****

**Department of Computer Science and Engineering**

**Assignment on Software Requirements Specification (SRS)**

**Assignment No. : 04**

**Assignment on : Software Requirements Specification of Online Restaurant management (SRS)**

**Course Code : CSE236**

**Course Tittle : Software Project II**

**Semester : Fall-2023**

Submitted To submitted by

Md. Golam Rabbani Rasedul Kader

Lecturer, Dept. of CSE ID: 221-15-4860

Raihan

Manager

Sajib Dine

Dhaka 1204

Subject: Application for agreement of Restaurant Management System Project.

Dear Mr Raihan

I hope you are well. I hope this letter finds you in good health and high spirits. I am writing to propose the development of a comprehensive Restaurant Management System that I believe will significantly enhance the quality and service of your restaurant.

I am the member of creative IT company. I have identified the hole system of your restaurant and decided to create a unique Restaurant management system.

Thank you for considering this proposal. I am excited about the potential this project

holds for our organization and look forward to the opportunity to discuss it in more detail.

Sincerely

Rasedul Kader

Head

Application development Brach

Rased IT

**1.Introduction**

Contents

[1.2 Document Conventions 5](#_Toc152160887)

[**1.3**  **Intended Audience and Reading Suggestions** 6](#_Toc152160888)

[**1.4**  **Product Scope** 6](#_Toc152160889)

[2. Overall Description 7](#_Toc152160890)

[2.1 Product Perspective 7](#_Toc152160891)

[**2.2**  **Product Functions** 8](#_Toc152160892)

[2.3 User Classes and Characteristics 9](#_Toc152160893)

[2.4 Operating Environment 10](#_Toc152160894)

[2.5 Design and Implementation Constraints 11](#_Toc152160895)

[2.6 User Documentation 11](#_Toc152160896)

[2.7 Assumptions and Dependencies 11](#_Toc152160897)

[**3. External Interface Requirements** 12](#_Toc152160898)

[3.1 User Interfaces 12](#_Toc152160899)

[3.2 Hardware Interfaces 16](#_Toc152160900)

[3.3 Software Interfaces 16](#_Toc152160901)

[3.4 Communications Interfaces 16](#_Toc152160902)

**4.User case design…………………………………………………………………………………………………………………**

**5.Data flow diagram**……………………………………………………………………………………………………………………………….

[**6.**  **System Requirements** 17](#_Toc152160903)

[6.1 Place Order 17](#_Toc152160904)

[**6.1.1 Description and Priority** 17](#_Toc152160905)

[**6.1.2 Stimulus/Response sequences** 17](#_Toc152160906)

[**6.1.3 Functional Requirements** 18](#_Toc152160907)

[6.2 Customer Help 18](#_Toc152160908)

[**6.2.1 Description and Priority** 18](#_Toc152160909)

[**6.2.2 Stimulus/Response sequences** 18](#_Toc152160910)

[**6.2.3 Functional Requirements** 19](#_Toc152160911)

[6.3 Chef Order Queue 19](#_Toc152160912)

[**6.3.1 Description and Priority** 19](#_Toc152160913)

[**6.3.2 Stimulus/Response sequences** 19](#_Toc152160914)

[**6.3.3 Functional Requirements** 20](#_Toc152160915)

[6.4 Edit Order 20](#_Toc152160916)

[**6.4.1 Description and Priority** 20](#_Toc152160917)

[**6.4.2 Stimulus/Response sequences** 20](#_Toc152160918)

[**6.4.3 Functional Requirements** 21](#_Toc152160919)

[6.5 Cancel Order 21](#_Toc152160920)

[**6.5.1 Description and Priority** 21](#_Toc152160921)

[**6.5.2 Stimulus/Response sequences** 21](#_Toc152160922)

[**6.5.3 Functional Requirements** 21](#_Toc152160923)

[6.6 Mark Dish as Cooked 22](#_Toc152160924)

[**6.6.1 Description and Priority** 22](#_Toc152160925)

[**6.6.2 Stimulus/Response sequences** 22](#_Toc152160926)

[**6.6.3 Functional Requirements** 22](#_Toc152160927)

[6.7 Request Bill 23](#_Toc152160928)

[**6.7.1 Description and Priority** 23](#_Toc152160929)

[**6.7.2 Stimulus/Response sequences** 23](#_Toc152160930)

[ **6.7.3 Functional Requirements** 23](#_Toc152160931)

[6.8 Customer Feedback 23](#_Toc152160932)

[**6.8.1 Description and Priority** 23](#_Toc152160933)

[**6.8.2 Stimulus/Response sequences** 24](#_Toc152160934)

[**6.8.3 Functional Requirements** 24](#_Toc152160935)

[6.9 Add/Edit/Delete Staff Members 24](#_Toc152160936)

[**6.9.1 Description and Priority** 24](#_Toc152160937)

[**6.9.2 Stimulus/Response sequences** 24](#_Toc152160938)

[**6.9.3 Functional Requirements** 25](#_Toc152160939)

[6.10 Add/Edit/Delete Menu Items 25](#_Toc152160940)

[**6.10.1 Description and Priority** 25](#_Toc152160941)

[**6.10.2 Stimulus/response sequences** 25](#_Toc152160942)

[**6.10.3 Functional Requirements** 26](#_Toc152160943)

[**7.**  **Nonfunctional Requirements** 27](#_Toc152160944)

[7.1 Performance Requirements 27](#_Toc152160945)

[7.2 Safety Requirements 27](#_Toc152160946)

[7.3 Security Requirements 27](#_Toc152160947)

[7.4 Software Quality Attributes 27](#_Toc152160948)

**8.Testing…**……………………………………………………………………………………………………………………………………………34

[**9. Milestone & Reporting** 35](#_Toc152160949)

[**10. Payment Terms & condition** 35](#_Toc152160950)

[**11.Responsibility** 36](#_Toc152160951)

[**12.Contact Us** 36](#_Toc152160952)

[**13. Agreement Signed By:** 36](#_Toc152160953)

[**14. Which methodology should be preferred by me to develop my application and why?** 42](#_Toc152160954)

**1.1 Purpose**

Dine Out is an android application that aims to digitalize the process of various restaurant management operations including ordering and inventory management and POS. This document aims to capture the system requirements and features particularly related to ordering and inventory management to be implemented in DineOut version 1.0., with the later releases on POS (Point of Sale).

## 1.2 Document Conventions

**1.2.1 Priority Conventions**

In this complete document, we will mention priority as “low” or “high” throughout the document. Secondly, priorities are only mentioned in section 4 and section 5 along with detailed description of the requirements. Any high-level requirements mentioned elsewhere are assumed to inherit priorities of their detailed counterparts in section 4,5.

**1.2.2 Fonts Conventions**

Throughout this document, All the user entities are written in capitalizations i.e. first letter as capital. Also, any significant term which has been described in the glossary is made bold and italic in the text. On the other hand, those terms which are significant (but not described in glossary) are bold in text.

**1.3 Intended Audience and Reading Suggestions**

The purpose of this document is to give a detailed description of the requirements for the “DineOut” software. It will illustrate the purpose, scope and complete description for the development of system. It will also explain external interface requirements and system requirements as well as non-functional requirements. This document is primarily intended to be proposed to a customer for its approval and also for further processing such as additions to be developed in later releases.

Customers can refer to section 3 and 4 for the list of requirements implemented in Version 1.0. Users are advised to refer to user documentation section for tutorials and online support information.

This document will also be used as a reference for developing and testing Version 1.0 by the development team as well as the testers. The development team can refer to section 2.3 and 2.6 for system level information and section 3 for system features that are to be implemented in this version of the software.

**1.4 Product Scope**

DineOut is a restaurant management system developed with the intention of automating the day to day tasks in a restaurant like order and inventory management, bill generation and taking feedback. This release of the software would deal with these tasks only whereas more areas might be automated in the future versions of this software. The main purpose is to improve the performance of the restaurant by eradicating the daily paperwork. With this system the tasks would be performed in less amount of time and more efficiently. An additional benefit of this software is that during the rush hours the load can be balanced effectively, and restaurants would perform better than usual. In addition to this, human error that occurs when performing tasks manually is also minimized and presence of queues in the system to assign tasks to chefs can reduce congestion in the kitchen. The system would also result in reduction of labor which would result in the reduction of expenses of the restaurant. Feedback module would help the restaurant check for how well they are performing, and monthly/yearly figures can be checked by the billing module to see the trends in sales and profits. These benefits can potentially result in generation of more revenues for the restaurant.

## 2. Overall Description

This section will give an overview of the DineOut application. The basic functionality of the system as well its context will be explored in detail. It also describes different kinds of stakeholders and user classes associated with the system and what functionality is available for each class. At last, the assumptions and dependencies for the system are presented.

## 2.1 Product Perspective

DineOut app will attempt to replace the traditional manual ordering process and is a new self contained software system that consists of two parts: one mobile application and the other is Firebase database. The mobile application will be used for ordering and interacting with the inventory while the Firebase database will be used for storing the inventory and ordering related information about the food items like pending and complete order queues.

The mobile application will have five interfaces. Each for Customer, Manager, Head Chef, Admin and Chef. Manager can see/edit the status of available/reserved tables. Customer’s interface will consist of a scrollable menu listing available items and their price. When the customer selects some dishes and place the order, it will be stored in “pending orders” table in Firebase database. Head Chef’s interface will be such that he is notified of the pending order and he is able to assign it to one the available queues of chefs who are then able to see the new order in their screens or on a central display in kitchen. After each item/dish in an order is prepared, the order is marked completed through the Head Chef’s interface, the hall manager gets notified through his interface. Customer’s interface has an option for requesting the bill. Bill is printed through the Manager’s interface. Admin can change and modify the Firebase database like add new menus or staff, edit current inventory stock etc.

**2.2 Product Functions**

Given below are the major functions that can be performed using DineOut app. Moreover, a Data Flow Diagram (DFD) for better understanding of the system is also given in Appendix B.

The system will:

* 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.
* Allow Hall Manager to mark the bill as paid.
* Notify the Hall Manager when a particular order is complete.
* Allow the Hall Manager to see/edit status of tables reserved and available and their capacities.

## 2.3 User Classes and Characteristics

There are four types of users that interact with our system (See appendix B). Firstly, there is a Hall Manager, then Customer, Head Chef and Admin. We’ll provide an interface for Chefs as well through which they are looking at the status of their order queues, but they will not interact with our system.

#### **2.3.1 Customer Class**

Customers interact with our system directly in order to place order, modify order, get bill and give feedback. We do not store any information related to customers in our system. The process of order taking starts from customers placing order and then the other series of events begin.

#### 2.3.2 Head Chef/Kitchen Manager

Head Chef can mark a dish as prepared when a chef tells him to do so. He can approve the cancellation of an order whenever a customer edits or removes a dish from his order. He can also assign a dish to a particular chef based on the specialty of the chef.

#### 2.3.3 Chef

Chefs don’t interact with the system. They just have to look at the dishes present in their queues and prepare the dishes accordingly. Chef’s name, address and specialty etc. are stored in the database.

#### 2.3.4 Admin

Admin’s job is to manage the inventory and other information related to menu and chefs in the system.

**2.3.5 Hall Manager**

Hall Managers will provide its input when he marks the bill as paid when customers pay for their order or get the bill printed. Moreover, he gets a notification whenever a particular order is complete, or some customer asks for help through the system. Hall manager can also see tables in the hall and their status i.e. empty or filled.

## 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

**2.5.1 Operating System Constraint**

System should be compatible and will smoothly run on Android version 6.0 or above.

**2.5.2 Device Constraint**

DineOut’s core 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.dinout.com

## 2.7 Assumptions and Dependencies

One assumption about the software is that it will always be used on tablets 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**

## 3.1 User Interfaces

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.

* 1. **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.

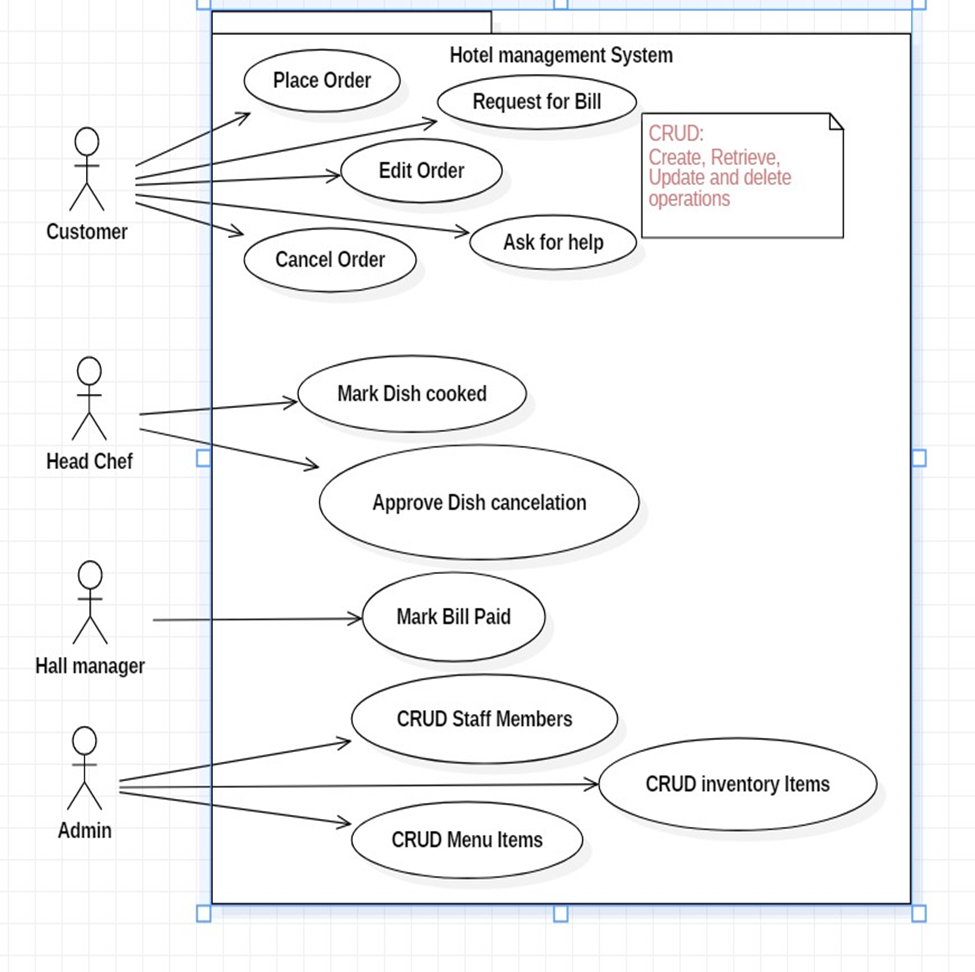
* 1. **Feedback**

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.

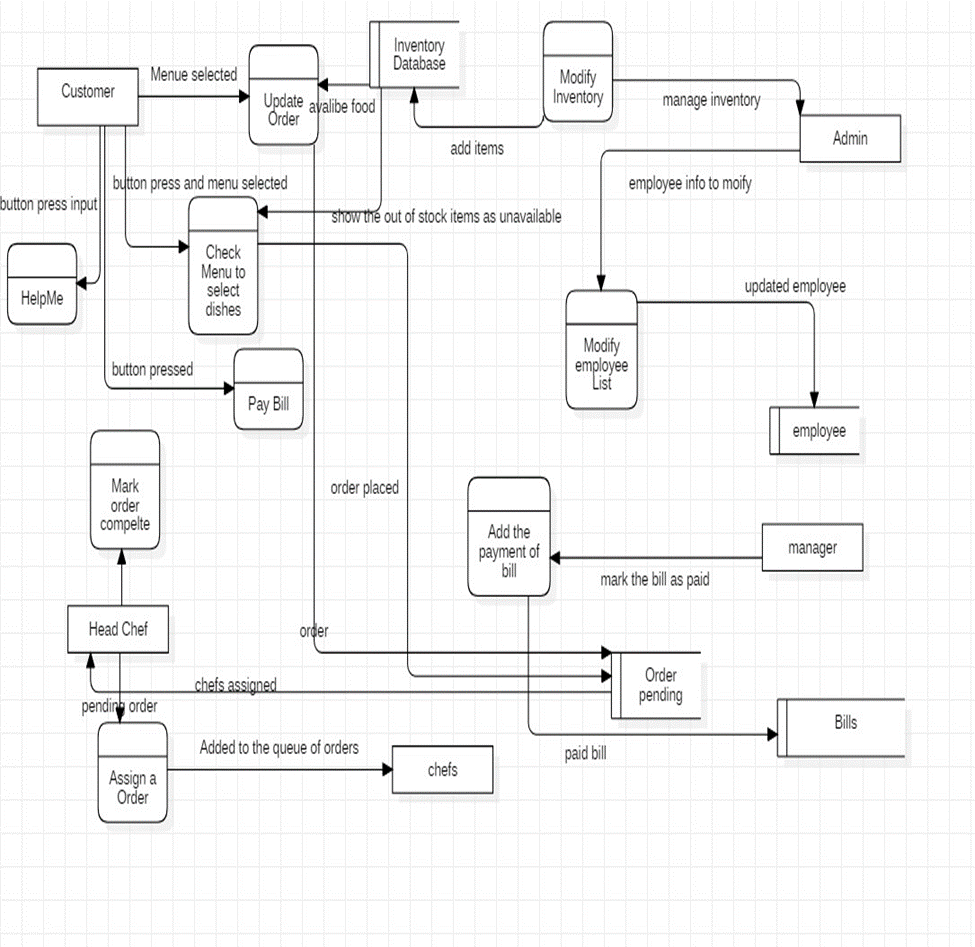
1. **Head Chef Interface**

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 dish cooked. 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.User case Design**



**4.Data flow diagram**



**5. Hall Manager Interface**

Hall manager will have a screen where he will get notification whenever an order is completed. System will notify the hall manager about the order number and table number. Hall Manager also has a screen where all orders are listed, and status button to mark the order as paid. Moreover, he also has an interface screen to see and the status of tables in the restaurant as free/available.

**5.1 Admin Interface**

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.

## 5.2 Hardware Interfaces

Our system can interact with a hardware device directly. We have to 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.

## 5.3 Software Interfaces

* For Database services system shall use to Firebase latest version released on October 16, 2018.
* System will run on android version above or equal to ***marshmallow 6.0***
* System shall use v4 support library ***Print Helper*** for connecting to the printer and a driver to connect to the kitchen screen.

## 5.4 Communications Interfaces

DineOut is an android application and it will communicate with Firebase (which is a storage server provided by Google for android developers). Firebase uses HTTP protocol for communication, so our device will follow HTTP protocol when connecting to Firebase.

# **6. System Requirements**

## 6.1 Place Order

### **6.1.1 Description and Priority**

The system will give customers the ability to place their orders using our product. It will display a list of available and unavailable dishes in the menu where unavailable dishes will be grayed out. Customer will be able to select multiple dishes and their quantity for a particular order.

**Priority:**

high

### **6.1.2 Stimulus/Response sequences**

When user enters the order activity/page, initially system displays a list of available and unavailable dishes along with their prices.

1. **Stimulus:**

Customer taps on an available dish.

**Response:**

System shows a popup having name of the dish and price per serving. Also, it contains a text box for the customer to enter the quantity, OK button and a Cancel button.

* 1. **Stimulus:**

Customer taps on an unavailable dish.

**Response:**

Nothing happens.

1. **Stimulus:**

Customer enters the quantity and press OK button.

**Response:**

System closes the popup, shows a small green tick mark at the side of dish.

Below the tick mark it shows quantity selected and total price of that dish.

* 1. **Stimulus:**

If Customer taps on cancel button after 1.

**Response:**

the system closes the popup, and nothing happens.

1. **Stimulus:**

Customer taps on confirm order button at the bottom **Response:**

System closes the order screen and displays a timer along with a “Cancel Order” button and “Edit Order” button

### **6.1.3 Functional Requirements**

**REQ-1:** The system will show a list of cards (UI element) of dishes. Each card will have a picture of the dish. Below the dish it shows the price in Rupees per serving.

**REQ-2:** The system must show all available and unavailable dishes to the Customer.

**REQ-3:** Tap on any of the displayed dish will result in a popup for quantity and a green mark after quantity has been selected.

**REQ-4:** The popup for quantity input will not allow the user to enter letters, negative numbers or any invalid characters.

**REQ-5:** After completing the order the system will display a timer “Time to complete the order” and it is the total time required to serve the dish keeping in view the previously queued orders. Moreover, it also shows a cancel order button.

**REQ-6:** Unavailable dishes must be displayed but their operations must be disabled.

## 6.2 Customer Help

### **6.2.1 Description and Priority**

Our system will provide help for the customer if the customer faces issues in using the tab, there will be a ‘help’ option in his interface. If he faces issues in using the tab or want some other assistance, he can notify the hall manager through the system.

**Priority:**

high

### **6.2.2 Stimulus/Response sequences**

The home screen for the customer shows a help button on top right corner of the screen.

1. **Stimulus:**

Customer taps on the help button. **Response:**

System shows a popup with two buttons, “Call a waiter to manage order”,

“Call a waiter for help”

1. **Stimulus:**

Customer taps on any of the button.

**Response:**

System closes the popup and sends a notification to the Hall Manager. Notification will include the table number of the Customer.

### **6.2.3 Functional Requirements**

**REQ-2:** The system must give Customer the ability to ask for help.

**REQ-1:** When the customer taps on “Call the waiter to manage order”. the system must store that the above order was given by the waiter.

## 6.3 Chef Order Queue

### **6.3.1 Description and Priority**

Whenever a new order is placed by the Customer, the dishes in the orders are classified into categories. The system has the information of specialty of each chef, it will assign each dish to a corresponding chef and place it in the order queue of that chef. There is a centralized screen in the kitchen which displays queues for each chef. Each item in the queue is labeled with the name of the dish.

**priority:** high

### **6.3.2 Stimulus/Response sequences**

**1. Stimulus:**

Customer taps the “Confirm Order” button in “Place Order screen”. **Response:**

Displays the dishes on kitchen screen in corresponding chef’s queue.

### **6.3.3 Functional Requirements**

**REQ-1:** System will classify the dishes in the order according to category and add this dish on a particular chef’s queue in the kitchen screen.

## 6.4 Edit Order

### **6.4.1 Description and Priority**

Customer can edit the order any time before the serving. In editing mode, the customer can change the quantity of the of the food ordered, add and remove dishes from the order.

**priority:** high

### **6.4.2 Stimulus/Response sequences**

The timer screen shows two buttons “Cancel Order” and “Edit Order” button

1. **Stimulus:**

Customer taps on “Edit Order” button. **Response:**

System shows the previous menu screen where selected dishes are already marked with green tick.

1. **Stimulus:**

Customer taps on any of the selected dish.

**Response:**

System opens a popup with previous quantity pre-filled. This popup will also contain a button “Remove Dish”.

1. **Stimulus:**

Customer enters new quantity and press “OK” **Response:**

System shows an error “Cannot edit <Name> dish” or System closes the popup and new quantity will be displayed on that dish in the list.

1. **Stimulus:**

Customer taps on “Remove Dish” **Response:** system responds with “Dish <Name> removed” or “Dish <Name> cannot be removed”

1. **Stimulus:**

Customer taps on any new dish which was not previously selected  **Response:**

stimuli/Responses of “Place Order” feature will be followed.

### **6.4.3 Functional Requirements**

**REQ-1:**

System must allow the Customer to increase, decrease or even remove the dish from the order any time before serving.

**REQ-2:**

System must remove the dish or decrease quantity of the dish with the approval of head chef.

## 6.5 Cancel Order

### **6.5.1 Description and Priority**

Our system will also provide an option to cancel the current order. When the customer taps on the “Cancel Order” button. Customer can cancel the order at any time before serving.

**priority:** high

### **6.5.2 Stimulus/Response sequences**

**1. Stimulus:**

Customer taps on the “Cancel Order” button **Response:** system responds with a popup “Order canceled successfully” or “Order cannot be cancelled”

### **6.5.3 Functional Requirements**

**REQ-1:**

System must allow the customer to cancel order at any time before serving.

**REQ-2:**

In cancel order, all the dishes will be presented for approval to the head chef. Only approved dishes will be dropped.

## 6.6 Mark Dish as Cooked

### **6.6.1 Description and Priority**

The head chef can mark the dish of a particular order complete when notified by the chef.

**priority:** high

### **6.6.2 Stimulus/Response sequences**

The system will show a list of current orders in earliest first order in head chef screen. Each order also shows order no and table no associated with the order. Moreover, it also shows a list of dishes for each order. Alongside of each dish there is a button saying, “Marked Cooked”.

**Stimulus:**

Head chef taps on the “Mark Cooked” button on a dish in an order. **Response:**

System changes that button to a green tick.

**2. Stimulus:**

All the dishes of a particular order have been marked “cooked” **Response:**

System shows a notification to the hall manager saying, “Order of Table No

<Table No> is ready for serving”. System shows a new screen having Title “Food Ready” to the Customer showing a button “Request Bill” and MCQ’s for customer feedback.

### **6.6.3 Functional Requirements**

**REQ-1:**

System must send a notification to the hall manager once all the dishes of a particular order has been marked “cooked”.

**REQ-2:**

System must replace the timer screen with a new screen having feedback and request bill options.

## 6.7 Request Bill

### **6.7.1 Description and Priority**

Request bill option gives the ability to the customer to ask for receipt and pay the bill.

**priority:** high

### **6.7.2 Stimulus/Response sequences**

**1. Stimulus:**

Customer taps on the request bill button **Response:**

The system prints the bill through a printer. System will add a bill to the hall manager’s view with the button that says “paid”.

### **6.7.3 Functional Requirements**

**REQ-1:** The system must notify the hall manager that a customer has request for a bill

**REQ-2:** The system must show Hall manager the order no, table no and total payable amount

**REQ-3:** The system must give ability to the hall manager to change the status of the bill to paid.

## 6.8 Customer Feedback

### **6.8.1 Description and Priority**

The system will give customers the ability to give a feedback for the food or overall services. In the feedback screen there are multiple choice questions each having two options “Satisfactory” and “Unsatisfactory”. At the end there is a submit button.

**priority:**

high

### **6.8.2 Stimulus/Response sequences**

**1. Stimulus:**

The customer taps on request bill option **Response:**

the system shows a feedback screen with multiple choice questions and a submit button.

### **6.8.3 Functional Requirements**

**REQ-1:** System must show the feedback screen to the user.

**REQ-2:** System must display multiple choice questions for feedback.

## 6.9 Add/Edit/Delete Staff Members

### **6.9.1 Description and Priority**

The system gives ability to the admin to add, edit and delete staff members. Using this feature an admin can add chefs, waiters, managers.

**priority:** high

### **6.9.2 Stimulus/Response sequences**

Admin/Manage screen shows a grid of staff members. There is a button at the top of grid which says Add Member. In the grid after every entry there is a “Edit” and

“Remove” button.

1. **Stimulus:**

Admin taps on “Add Staff” button **Response:**

System opens another screen with a form

1. **Stimulus:**

Admin fills the information and hit submit **Response:**

System responds with “<Staff Member> added successfully”

1. **Stimulus:**

Admin taps on edit button **Response:**

System opens a screen with a form prefilled with the existing values.

1. **Stimulus:**

Admin edits the information and hit submit **Response:**

System responds with “<Staff Member> edited successfully”

1. **Stimulus:**

Admin taps on remove button on a particular row **Response:**

responds with a “<Staff Name> removed successfully”

### **6.9.3 Functional Requirements**

**REQ-1:** Admin should be able to add all necessary information about the staff member

**REQ-2:** System must give admin the ability to edit information about all staff members

**REQ-3:** System must give admin the ability to remove staff members.

## 6.10 Add/Edit/Delete Menu Items

### **6.10.1 Description and Priority**

The system gives ability to the admin to add, edit and delete staff members. Using this feature an admin can add chefs, waiters, managers.

**priority:** high

### **6.10.2 Stimulus/response sequences**

Admin screen shows all the previously added dishes. It also shows a “Add Dish” button along with “Edit” and “Remove” with all the available dishes

1. **Stimulus:**

Admin taps on “Add Dish” button **Response:**

System opens another screen with a form

1. **Stimulus:**

Admin fills the information and hit submit **Response:**

System responds with “<Dish> added successfully”

1. **Stimulus:**

Admin taps on edit button **Response:**

System opens a screen with a form prefilled with the existing values.

1. **Stimulus:**

Admin edits the information and hit submit **Response:**

System responds with “<Dish Member> edited successfully”

1. **Stimulus:**

Admin taps on remove button on a particular row **Response:**

responds with a “<Dish> removed successfully”

### **6.10.3 Functional Requirements**

**REQ-1:** Admin should be able to add all necessary information about the staff member

**REQ-2:** System must give admin the ability to edit information about all staff members

**REQ-3:** System must give admin the ability to remove staff members.

# **7. Nonfunctional Requirements**

## 7.1 Performance Requirements

The system must be interactive, and the delays involved must be less. So, in every action response of the system, there are no immediate delays. In case of scrolling through the menu there should be a delay of no more than 2 second before the next page of menu items is displayed otherwise our people’s dining experience is affected. The order should be placed in pending orders and be visible to the head chef/chefs in less than 1 second to start the preparation.

## 7.2 Safety Requirements

The software is completely environmentally friendly and does not cause any safety violations. The menu will have a flexible font that can be zoomed so as to not over constrain the eyes.

## 7.3 Security Requirements

There is a need for a proper and encrypted login authentication for head chef and admin as employee sensitive information as well as inventory should be protected from hacking. Information transmission should be securely transmitted to Firebase without any changes in information to avoid disturbances in orders and billing

## 7.4 Software Quality Attributes

**7.4.1 Adaptability:**

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

**7.4.2 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.

**7.4.3 Correctness:**

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

**7.4.4 Flexibility:**

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

**7.4.5 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.

**7.4.6 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.

**7.4.8 Reliability:**

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

**7.4.9 Reusability:**

Current version can be used in the future versions with more functionality added.

**7.4.10 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.

**7.4.11 Testability:**

All the requirements are fulfilled, response time is low, and all functions are working perfectly.

**7.4.12 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.

**Testing:**

Following features will be used for testing

* This Application will be tested with Agile model
* Application will be tested by waterfall model
* Application will also be tested by scrum .

**8.1Testing Approach:**

The best software testing approach for an online shopping system project is a combination of black-box testing and white-box testing. Black-box testing focuses on the external behavior of the system without considering its internal structure, while white-box testing focuses on the internal structure of the system to ensure that it is implemented correctly.

**Black-box testing** techniques such as equivalence partitioning, boundary value analysis, and use case testing are well-suited for testing online shopping systems because they can uncover a wide range of bugs and defects. These techniques are also relatively easy to apply, even for testers who are not familiar with the internal workings of the system.

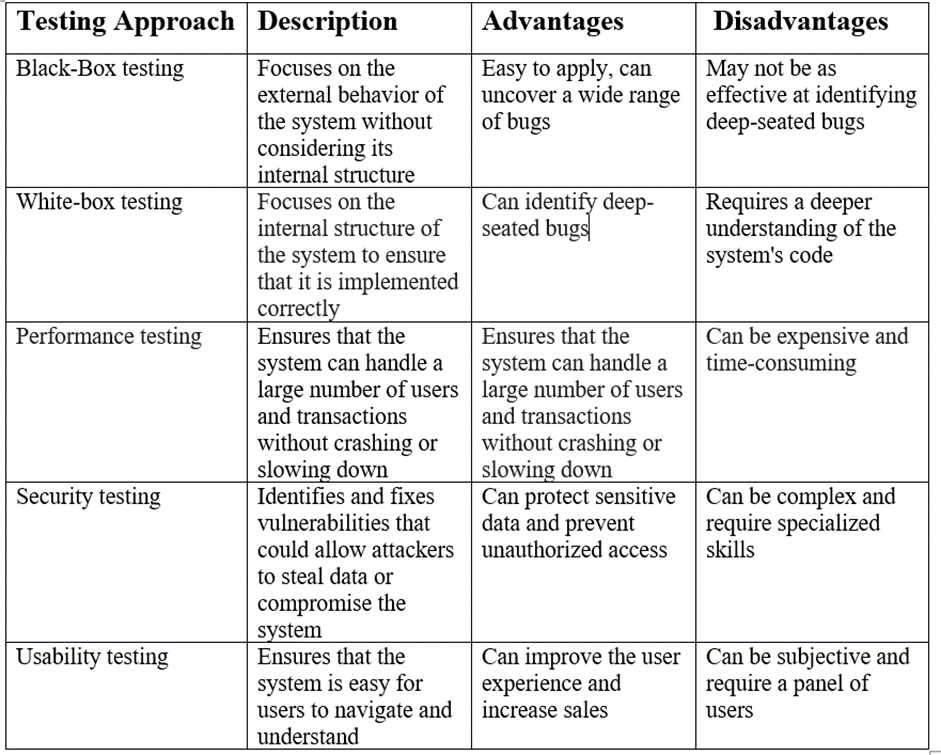
**White-box testing** techniques such as unit testing, integration testing, and code coverage can also be used to test online shopping systems. These techniques are particularly important for ensuring that the system's code is well-designed, bug-free, and secure.

In addition to black-box and white-box testing, other important testing approaches for online shopping systems include:

* **Performance testing:** This type of testing ensures that the system can handle a large number of users and transactions without crashing or slowing down.
* **Security testing:** This type of testing identifies and fixes vulnerabilities that could allow attackers to steal data or compromise the system.
* **Usability testing:** This type of testing ensures that the system is easy for users to navigate and understand.

The specific mix of testing approaches that is used will depend on the specific requirements of the project. However, all of the testing approaches listed above are important for ensuring that online Online management systems are reliable, secure, and user-friendly.

Here is a table comparing black-box testing, white-box testing, and other important testing approaches for online shopping systems:



* 1. Testing types, techniques and tactics:

**1. Functional Testing:** Functional testing verifies that the system's functionalities operate as intended according to the specified requirements. It encompasses various techniques, including:  
   
 **a. Unit Testing:** Unit testing focuses on individual units of code, ensuring they function correctly in isolation.

**b. Integration Testing:** Integration testing validates the interaction between different modules or components of the system.

**c. System Testing:** System testing assesses the entire system as a whole, ensuring it meets all functional requirements.

**2.Non-functional Testing:** Non-functional testing extends beyond the core functionalities, evaluating aspects like performance, usability, security, and compatibility. It includes techniques such as:

**a. Load Testing:** Load testing evaluates the system's ability to handle increasing user loads and maintain performance under pressure.

**b. Stress Testing:** Stress testing pushes the system beyond its expected capacity to identify performance bottlenecks and breaking points.

**c. Usability Testing:** Usability testing assesses the user interface's ease of use, intuitiveness, and overall user experience.

**d. Penetration Testing**: Penetration testing involves ethical hacking attempts to identify and exploit security vulnerabilities.

Effective system testing for an online restaurant management system involves a combination of techniques to ensure thorough coverage and reliability. Here are some techniques you can use for testing:

**1. End-to-End Testing:**

Validate the entire system flow from user registration to order placement, payment processing, and order fulfillment.

Ensure that all integrated components work seamlessly together.

**2. Security Testing:**

Perform penetration testing to identify vulnerabilities in the system.

Test for secure data transmission, encryption, and protection against common security threats.

**3. Performance Testing:**

Conduct stress testing to assess the system's behavior under extreme conditions.

Analyze response times for critical operations to ensure they meet acceptable standards.

**4. API Testing:**

If the system relies on external APIs, test the integration points and data exchange.

Verify that data is accurately transmitted between the system and external services.

**5. Scalability Testing:**

Assess the system's ability to handle increased loads as the user base grows.

Evaluate scalability in terms of both performance and database capacity.

**6. Automated Testing:**

Implement automated tests for repetitive and critical test scenarios to improve efficiency and consistency.

By combining these testing techniques, you can ensure a robust and reliable online restaurant management system that meets user expectations and performs well in various scenarios.

8.2Testing tools  
 1.Functional Testing Tools:

* **Selenium:**

**Purpose:** Automated web browser testing.

**Features:** Supports various programming languages, browsers, and operating systems.

* **Cypress:**

**Purpose:** End-to-end testing for modern web applications.

**Features:** Fast execution, real-time reloading, and easy setup.

* **Test Complete:**

**Purpose:** Comprehensive automated testing across desktop, web, and mobile applications.

**Features:** Record and playback, script-free testing, and support for various scripting languages.

**2.Performance Testing Tools:**

* **Apache JMeter:**

**Purpose:** Load testing and performance measurement.

**Features:** Supports a variety of applications, protocols, and provides graphical analysis of performance reports.

* **LoadRunner:**

**Purpose:** Load testing for applications and websites.

**Features:** Simulates virtual users, measures system performance, and identifies bottlenecks.

* **Gatling:**

**Purpose:** Open-source load testing framework based on Scala.

**Features:** Highly scalable, supports scripting in Scala, and provides detailed performance metrics.

**3.Security Testing Tools:**

* **OWASP ZAP (Zed Attack Proxy):**

**Purpose:** Automated security testing for web applications.

**Features:** Scanning for common security vulnerabilities, automated scanners, and various tools for manual testing.

* **Burp Suite**:

**Purpose:** Web application security testing toolkit.

**Features:** Performs security testing, including scanning for vulnerabilities, crawling content, and analyzing requests.

* **Net sparker:**

**Purpose:** Web application security scanner.

**Features:** Automatic detection of vulnerabilities, reporting, and prioritization of security issues.

**4.Cross-Browser Testing Tools:**

* **Browser Stack:**

**Purpose:** Cross-browser testing platform.

**Features:** Allows testing on a wide range of browsers and devices, including mobile.

* **Cross Browser Testing:**

**Purpose:** Cloud-based cross-browser testing platform.

**Features:** Test on real devices and browsers, visual testing, and automated testing support.

**5.Load Balancer Testing Tools:**

* **Locust:**

**Purpose:** Open-source load testing tool.

**Features:** Define user behavior using Python code, distributed load testing, and real-time monitoring.

* **Artillery:**

**Purpose:** Modern, powerful, and easy-to-use load testing toolkit.

**Features:** Scenario-based testing, WebSocket support, and detailed reporting.

# **8. Milestone & Reporting**

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Task** | **Required Time** |
| **Analyzing Project theme** | **(Submit Idea & Design)** | **4 Days** |
| **Requirements Study** | **(Gathering Data & submit)** | **4 Days** |
| **Project Working & Development** | **Work with the project from root** | **21 Days** |
| **Testing** | **Testing the entire application system** | **40 Days** |
| **Deployment** | **Fining and Review Final project** | **7 Days** |
| **Delivery Project** | **Available to Online platform** | 1. **Days** |

# **10. Payment Terms & condition**

15% payment will be accepted for the Project proposal and design Submission.

35% Payment will be accepted for the Application Development

70% payment will be accepted after application review and Testing

100% .Payment will be accepted after handover the fully completed Application

# **10.Responsibility**

The entire Application has been done by Asif Hossain and all the responsibility including terms and condition will goes to him.

# 

# **11.Contact Us**

You can get in touch with us in any of the below ways:

Golam Rabbany

By Phone: +8801787774996

By Email

grabbany1234@gmail.com

On our website

[www.Rased](http://www.Rased) IT

### 

# **12.** **Agreement Signed By:**

|  |  |  |
| --- | --- | --- |
| ……………………………    Client Signature  Raihan  Manager  Food Corner | …………………………..    Order Provider Signature  Rased  Officer  Creative IT | ……………………………    Authority Signature  Golam Rabbany  Managing Director (MD)  Creative IT |

13.Software Development Methodology :

A software development methodology is a structured and systematic approach to designing, building, testing, and delivering software applications or systems. These methodologies provide a framework for organizing and managing the software development process. Here are some key points about software development methodologies:

**Purpose:** The primary purpose of a software development methodology is to provide a set of best practices and guidelines to help development teams plan, execute, and complete software projects successfully. It aims to ensure that software is developed efficiently, meets quality standards, and satisfies customer requirements.

**Phases and Processes:** Software development methodologies typically define phases or stages that a project goes through, such as requirements gathering, design, implementation, testing, and deployment. They also outline the processes and activities that occur within each phase.

**Principles and Practices:** Each methodology is based on a set of principles and practices that guide how software development tasks are performed. These principles may emphasize collaboration, adaptability, documentation, or other aspects of the development process.

**Types of Methodologies:** There are various software development methodologies, each suited to different project types and contexts. Some common methodologies include Agile (e.g., Scrum, Kanban), Waterfall, Iterative, and Incremental, among others. Hybrid methodologies, combining elements from multiple approaches, are also common.

**Adaptability:** The choice of a methodology often depends on project requirements, organizational culture, and the nature of the work. Some methodologies are highly adaptive and suitable for projects with evolving or uncertain requirements, while others are more structured and better suited for projects with well-defined requirements.

**Customer-Centric:** Many modern methodologies emphasize customer collaboration and feedback. They prioritize delivering value to customers and end-users, ensuring that the final product aligns with their needs and expectations.

**Continuous Improvement:** Agile methodologies, in particular, stress the importance of continuous improvement. Teams regularly reflect on their processes and practices and make adjustments to become more effective over time.

**Documentation:** The level of documentation required varies between methodologies. Some, like Waterfall, emphasize comprehensive documentation at each phase, while others, like Agile, prioritize working software over extensive paperwork.

**Tools and Practices:** Methodologies often come with associated tools and practices to facilitate project management, collaboration, and development tasks. For example, Scrum uses daily stand-up meetings and a product backlog.

**Team Roles:** Many methodologies define specific roles within a development team, such as product owners, scrum masters, and developers. These roles have specific responsibilities and contribute to the successful execution of the methodology.

14.Types of Software Development Methodologies :

There are several types of software development methodologies, each offering a distinct approach to managing and executing software projects. Here are some of the most commonly used software development methodologies:

**Waterfall Model :**

Waterfall is a linear and sequential methodology where each phase of the project (requirements, design, implementation, testing, deployment) is completed before moving to the next. It is well-suited for projects with well-defined, stable requirements.

**Agile Methodologies :**

**Scrum:** An Agile framework that divides work into time-boxed iterations called sprints. It emphasizes collaboration, adaptability, and customer feedback.

**Kanban:** Focuses on visualizing work on a board and optimizing workflow. It allows for continuous delivery and minimizes work in progress.

**Extreme Programming (XP):** Emphasizes close collaboration among team members, frequent testing, and continuous integration. It places a strong emphasis on code quality and customer involvement.

**Iterative and Incremental Methodologies :**

Rational Unified Process (RUP): An iterative and incremental approach that divides the project into phases and produces a series of intermediate deliverables.

Feature-Driven Development (FDD): Focuses on breaking down a complex software system into smaller, well-defined features or functions.

**DevOps :**

A set of practices that combine software development (Dev) and IT operations (Ops) to streamline and automate the software delivery and deployment process.

**Spiral Model :**

An iterative and risk-driven approach that combines elements of both Waterfall and iterative development. It emphasizes risk analysis and management.

**Big Bang Model :**

A simplistic approach where development begins with no specific planning or requirements, often used in small projects or prototypes.

**Dynamic Systems Development Method (DSDM) :**

An Agile approach that focuses on delivering business value and adapting to changing requirements while maintaining a strict timeline.

**Hybrid Model :**

Combining elements from multiple methodologies to create a customized approach that suits the specific needs of a project or organization.

# **15. Which methodology should be preferred by me to develop my application and why?**

Selecting the right methodology for an online restaurant management system project depends on various factors such as project size, complexity, team size, and the level of flexibility required. Here a methodologies commonly used in software development, and their suitability for an online restaurant management system

**Agile Methodology:**

* Iterative and flexible, allowing for changes throughout the development process.
* Regular customer feedback and involvement.
* Well-suited for projects with evolving requirements.

**Suitability for Online Restaurant Management System:**

* Agile can be a good fit, especially if the project requirements are expected to evolve or if there is a need for continuous feedback and adaptation.

In conclusion, for an online restaurant management system, Agile methodologies are often preferred. They provide flexibility to adapt to changing requirements, allow for continuous customer feedback, and support incremental development.