Wait Less Testing and Inspection Report

Prepared by
Spancer Guo, Jiajie Lin,
Hongcheng Wu, Zachary Flebbe
***for use in* CS 440**
at the
University of Illinois Chicago
April 2020

Table of Contents

I Project Description

1 Project Overview

Wait-Less is an application created to optimize a restaurant's efficiency by facilitating worker communication with an easy to user interface. Through the creation of different modes (manager, host and server), essential restaurant tasks are separated among employees and information is distributed accordingly. All information is stored in a database to be accessed across multiple devices. Privileged information (such as daily sales/statistics) is protected and only accessible if login type is correct. The manager can generate employee schedules and check restaurant statistics; the manager will also be able to operate as a host or server situationally. The host can generate a wait list for customers on location and a reservation list for call-ins. These lists include the customers name, party size and time of arrival (or current time) based on reservation/wait classification; this information can be sorted by time or party size. Once a party is ready to be seated, the table is assigned to a server and removed from the list. The server can access their assigned tables, order/remove food items, add comments to orders and check out a table.

2 Project Domain

Wait-Less was created to aid restaurants in the many intricate processes involved with running a business in the service industry. Some of the main challenges that a business may present are addressed within this application. These include, but are not limited to: scheduling employees, recording worked employee hours, generating daily revenues, attending to customers, placing orders and relaying key information between employees in a timely manner.

3 Relationship to Other Documents

This project has been adapted from the project summary report offered under the “Project Archives” tab on the CS 440 website. Wait-Less was a project proposed by Group 10 during the Fall 2019 semester. From this project summary, all requirements and design specifications were developed.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Manager: Employee who is responsible for the highest level operations of the restaurant including generating schedules and accessing revenue information. This employee should have access to all information stored by the application and any operations that can be performed.

Host: Employee who is responsible for the reservation/waitlist lists containing customer information. This employee will only have access to the operations listed above.

Server: Employee who is responsible for seating customers, placing orders and checking out tabs. This employee will only have access to the operations listed above.

Database: SQL database on local host. Contains information necessary for running the application including schedules, daily revenues and shift times.

Revenue: Total sum of tabs on a particular date. A float value updated as each tab is closed out.

Tab: A sum of of a table's ordered items. A float value composed as the sum of each item ordered by a table.

4b UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in [1]. Any exceptions are noted where used.

4c Data Dictionary for Any Included Models

Customer = name + phone number + party size + timestamp

timestamp = year + month + date + hour + minute + second + nano

year = the year minus 1900

month = 0 to 11

date = 1 to 31

hour = 0 to 23

minute = 0 to 59

second = 0 to 59

nano = 0 to 999,999,999

Employee = ID + name + position + time-in + time-out

time-in = 00:00 to 23:59

time out = 00:00 to 23:59

Host = super Employee + reservation list + wait list

Manager = super Employee

Menu Item = name + price + instructions

Database = host + port + username + password + url + connection + statement + reset

Tab = items + total

Table = id + availability + table size + customers + response server + tab

table size > 0

II Testing

5 Items to be Tested

- ID T-01: Manager check table
- ID T-02: Add/delete waitlist and reservation
- ID T-03: Sign in

6 Test Specifications

ID# - T-01

Description: Test if the check table feature works properly for manager

Items covered by this test: Manager

Requirements addressed by this test: FUNC-4

Environmental needs: Wait Less application

Intercase Dependencies: Manager has to be created, and logged in.

Test Procedures: First log in the manager account, then select check table button.

Input Specification: None

Output Specifications: A layout of the restaurant with tables should be displayed, and the status of the table should be distinguished by color.

Pass/Fail Criteria: All the tables are properly displayed and the status of each table is displayed correctly.

ID# - T-02

Description: Add/delete from waitlist and reservation list

Items covered by this test: waitlist, reservation

Requirements addressed by this test: FUNC-1

Environmental needs: Wait Less application

Intercase Dependencies: Host has to be created, and logged in.

Test Procedures: First log in the host account, select check waitlist or reservation button, select add or delete, then select show waitlist or reservation list.

Input Specification: String of names, integer of party size, integer of phone numbers and time with the format of “MM-DD-YYYY HH:MM” for reservation

Output Specifications: A list a customers will be displayed

Pass/Fail Criteria: The new customer is added properly in the list, and estimated wait time will be displayed.

ID# - T-03

Description: The product shall allow signing into the application to take no longer than 3 seconds.

Items covered by this test: host, server, manager

Requirements addressed by this test: SPLA-2

Environmental needs: Environment that can run Java code and unit test

Intercase Dependencies: the account has to be created

Test Procedures: Run the unit test

Input Specification: None

Pass/Fail Criteria: pass unit test

7 Test Results

ID# - T-01

Date(s) of Execution: 4/20/2020

Staff conducting tests: Spancer Guo

Expected Results: A layout of the restaurant with tables should be displayed, and the status of the table should be distinguished by color.

Actual Results: A layout of the restaurant with tables should be displayed, and the status of the table should be distinguished by color.
Test Status: Pass

ID# - T-02
Date(s) of Execution: 4/20/2020
Staff conducting tests: Jiajie Lin
Expected Results: The new customer is added properly in the list, and estimated wait time will be displayed.
Actual Results: The new customer is added properly in the list, but estimated wait time is not displayed.
Test Status: Fail

ID# - T-03
Date(s) of Execution: 4/19/2020
Staff conducting tests: Zachary Flebbe
Expected Results: pass unit test
Actual Results: passed unit test
Test Status: pass

8 Regression Testing

None

III Inspection

9 Items to be Inspected

There are three union tests: host, server, and manager.

Then there are unit tests corresponding to each union test. Host is testing adding and deleting customers from reservation or waitlist & sorting the waitlist by party size or time. For testing server, the server needs to correctly checkout and order food from the menu. For testing manager, manager needs to successfully open/close by resetting table and total revenue every day; also

testing manager by generating a schedule with or without time slot overlap in one employee or multiple employees. There is a SQL database used for login and sign in, sql and sql adapter needs to correctly respond by giving information such as login username and password.

10 Inspection Procedures

Under the groups; github, there is a folder called Minutes. This folder is used for keeping track of distribution and contribution for every team member, then generating our weekly meeting report. We had over 9 meetings, we discussed coding and processing for every week.

Even after quarantine, all of us are still meeting weekly on Zoom(online meeting), and reporting our weekly processing.

11 Inspection Results

After the first demo, we rebuilt almost 50% of our code to become more formed and organized. During the reconstruction we found a few errors that did not affect the whole program and could be easily fixed, such as double value overflow or UI can not going back and forth.

IV Recommendations and Conclusions

After each accurate and comprehensive test an improvement is limiting the user action to improve program stability. Since users can do so many unexpected behaviors, by providing the user with a fully tested selection to limited user interaction with the program, so every step of the user is being tested.

V Project Issues

12 Open Issues

Wait-Less will have a lot of competitors that offer similar software. because the demand for food-serving applications is very high. We have to make sure

our application has an advantage over others. Also, cybersecurity may be one issue, we have to reinforce how to protect the customers information and in order to ensure security.

13 Waiting Room

Feature that would be added was a waitlist for the food serving, to make sure that customers who order the food first would get the food first. This system would make it easier for the employees to know important information such as what they ordered, how long it's been since the server checked up on the table. In the feature, we would be added to some type of the AI server, they would have features like automatically serving food, and filling water for the customers who need help.

14 Ideas for Solutions

Features would be added by our software development team to improve the security issue and the additional features. The additional features must be implemented and tested by the different teams. We have to get the advanced robot which can cooperate with our restaurant system, and the accuracy of the robot that serves in the correct table.

15 Project Retrospective

In conclusion, our product Wait-less was created to aid the food-serving store and make the customers have a better experience in the restaurant. Also, it could help restaurants to analyze their data avenue and in order to improve their serving methods.

VI Glossary

Waitlist: a list contains customers that are currently waiting to be seated.

Reservation: a list contains customers that called ahead to book a table at a certain time.

VII References / Bibliography

[1] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.