Software Requirements Specification

for

RESTAURANT MANAGEMENT SYSTEM

Prepared by:

MIDHUN KRISHNA

15 FEBRUARY2021

Table of Contents

1. Introduction

1.1 Purpose

RMS 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 RMS 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 “RMS” 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

RMS 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.

1.5 References

1. Android User Interface Guidelines, Available at: https://developer.android.com/guide/practices/ui_guidelines/

2. IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2. Overall Description

This section will give an overview of the RMS 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

RMS 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.

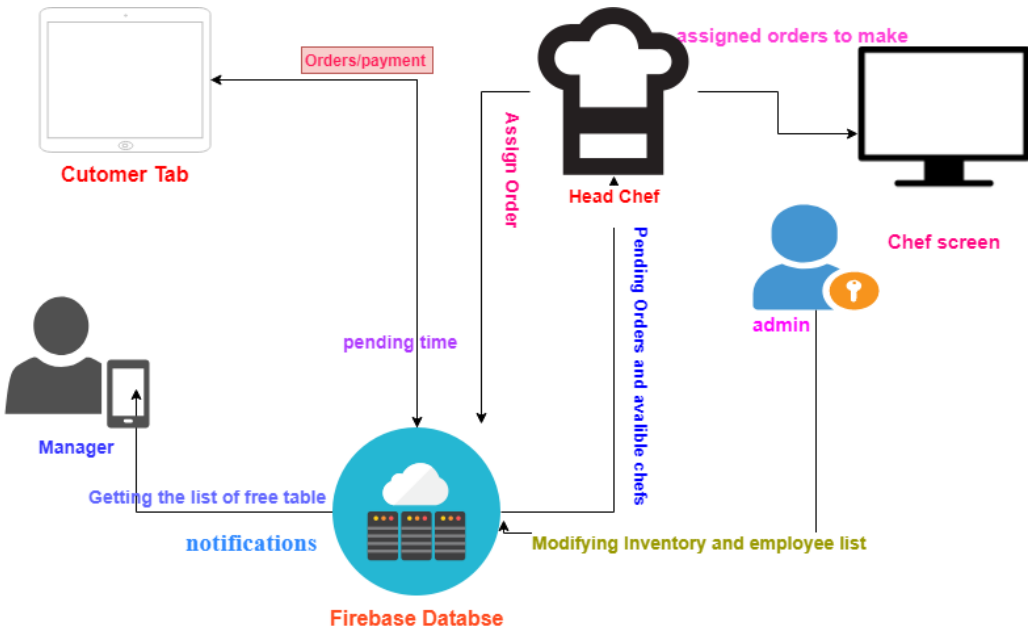


Fig1 - System Perspective Diagram

2.2 Product Functions

Given below are the major functions that can be performed using RMS 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

RMS'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.

1.1. 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.

1.2. 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.

1.3. 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.

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 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. 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.

4. 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.

3.2 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.

3.3 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.

3.4 Communications Interfaces

RMS 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.

4. System Requirements

4.1 Place Order

4.1.1 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

4.1.2 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.

   1.1. **Stimulus:**
   Customer taps on an unavailable dish.
   Response:
   Nothing happens.

2. **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.

   2.1. **Stimulus:**
   If Customer taps on cancel button after 1.
   Response:
   the system closes the popup, and nothing happens.

3. **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

4.1.3 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.

4.2 Customer Help

4.2.1 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

4.2.2 Stimulus/Response sequences

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

4. **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”

5. **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.

4.2.3 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.

4.3 Chef Order Queue

4.3.1 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

4.3.2 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.

4.3.3 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.

4.4 Edit Order

4.4.1 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

4.4.2 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.

2. **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".

3. **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.

4. **Stimulus:**
   Customer taps on "Remove Dish"
   Response:
   system responds with "Dish <Name> removed" or "Dish <Name> cannot be removed"

5. **Stimulus:**
   Customer taps on any new dish which was not previously selected
   Response:
   stimuli/Responses of “Place Order” feature will be followed.

4.4.3 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.

4.5 Cancel Order

4.5.1 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

4.5.2 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”

4.5.3 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.

4.6 Mark Dish as Cooked

4.6.1 Description and Priority

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

priority:
high

4.6.2 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.

4.6.3 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.

4.7 Request Bill

4.7.1 Description and Priority

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

priority:
high

4.7.2 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".

4.7.3 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.

4.8 Customer Feedback

4.8.1 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

4.8.2 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.

4.8.3 Functional Requirements

REQ-1: System must show the feedback screen to the user.
REQ-2: System must display multiple choice questions for feedback.

4.9 Add/Edit/Delete Staff Members

4.9.1 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

4.9.2 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

2. **Stimulus:**
   Admin fills the information and hit submit
   Response:
   System responds with “<Staff Member> added successfully”

3. **Stimulus:**
   Admin taps on edit button
   Response:

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

4. **Stimulus:**
   Admin edits the information and hit submit
   Response:
   System responds with "<Staff Member> edited successfully"

5. **Stimulus:**
   Admin taps on remove button on a particular row
   Response:
   responds with a "<Staff Name> removed successfully"

4.9.3 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.

4.10 Add/Edit/Delete Menu Items

4.10.1 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

4.10.2 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

2. **Stimulus:**
   Admin fills the information and hit submit
   Response:
   System responds with "<Dish> added successfully"

6. **Stimulus:**
   Admin taps on edit button
   Response:
   System opens a screen with a form prefilled with the existing values.

7. **Stimulus:**
   Admin edits the information and hit submit
   Response:
   System responds with "<Dish Member> edited successfully"

8. **Stimulus:**
   Admin taps on remove button on a particular row
   Response:
   responds with a "<Dish> removed successfully"

4.10.3 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.

5. Nonfunctional Requirements

5.1 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.

Cancel Order/ updates must be made with little delay to avoid delivery delay. Also, when connecting to the Firebase server the delay to make a successful connection should be less for effective real time communication.

5.2 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.

5.3 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

5.4 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 Reusability:

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

5.4.10 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.

5.4.11 Testability:

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

5.4.12 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.

5.5 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 A: Glossary

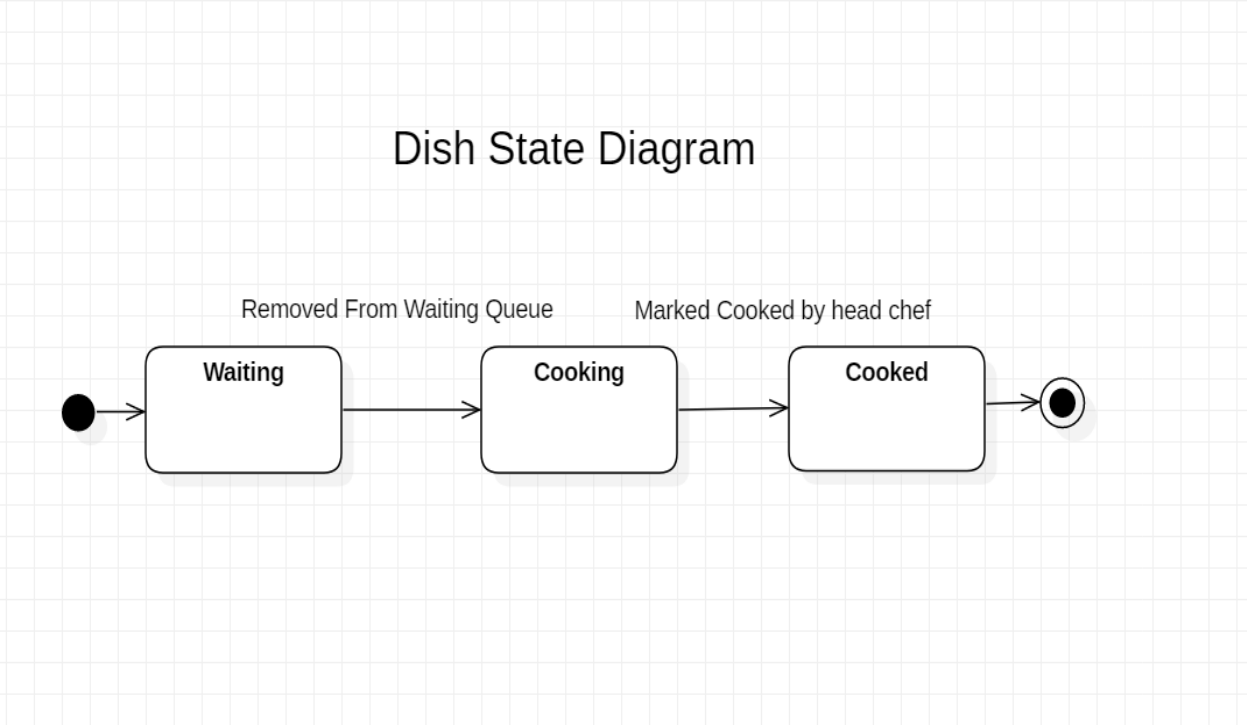
CRUD: *In computer programming, create, read, update, and delete (CRUD) are the four basic functions of persistent storage. Alternate words are sometimes used when defining the four basic functions of CRUD, such as retrieve instead of reading, modify instead of update, or destroy instead of deleting.*

Print Helper: *It is an android library that is used to connect to remote printer and send commands to that printer for printing.*

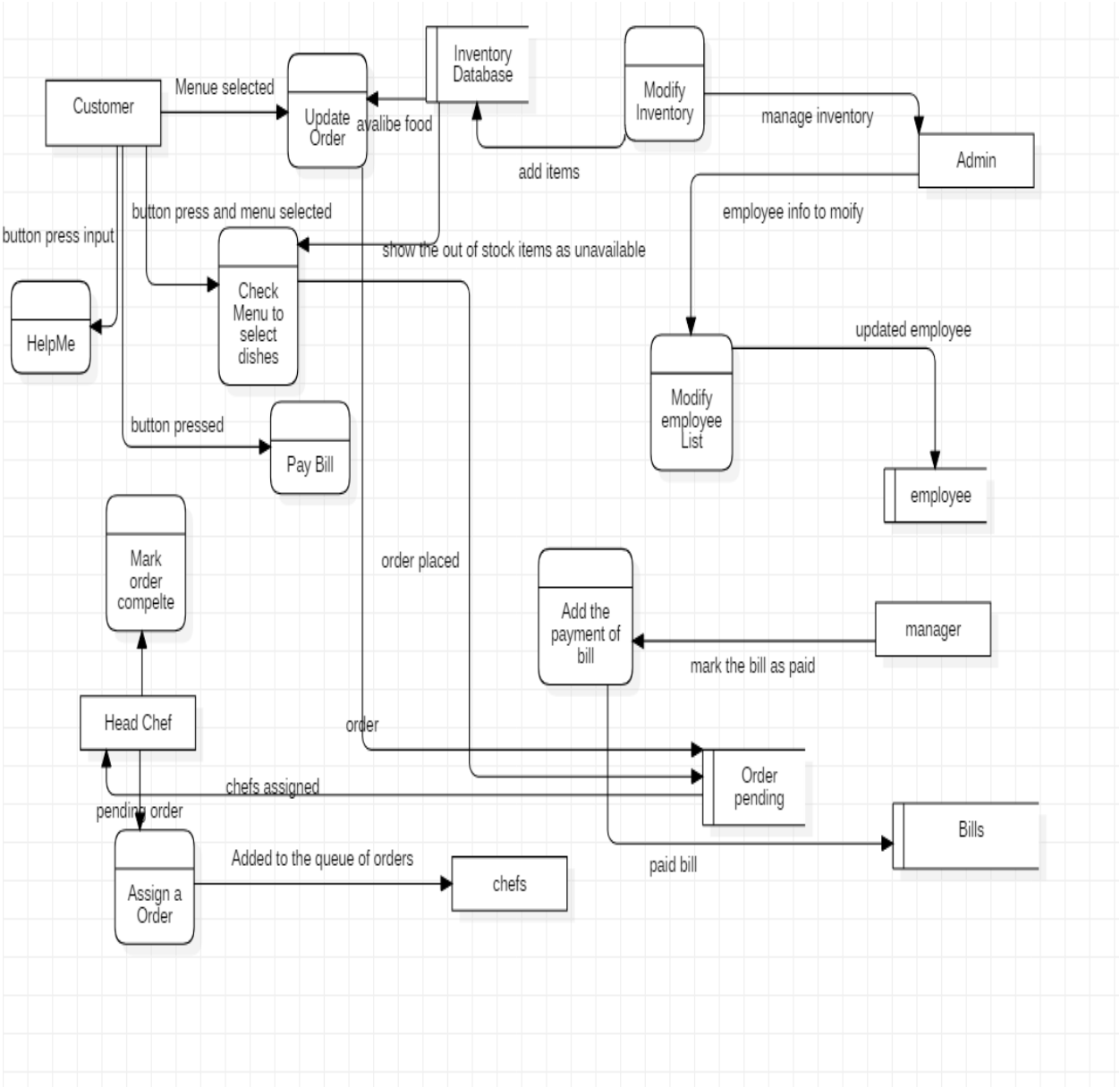
marshmallow 6.0: *Android 6.0 "Marshmallow" is the sixth major version of the Android operating system and the 13th version of Android. First released as a beta build on May 28, 2015, it was officially released on October 5, 2015, with Nexus devices being the first to receive the update.*

Appendix B: Analysis Models

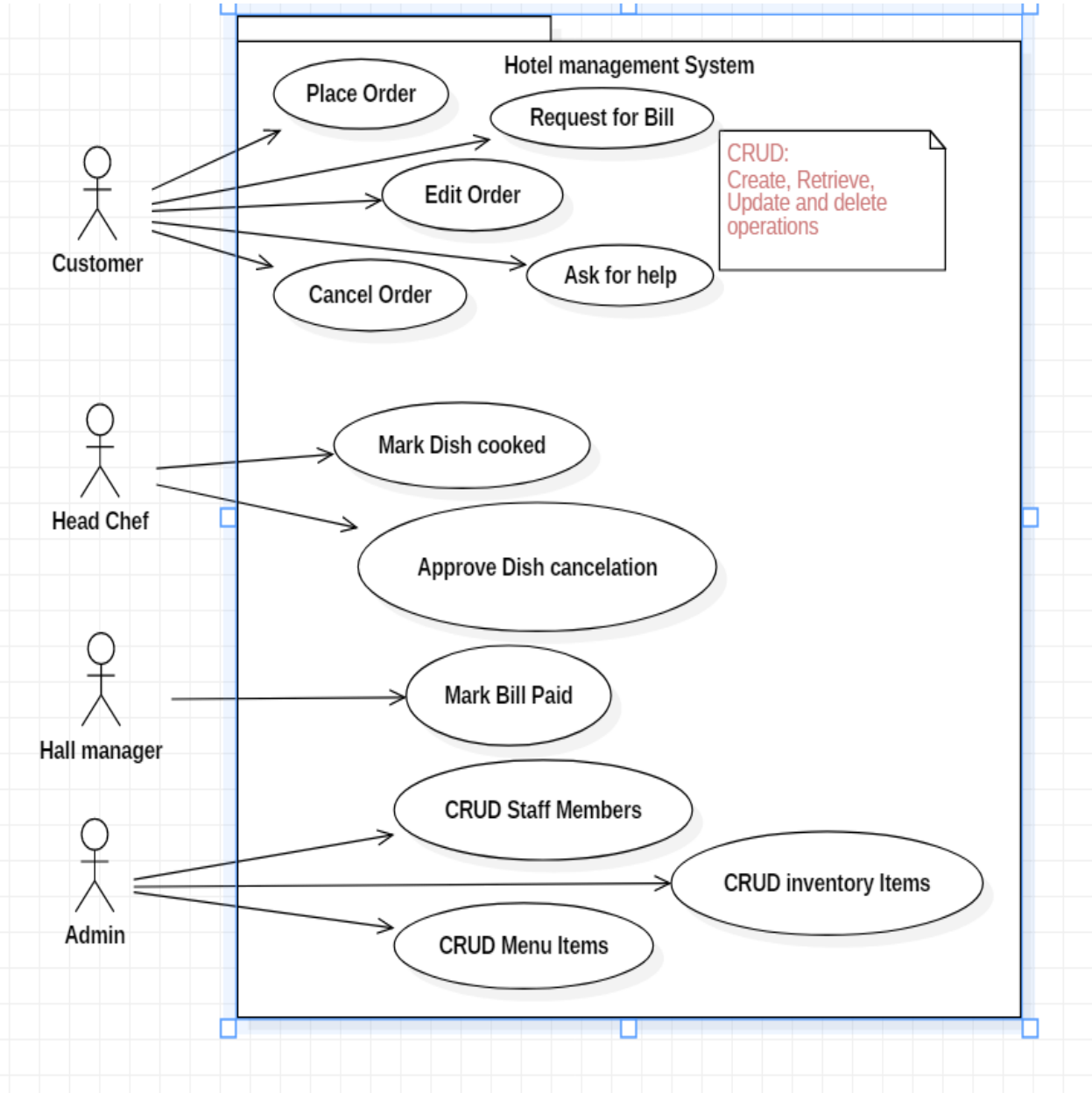
Dish State Diagram



Data Flow Diagram



Use Case Diagram



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.
Adding POS (point of sale) features to the application is yet to be determined as well.