Restaurant Manager Assistant

Evaluation Manual

**November 27, 2022**

**Slippery Rock University**

Bradley Smith **bms1040@sru.edu**

Ethan Pasman **eap1008@sru.edu**

# Overview

This document describes an evaluation of the Restaurant Manager Assistant system. The evaluation will consist of the following:

* Problems - Listing the current problems with the system
* Cleanup - Listing methods of how the aforementioned problems can be fixed
* Improvement and Expansion - Listing how the system can be improved and expanded

# 2. Navigation

* To assist in navigating this document, Headers have been made for each major topic of the document.
  1. Ex. ‘Problems’ header for the section about the current problems the system faces.

# 3. Problems

## Problems are anything wrong or unintended within the system’s behavior or how it was made, need to be worked around in the system, and would be fixed given more time for this project. These problems should be dealt with by anyone picking up this project to expand on it in the future, likely through the methods mentioned in the **Cleanup** section of this document.

# Problem 1

## One of the problems in the system is the relatively low amount of unit testing done for the amount of code in the project.

## Despite the fact that the majority of the system’s code and methods are contained within the RestaurantManagerController class, the fact that this class was the only one tested by a unit test is something that can be improved upon with more thorough testing.

* Some of the methods in the controller test class have been commented out due to an error that is thrown only from the test methods, and due to time constraints we have not been able to figure out how to fix said error. Each method that had been commented out was tested thoroughly through human factor testing, however.

# Problem 2

## Another of the problems we were unable to get to fully is system optimization.

## Currently, there are a lot of methods in our controller class that do very similar things or display very similar pages, but subtle differences, such as the navigation bar needing to be different for logged in users or forms to update database entries being slightly different depending on the user’s role in the program, have left us with needing to create different pages and methods for each specific thing we wanted to achieve.

* Some of these pages, such as the difference between customer and guest pages, could likely have been done more efficiently by using some more of Thymeleaf’s features on a single page instead of splitting them into two.
* On a similar note, some functions loop through every entry in a repository’s “findAll” method, instead of creating a new method with the specific query intended. Adding more methods in our repository classes to use the specific queries needed for these methods could optimize the program by getting rid of unnecessary looping.

# Problem 3

## A third problem we wanted to fix but didn’t get done is the fact that all of our controller code is loaded through a single class, RestaurantController, instead of organized and split into several different controller classes.

## This problem makes the system seem disorganized when looking through the code, and can make trying to find a specific method more confusing.

# 4. Cleanup

## Cleanup would consist of fixing the problems listed in the Problems section of this document. We will list the following as potential solutions to each problem.

# Solution to Problem 1

## The solution to the lack of unit testing would be to further test the other classes within the project. With more time, one could also fix the error affecting the commented out methods in RestaurantControllerTests.

# Solution to Problem 2

## The solution to the optimization of the Restaurant Manager Assistant system would be to determine which pages and methods are taking up the most time and resources, as well as which would be easiest to optimize, and take action with the results found.

* Unnecessary pages and their methods returning them could be deleted if the pages with very similar things used smarter code, such as adding an attribute of the current user in the model and using conditional statements for the user’s role in the HTML to change small things around on a page.
* Unnecessary looping can be removed by using more query methods in the repository classes, as the controller will not have to loop through every entry in the table and instead can more efficiently get every entity that fits the attribute that would’ve been looped for.

# Solution to Problem 3

## This solution would be the most simple, as one would only need to create more organized controller classes and then move over the existing code from RestaurantController.

# 5. Improvement and Expansion

## The improvement and expansion section is to be used for ideas for any project attempting to build off of the current system, making it better or more complete.

# Improvement 1

## One improvement that could be made is overhauling the website’s CSS, as the GUI for many of the pages looks less appealing than many professional restaurant websites.

# Improvement 2

## Another improvement to be made is adding a system for adding new HQ administration and management, as the HQ staff are currently hardcoded in the logins, and the system provides no way to add more HQ staff accounts.

# Expansion 1

## Currently, the system keeps track of sales and profits for each restaurant under the hood, but nothing is done with this information.

## The same goes for customer payment details taken from the payment page, and the payout amounts counted for each employee as calculated in RestaurantController.payCalc(), as the project is not connected to any payment gateway API, and cannot actually process the transaction by moving money from their account.

* If the project is to be deployed in a real-world environment, connecting the code to a payment gateway will be necessary to actually process payments.

# Expansion 2

## Another of the ways this system could be expanded is through the inclusion of employee scheduling.

* As of the current system, restaurant serving staff and all management are able to clock in and out freely, with no regard for scheduling or validation for the hours worked.
* A system for scheduling, possibly done by the restaurant administration, would be one way to expand this project into a more complete system.
* Adding a page to view currently clocked in employees would also be a good idea for the management.

## 