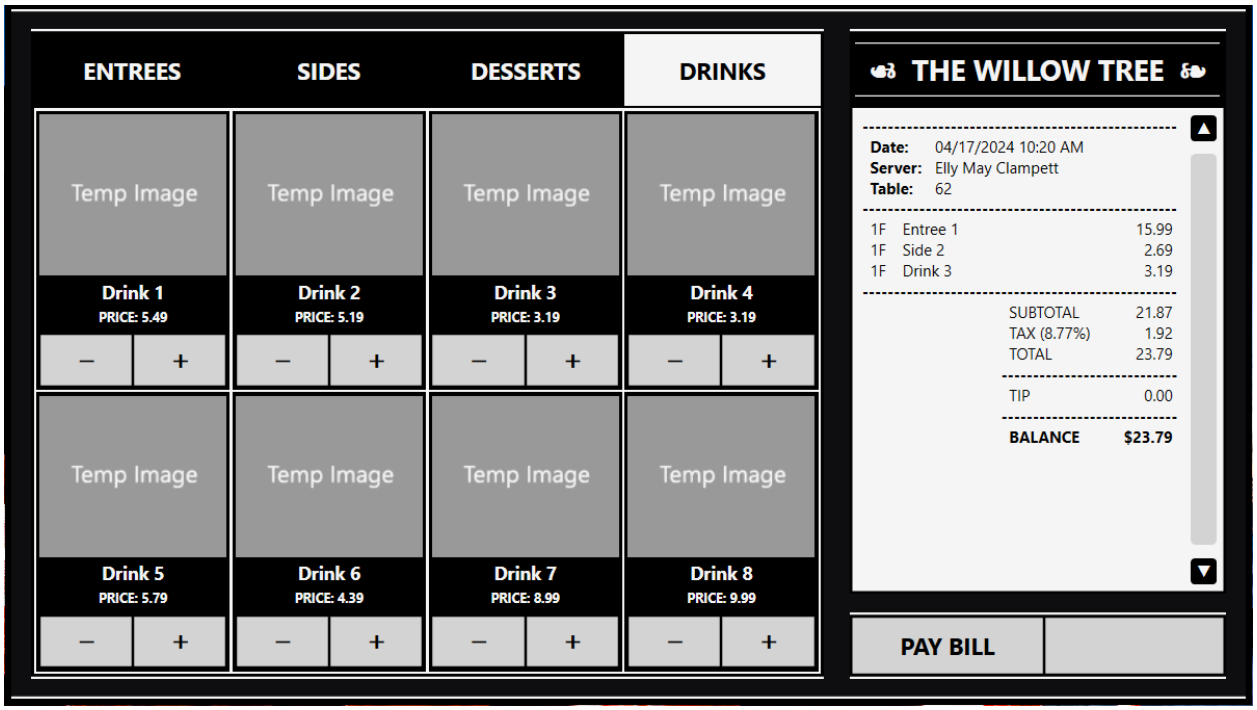

Figure 9. Post-Order GUI State.

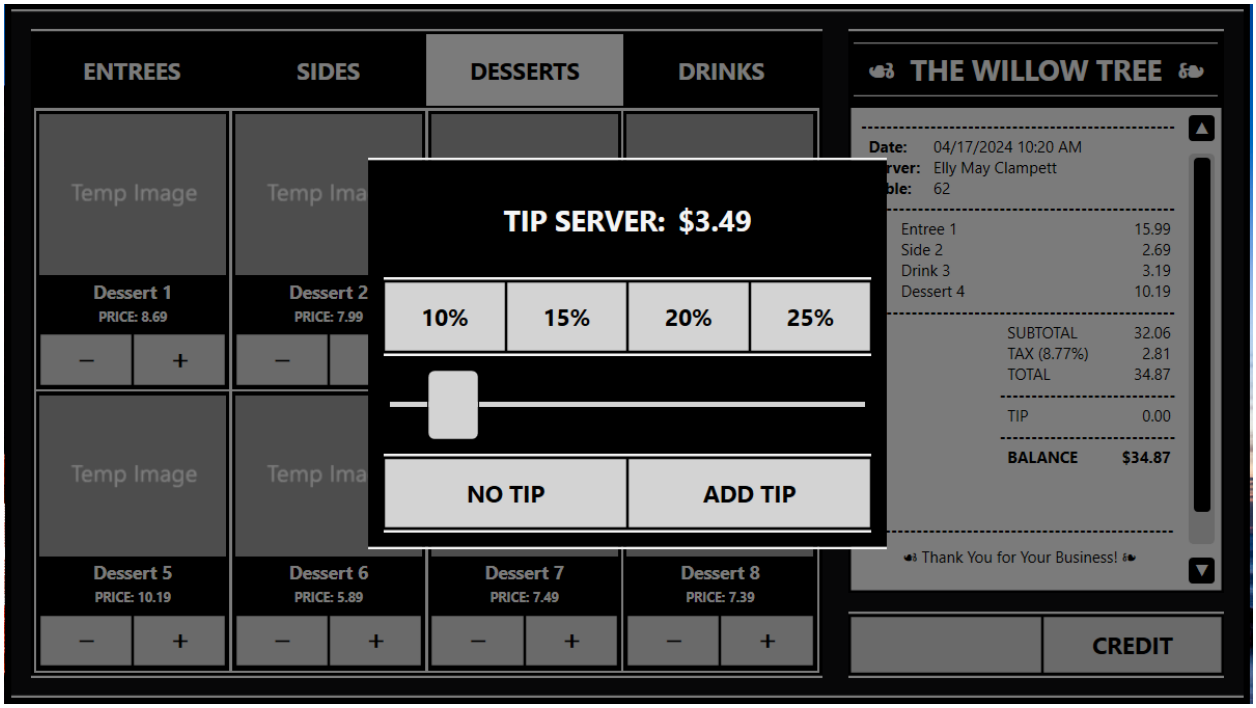

Figure 10. Add Tip GUI State.

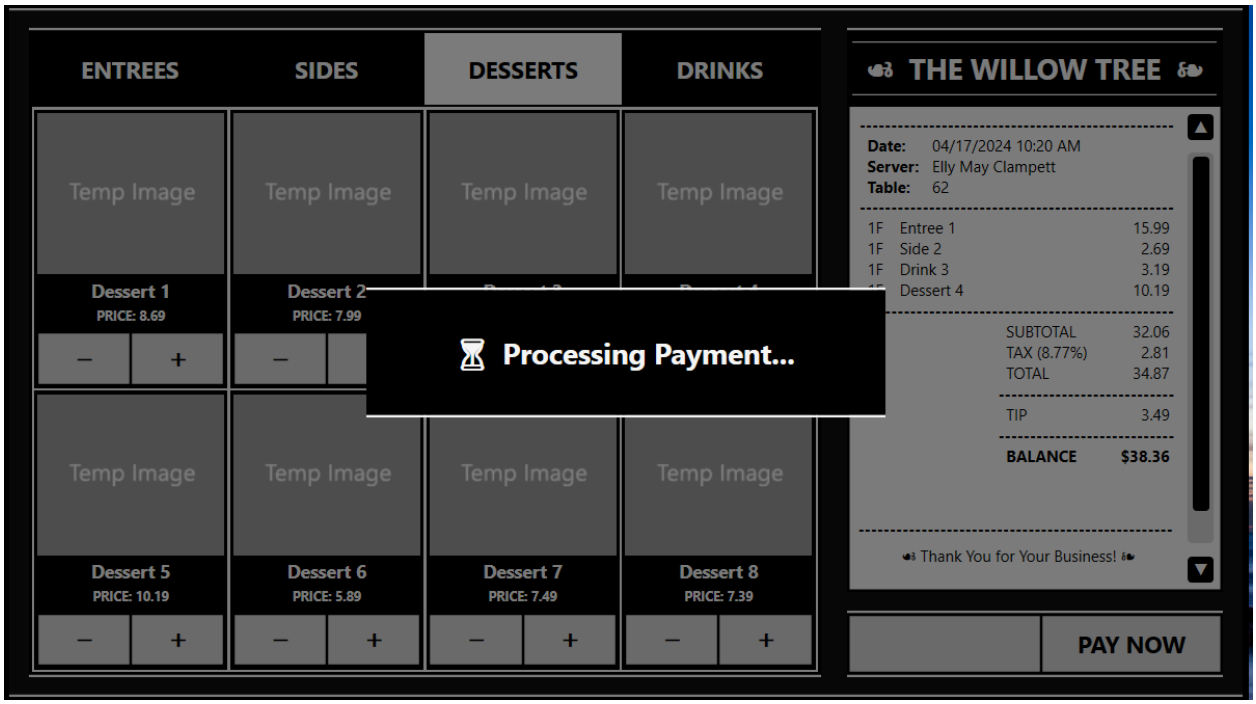

Figure 11. Process Payment GUI State.

Since the GUI states for each activity in the control flow were what was expected (Figures 8, 9, 10, 11), the Customer User Interface Test is considered a success. Therefore, the Customer User Interface Iteration will be closed, and the Release Iteration will be opened.

For further details and information, please contact:

Mark Sarasua, Jr.
Application Development
918-555-2240 | 918-555-2645
mark.sarasua@code-logik.com