

**CZ2002 Object Oriented Design & Programming**

**Assignment Submission**

**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 and Date |
| Roxas Abby Maurea Imus | CZ2002 | FSP7 | Abby / 19/4/19 |
| Karen Wong Ke Xin | CZ2002 | FSP7 | KarenWkx / 19/4/19 |
| Koh Liang Jing | CZ2002 | FSP7 | LiangJing / 19/4/19 |
| Rachael Neoh Li Yii | CZ2002 | FSP7 | Rachael / 19/4/19 |
| Song Bogyeong | CZ2002 | FSP7 | Bogyeong / 19/4/19 |

1. **Design consideration**

The application follows an MVC architecture so that each components and classes are well-divided and defined, making code easier to understand and maintain. Overall, our the application is divided into 5 key areas: Menu, Reservation, Order, Sales and a main application for user to interact with.

The logic behind each area is available in the ‘Controller’. Each controller is independent and perform only the tasks assigned to it such that one ‘controller class’ contains one key responsibility. For instance, the maindmenucontroller containd tasks associated with the features required for menu items and promotional packages. While reservation controller is responsible for creation, viewing and deletion of reservation only. With this design, it complies with the ‘S’ in ‘SOLID’ design principle where the impact of change can be minimized when tasks are separated and there is no ‘God’ classes. Aside from that, with the division of classes, the application achieves loose coupling when dependencies are not so complicated to form a network design.

Some of the classes are designed with ease of extension and inheritance. For example, the mainmenu class can be easily used to extend to other types of menus other than the ala-carte and promotion packages used currently. This applies for the MenuApp as well. As it contains a general function and is current used to extend to orderApp, reservationApp. This provide the opportunity to extend beyond those classes. With this, it provides a certain degree of extensibility and meet the description of ‘O’ in SOLID’ design principle by fulfilling the ability to be close for modification while open for extensibility.

The ‘L’ stands for liskov substitution specific the pre and post conditions in the function. The application defines certain precondition for various functionalities in the application. For example, there are authentication in place to restrict staff access for certain functionalities. This include the condition that only staff with the assigned role “Manager” is able to add/delete menu items and promotional packages as well as to view the sales report.

In the menucontroller, it will do a Staff ID check to make sure role is manager and print out the feature the staff ID is able to access to before there are able to clear the requirement to access certain feature. Or else there will be error messages to restrict access if the non-managerial staff is trying to access restricted function. With this restriction, it increases the security so that not all staffs are able to gain access to confidential information.

The derieved class, MenuApp, OrderApp and ReservationApp can be used to use to replace the base class AppClass when conditions are met such as the function is being called.

As for the “I’ for interface segregation principle (ISP), there are 2 interfaces in this application, serialization to retain the data and comparable to sort the data according to display needs. They are only use in required classes. Such classes include mainorder, reservation, reservation booking, customer, order invoice for serialization. While ala-carte and promotion package for comparable. This is to avoid FAT interfaces and avoid having it in classes that does not require them.

Lastly, certain classes contain encapsulation of data. Such that there is control in place to mask the data whenever required. Such as staff validation to restrict confidential information from being accessed by unauthorized personnel. This contribute to data protection.

1. **Proposed features**

As part of the system enhancement efforts, 2 new features are stated below which would be integrated with current system using OO design principles.

**2.1 Loyalty points system**

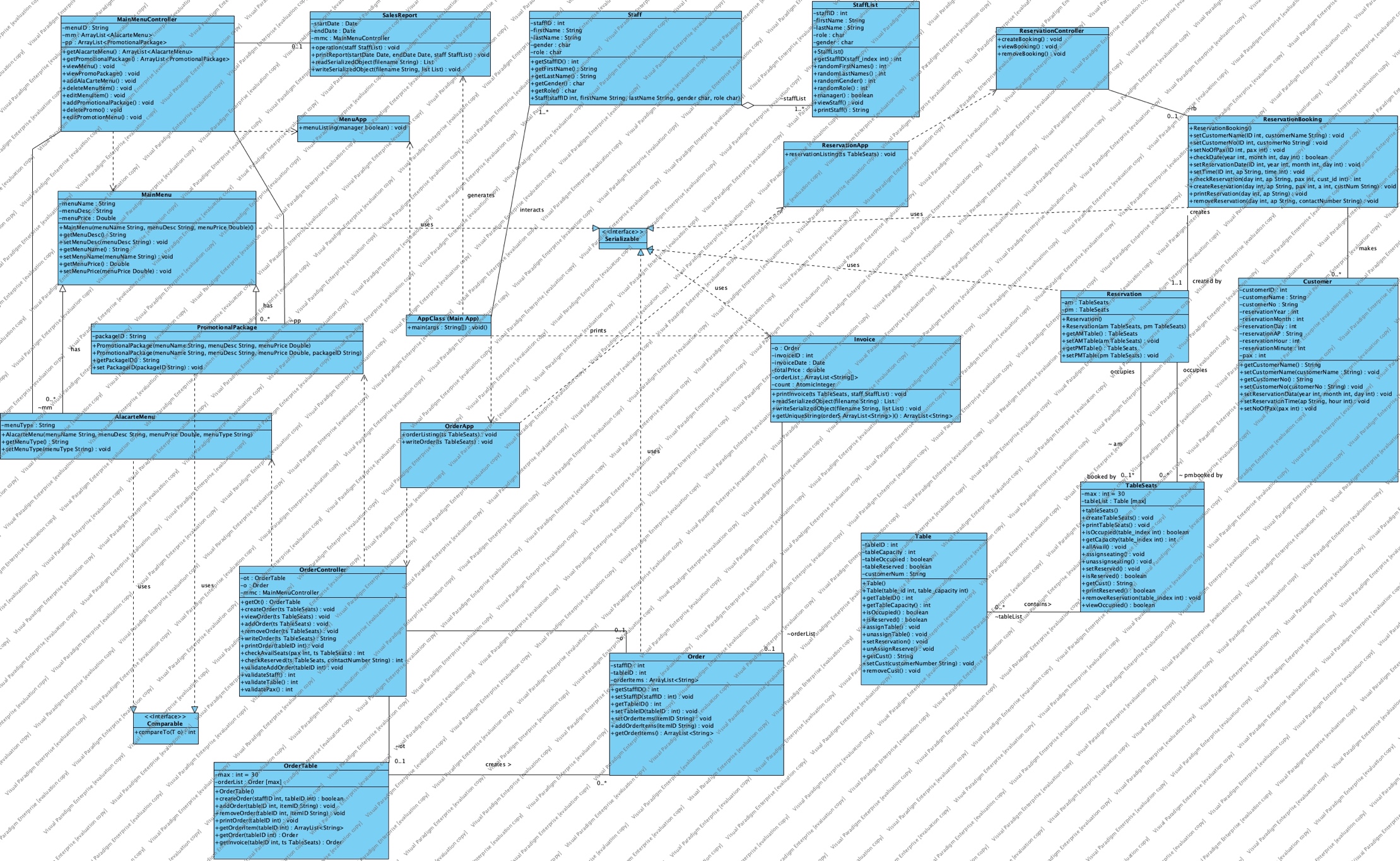
Allow customers to accumulate points after payment if they place reservation. Points can be used to offset subsequent purchases. Our current system can integrate with new feature by having an extension from customer class with an additional attributes such as customer name, customer ID and loyalty point count. As an extension is possible, it is extensible and it can reuse most of the function of the parent class. As there is a need to convert the invoice amount to point, it is required to link with invoice class to obtain the necessary information.

* 1. **Delivery service system**

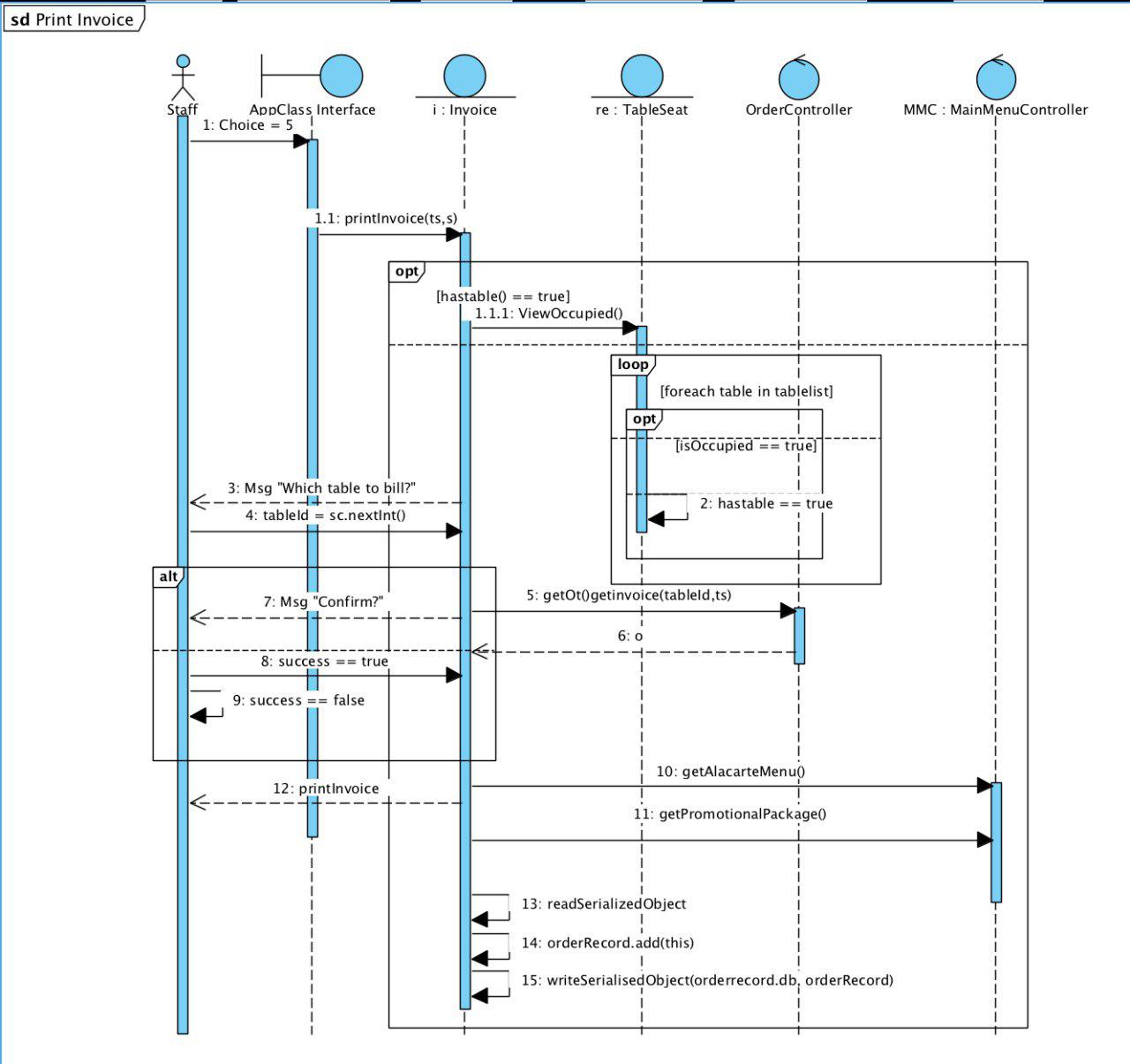
Aside from reservation and dine in orders, this feature allows customer to order food and have it delivery to their location or have it self-collected at a predetermined date and time. For this feature, we will be creating a new class called “Delivery” which will be extended to Order class. So Delivery class can call its parent class function of create order. Instead of assign a table to it, it will prompt for customer information, delivery address, date and time. This ensures code reusability.

Overall, as both functions apply the same method of inheriting the parent class with its addition features. It have only one dependency and so there will be minimum changes. Each classes contain only one responsibility and adhere to ‘S’ principle.

**2. UML diagram – Class diagram**



**3.1 UML diagram – Sequence diagram**



**3.2 Sequence diagram – use case description for “print bill” function**

The function for print bill will begin when the staff approach the table and use the RPSS system to access the print invoice function. In the main application (Appclass.java) staff can enter option 5 to print invoice. The application will do a check to find unbilled table with valid order and display it on the application. If there is no unbilled table, there will be a message to inform staff and revert back to main application.

Staff will then select the table ID and system will fetch the order object based on table ID. The system will ask staff for confirmation to proceed with it and print out the invoice’s header. System will then go through a loop to find unique ID for menu item and promotion item and their respective details (E.g. price and description) and stored it in an array list. When the loop end, the application will go through the order array list created earlier on and print item out in the required format. It will also calculate the respective taxes where GST will be 0.07 and service charge to be 0.1 of the total order

**4. Test cases and result**

The test cases for the application’s key functionalities will be included in this section. Due to page limitation, only successful scenario test cases will be shown.

**4.1 Create/Update/View/Delete of menu items (ala-carte and promotion items)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case No:** 1 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | | |
| **Test tittle:** Create menu item (successfully created) and view the menu item list. | | | | |
| **Description:** Staff validation is required, where only manager is able to create new menu item. In this test, StaffID 7 is assigned a managerial role and is able to create and delete. | | | | | |
| **Test Steps:** | **Input data** | **Expected result** | | **Actual result** | **Pass/Fail** |
| Enter choice:  (in main menu)  ((1) Add/Update/View/Delete Menu Items)  Enter StaffID: | 1  7 | Prompt for staff ID  Display all the options for menu item’s display (1 -10). This include menu item and promotion item create option. |  | | Pass |
| Enter choice:  ((3) Add New Menu Item)  Enter menu type, ID (unique), price and description | 3  2  89  Steak with potato in country sauce  30 | Since the ID is unique, it will successfully create the menu item object.  Loop will continue to ask user for menu type until -1 is given. |  | | Pass |
| Enter choice:  (1) View Existing Menu | 1 | Display existing menu item. Ordered by menu types.  As seen, M89 is created and added in. |  | | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 2 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | |
| **Test tittle:** Create promotional package (successfully created) and view it | | | |
| **Description:** Assume staff is validated from previous validation (test case 1). This case test for successful creation of promotion package from existing menu items. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  ((7) Add New Promotional Packages) | 7 | Display existing menu item and prompt for ID |  | Pass |
| Ask for item to add into promotional package until -1 is received | P90  M17  M4  M14  -1 | Since the ID is unique, it will successfully create the menu item object.  Loop will continue to ask user for menu type until -1 is given.  Then, it will create a promotional package object. |  | Pass |
| Take a look at created promotional package  (choice = 6)  6) View all Promotional Packages | 6 | Created promotional package with assigned menu item to be reflected in the view promotion package table. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 3 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | |
| **Test tittle:** Edit menu item. | | | |
| **Description:** Assume staff validation is completed where user is a manager. This test case will try to modify a menu item (M14’s description, price and type). | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  ((4) Edit Menu Item)  Edit Option:  Enter description to change into: | 4  1  “Potato Sauce salad ^^” | Display current menu ID list and ask for edit option.  According to the choice, ask for the input change. |  | Pass |
| Edit Option:  Enter price: | 2  3 | Display and prompt for price change. |  | Pass |
| Edit Option:  Enter type: | 3  1 | Display and prompt for type to change. |  | Pass |
| Check to see if changes are reflected | 1 | Display menu item.  Where price = 3.0, description = “Potato Sauce salad ^^” And it moves up to be displayed along with the other appetizers as item is grouped according to type. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 4 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | |
| **Test tittle:** Edit promotion item. | | | |
| **Description:** Assume staff validation is completed where user is a manager. This test case will try to modify a promotional item. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  ((8) Edit Promotional Packages  Promotion package: | 8  P90 | Display current Package ID list and ask for edit option. |  | Pass |
| Select edit action:  Enter menu to be added in | 1  M5 | Display and prompt for price change. |  | Pass |
| Edit Option:  Enter type: | 2  M14 | As loop is not complete, select 2 to delete M14 from the package P90 |  | Pass |
| Check to see if changes are reflected | 1 | Display promotion item.  Where M5: honey chipotle wings is added to P90 and M14: potato salad is not included in the package. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 5 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | |
| **Test tittle:** Delete menu item. | | | |
| **Description:** Delete menu item | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (5) Delete Menu Item  Enter item to delete: | 5  M13 | Delete menu item |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 6 | **Module/Function Name:** (1) Add/Update/View/Delete Menu Items | | | |
| **Test tittle:** Delete promotional package | | | |
| **Description:** Delete promotional package. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  |(9) Delete Promotional Packages  Enter item to delete: | 9  P90 | Delete selected package. |  | Pass |

**4.2 Check table availability**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 7 | **Module/Function Name:** (2) Check Table Availability | | | |
| **Test tittle:** Check table availabilty | | | |
| **Description:** Check unoccupied table list. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (2) Check Table Availability | 2 | Show all available table and seats. |  | Pass |

**4.2 Reservation related (create/view/delete)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 8 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Create reservation (successful) and check it | | | |
| **Description:** Create reservation. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (4) Create/Check/Remove Reservation  Enter choice:  (2) Create new reservation  Enter reservation details | 4  2  2019,5,1. AM, 1pm, 4, Karen, 88776729 | Prompt for reservation details. |  | Pass |
| Check if reservation is made,  Enter choice  (1) Check Existing reservation | 1 | Display reservation details. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 9 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Create reservation (unsuccessful) | | | |
| **Description:** Test for unsuccessful creation when user try to book more than 1 month in advance/ invalid date. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (4) Create/Check/Remove Reservation  Enter choice:  (2) Create new reservation  Enter reservation details | 4  2  2020,5,1 | Prompt for reservation details.  As it is more than 1 month from today, it will display error message |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 10 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Delete reservation | | | |
| **Description:** Test to delete reservation list. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (3) Remove Existing Reservation  Enter reservation details | 3  1, AM, 88776729 | Prompt for reservation details.  Removed the reservation entry |  | Pass |
| Enter choice:  (1) Check Existing reservation  Enter reservation details | 1  1, AM | Return the day reservation list which will be empty. |  | Pass |

**4.3 Order creation/view/delete (include checking for table availability)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 11 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Create order without reservation and view it | | | |
| **Description:** Create order for walk in guest. Assume that customer are going to order a mix of promotion package and ala-carte items. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (3) Create/View/Edit Order  Enter choice:  (2) Create new order  Enter reservation?  Enter no.pax | 3  2  False  3 | Customers will be assigned a table based on number of people and seats. For this case, it will be assigned a table with 4 seats. (table 11 as it’s the first for the 4 seats table |  | Pass |
| Enter staffID to begin order | 8 | Display list of menu item and package and a loop to ask for the order based on number of pax indicated. (only apply for package items) |  | Pass |
| Enter order item | P4, P2, M18, M16, -1 | Keep asking until -1 is received. |  | Pass |
| Enter choice:  (1) View Existing Order  Enter table ID | 1  11 | Display order items |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 12 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Remove order item from existing order | | | |
| **Description:** Remove one order item from exiting order created in test case 11. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (3) Create/View/Edit Order  Enter choice:  (4) Remove Existing Order Items  Enter table ID:  Enter menu id to remove | 3  4  11  M18 | M18 will be removed from order for table 11. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 13 | **Module/Function Name:** (4) Create/Check/Remove Reservation | | | |
| **Test tittle:** Add order item from existing order | | | |
| **Description:** Add one order item from exiting order created in test case 11. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (3) Create/View/Edit Order  Enter choice:  (3) Add New Order Items  Enter table ID: | 3  3  11 | Display list of menu items |  | Pass |
| Enter item to add: | P4  -1 (to end loop) | P4 is added to order for table 11. |  | Pass |

**4.4 Printing Invoice**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 14 | **Module/Function Name:** (5) Print Invoice | | | |
| **Test tittle:** Print bill for existing order. | | | |
| **Description:** Print bill for table 11’s order created in previous test cases. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (5) Print Invoice | 5 | Display table with order. |  | Pass |
| Enter table ID and confirm billing | 11  true | Display the bill with details such as date, time, item, amount price. Including GST and service charge. |  | Pass |

**4.5 Printing Sales report by period of month/year**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 15 | **Module/Function Name:** (6) Print Sales Revenue | | | |
| **Test tittle:** Print sales revenue function (unsuccessful) | | | |
| **Description:** Only manager can access this function. This test case test if a non-managerial staff can print it. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (6) Print Sales Revenue  Please enter your StaffID: | 6  6 | Staff ID 6 is a waiter and hence, not able to access this function. It will display error message to restrict access. |  | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case No:** 16 | **Module/Function Name:** (6) Print Sales Revenue | | | |
| **Test tittle:** Print sales revenue function | | | |
| **Description:** Sales report can be printed daily/ monthly or yearly. In this test case, it will print daily, then monthly then yearly. | | | | |
| **Test Steps:** | **Input data** | **Expected result** | **Actual result** | **Pass/Fail** |
| Enter choice:  (6) Print Sales Revenue  Please enter your StaffID: | 6  10 | Staff ID 6 is a manager and hence able to access it. It will ask whether user want daily/ monthly/ yearly report. |  | Pass |
| Enter option: | 1 (daily report) | Display the sales for daily report by date. |  | Pass |
| Enter option:  Enter start month period (MM/yyyy):  Enter end month period (MM/yyyy): | 2 (monthly report)  03/2019  04/2019 | Display the sales report by month. |  | Pass |
| Enter option:  Enter start year (yyyy):  Enter end year (yyyy): | 3 (Yearly report)  2018  2019 | Display the sales report by year. |  | Pass |