Text

Description automatically generated

PROJECT REPORT

CZ2002 OBJECT ORIENTED DESIGN AND PROGRAMMING

PREPARED BY:

SS5, GROUP 1

|  |  |
| --- | --- |
| TAN PEI LUN  AHMAD ALEENA | U2022841A  U2023405K |
| ANANDARAJAN SINDINI |  |
| LIM JIEXIAN | U2021179G |
| TAY LU KANG |  |

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

**Table of Contents**

[**Declaration of Original Work for CE/CZ2002 Assignment**](#_ktn68olo12df) **1**

[**1. Introduction**](#_h5qzgmzo5dp) **4**

[**2. Design Considerations**](#_wgeer1ddyg7n) **5**

[**3. Design Principles Used**](#_iuyjipirnr3n) **5**

[3.1. Single Responsibility Principle (SRP)](#_tfydsvzgzg8d) 5

[3.2. Open-Closed Principle (OCP)](#_8k1txxjc1jve) 6

[3.3. Liskov Substitution Principle (LSP)](#_5qku70conlh3) 6

[3.4. Don’t Repeat Yourself (DRY)](#_vas3qhd64bsc) 7

[**4. UML Class Diagram**](#_pngzkw4u6x97) **8**

[**5. UML Sequence Diagram**](#_psy4dum67lr) **9**

[**6. Black Box Testing**](#_sfmc6g62rstj) **10**

# 

# 1. Introduction

**Restaurant Reservation and Point of Sale System (RRPSS)** is a console-based application designed to assist restaurant staff to manage restaurant operations. The main goal of the application is to computerize the process of making reservation, recording of orders and displaying of sale records

This report covers the object-oriented programming (OOP) concepts and key design considerations used to implement the application. The design will also be represented in a UML Class Diagram showing the interaction and relationship between the objects, and UML Sequence Diagram for one of the functions “Check/Remove reservation booking”. Several test cases are also included to ensure that the application meets the requirements stated.

Implementation of the program are done with these assumptions:

* Reservation can only be made in advance. Reservation will automatically removed 30minutes after the actual booking time.
* The currency will be in Singapore Dollar (SGD) and 7 % Good and Services Tax (GST) and service charge must be included in the order invoice.
* Once an order invoice is printed, it is assumed that payment has been made and the table is vacated.
* Customer with membership card will be entitled to discount
* There is no requirement for access control and there is no need for authentication (login/logout) in order to use the application

# 2. Design Considerations

## 2.1 Approach Taken

For this OOP project, our team have applied OOP concepts in our design and implementation of the RRPSS application, to avoid bad design principles.

Multi-tier architecture or also known as N-tier architecture provides multiple benefits, therefore, the team decided to design the code to have a similar layout as N-tier architecture. N-tier architectural style involve dividing an application into three different tiers, presentation tier, logic tier and data tier. The higher tier or layers will have employ services provided by lower tier, but lower tier is independent of higher tier. This approach will allow us to have better flexibility in the future as modification to a specific tier will not affect the rest of the tier. Our team aims to make our application easy to maintain and extend.



*Figure 1: 3 Tier Architecture*

# 3. Design Principles Used

## 3.1. Single Responsibility Principle (SRP)

SRP states that each class in the program should only have a single responsibility. Each class will have its own specific role/job to do, therefore it should only have one reason to change. Classes in our program are defined such that attributes and methods that are directly related to the task/role that the class supposed to do, thus achieving the single responsibility principle. This will limit the ripple effect to other classes if modification is done to an existing class. For example, in our RRPSS, our “ControllerReservation” class is only responsible for all the task related to making or deletion of reservation. It manages attributes such as the list of completed reservation and ongoing reservation. Another such controller class that has its own role is, “ControllerOrder”, which is only concerned with the modification of orders.

Text

Description automatically generated

*Figure 2: Compact Version of ControllerReservation Class*

## 3.2. Open-Closed Principle (OCP)

The purpose of OCP is to ensure that classes, modules and function should be available for extension but closed for modification. This allows us to be able to write new functionality or features without having the need to go into the source code and modify it. For example, in our program we have “Promotion” and “Meal” that inherits from “Menu” which is an abstract class. It allows “Promotion” and “Meal” class to be able to access to the common method in “Menu”.

## 3.4. Don’t Repeat Yourself (DRY)

DRY states that every piece of knowledge must have a single, unambiguous, authoritative representation within a system. This means that there should be no duplication of code and functionality, encouraging code reuse and efficiency. In order to achieve DRY principle, we have classes that uses inheritance which allows us to be able to reuse attributes or methods from the parent class. For instance, our class “Meal” extends “Menu”, which allows “Meal” class to have access to attributes such as name, price, description without the need to declare them again. (Shown in Figure 2) In addition, it allows “Meal” class to be able to reuse “Menu” method such as getName(), setPrice(), getDescription() even though it is not declared in its own class. In the “ControllerMeal”, “Meal” class reused method from “Menu” class. (Shown in Figure 3)

Text

Description automatically generated

*Figure 3: “Meal” Class Using Attributes From “Menu”*

Graphical user interface, text, application

Description automatically generated

*Figure 4: “ControllerMeal” Class, “updateMealName” method*

# 4. UML Class Diagram

*Figure 5: UML class diagram for the RRPSS application (Better resolution image may be found in submission folder)*

# 5. UML Sequence Diagram

Graphical user interface, application, Word

Description automatically generated

*Figure 6: UML Sequence Diagram for “Check/Remove Reservation Booking” function*

*(Better resolution image may be found in submission folder)*

# 6. Black Box Testing

|  |  |  |
| --- | --- | --- |
| **Test No.** | **Test Scenario** | **Test Outcome** |
| **1** | Making a reservation that is based on a previous date  Here, the current date and time used is:  2021-11-13 12:15 | Unsuccessful reservation:  Successful reservation: |
| **2** | Ordering an item not on the menu |  |
| **3** | Removing an item not on the menu |  |
| **4** | Printing invoice of a table that is not occupied |  |
| **5** | Creating a table of odd size / larger than max size (10) OR smaller than min size (2) |  |
| **6** | Remove an existing reservation | Before:  After: |

# Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Course** | **Lab Group** | **Signature / Date** |
| Lim Jie Xian | CZ2002 | SS5 |  |
| Ahmad Aleena | CZ2002 | SS5 |  |
| Anandarajan Sindini | CZ2002 | SS5 |  |
| Tan Pei Lun | CZ2002 | SS5 |  |
| Tay Lu Kang | CZ2002 | SS5 |  |