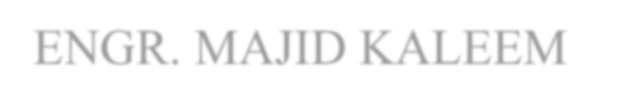
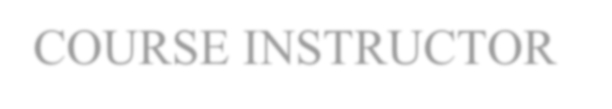
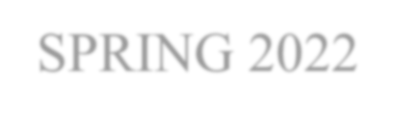


*Department of Software Engineering*



SEN-221-SOFTWARE DESIGN & ARCHITECTURE

SPRING 2022

COURSE INSTRUCTOR

ENGR. MAJID KALEEM

|  |  |
| --- | --- |
| Assignment No. | 04 |
| Assignment Title | Producing Design Models from Requirements |
| Course Learning  Outcome | CLO-05  “Design object-oriented design models to reflect implementation details.” |
| Full Name | M MUAZ SHAHZAD (02-131202-081)  SHAHAB TAHIR (02-131202-011)  TALHA ZAFAR (02-131202-011)  AWAIS ASAD (02-131192-062) |
| Semester | BSE 4 B |
| Submission Deadline | 3rd July 2022 |

* **INSTRUCTIONS:**
* Use only A4 size blank white paper for printing.
* Use only Times New Roman size 12 font.
* Do not use color printer to print this title page or assignment.
* Each heading (underlined, bold and in capital letters) and example must start from a new line.
* Submit hard copy of your assignment and upload softcopy on LMS as a DOCX file.
* Do not enclose your assignment in file/folder, staple assignment pages only at the left-top corner.
* Do not edit (this) assignment file given as a pdf file.
* Last page of your assignment must contain sources/references (use IEEE referencing style).
* Assignments will only be accepted in the scheduled class/room.
* No makeup assignments will be given.
* Violation of any of the instructions mentioned here will result in marks deduction.



BAHRIA UNIVERSITY (KARACHI CAMPUS)

Software Design & Architecture (SEN-221)

ASSIGNMENT # 4 – Spring 2022

Based on: CLO-5

Class: **BSE-4B** Submission Deadline: **29th June 21**

Course Instructor: **ENGR. MAJID KALEEM** Max Marks: **05**

1. In this task you are required to produce object-oriented design models to reflect implementation details from those requirements that you documented in SRS/user stories in your previous semester. You will submit your SRS as well along with this assignment.
2. You may complete this assignment in a group of maximum three members. Ideally those who were members in *Software Requirements Engineering*.
3. Viva/presentations regarding this assignment will be conducted individually. Each member must specify and present his/her role.
4. Produce *Use Cases* (diagrams) from the functional requirements (as mentioned in your SRS). In addition, use following use case template to describe your use cases.
5. Mention which types of *Software Architecture* (based on non-functional requirements) is suitable in the given scenario (as mentioned in your SRS). You are required to produce relevant diagram(s).
6. You are also required to specify the use of *Software Design Patterns* (with justifications) wherever applicable.
7. Justify why you would select a particular type of programing language to implement this.

**SOLUTION:**

# USECASE SCENARIOS:



Restaurant Management System

Book Table

Check/Deduct Stock

Waiter

Chef

<<include>>

Confirm Order

Order Food/Drink

Prepare Drink

Cook Food

<<include>> <<include>>

Staff

Bar Staff

Call Server

Customer

Serve Food/Drink

Eat Food

Pay for Meal

Pay

Accept Payment

<<include>>

Manager

Check Offers

Cashier

Generate Reports

Manage menu & ingredients

Order Stock

<<include>>

Update Stock Levels

<<include>>

Stock Check

Login

Collect Order

Place Order

Receive Order

The following tables document the use cases from Figure.

|  |  |
| --- | --- |
| **Use case ID:** | UC1 |
| **Use case name:** | Order Food/Drink |
| **Description:** | Order creation and food/drink selection. |
| **Preconditions:** | None |
| **Standard flow:** | 1. Order object created. 2. Item selected from the menu in the order system implementing UC3. 3. Order conﬁrmed and sent to kitchen application. |
| **Post conditions:** | Stock used deducted from the database. |

Use case 1 Order Food/Drink.

|  |  |
| --- | --- |
| **Use case ID:** | UC2 |
| **Use case name:** | Check/Deduct Stock |
| **Description:** | Method to deduct stock on selection of menu item. |
| **Preconditions:** | Implemented by UC1. |
| **Standard flow:** | 1. Stock checked before any item selected; options disabled if empty. 2. Item selected from the menu in the order system. |
| **Post conditions:** | Stock used deducted from the database. |

Use case 2: Check/Deduct Stock.

|  |  |
| --- | --- |
| **Use case ID:** | UC3 |
| **Use case name:** | Prepare Drink |
| **Description:** | Drink prepared by bar staﬀ. |
| **Preconditions:** | UC2. |
| **Standard flow:** | 1. Order retrieved. 2. Drinks prepared. |
| **Post conditions:** | Waiter called; implementing UC6. |

Use case 3: Prepare Drink

|  |  |
| --- | --- |
| **Use case ID:** | UC4 |
| **Use case name:** | Prepare Food |
| **Description:** | Food prepared by Chef. |
| **Preconditions:** | UC3. |
| **Standard flow:** | 1. Order retrieved. 2. Chef conﬁrms order has started. 3. Food is cooked. |
| **Post conditions:** | Waiter called; implementing UC6. |

Use case 4: Prepare Food.

|  |  |
| --- | --- |
| Use case ID: | UC5 |
| **Use case name:** | Call Waiter |
| **Description:** | Waiter called when food or drink is complete. |
| **Preconditions:** | Implemented by UC4 |
| **Standard flow:** | 1. Waiter called to inform either food or drink is ready. |
| **Post conditions:** | Waiter collects the dishes |

Use case 5: Call Waiter.

|  |  |
| --- | --- |
| **Use case ID:** | UC6 |
| **Use case name:** | Serve Food/Drink |
| **Description:** | Waiter serves food/drink. |
| **Preconditions:** | UC5. |
| **Standard flow:** | 1. Waiter serves food/drink once waiter called. |
| **Post conditions:** | No post conditions. |

Use case 6: Serve Food/Drink.

|  |  |
| --- | --- |
| **Use case ID:** | UC7 |
| **Use case name:** | Pay |
| **Description:** | Customer pays for the food. |
| **Preconditions:** | UC6 |
| **Standard flow:** | 1. Order completed. 2. Price calculated. 3. Discounted price calculated by implementing UC10. 4. Customer pays. |
| **Post conditions:** | No post conditions. |

Use case 7: Pay

**2**. **PROPOSED ARCHITECTURE FOR THE SYSTEM:**

MVC or Model View Controller architecture was used to develop this system. As a software design pattern for developing web applications MVC is popular.

MVC architecture divides web application in to three parts. All those parts are interconnected. It is fully capable to support rapid web application development and dynamic interactivity with the database.

**MODEL:**

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**VIEW:**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**CONTROLLER:**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

**3**.  **SOFTWARE DESIGN PATTERNS IMPLEMENTED:**

We have implemented a food delivery application by using different types of design patterns. The below design patterns are involved according to module,

1. User sees Menu Using Iterator Pattern
2. Adding food items for order using Builder Design Pattern
3. Order and Cancellation of food items using Command Design Pattern
4. Track an order in real time using Observer Pattern

**MODULE 1 USER SEES MENU - ITERATOR PATTERN**

The Iterator Pattern provides a way to access elements of an aggregate object sequentially without exposing the underlying structure of the object. I have used this design pattern for the displaying of food menus based on the restaurant because different food items form a menu, and the possibility of multiple traversals can be there, and I don't want to expose its internal representation to the outside.

**MODULE 2 ADDING FOOD ITEMS FOR ORDERING - BUILDER DESIGN PATTERN**

The builder pattern is a creational design pattern used to configure and assemble complex objects. It provides a way to create the same object with different kinds of objects. I have used this design pattern for adding food items because different kinds of food items create one meal.

**MODULE 3 ORDER OR CANCELLATION OF FOOD - COMMAND DESIGN**

Command design pattern falls under behavioral design pattern. It wraps the request under an object that is called Command. That command object is being passed to invoker object. I have used this design pattern because user can order or cancel food menus for delivery. Two different kinds of requests can be done by the user for food ordering.

**MODULE 4 TRACK AN ORDER IN REAL TIME - OBSERVER DESIGN PATTERN**

Observer design pattern is used in a situation where an object should be able to notify other objects about state change. I have used this pattern for the development of tracking an order in real time because whenever the food delivery states something like "food is packed for delivery, " "delivery boy is on the way," "It will be there in two minutes," "food delivered" or "food canceled,” the user will be notified automatically.

**4.** **PROGRAMING USED LANGUAGE TO IMPLEMENT (RMS):**

* HTML – Hyper Text Markup Language is used to build the base interfaces of the system
* CSS – Cascading Style Sheets was used to make the system more attractive and user friendly.
* C# –used to build the system
* JavaScript- this is a programing tool to web development and used for validations.
* SQL SERVER – Is a Relational Database used to store and retrieve data

SOFTWARE REQUIREMENTS SPECIFICATION

for

< FOODIES RESTAURANT >

Prepared by

**M MUAZ SHAHZAD 02-131202-081**

**SHAHAB TAHIR 02-131202-011**

**TALHA ZAFAR 02-131202-80**

**AWAIS ASAD 02-131192-062**

<27th JAN 2022>

Table of Contents

Table of Contents ii

Revision History **Error! Bookmark not defined.**

1. Introduction 1

1.1 Purpose **Error! Bookmark not defined.**

1.2 Document Conventions 2

1.3 Intended Audience and Reading Suggestions 2

1.4 Product Scope 2

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 3

2.7 Assumptions and Dependencies 4

3. External Interface Requirements 4

3.1 User Interfaces 4

3.2 Hardware Interfaces 5

3.3 Software Interfaces 5

3.4 Communications Interfaces 5

4. System Features 5

4.1 System Feature 1 6

4.2 System Feature 2 (and so on) 7

5. Other Nonfunctional Requirements 16

5.1 Performance Requirements 16

5.2 Safety Requirements 17

5.3 Security Requirements 17

5.4 Software Quality Attributes 17

5.5 Business Rules 18

Appendix B: Analysis Models 19

Appendix C: To Be Determined List 21

# INTRODUCTION:

Restaurants are a place to relax. During weekends, it gets very crowded. Moreover, the manual system results in long hours of waiting and inconvenience. Workers have load burden and have short time to manage orders, we feel there is a need for an automatic system which makes it convenient for customers to dine inn and relax, pre-order and takeaway.

## **1.1 PURPOSE:**

## The purpose of this SRS is to outline both the functional and non-functional requirements of the subject RMOS. In addition to said requirements, the document also provides a detailed profile of the external interfaces, performance considerations and design constraints Document Conventions. following qualities; correctness, unambiguousness, completeness, consistency, verifiability, modifiability, and traceability that presented set of requirements.

## **1.2 INTENDED AUDIENCE AND READING SUGGESTIONS**

This document is untended for different types of readers such as restaurant owner, system designer, system developer and tester. By reading this document a reader can learn about what the project is implemented for what and how it will present its basic ideas.

## **1.3 PRODUCT SCOPE**

In current formal dining environments, some form of physical static menu is utilized to convey the available food and beverage choices to customers. Said menus are generally paper based and hence impose restrictions on the textual real estate available and the ability a restaurateur must update them. This document specifies the requirements for a restaurant paper menu and ordering replacement strategy to alleviate the problems associated with the current archaic method. Three related concepts are encompassed by the general scope of the Restaurant Ordering System.

## 1.4 References

**YOUTUBE**:

https://www.youtube.com/watch?v=4fYy4l3sSKU

**GOOGLE**:

<https://www.google.com/search?q=srs+for+restaurant+management+system>

# OVERALL DESCRIPTION

## 2.**1 PRODUCT PERSPECTIVE:**

The software described in this SRS is the software for a complete RMOS system. The system merges various hardware and software elements and further interfaces with external systems. Thus, while the software covers the majority of the system's functionality, it relies on several external interfaces for persistence and unhandled tasks, as well as physically interfacing with humans.

## **PRODUCT FUNCTIONS**

* Allow Customers to scroll through the menu and select the dishes he/she wants.
* Allow the Customers to cancel/edit the order any time before its prepared.
* Allow Customers to provide feedback regarding the food and overall service of the restaurant.
* Allow Customers to request for bill.
* Allow Customers to ask for help through the system.
* Assign Head Chef to assign the dishes in an order to chefs according to their specialties.
* Show dish queues and their status, for Chefs.
* Allow admin to perform CRUD (create, retrieve, update, and delete) operations on Staff Members,
* Menu Items, and Inventory.
* Allow Head Chef to mark orders complete.
* Allow the Head Chef to approve cancellation of dish or order.

## **2.3 USER CLASSES AND CHARACTERISTICS**

The Restaurant Management System have five main active actors and one cooperating system. Customer can access the System through Wi-Fi connection and order food,

The chef can see the order and after preparing the food he will tell the system that the food is ready. The waiter can get the confirmation of food from die chef through the system and deliver it to the right table. The cashier can access the system and

receive the payment from customers. The admin can edit the price, count total earning and expenditure.

## **2.4** **OPERATING ENVIRONMENT**

It is an android application running on a tablet and the tablets are present in a restaurant. Firstly, manager would be present at the entrance and system in his tab would show the tables that are empty/reserved. There would be a tab present at every table for customers which they will use to give order. When an order is placed the server would notify the head chef/ kitchen manager who would be in the kitchen. Head chef would use his tab which also would have the system installed and would add the order to the appropriate queues of the chefs. The chefs would be present in the kitchen area and their interface would allow them to check for the dishes they have to prepare. So, the system is running on various tablets but the operating environment and purpose of each is different for each user.

## **2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS**

## System should be compatible and will smoothly run-on Android version PIE 9.0 or above

* FOODIES RESTAURANT system and its user interfaces should be compatible with tablets. However, running on small android mobile devices is not necessary.

## **2.6 USER DOCUMENTATION**

* The software is accompanied by the following materials for further help:
  + User Manual Version 1.0
  + Online support at [www.reactech.com](http://www.reactech.com)

## **2.7 ASSUMPTIONS AND DEPENDENCIES**

## One assumption about the software is that it will always be used on tablets and mobile having android version above or equal to PIE 9.0 that have enough resources to run the application. If the tablet does not have enough hardware resources available for the application, there may be scenarios where the application does not work as intended or not even at all. The application uses Firebase database for online storage of information like orders and menu items that needs to be in working state. If the Firebase interface changes the application needs to be adjusted accordingly

# 3. EXTERNAL INTERFACE REQUIREMENTS

## **3.1USER INTERFACES**

**3.1.1 CUSTOMER INTERFACE**

The customer interface will contain three screens. All three screen will have a consistent layout.

1. Place Order In this screen, system shows a list of cards (UI Elements) of dishes. Each dish will have an image, its price per serving.

2. Timer and Edit/Cancel Order After confirming the order, the user will be shown a timer screen. In this screen customer will be shown “Edit Order” and “Cancel Order” buttons and a timer which shows the completion time of the order. There will also be a button to request for bill.

3. Feedback In feedback screen, at the top right corner a button for “Request Bill” will be shown. Beneath this button we will display a form which will have different multiple-choice questions and a submit feedback button.

# 3.1.2 HEAD CHEF INTERFACE

In head chef interface, system will show all the current orders in detail i.e., all the dishes of a particular order. In each order, there is a button which will be used to mark that the order is ready to be served. Moreover, when customer wants to remove a dish from his order, system will show head chef a notification to approve the removal of the dish.

# 3.1.3 ADMIN INTERFACE:

Software Requirements Specification for FOODIES RESTUARANT As Admin is authorized to perform CRUD operations on Staff Members, Menu Items, and Inventory Items. He’ll be having three different screens for Staff Members, Menu Items, and Inventory.

## **3.2 HARDWARE INTERFACES:**

Our system can interact with a hardware device directly. We must connect our system to the bill printer for handing the hard copy of the bill to the customer. For billing module, we may have to use a credit card reader for payment, but the interaction and the results generated by that reader are just entered into our system manually by the user. Moreover, the central screen in kitchen which will be displaying the status of order queues.

## **SOFTWARE INTERFACES:**

* For Database services system shall use to SQL latest version released.
* System will run on android version above or equal to PIE 9.0
* System shall use support library Print Helper for connecting to the printer and a driver to connect to the kitchen screen.

## **COMMUNICATIONS INTERFACES**

FOODIES RESTAURANT MANAGEMENT SYSTEM is an android application, and it will communicate with SQL (a relational database management system,).

## **4. SYSTEM FEATURES:**

### **2.2.1 FUNCTIONAL REQUIREMENTS**

A functional requirement is a requirement that, when satisﬁed, will allow the user to perform function. The functional requirements of the kitchen, management and order application can be found in tables 3.3, 3.4 and 3.5 respectively. The priority value is in the range from 1 to 3 where 1 is high priority and 3 is low priority.

Kitchen application functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Organized display of active orders. | 1 |
| View preferences and optional choices of every meal. | 1 |
| Inform waiter; update order to ‘in progress’. | 2 |
| Inform waiter; update order to ‘ready’. | 2 |
| Display elapsed time and progress2 of each order. | 3 |

Management application functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Ability to display the real time stock levels. | 1 |
| Ability to add ingredients, meals, and menus. | 1 |
| Ability to edit ingredients, meals, and menus. | 1 |
| Ability to remove ingredients, meals, and menus. | 1 |
| Ability to display statistics of complete orders. | 3 |
| Ability to add new waiters to the system. | 3 |
| Ability to add, edit and remove oﬀers. | 1 |
| Ability to add time intervals3 to menus and oﬀers. | 1 |
| Ability to alter the grid size of the meals, menu sections and menus. | 1 |
| Ability to alter the seating layout. | 3 |
| Ability to alter the price of ingredients. | 1 |
| (Note: Ingredient might still be in stock at a diﬀerent price) |  |
| Ability to change transparency of an image that represents a meal. | 3 |
| Means of stock checking. | 1 |
| Means of updating the minimum stock level. | 1 |

Order application functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Ability to view the active suborder details. | 2 |
| Ability to add a new order to a table. | 1 |
| Ability to add a new order without deﬁning a table. | 1 |
| Ability to add a suborder to an existing order. | 1 |
| Ability to delete a suborder that has not yet been conﬁrmed within the order system. | 1 |
| Ability to view optional ingredients in a meal. | 1 |
| Ability to view cooking preferences of ingredients that can be cooked diﬀerent ways. | 1 |
| Ability to only view the active4 menus. | 1 |
| Ability to view the status of all active suborders. | 2 |
| Ability to print customer receipts on order completion. | 3 |
| Ability to view transaction list of current order. | 3 |
| Ability to alert the waiter when the drinks are complete. | 3 |
| Ability to delete an item or clear the transaction list. | 2 |

### **2.2.2 NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements are usually some forms of constraint or restriction that must be considered when designing the solution. The non-functional requirements of the kitchen, management and order application can be found in tables 3.6, 3.7 and 3.8 respectively. The priority value is in the range from 1 to 3 where 1 is high priority and 3 is low priority.

Kitchen application non-functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Ability to interact with the MySQL database. | 1 |
| Means of refreshing the orders within the ordered list. | 1 |
| Means of refreshing the status of orders. | 3 |
| (How many items in the suborder have been cooked) |  |
| Means of refreshing the status of orders. | 3 |
| (How many items in the suborder have not been cooked) |  |
| Means of refreshing the menu section list. | 2 |
| (How many items in the menu section have been cooked) |  |
| Means of refreshing the menu section list. | 2 |
| (How many items in the menu section have not been cooked) |  |
| Means of refreshing the meal colors within the ordered list: | 3 |
| ==> White: not started |  |
| ==> Yellow: in progress |  |
| ==> Green: complete |  |
| Means of displaying only the food items so drink items are omitted. | 1 |
| Means of clearing the suborder from the kitchen display. | 1 |

Management application non-functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Ability to interact with the MySQL database. | 1 |
| Means of refreshing menus, menu sections and meals live. | 3 |
| Only accessible by management staﬀ. | 3 |
| Means of displaying the links therefore being able to view the ingredients in a meal etc. | 1 |
| Means of searching the database using wild cards for easier data input. | 2 |

Order application non-functional requirements.

|  |  |
| --- | --- |
| Requirements | Priority |
| Ability to interact with the MySQL database. | 1 |
| Ability to update the suborder status every 5 seconds. | 1 |
| Ability to run on a PDA by deﬁning smaller grid and font sizes. | 2 |
| Ability to use the application using a touch screen | 1 |
| without the need of a keyboard or mouse. |  |
| Color coordinated meal buttons to visually warn the user of low stock. | 2 |
| Ability to disable meals that have run out of compulsory stock items. | 1 |
| Ability to only show an ingredient as an optional ingredient choice | 1 |
| if there exists enough ingredient stock. |  |
| Ability to calculate stock ingredient price, order price and discounted price. | 1 |
| Live refreshing of the menus depending on the starting time of the order. | 1 |
| Ability to apply oﬀers depending on the starting time of order | 1 |
| even if the oﬀer is inactive on completion of the order. |  |
| Minimum clicks from the beginning to the end of an order. | 1 |

## **SOFTWARE QUALITY ATTRIBUTES**

**5.4.1 ADAPTABILITY:**

There can be a change in the menu and information stored in the database about employees and inventory.

**5.4.2 AVAILABILITY:**

The system is up and running for most of the time and server is not down for more than a few minutes to avoid inconvenience of the customers.

**5.4.3 CORRECTNESS:**

The bill generated by the application must be accurate and the orders placed should exactly be the same which the user has selected.

**5.4.4 FLEXIBILITY:**

If need arises in the future, software can be modified to change the requirements.

**5.4.5 INTEROPERABILITY:**

The data is transferred from the customer’s end to the kitchen and then head chef assigns orders to each chef. This way data is transferred from one part of the system to another.

**5.4.6 MAINTAINABILITY:**

Software can be easily repaired if a fault occurs.

**5.4.7 PORTABILITY:**

Software can be easily installed on devices and would run smoothly according to the requirement.

**5.4.8 RELIABILITY:**

No matter how many orders are placed, system must give the correct results.

**5.4.9 ROBUSTNESS:**

Software must have checks to ensure that the items that are not available in the menu cannot be selected and the emails, phone numbers added are all valid.

**5.4.10 USABILITY:**

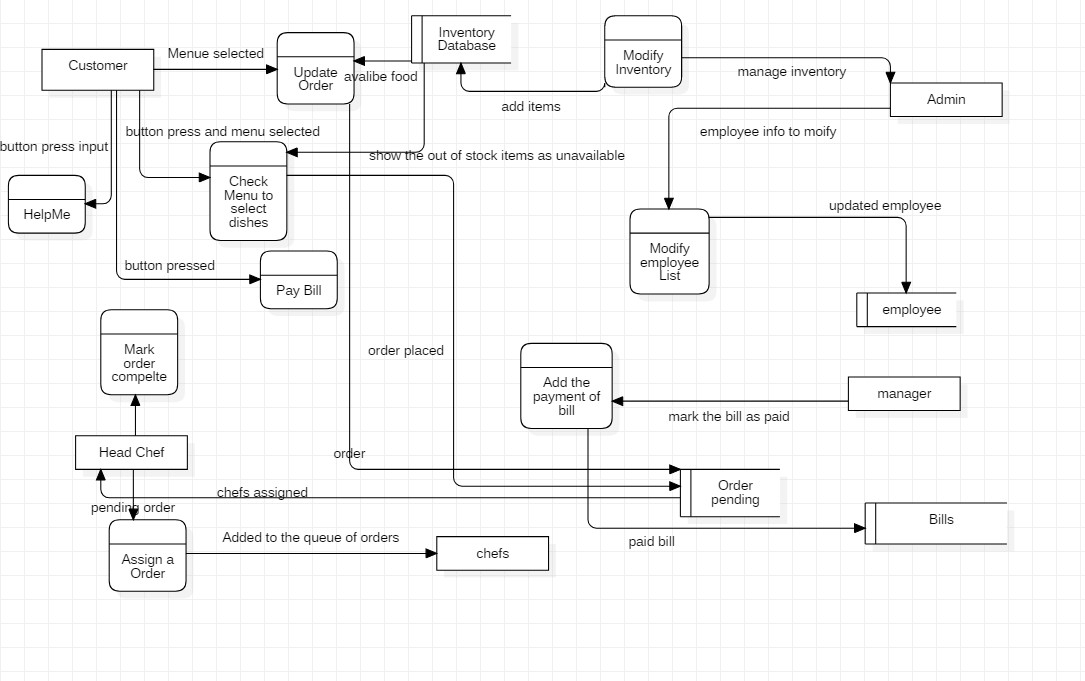
Interface of the software must be easy to use. It would not be complex since managers, chefs have a view, so interface should be simple.

## **BUSINESS RULES**

1. Manager’s interface contains the view of tables that are free, and manager can just view and doesn’t provide any input to the system.
2. Once the bill is paid, manager can mark the order as paid.
3. Admin has access to perform add, delete, update operations on the database for menu, inventory, employees, and no other person can modify the data in the db.
4. Customers can place order from the list of available items and can update order and pay bill.
5. Head chef assigns orders to chefs and can update the queues and has an additional functionality of load balance.
6. Chefs can only view the orders and cannot remove an order from their queue. Only head chef can interact with the queues containing orders.

**APPENDIX B: ANALYSIS MODELS**

## **DATA FLOW DIAGRAM**



**USE CASE DIAGRAM:**



Restaurant Management System

Book Table

Check/Deduct Stock

Waiter

Chef

<<include>>

Confirm Order

Order Food/Drink

Prepare Drink

Cook Food

<<include>> <<include>>

Staff

Bar Staff

Call Server

Customer

Serve Food/Drink

Eat Food

Pay for Meal

Pay

Accept Payment

<<include>>

Manager

Check Offers

Cashier

Generate Reports

Manage menu & ingredients

Order Stock

<<include>>

Update Stock Levels

<<include>>

Stock Check

Login

Collect Order

Place Order

Receive Order

**ACTIVITY DIAGRAM:**

Diagram

Description automatically generated

# Appendix C: To Be Determined List

Weekly sales report and tracking most ordered dish and prioritizing its inventory stocking feature (restock the items that are most ordered often) is yet to be determined by the client and may need further meetings for elaboration.