Restaurant Booking System

Contents

[Analysis 3](#_Toc24045659)

[Background 3](#_Toc24045660)

[Identification of problem 3](#_Toc24045661)

[Description of current system 4](#_Toc24045662)

[Summary 5](#_Toc24045663)

[Identification of end users 5](#_Toc24045664)

[Identification of end user’s requirements 5](#_Toc24045665)

[Project Scope 6](#_Toc24045666)

[Analysis of similar systems 7](#_Toc24045667)

[tablein 7](#_Toc24045668)

[Restaurant Live 11](#_Toc24045669)

[Comparison between similar systems 16](#_Toc24045670)

[Questionnaire 17](#_Toc24045671)

[Working in a restaurant 18](#_Toc24045672)

[Ordering and reservations 20](#_Toc24045673)

[Summary 22](#_Toc24045674)

[Data Flow Diagram 23](#_Toc24045675)

[Data sources and destinations 24](#_Toc24045676)

[Data volume 24](#_Toc24045677)

[Limitations 24](#_Toc24045678)

[Final objectives 25](#_Toc24045679)

[Start / ordering 25](#_Toc24045680)

[Login 25](#_Toc24045681)

[Register 26](#_Toc24045682)

[Main 26](#_Toc24045683)

[Proposed solution 27](#_Toc24045684)

[Design 28](#_Toc24045685)

[Overall System Design 28](#_Toc24045686)

[Start Page 29](#_Toc24045687)

[Searching up for a restaurant 30](#_Toc24045688)

[Reserving 31](#_Toc24045689)

[Ordering 32](#_Toc24045690)

[Login 33](#_Toc24045691)

[Register 34](#_Toc24045692)

[Modular Design 35](#_Toc24045693)

[IPSO table 35](#_Toc24045694)

[Validation 36](#_Toc24045695)

[Entity Relationship Diagram 37](#_Toc24045696)

[Entity Attribute Model 38](#_Toc24045697)

[Normalisation 38](#_Toc24045698)

[SQL Statements 39](#_Toc24045699)

[Data Dictionaries 42](#_Toc24045700)

[OOP Classes 43](#_Toc24045701)

[Data structures 43](#_Toc24045702)

[Algorithms 43](#_Toc24045703)

[Interface Design 43](#_Toc24045704)

[References 44](#_Toc24045705)

# Analysis

## Background

I am going to be creating a general restaurant booking system by implementing the system in a web app. Booking systems have been around for a long time and companies have been taking advantage of booking systems as it helps them to increase their availability and over the years, more customers prefer to book online. Also, because the data is saved online, it saves people time and managing data is easier.

The system will be for restaurants who would like their clients to book tables online, which will help them to be efficient with time by managing their workload when they are dealing with customers at busy times. The main use of a restaurant booking system is to manage the bookings of clients who want to dine at the restaurant but with features and tools that will support the user to do this in an effective way. The main features that a restaurant booking system should have is a calendar to help staff organise bookings with a database in the back end, and instructions to assist the user with the booking.

## Identification of problem

During my time working at a restaurant as a waitress, I have encountered a lot of problems due to time management and meeting the customer’s requirements. The restaurant has two floors, a device to manage the takeaway orders and a device with software to manage tables in real time. From my experience, the main problem is the amount of work that is dealt with during peak times. I would have to manage my time by taking the customer’s orders, manage and reserve tables, serve food and deal with the bills at a fast pace. Doing tasks at a fast pace might get all the tasks done but the tasks might not be done efficiently.

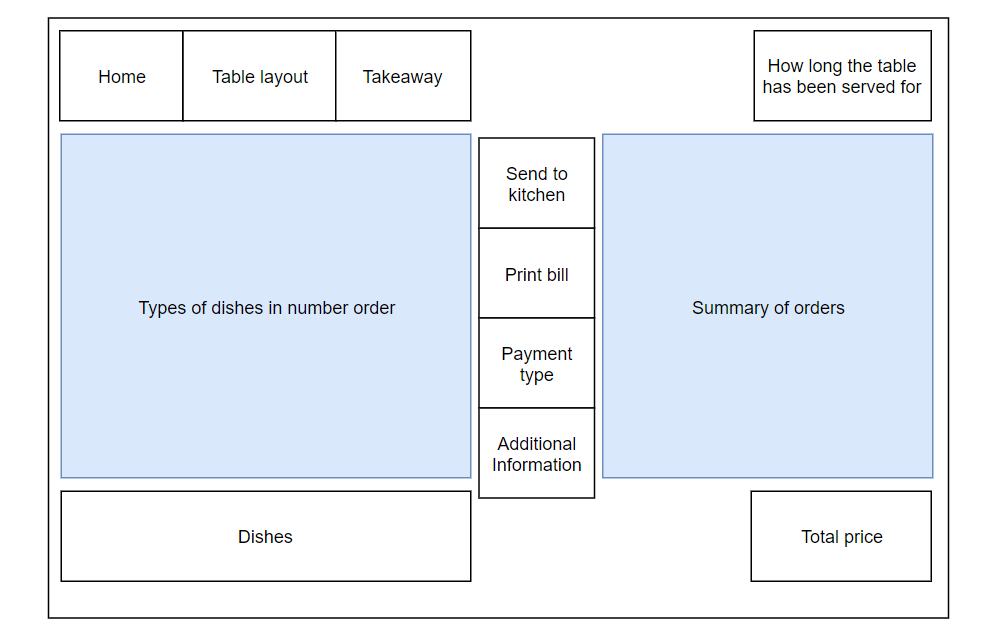
Also, for a customer to reserve a table at the restaurant, they must phone the restaurant and a member of staff would ask the customer for their name, date and time of reservation and the number of people that will be dining. Most customers would reserve tables before peak times, which would be 6 to 8 in the evening. The waiter would then put the booking down on paper. However, the problem with this is that the staff’s workload can increase because most likely, they would be dealing with the customers in the restaurant, serving food or dealing with takeaway orders and this will cause difficulties, especially in busy times. From this experience, it would be better if the customer can reserve tables online and outside of opening hours. Therefore, implementing a general restaurant booking system would be useful in this case.

The software used at the restaurant was effective in taking the customer’s orders, but it was difficult to do a lot of tasks in a reasonable amount of time because there was only one machine available to use and the communication between the kitchen and the waiting staff was slow. However, it was effective in taking orders because of the layout in the software. Since there is only one machine available, the proposed solution should allow the staff to use the system on any device. If this is implemented, the staff can manage their workload on any device and the chances of customers waiting for their food or the bill would be low.

## Description of current system

Firstly, staff would choose the table that they are managing from the table layout. The table layout reflects the layout of the restaurant and each table is coded by numbers. This would then take the staff to the main page. In the corner, there is information on how long the table has been served for, which can give the staff an idea of what action they should take next. On the left side, there is a section where the dishes were organised by their numbers from the menu. This shows that the software was tailored for the restaurant itself, which is an advantage because it suits the needs of the restaurant. Since I am implementing a general restaurant booking system, the features should suit the needs of any restaurant. On the right side, there is a summary of the orders that have been placed and the total price of the orders. Notes can also be taken for each table. The notes could include dietary requirements or any additional information. For the bills, the staff would choose the ‘print bill’ option to print the bill.

In terms of the communication between the kitchen and the waiters, it was hard for the staff in the kitchen to know which dishes the customers would want first and if they decided to have no starters. From this problem, the proposed solution should support the communication between the kitchen and the staff. For every order of food, there should be a time stamp to help the kitchen know when the order was placed.

For takeaway orders, the restaurant would use a device that will automatically receive orders from customers that have ordered on a mobile app. The order will include a summary of the orders, the total price and any other additional notes. For every takeaway order, there is a unique reference which helps the staff to manage orders.

### Summary

|  |  |  |
| --- | --- | --- |
| Positives | Problems | Improvements |
| Table layout of the restaurant is clear | One machine available | Staff should be able to manage reservations on any device |
| Tables are determined by numbers | Difficult for kitchen to know what dishes customers would want first | Timestamps for every order for kitchen to know which order should be first |
| Managing each table is simple | Difficult for kitchen to know when to serve main dishes | Customers should be able to reserve tables online |
| Taking notes for each order and table is easy | Customer can only reserve tables by phone | Notifications for staff to help manage their workload |
| Dishes were organised by their numbers from menu | Staff takes orders on paper | Main dishes and starters should be separated |
| Takeaway orders have a unique reference |  |  |

## Identification of end users

The end user could be the manager of the restaurant or the staff working for the restaurant or the client that wants to book at the restaurant. Every staff member will have their own account and each one will have different access permissions implemented in place. The customer will not have to create an account to reserve tables. For each end user, the inputs and outputs will be different. For the customer, the main inputs should be their name, the number of people that are dining, the date of reservation and their contact information. For the staff, the inputs should be the table that they are managing, and the dishes ordered, and the outputs should be everything that the customer has inputted. I will be testing the booking system on family and friends and those who work in a restaurant. Therefore, the system should be user-friendly because some people will have limited IT knowledge.

## Identification of end user’s requirements

Firstly, in order to propose a solution and the final objectives, I will analyse restaurant booking systems that are for offer online. For each system, I would need to extract the different features and the tools that are available for the users to use and decide whether their use is significant or not. Analysing these features will help me to decide whether the system can be more efficient since there are some existing features that have been implemented already to make this happen. After analysing similar systems, I will then conduct a questionnaire to analyse people’s opinions and ideas on restaurant booking systems and restaurants in general. This will be useful for me since I will be creating a general restaurant booking system for multiple users and it will help me to clarify which features should be implemented at least. Finally, after analysing the questionnaire results, I will decide the limitations and final objectives of the proposed solution. I will model the new system in the analysis that will inform the design stage later in the project.

## Project Scope

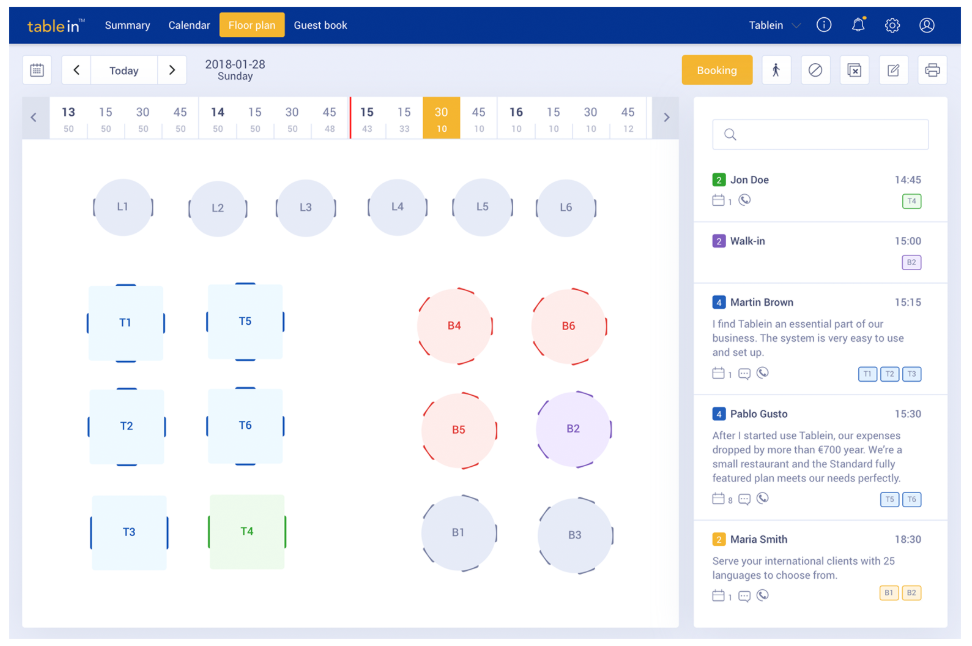
This will give me a rough outline of what I need to do and how long I would have to complete each section for this project.

|  |  |  |
| --- | --- | --- |
| Section | Tasks | Due Date |
| Analysis | Background |  |
|  | Identification of problem |  |
|  | Description of current system |  |
|  | Identification of end users |  |
|  | Analysis of similar systems |  |
|  | Questionnaire |  |
|  | Proposed solution |  |
|  | Limitations |  |
|  | Objectives |  |
|  |  | October 2019 |
| Design | Prototyping |  |
|  | Data Flow Diagrams |  |
|  | IPSO tables |  |
|  | Data dictionaries |  |
|  | ERD and EAM |  |
|  | SQL statements |  |
|  | Normalisation |  |
|  | Interface design |  |
|  | OOP Class design |  |
|  | Algorithms |  |
|  | Top Down diagram |  |
|  |  | November 2019 |
| Implementation | Login Screen |  |
|  | Database for bookings |  |
|  | User interface |  |
|  |  | December 2019 |
| Technical Solution | CREATE TABLE statements |  |
|  | OOP classes |  |
|  | Screenshots with annotations |  |
|  |  | January 2020 |
| Testing | Testing strategies and plans |  |
|  | Screenshots |  |
|  | Improvements |  |
|  |  | February 2020 |
| Evaluation | Evaluate performance |  |
|  | Feedback on objectives |  |
|  | Improvements |  |
|  |  | February 2020 |

## Analysis of similar systems

### tablein

‘tablein’ is a booking system that advertises a restaurant management system through a website. The system aims to help small restaurants to maximise guest capacity and to allow their customers to make more reservations online to save time. Managing small restaurants are easy because of their size. However, I will try to implement features that are useful for restaurants of all sizes. The website promotes a range of features and shows examples to show users what they would be expecting.



Date and time information

Notifications with timestamp and table names

Seating layout

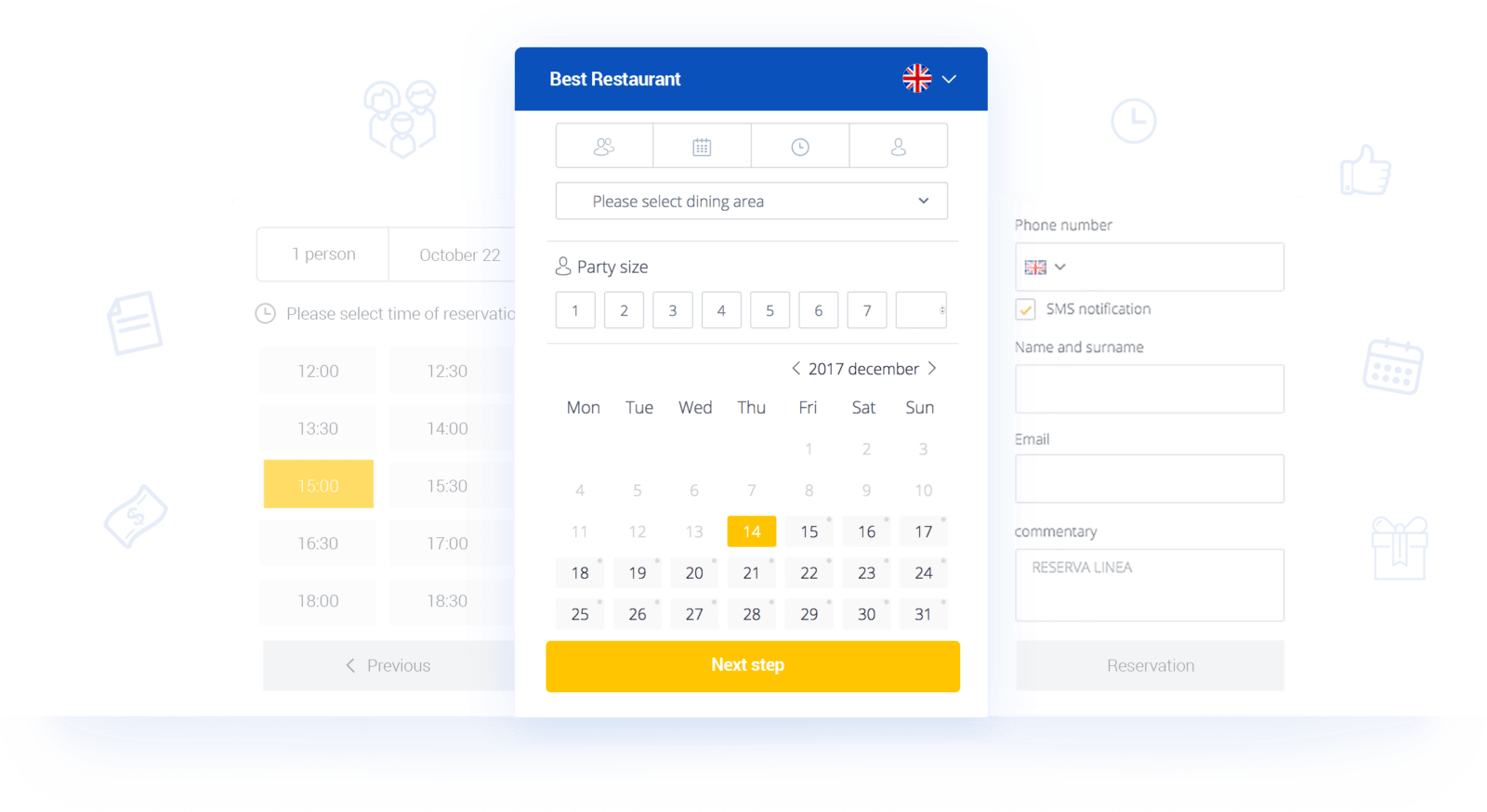
From this example of the floor plan, there is basic information provided such as the date and time from the calendar at the top and the seating layout. The section for the seating layout is large and clear for the staff to know where to attend to the customers, which is an advantage as it can help them to manage tasks effectively. There is also a section on the side for notifications which can notify the staff of tasks that needs to be carried out during the day. It includes timestamps and the tables that request staff which can help the user to prioritise certain tasks. Therefore, it is important for restaurant booking systems to have a clear design for the user to handle tasks in the most efficient way possible.

Secondly, the system emphasises on 3 main features: online reservations, restaurant management and marketing.

For online reservations, the customer is free to make and cancel the reservations even when the restaurant is closed. This is an advantage for the restaurant because restaurants can manage their availability. The booking is made by choosing the amount of people that will be dining and the desired time and date. The customer is also allowed to choose the dining area which gives the customer more freedom. This is an advantage because it can allow restaurants to enhance customer service. Another feature of the booking is that the customer can pick the language that they are familiar with. In this way, restaurants can attract customers from around the world. To support the customers more, the customers can be notified of their reservations by sending reminders using phone and email. In this way, customers are more likely to attend their reservations and the service of the restaurant can be maintained. In terms of payment, the client can pay for their reservations online or make deposits during busy shifts or sell empty tables. Enabling the client to do this can reduce the risk of decrease in profits and most importantly, it can save time for the staff to manage bookings.

Section to send notifications by phone or email

Option to select dining area, date, number of people dining, language



For restaurant management, the user can create staff accounts and the manager is granted permission to access staff activity. This is an advantage because the manager can handle staff workload and request staff to deal with customers quickly, so time management and communication is effective. Along with this, each account is given a calendar with tools to support them with their daily tasks. Tools include managing seating and tasks for staff. This calendar can also be accessed on multiple devices which is an advantage because it saves time for staff when they are dealing with a lot of customers in busy periods and it is easier for them to manage a lot of customers at once.

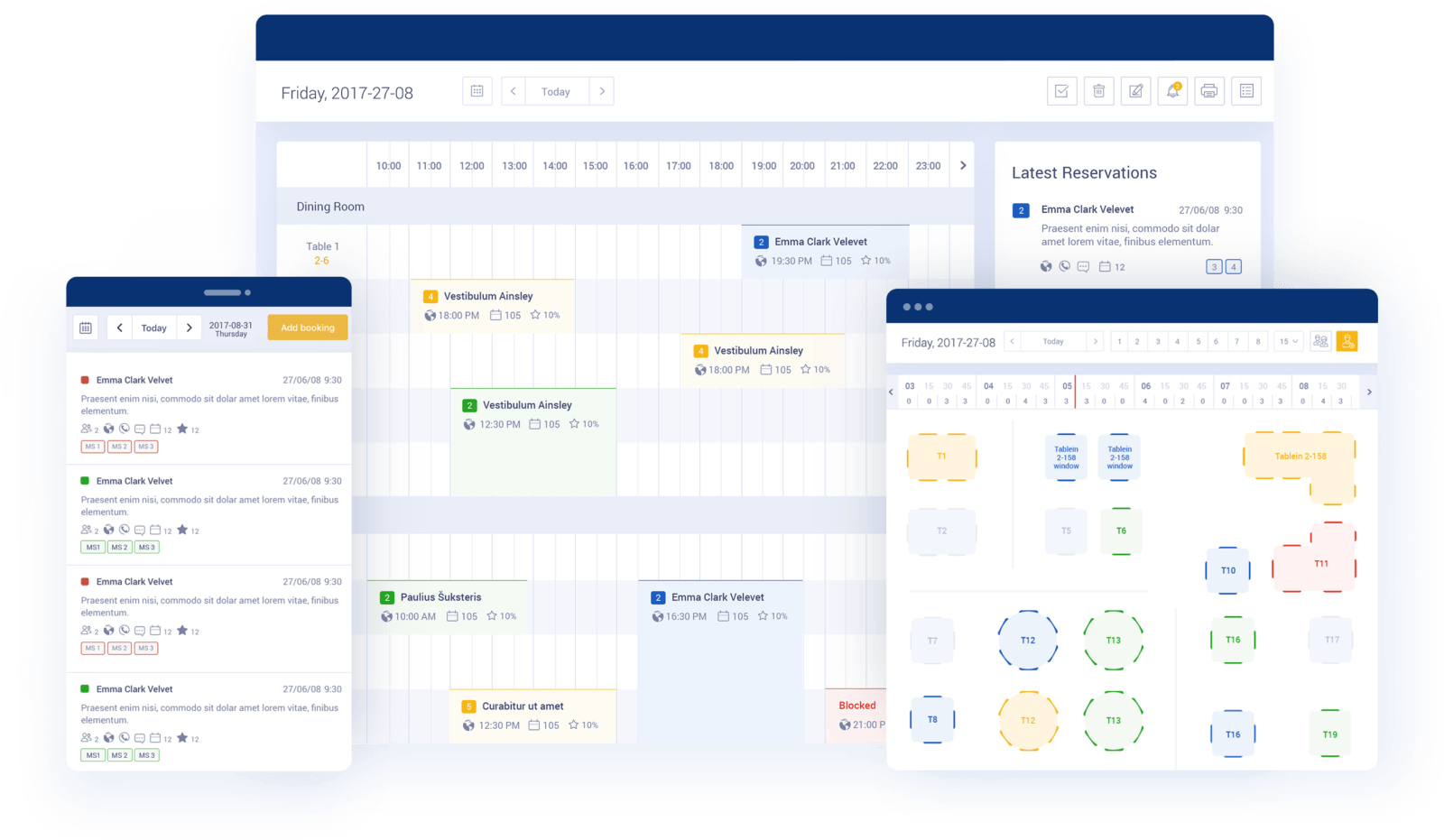
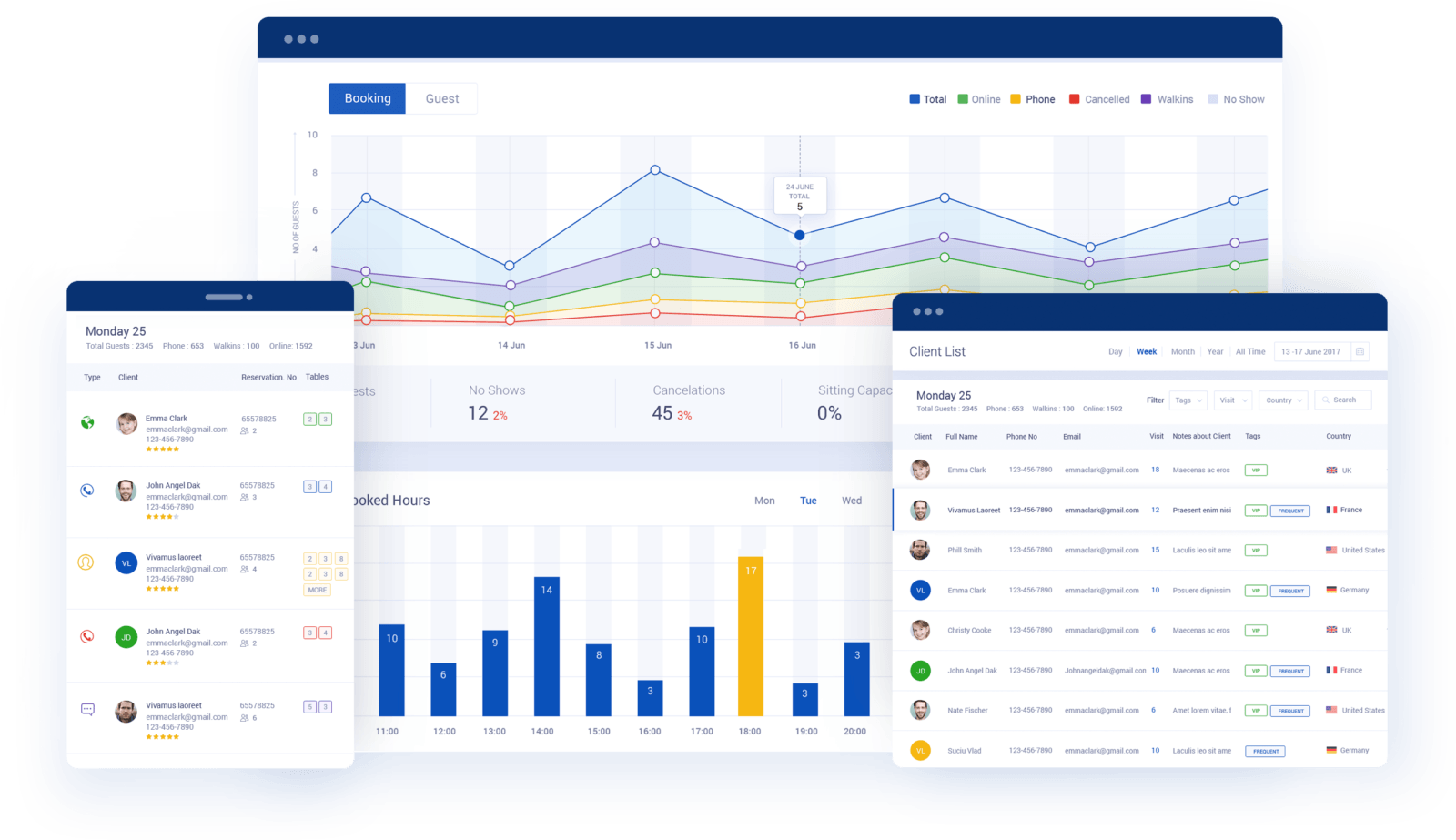
For marketing, the restaurant can analyse their performance with marketing tools such as graphs that show information on bookings and seating capacities. Using tools like this can help the restaurant to improve performance and maximise profits. Also, managing client database can improve restaurant performance. Implementing a loyalty system and adding notes for clients can increase the likelihood of regular visits. As well as this, restaurant service can be improved by enabling the customers to give feedback. Communication between the restaurant and the customers is important because it allows the restaurant to analyse their service so that they could make any changes to their system and it also allows the customers to give feedback.

Table layout

Calendar and notifications

Graphs and client list to analyse performance

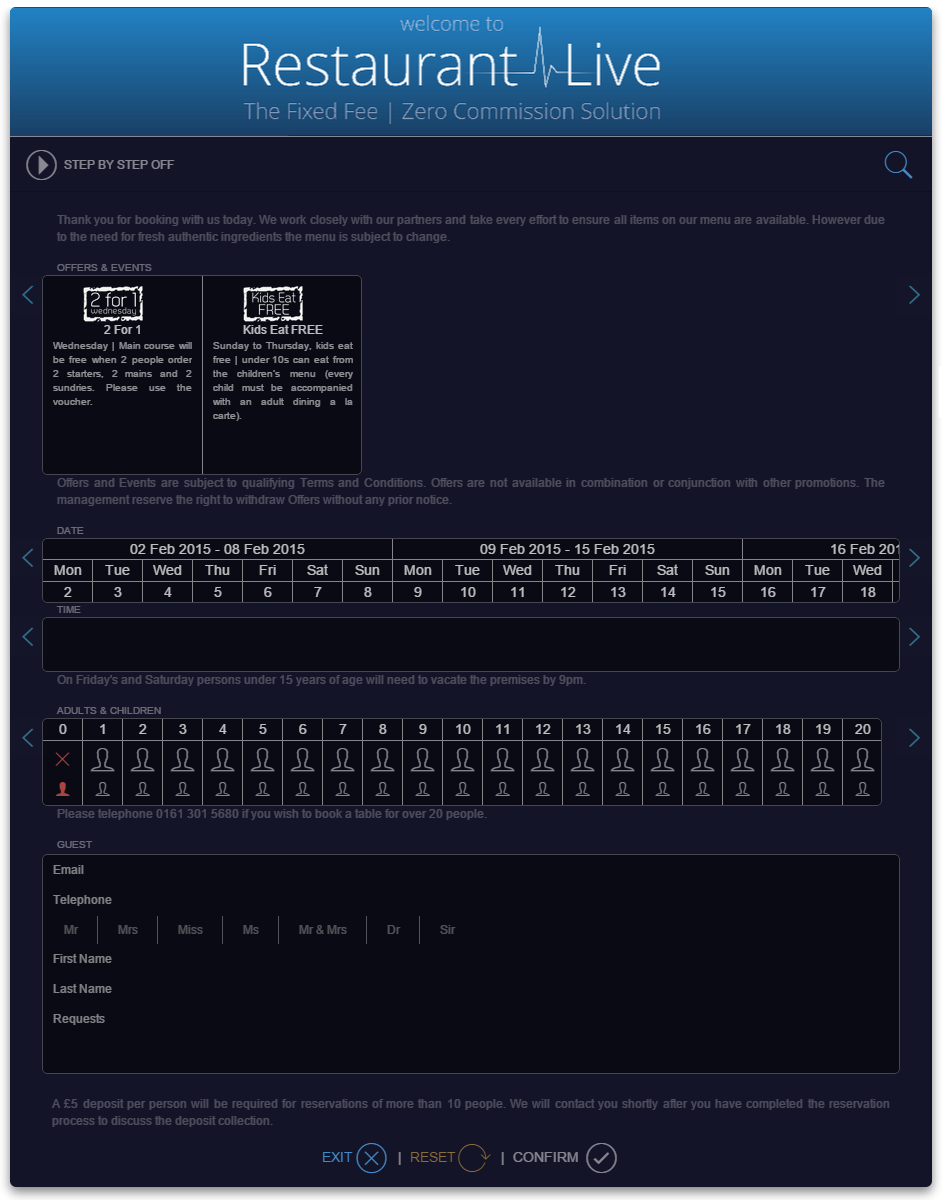
Finally, the booking system offers pricing plans for people who own one or multiple restaurants. The first plan is a free trial with the basic features of booking which is useful for owners to have a taste of what the booking system is like. For the rest, the prices of the plans will increase as the number of features increase.

#### Summary

|  |  |
| --- | --- |
| Positives | Improvements |
| Seating layout is large and clear | Section for customers to be able to tell staff of any dietary requirements |
| Notification panel for staff to manage workload | Customers should write down alternative times if they cannot make it to their preferred time |
| Calendar for managing bookings and events | Booking system should include menu for customers to look at |
| Customers can choose language | System should be free for anyone |
| Customer can choose dining area | Customers can choose offers |
| Customer can manage reservations even when restaurant is closed |  |
| Customer can be notified of reservations |  |
| Customer can pay online or make deposits or sell empty tables |  |
| Accounts can be made for any user |  |
| System can be accessed on multiple devices |  |
| Restaurant performance can be analysed |  |
| Client database can be managed |  |
| Loyalty system can be implemented |  |
| Customer can give feedback |  |
| System offers pricing plans |  |

For the proposed solution, the accounts will be made for the staff only as customers don’t need to make an account and waste time. The system can be accessed on multiple devices as I will be implementing it in a web app. For the bookings, a calendar is displayed clearly for the customers and notifications will be sent to the customer to remind them of their bookings. A section should be provided for the customer to tell the staff of any additional information and they should write alternative times if they cannot make it to their preferred time. Most importantly, a menu and offers should be displayed for the customer. The customer will not pay until they have dined at the restaurant unless they need to reserve a table for a special event. For the staff, the seating layout must be displayed large and clear in the user interface for them to attend to customers quickly. A notification panel should be implemented for the staff to manage their workload. The staff can also manage the client database.

### Restaurant Live

This is a booking system that is designed for restaurants to improve their service using a variety of features. Their website promotes the idea that they want to make their system affordable for everyone by limiting it to £50 a month – the number of bookings does not affect this. This is an advantage because customers will not worry about dealing with subscription plans, but it could not be an advantage for small restaurants due to their size. They have also emphasised the fact that paperless bookings are better than paper bookings. A client who has used this system says that it has ‘dramatically increased bookings’, which is evidence that making online bookings is efficient. In addition to this, they have presented us with some statistics that proves online reservations are useful.   
*‘67% of diners expect to be able to make table reservations directly online’* – this shows that many customers would like to reserve tables online.  
*‘On average 44% of reservations are taken during peak service hours when staff are potentially too busy to take the reservation’* – online reservations are useful because they save staff time when managing their workload.  
*‘14% of all online reservations are taken outside restaurant opening hours’* – allowing customers to book in online outside opening hours increases the restaurant service.

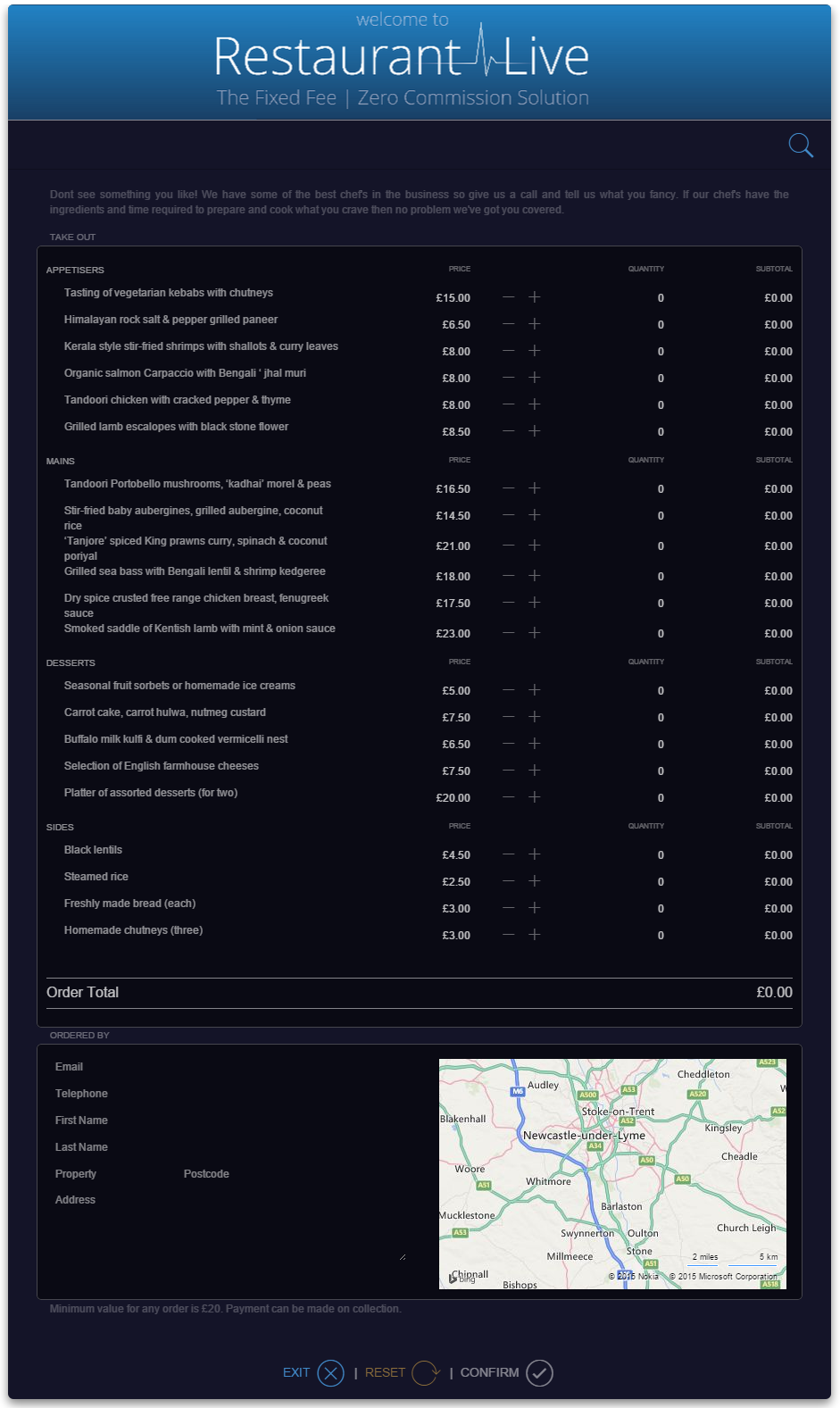
Option to pick a date

Option to pick an offer

Option to pick number of people

Contact information

Next, they have given us an example of the booking widget, which can be added to the restaurant’s website. Allowing restaurants to do this will make it easier for customers to reserve tables but also, easier for staff to handle with workload during peak times. In this widget, they allow customers to choose the date and the number of people dining. They also allow customers to choose an offer that is proposed on the day of the booking. The advantage of this is that customers can book outside restaurant opening hours.

In the next example below, it shows how a takeout order is displayed to the staff. The layout is clear because it shows the dishes that the customer has ordered and the contact information of the customer. The advantage of having the customer’s contact information and a summary of their order supports the staff in what they need to prepare for the customer.

Map to show where the order will be delivered to

Customer contact info

Summary and order total

List of dishes with info

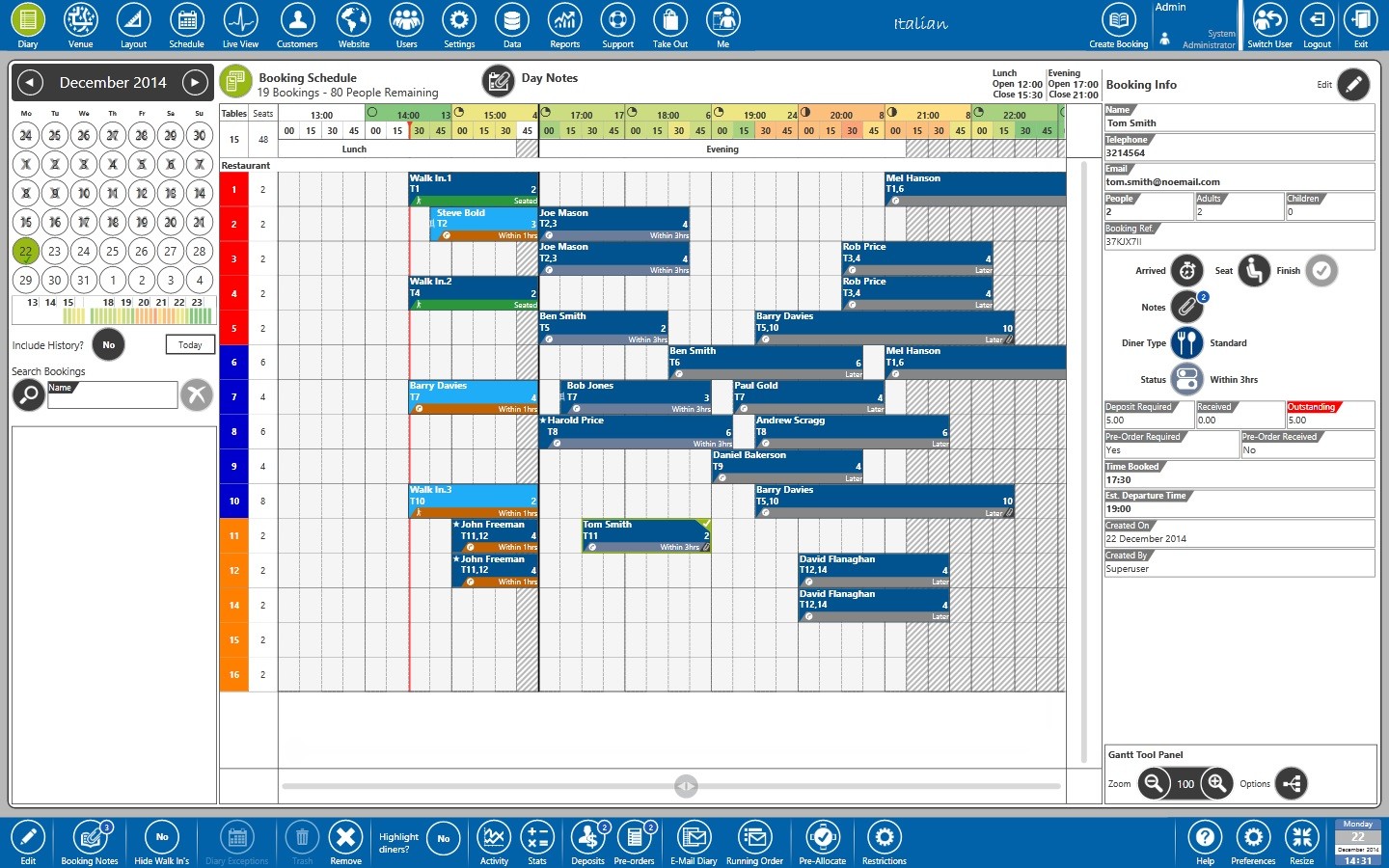
They have also given us statistics that show us online takeaways have increased and most people order from a mobile device. *‘More than a quarter (27%) of all online takeout orders come from a mobile device.’* This shows that more customers are ordering food online because of the accessibility. Allowing the customer to order food online will enhance customer service. Therefore, having a clear layout and a menu will increase the likelihood of the customer ordering from the restaurant.

The next feature is the ‘Reservation Diary’. This is a feature that helps restaurants to maintain customer service and manage reservations by using a calendar. This can be accessed on multiple devices which is useful for staff when managing with workload at peak times. Making the system more accessible on multiple devices would be a good feature. In this example, there is a calendar with different events that have been reserved and for each event, there is a section for the booking info. The booking info includes contact details of the customer and the status of the booking to help the staff.

Overview of events

Calendar

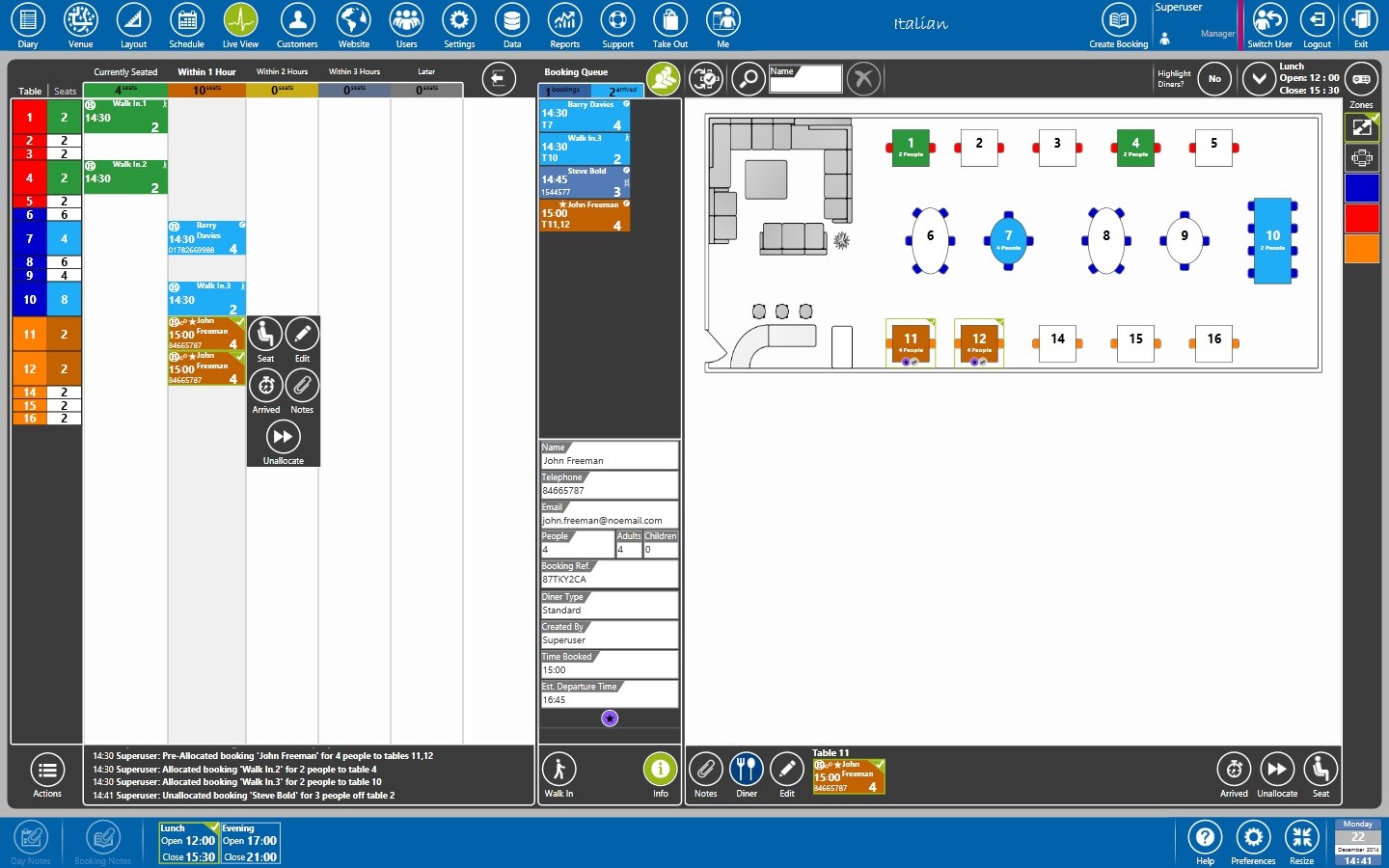
Booking info



*‘Restaurants with customised loyalty programmes average 88% more profits than competitors who do not’* – Implementing a loyalty system increases the profits on average.  
*‘Existing customers spend 67% more than new customers’ –* Regulars are more likely to spend more.  
*‘64% of restaurants collect and actively use customer satisfaction information in determining opening hours, targeted customer offers and special events’* – doing more for the customers’ satisfaction will improve the performance of the restaurant.

The next feature is table management. Tables can be managed by dragging and dropping reservations. The restaurant layout includes the table numbers and sizes. To also support the staff with this, the tables can be colour coded. Table management is important when maintaining restaurant service.

*‘65% of diners have noticed an increase in the use of and attribute improvements to the use of restaurant management services.’* – Using these tools will improve the management of the restaurant, so the tools should be used to its fullest extent.  
*‘70% of diners believe services adoption leads to an improvement in speed/attentiveness of service and/or increase in overall accuracy’* – tasks can be completed quicker and customers can also be attended to quickly while maintaining accuracy.  
*‘35% of diners tend to dine out more often when service is timely and as requested’ –* If the customers’ requirements are met and their satisfaction is high, they are more likely to dine out more.



Bookings

Table layout

Info of bookings and table management

Customer information

The next feature that the system proposes is ‘Diner Marketing’. This feature includes marketing tools that supports the staff in promoting their restaurant to customers. This marketing template can include information about the restaurant and the menu. Customers can be engaged by sending personalised emails about changing menus or special promotions and offers. This is an advantage for establishing or maintaining relationships with the customers.



Template for restaurant marketing

Online presence is important when it comes to targeting customers. These statistics that they have presented to us prove why restaurants should make a good impression online.  
*‘80% of people will look up a restaurant online prior to visiting for the first time. With 62% saying they were less likely to choose a restaurant if they can’t read the menu on a mobile device.’* – when promoting the restaurant, the presentation has to be clear for the customers. First impressions always count.  
*‘76% of restaurant searches lead to sales. Restaurant Searchers have the fastest turnaround and highest conversions of any online segment.’* – people who are travelling are more likely to search up for restaurants and it’s important to gain attention from these people.  
*‘64% of diners with reservations need direction/location details. 23% of smartphone search users expect to be within walking distance and provided with directions online.’* – it is important for restaurants to present customers with clear contact information and location details so that customers get a good impression of the restaurant they would be dining at.   
It will be an advantage if restaurants present themselves clearly online because profits for the restaurant will increase and the number of regular customers will also increase.

|  |  |
| --- | --- |
| Positives | Improvements |
| System is affordable to anyone with unlimited bookings | Customer should be able to cancel bookings |
| Customer can reserve tables at any time | A notes section to write the customer’s information |
| Customer can order food online | The staff should be able to move tables if there is a large group of people dining |
| Calendar for staff to manage bookings and edit customer’s information | Customer should write alternative times of the booking |
| Tables can be managed for any customer easily | Menu should be presented when customer is reserving a table |
| Personalised emails can be sent to customers to enhance customer service | A search bar to search for reservations or customer information |
| Customers can be informed of menu changes and special promotions and offers |  |
| Marketing tools to give a good impression of the restaurant |  |
| Staff can deliver food to clients easily with a clear summary of food ordered and address of client |  |
| Staff can attract diners by promoting their restaurant using tools to edit their website |  |
| System can be accessed on multiple devices |  |

#### Summary

For the proposed solution, the system will be free for everyone to use. The customer can reserve tables at any time, and they can order food online. Also, they should be informed of their reservations. For the staff, tables should be managed for any customer easily by implementing a calendar to manage bookings and edit information.

## Comparison between similar systems

‘tablein’ was a good system in terms of its restaurant management such as allowing the customer to choose their own language when they are reserving and notifying the customer of their reservation. Notifying customers will decrease the chances of people showing up late to their reservation. It was also good in terms of helping restaurants to enhance restaurant service by allowing staff to review performance and allowing customers to sell empty tables. In terms of its objectives, it was successful in helping small restaurants to manage their bookings and maximising profits.  
‘Restaurant Live’ was a good system in terms of promoting the restaurant and giving a detailed summary for takeout orders. It allowed people to promote their restaurant with marketing tools such as a marketing template and the ability to present a menu. Also, the takeout orders included a detailed summary and a map that clearly presents where the order needs to be delivered to. In terms of its objectives, it was successful in promoting restaurants and presenting a lot of details for the staff to help them do their tasks.

## Questionnaire

To clarify the end user’s requirements even further, I have conducted a questionnaire using Google Forms for people to complete. Since I will be creating a general restaurant booking system, multiple users will be using the system so a questionnaire will be useful in this case.

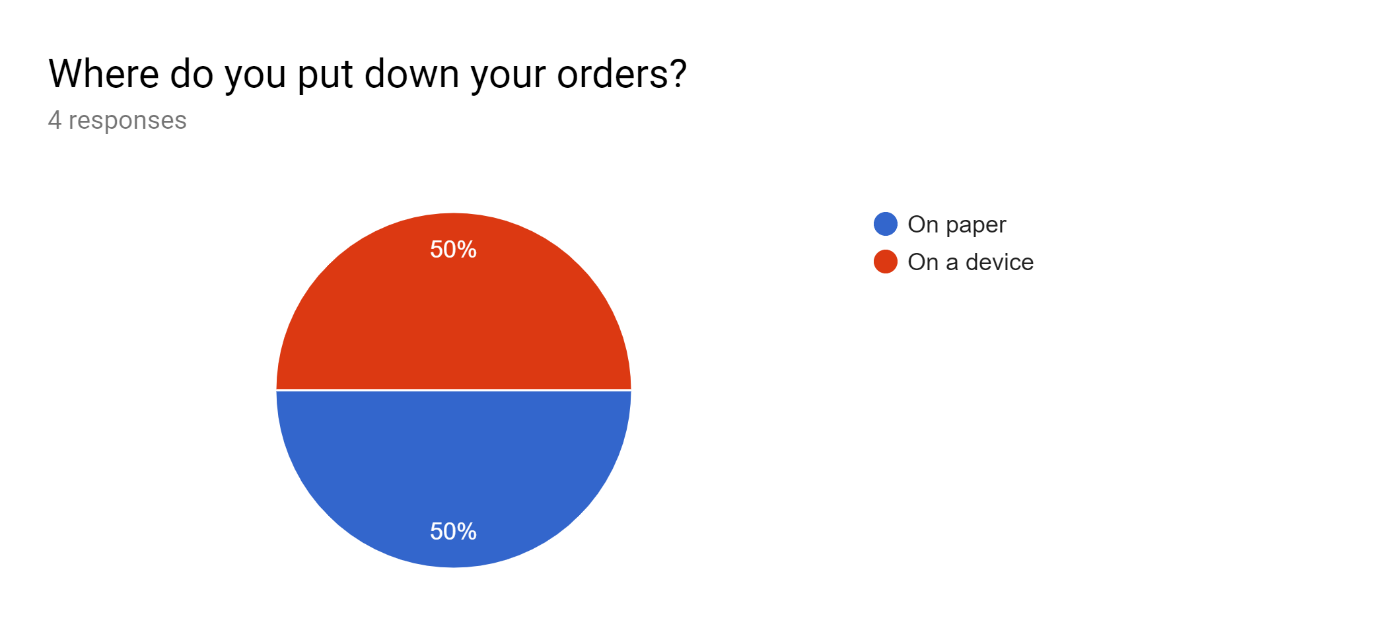
Most of the respondents are students. I have left the questionnaire up for a week. In this time, 60 people have replied. The questionnaire will target any potential end users, and this could be the staff or the customers. This helps me to categorise answers to decide what the potential end users of the system needs and the objectives of the proposed solution. I have separated the questionnaire in 2 main sections. The first question asks whether the user has worked at a restaurant or is working at one.



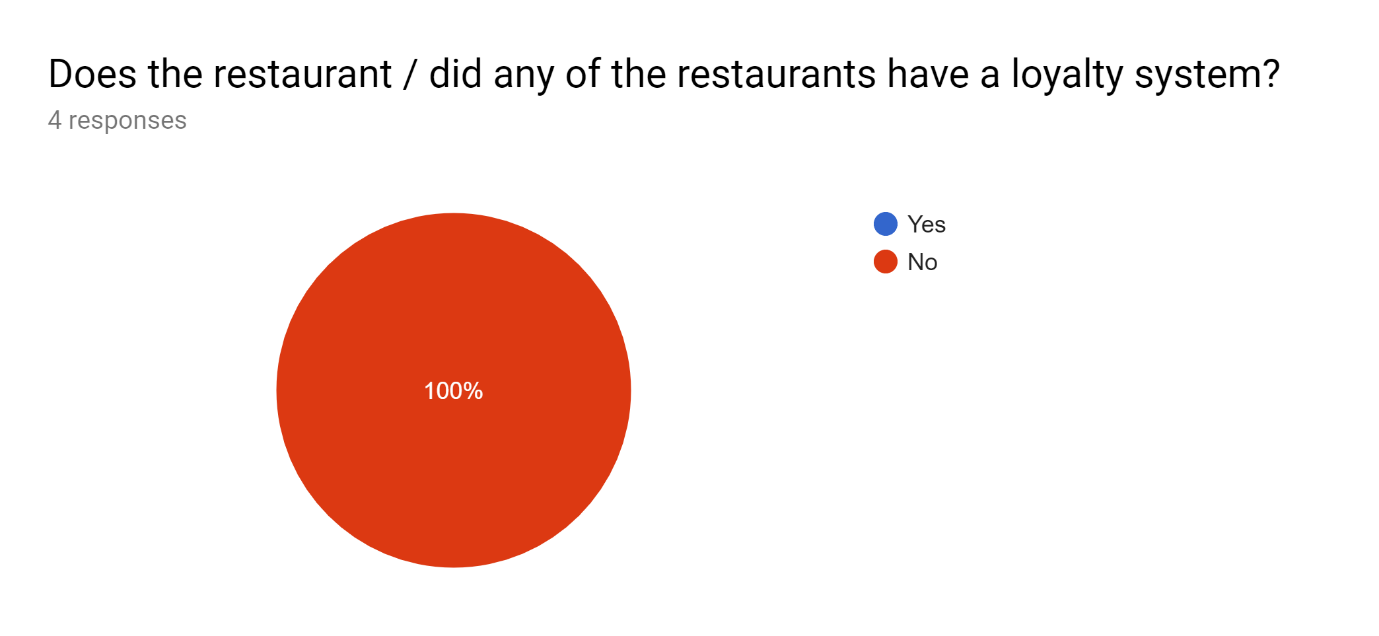
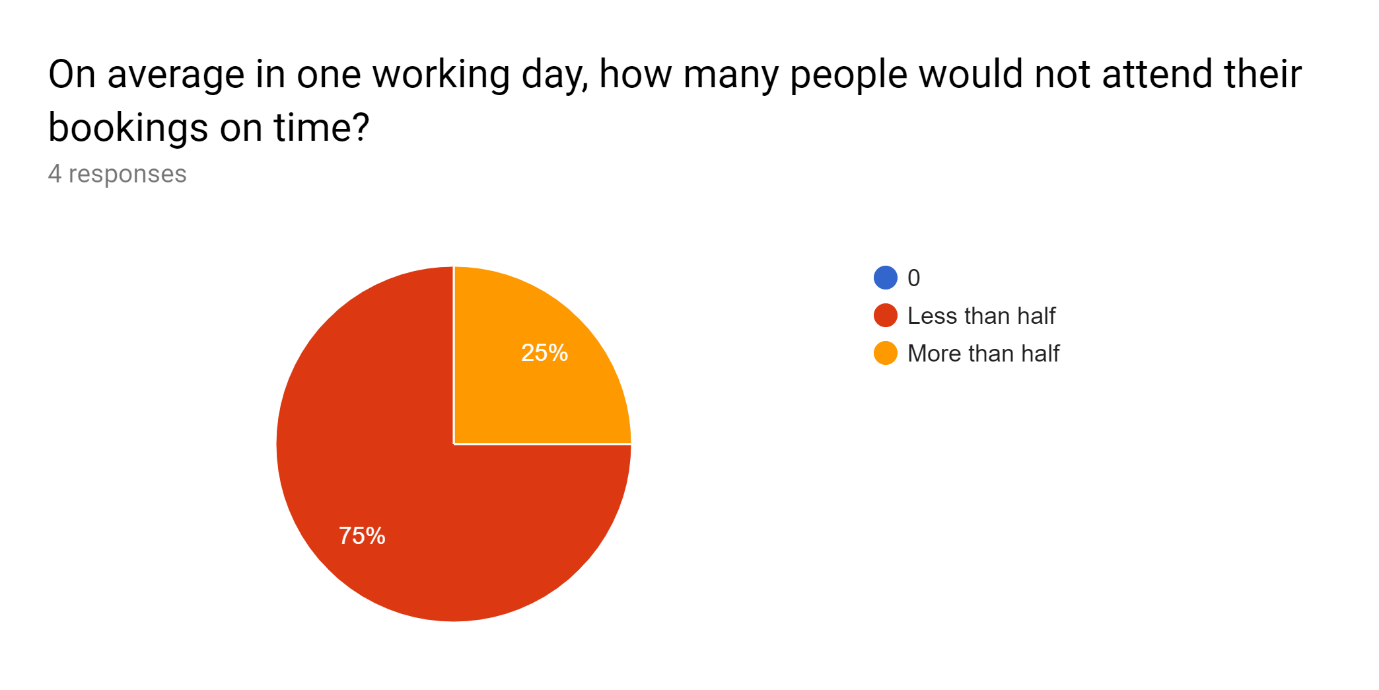
From this question, most of the people who replied do not work in a restaurant. However, since the project will mainly focus on the bookings, the questions about ordering and reservations is more important. The user interface should be user-friendly, and the layout should be clear for the user to use.

Depending on the answers, the people who have worked in a restaurant will complete the whole questionnaire and the people who did not work in a restaurant will skip the next section about working in a restaurant and go on to the section about ordering and reservations. The next section is targeted for people who have worked in a restaurant. Analysing the answers of those who worked in a restaurant will be useful when it comes to implementing features for staff accounts.

### C:\Users\kelly\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\1308482D.tmpWorking in a restaurant

From this question, more than half of them think that they do not have effective time management. This shows me that managing workload could be a problem and therefore the system should aim to make this better. The system should have a clear layout to help the user.

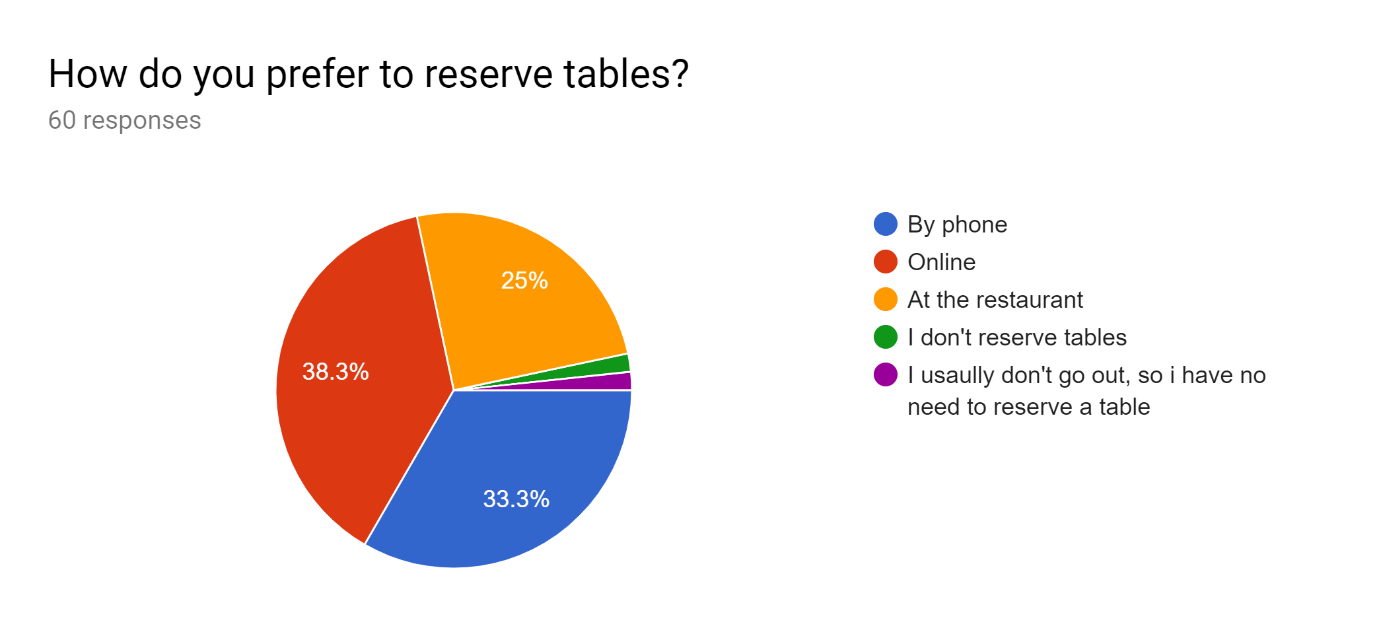
From this question, half of the people who have worked in a restaurant either put their orders down on paper or on a device. The advantages of putting orders down on paper is that it can reduce the likelihood of making errors or data being lost because the device can shut down at any time and data can be lost. But the disadvantage is that it could waste paper and more money would have to be spent on paper and transferring data on to the computer for example can take time. However, for this project, I aim for the system to be available on a device.

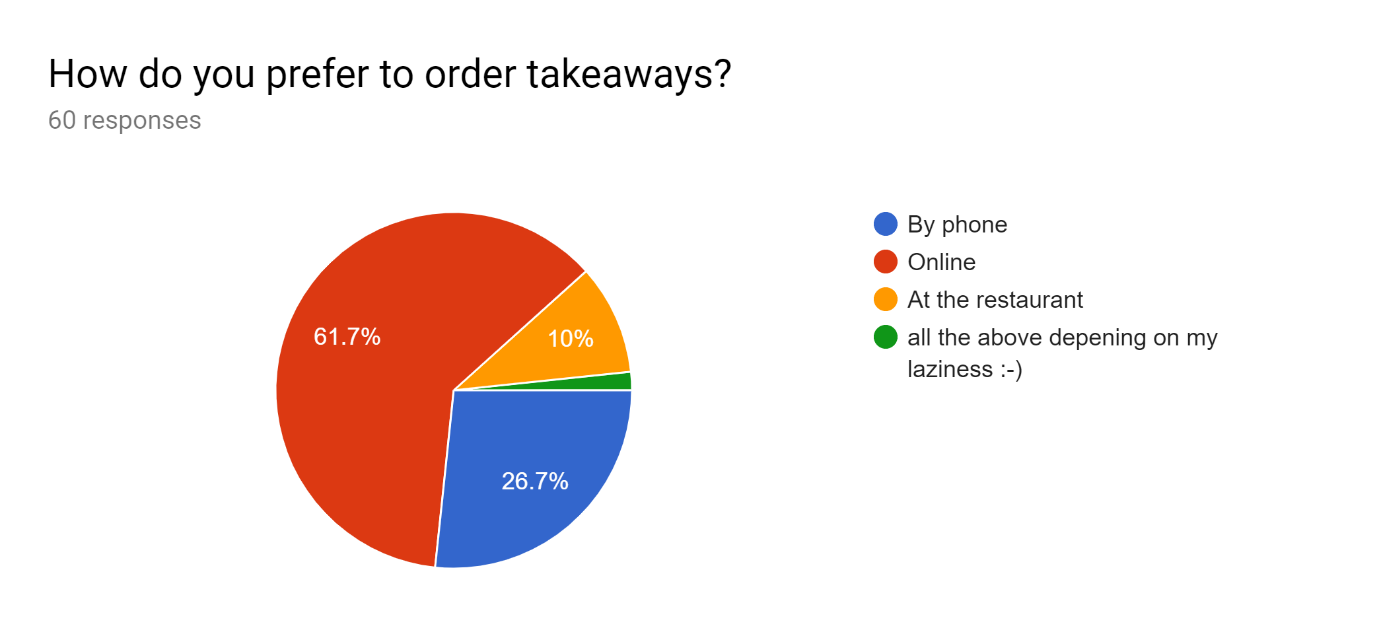
From this question, it is clear to see that the restaurants that they worked at do not have a loyalty system. This is probably due to the amount of money that they make or the size of the restaurant because implementing a loyalty system in a small restaurant for example would not be beneficial. Therefore, if I will have time, I will try to implement this feature.

From this question, it is clear to see that no one would attend their bookings on time which means it is useful if a feature to minimise the chances of this happening is implemented because in reality, it is difficult for clients to attend their bookings exactly on time. A feature like asking the customer for alternative times would be useful or sending notifications to the customer.

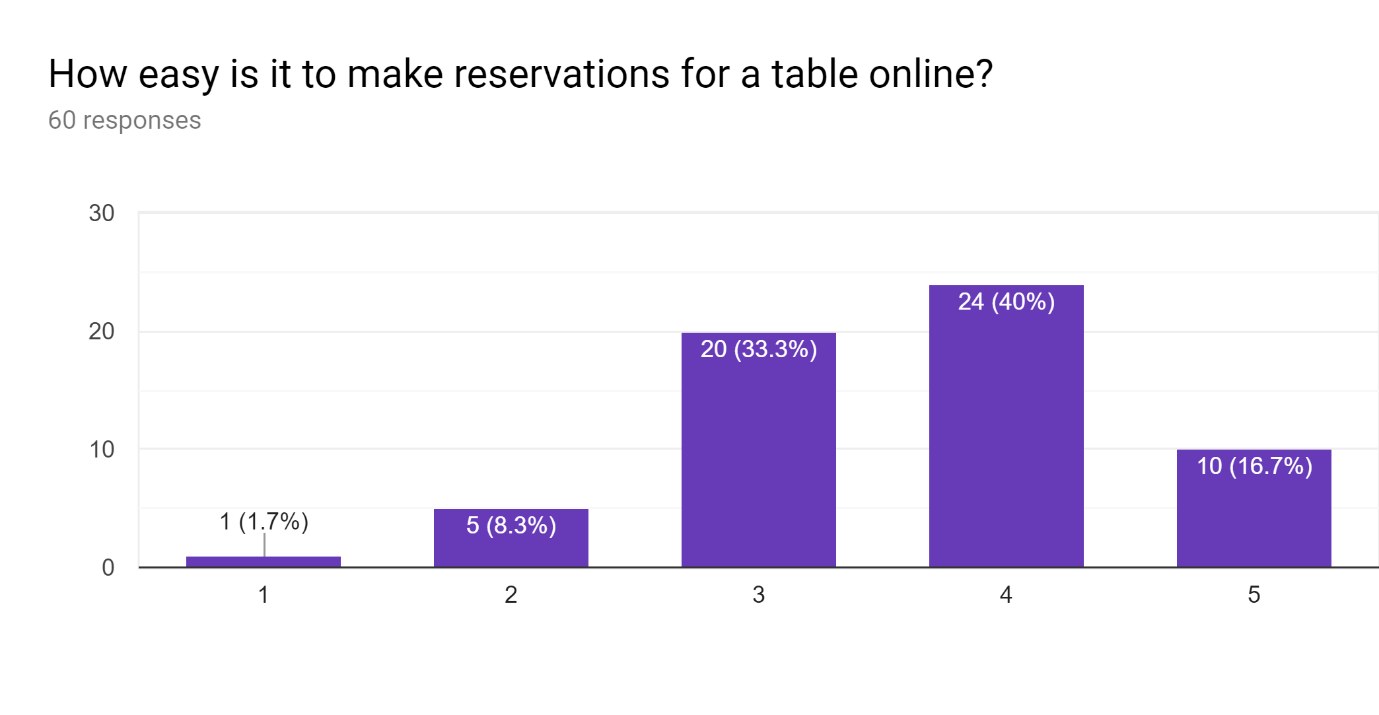
### Ordering and reservations

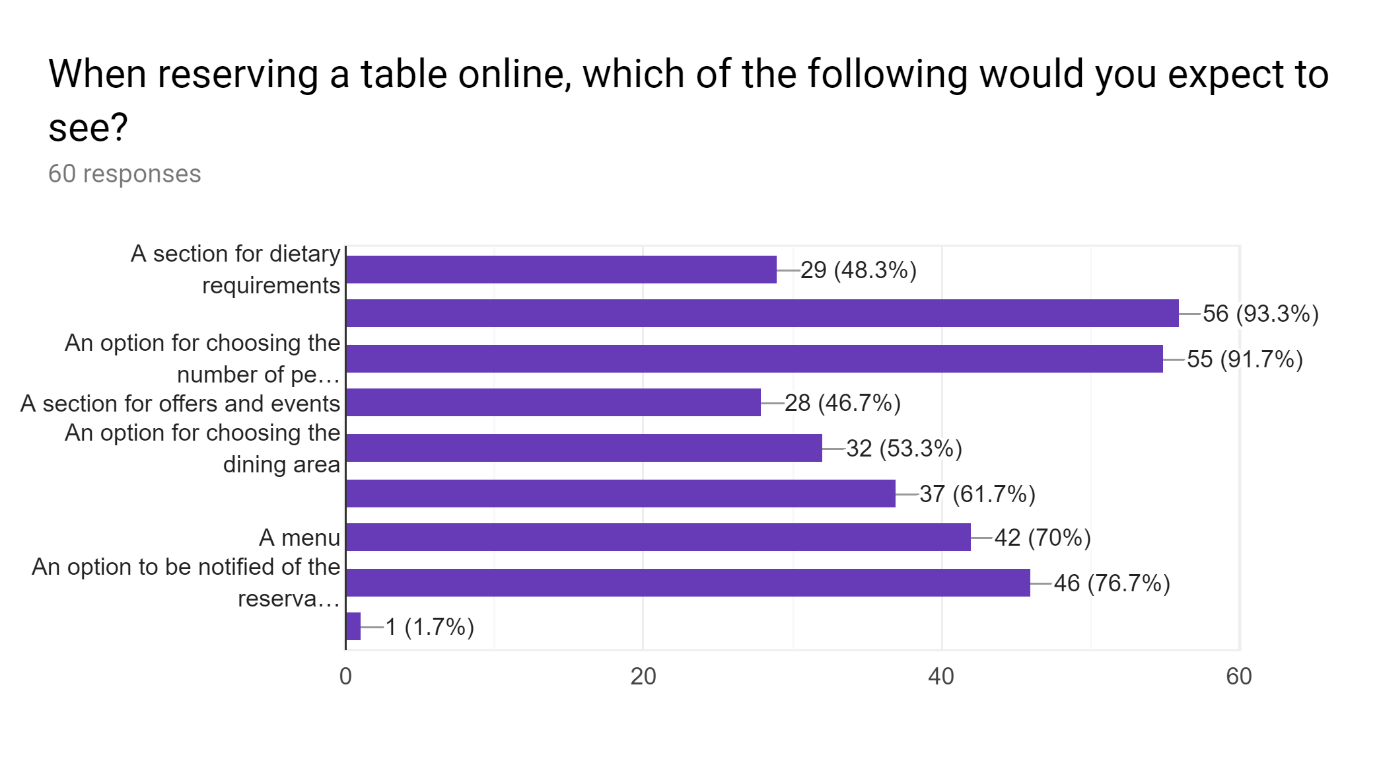
Everyone will answer the questions in this section and the results from this section is important as I will be making a booking system.

****From this question, a third would prefer to reserve tables by phone but more than a third would prefer to reserve tables online. A reason why people would prefer to reserve tables is probably because customers would like to talk to the staff directly or that they have limited knowledge of reserving tables online. 25% would prefer to reserve at the restaurant. However, more people will prefer to book online as the time goes by.



From this question, a lot of people prefer to order takeaways online. People like reserving tables by phone but more people like ordering takeaways online. Ordering takeaways have been around for a long time and it has increased significantly after the use of smartphones. However, for time efficiency, it is important that I implement a feature where customers can reserve tables online.

From this question, more than half say that reserving tables online are easy or very easy. This shows existing restaurant booking systems have the features to help their customers for reserving tables. But since only a few people think it is not easy to make reservations for a table online, I will aim to implement features in the system that will make this better. To make this better, I will make a user-friendly interface and tools to help the user to do the tasks that they need to do.

The last question is important when it comes to implementing the booking feature in the system. The respondents will choose from the options that they think should be implemented or write their own idea down. The options are:

* A section for dietary requirements – 29 people would like this; the communication between the restaurant and the customer is important.
* A calendar to choose the date and time – 56 people would like this; this is important for the customer to choose when they would like to reserve a table. Therefore, I should implement this clearly in the system.
* An option to choose the number of people dining – 55 people would like this; this is important for the staff to know how many tables they should be reserving, and this should help them prepare for their customers. Therefore, I should implement this.
* A section for offers and events – 28 people would like this; this feature is not highly expected, so this feature is an option.
* An option for choosing the dining area – 32 people would like this; therefore, it is important to present this clearly for the customers.
* A section for contact information – 37 people would like this; it would be beneficial to implement this to help the staff to update the customer for example.
* A menu – 42 people would like this; the menu is an important aspect of a restaurant because customers are most likely going to judge the restaurant just from its menu.
* An option to be notified of the reservation – 46 people would like this; notifying customers will decrease the chances of them showing up late to their reservation. Therefore, I should implement this in the system.
* Since the question also allows the respondent to put their own answer in, one person has mentioned about a feature to let them know if the restaurant is full. This will be important as it tells the customer that they cannot reserve tables at the restaurant.

### Summary

* I will aim to implement a feature that will help the staff to manage their time effectively – the system should be accessible on a device.
* A calendar for the user to pick a date and time of reservation
* An option to choose the number of people dining
* An option to be notified
* A section for dietary requirements
* A section for contact information
* A menu to show the customer of what they will be expecting at the restaurant
* The customer should be able to reserve tables online - this should decrease the chances of a customer showing up late to their reservations.
* A feature to let the customer know if the restaurant is full

From analysing the answers of this questionnaire, I have identified and clarified the user’s requirements. Next, I will model the new system using a Data Flow Diagram to show the data processes and the inputs and outputs.

## A screenshot of a video game Description automatically generatedData Flow Diagram

## **Data sources and destinations**

This table shows where the data is inputted and where it would be going to.

|  |  |  |
| --- | --- | --- |
| Data | Source | Destination |
| Username | User | Accounts database table |
| Password | User | Accounts database table |
| First name | User | Accounts database table |
| Second name | User | Accounts database table |
| Email | User | Accounts database table |
| Number of people dining | User | Restaurant database table |
| Date of booking | User | Restaurant database table |
| Time of booking | User | Restaurant database table |
| Table layout | Restaurant database table | User interface |
| Menu | Restaurant database table | User interface |
| Restaurant information | Restaurant database table | User interface |

## **Data volume**

I will be using an SQLite database to store the data. SQLite is good at handling data when thousands of users are using the booking system. I will be storing a lot of data from customers and restaurants. A lot of users will be using the system at the same so the system will be implemented in a web app. SQLite can handle about 400 to 500 thousand HTTP requests every day and can fit databases up to 140 terabytes in size so storing data such as user information and booking information is not a problem.

Limitations  
There are some limitations to this project. The biggest one would be time because there might not be enough time to implement features such as managing the service of the restaurant or reviewing the restaurant’s performance. Therefore, I will aim to at least implement features that will help the system to do the main task of the booking system – to book reservations and manage them. The features should help the customer to book easily and the staff to manage their workload. Also, working on how the system would look in a mobile browser would take a long time, I will just aim for the system to be able to work on a computer at the very least. Therefore, the hardware constraints will be on Windows computers. Another limitation would be my skill level, I might not have enough time to implement different algorithms for increasing efficiency and therefore, it is important that I should aim to implement a searching algorithm at least for searching information and to help staff to prioritise certain tasks.

## Final objectives

Overall, the restaurant booking system should aim to help customers to reserve tables and search for existing restaurants. The restaurant booking system should also aim to help staff to manage their time effectively. Identifying the end user’s requirements by analysing similar systems and questionnaire results has helped me to identify the final objectives of the new system. The user interface must be clear for the user to identify what they need to do as they will have limited IT knowledge.

### Start / ordering

The system must have a starting page that will take the user to the page that they will need to go to. This page will also be for customers to reserve tables immediately. This will include:

* A section that will have buttons for the user to login or register – this will take redirect them to the login page or the register page.
* A search bar for the customer to search up the restaurant – the customer will search up the name of the restaurant and find information about it.
* The restaurant information should at least include the name and type of the restaurant, the menu and the address of the restaurant and most importantly, whether it is full or not.
* The customer can pick a date and input information such as the number of people dining, the dining area, an option to be notified, a menu and any other additional information which may include dietary requirements and alternative times.
* The customer should also be able to order takeaway from the restaurant by the menu. Customer should be able to choose the dishes easily and provide contact and delivery information.
* A summary of the orders should be displayed clearly.

### Login

The system must have a login page that will then redirect them to the home page. This will include:

* A section for the user to input their username and password – the form should be submitted, and the database should be updated.
* A button that will redirect them to the start page – for the user to go back to the start page.
* A button that will redirect them to the register page – this is if the user has not created an account yet.
* Instructions to help the user where to put their username and password in or whether they are looking for something else.

### Register

The system must have a register page that will then redirect them to the home page. This will include:

* A section for the user to input their username, password, first name, second name and email – the form should be submitted, and the database should be updated.
* After they have submitted their form, this should login them in straight away and redirect them automatically to the home page.
* A button that will redirect them to the start page – in case they are not at the page that they need to be on.
* Instructions to help the user how to create an account.

### Main

The system must have a home page that will display all the features for the staff to use for managing their work. This will include:

* A section that displays the table layout of the restaurant – the user should be able to label the tables and manage tables easily.
* A section for notifications – this will update the user on any work that they need to do, and this will include timestamps and the tables that needs assistance.
* Notifications that should consist of the tables that needs assistance and any takeaway orders for example.
* A calendar – the staff should be able to manage reservations and edit information if necessary.
* A section to order dishes for the table that they are managing – the option to select a dish and a summary of the orders.
* A section to write additional notes for the customer or the order.
* A button to separate the main dishes from the starters or to separate the dishes so that the staff would know which dishes should come in order.
* A search bar for the staff to search up the dishes or the customer’s information.

## Proposed solution

I will be using the programming language Python in Visual Studio 2019. Visual Studio is a popular IDE (Integrated Development Environment) because it is free for everyone and it will allow me to execute the program and use the program on any Windows computers. Visual Studio is also good for its debugging tools. I could set breakpoints on any line of the code and fix any errors if the program cannot execute properly. Therefore, fixing the program will be efficient. Also, Visual Studio allows me to peek definitions to view and edit code without leaving from the code that I am currently writing. This will help me to be time efficient and to fix code simply without having to look through the whole code. Additionally, Visual Studio includes a feature called IntelliSense which supports programmers in code completion and other errors to speed up the process in programming. This will be useful to me as I will have limited time in implementing this project.

I have chosen Python as my programming language because I can build web apps with Python Django. Python Django allows me to build web apps with HTML, CSS. Web browsers use HTML documents to render pages and CSS to design the HTML templates. Python Django uses an MVC architecture which stands for Model-View-Controller. A model can be created, and this will have data that will be stored in a database for example. The user can edit this model by retrieving information through the view. The view is the user interface. For the models to be edited, the users will interact with the view through the controllers. The controllers will take the inputs of the user and display the information through the view. This MVC architecture is beneficial for me as it helps me to understand where and how data will be processed to and from.

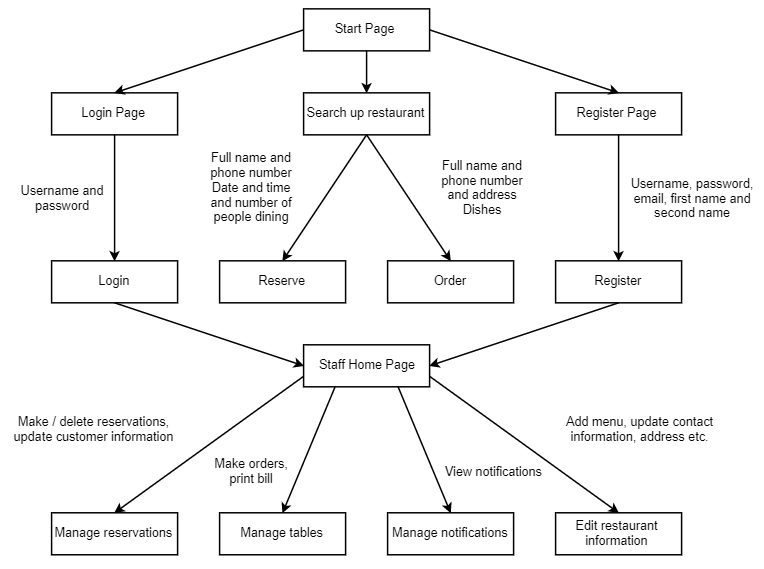
For handling with data, I will be using SQLite. Visual Studio enables me to download extensions such as SQLite which will support me in creating and managing databases. Python includes a module called ‘SQLite3’. However, Python Django’s default database is SQLite, so I don’t have to install SQLite into Visual Studio. For this booking system, it is essential that databases are used for storing user credentials when staff or customers login and for storing information on the restaurant or the customers. For a booking system, it is essential that a database is used in order to keep the customer’s information. It is also essential that a booking system must have a clear layout for customers to order from the restaurant. Python Django relies on object-relational mapping, data inputted from the user will be stored in databases and SQL statements will be automatically created.

# References

* [*https://www.tablein.com/#why-tablein*](https://www.tablein.com/#why-tablein)
* [*http://restaurant-live.co.uk*](http://www.restaurant-live.co.uk/)

# Design

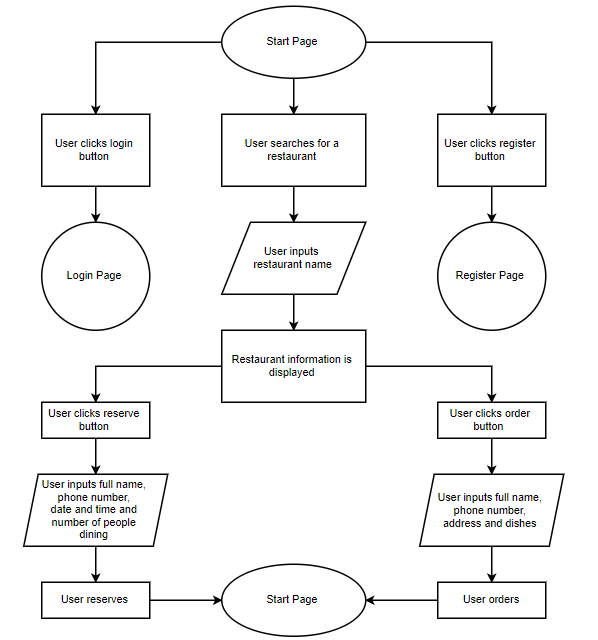
## Overall System Design

This diagram shows the overall structure of the system and shows how the inputs of the users will act or how it will redirect the users to a different page.

* The user will open the system on the web, and it will take them to the start page.
* On the start page, they can click to login or register, and each action will require an input.
* The login page and register page will take them to the staff home page once they have logged in or created an account.
* On the start page, they can reserve and order dishes by searching up the name of the restaurant if they are a customer.
* On the staff home page, they will have many features and tools to help them do their tasks.

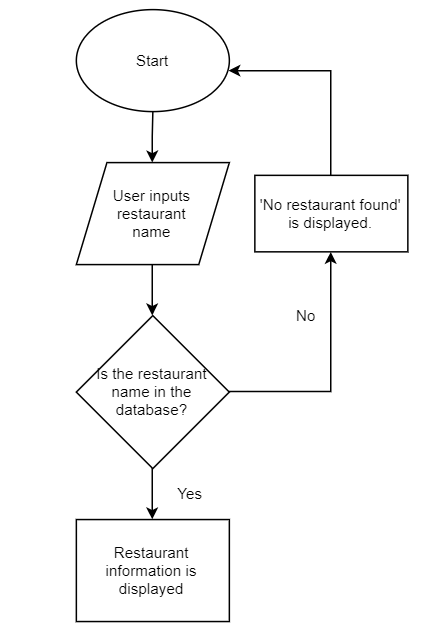
The following flowcharts will then show how each page or action will act. The information submitted from the user will be the minimum.

### Start Page



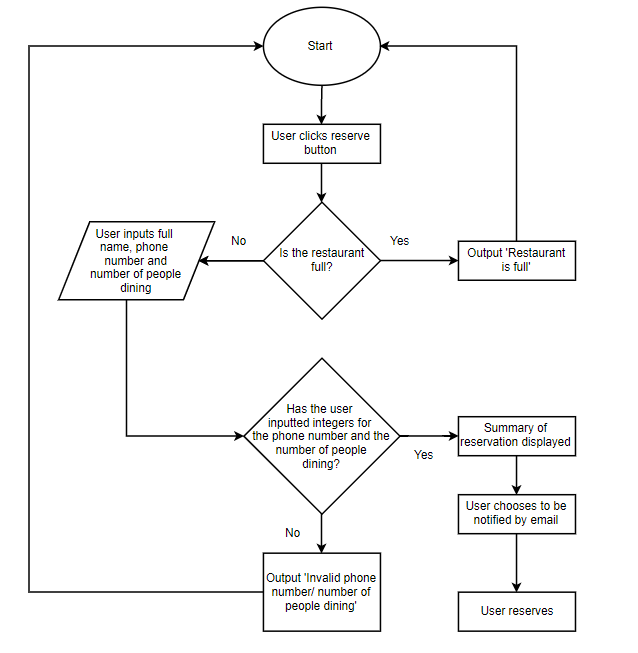
* When the user searches up for a restaurant, the restaurant information for each restaurant will be displayed in a list.
* The search feature should allow the customer to filter restaurants to make it easier for them. (e.g. in alphabetical order)
* The user can then choose to order food or reserve tables at that restaurant.
* The user will be required to input details which will be sent to the staff.
* The details from the customer should be displayed to the staff when they have logged in to their account.

### Searching up for a restaurant



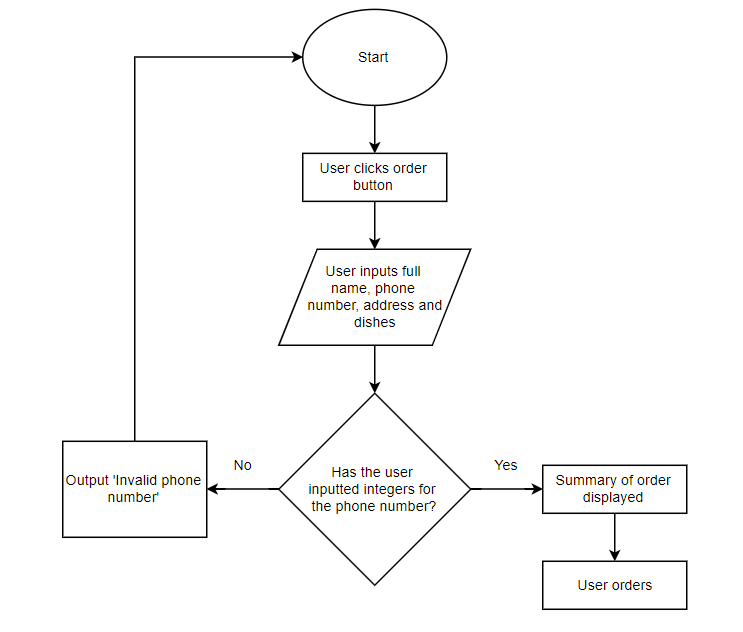
On the start page, a list of restaurants will be displayed in alphabetical order. The restaurant information will consist of the name, address, phone number, and the menu. The user can search up for the restaurant by name and a sorting algorithm is used to sort the list based on the input of the user. If the restaurant name is in the database, the restaurant information can be displayed. If it is not in the database, ‘No restaurant found’ will be displayed.

### Reserving



When a user has decided the restaurant that they want to reserve a table at, the user will click the reserve button and a test will be made to see if the restaurant is full or not. If it passes the test, the user can input the information and a validation check will be made. If the restaurant is full, ‘Restaurant is full’ will be displayed and the user will have to search for another restaurant. Error messages will also be displayed if it did not pass the validation tests and a summary of the reservation will be displayed for the user if it has passed the tests. The user can then choose to be notified. The user will input their email so that they can be notified by email. The information that the user has inputted should be submitted and displayed to the staff on the home page.

### Ordering



Once the user has decided the restaurant that they want to order from, the user will click on the order button. The user will input their full name, phone number, address and the dishes that they want to order. A list of dishes will be displayed based on the restaurant’s menu and the user will choose the dishes. Once the validation tests are passed and the price of the dishes are all added up, a summary of the order is displayed. The summary will include the name of the restaurant, the information inputted from the user and the dishes ordered with the total price. Once the user has submitted the form, these orders should be sent to the staff and displayed in the notification panel in the staff’s home page.

### Login

The user will be taken to the login page from the start page. The user will input the username and password and click the login button to login. The username and password will be validated to see if it is lower than or equal to 20 characters and if they are in the database. If they have passed these two tests, the user can login to the home page. If not, an error message will be displayed, and the user will have to follow the instructions again.

### Register

The user will be taken to the register page from the start page. The user will register for an account by inputting the username, password, email, and full name into the form. Validation tests will be made and if it passes all the tests, the account is made by saving the information to the database. The user will then login to the home page with the account that they have just made.

## Modular Design

* Start Page
  + Searching up a restaurant
    - Order takeaway
    - Reserve a table
  + Login Page
    - Login with account
      * Home Page
  + Register Page
    - Register an account
      * Home Page
* Home Page
  + Manage reservations
  + Manage tables
  + Manage notifications
  + Edit customer information

The user will begin on the start page. They can choose to search up a restaurant which will then give them two options: to order or to reserve a table. They can also choose to login, which will take them to the login page or to register, which will take them to the register page. Both pages can take them to the home page. The home page will have features that will help the user to manage their workload or the restaurant in an efficient way.

## IPSO table

From the analysis and the flowcharts, I can create an IPSO table which will help me understand the inputs and outputs of the data and where the data should come from and the processes that the data will go through of the booking system.

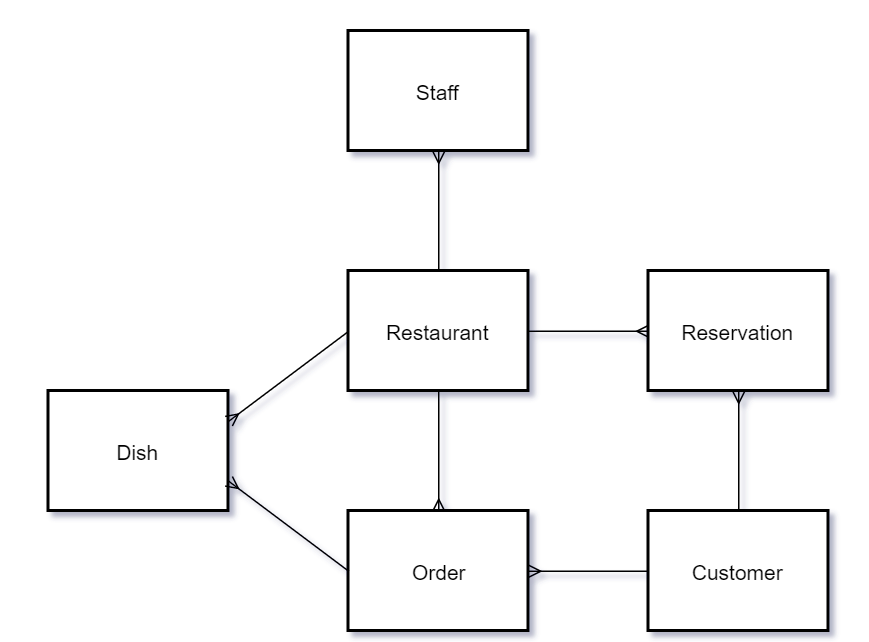
|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| * Staff account – username, password, email, first and second name * Reservations – full name, date and time, number of people dining, phone number, email * Orders – full name, phone number, address, dishes * Restaurant – name, address, phone number, maximum size, menu | * Searching for a restaurant – data from restaurant table should be sorted and outputted to customer * Summary of orders need to be displayed * Total price of orders needs to be calculated * Total number of people dining in the restaurant needs to be calculated | * Everything will be stored in the database tables * Notifications will be temporary * Orders will be temporary | * Staff – table layout, orders, reservations * Customer – reservation summary, order summary, menu and restaurant information * Notifications – the tasks that needs to be completed or orders |

## Validation

The information that the user will input should be validated so that the system works efficiently with no errors. The validation checks will also make sure that the user creates an account.

|  |  |  |  |
| --- | --- | --- | --- |
| Field name | Validation checks | Description | Error message |
| Username | 0 < Length ≤ 20 | Username must be of a sensible length | Username must not be over 20 characters in length. |
| Password | 0 < Length ≤ 20 | Password must be of a sensible length | Password must not be over 20 characters in length. |
| Email | 0 < Length ≤ 300  e.g. hello@example.com | Email must be of a sensible length and in email format | Invalid email / email must not be over 30 characters in length. |
| Date | dd/mm/yyyy  e.g. 01/01/2020 | Date must be in date format | Invalid date |
| Time | 00:00 AM/PM | Time must be in time format | Invalid time |
| No. of people dining | Data type: Positive integer e.g. 10 | Positive integer must be inputted | Invalid number |
| Phone number | 0 < Length ≤ 15  Data type: Positive integer  e.g. 12345678910 | Positive integer must be inputted, and phone number must be of a suitable length | Invalid phone number / phone number must not be over 15 characters in length |
| Maximum size | Data type: Positive integer e.g. 100 | Positive integer must be inputted | Invalid number |
| Price | Data type: Float  e.g. 19.99 | Positive float must be inputted | Invalid price |

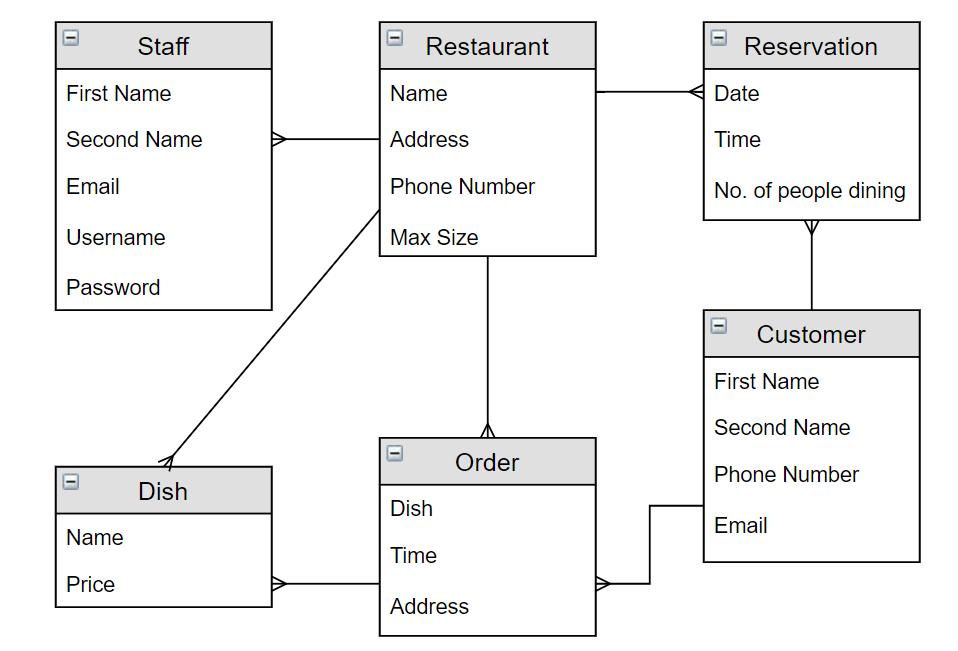
## Entity Relationship Diagram

This diagram shows the entities of the system and it will help me to create the tables for the database.

* A customer can make many reservations; therefore, it is a one-to-many relationship.
* A customer can make many orders; therefore, it is a one-to-many relationship.
* A restaurant will have many staff; therefore, it is a one-to-many relationship.
* A restaurant will have many dishes; therefore, it is a one-to-many relationship.
* A restaurant will have many orders; therefore, it is a one-to-many relationship.
* A restaurant will have many reservations which are made by the customers; therefore, it is a one-to-many relationship.
* An order will have many dishes; therefore, it is a one-to-many relationship.

## Entity Attribute Model

This diagram shows the attributes for each entity and this will help me understand the field names and field types for each one.



* The model ‘Reservations’ will contain data for the date and time of the reservation and the number of people dining.
* The model ‘Restaurant’ will contain data for the name and address of the restaurant, the phone number and the maximum size.
* The model ‘Customer’ will contain data for the first and second name of the customer, the phone number and the email.
* The model ‘Staff’ will contain data for the first and second name of the staff, the email and the username and password of the account that the staff will make.
* The model ‘Dish’ will contain data for the name of the dish and the price.
* The model ‘Order’ will contain data for the names of the dishes, the price of each dish, the time it has been ordered and the address of the customer.

## Normalisation

From the entity relationship diagram, the database tables will be ‘Restaurants’, ‘Customers’, ‘Staff’ and ‘Reservations’. All the fields in the tables will have their own unique ID because Python Django’s default database management system is SQLite and an ID is automatically created for each field and this will be the primary key. However, custom primary keys can be created.

1st Normal Form:

**tblReservation**(**reservationID**, date, time, no\_of\_people\_dining, **personID**)

**tblCustomer**(**customerID**, first\_name, second\_name, phone\_no, email, **reservationID**)

**tblRestaurant**(**restaurantID**, name, address, phone\_no, max\_size)

**tblStaff**(**staffID**, username, password, first\_name, second\_name, email, **restaurantID**)

**tblDish**(**dishID**, name, price, **restaurantID**)

**tblOrder**(**orderID**, time, address, **dishID**, **personID**)

The tables ‘dish’ and ‘order’ are separated because the staff can order dishes when managing tables from the home page and the customer can order dishes from the home page. Also, the menu of the restaurants will have to be saved in the database and in this case, a table for the dishes should be made.

2nd Normal Form/3rd Normal Form:

The tables will be in 2nd/3rd normal form if there are no repeating attributes and there are no partial dependencies. In the tables ‘customer’ and ‘staff’, the attributes ‘first\_name’ and ‘second\_name’, ‘phone\_no’ and ‘email’ are repeated. Therefore, the tables should be edited to separate these attributes.

**tblPerson(personID**, first\_name, second\_name, phone\_no, email)

**tblAccount(accountID**, username, password, **personID**)

Finally, these are the tables that will be created for the database in the system.

**tblReservation**(**reservationID**, date, time, no\_of\_people\_dining, **personID**)

**tblRestaurant**(**restaurantID**, name, address, phone\_no, max\_size)

**tblPerson(personID**, first\_name, second\_name, phone\_no, email)

**tblAccount(accountID**, username, password, **personID**, **restaurantID**)

**tblDish**(**dishID**, name, price, **restaurantID**)

**tblOrder**(**orderID**, time, address, **dishID**, **personID**)

## SQL Statements

First, I will need to create a table for each entity. These SQL statements define the table and the attributes.

Reservation table:

CREATE TABLE Reservation (

reservationID int() NOT NULL,

date date() NOT NULL,

time datetime() NOT NULL,

no\_of\_people\_dining int() NOT NULL,

personID int() NOT NULL,

PRIMARY KEY (reservationID),

FOREIGN KEY (personID)

);

Restaurant table:

CREATE TABLE Restaurant (

restaurantID int() NOT NULL,

name varchar() NOT NULL,

address varchar(),

phone\_no int(15),

max\_size int() NOT NULL,

PRIMARY KEY (restaurantID)

);

Person table:

CREATE TABLE Person (

personID int() NOT NULL,

first\_name varchar() NOT NULL,

second\_name varchar() NOT NULL,

phone\_no int(15),

email varchar(300) NOT NULL,

PRIMARY KEY (personID)

);

Account table:

CREATE TABLE Account (

accountID int() NOT NULL,

username varchar(20) NOT NULL,

password varchar(20) NOT NULL,

personID int() NOT NULL,

restaurantID int() NOT NULL,

PRIMARY KEY (accountID)

FOREIGN KEY (personID)

FOREIGN KEY (restaurantID)

);

Dish table:

CREATE TABLE Dish (

dishID int() NOT NULL,

name varchar() NOT NULL,

price float() NOT NULL,

restaurantID int() NOT NULL,

PRIMARY KEY (dishID),

FOREIGN KEY (restaurantID)

);

Order table:

CREATE TABLE Order (

orderID int() NOT NULL,

time datetime() NOT NULL,

address varchar() NOT NULL,

dishID int() NOT NULL,

personID int() NOT NULL,

PRIMARY KEY (orderID),

FOREIGN KEY (dishID),

FOREIGN KEY (personID)

);

When a user inputs data, the data needs to be saved into the table. These statements will show when a table can be updated. All the IDs will be auto incremented.

Booking for a table:

INSERT INTO Reservation(reservationID, date, time, no\_of\_people\_dining, personID)

VALUES(AUTO, ‘2020-01-01’, ‘10:30’, 2, AUTO)

Adding dish information:

INSERT INTO Dish(dishID, name, price)

VALUES(AUTO, “Chips”, 5.99)

Ordering food:

INSERT INTO Order(orderID, time, address, dishID, personID)

VALUES(AUTO, ‘10:30’, “123 Example Street”, AUTO, AUTO)

Adding a person:

INSERT INTO Person(personID, first\_name, second\_name, phone\_no, email, accountID)

VALUES(AUTO, “Hello”, “Smith”, 12345678910, ‘abc@example.com’, AUTO)

Registering for an account:

INSERT INTO Account(accountID, username, password, personID, restaurantID)

VALUES(AUTO, “abc”, “123”, AUTO, AUTO)

Adding restaurant information:

INSERT INTO Restaurant(restaurantID, name, address, phone\_no, max\_size)

VALUES(AUTO, “Restaurant”, “123 Abc Street”, 12345678910, 50)

Editing dish information:

UPDATE Dish

SET name = “Chips”, price = 5.99

WHERE dishID = 123

Editing person information:

UPDATE Person

SET first\_name = “Hello”, second\_name = “Smith”, phone\_no = 12345678910, email = ‘abc@example.com’

WHERE personID = 123

Editing account information:

UPDATE Account

SET username = ‘abc’, password = ‘123’

WHERE accountID = 123

Editing restaurant information:

UPDATE Restaurant

SET name = “Restaurant”, address = “123 Example Street”, phone\_no = 12345678910, max\_size = 100

WHERE restaurantID = 123

Aggregate SQL functions can also be used to calculate the total price of the dishes which will then be returned and displayed to the user in the order summary.

Calculating total price of dishes:

SELECT SUM(price)

FROM Order

JOIN Dishes ON Order.personID = Dish.personID

## Data Dictionaries

I will create a data dictionary for every table to help me understand what type of data is being used and how it is going to be validated.

Reservation table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| reservationID | integer | 123 | To make the identifier unique |
| date | date | 2020-01-01 | Date of reservation |
| time | datetime | 14:30 | Time of reservation |
| no\_of\_people\_dining | integer | 2 | How many people will be dining |

Person table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| personID | integer | 123 | To make the identifier unique |
| first\_name | varchar | Hello | Customer’s first name |
| second\_name | varchar | Smith | Customer’s second name |
| phone\_no | integer | 12345678910 | Customer’s phone number |
| email | email | abc@example.com | Customer’s email |

Account table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| accountID | integer | 123 | To make the identifier unique |
| username | varchar | abc | The user’s username |
| password | varchar | 123 | The user’s password |

Restaurant table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| restaurantID | integer | 123 | To make the identifier unique |
| name | varchar | Restaurant | Name of restaurant |
| address | varchar | 123 Example Street | Address of restaurant |
| phone\_no | integer | 12345678910 | Restaurant’s phone number |
| max\_size | integer | 50 | The maximum size of people that can dine at the restaurant |

Dish table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| dishID | integer | 123 | To make the identifier unique |
| name | varchar | chips | Name of dish |
| price | float | 2.99 | Price of dish |

Order table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Sample data** | **Description** |
| orderID | integer | 123 | To make the identifier unique |
| time | datetime | 14:30 | Time of order placed |
| address | varchar | 123 Example Street | Address for order to be delivered to |

## OOP Classes

In Python Django, OOP is used to make models, which are database tables and each attribute is a field in the table. From the entity relationship diagrams, I can create models for each table. Each model is a subclass of ‘django.db.models.Model’.

Reservation model:

Class Reservation(models.Model):

reservationID = models.AutoField(primary\_key=True)

date\_of\_booking = models.DateField()

time\_of\_booking = models.DateTimeField(unique=True)

no\_of\_people = models.PositiveIntegerField()

Person = models.ForeignKey(Person, on\_delete=models.CASCADE)

Person model:

class Person(models.Model):

personID = models.AutoField(primary\_key=True)

first\_name = models.CharField(max\_length=50)

second\_name = models.CharField(max\_length=50)

phone\_no = models.PositiveIntegerField()

email = models.EmailField(max\_length=300)

Account model:

class Account(models.Model):

accountID = models.AutoField(primary\_key=True)

username = models.CharField(max\_length=20)

password = models.CharField(max\_length=20)

Person = models.ForeignKey(Person, on\_delete=models.CASCADE)

Restaurant = models.ForeignKey(Restaurant, on\_delete=models.CASCADE)

Restaurant model:

class Restaurant(models.Model):

restaurantID = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=300)

address = models.TextField()

phone\_no = models.PositiveIntegerField()

max\_size = models.PositiveIntegerField()

Dish model:

class Dish(models.Model):

dishID = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=200)

price = models.DecimalField(max\_digits=None, decimal\_places=2)

Restaurant = models.ForeignKey(Restaurant,

on\_delete=models.CASCADE)

Order model:

class Order(models.Model):

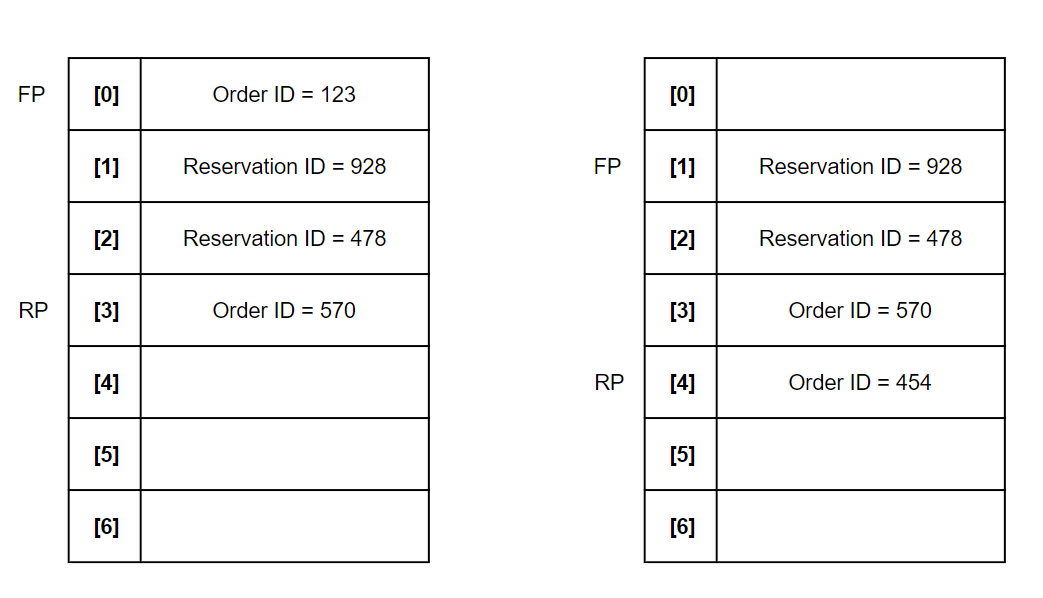
orderID = models.AutoField(primary\_key=True)

time = models.DateTimeField()

address = models.TextField()

Dish = models.ForeignKey(Dish, on\_delete=models.CASCADE)

## Data structures

Staff will manage reservations on the calendar in the staff home page. To manage the reservations on the calendar, the reservations will be ordered in date and time order. Also, the notifications that will displayed in the notification panel must be ordered in date and time order. Therefore, a queue should be implemented in this case. A queue uses a FIFO (first-in-first-out) structure which means the first reservation that is added to the queue will need to be the first one to be removed from the queue. I will use a circular queue because the changes will be dynamic, and the use of memory needs to be efficient. An example of implementing this structure is displayed below:

The front and rear pointers will move when an item is added or removed, and this is useful for when managing notifications for example because staff would need to view and delete notifications in real time and this needs to be done in an efficient way.

Implementing a circular queue in Python:

class CircularQueue:

def \_\_init\_\_(self):

self.queue = list()

self.FrontPointer = 0

self.RearPointer = 0

self.QueueSize = 8

#Add to queue

def enqueue(self, data):

if self.size() == self.QueueSize – 1:

return(“Queue full”)

self.queue.append(data)

self.RearPointer = (self.RearPointer + 1) % self.QueueSize

return(“Item added”)

#Remove first item from queue

def dequeue(self):

if self.size() == 0:

return(“Queue empty”)

data = self.queue[self.FrontPointer]

self.FrontPointer = (self.FrontPointer + 1) % self.QueueSize

return data

#Calculating size of queue

def size(self):

if self.RearPointer >= self.FrontPointer:

return(self.RearPointer – self.FrontPointer)

return(self.QueueSize – (self.FrontPointer – self.RearPointer))

## Algorithms

When the customer is searching up for a restaurant, a list of the restaurants should be displayed in order. The customer will search up the restaurant by name so the restaurants should be displayed by alphabetical order. Therefore, I will implement merge sort in this case.

Sorting the list by alphabetical order:

#List is split into two lists

def merge(left\_list, right\_list):

sortedlist = []

left\_list\_size = 0

right\_list\_size = 0

x = 0

y = 0

#Check if list size is small

if x < left\_list\_size and y < right\_list\_size:

#Check if first item in first list is smaller or equal to first item in second list

#First item in first list is smaller

if left\_list[x] <= right\_list[y]:

sortedlist.append(left\_list[x])

x = x + 1

else:

#First item in second list is smaller

sortedlist.append(right\_list[y])

y = y + 1

else if x == left\_list\_size:

#Add items from second list when first list is finished

sortedlist.append(right\_list[y])

y = y + 1

else if y == right\_list\_size:

#Add items from first list when second list is finished

sortedlist.append(left\_list[x])

x = x + 1

return sortedlist

def merge\_sort(list\_to\_be\_sorted):

#Returns list if length of list is 1

if len(list\_to\_be\_sorted) <= 1:

return list\_to\_be\_sorted

#Get midpoint of the list to be sorted

midpoint = len(list\_to\_be\_sorted) / 2

#List is split

left\_list = merge\_sort(list\_to\_sorted[:midpoint])

right\_list = merge\_sort(list\_to\_be\_sorted[midpoint:])

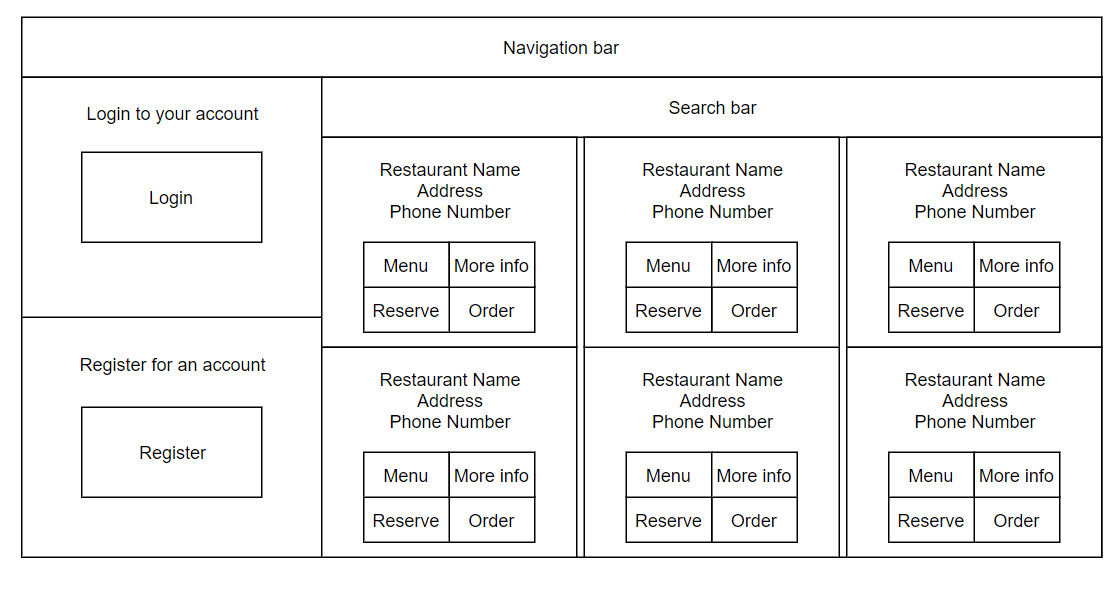
return merge(left\_list, right\_list)

## Interface Design

From the analysis, each page should reflect on the objectives. The interface design for each page will need to be presented in a way where the users can do the tasks that they need to do, and each page should have instructions which can guide them to where they want to go or what they would like to do. The elements of each page should be presented clearly since this system aims to help users do their tasks efficiently. The colour scheme needs to be bright and clear for the user to see the elements.

Start Page

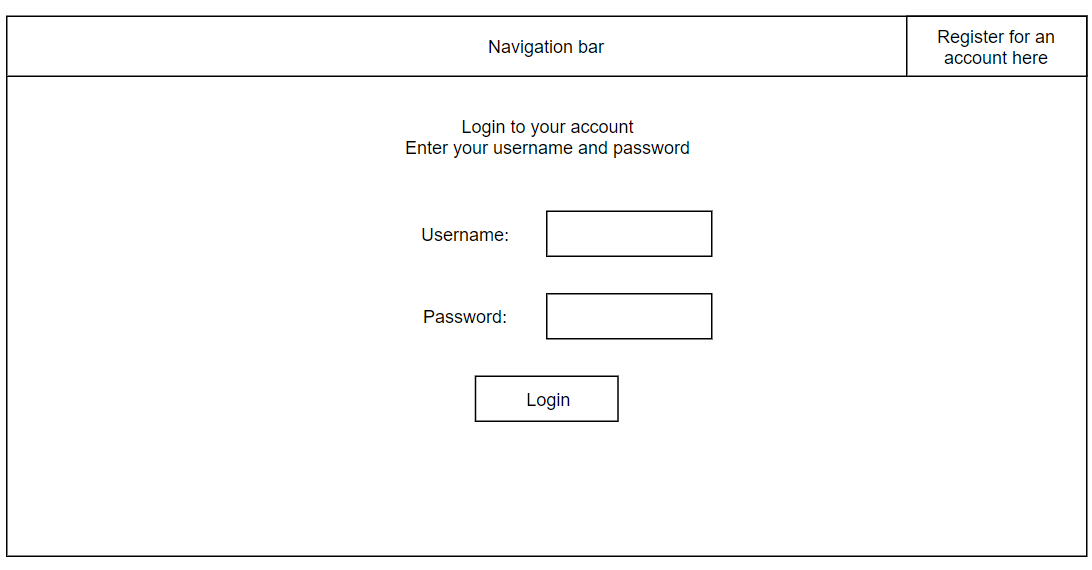
* A section that will have buttons for the user to login or register – this will take redirect them to the login page or the register page.
* A search bar for the customer to search up the restaurant – the customer will search up the name of the restaurant and find information about it.



The start page will have a navigation bar which can take them to the login page or the register page. On the left side, the user can go to the login page by clicking on the login button or the user can go to the register page by clicking on the register page. On the right, the user can search for a restaurant by typing in the name of the restaurant on the search bar. A grid of restaurants will be shown like in the layout above. Each restaurant will have information displayed such as the name, the address and the phone number. In each grid, there are buttons that the user can click. The ‘menu’ button will display the menu, the ‘more info’ button will display more information of the restaurant, the ‘reserve’ button will allow the user to reserve a table and the ‘order’ button will allow the user to order food.

Login Page

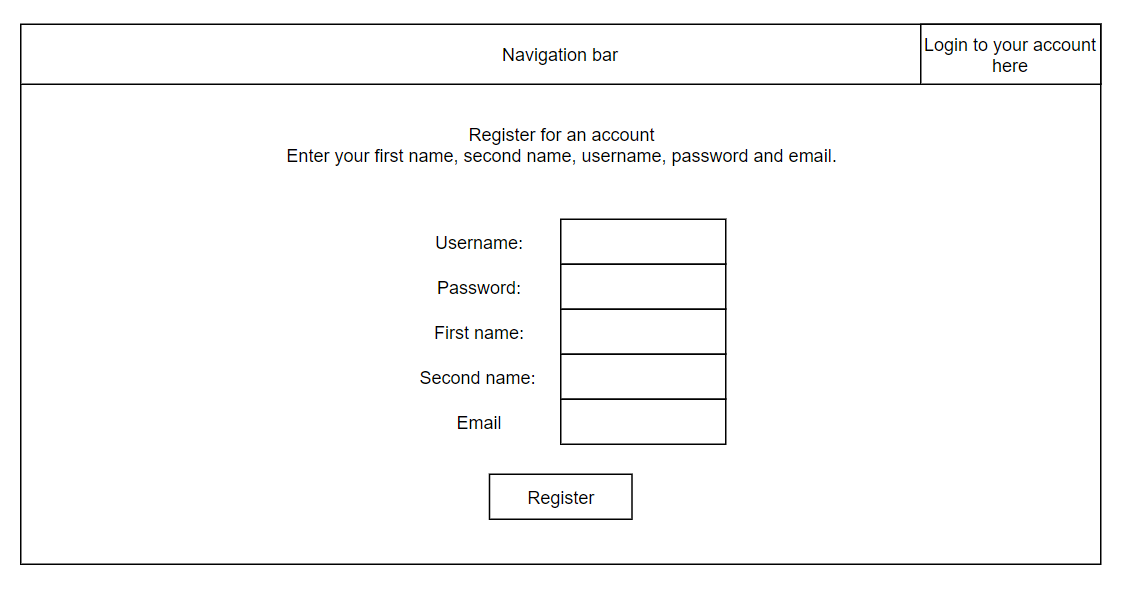
* A section for the user to input their username and password – the form should be submitted, and the database should be updated.
* A button that will redirect them to the start page – for the user to go back to the start page.
* A button that will redirect them to the register page – this is if the user has not created an account yet.
* Instructions to help the user where to put their username and password in or whether they are looking for something else.



The login page will also have a navigation bar at the top of the page. On the top right corner, the user can choose to register for an account which will take them to the register page. The user will enter their username and password and click the login button to login. The instructions will be at the top to tell the user what to do. Clicking the login button will take them to the home page.

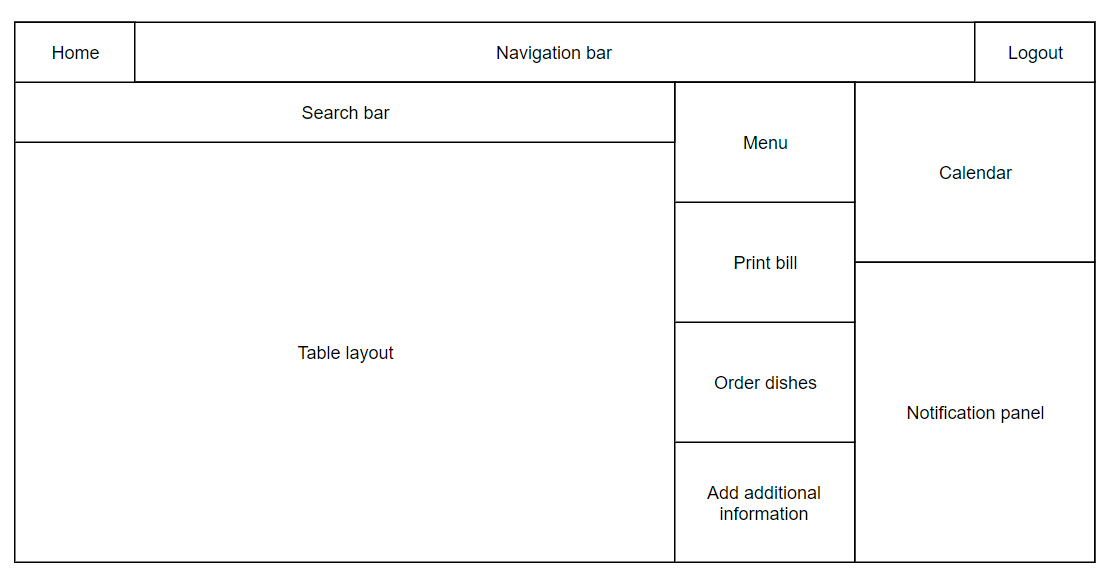
Register Page

* A section for the user to input their username, password, first name, second name and email – the form should be submitted, and the database should be updated.
* After they have submitted their form, this should login them in straight away and redirect them automatically to the home page.
* A button that will redirect them to the start page – in case they are not at the page that they need to be on.
* Instructions to help the user how to create an account.



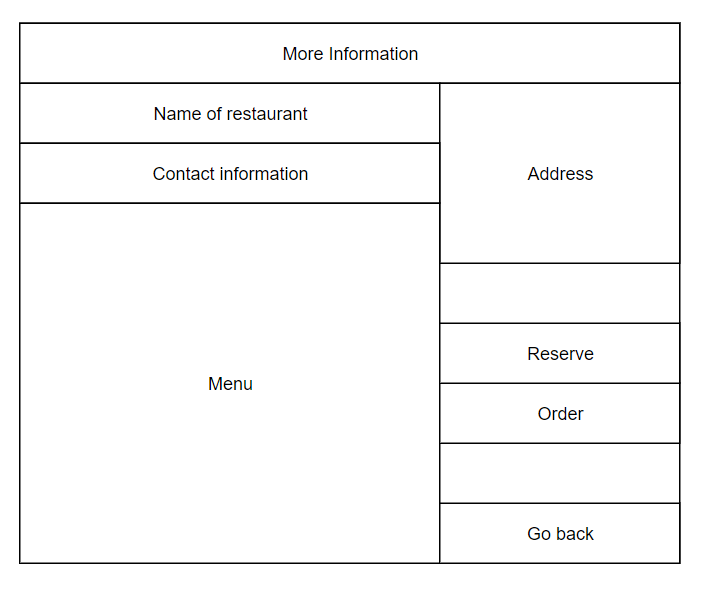
The register page will have a navigation bar at the top of the page and on the top right corner, the user can choose to login to their account, if they have already made an account. This will take them to the login page. The user will fill out the form below the instructions and click on the register button to register for an account and this will automatically redirect them to the home page if the information inputted is validated.

Home Page

* A section that displays the table layout of the restaurant – the user should be able to label the tables and manage tables easily.
* A section for notifications – this will update the user on any work that they need to do, and this will include timestamps and the tables that needs assistance.
* Notifications that should consist of the tables that needs assistance and any takeaway orders for example.
* A calendar – the staff should be able to manage reservations and edit information if necessary.
* A section to order dishes for the table that they are managing – the option to select a dish and a summary of the orders.
* A section to write additional notes for the customer or the order.
* A button to separate the main dishes from the starters or to separate the dishes so that the staff would know which dishes should come in order.
* A search bar for the staff to search up the dishes or the customer’s information.

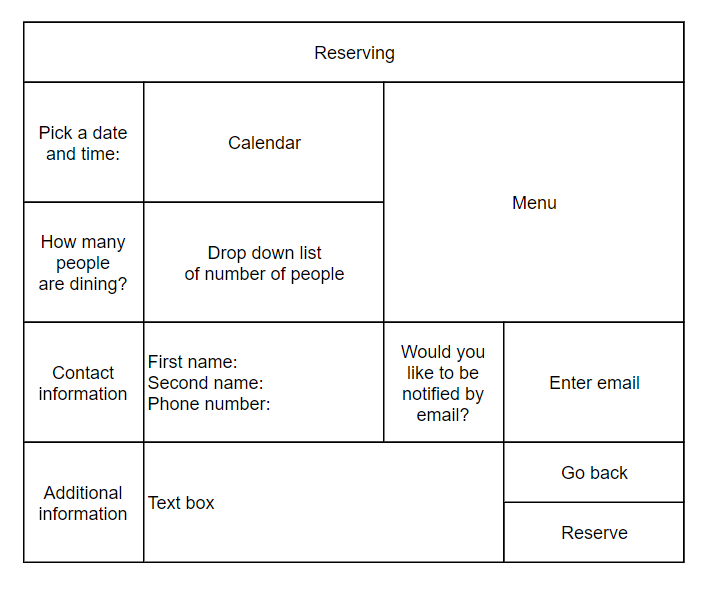
When the user has clicked login or has successfully registered an account, they will be redirected to the home page. The home page will have a navigation bar at the top which will include buttons for them to go back to the home page or to logout. There is a search bar which will allow the user to search for a dish or a customer’ name. The table layout must be clear and big for the user to attend to the customers quickly. The user can manage tables from the table layout. There are buttons for the menu, to print out the bill, to order dishes when a table is selected and to add additional information for a customer or an order. On the right side, there is a calendar which will allow the user to manage reservations and there is a notification panel which will allow the user to view takeaway orders that are made by the customers.

Restaurant Information

When the user clicks on ‘More info’ when they search up a restaurant, a box should pop up and the user should see this. The name of restaurant, the contact information and the restaurant’s information should be displayed. The menu should also be displayed clearly for the customer to get an idea of what dishes the restaurant are serving. On the right side, there should be buttons for the customer to reserve a table or to order dishes. There should also be a button for the user to go back to the home page so that they can search up a restaurant.

Reserving

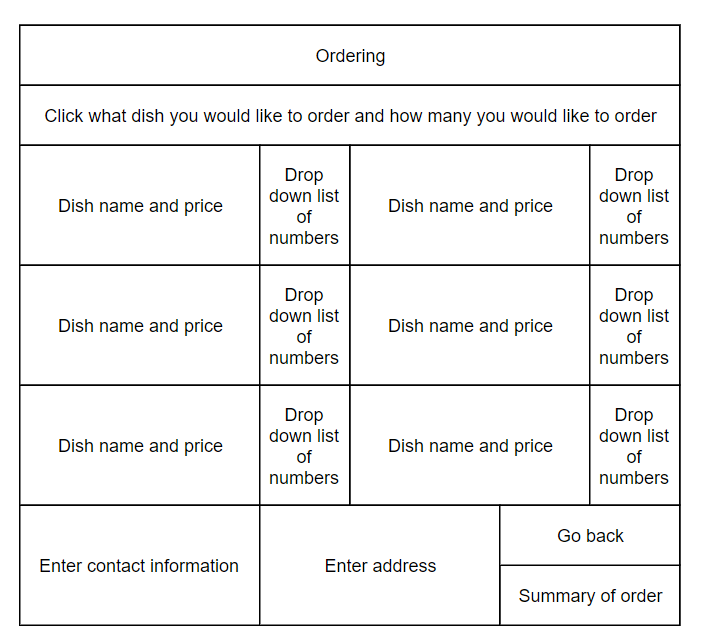
* A calendar for the customer to book reservations – the customer can pick a date and input information such as the number of people dining, the dining area, an option to be notified, a menu and any other additional information which may include dietary requirements and alternative times.
* The restaurant information should at least include the name and type of the restaurant, the menu and the address of the restaurant and most importantly, whether it is full or not.



When the customer clicks on reserve, the customer will fill out this form. They will pick a date and time from the calendar. A drop-down list will be used for the number of people that are dining. When the customer has finished filling out the form, they will click the reserve button and the information should be sent to the staff on the home page. If the user has chosen to be notified by email, an email should be sent to the user to remind the user of the reservation.

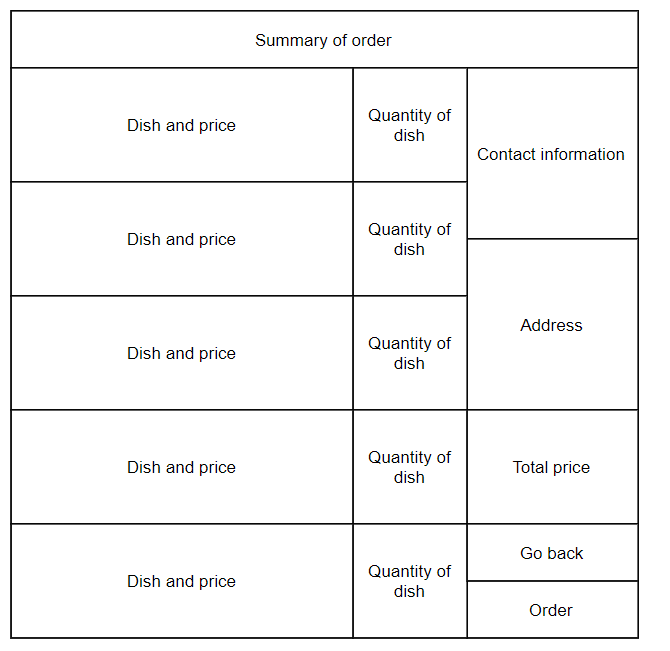
Ordering

* The customer should also be able to order takeaway from the restaurant by the menu. Customer should be able to choose the dishes easily and provide contact and delivery information.
* A summary of the orders should be displayed clearly.



When the customer clicks the order button, a list of dishes with their prices should be displayed. The list of dishes should reflect the restaurant’s menu. Once the customer has decided which dishes to order, the customer should enter their contact information and their address. Then the customer clicks on ‘Summary of order’ button to see the summary of their order.

Summary of order



In the order summary, a list of dishes with their prices should be displayed with the quantity displayed next to it. On the right-hand side, the information that the customer has inputted should be displayed - their contact information and their address. Also, the total price calculated should be displayed. Once the customer has checked everything, they can choose to order or go back. The information should be sent to the staff and displayed on the home page.

# Technical Solution

For my restaurant booking system, I have used Python Django, which is a web framework that uses Python. HTML and CSS is used for the layout and style of the web pages and it is used for the server-side scripting.

For each objective, I will show the different techniques or algorithms that are used.

### Start / ordering

* A section that will have buttons for the user to login or register – this will take redirect them to the login page or the register page.

**Buttons in HTML are used for the user to click to go to the login or register page. Python is used to display the pages using HTTP Response in the ‘views.py’ file and it is also used for URL configuration so that the views can be displayed in the correct order.**

* A search bar for the customer to search up the restaurant – the customer will search up the name of the restaurant and find information about it.

**The search bar will be displayed using HTML and CSS. The searching function will use API that will allow me to get a list of objects from the databases and sort them out into order.**

* The restaurant information should at least include the name and type of the restaurant, the menu and the address of the restaurant and most importantly, whether it is full or not.

**The user will input the information using HTML forms and the form data will be sent using the POST method in the ‘views.py’ file. The ‘models.py’ file will contain models made from OOP that will store data into the database.**

* A calendar for the customer to book reservations – the customer can pick a date and input information such as the number of people dining, the dining area, an option to be notified, a menu and any other additional information which may include dietary requirements and alternative times.

**HTML forms will display the form for the model. ‘ModelForm’ is used to create a form that is linked to a model. The form data that the user inputs is used to create an object that will be saved into the database.**

* The customer should also be able to order takeaway from the restaurant by the menu. Customer should be able to choose the dishes easily and provide contact and delivery information.

**HTML forms and models are used.**

* A summary of the orders should be displayed clearly.

**The form data should be sent, and HTML is used to display the orders and SQL queries using Python is used to retrieve the objects so that the dishes can be displayed.**

### Login

The system must have a login page that will then redirect them to the home page. This will include:

* A section for the user to input their username and password – the form should be submitted, and the database should be updated.

**HTML forms and ‘ModelForm’ is used to display the forms to the user and allow the user to input information.**

* A button that will redirect them to the start page – for the user to go back to the start page.

**A button is displayed using HTML and the user should click on this when they want to go back to the start page.**

* A button that will redirect them to the register page – this is if the user has not created an account yet.

**A button is displayed using HTML and the user should click on this when they want to go to the register page.**

* Instructions to help the user where to put their username and password in or whether they are looking for something else.

**Instructions are displayed using HTML and Python.**

### Register

The system must have a register page that will then redirect them to the home page. This will include:

* A section for the user to input their username, password, first name, second name and email – the form should be submitted, and the database should be updated.

**HTML forms and ‘ModelForm’ is used to display the form and allow the user to input information. The form data is saved, and an object is created.**

* After they have submitted their form, this should login them in straight away and redirect them automatically to the home page.

**The form data is sent, and the user should be automatically signed into the home page using POST request.**

* A button that will redirect them to the start page – in case they are not at the page that they need to be on.

**A button is displayed using HTML and the user should click on this when they want to go back to the start page.**

* Instructions to help the user how to create an account.

**Instructions are displayed using HTML and Python.**

### Main

The system must have a home page that will display all the features for the staff to use for managing their work. This will include:

* A section that displays the table layout of the restaurant – the user should be able to label the tables and manage tables easily.
* A section for notifications – this will update the user on any work that they need to do, and this will include timestamps and the tables that needs assistance.  
  **A queue structure will be used to sort the notifications out by their status and they can be managed with the data structure. They can be added to the queue and removed from the queue.**
* Notifications that should consist of the tables that needs assistance and any takeaway orders for example.

**Notifications will be displayed and managed using the queue structure.**

* A calendar – the staff should be able to manage reservations and edit information if necessary.
* A section to order dishes for the table that they are managing – the option to select a dish and a summary of the orders.

**Objects from the ‘Dish’ model should be retrieved and displayed using HTML and Python.**

* A section to write additional notes for the customer or the order.  
  **A HTML form is used to allow the user to input information for the customer or for the order which can be saved into the database.**
* A button to separate the main dishes from the starters or to separate the dishes so that the staff would know which dishes should come in order.  
  **A HTML button is used and when the user clicks on this, it should separate the dishes.**
* A search bar for the staff to search up the dishes or the customer’s information.  
  **HTML and CSS is used to display the search bar and the searching function will use API to retrieve the objects from the database and sort it out in order.**

For each algorithm or technique, I will show what group it is placed in.

|  |  |  |
| --- | --- | --- |
| Algorithm / technique | Description | Group |
| OOP Models | Models contain data for each object created | B |
| Server-side scripting using request and response objects | HttpRequest and HttpResponse objects are used to display views | A / B |
| Single table SQL | Each model is a database table | B |
| SQL functions | Raw SQL queries can be executed or a QuerySet is used | A / B |
| Queue | Notifications are added to the queue and can be popped out of the queue based on their status | A |

### views.py

from django.shortcuts import render

from django.http import HttpResponse, HttpResponseRedirect

from bookings.models import \*

import datetime

#Start Page

def start(request):

restaurants = Restaurant.objects.all()

context = {'restaurants' : restaurants}

return render(request, 'bookings/start.html', context)

#Staff Login Page

def staffloginscreen(request):

loginform = LoginStaffAccountForm(request.POST)

return render(request, 'bookings/stafflogin.html', {'form' : loginform})

#Staff Register Page

def staffregister(request):

registerstaffaccountform = RegisterStaffAccountForm(request.POST)

registerstaffForm = RegisterStaffForm(request.POST)

#Validation

if registerstaffForm.is\_valid() and registerstaffaccountform.is\_valid() :

Account = registerstaffaccountform.save()

Person = registerstaffForm.save()

else:

registerstaffaccountform = RegisterStaffAccountForm()

registerstaffForm = RegisterStaffForm()

return render(request, 'bookings/staffregister.html', {'registerstaffaccountform' : registerstaffaccountform,

'registerstaffForm' : registerstaffForm})

#Staff Home

def staffhome(request):

return render(request, 'bookings/staffhome.html',)

#Reserve Page

def reserve(request):

reserveform = ReserveForm(request.POST)

#Validation

if reserveform.is\_valid():

Reservation = reserveform.save()

else:

reserveform = ReserveForm()

return render(request, 'bookings/reserve.html', {'form' : reserveform})

### models.py

from django.db import models

from django.forms import ModelForm

from django import forms

#Restaurant Model

class Restaurant(models.Model):

restaurantID = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=300)

address = models.TextField()

phone\_no = models.PositiveIntegerField()

max\_size = models.PositiveIntegerField()

#Display information in string format

def \_\_str\_\_(self):

return "{}, {}, {}", self.name, self.address, self.phone\_no

#Person Model

class Person(models.Model):

personID = models.AutoField(primary\_key=True)

first\_name = models.CharField(max\_length=50)

second\_name = models.CharField(max\_length=50)

phone\_no = models.PositiveIntegerField()

email = models.EmailField(max\_length=300)

#Display information in string format

def \_\_str\_\_(self):

return "{}, {}, {}, {}", self.first\_name, self.second\_name, self.phone\_no, self.email

#Account Model

class Account(models.Model):

accountID = models.AutoField(primary\_key=True)

username = models.CharField(max\_length=20)

password = models.CharField(max\_length=20)

Person = models.ForeignKey(Person, on\_delete=models.CASCADE)

Restaurant = models.ForeignKey(Restaurant, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return self.username

#Reservation Model

class Reservation(models.Model):

reservationID = models.AutoField(primary\_key=True)

no\_of\_people = models.PositiveIntegerField()

date\_of\_booking = models.DateField()

time\_of\_booking = models.TimeField(unique=True)

#Display information in string format

def \_\_str\_\_(self):

return "{}, {}, {}", self.no\_of\_people, self.date\_of\_booking, self.time\_of\_booking

#Dish Model

class Dish(models.Model):

dishID = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=200)

price = models.DecimalField(max\_digits=5, decimal\_places=2)

#Order Model

class Order(models.Model):

orderID = models.AutoField(primary\_key=True)

time = models.DateTimeField()

address = models.TextField()

Dish = models.ForeignKey(Dish, on\_delete=models.CASCADE)

#Display information in string format

def \_\_str\_\_(self):

return "{}, {}", self.time, self.address

#Form to reserve with Reservation Model

class ReserveForm(ModelForm):

class Meta:

model = Reservation

fields = ['no\_of\_people', 'date\_of\_booking', 'time\_of\_booking']

widgets = { 'no\_of\_people' : forms.NumberInput(attrs={'placeholder' : 'Number of people dining'}),

'date\_of\_booking' : forms.DateInput(attrs={'placeholder' : 'Date of booking'}),

'time\_of\_booking' : forms.DateTimeInput(attrs={'placeholder' : 'Time of booking'})

}

#Form for customer's input when reserving

class CustomerForm(ModelForm):

class Meta:

model = Person

fields = ['first\_name', 'second\_name', 'phone\_no', 'email']

#Form to register staff with Account Model

class RegisterStaffAccountForm(ModelForm):

password = forms.CharField(max\_length=20, widget=forms.PasswordInput)

class Meta:

model = Account

fields = ['username', 'password']

widgets = { 'username' : forms.TextInput(attrs={'placeholder' : 'Username'}),

'password' : forms.PasswordInput(attrs={'placeholder' : 'Password'})

}

#Form to register staff with Person Model

class RegisterStaffForm(ModelForm):

email = forms.EmailField(max\_length=300, widget=forms.EmailInput)

class Meta:

model = Person

fields = ['first\_name', 'second\_name', 'phone\_no', 'email']

widgets = { 'first\_name' : forms.TextInput(attrs={'placeholder' : 'First Name'}),

'second\_name' : forms.TextInput(attrs={'placeholder' : 'Second Name'}),

'phone\_no' : forms.NumberInput(attrs={'placeholder' : 'Phone Number'}),

'email' : forms.EmailInput(attrs={'placeholder' : 'Email'})

}

#Form to login staff

class LoginStaffAccountForm(ModelForm):

password = forms.CharField(max\_length=20, widget=forms.PasswordInput)

class Meta:

model = Account

fields = ['username', 'password']

widgets = { 'username' : forms.TextInput(attrs={'placeholder' : 'Username'}),

'password' : forms.PasswordInput(attrs={'placeholder' : 'Password'})

}