Restaurant Management Application

Team 3

Sean Collins

Nam Do

Tony Guirguis

Sara Hamidi

Danny Nguyen

Tuan Nguyen

Brian Vu

Preface

Big or small, restaurants always run into the problem of either managing an ever-growing customer population or having trouble increasing their customer rates. For instance, some restaurants may have trouble managing customer traffic with reservations tracked through analog methods. This leads to inaccurate waiting times, poor tabling methods, and unhappy customers. On the other hand, restaurants may have trouble keeping their traffic of customers consistent. Some weeks or months may seem like business is booming while other weeks restaurants may yo-yo between being packed and then practically deserted..

That is why our product will help restaurant owners manage and grow their businesses to the standards they could only dreamed of being at before. This document is therefore intended to be read by Restaurant Owners and their staff of all business sizes. That way, all parties involved will be able to operate our application that benefits them the most.

Version updates will happen for the following reasons:

- A new security protocol needs to be implemented
- A core function of the application needs to be fixed
- The properties of a function / feature has been proven to be unnecessary

Current Version: 3.0 (Changes for Implementation)

- Added architectural changes from the last version
- Added detailed design changes from the last version
- Added requirements changes from the last version

You can find more information on our website by clicking this link.

Introduction

Goal

To provide a useful and efficient tool for Restaurant Owners to use to either maintain or improve their businesses.

Target Audience

- Restaurant Owners/Managers
- Restaurant Staff
  - Servers
  - Cooks

Needs

- Need to manage customer traffic during rush-hours

- Need to track restaurant activity during different times of the day
- Need to process and manage orders between kitchen and server staff
- Need to determine which dishes to sell or discard from the menu

Product

Restaurant Management System

Architectural Changes

- **Minor Change in Purpose | Models and Controllers**
    - Controllers were originally designed to process information pulled by Models. Models would pull / save information from / into persistent memory
        - *CHANGE*: Controllers also now pull information from memory as a part of processing information. Models serve as objects to transfer data from memory into and work with in their operations. Almost serves as a "container" to pour data into in order for it to be more accessible by the system overall
        - *RATIONALE***:** We realized that we needed to store multiple objects (categorized as models) into lists that all needed to be saved into persistent data. Going through each object to perform that operation would not be efficient. We thus opted to have the controllers go through the list of objects and save them all into persistent data in one go.

Detailed Design Changes

- **Class Diagram**
    - *CHANGE 1:*
        - Controllers have been added as Components in the implementation but are not shown in the diagram
        - They also take on some of the functions included in the original components
        - They should be attached to the Component they are named after
    - *RATIONALE 1:*

        - As stated previously in another report, controllers were meant to take on operations stacked into a singular component and distribute the workload of a component as well as the hard-code in them
- **JSON**
    - *Original:*
        - We were going to save everything in a .txt file or serialize it in a .dat file
    - *CHANGE:*
        - We instead went with using JSON files to store everything
    - *RATIONALE:*
        - Easier to read directly by Users
        - Can be edited directly by the User if needed
        - Reading from a text file means that we ourselves need to implement a read / write system to process information between text and system objects
        - JSON allows for simple data processing so we can focus on developing the system instead
- **Placing Orders**
    - *CHANGE 1:*
        - Server Controller replaces the Order Controller
    - *RATIONALE 1:*
        - The Server View should not be performing all the operations of a Server. Should focus mainly on displaying information
        - Creates a separation of duties
        - Order Controller is not necessary as the Server themselves will be managing the Orders (carried out as well by the Server Controller)
    - *CHANGE 2:*
        - Serview View calling openServerMenu() -> mainMenu_Server
            - Also applied to other sequences with the same flow
    - *RATIONALE 2:*
        - Server has multiple submenus that have their own submenus. Defining where the main menu for the Server is crucial not to get lost when navigating the menus

    - *CHANGE 3:*
        - displayOptions() directs information back towards Actor (Waiter) -> specific submenu functions will direct unique information back to the Server
    - *RATIONALE 3:*
        - Each operation performed by the Server Controller has multiple steps to them
        - Trying to have one function to display all those steps is near-impossible. Solution was to separate them into their own submenu functions in the Server View
    - *CHANGE 4:*
        - Entering Dishes into an Order gets sent and processed into the Server Controller first
        - Server Controller dumps data into the Order Model and is validated back to the Server Controller
        - Operation exists from Server Controller to the Server View
    - *RATIONALE 4:*
        - Given the Architectural changes stated above, the Server Controller now has the duty to take information and put them into the necessary Object Models to be processed and operated on
        - Results in more fluid operation flow
        - Server Controller now acts as an official bridge between Server View and Order Model. No direct interactions should be made between them to avoid clutter in the operations and in the code itself
- **Adding Menu Items**
        - *CHANGE:*
            - The Manager View and Menu Controller have been replaced by the POS component
        - *RATIONALE:*
            - The POS should be accessed only by Managers and higher

        - Specifying a Manager View and Controller would mean we would need to create individual components for the rest of the staff above Managers
        - Delegating it all in POS makes it easier for more staff to access the same operations
- **Making Reservations**
    - No outstanding changes were made for this sequence

<u>Requirement Changes</u>

- **Table and Order Management**
    - Orders are processed and managed by the Servers themselves, not through the POS
        - This changes our design overall as now the Server Controller has more duties to fulfill than the POS
        - We have implemented it and it is much more straightforward; especially with operations requiring components to read and write from JSON.
        - Reduces the effect of big call changes between multiple components
    - Removed feature for displaying reserved times as grayed out
        - Does not change the design as it is more of a feature to be seen
        - Does not change how the system operates
        - No need to be implemented in the feature unless future updates for color-coding are implemented
- **Kitchen Receives Orders Placed by Servers**
    - Chefs won't be able to take notes anymore to take inventory on ingredients for specific dishes
        - This does not affect the design much as it is a stand-alone feature that is a quality-of-life feature
        - This will be implemented in the future
- **The Customer Pays the Bill**
    - This whole requirement has been deprecated
        - Removes a good chunk of the design but does not affect how the system operates

        - The only difference is that Servers won't be able to calculate the bill on their device. Will need to go to a cash register
        - Depending on our client, might be implemented in the future
- **Data Aggregation of Top Selling Dishes**
    - This whole requirement has been deprecated
        - Also removes a good chunk of the design but does not affect how the system operates
        - This only affects the owner as a whole as they would need to manually analyze the data accumulated over time
        - This will be implemented in the future to aid the owner and help them catch anything they won't see initially