Restaurant Project Report

(Little India)

**Project Advisor**

Ahmad Reza Madabadi

**Group # 6 Project Members**

Rajbir Kaur

Varinder Kaur

Manpreet Singh

Amarjeet Singh

Raman Kumar Verma

1. **Individual Contributions**

All team members contributed equally.

Name Id Work Description

Rajbir Kaur 1892662 Work on Wireframe

Varinder Kaur 1893208 Work on Wireframe

Manpreet Singh 1892880 Work On Prototype

Amarjit Singh 1892781 Work On Prototype

Raman Kumar Verma 1892847 UML, case diagrame & Designing.

**2. Table of Contents**

1. Summary of Changes……..……………………………………………………….........

2. Customer Statement of Requirements……………………………………………….....

1. Glossary of Terms…………….………………………………………………………...
2. Functional Requirements Specification…..………………………………………….....
3. Nonfunctional Requirements………………………………………………………….
4. Domain Analysis……………..………………………………………………………..
5. Interaction Diagrams…………………………………………………………………..
6. Class Diagram…………………………….………………………………………......
7. User Interface Design and Implementation………………………………………….
8. Conclusion and Future Work……………………………………………………….
   1. **Summary of Changes**

* Addition of new Use Cases in the Functional Requirements Section
* Elaborated on existing Use Cases
* Corrections of Nonfunctional Requirements
* Domain Analysis has been corrected to include the new Use Cases
* Interaction Diagrams have been updated and fixed. The new Use Cases have been diagramed.
* Addition of Design Patterns to existing Interaction Diagrams
* Class Diagrams updated to reflect all the new Classes
* Classes have been mapped to their appropriate Subsystems
* Explanation of User Interface has been expanded upon and reasoning for certain choices has been made clearer

**4. Customer Statement of Requirements**

**Aim:**

The aim of this project is to develop a app that would eliminate the need of traditional pen/paper approach for privately- owned restaurants. The project is focused on making the restaurant fully automated such that it is easier to coordinate various work activities that go on inside a typical restaurant. The main features of the project include:

* Coordinating work activities of the various actors – Host, Waiter, Cook, Busboy and Manager
* Increase efficiency by minimizing time between an order is placed and the billing
* Increase profits by reducing operating costs and increasing revenues by increasing efficiency
* Archiving information of the workers and hours worked

**Problems with the primitive system:**

The traditional pen paper approach has the following drawbacks:

* Keeping track of empty tables requires either keeping a “dry erase” diagram of tables or the host constantly keeps track of the status of the tables if it is a small restaurant
* The waiter jots the order on paper and has to transfer redundant information to the terminal system. This takes some time and reduces efficiency in peak hours of patron service, also there may not be enough terminals available in the restaurant as there are number of tables which would require particular waiters to wait until the others are done entering their orders into the systems
* The cooks could not notify the waiter that the food was ready
* Keeping billing and other statistical information was an issue of concern

The manager has administrative power over employee profiles. They can do the following:

1. The ability to create and modify profiles
2. Track employee activities
3. Authorize restricted waiter activities.

We will take into account the number of clicks that are necessary to accomplish the individual tasks and try to minimize the number of clicks for efficient deployment of our system.

**5. Glossary Of Terms Used**

• Manager - Manages inventory, payroll, employee list and charts and statistics for the restaurant

•

Host

- Assigns and seats people who come to the restaurant

* Waiter – Takes the order from the customer onto a PDA and delivers the order to the customer
* Cook - Reads the order placed from a terminal in the kitchen and cooks food accordingly. Also the cook informs the waiter when the order is ready.
* Add/Edit Employee – Button on the Management page to add or edit the information of an employee at the restaurant such as employee identification number, their password, employee type and their wages.
* Reports Screen – statistical data analysis of the traffic flow in the restaurant
* Grid – GUI layout of the tables in the restaurant
* Efficiency analysis – Performance measure of all waiters in the restaurant

**6. Functional Requirements Specification**

**Stakeholders**

The following are the stakeholders in the application. They have a vested interest of some sort in the way that the application works. It is important to them that the application is easy to use.

* Owner of the restaurant
* Employees (Manager, Waiters,Cook, & Customers)

**Actors and Goals**

The table lists all the actors, their various goals and the related use cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Actor | Goal | Type | Use Case |
|  |  |  |  |
| Manager | Add new items to menu | Initiating | AddToMenu (UC-1) |
|  |  |  |  |
| Manager | Delete item from menu | Initiating | Delete from Menu (UC-2) |
|  |  |  |  |
| Manager | Graphical Analysis | Initiating | Graphs (UC-3) |
|  |  |  |  |
| Host | Seat customers and assign waiters | Initiating | Seat Customer (UC-4) |
|  | to tables |  |  |
|  |  |  |  |
| Cook | Prepare and cook the order | Initiating | Preparing Order (UC-5) |
|  |  |  |  |
| Waiter | Take orders | Initiating | Order (UC-6) |
| Waiter | Deliver order to table | Participating | Deliver (UC-7) |
| Waiter | Collect payment from customer | Initiating | PayBill (UC-8) |
|  |  |  |  |

**Use Case Casual Description**

These are the various use cases that are contained within the program. A quick description of all the use cases follows below.

1. AddtoMenu(UC – 1): Allows the manager to add new items to the menu list for the restaurant.
2. Delete from Menu(UC- 2): Allows the manager to delete items from the menu. When the manager wishes to discontinue an item in the restaurant he can delete that item from the list.
3. Graphical Analysis( UC- 3): Allows the manager accesses the database to create performance reports such as employee efficiency.
4. Seat Customer (UC-4) - The Host selects a table for the customers and assigns a waiter to them.
5. PreparingOrder (UC-5) – The cook prepares the order after the waiter gives it to him and after the food is prepared, the cook changes the status of the order to ready, indicating to the waiter that it can be delivered. Also, each time a food item is prepared the inventory is updated directly. The system will know the names and quantity of the items used and will update the inventory.
6. Order (UC-6) – The waiter takes the order from customers and gives it to the cook.
7. Deliver(UC-7) – After the waiter gets a ready signal from the cook indicating that the food is ready, he gets it from the kitchen and delivers it to the table.
8. PayBill(UC-8) – After the customers are finished with the food, the waiter gives them the bill and collects payment.

**Fully-Dressed Description**

|  |  |
| --- | --- |
| Use Case UC-1 | AddtoMenu |
|  |  |
| Primary Actor | Manager |
|  |  |
| Actor's Goal | Add new items to the menu database |
|  |  |
| Stakeholders |  |
|  |  |
| Preconditions | Menu database already exists. |
|  |  |
| Post conditions | New items are added to the existing |
|  | menu. |
|  |  |
| Main Success Scenario |  |
|  |  |

> 1. Manager selects the interface to enter in new item information into the menu database.

2. Validates data.

3. Item added to database.

|  |  |
| --- | --- |
| Use Case UC-2 | Delete from Menu |
|  |  |
| Primary Actor | Manager |
|  |  |
| Actor's Goal | Delete items from the menu |
|  |  |
| Stakeholders |  |
|  |  |
| Preconditions | The food item already exists in the |
|  | menu. |
|  |  |
| Post conditions | The food item is deleted from the |
|  | menu. |
|  |  |
| Main Success Scenario |  |
|  |  |

> 1. Manager selects the interface to delete the food item.

> 2. Manager selects food item from drop down menu.

3. Food item is then deleted from the system.

|  |  |  |
| --- | --- | --- |
| Use Case UC-4 |  | Seat Customer |
|  |  |  |
| Primary Actor |  | Host |
|  |  |  |
| Actor's Goal |  | Seat a customer at a clean empty table |
|  |  | and assign a waiter to that table |
|  |  |  |
| Stakeholders |  | Employee |
|  |  |  |
| Preconditions |  | There is a new customer who hasn’t |
|  |  | been seated yet. A waiter hasn’t been |
|  |  | assigned to that table yet. |
|  |  |  |
| Postconditions |  | Customer is seated and a waiter is |
|  |  | assigned to the table. |
|  |  |  |
| Main Success Scenario |  |  |
|  |  | |
| > 1. Host selects the view table interface, and selects a table. | | |
| > 2. Host assigns waiter to table. | |  |
| 3. System changes table status to occupied. | |  |
|  |  |  |

|  |  |
| --- | --- |
| Use Case UC-5 | Preparing Order |
|  |  |
| Primary Actor | Cook |
|  |  |
| Actor's Goal | Prepare food for the order and notify |
|  | the waiter when it is ready |
|  |  |
| Stakeholders | Waiter |
|  |  |
| Preconditions | An order has been made by a customer |
|  | and the waiter has brought it to the |
|  | cook. |
|  |  |
| Postconditions | The waiter is notified when the food is |
|  | ready to be delivered to the table. Also |
|  | each time a food item is prepared, the |
|  | corresponding inventory items are |
|  | decreased by the pre-specified |
|  | quantity. |
|  |  |
| Main Success Scenario |  |
|  |  |

> 1. The cook selects one of the pending orders from the database and clicks on item to cook.

2. System changes the status of the item to ‘ready’

< 3. The waiter is then notified by the system that the order is ready.

|  |  |
| --- | --- |
| Use Case UC-6 | Order |
|  |  |
| Primary Actor | Waiter |
|  |  |
| Actor's Goal | Place the order in the queue for the |
|  | cook to prepare. |
|  |  |
| Stakeholders | Cook |
|  |  |
| Preconditions | There is an order to be placed |
|  |  |
| Postconditions | The order is placed in the queue to be |
|  | prepared |
|  |  |
| Main Success Scenario |  |
|  |  |

> 1. Waiter chooses table and clicks the “Place Order” button

< 2. System changes to the order page and displays only items that are in stock > 3. Waiter inputs the various items and quantity for the order and clicks the

|  |  |  |
| --- | --- | --- |
| “Place Order” button |  |  |
| 4. System stores items into the database. |  |  |
| Alternate Scenario: |  | If the customers choose to order more |
|  |
|  |  | food. |
|  |  | Exceptions: The order should not be |
|  |  | closed(Bill hasn’t been paid yet) |
|  |  |  |

> 1. If the customer wants to add more food and the bill hasn’t been paid yet, the waiter will still have access to the order page for that table and he will be able to make the necessary additions.

However, if the bill has been paid already, the new items will be associated with a different order ID.

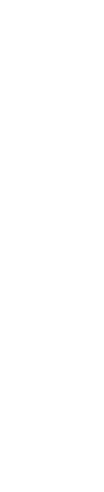
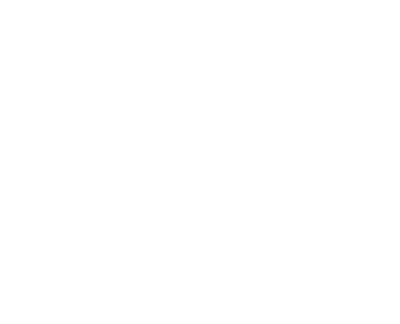
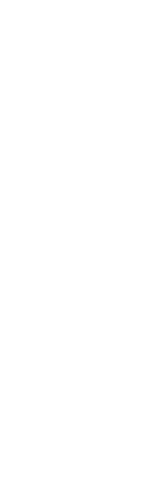
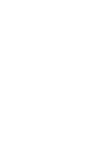
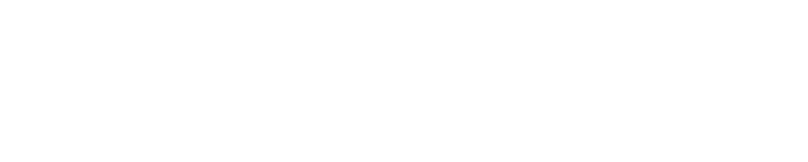
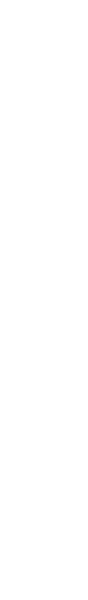
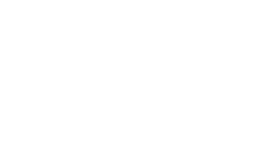
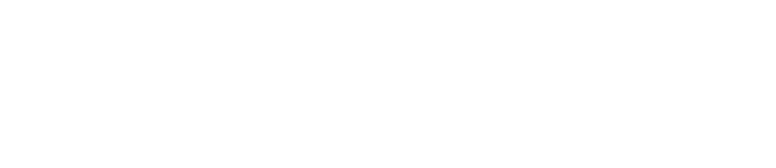
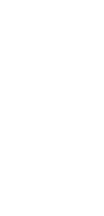
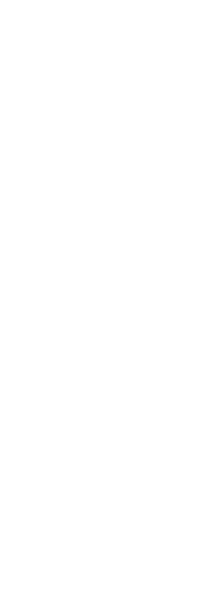
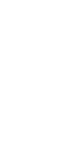
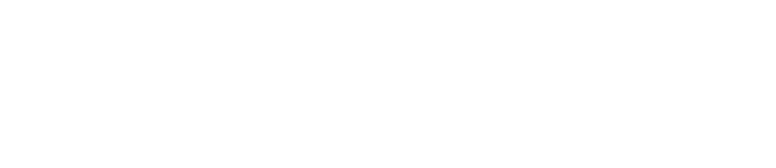
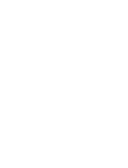
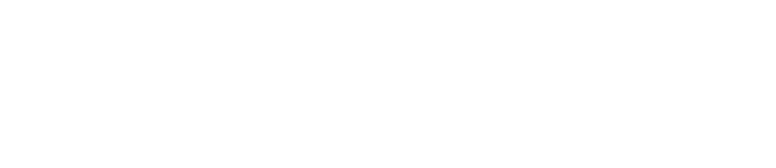
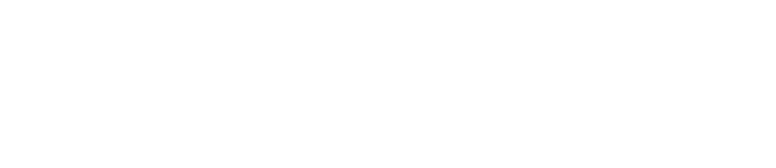
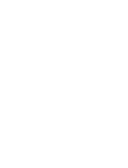
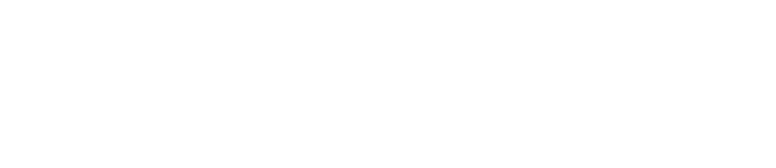
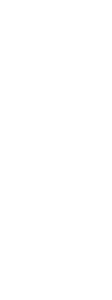
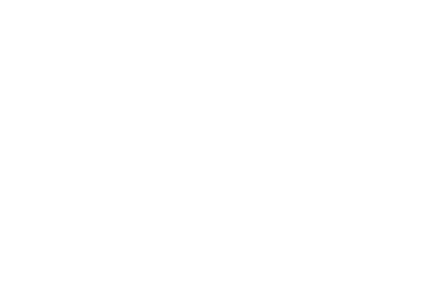
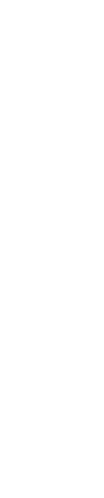
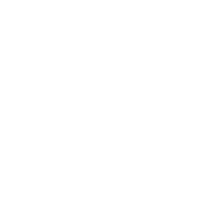
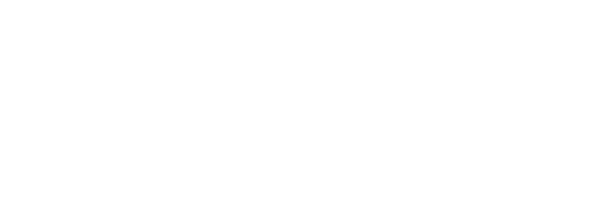
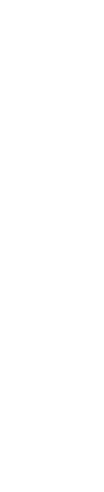
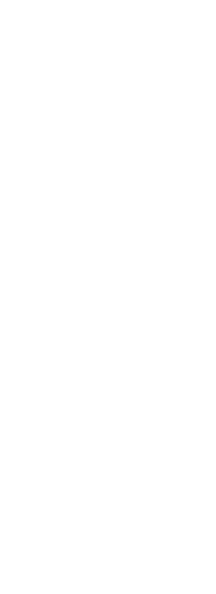
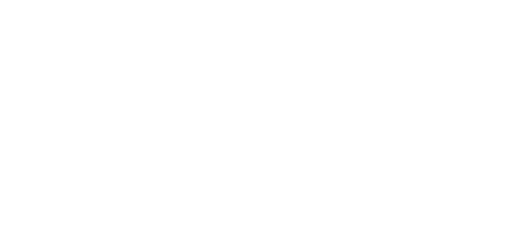
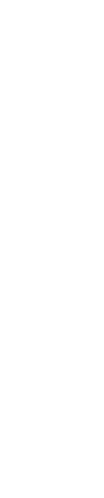
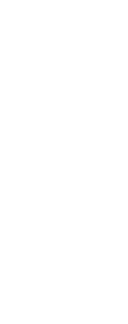
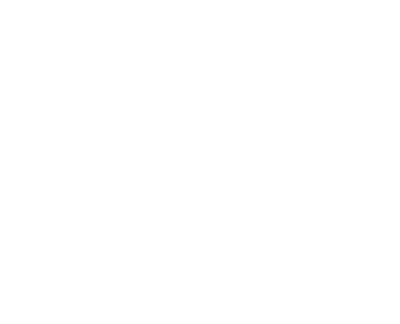
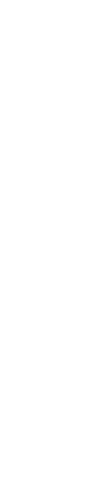
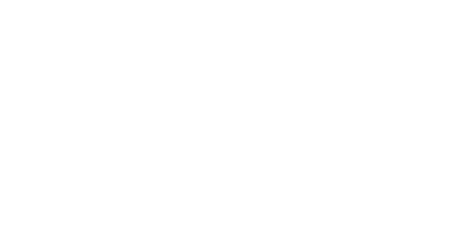
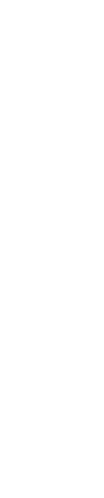
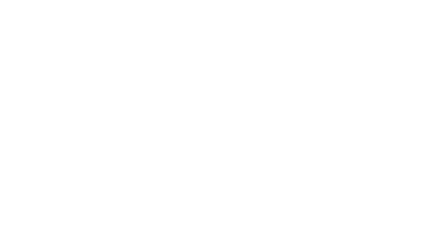
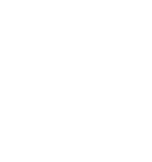
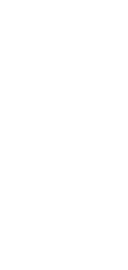
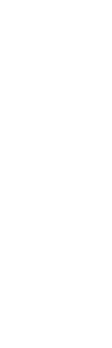
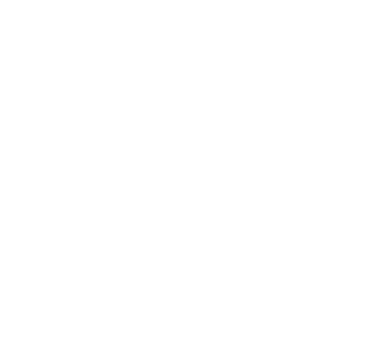
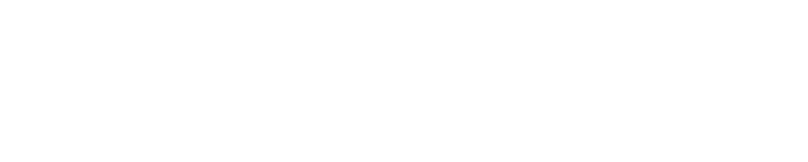
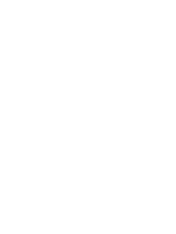
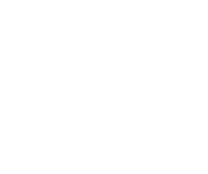
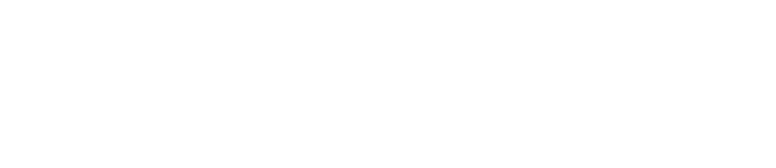
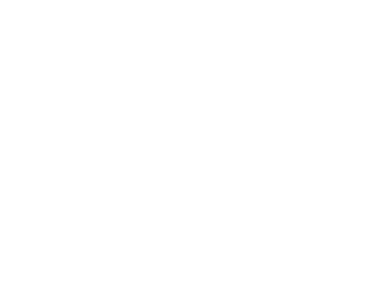
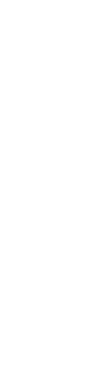
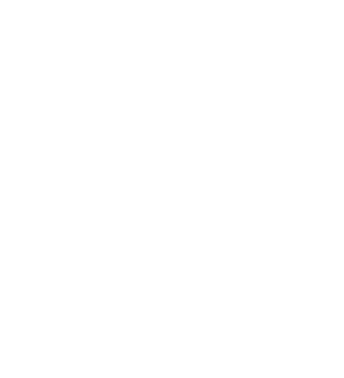
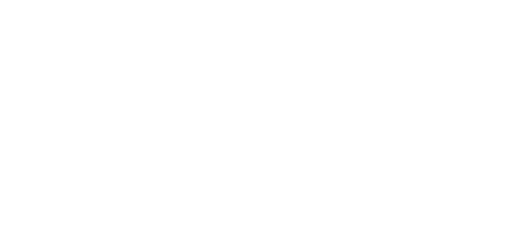
|  |  |
| --- | --- |
| Use Case UC-7 | Deliver |
|  |  |
| Primary Actor | Waiter |
|  |  |
| Actor's Goal | Deliver the food prepared by the cook |
|  | to the customers. |
|  |  |
| Stakeholders | Manager |
|  |  |
| Preconditions | The cook has already prepared the |
|  | food and it is ready for delivery. |
|  |  |
| Postconditions | The order is deleted from the cooks list |
|  | of orders to do. |
|  |  |
| Main Success Scenario |  |
|  |  |
| < 1. The waiter gets an indication from the system that an item is ready. | |
| > 2. Waiter clicks on item to deliver. |  |
| 3. System updates status of item. |  |
|  |  |

|  |  |
| --- | --- |
| Use Case UC-8 | PayBill |
|  |  |
| Primary Actor | Waiter |
|  |  |
| Actor's Goal | Place the order in the queue for the |
|  | cook to prepare. |
|  |  |
| Stakeholders | Manager |
|  |  |
| Preconditions | The customers have ordered food and |
|  | have finished eating it |
|  |  |
| Postconditions | The money has to be paid before the |
|  | customers leave the restaurant and the |
|  | table status has to be changed to dirty |
|  | by the waiter. |
|  |  |
| Main Success Scenario |  |
|  |  |

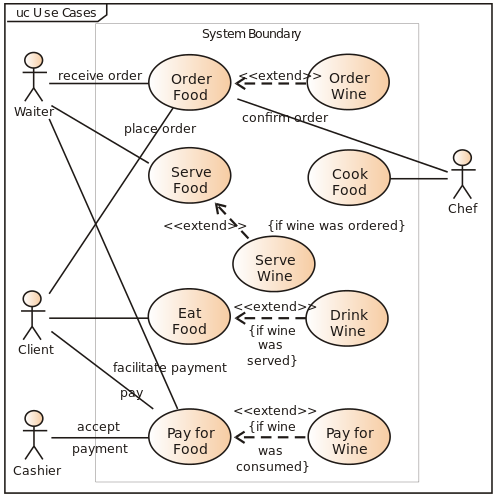
< 1. System generates check.

> 2. Waiter processes payment and notifies system.

< 3. System changes table status to dirty.



UML Diagram Of Restaurant



**7. Nonfunctional Requirements**

**Usability**

The system’s user interface will be very simple anself-explanatory. The manager and the cook will interact with the system through touch-screen LCDs. They are very easy to adapt to and use. The users just have to touch the option they want. The waiters will interact with PDAs. They will each be able to access the tables that they are responsible for. The menus for the restaurant will show up as drop-down menus on the PDAs. The waiters only have to select the item that the customer ordered from the drop-down list.

**Reliability:**

The system is guaranteed to be reliable. All inputs to the system will be selections from options that the system will show on the screen. Since the system will only provide valid options, there is absolutely no possibility that invalid inputs can be entered. Each user will have a unique username and password. This removes the risk of unauthorized access to the system. Also, users can only access parts of the system that they require for their job. For example, a waiter cannot access the inventory management or the payroll feature of the system. Those parts among a few others can only be accessed by the manager. All these constraints will ensure the reliability of the system.

**Performance:**

The system will be used by many employees of the restaurant at the same time and can handle it without any errors. However, for this to happen, the server at the restaurant should be able to handle all the traffic without creating any problems. The restaurant should also have a high speed wireless connection for the system to perform its best. The waiters will be using PDAs and will only be able to connect to the system with a wireless connection. Since some tasks like placing and cooking and an order are sequential, the waiter has to update the system with the order first before the cook will see it. In order to for this to happen efficiently, the internet connection needs to be fast so that changes are reflected on the website instantly without any delay.

**Supportability:**

The system will support changes that the restaurant might need to make in future. The manager will have the ability to modify items from the menu. He will also be able to add or delete users to the system for layoffs, retirements or new hires.

**Operations Requirements:**

The restaurant manager will be the sole administrator of the system. He/she can modify the database and add or delete users from the system. The manager can also compile reports to analyze the restaurant performance in general or to see the best selling item or to compare different waiters.

**Legal Requirements:**

We will give each of our clients a license to use the App.

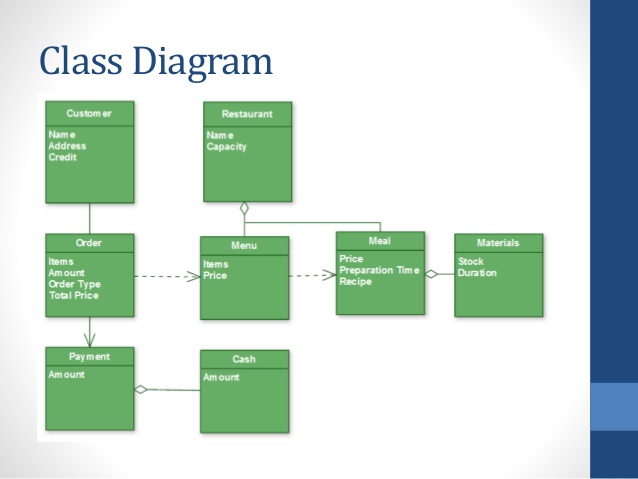


**System Operations Contract**

|  |  |
| --- | --- |
| **Name:** | Delete from Menu |
| **Responsibilities:** | Deletes an existing item from the menu. |
| **Type:** | System |
| **Exceptions:** | None |
| **Preconditions:** | Item exists in the menu. |
| **Postconditions:** | Item information is removed from the |
|  | menu and database. |
|  |  |
| **Name:** | Paycheck |
| **Responsibilities:** | Weekly paychecks are prepared, and the |
|  | information is also stored into the database. |
| **Type:** | System |
| **Exceptions:** | None |
| **Preconditions:** | The employee information exists in the |
|  | system; the number of hours are available. |
| **Postconditions:** | Information is calculated, returned, and |
|  | stored into the database. |
|  |  |
| **Name:** | Graphical Analysis |
| **Responsibilities:** | Performance charts are displayed and |
|  | stored. |
| **Type:** | System |
| **Exceptions:** | Information required to compute the graph |
|  | is not available and no graph can be |
|  | displayed. |
| **Preconditions:** | Information regarding the graph is |
|  | available and stored in the database. |
| **Postconditions:** | Graph is displayed, and stored in database. |
|  |  |
| **Name:** | Seat Customer |
| **Responsibilities:** | Table status is changed and waiter is |
|  | assigned |
| **Type:** | System |
| **Exceptions:** | Table is already full and new customer |
|  | cannot be seated there. |
| **Preconditions:** | The customer has not already been seated |
|  | and the table is empty. |
| **Postconditions:** | Customer is seated, waiter is assigned. |
|  |  |
| **Name:** | Preparing Order |
| **Responsibilities:** | Change the status of items, and notify |
|  | waiter when ready. |
| **Type:** | System |
| **Exceptions:** | None |

|  |  |
| --- | --- |
| **Preconditions:** | Order has been made. |
| **Postconditions:** | Waiter is notified as soon as food status is |
|  | changed. |
|  |  |

**Class Diagram:**





**Description**

Class diagrams are the mainstay of object-oriented analysis and design. UML class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes.

The simple UML class diagram above is the conceptual model of the restaurant automation system we will be implementing. Classes are depicted as boxes with three section, the top one indicates the name of the class, the middle list is for the attributes of the class, and the third one lists the methods. Some of the methods have a return type mentioned.

Some of the classes that will be used for our system are as follows:

* infoChecker
* employeeInfo
* MenuItemController
* Iteminfo
* Item Checker
* OrderController
* Order Database
* Order Info
* TableController
* TableStatusChecker

Explanation of some of the Classes Used:

1. infoChecker: This class is used to get employees information from the database and compare the information retrieved from the database to the information provided in the user interface. For example in case we want to delete an employee

from our database the infoChecker checks whether the Social Security Number for the employee exists in the Employee Database or not.

1. employeeInfo: This class is used to create an employee’s account, also to calculate the wage for that employee and cut a paycheck.
2. MenuItem Controller: This class controls the use cases that are applicable to the menu/food database. This controller is available to Manager, where the manager can Add new Items, or delete existing items from the database.
3. ItemInfo: This class is used to create a new item.
4. Item Checker: This class is used to get Food item information from the database and compare the information retrieved to the information provided in the user interface. For example in case we want to delete an item from our database, the item Checker checks whether the FoodID for the item exists in the Food Database or not.
5. OrderController: This class controls the use cases that are applicable to the orders database. This controller is available to the waiter and the cook. The manager can also access it, but usually does not need to add to it. The waiter adds new orders to the order table, and the cook sets the status symbol when the order is ready to serve.
6. Order Database: Has methods to deal with adding new order information.
7. Order In: This class is used to create a new order object.
8. TableController: This class controls the use cases that are applicable to the table database. This controller is available to the Host and the waiters. The manager can also access it but does not usually modify it. The host uses it to assign tables to new customers.
9. TableStatusChecker: This class is used to get Table information from the database and compare the information retrieved to the information provided in the user interface. For example if the Host needs to assign a table for some new customers, he can use this to check for a clean empty table.

**7. Conclusions and Future Work**

**Challenge:** The setup and implementation of the overall architecture was one of the mostchallenging tasks at the beginning of this project. The group has some trouble reaching a decision on the necessary functionalities required for different users of the restaurant automation model.

**Solution:** The used case and domain models discussed during the building blocks for thisproject. Using the fully dressed descriptions, we were able to implement use cases and modify them according to the individual needs of the host, waiter, manager, busboy, and chef. The domain model was crucial in tying everything together. It became evident that everything is interlinked and much of the code can be easily reused for different functions.

**Challenge:** When we began the implementation of the database and coding, the structureand organization of the class as well as their functionalities were very unclear. We were not sure of the required parameters and information required to implement the database. The various functionalities of different users caused some unnecessary repetitiveness in the database.

**Solution:** Using the class diagram and interface specification discussed in class, we wereable to coordinate and implement an organized database containing only the necessary information and eliminating unnecessary repetitiveness. This also made the distribution of tasks easier within the group as the classes and functionalities required had now become very clear.

**Challenge:** The next challenge we faced in the group was communication among theusers of the restaurant automation model (host, waiter, etc). We realized that when one

user enters the model and performs a certain task, it affects all other users. For example, when the host seats a customer, the waiter needs to acknowledge this and serve the table. When the waiter serves the table, the order needs to be sent to the chef. The chef then needs to alert the waiter when the food is ready, and finally, the waiter needs to alert the busboy at the end of the meal. This communication seemed tedious and caused some problems with shared access.

**Solution:** The Publisher-Subscriber pattern discussed during the course was a perfectimplementation for this project. This way, all the users would be subscribed to their related tasks, and when one user performs a task, all other users are automatically updated.



**Possible Directions for Future Work:** This model can be easily redefined for personalneeds of consumers (menu items, waiter functions, etc).