***Reserve Your Table Final Report***

**

**Prepared by**

**Juan Camacho, Miran Member, Mirac Kara, and Latif Vaid**

**for usein CS 440**

**at the**

**University of Illinois Chicago**

**September 2022**

**Table of Contents**

[*REMOVE OR REPLACE ALL TEXT IN RED ITALICS BEFORE SUBMITTING REPORT* 2](#_gjdgxs)

[*How to Use This Document* 2](#_30j0zll)

[List of Figures 6](#_1fob9te)

[List of Tables 7](#_3znysh7)

[I](#_2et92p0) Project Description 8

[1](#_tyjcwt) Project Overview 8

[2](#_3dy6vkm) Project Domain 8

[3](#_1t3h5sf) Relationship to Other Documents 8

[4](#_4d34og8) Naming Conventions and Definitions 8

[4a](#_2s8eyo1) Definitions of Key Terms 8

[4b](#_17dp8vu) UML and Other Notation Used in This Document 10

[4c](#_3rdcrjn) Data Dictionary for Any Included Models 10

[II](#_26in1rg) Project Deliverables 11

[5](#_lnxbz9) First Release 11

[6](#_35nkun2) Second Release 11

[7](#_1ksv4uv) Comparison with Original Project Design Document 11

[III](#_44sinio) Testing 12

[8](#_2jxsxqh) Items to be Tested 12

[9](#_z337ya) Test Specifications 12

[10](#_3j2qqm3) Test Results 13

[11](#_1y810tw) Regression Testing 13

[IV](#_4i7ojhp) Inspection 13

[12](#_1ci93xb) Items to be Inspected 14

[13](#_3whwml4) Inspection Procedures 14

[14](#_2bn6wsx) Inspection Results 14

[V](#_qsh70q) Recommendations and Conclusions 14

[VI](#_3as4poj) Project Issues 14

[15](#_1pxezwc) Open Issues 14

[16](#_49x2ik5) Waiting Room 15

[17](#_2p2csry) Ideas for Solutions 16

[18](#_147n2zr) Project Retrospective 16

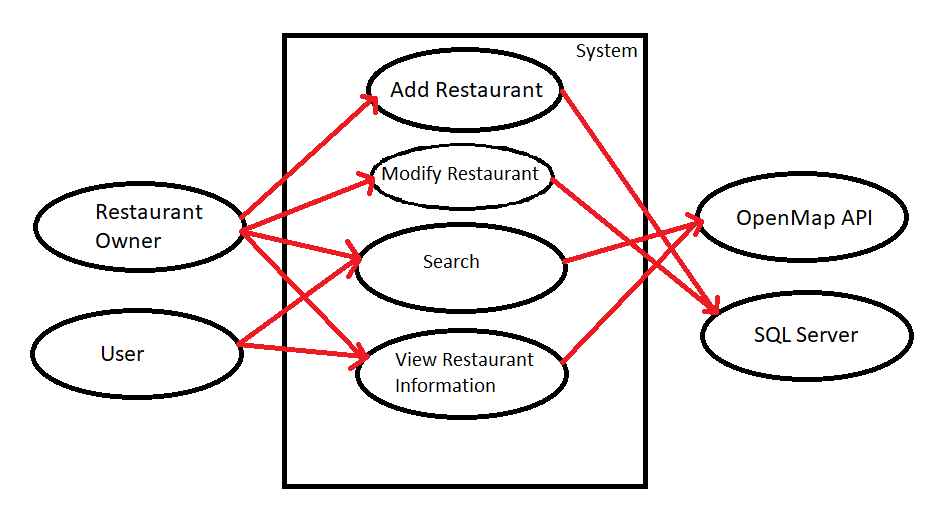
[VII](#_23ckvvd) Glossary 17

[VIII](#_ihv636) References / Bibliography 17

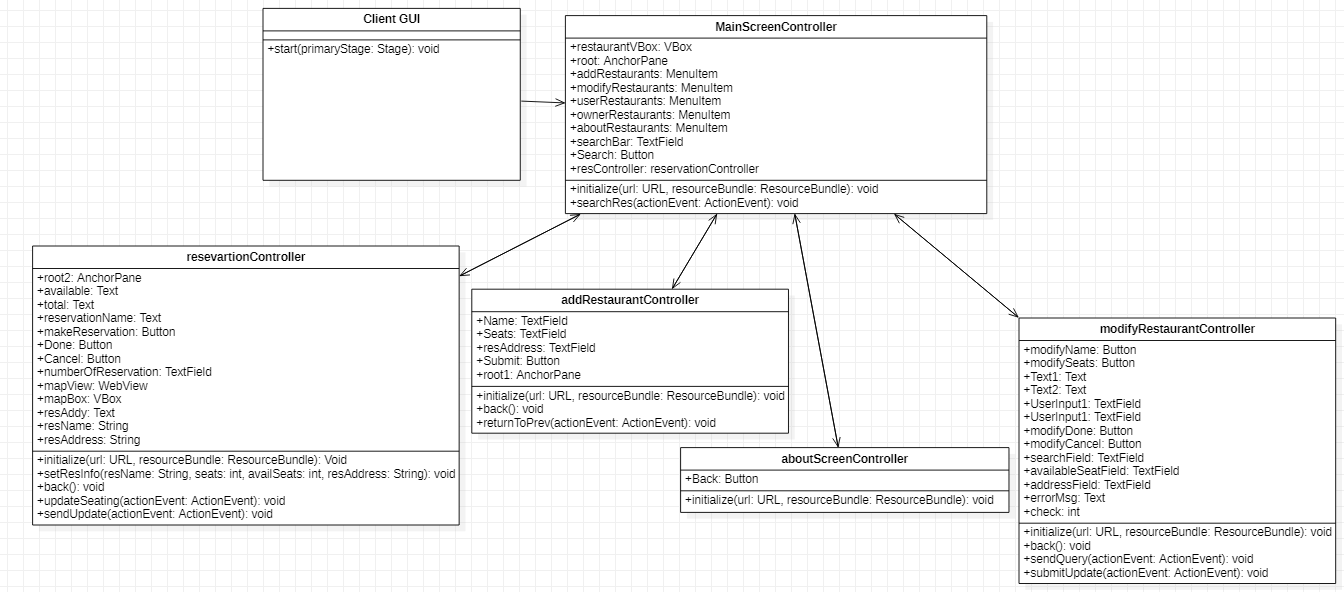
[IX](#_32hioqz) Index 18

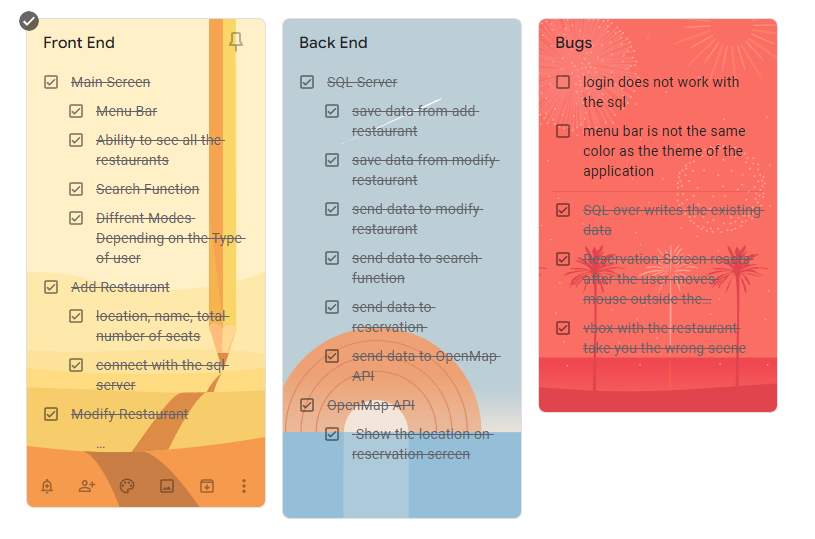
### **List of Figures**

**Figure 1 - Use Case Diagram**

******

**Figure 2 - UML Diagram**

**Figure 3: To-do List**

**

# Project Description

## Project Overview

Reserve your table is a two-sided program. On one side, users are able to search for restaurants and reserve a table for them and their party. This allows users to no longer have to wait in line at a restaurant and waste time, instead knowing their table is available ahead of time while on their way to the establishment. One the other side, you have the restaurant side of this program. This is where new restaurants can be added with the name and seats available or modify an existing restaurant.

## Project Domain

Users must be able to navigate through pages easily and the program revolves around the storage of information into a SQL database and accessing said information. That being said, all the testing revolves around these two principles, ensuring pages are working properly and easy to navigate as well as the data being stored in the proper way.

## Relationship to Other Documents

This document is the culmination of all other work done on this project. Work from previous documents will be summarized and reiterated throughout this document.

## Naming Conventions and Definitions

### Definitions of Key Terms

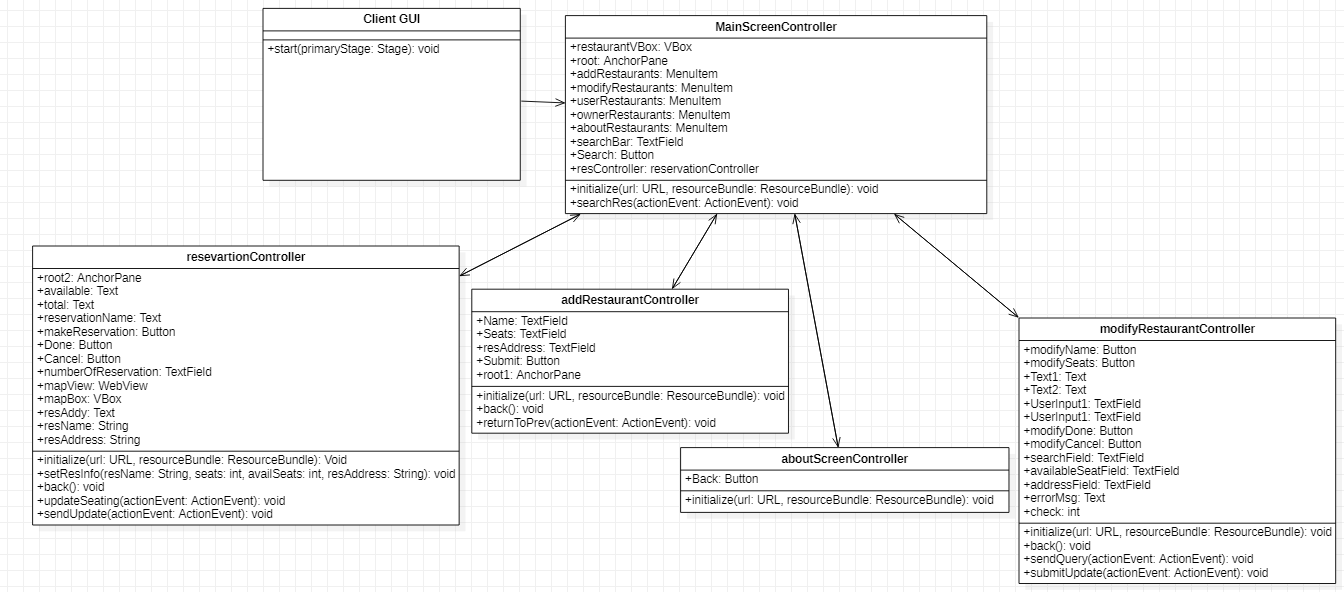
SQL Server: Local server set up to manage all the restaurant data such as the name, and the total number of seats.

OpenMap API: Map integrated in the system to allow users to see the location of the restaurant.

User Mode: This will have limited access to the application to only allow the user to see restaurant information.

Restaurant Owner Mode: This mode will allow full access to the application to allow the user to add restaurants, and modify existing restaurants data.

### UML and Other Notation Used in This Document



### Data Dictionary for Any Included Models

Restaurant's data: The restaurant's data will be collected and stored in a database. This database will have different tables for different restaurants. Different tables will be based on different details entered by the restaurant which include: name, address, available seats, total seats.

User = {reservations}

Restaurant = {name, address, available seats, total seats}

# Project Deliverables

In the end, the product we have produced is a program where you can select, search, and add restaurants we have stored. When a restaurant is selected, the amount of total seats and available seats are displayed. From here, the user is able to input how many seats they would like to reserve and are able to hold onto the seats until they are done at the establishment. Users can also search through the stored restaurants to find a specific one they would like. For restaurant owners, they can add their establishment with the name and number of seats as the only information needed. They can also modify the restaurant they have in case they would like to change the name. All of this data is then stored into the SQL database previously mentioned.

## First Release

In the first release (2/25), we had a functional front page along with data that we put in to model what the program would look like once connected to an API. The user was able to select which restaurant they would like to eat at and would be taken to the corresponding page although it was empty. The first release was more UI implementation and less functional back end.

## Second Release

In the second release, we built upon the first release by adding functionality to add restaurants as well as adding a seating chart for restaurants. In this version, restaurant owners could store their establishment in the program using the name, number of seats, and the rows/columns (for the seating chart). After selecting a restaurant from the main page, the user could click on the seats they would like to reserve and they would be saved until unchecked.

## Comparison with Original Project Design Document

Compared to the vision the previous group had, this project fell short. However, we do believe we were able to encapsulate the basics of what they had hoped for this program to become. We simply had a simpler version of what the report had asked for. However, there were some additions not originally mentioned that proved to be useful like accessing a map API so that restaurants could show their location.

# Testing

## Items to be Tested

**ID 1 - MainScreenController**

**Description:** This is the main screen of the application and holds a lot of information. This test will include switching to different scenes, grammar errors, and the sql server properly sends the data to the main screen.

**ID 2 - AboutScreenController**

**Description:** This is a screen dedicated to only crediting the creators and stating how it's only a demo of the class. The testing will include grammar check and making sure the buttons go back to the main screen function properly.

**ID 3 - AddRestaurantController**

**Description:** While this is not the main screen of the application, this is an important part of the application. This screen communicates with the sql server to add restaurant data. The testing will have to check if the restaurants are properly added. Also there will be a lot of edge cases such as the total number of seats input fields not allowing users to enter letters.

**ID 4 - ModifyRestaurantController**

**Description:** Also an important part of the application as it modifies existing data. This will also include edge cases such as including letters in the field for number of seats, or incorrect address etc. Also an important test because the sql server should properly modify the data in the database.

**ID 5 - reservationController**

**Description:** Test will include properly pulling the data from sql server and displaying it on the screen. Also grammar checks the spellings of the text fields.

## Test Specifications

**ID6 - Test 1**

**Description:** Grammar Check of the texts.

**Items covered by this test:** ID 1, ID 2, ID 3, ID 4, ID 5

**Requirements addressed by this test:** Allows users to understand the application.

**Environmental needs: NA**

**Intercase Dependencies:** ID 1, ID 2, ID 3, ID 4, ID 5, ID 7, ID 8

**Test Procedures:** Check all the text fields in the application and make sure they are grammatically correct.

**Input Specification:** **NA**

**Output Specifications: NA**

**Pass/Fail Criteria:** All the textfields make sense grammatically.

**ID7 - Test 2**

**Description:** SQL Server

**Items covered by this test:** ID 1, ID 3, ID 4, ID 5

**Requirements addressed by this test:** Proper modification of the database in the sql servers allows the application to make reservations, and add, modify restaurants.

**Environmental needs:** SQL server should function in any environment.

**Intercase Dependencies:** ID 1, ID 3, ID 4, ID 5

**Test Procedures:** Check whether ID 1, ID 3, ID 4, ID 5 work properly as they rely on the sql database. Check whether ID 1 can receive queries from sql database. Check weather ID 3 can add data to the database, Check weather ID 4 can modify data, and lastly check ID 5 can pull the data from the sql database.

**Input Specification:** **NA**

**Output Specifications: NA**

**Pass/Fail Criteria:** ID 1, ID 3, ID 4, ID 5 can pull data from the database and add, modify data on the database.

**ID8 - Test 3**

**Description:** OpenMap API

**Items covered by this test:** ID 1, ID 3, ID 4, ID 5, ID 7

**Requirements addressed by this test:** To display map location of the restaurant.

**Environmental needs:** NA

**Intercase Dependencies:** ID 1, ID 3, ID 4, ID 5

**Test Procedures:** ID 5 should show proper map location of the restaurant based on the input from ID 3 and ID 4.

**Input Specification:** **NA**

**Output Specifications: NA**

**Pass/Fail Criteria:** ID 5 can show the map location of the restaurant. ID 3 and ID 4 input for the address are properly checked to see if they are correct.

## Test Results

**ID9 - Test 1**

**Date(s) of Execution:** April 1, 2022 through April 2, 2022

**Staff conducting tests:** Miran Member and Latif Vaid

**Expected Results:** The text in the application should have proper text which is grammarly correct.

**Actual Results:** The text fields in the application are grammatically correct, and spelling error free.

**Test Status:** Pass

**ID10 - Test 2**

**Date(s) of Execution:** April 7, 2022 through April 8, 2022

**Staff conducting tests:** Mirac Kara and Juan Camacho

**Expected Results:** Check whether ID 1 can receive queries from sql database. Check weather ID 3 can add data to the database, Check weather ID 4 can modify data, and lastly check ID 5 can pull the data from the sql database.

**Actual Results:** ID 1, ID 3, ID 4, ID 5 work properly as they rely on the sql database.

**Test Status:** Pass

**ID11 - Test 3**

**Date(s) of Execution:** April 7, 2022 through April 15, 2022

**Staff conducting tests:** Miran Member and Mirac Kara

**Expected Results:** ID 5 can show the map location of the restaurant. ID 3 and ID 4 input for the address are properly checked to see if they are correct.

**Actual Results:** ID 5 shows proper location of the restaurant, and ID 3, ID 4 are error checked to determine if the restaurant location is correct.

**Test Status: Pass**

## Regression Testing

**ID12 - Test 1**

**Description:** Switching between Scenes

**Items covered by this test:** ID 1, ID 3, ID 4, ID 5, ID 7

**Requirements addressed by this test:** Check weather scenes can switch to other scenes. Refer to Figure 2.

**Environmental needs: NA**

**Intercase Dependencies:** ID 1, ID 3, ID 4, ID 5

**Test Procedures:** There are multiple buttons and menu items that allow the scenes to change, check all of them individually to see if they switch scenes as intended.

**Input Specification:** **NA**

**Output Specifications: NA**

**Pass/Fail Criteria:** switch scene as intended.

# Inspection

## Items to be Inspected

**MainScreenController**

**Description:** This is the main screen of the application and holds a lot of information. This test will include switching to different scenes, grammar errors, and the sql server properly sends the data to the main screen.

**AboutScreenController**

**Description:** This is a screen dedicated to only crediting the creators and stating how it's only a demo of the class. The testing will include grammar check and making sure the buttons go back to the main screen function properly.

**AddRestaurantController**

**Description:** While this is not the main screen of the application, this is an important part of the application. This screen communicates with the sql server to add restaurant data. The testing will have to check if the restaurants are properly added. Also there will be a lot of edge cases such as the total number of seats input fields not allowing users to enter letters.

**ModifyRestaurantController**

**Description:** Also an important part of the application as it modifies existing data. This will also include edge cases such as including letters in the field for number of seats, or incorrect address etc. Also an important test because the sql server should properly modify the data in the database.

**ReservationController**

**Description:** Test will include properly pulling the data from sql server and displaying it on the screen. Also grammar checks the spellings of the text fields.

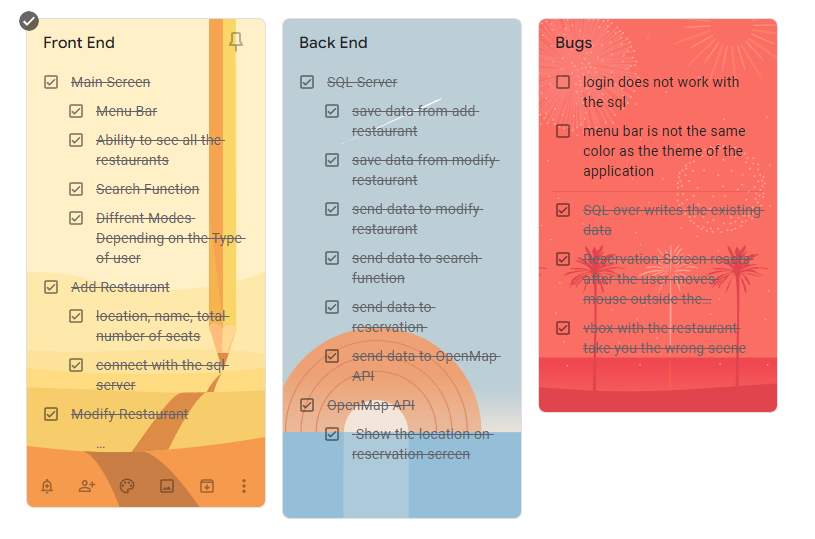
**SQL Server**

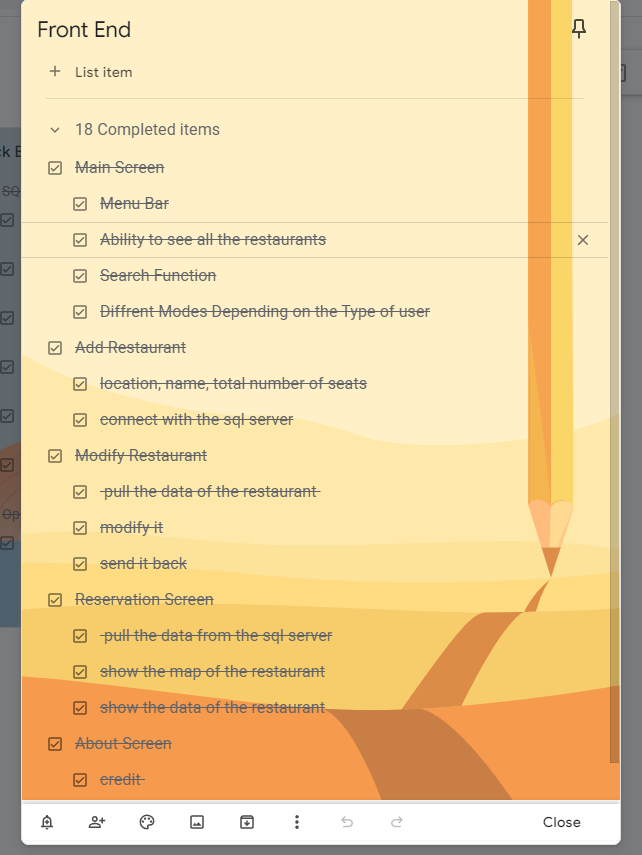
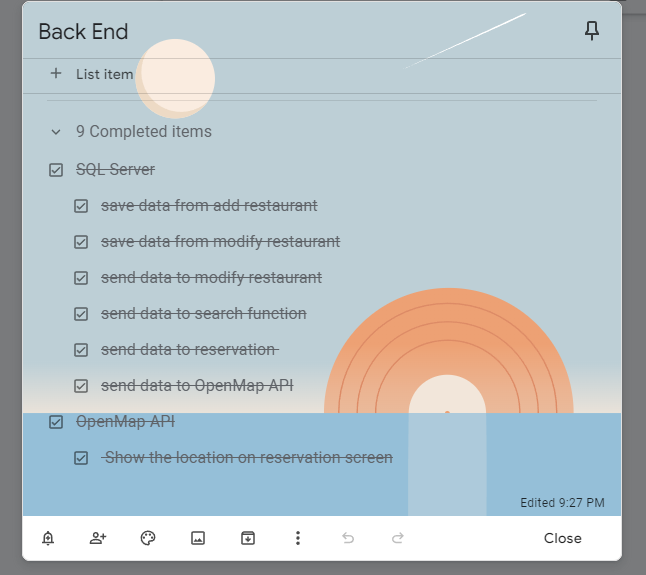
**Description:** Proper modification of the database in the sql servers allows the application to make reservations, and add, modify restaurants.

**OpenMap API**

**Description:** To display a map location of the restaurant in the reservation screen.

## Inspection Procedures

**

**

## Inspection Results

Refer to Section 4.2

# Recommendations and Conclusions

Everything we tested passed the requirements we set for ourselves. We were very happy with what we accomplished this semester with our project. For further Improvements, we would make a dedicated SQL server rather than running it on a local machine. Also to expand we would most likely need a login function to distinguish between normal users and Restaurant owners.

# Project Issues

## Open Issues

Payment: We were really unsure of how we would go out to set up a payment system. It was one of the requirements we tried to tackle however none of us were really experienced in that, not to mention going to do that in javafx.

## Waiting Room

Reimplementation of the seating chart would be nice so the user can choose specifically where they want to sit.

Connecting this program to an API to get an official database of restaurants would be incredibly useful.

The ability to access the location of the user to determine the distance between the restaurant and the user along with the best ways to get there would really improve the program.

## Ideas for Solutions

To get an API, it would be good to request access from Opentable like we had initially thought, the only downside would be certain things that cost money to use.

Getting access to the Google Maps API would easily solve the problem of being able to access the location of the user. It would also help in finding the best route the user can take to the restaurant.

## Project Retrospective

The project required unexpectedly detailed work on every part of it. The idea and the implementation was great. Working with a team made it more fun since there were many different ideas of how the project can be implemented. The project can be improved by adding more details. The idea of the project wasa great and it wass fun to work on it.

# Glossary

**The application/ The product**: Reserve Your Table.

# *References / Bibliography*

| **[1]** | **Robertson and Robertson, Mastering the Requirements Process.** |
| --- | --- |
| **[2]** | **A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.** |
| **[3]** | **J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.** |
| **[4]** | **M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.** |

# Index

Description................................................. 6, 7 Testing............................................ 8, 12