Report on
"Online Restaurant Management System "

Submitted in partial fulfillment of the requirements for ***Sem VII***

Software Engineering

Bachelor of Technology

in

Computer Science & Engineering

Submitted by:

Paromita Roy	***01FB16ECS247***
Poorva Tiwari	***01FB16ECS251***
Poorvi G Achar	***01FB16ECS252***
Pragna Prasanna	***01FB16ECS256***
Prajna Nagaraj	***01FB16ECS259***
Prerna Rao V	***01FB16ECS276***

Under the guidance of

Prof. SHREEKANTH M PRABHU

August – December 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Table of Contents

1. Feasibility Study

1.1. Problem Statement

Restaurants are getting increasingly popular day by day. There are several cuisines that customers enjoy, and *Tomatoz* is one such restaurant which provides the customers a satisfaction for their hunger. *Tomatoz* is a one stop food destination for all people who want to stop by for a quick bite, or a heavy meal.

Our application provides separate UIs for users, through which they can book a table, and view the menu. The waiters and restaurant staff have another view of the same application to view the order and it delivers to the customers.

Tomatoz application provides the users with an easy to use interface with a hassle free experience of placing orders, booking tables and paying bills.

1.2. Client/Sponsor/User

- **Client:** This application is required for the overall digitised administration of the restaurant.
- **Users:**
    - **Waiter:** Taking the order from the customers and the input is fed into the application which is accessible by the kitchen staff.
        - **Front Desk :** for assigning the tables for walk in customers
    - **Customer:** Any customer who wishes to dine at this restaurant can use the application. It will be publicly available as a web application.
    - **Kitchen Staff:** Update the app with the status of the order.

1.3. Financial Projections

Currently we would be giving away our platform for free (upto a duration of 6 months) to the customers (restaurants) as it is the beginning stage of the company. In future we could charge the company on a subscription basis which could either be quarterly or yearly.

Large part of the investment would go into expansion of the current product to increase the users as well as development of new features into the product.

Our investments would be targeted towards the cost of advertising, online cloud and hosting charges , marketing team, contract support for IT and training needs and additional developers (increasing the staff in general as the company grows).

1.3.1.Expenses:

1. Software Engineers: ₹5-₹20 LPA (per employee)
2. Marketing Staff: ₹3-20 LPA (per employee)
3. Advertising: ₹15-20 Lakhs
4. Cloud and Hosting charges: ₹50,000 per year

1.3.2 Revenue

The food industry is huge, even if we consider just Bangalore, which would consist of over 80k restaurants. This would mean a revenue of at least ₹50K per annum from each restaurant. Let's say the present competitors charge around ₹75,000-₹100,000 for complete outsourcing of the product for a year, we would keep a competitive price of

₹50,000 for such a tool. If we are able to capture the entire market we would be able to bring in a revenue of around **₹30-40 cr** over one year just in Bangalore. The sky's the limit if we are able to capture the entire Indian food industry.

1.4. Systems Objectives

The following are the services provided by our solution:

- ●Separate UIs for customers, the waiter, manager and kitchen staff (chef)
- ● The customer will be able to do the following:
  - ○ Reserve a time slot & a table even before entering the restaurant
  - ○ View the menu
  - ○ Place order for items
  - ○ Ask for help and give feedback
- ● The manager will be able to do the following:
  - ○ Monitor the inventory
  - ○ Add/Remove other employees
  - ○ View feedback
- ● The chef will be able to do the following:
  - ○ View/Update status of an order in the queue
  - ○ Modify the menu

The beneficiaries of our solution include the managers of restaurants and the customers who visit the restaurant. Our application allows the customers to reserve tables and place orders easily.

Our solution also helps the restaurant managers by displaying customer feedback and

item-wise number of orders so that the menu can be altered if needed.

1.5. Potential Benefits

- **Tracks the sales down to each item:** All transactions are captured by the system like the payments, the table scheduling to the customer, payment methods, online menu etc.
- **Generates the financial statements:** The bill is generated digitally so that the manual errors are avoided
- **Better Customer service:** This module records the customer details and it is very useful for getting the customer ratings and reviews.
- **Access the data from anywhere:** customer can access the menu of the restaurant from online anytime and place of their liking.
- **Efficient staff management :** The application with an employer can help in the overall staff management which in turn increases the efficiency of the restaurant. Eg: downsizing the staff during less peak hours etc.
- **Better efficiency between the kitchen, waiter and manager:** there is less manual error and increases customer satisfaction by encouraging faster service.

1.6. Resources Needed

- **Technical resources**
  - Domain name for the online web application
  - Software development tools for building a website
  - Web hosting platform

- Technologies : HTML, CSS, Javascript, GCP (Google cloud platform) firebase - using their real time data storage
- Hardware resources :
  - Laptops : Mainly for application development purposes.
  - Tablets : For day-to-day use in restaurant. Ex: scheduling table, taking orders, status update by kitchen staff etc
- Cloud technology would be used so as to scale up/down. This would also reduce cost and risk of infrastructure management.

- **Human Resources**
  - Product Designer : the overall product designer who can present to the client as to how the product should appear. Creating a bridge between the implementer and the business developers.
  - Full Stack developers: developers who can work on the front end and the back end of the application.
  - Sales and marketing team: selling the product and identifying the customers and identifying the potential customers.
  - Creative designer: to design the logo and the creative elements of the application
  - Software development Engineers: for optimisation of the code and testing of the product.
  - Cloud Platform Service Provider and cloud operator: for the smooth functioning of the application running on the cloud
  - Customer service executives: people required for answering the client and

user queries.

- **Time**
  - The time will be a major resource as the project can be phased on the time required for the different development cycles like project schedule development, design approvals, marketing, testing and debugging etc.
  - The overall development cycle of the project breakdown:
    - Design approval and project planning: 1 week
    - Project budgeting and discussion: 1 week
    - Front end and back end development (with cloud hosting): 4 weeks
    - Testing and debugging: 1 week
    - Deployment: 1 week

1.7. Potential Project Organization

Team members are assigned responsibilities over the course of developing the project; responsibility demarcations may disappear, to meet the project deadlines.

It is a flat organization, where progress, backlogs, new requests and responsibilities are handled, discussed as a team and new phase plans are mapped out as and when required.

- Backend Developers
  - Responsible for setting up and maintaining the database.
  - Manages the code for interacting and communicating between the front end, backend, stack and database.
- Frontend Developer

- Responsible for developing the UI for the project.
- Responsible for making the website attractive and user friendly.
- The website should be transparent with no additional hidden costs.

- Software Testing Engineering
  - Responsible for checking if the different modules meet the specifications and requirements of the project.
  - Identify, bugs/ faults and requirements that have not been met in the project.
  - The tester will ensure if the correct functionality is fired and data is displayed correctly.
- Currently assigned roles:
  - Paromita Roy — Frontend, Testing
  - Poorva Tiwari — Backend, Testing
  - Poorvi Achar — Frontend, Testing
  - Pragna Prasanna — Frontend, Testing
  - Prajna Nagaraj — Backend, Testing
  - Prerna Rao V — Frontend, Backend

1.8. Marketing Strategies

- Marketing is very much required to expand the product as well as our customer base.
- The first few customers will be awarded with 25-50% discount on the yearly and quarterly subscriptions.
- If the client suggests the product to other potential clients in the market then there

will be subsequent promotional codes and discounts that will be offered.

- Digital marketing on various social media websites with people assigned to do the same with everyday follow ups.
- Face to face interaction and and business presentations by identifying prospective customers
- Direct mailing and messages to the customers who show interest with regular follow ups.

2. PROJECT PLAN

2.1 INTRODUCTION

2.1.1 PROJECT OVERVIEW

Our application provides separate UIs for users, through which they can book a table, and view the menu. The waiters and restaurant staff have another outlook of the same app to view the order and hand it over to the customers. '*Tomatoz'* application provides the users with an easy to use interface with a hassle free experience of placing orders, booking tables and paying bills.

2.1.2 PROJECT DELIVERABLES

Task/ Deliverable	Completion Date	Status
Requirement document	22/10/19	Complete
Planning document	22/10/19	Complete
Design document	22/10/19	Complete
Mock Backend setup	29/10/19	Complete
Mock Frontend setup	29/10/19	Complete
Implementing API's	29/10/19	Complete
Refined Backend	12/11/19	Complete
Refined Frontend	12/11/19	Complete
Final Testing of system	20/11/19	Complete
Release of product	25/11/19	Complete

2.2 Project Organization

2.2.1 Process model

Agile model with SCRUM approach will be followed.
As the team will be divided into sub teams that will handle a specific part of the project.
Any changes in requirements, updates will be discussed at regular intervals of time.

Figure 1: Agile model, one scrum cycle

2.2.2 Organizational Structure

The development of the project will require an iterative model with continuous phases of development followed by phase by phase testing, with a window to accept new requirements if any required.

The project will move through the following:

Customer
The customer is responsible for specifying the different aspects of the project including the various functionalities and customer specifications.

Requirement manager and an Analyser

Responsible for eliciting all the requirements, and verifying that the software developed is aligned with the requirements specified initially.

Project Manager/ Lead
Oversees all aspects of the project including backend and frontend development

Software - Developers
Responsible for setting up and maintaining the database, and managing all the code for communication between the frontend and backend database.

Front End Developers
Responsible for developing the user interface for the project.

Tester
Responsible for stress testing the product and identifying any bugs/falls in the product development phases.

2.3 Managerial Process

2.3.1 Management objectives and priorities

Team leadership will be assigned the responsibility of maintaining and containing the project schedule, and ensuring that all deliverables are completed as per the plan. Emphasis will be on the efficiency, transparency, accountability, and effectiveness of the product that is released.

As a team some goals would be:

- Prioritize highest-value items first.
    - Storing and keeping tabs on orders, reservations updated
- Make time for important but not urgent work.
- Members holding each other accountable to top priorities.
- Members making suggestions for process improvements.

2.3.2 Assumptions, Dependencies, and Constraints

2.3.2.1 Assumptions

- It is assumed that the members of the team have the necessary knowledge required to execute this project, and that project management is available.
- The Tomatoz system also assumes that the device being utilized has all the resources(memory and processor) needed to run the application, i.e, an Android-based mobile phone or tablet or through a web browser on a Windows-based computer.
- It is assumed that the client has a cloud-based database available to run application.
- It is assumed that application users will have a functional internet connection with the designated server.
- It is assumed that the users of the application are aware of the application's functionalities.

2.3.2.2 DEPENDENCIES

- GCP (google cloud platform) is available throughout the development of the project
- Stable internet connection for development of the project

2.3.2.3 CONSTRAINTS

Time is considered the most important constraint in developing the *'Tomatoz'* - Online Restaurant Management System. Because of the limited development time, the document submissions must not stray too far from the stated deadlines. Progress must be maintained at a consistent pace to deliver the required deliverables by the target dates. Availability of hardware and software, too, would be a constraint.
Apart from these, some general constraints are:

- It must be coded efficiently enough to run well on the provided server hardware.
- Client-side code and/or web pages must be able to run efficiently
- The database will be created and maintained in a way that makes it reasonable and of manageable size.
- Must write secure code and the system should be capable to handle any security threats.
- Must pass the required Acceptance Criteria

2.3.3 RISK MANAGEMENT

NO.	RISK	CATEGORY	MONITORING AND CONTROLLING
1.	Time required for each phase is not accurately known, i.e, failure to meet	Estimation/Management	Set up milestones, and due dates in

	deadlines.		advance for each deliverable.
2.	Inappropriate version of tools and components	Software	Select specific versions to use and make it a standard throughout the project.
3.	Team member unavailability	Staff/Management	Ensure scheduling/work allocation is done properly. Substitute members should always be available.
4.	Requirements Change	Customer	Client will be made aware, in advance, of the amount of change that can be accommodated within the term of the project.
5.	Inefficient/heavy code slowing down the application	Programming	This would cause a lot of waiting and frustration for customers, can be avoided by simplifying code functionality, having competent tech experts and setting certain standards that the application has to match.

2.3.4 Monitoring and Controlling Mechanisms

Monitoring and controlling processes oversee all the tasks and metrics necessary to ensure that the approved and authorized project is within scope, on time, on budget so that the project proceeds with minimal risk.
It is continuously performed throughout the life of the project.

Monitoring:

1. **Weekly meetings**
    a. with a status report by each team member, are held to discuss progress, issues faced, new requirements and specifications that need to be handled.

2. **Monthly review:**
    a. Overall review of where the product is at.
    b. Checks the health of the product
    c. Go over the specifications to find any that are missed out or issues that have not been addressed

Controlling Mechanisms:

1. Strenuous testing of the system to identify bugs and cases that have not been handled

2.3.5 Staffing Plan

Role	# Staff	Skill set	Duration needed
Frontend developer	2	Javascript, HTML, CSS	2 months
Backend developer	2	Firebase Cloud Service : Authentication & Realtime Database	2 months
Tester	2	Understanding of app	2 months

2.4 Technical Process

2.4.1 Methods, Tools and techniques

For building this platform, the following tech stack would be required:

1. HTML, CSS, JavaScript: For building the front end of the '*Tomatoz'* web application.
2. Synchronous connection with firebase
3. Firebase Cloud Service: Back-end as a service.
    a. Realtime Database - A real-time cloud hosted database system which uses real-time processing to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time. The Firebase

Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.

b. Authentication - Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It uses an internally modified version of scrypt to hash account passwords. Even when an account is uploaded with a password using a different algorithm, Firebase Auth will rehash the password the first time that account successfully logs in. Accounts downloaded from Firebase Authentication will only ever contain a password hash if one for this version of scrypt is available, or contain an empty password hash otherwise.
c. Free Services: Firebase supports 1GiB of stored data. Firebase provides the best back-end server, great database and analytics solution, and useful integrations with other Google products. Most of all, users like that it's free to use and has affordable subscription options. A wisely designed CMS solution guarantees project scalability and data security. In the development process, up to 40% of efforts are made on building a flexible CMS structure and Admin panel.
d. Storage - Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

2.4.2 Software Documentation

All the documentation will be available from the application. It will be stored on a cloud platform, so that multiple users can access the files at once in real time. In the event that a document is edited improperly, revision control will be used to restore the document.
The documentation will include:

- Feasibility Study
- Software Project Management Plan (SPMP)
- Software Requirements Specifications (SRS)
- Software Design Document (SDD)
- Software Testing Document (SDM)

2.5 Work, Packages, Schedule and Budget

2.5.1 Work packages

A work package is a group of related tasks that are defined at the same level within a work breakdown structure. It will be divided into 3 functional sections:

- **Software Engineering** - this section will encompass all the engineering research, planning, design, and required documentation.
  It will comprise of the Software Development Phase, i.e,
  - Data Management Process (Database activity)
  - GUI Process (Designing UI)
  - Web Interface Development (Functionality, tools)
- **Software Quality and Testing** - This section involves testing and validation of the project design and documenting the development process. This section will test the interface (GUI) of the application, the process of restaurant management, and the ease of accessibility.
- **Software Implementation** - This section involves software integration and module programming need for a successful project implementation.

2.5.2 Resource Requirements

Resources required	Support software	Computer hardware	Time required
Computers / laptops	linux/ windows OS	memory/ storage	Throughout project
Cloud support	Firebase Cloud Service	nil	Throughout project
Testing methodologies	Unit/System testing	memory/ storage	Throughout project

2.5.3 Budget and resources allocation

2.5.3.1 Budget

Covered in the feasibility study under section 3.

2.5.3.2 Resource Allocation

1. Setting up and running the server
2. Cloud based services pricing.

2.5.4 Schedule

'Tomatoz' - the online restaurant management system is expected to take three months from project approval to launching of system. The system has to be developed from scratch since it's a new idea. The following is a high-level schedule of some significant milestones for this initiative.

Task/ Deliverable	Date
Mock Backend setup	29/10/19
Mock Frontend setup	29/10/19
Implementing API's	29/10/19
Refined Backend	12/11/19
Refined Frontend	12/11/19
Final Testing of system	20/11/19
Release of product	25/11/19

Upon approval of this project a detailed schedule will be created by the assigned project team to include all tasks and deliverables.

3.REQUIREMENTS

3.1 INTRODUCTION

3.1.1. PURPOSE

'Tomatoz' is a web application that aims to digitalize the process of various restaurant management operations including ordering and inventory management and Point of Sale(POS). This section aims to capture the system requirements and features particularly related to ordering and inventory management to be implemented in *'Tomatoz'* version 1.0., with the later releases on POS (Point of Sale).

3.1.2 DOCUMENT CONVENTIONS

3.1.2.1. PRIORITY CONVENTIONS

We will mention priority as "low" or "high" throughout the section. Secondly, priorities

are only mentioned in section 4 and section 5 along with a detailed description of the requirements. Any high-level requirements mentioned elsewhere are assumed to inherit priorities of their detailed counterparts in section 4.

3.1.2.2. FONT 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 italics in the text. On the other hand, those terms which are significant (but not described in glossary) are bold in text.

3.1.3. INTENDED AUDIENCE AND READING SUGGESTIONS

The purpose of this document is to give a detailed description of the requirements for the '*Tomatoz*' 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.

3.1.4. PRODUCT SCOPE

In competitive business world, reducing operational cost and increasing productivity becomes paramount. *'Tomatoz'* 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, table allocation & reservation, and taking feedback. This release of the software would only deal with these tasks, 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 used for noting down customer orders, list of people waiting for tables etc. With this system the tasks would be performed in less time in a more efficient manner. 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. Furthermore,

customers will have the option to book tables if available. 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.

3.2 OVERALL DESCRIPTION

This section will give an overview of the *'Tomatoz'* 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.

3.2.1. PRODUCT PERSPECTIVE

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

The application will have five interfaces for various mentioned roles - Customer, Manager, Waiter, Chef and Admin. Manager can see/edit the status of available/reserved tables. Customer's interface will consist of a menu listing the different items and their prices. When the customer selects some dishes and places an order, it will be stored in "Orders" table in the Firebase database. Head Chef's interface will be such that he is notified of a pending order and is able to assign it to one of the available chefs, who are then able to see the new order on 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 waiter assigned for that table gets notified through the interface. Customer's interface has an option to request for the bill. Bill is printed from the stored value in Bill table. Admin can change and modify the Firebase database to add new menus or staff, edit current inventory, stock etc.

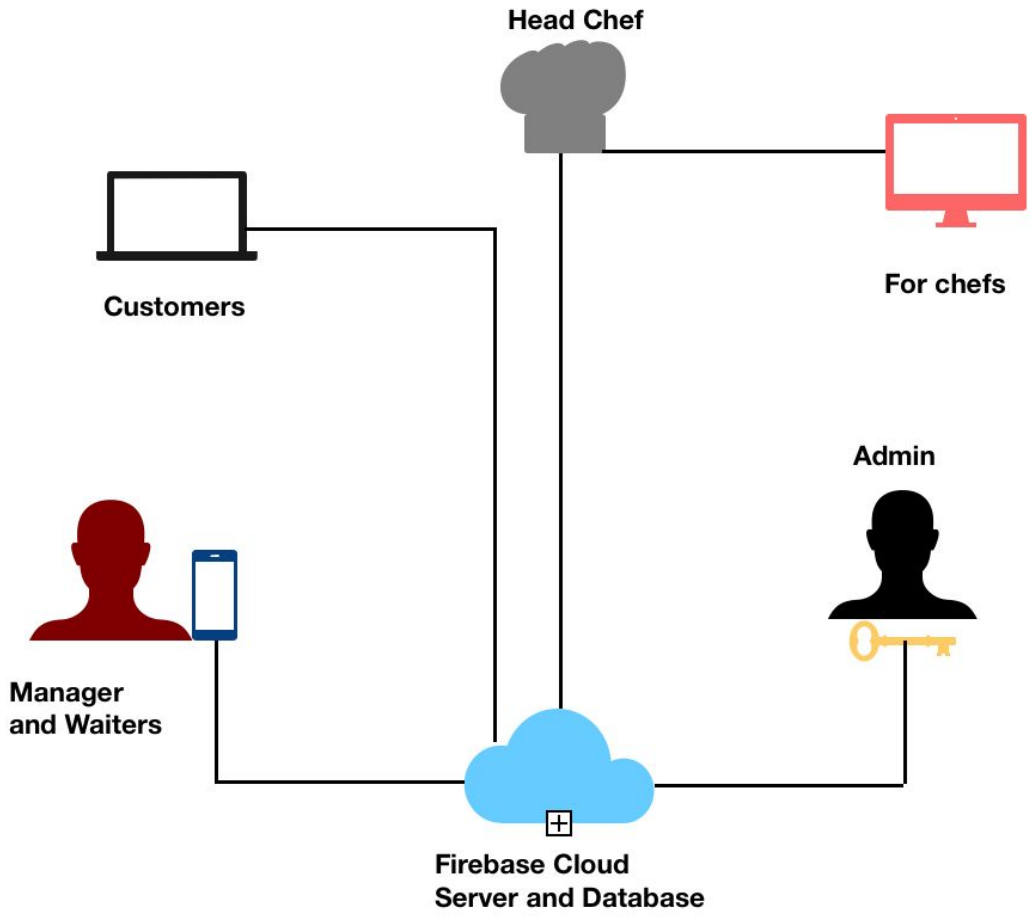


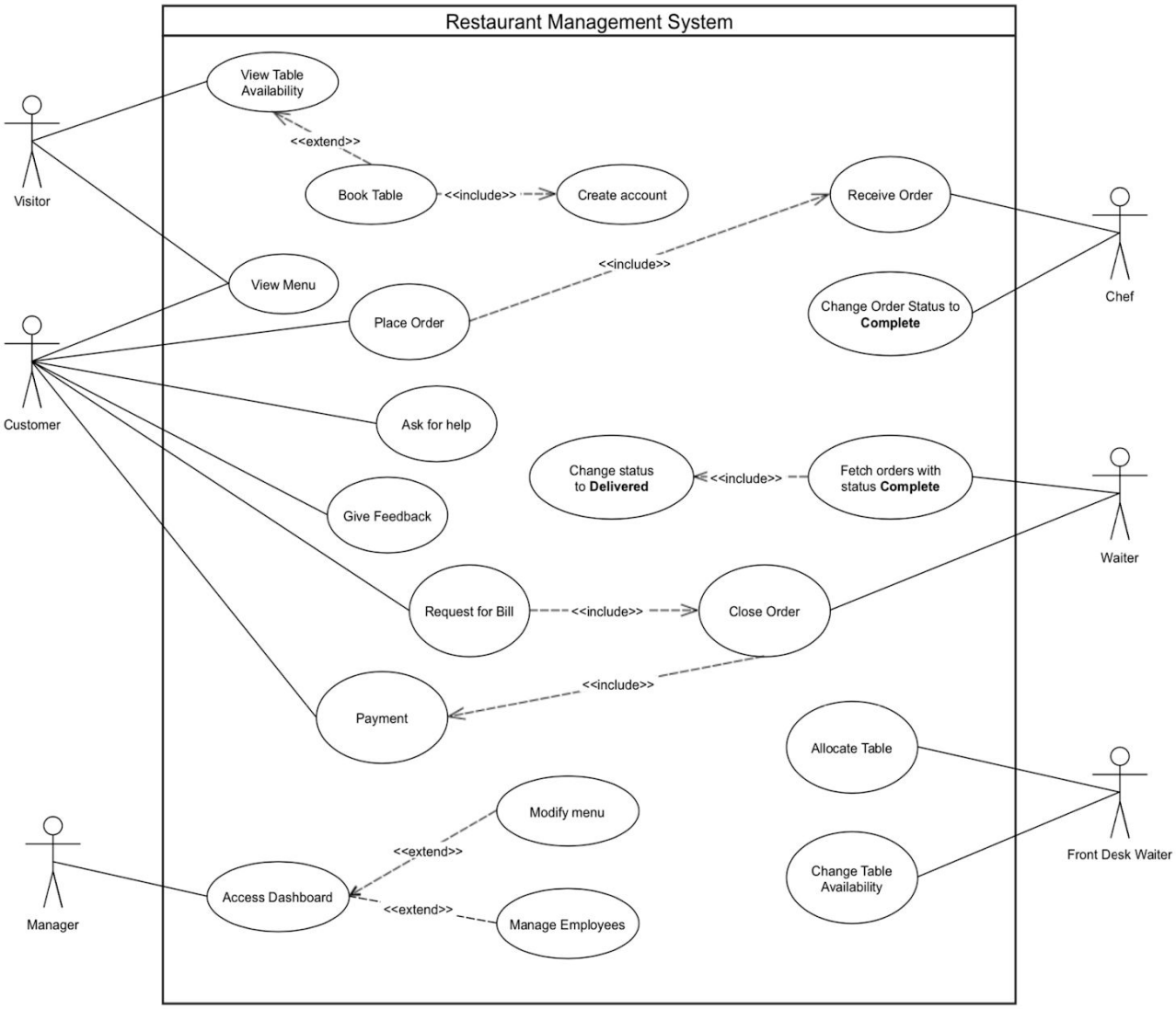
Fig 1.1 Product Perspective

3.2.2. PRODUCT FUNCTIONS

The various functionalities of the product includes:

1. Allow the customer to scroll through the information regarding the restaurant and go through the menu; this can be done through the web app, which can be accessed from home as well.
2. Allows the customer to book a table via the app or through Walk-in entry.
    a. There will be a 50% reservation for online booking and the remaining for bookings via walk-in or through a phone call. This ratio can vary depending on the number of tables available and the flow of the crowd.

3. Allows the restaurant manager to keep a constant check of the tables that are available and booked
4. Allow the admin to perform ***CRUD*** (create, retrieve, update and delete) operations on the staff members, menu items etc.
5. Allocating the table number to the user.
6. Allow the customer to view the menu from anywhere.
7. The customer seated within the restaurant can select/edit/cancel the food upon entering the table number via the account.
8. Once the order is placed by the Customer it will allocate a waiter for each order placed.
9. The order placed will be sent to the chef directly.
10. Show the order queues and the status for the chefs.
11. Once the chef receives the order he/she will update the status of the food upon the completion of preparation.
12. The waiter will check the status of the food prepared in the kitchen and deliver it to the table assigned to him/her.
13. Upon completion of the services to the user a bill will be generated on the app to the customer on the table
14. Allow the customer to give a feedback on the services that are provided by the restaurant and their overall experience



3.2.3. USER CLASSES AND CHARACTERISTICS

We will provide an interface to each of the classes depending on their characteristics and their functionalities pertaining to it. These are the following user classes that will be using our application:

3.2.3.1. CUSTOMER OR USER CLASS

Customer interacts with our online available application directly in order to book

a table in the restaurant. The initial process of booking the table is initiated by the customer from home.

In case, the customer walks-in to the restaurant, front desk waiter will assign table based on availability. The customers, however, can view menu and place order by using the given device.

3.2.3.2. WAITER CLASS

The waiter is responsible for keeping a check on status updates of the orders associated to the tables assigned and update status to **Delivered** upon serving the customer. This removes the order from the chef and waiter interfaces.

3.2.3.3. RESTAURANT MANAGER CLASS

The main job of the restaurant manager is to continually check the smooth functioning of the restaurant. He/she has a view of all the orders & their status, bill information and seating availability. The manager can also view complaint or feedback submitted by the customer.

3.2.3.4. ADMIN

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

3.2.3.5. CHEF

Chefs will receive a queue of orders. Once the order is prepared, the chefs will change the order status from **Progress** to **Complete.**

3.2.4. OPERATING ENVIRONMENT

It is an application running on a tablet and the tablets are present in a restaurant. Firstly, the manager would be present at the entrance and system in his tab would show the tables that are empty/reserved, directly from the database. The customers can reserve the tables online via their own accounts from everywhere. 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 are present in the kitchen area and their interface would allow them to check for the dishes they have to prepare. System is running on various tablets but the operating environment and purpose of each is different for each user.

3.2.5. DESIGN AND IMPLEMENTATION CONSTRAINTS

3.2.5.1. NETWORK CONSTRAINT

There should be a working internet connection at all times for the time to time updation to happen to all the classes in the application.

3.2.5.2. DEVICE CONSTRAINT

It should be compatible with tablets and should be supported by all the browsers.

3.2.6. 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 resources such as processing speed and memory 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.

3.3. EXTERNAL INTERFACE REQUIREMENTS

3.3.1. USER INTERFACES

The different interfaces that will be developed by this interface are:

3.3.1.1. CUSTOMER INTERFACE (Reservation/ booking)

1. Booking a Table

   The customer enters his/her details (Name, time of dining and the number of people joining) using the application and is notified if a table is not available.

3.3.1.2. CUSTOMER INTERFACE (In the restaurant)

1. Place an order

   a. While placing an order, the customer selects dishes on the web app by clicking on the item they want to order.
   b. They can mention the quantity of the dish by increasing the count of the dish.

2. Feedback

   a. After the user finishes The dining at '*Tomatoz'* he or she will be able to enter the feedback by an automatic feedback form that will be generated after paying the bill.

3.3.1.3. HEAD CHEF INTERFACE

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

3.3.1.4. RESTAURANT MANAGER INTERFACE

Restaurant manager will have a screen where he will get notification whenever an order is completed. System will notify the manager about the order number and table number. Manager also has a screen where all orders are listed. Moreover, he also has an interface screen to see table number and the status of tables in the restaurant as free/available. He will also have a screen which will display the different allocations as well as monitor the kitchen functions.

3.3.1.5. ADMIN MANAGER INTERFACE

As Admin is authorized to perform ***CRUD*** operations on Staff Members, Menu

Items and Inventory Items. He will have three different screens for Staff Members, Menu Items, and Inventory.

3.3.1.6. WAITER INTERFACE

The waiter in the restaurant will be alerted when the status of the food changes, when a food item is prepared and ready to be delivered, the waiter is notified and can then deliver the item to the respective tables. The waiter will also be responsible for manually doing the payment when the customer clicks on the "Request bill" option on the application.

3.3.2. HARDWARE INTERFACES

- processors(i7), and SDD cards that will hold the large scale data that is generated efficiently.
- Tablets kept at every table (along with charging ports/ stations)
- Fast and steady internet/network connection
- Multiple monitors in the kitchen to notify the chef's on new orders and the status of current orders.

3.3.3. SOFTWARE INTERFACES

- For Database services system shall use the latest version of Firebase released on October 16, 2018.
- Appropriate software for authentication that supports federated sign-in.
- Realtime Database in Firebase is a NoSQL database which supports JSON format.

3.3.4. COMMUNICATIONS INTERFACES

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

3.4. SYSTEM FEATURES & FUNCTIONAL REQUIREMENTS

3.4.1. PLACE AN ORDER

3.4.1.1. DESCRIPTION AND PRIORITY

The system will give customers the ability to place their orders using our product. Customer will be able to select multiple dishes and their quantity for their order.

Priority : HIGH

3.4.1.2. STIMULUS/RESPONSE SEQUENCES

When user enters the order activity/page, system displays a list of available dishes.

1. **Stimulus :** The customer taps on the available dish
   Response: System shows the names of the dishes.
   Contains a **select option** for the customer to select the quantity,
   Add to Order for each dish.

   1.1 **Stimulus :** The customer taps on the unavailable dish
   Response: Message "Dish unavailable".

2. **Stimulus:** Customer enters quantity, then presses **Place Order** button.
   Response: Order is placed in a queue displayed to the chef.
3. **Stimulus:** Customer taps on **Confirm Order** button at the bottom.
   Response: Order added to database

3.5. OTHER NON FUNCTIONAL REQUIREMENTS

3.5.1. PERFORMANCE REQUIREMENTS

- The system must be interactive, and the delays involved must be no more than 3 seconds. For every action-response delays are small and lags in responses are not evident.
- As a user views multiple pages of the website scrolling delay and time to access the next web page is no more than 2 seconds. Longer delays can be a detriment to the user experience.
- 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 available with little delay to avoid delivery(food to table) delay.
- On connecting to the server, the delay to make a successful connection should be less for effective real time communication.

3.5.2. 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.

3.5.3. SECURITY REQUIREMENTS

- There is a need for a proper and encrypted login authentication for chef, admin and waiters as employee sensitive information and inventory should be protected.
- Information transmission should be securely transmitted to the server without any changes in information to avoid disturbances in orders and billing.
- Customer information such as name, mobile number is kept confidential and will not be available to any other third party.

3.5.4. SOFTWARE QUALITY ATTRIBUTES

1. **Adaptability**

   Changes in the menu, information stored in the database about employees and inventory, seating details etc. should be incorporated easily.

2. **Availability**

   The system should be up and running during restaurant hours and server should not be down for more than a few minutes to avoid inconvenience for the customers.

3. **Flexibility**

   Software can be modified to change the requirements in the future, if needed.

4. **Interoperability**

   The data is transferred from the customer's end to the kitchen and then orders are assigned to chefs for preparation. This way data is transferred from one part of the system to another. Although, the entire system is connected to the same database and server, different actors/roles have access to different parts of the system.

5. **Maintainability**

   Software should be easily repaired if a fault occurs. Check for bugs and patch and update software is required.

6. **Portability**

   Software should be easily installed on devices and run smoothly as per the requirements specified

7. **Reliability**

   The bill generated by the application must be accurate and the orders placed should be the same as what the user selected. The system as a whole should be reliable. System must give the correct results without any discrepancy.

8. **Robustness**

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

9. **Testability**

   All the requirements are fulfilled, response time is low, and all functions behave as required.

10. **Usability**

    Interface of the software must be easy to use. It should not be complex since non technical people will be using it, for example managers, chefs, customers etc. Since they all have a view, interface should be simple.

3.5.5. BUSINESS RULES

- Manager's interface contains the view of tables that are free, and manager can only view and cannot provide any input to the system.
- Admin has access to perform add, delete, update operations on the database for menu, employees and no other person can modify the data in the database.
- Customers can place orders from the menu items and can update order and pay bills.
- Chefs are assigned orders and can update the queues and has an additional functionality of load balance.
- Chefs can only view the orders and cannot remove an order from their queue. Only head chef can interact with the queues containing orders.

3.6. APPENDIX: GLOSSARY

CRUD: In computer programming, Create, Read, Update and delete are the 4 basic functionalities of persistent storage.

POS**:** point of sale or point of purchase is the time and place where a retail transaction is completed

4. Design and Architecture

4.1 INTRODUCTION

The system we want to create is a Restaurant Management Application, *'Tomatoz'*. It would allow customers to perform functions like reserving a table and placing an order on the app, and separate interfaces for waiter, restaurant manager, chef, and administrator.

4.1.1. SYSTEM OVERVIEW

Module	**Description**
Browse Menu	Customers can browse the menu.
Reserve Table	Customers can reserve table on providing table size and time of arrival.
Place/Receive Order	Customers/Waiters can select items and build an order
Payment Form	Checks out to a third party payment form and once the payment is successful, the database is updated

4.1.2. DESIGN CONSIDERATIONS

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

- Simple menu setup
- User-friendly order management
- Good technical support
- Employee management

4.1.2.1 ASSUMPTIONS, DEPENDENCIES AND CONSTRAINTS

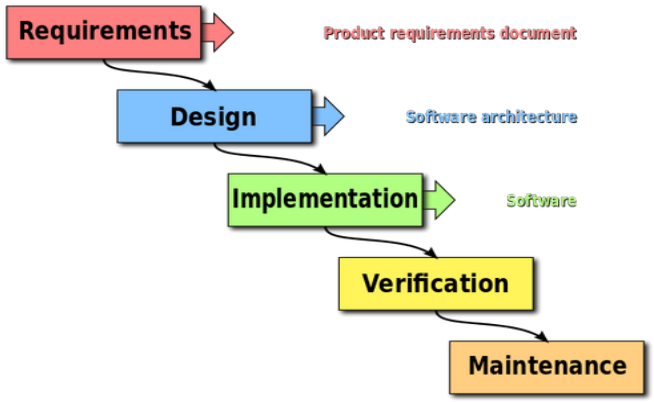
Covered under Project Planning, section 3.2.

4.1.2.2 GOALS AND GUIDELINES

Guidelines for the software
1. Emphasis on feel of UI and it's responsiveness.
2. Transaction consistency of requests is a must since it's a management system
3. Intuitive interface to help users navigate effortlessly and not migrate to another product.
4. It should be a real time application.

4.2 DEVELOPMENT METHODS

We are using the Agile with SCRUM approach. This is used as the team will be divided into sub teams that will handle a specific part of the project and changes in the requirements. Updates will be discussed at regular intervals of time.

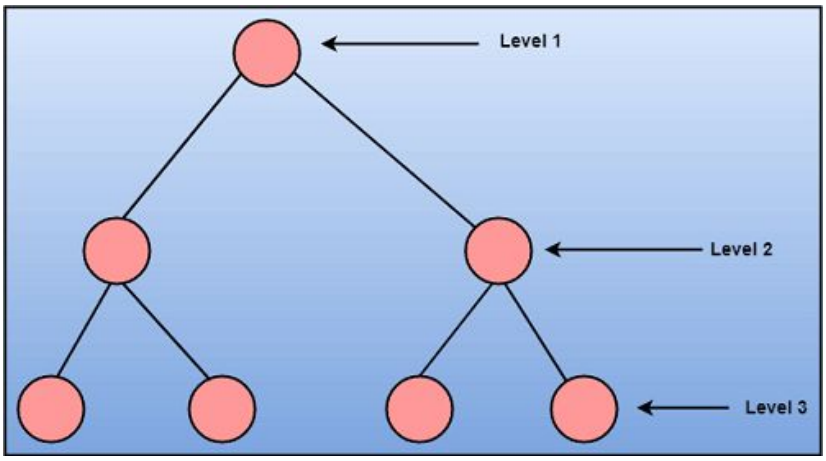


4.2.1 ARCHITECTURAL STRATEGIES

To implement software design, we have used a function oriented design to help define the respective tasks and to assist with the designing process. It may not be required every time, but it is immensely helpful in getting expected results and in mitigating the risks of any course of action. We were able to design a proper plan of software development and ensure that all the demands, requirements and requests of the clients are taken care of.

Function-Oriented Design: The decomposition centres on identifying the major software function and then elaborating and refining them in a top-down manner. The system is comprised of many smaller sub-systems known as function. These functions are capable of performing significant tasks in the system. Also, in this, the system is considered as top view of all functions. Furthermore, function oriented design inherits some qualities and properties of Structured Design, where divide and conquer methodology is used.

A good system design strategy is to organize the program modules in such a method that are easy to develop and latter too, change. Structured design methods help us deal with the size and complexity of programs. To design our system, we used a top-down approach where in we start with the identification of the main components and then decomposing them into their more detailed sub-components.



4.2.2 SYSTEM ARCHITECTURE

4.2.3 SUBSYSTEM ARCHITECTURE

- For Database services system shall use to Firebase latest version released on October 16, 2018.
- Proper software for authentication purposes. Also supports federated sign-in.
- Realtime Database in Firebase is a NoSQL database which supports JSON format.
- Data is centralised to ensure consistency
- Error detection and recovery is done using exception routines, in terms of failure in database and returning the appropriate error code. Data format is also checked in the client and server side whenever there is transfer.

- Concurrency and synchronisation is delegated to the database as operations are atomic

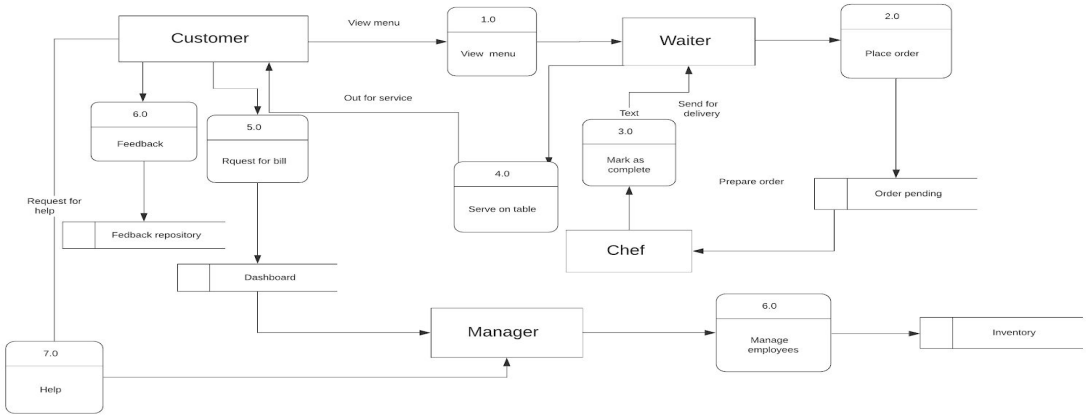
4.3 POLICIES AND TACTICS

- Programming Methodology
  - ○ All variables and functions will be named using camel casing notation.
  - ○ Code will not be repeated multiple times, i.e, functions will be used appropriately. Also, functions will not be created unnecessarily.
- Languages Used
  - ○ Frontend - HTML, CSS, Javascript.
  - ○ Backend - Firebase Web App
- Database Used: Firebase Realtime Database
- Testing : We will have unit tests.
- Maintainability : We will split the web application into different features so that maintaining and updating a particular feature of the application is made easier.

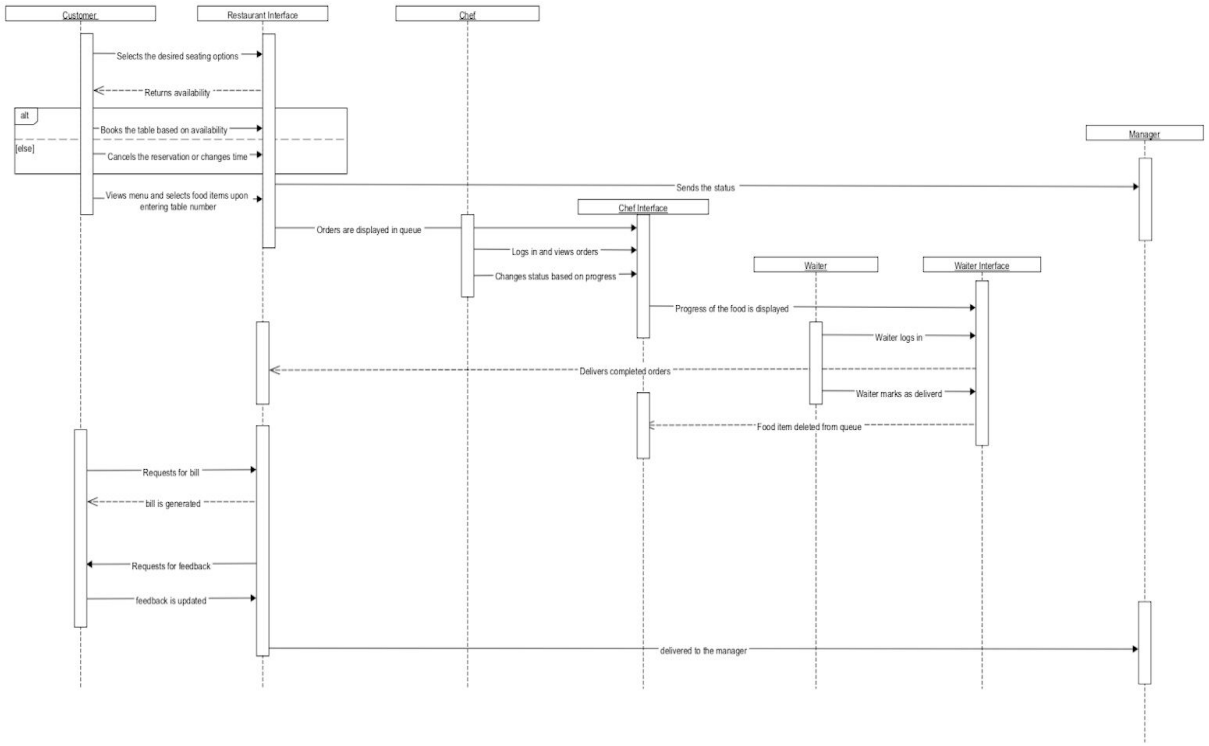
4.4 DETAILED SYSTEM DESIGN

4.4.1 CLASSIFICATION

DATA FLOW DIAGRAM



SEQUENCE DIAGRAM



It depicts all the activities that will take place in sequence while running our application.

4.5 RESPONSIBILITIES

Covered under Requirements Specification, section 2.3.

4.6 CONSTRAINTS

Time is considered the most important constraint in developing the *'Tomatoz'* - Online Restaurant Management System. Because of the limited development time, the document submissions must not stray too far from the stated deadlines. Progress must be maintained at a consistent pace to deliver the required deliverables by the target dates. Availability of hardware and software, too, would be a constraint.
Apart from these, some general constraints are:

a. It must be coded efficiently enough to run well on the provided server hardware.
b. Client-side code and/or web pages must be able to run efficiently
c. The database will be created and maintained in a way that makes it reasonable and of manageable size.

d. Must write secure code and the system should be capable to handle any security threats.
e. Acceptance Criteria

4.6.1 RESOURCES

1. Database components
    - This resource takes care of the race condition by itself since operations are atomic.
    - We are using the firebase database for taking care of the different constraints
2. Server component

    The consideration in this resource is memory as it needs to be able to handle multiple concurrent requests.
3. Client component

    The core consideration in this component is to minimise the load on the client which means that the front end code needs to minified.

4.6.2 PROCESSING

1. Table availability for reservation:
    a. To check whether a table is available or not, a database table consisting of present tables and status (available, reserved, unavailable).
2. Placing an order:
    a. The customers select dishes of their choice and confirm their order
3. Payment:
    a. For restaurant goers: customers request for bill once they are done. The bill is printed upon approval by a waiter**.**

4.6.3 USER INTERFACE

The user has two ways of placing orders in '*Tomatoz'*:
1. Ordering Online
2. Walking into the restaurant

I. Visitors**:** (At home /online)
    A. View menu- Can access the menu on their mobile app from anywhere.

B. About '*Tomatoz'*- Can get to know details such as inspiration of *'Tomatoz'*, cuisine, address, specialities, etc.
C. Check seat availability- Can enter their preferred time slot and get the table availability
D. Reserve table
    1. Provide details- Name, contact number, email-id, no. of seats, time slot in case of table availability

II. Walk-in customers
A. Table availability is checked by the front desk in-charge
    1. Table number, status (Reserved/ available)
B. Select table number and mark reserved once allocated

III. Tabs on every table (customers)
A. Table number
B. View menu
C. Place an order
D. Request for bill
E. Help
F. Feedback (optional)- contains feedback and rating form

IV. Kitchen staff
A. Maintains a list queue having
    1. Table number
    2. Dishes in order of preparation and service- assign to respective chefs. Assigning of chefs will not be a part of the system. Happens internally
    3. Status of order- In progress/ completed/ delivered

V. Waiter
A. Get assigned to their respective table numbers
B. Copy of the kitchen staff list along with the bill
C. They can see the meal progress of the table assigned to him
D. When the chef updates order status to "completed", it is delivered to the table by the waiter
E. Mark the order as delivered once successfully delivered.

VI. Manager
A. Oversees functionalities II, IV, V along with the bill

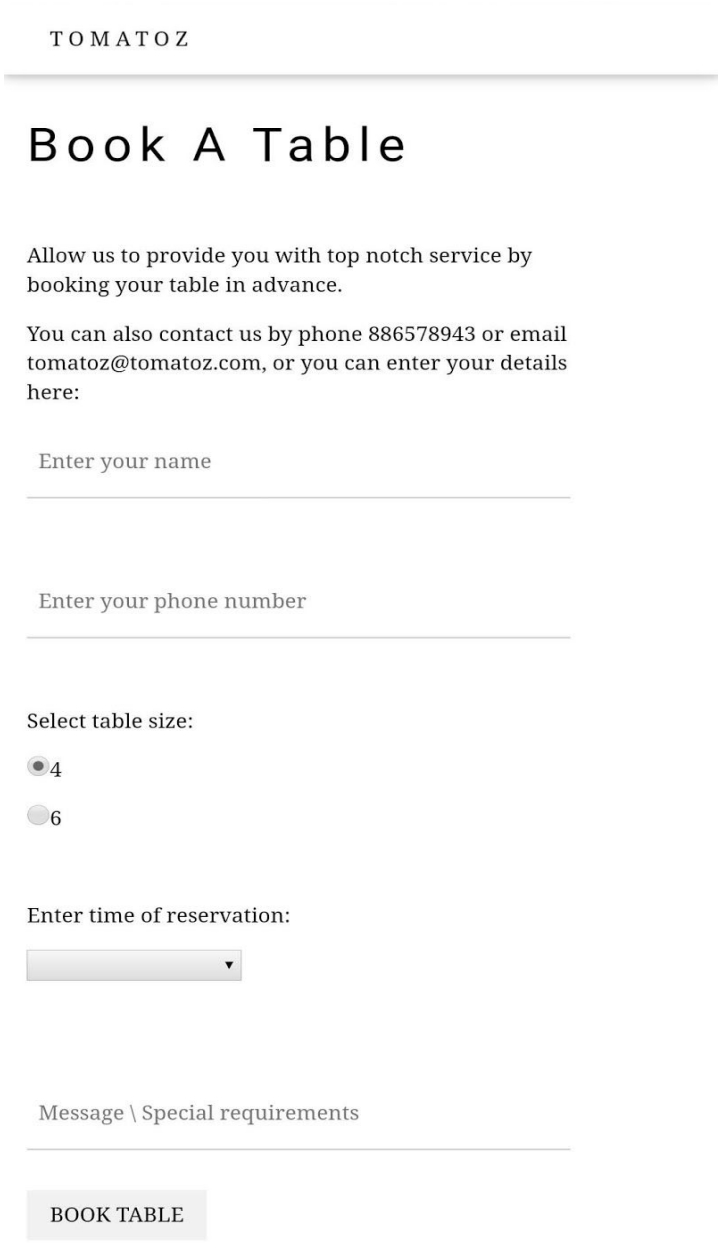
4.6.4 GLOSSARY

CRUD: In computer programming, Create, Read, Update and delete are the 4 basic functionalities of persistent storage.

5. IMPLEMENTATION

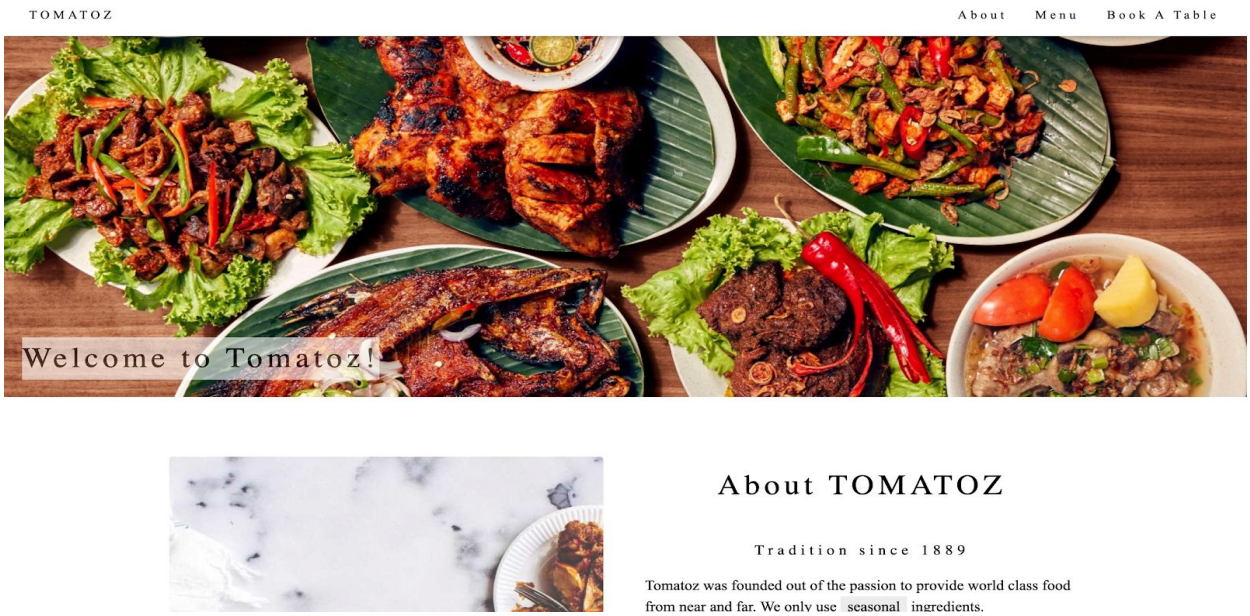
We have carried out our implementation based on the user profiles using it:

1. Customer booking a table outside the restaurant



The above screenshot is for the view on a mobile phone.

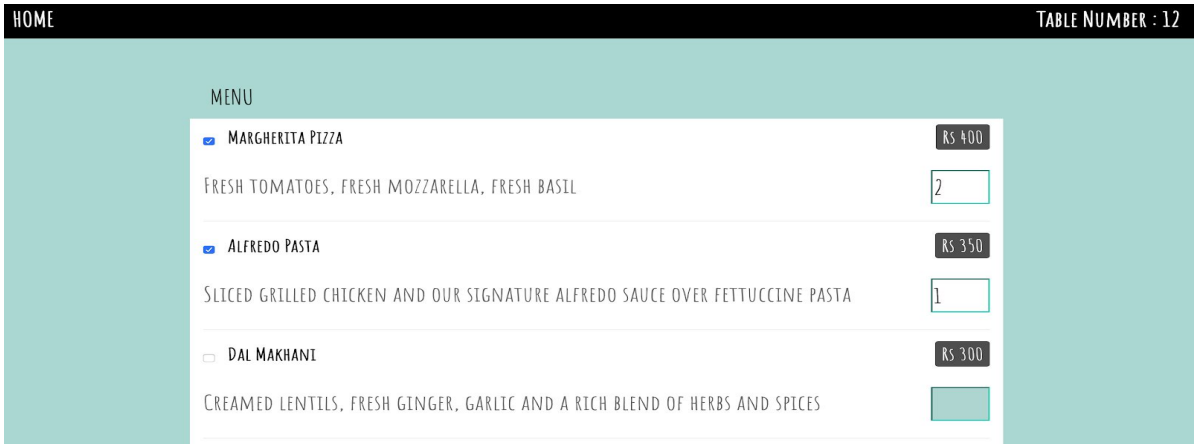
About the restaurant



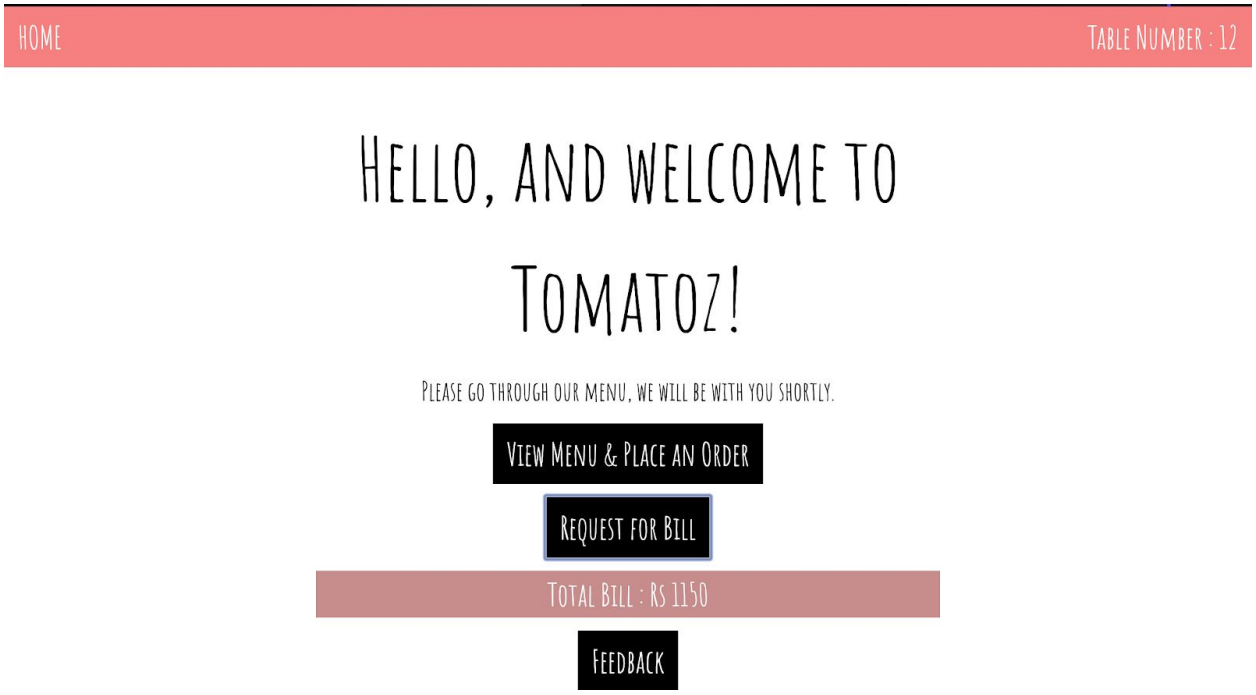
1**. View and select the menu**

The menu is displayed on the index page with the description of each food item.

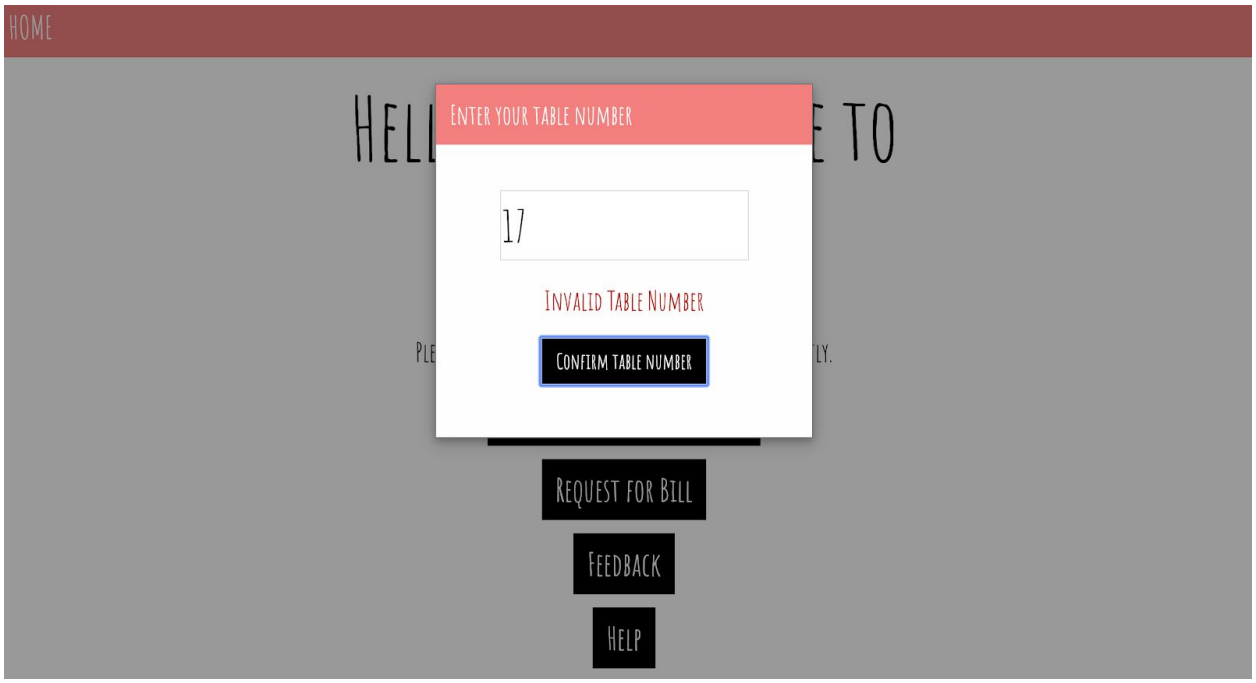
Once the customer enters the restaurant, he can view the menu and select the various food items that he/she desires. On clicking place order of the items selected the order is directly placed to the chef in the kitchen.



2. Displays the bill of the items ordered



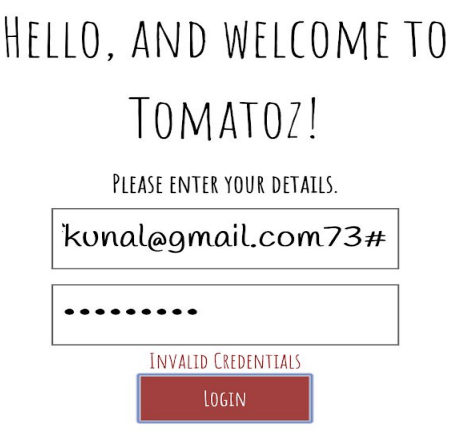
Checks for a valid table number while entering the order else throws an invalid table number error



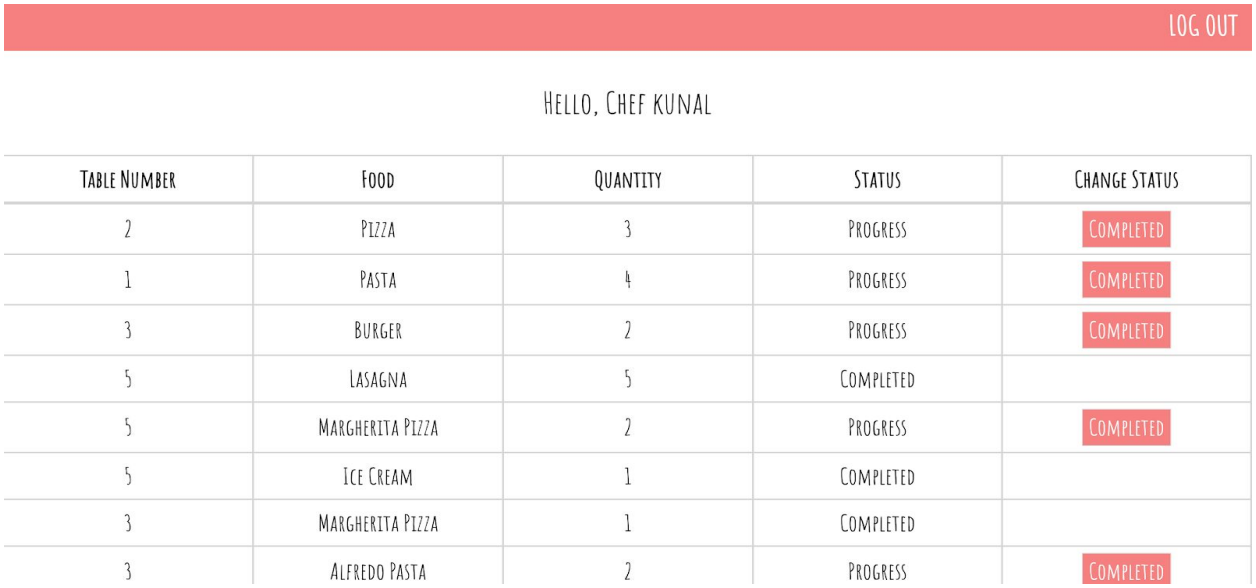
The feedback feature is implemented statically

- **CHEF VIEW**

The chef can login only with his valid credentials and all the invalid credentials will be displayed with an “invalid credentials” message.

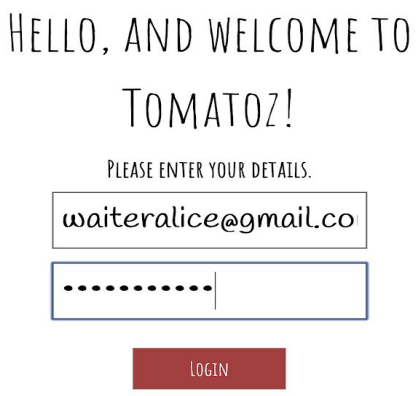


The chef logs into his/her account to check for the list of orders that has to be prepared. All the orders that have been prepared, can be updated by clicking the "completed" button. The orders appear as and when they are placed in a queue format.



- **WAITER VIEW**

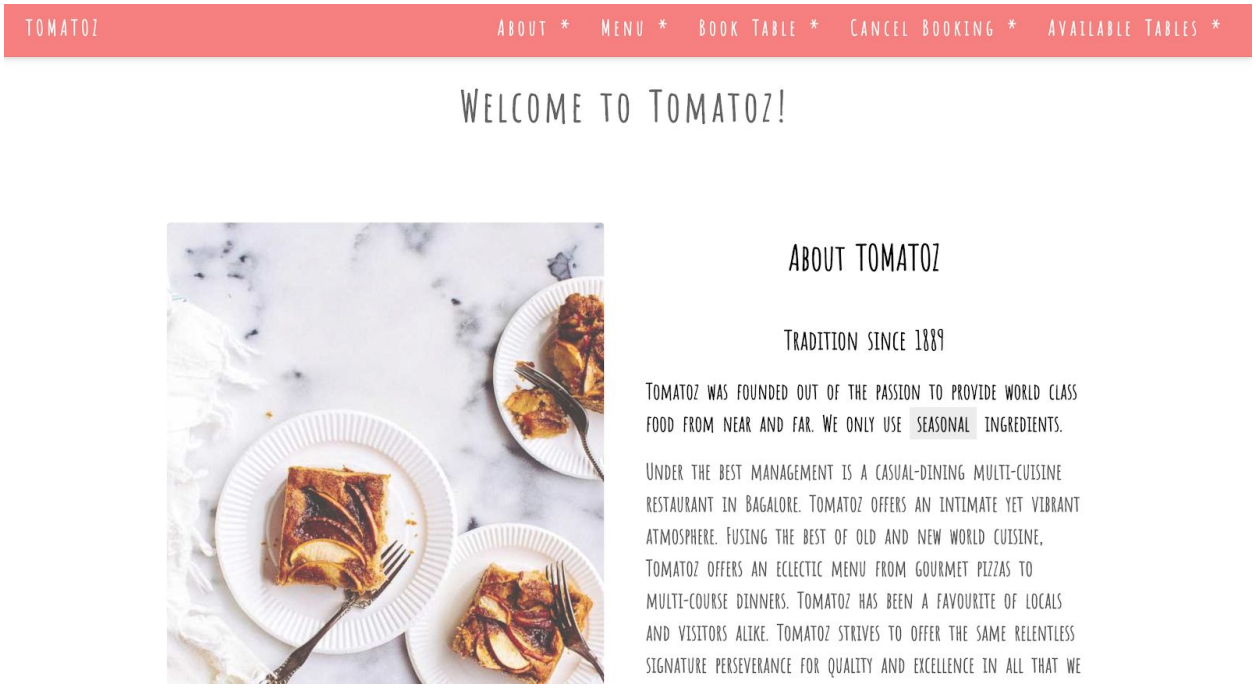
  The waiter has to login to the waiter login page and it displays invalid if the wrong credentials are entered. As and when the chef clicks on complete the waiter can view the order that has to be delivered to the respective tables based on the table number.



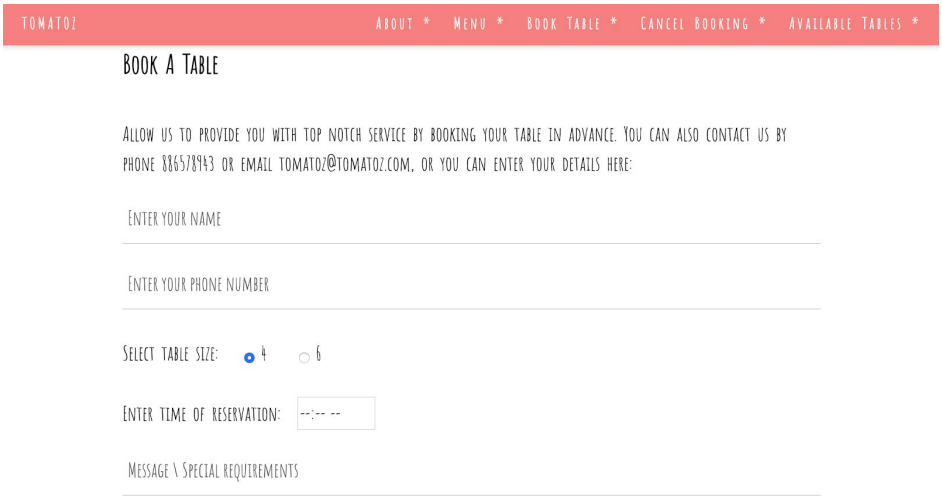
Orders that are completed in the kitchen can be viewed by the waiter.

AFTER REVIEW

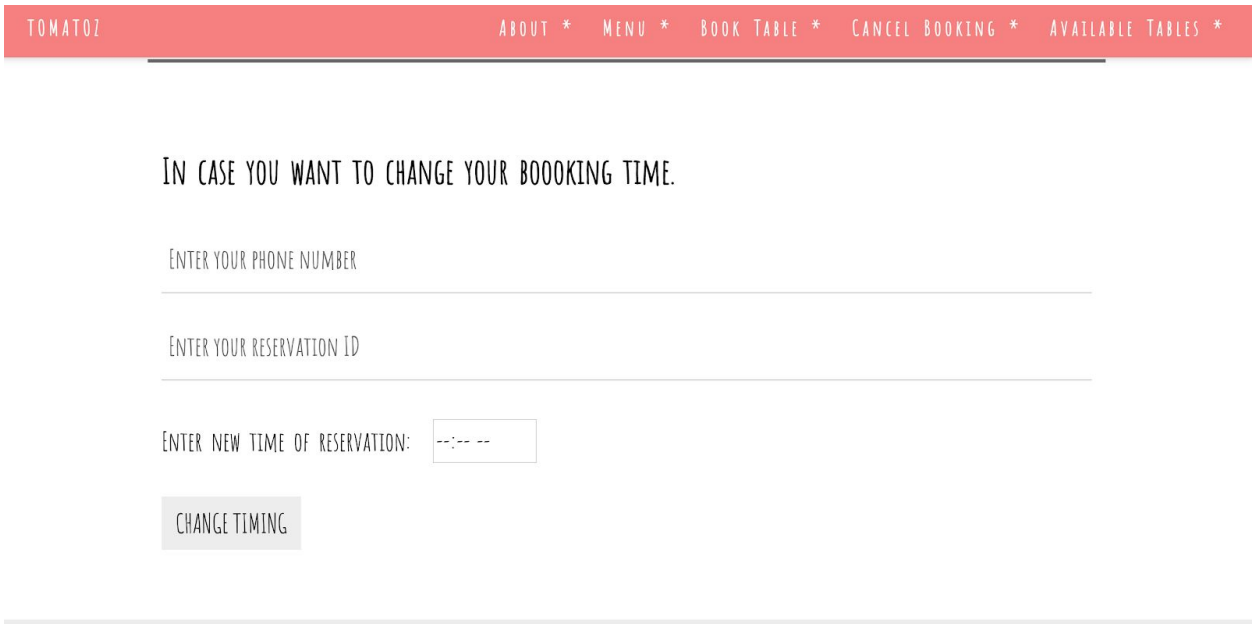
1. Changed UI of the about page so that it syncs with the subsequent pages.



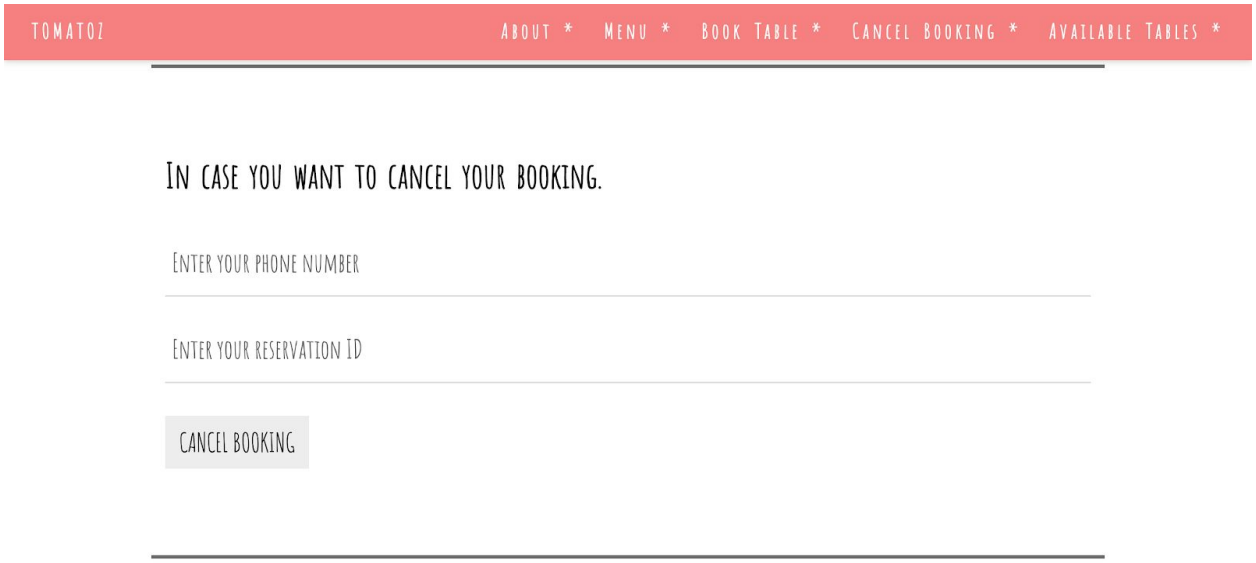
2. Book a table
    a. The UI has been changed and the additional features have been added. We have made changes in the display of the page, added cancel booking and booking time feature is added



Change in Booking time feature added

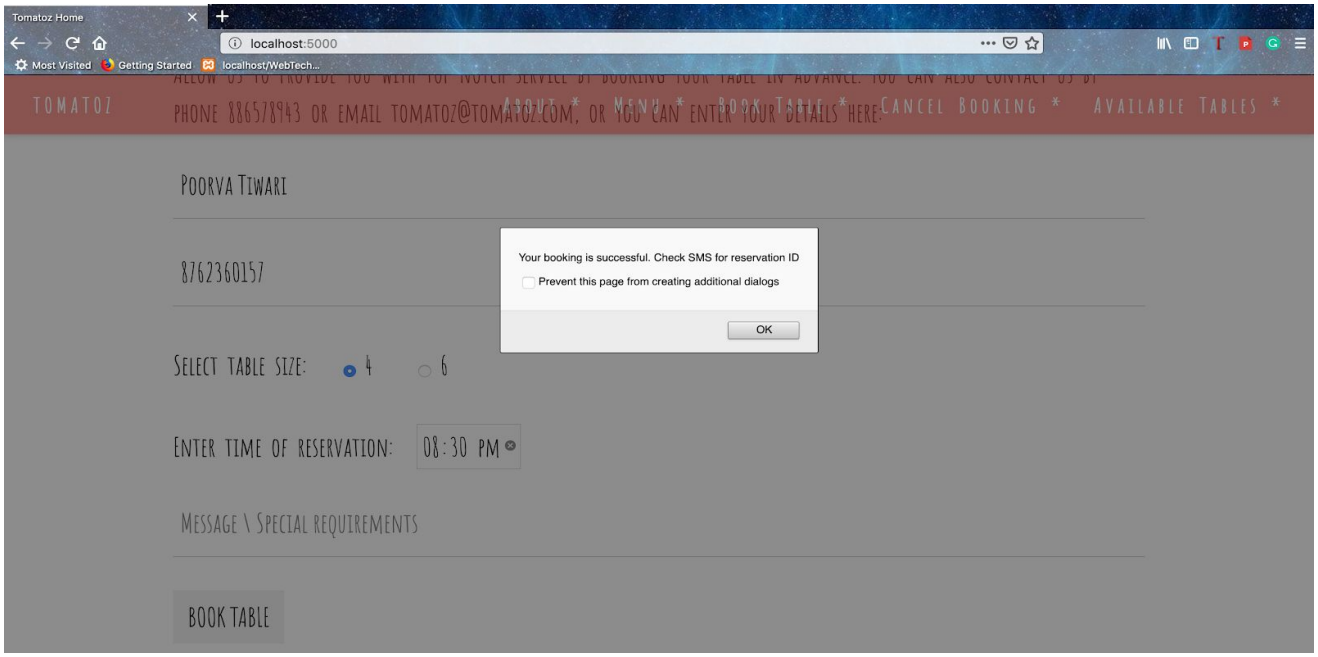


Cancel table feature added

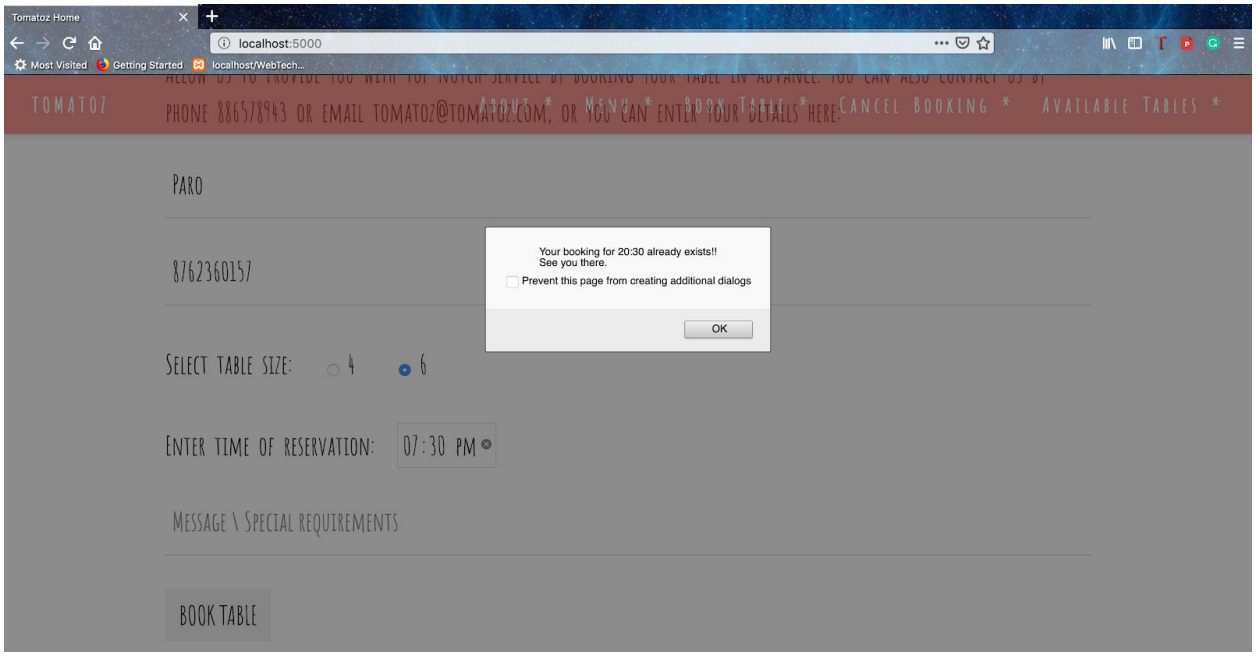


- Cannot cancel if a booking does not exist

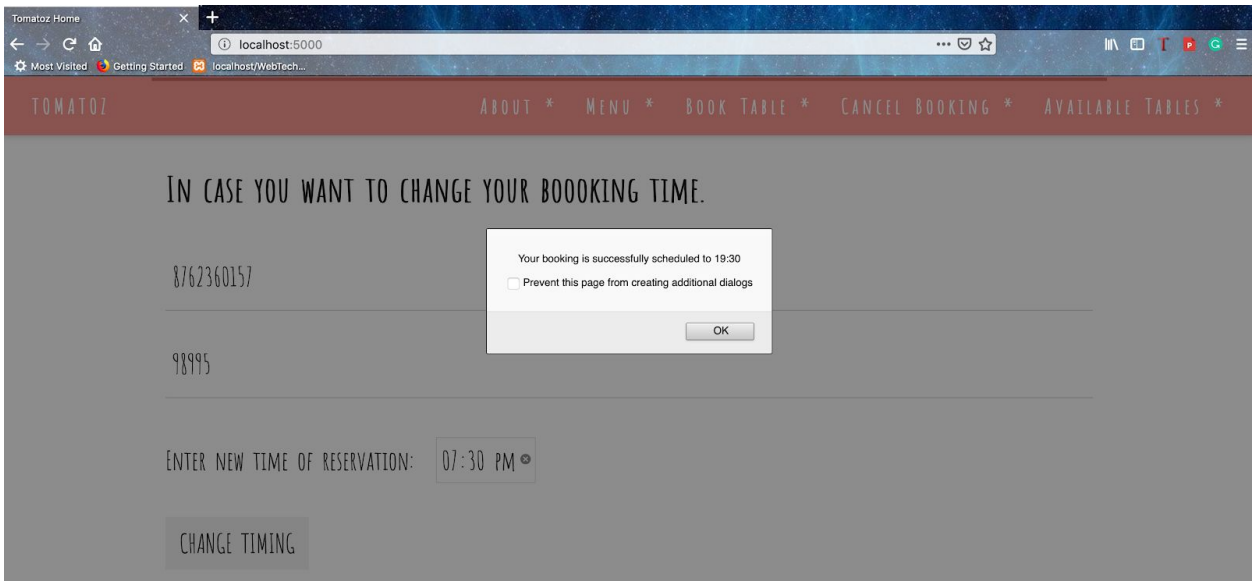
A message is displayed if the booking is successful

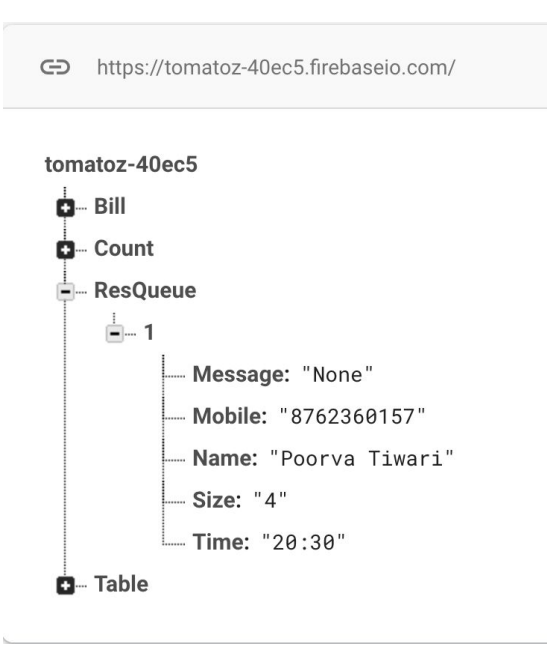


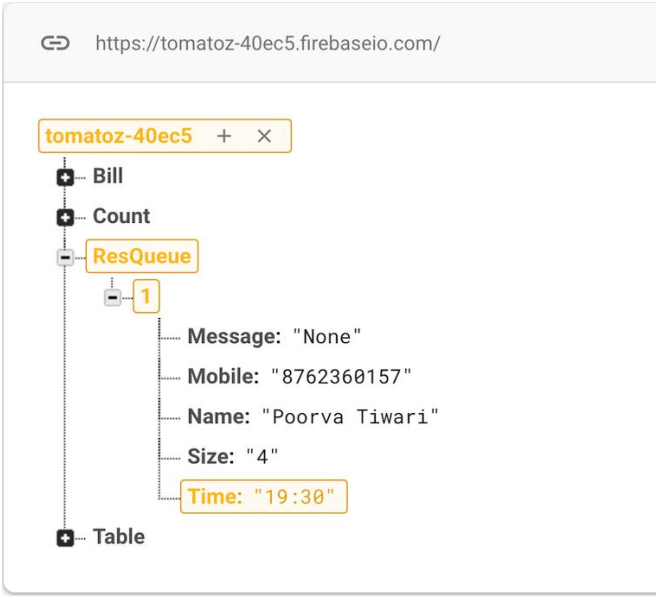
Avoids repetitive booking if the booking already exists

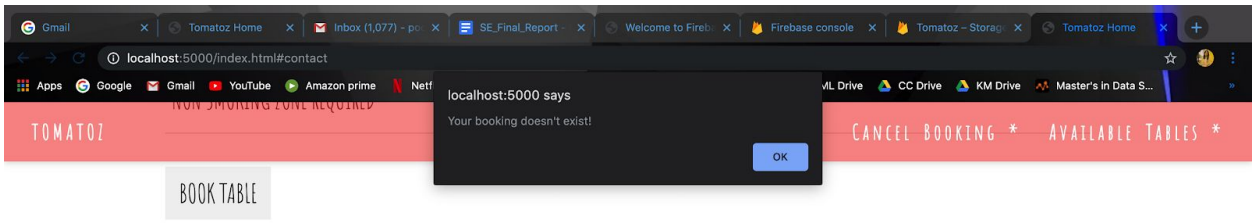


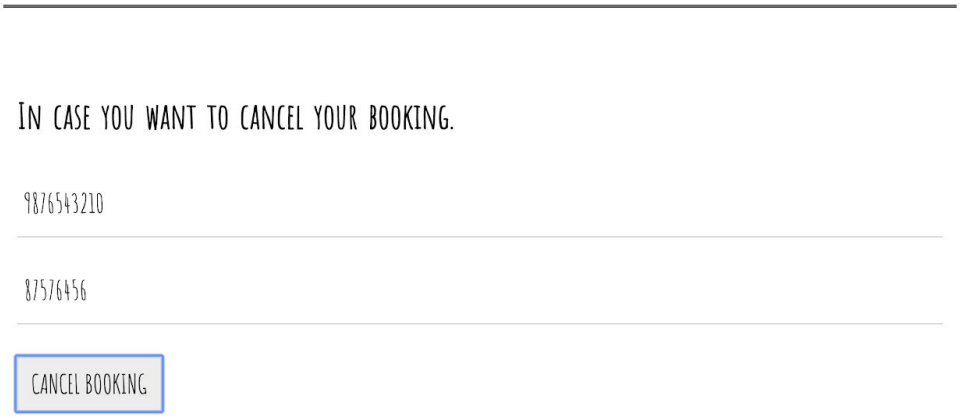
Changes the reservation timing based on the mobile number and the reservation ID. It changes the booking time successfully and alerts if the booking does not exist.





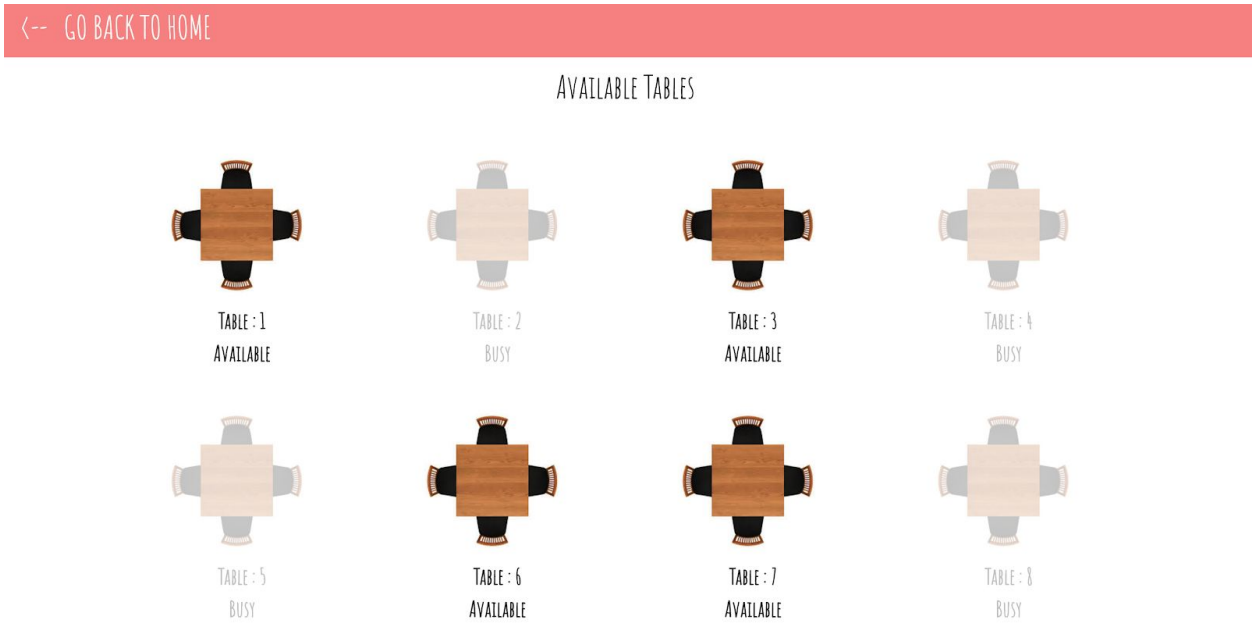






The Feedback feature is added and is checked by the manager of the restaurant

The tables that are available are displayed for the booking purposes of the custome



- **MANAGER VIEW**

The manager can view all the order details that are completed by the chef & waiter by logging into his/her account. He/ she can also change the menu based on the stock of ingredients available.

HELLO, JACK. THESE ARE THE CURRENT ORDERS

TABLE NUMBER	ORDER NUMBER	STATUS	CHEF	WAITER
1	12	DELIVERED	POORVA	ALICE
2	12	DELIVERED	POORVA	ALICE
3	4	COMPLETED	POORVA	ALICE
4	4	COMPLETED	POORVA	CHARLES
5	4	COMPLETED	PRERNA	ALICE
6	3	PROGRESS		
7	3	PROGRESS		
8	4	PROGRESS		

6. TESTING

In our project the test technique that we have used is the White Box testing technique where we have made sure that the following steps in it are carried out:

1. Checking the internal security loopholes
2. Making sure that there are no poorly structured paths in the coding process
3. Making sure that there is proper flow of the specific inputs in the code
4. The expected output is generated at every phase
5. Peer testing of each individual statement, object and the function on an individual basis.

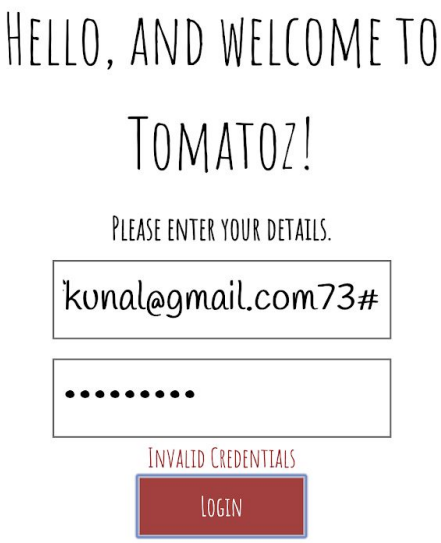
- CODE REVIEW

  It is the software quality assurance activity in which one or several persons check the program by viewing and reading the parts of the source code and it is generally done after the implementation phase.

  In our project we have done code review in a combination of ways:
  1. Ensuring better code quality: we have implemented this by making sure that code can be easily maintained. This is done by maintaining uniformity in terms of writing the code and making sure that it is well commented so that it can be helpful for the other developers and future enhancements.
  2. Finding defects : During the testing phase we encountered various loopholes in terms of security and made sure that it is implemented properly across the application
     - We have ensured authentication via firebase and input validation for the login details entered so no SQL injections can be performed.
     - Input validation ensure that no special characters can be added in the email ID section

- BORDER CASES TESTED
  - To check if the table number is passed every time an order is placed so that this acts as a primary key in the chef and the water functionality. The table number has to be unique and can be passed multiple times as and when the order from that table is generated.

- Password and Email ID authentication, has to be unique and no special characters can be added in the Email ID section
- Checking for the maximum and the minimum orders that are pending for a table
- Each order that is placed by a table is taken into account as a unique request
- There can be only 15 tables that can be



- UNIT TESTING

  It is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. It usually has one or a few inputs and usually a single output. Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

  In our project we have done unit testing in a phase wise fashion, i.e as and when we have developed each functionality we have checked it against the model inputs to generate the desired results. Refer to the screenshots in implementation section.

- INTEGRATION TESTING

  Integration testing is the process of testing the interface between two software units or modules. It helps determining the correctness of the interface. The purpose of integration

testing is to expose faults in the interaction between integrated units. It is performed once all modules have been tested.

- Our application is fully integrated with the firebase(backend as a service) real time database and authentication.

- SYSTEM TESTING

System testing is a type of software testing that is performed on a complete, integrated system to evaluate the compliance of the system with the corresponding requirements.

In this, software is tested such that it works fine for different operating system. It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working.

In this we have security testing, recovery testing , stress testing and performance testing

Our application is compatible with various browsers such as Chrome, Firefox, Safari and Operating Systems such as Windows, Linux, MacOS and Android Mobile devices.

REFERENCES

1. https://medium.com/@hasangi/writing-deleting-and-updating-data-in-firebase-realtime-database-with-javascript-f26113ec8c93
2. https://firebase.google.com/docs/auth/web/manage-users
3. https://firebase.google.com/docs/reference/js/firebase.database.DataSnapshot
4. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
5. https://medium.com/swlh/using-firebase-to-provide-real-time-notifications-2f86b2f7d499

CONCLUSION

We have thus implemented the Online Restaurant Management System in an iterative and incremental software development process.The top priority functionalities have been successfully implemented.