Software Requirements Specification

for

Restaurant Management System

**Version 1.0 approved**

**Prepared by Aiza, Ali & Haseeb**

**Sphynx Software Solutions**

**November 26**

**Table of Contents**

[**Introduction**](#_heading=h.ejv3q7sqet0t) **3**

[Purpose](#_heading=h.3znysh7) 3

[Document Conventions](#_heading=h.wmwsfj7yy0af) 3

[Intended Audience and Reading Suggestions](#_heading=h.whh1nfcjj0pw) 3

[Product Scope](#_heading=h.x6b2m4aqfndf) **3**

[**Overall Description**](#_heading=h.usjc2kne17tf) **4**

[Product Perspective](#_heading=h.2s8eyo1) 4

[Product Functions](#_heading=h.hi538oljd50o) 4

[User Classes and Characteristics](#_heading=h.ho4dh5pyeaie) 4

[Operating Environment](#_heading=h.16bl2nseizkm) 4

[Design and Implementation Constraints](#_heading=h.qh3eftcn6f9g) 5

[User Documentation](#_heading=h.k20clyx5meod) 5

[Assumptions and Dependencies](#_heading=h.842ztz49wqas) **5**

[**External Interface Requirements**](#_heading=h.u2m44fx1e87e) **6**

[User Interfaces](#_heading=h.2jxsxqh) 6

[Hardware Interfaces](#_heading=h.vk36ryodlowc) 6

[Software Interfaces](#_heading=h.jsxopt408q8l) 6

[Communications Interfaces](#_heading=h.f23bz1h6f9pf) **6**

[**System Features**](#_heading=h.6vyq2todkmbx) **7**

[Automated Food Ordering System](#_heading=h.1jyura6xnf5p) 7

[Table Management System](#_heading=h.frk7vsr7u1da) **7**

[**Other Nonfunctional Requirements**](#_heading=h.3whwml4) **8**

[Performance Requirements](#_heading=h.2bn6wsx) 8

[Safety Requirements](#_heading=h.hw4ujasx6yd9) 8

[Security Requirements](#_heading=h.ecr72sltixtv) 8

[Software Quality Attributes](#_heading=h.8nqeljavpssm) 9

[Business Rules](#_heading=h.9mfb12sjllwv) **9**

[**Other Requirements**](#_heading=h.5biost8f4chs) **9**

# Introduction

## Purpose

This project is planned to enhance the customer and consumer experience in the challenging area of restaurant management. Since we are following the Agile Process Model, specifically Scrum, there will be several increments to this software, which will help us build the software according to the dynamic needs of our unique customer. The scope of this project will include the reception-side customer handling, customer-side food ordering system and chef-side order assignment.

## Document Conventions

We have used the standard IEEE formatting conventions to write this document. Please expect no changes from the official format other than the following:

|  |  |
| --- | --- |
| DB | Database |
| RMS | Restaurant Management System |

## Intended Audience and Reading Suggestions

This SRS document is intended for the use of the developers at Sphynx Software Solutions mainly, and of the customer (the restaurant) contact who follows up with this project.

## Product Scope

The product we are developing is intended to deal with the inefficiencies of the paper-based manual restaurant management system at the customer restaurant. We are developing a system that will computerize handling customers, ordering food and assigning orders in the kitchen in an efficient manner. Restaurant’s supply chain management is out of the scope of this product.

# Overall Description

## Product Perspective

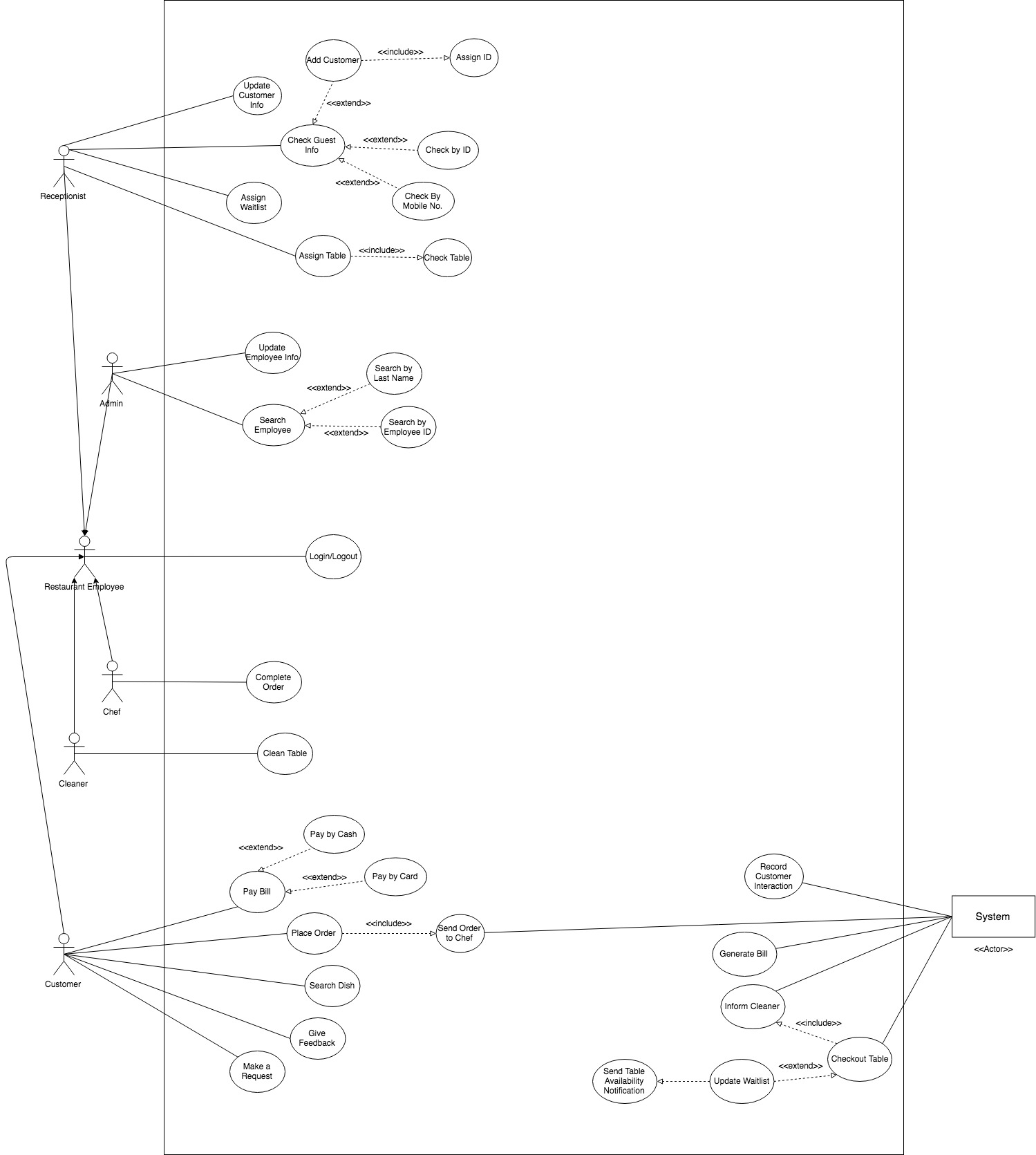
We are building a restaurant management system. It is a self-contained product that the staff of the restaurant will use to create a smooth experience for customers and as well as the restaurant staff.

## Product Functions

The functionalities of the system would be the following:

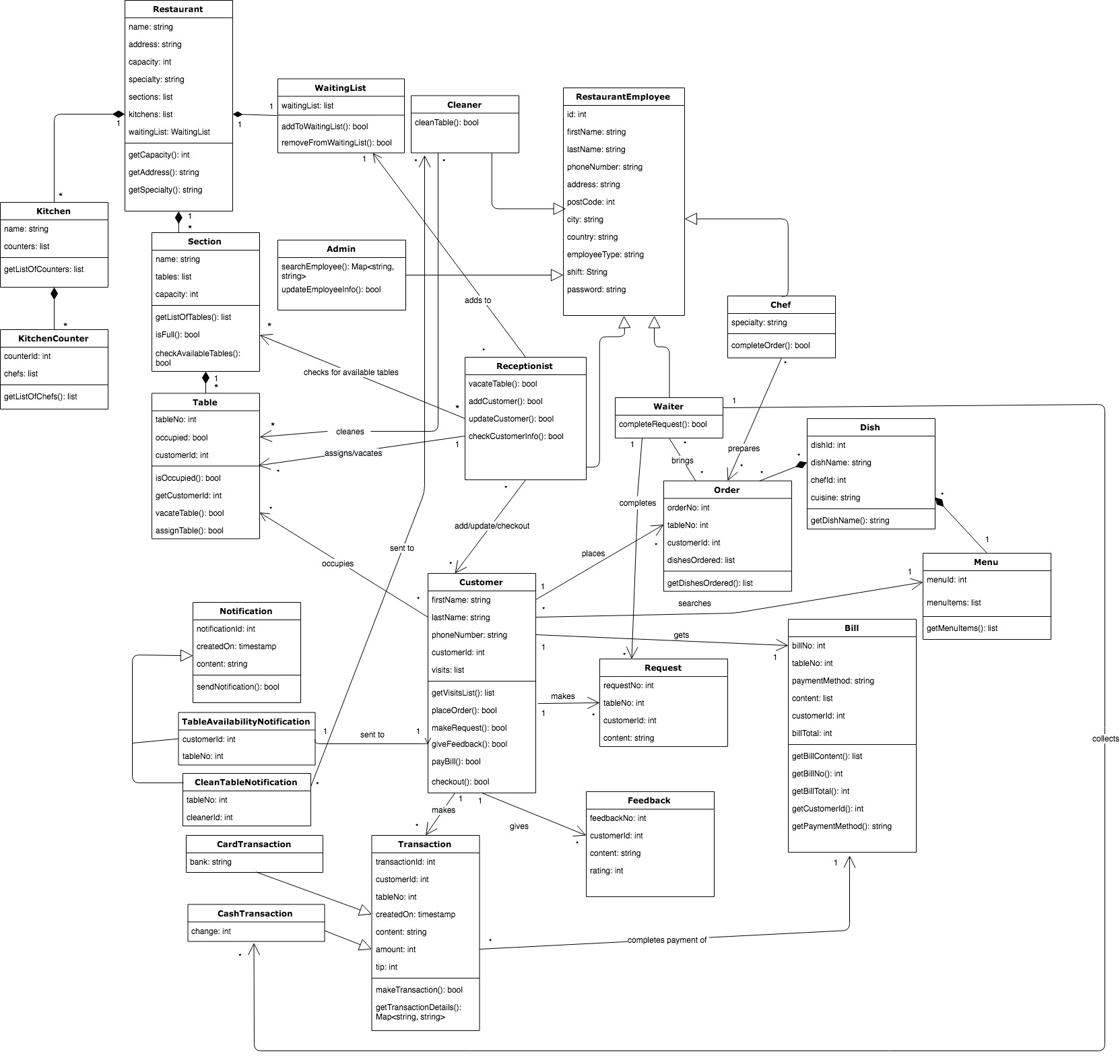
* Present the receptionist with an interface to assign table to customer.
* Allow receptionist to place customers on a waiting list.
* Present the menu to customer for ordering food.
* Store an order and send it to the appropriate chef.
* Upon the confirmation of the order, show an approximate countdown timer for the order.
* Present a customer with their bill and allow them to make payment.
* Allow customers to make requests or leave feedback.
* Notify janitor after a table has been vacated.

The detailed use case document can be viewed below:



## User Classes and Characteristics

The user classes and characteristics can be seen in the class diagram below:



The user classes are the following:

* RestaurantEmployee: this is a parent class for all the classes related to the staff members in the restaurant. Its children are the following classes:
  + Receptionist: this class will be most frequently used. The receptionist will be able to check the customer in the system, add a new customer, edit a customer’s information, assign a table to a customer, assign the waiting list, and check-out a customer.
  + Chef: the chef will be assigned dishes according to customer’s orders. This class will be frequently used for assigning orders and marking completed orders.
  + Waiter: this class will be used to interact with the Requests class which corresponds to the requests of the customers
  + Admin: the admin can solely search and update employee information. This will be privileged information, so it will be secure.
  + Cleaner: they will be assigned a table as soon as it is vacated to clean it.
* Restaurant: this class represents the restaurant in a high-level sense. It contains the demographics of the restaurant, its specialty and capacity.
* Kitchen: this class represents the kitchen of the restaurant and contains all the kitchen counter.
* KitchenCounter: this class represents the working place of chefs. Every counter has a function; for example, a dessert kitchen counter which will have all the pastry chefs in it.
* Section: the restaurant’s seating sections are represented by this class. It contains tables.
* Table: the customer will be assigned a table. This class interacts with the Customer class and Receptionist class.
* WaitingList: this is a singleton class. The Receptionist class will be able to add Customers to it.
* Notification: this is a parent class for notifications:
  + TableAvailabilityNotification: this class is initialized when a table is vacated and there is someone on the WaitingList. The notification is sent to the mobile of the customer who will get off the waitlist.
  + CleanTableNotification: this class is initialized when a table is vacated. The notification is sent to one of the cleaners.
* Transaction:

## Operating Environment

* Firebase Database
* Heroku server
* Platform: Node.JS

## Design and Implementation Constraints

The following are regulatory and corporate policies:

* Code of Conduct
* Privacy Policy
* Data Handling Policy
* Anti-Harassment and Non-Discrimination Policy

The following are software design and implementation constraints:

* Access to the web is required.
* Implement the database at least using a centralized database management system.
* Use the latest version of Node.JS.
* Follow the Javascript naming conventions.
* Don't use protected inheritance.

## User Documentation

Users will be given an online tutorial to understand the workings of the system. We will also set a quiz for users as a prerequisite before using the system.

## Assumptions and Dependencies

It is assumed that the user is familiar with an internet browser and also familiar with handling the keyboard and mouse.

Since the application is a web based application there is a need for the internet browser. It will be assumed that the users will possess decent internet connectivity.

# External Interface Requirements

## User Interfaces

The user interface of our system will be kept as simple as possible to make it as lay-person friendly as possible. Our market survey indicates that the hospitality industry management prefers simplistic graphical interface and layout when it comes to computer-based management systems. We will use Bootstrap 4 for this. Every page will have a navigation bar. The front desk page (accessed by the receptionist) will have the following: a map of the seating area with tables color coded according to their availability, an add customer button, a waiting list, and an assign-waitlist button that will open a pop-up form where the receptionist can enter guests’ details to assign them on the waiting list. The menu page, which will be accessed by the seated customers, will have the following user interface: list of menu items with a quantity button next to them, and a button to place the order. The orders page will have a list of orders displayed according to which chef has the page open. This will help by only displaying orders that are relevant to each chef in the kitchen. The pay-bill page will contain a card payment system, a cash payment option and a feedback survey.

## Hardware Interfaces

The hardware interface of the system will consist of a database and a server. The communication protocol will be standard.

## Software Interfaces

The software used by this system is Node.JS, Express, Firebase database, Javascript for front-end and back-end, and Bootstrap for the user interface. The database must be the central storage of data.

## Communications Interfaces

The payment system will utilize the SSL protocol for secure payments and to protect the sensitive information of the customers. The SMS part of the system (required for the waiting list) will follow the MAP part of the SS7 protocol. And, we will utilize the sockets and the database for synchronization across pages and uses.

# System Features

Our software product will the following features:

* Reception-side features:
  + Map of tables in the restaurant
  + Assign customer to a table
  + Checkout customer
  + Assign to waiting list
* Customer-side features:
  + Digital menu on table
  + Make a request
  + Give feedback
* Chef-side features:
  + Dishes assigned according to type of chef
* Cleaner is informed when a customer checks out

## Reception-Side

4.2.1 Description and Priority

The reception-side features will act as the backbone of the restaurant. Using the app, the receptionist will be able to look up existing customers, add a new customer, add someone to the waiting list, and assign a table using the tables map. These features are high priority.   
Priority Ratings:

|  |  |
| --- | --- |
| Benefit | 9 |
| Cost | 3 |

4.2.2 Stimulus/Response Sequences

These features will provide a simple interface to the receptionist to handle customers coming to the restaurant efficiently. The receptionist will not have to look at any registers or ask the waiter for available seats. They will simply be able to open the tables reception side of the app and do everything from there. This will cut customers’ waiting time in half, lead to a more stress-free environment and increase the satisfaction of everyone in the process. It will also reduce errors.

4.2.3 Functional Requirements

We will use Firebase DB and a Node.JS server as mentioned in the previous section.

REQ-1: Node.JS Server

REQ-2: Firebase DB

REQ-3: Trained Receptionist

## Food Ordering System

4.1.1 Description and Priority

This feature is the main component of this project. It is an extremely high priority.  
Priority Ratings:

|  |  |
| --- | --- |
| Benefit | 9 |
| Cost | 5 |

4.1.2 Stimulus/Response Sequences

The system will provide a simple ordering system on tablets on each table. The restaurant’s customer will order food through the tablet on the system and the chef in the kitchen will be effectively notified of what is to be prepared. This will cut the waiters' work in half and also reduce errors and pedestrian traffic in the restaurant.

4.1.3 Functional Requirements

We will use Android tablets that will be more responsive in their compatibility.

REQ-1: Android Tablets

REQ-2: Trained Chefs

# Other Nonfunctional Requirements

## Performance Requirements

All the reception side features must take less than 3 seconds to execute including making necessary database calls. This is important because we want to cut the waiting time of the user and assign them a table or a waiting list spot as soon as possible.

On the customer’s side the menu should always load with images. In case of a bad internet connection, the photos can downgrade in quality but they should still accompany the dishes.

On the chef’s side, the order cards should always be updated as pushes as soon as a new order comes in; the chef shouldn’t have to load the page.

## Security Requirements

Since the app contains sensitive information of the customers and employees, we must make sure that we adhere to the data privacy policy. Moreover, it must be made sure that chefs can only see their side of the app, customers can only see their own side and the receptionist can see the receptionist’s side of the application.

## Software Quality Attributes

The system should be adaptable to any size of the restaurant to accommodate possible changes in the restaurant structure. It should always be available during the restaurant’s operating hours. It should display correct information in all parts of the app and it should be robust. It should also be extremely easy to use.

## Business Rules

* Only the receptionist can search a customer’s information, add a new customer, add someone to the waitlist, assign a table, and checkout a customer. These are the only set of functions that they perform
* The customers can only look at the menu, place an order, make a request, leave feedback and pay their bill.
* The chef can only complete an order once it is made.
* Only the admin can search and update employee information.
* The cleaners can only clean a table. They don’t have access to any other information or function. They are automatically informed by the system once a table is emptied.
* The waiters can only complete a request. They don’t have access to any other information or function.
* As soon as a table is checked out, if the waiting list has someone on it, a notification of table availability should go to them.