Design Stage

Group 17

Students:

100720666

100706099

100682643

Contents

[Design Stage 3](#_Toc197683162)

[Fully Dressed Use case 3](#_Toc197683163)

[Customer Reservation 3](#_Toc197683164)

[Waitstaff Reservation 4](#_Toc197683165)

[Kitchen Order 5](#_Toc197683166)

[Staff Order 6](#_Toc197683167)

[Inventory Management 7](#_Toc197683168)

[Staff Scheduling for Managers 9](#_Toc197683169)

[Staff time-off requests 10](#_Toc197683170)

[System Overview 11](#_Toc197683171)

[Sequence Diagram (UML) 12](#_Toc197683172)

[Inventory Management 12](#_Toc197683173)

[Orders Management (Waitstaff) 13](#_Toc197683174)

[Orders Management (Kitchen Staff) 14](#_Toc197683175)

[Scheduling: Staff Scheduling (Managers Side) 15](#_Toc197683176)

[Scheduling: Staff Time off Request 16](#_Toc197683177)

[Reservation management: Customer 17](#_Toc197683178)

[Reservation management: Waitstaff 18](#_Toc197683179)

[Naming Convention 19](#_Toc197683180)

[Partial Class Diagrams 19](#_Toc197683181)

[Orders Management 20](#_Toc197683182)

[Scheduling: Staff Time off Request 21](#_Toc197683183)

[Scheduling: Scheduling for Managers 23](#_Toc197683184)

[Inventory Management 23](#_Toc197683185)

[**Reservation: Customer Reservation** 24](#_Toc197683186)

[**Reservation: Waitstaff Reservation** 24](#_Toc197683187)

[Full Class Diagram 25](#_Toc197683188)

[Component Diagram 25](#_Toc197683189)

[Functional Requirements to Design Artifacts 26](#_Toc197683190)

[Architectural Model – Client-Server Architecture 29](#_Toc197683191)

[Data model 30](#_Toc197683192)

[Design Patterns – Façade 31](#_Toc197683193)

[Implementation Stage 33](#_Toc197683194)

# Design Stage

## Fully Dressed Use case

In this stage, we transformed our use case into a fully dressed use case by detailing the preconditions, special requirements, technology and data variation list and frequency of occurrence. This stage was beneficial for us as it allowed us to gain more valuable context of our system by taking into consideration the extra conditions and requirements. All the associated functions are still linked to ensure a comprehensive understanding of the system.

### Customer Reservation

**Primary actor**: customer

**Supporting actor**: Waitstaff (For modifications of any current reservations)

**Stakeholders and interest**: (**Customers**) Want a smooth and user-friendly system to reserve tables and views real time availability. (**Waitstaff**) A user-friendly system which is easy to navigate around customer reservation and the possibility of making any necessary changes.

**Preconditions:** The customer must have access to an internet connected device and the reservation system, must be online and functional.

**Goal**: Customers will be able to use the system to book tables, see availability and receive booking confirmations and reminders.

**Main success scenario**: The customer opens the booking reservation page to make a booking where they are exposed to the real time availability. The then proceed to pick a date, time and table size for their booking along with their personal details. After confirming all this information, they then receive an email which confirms their booking.

**Alternative Scenario**: The customer has made a reservation but at the last minute, they needed to make some changes. They access the system to view their reservation, and they decide to modify it based on the changes which was needed to be made. After making the changes and updating it, they receive a confirmation email about the changes which have been made.

**Special Requirements:** It’s important to ensure that the system is compatible with various devices and operation systems and that real-time availability bookings are accurate to ensure double booking are prevented.

**Technology and data variation list:** This will need multiple input methods such as touchpads or keyboards.

**Frequency of occurrence:** Systems will be used during the days of the week in which the restaurant is open, and during peak periods.

**Constraints**: There are a few constraints which we may face concerning this particular use case. Some of the technical restraints are the responsiveness and compatibility between all the devices regardless of their size. Real-time availability must also be in sync to avoid any confusions. There are also business constraints such as capacity limits to ensure there is no overbooking in the system. Operational constraints may occur such as a system down time, which if it happens during working hours customers will temporarily not be able to book any reservations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Functional requirements | Attributes |
| Customer reservation | Customer | Waitstaff | **(M) 5F1-2** View Slots with Availability  **(M) 5F3-4** Book Reservation Slot  **(M) 5F5** Receive Confirmation Notification  **(M) 5F6-8** Send Reminders  **(S) 5F9-10** Modify Booking  **(C)** Customer Account | Attributes: Reservation ID, Customer Name, Table Number, Date and Time, Number of Guests, Status |

### Waitstaff Reservation

**Primary actor**: Reservation Staff

**Stakeholders and interest**: (**Waitstaff**) Require a user-friendly system which allows them to easy modify reservations. They would like it to be compatible with the existing hardware as this will be more convenient. (**Customers**) Would like to be aware and notified of the changes which have been made.

**Preconditions:** Reservation data must be accurate and waitstaff must be authorised to access and modify the bookings

**Goal**: The staff should be able to view the booking which have been made and have the option to modify the bookings to ensure a smooth operation of the business.

**Scenario:** The staff members start their shift and logs into the management system to view the current bookings for the day. The staff identifies that one of the tables has been damaged. They then go onto their management system and view who reserved the table so that they can relocate them. When this is done, the customer will receive an update of the changes which has been made.

**Constraints:** The technical constraints used in customer table reservation use can also be used here. Ensuring that the system is compatible with all devices and that the real-time availability is in sync ton avoid confusion. The modification of reservation will only be available to authorised staff members. At certain times of the year, demand may be high and there may be a lot of reservations being made. There for capacity limits are enforced. Changes within the business can be made which will therefore allow the limit to be increased, but during these busy periods, ensuring the performance of the system isn’t degraded will allow for a smooth business operation.

**Special Requirements:** Only authorised staff can modify the bookings and conflicts must be prevented through real-time synchronisation.

**Technology and data variation list:** Desktops terminals and tables for the waitstaff to use. Notifications via emails and SMS.

**Frequency of occurrence:** This must be available daily during the restaurant opening hours.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary Actor | Supporting actor | Functional requirements | Attributes |
| Staff reservation | Waitstaff | Customer | **(M) 1F1-2** View Reservations  **(M)** **1F3-4** Modify Reservation  **(M)**: Compatibility with tablets/terminals. | Reservation ID, Customer Name, Table No., Date/Time, Number of Guests, Status. |

### Kitchen Order

**Primary actor:** Kitchen staff

**Supporting actor**: Waitstaff

**Stakeholders and interests**: (**Kitchen Staff**) Want a user-friendly system and updates on real time orders and their statuses to ensure preparation and the delivery of the food is timely. (**Waitstaff**) Would like to view the statuses of the order to ensure that once they are prepared, they can be delivered to the tables. (**Inventory workers**) Would like a user-friendly system to make sure they are able to keep track of the stock so they can restock when necessary.

**Preconditions:** The kitchen staff must be logged into the order management system.

**Goal**: Kitchen staff should be able to view real time orders and should have the ability to update the status on It.

**Scenario**: The kitchen staff has changed into their work uniform and have started their shift. After 10 mins, they receive and notification request for an order which they then accept. The staff the accept the request which then shows on the system as “In progress.” They have now finished the order and procced to update the status as “Ready” which is then made visible to the staff members so that they can come and collect the food.

**Constraints**: System usability should be user-friendly otherwise; it may require extensive training for the staff members. This means that updating order statuses and tracking inventory should be easy to do. The system should be compatible with the existing hardware. If this is not the case, there may be additional costs to ensure a smooth integration. The notifications of order statues between floor staff and kitchen staff should be efficient to prevent any backlog and delays.

**Special Requirements:** The system must support real-time updates and notifications.

**Technology and data variation list:** Kitchen display screens. Real time notification to waitstaff and kitchen staff terminals.

**Frequency of occurrence:** System must be continual throughout service hours.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Functional Requirements | Attributes |
| Kitchen Order | Kitchen Staff | Waitstaff | **(M) 2F5** Display Orders with Details  **(M) 2F6** Update Order Progress Status | Order ID, Items, Status, Preparation Time. |

### Staff Order

**Primary actor** - Waitstaff

**Supporting actor –** Kitchen staff

**Stakeholders and interests – (Waitstaff)** Would like a user-friendly system to send kitchen staff the order of the reserved customers, and they want to view orders with progress status. **(Kitchen staff)** A user-friendly system which allow them to view order coming in from staff, and to update their statuses along the way.

**Preconditions:** Order management system must be operational and accessible. Staff members must be trained to use the system.

**Goal –** To be able to input into the system the staff orders and send it off to the kitchen staff.

**Scenario –** The customer arrives at the restaurant; their reservation has been confirmed, and they are being taken to their table. A staff member approaches them, greets them, then proceeds to ask what they would like to order. After noting it all down in the system, they then send it off to the kitchen staff for them to start preparing.

**Constraint** – The system should be able to facilitate any large order made by the customer and it should be presented clearly to avoid any confusions.

**Special requirements:** The system must support customisation for special requests.

**Technology and data variation list:** Tablets or desktop terminals to place the order. Notifications to be sent to kitchen staff.

**Frequency of occurrence:** Continual throughout service hours.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Functional requirements | Attributes |
| Staff Order | Waitstaff | Kitchen Staff | (M) 2F2 Display Orders with Details  **(M)** **2F3** Enter New Order (**M)** **2F4** Receive Notification on Ready Orders  **(C)** Modify Current Order | Order ID, Progress status, Quantity, Special Requests |

### Inventory Management

**Primary actor –** Inventory staff

**Supporting actor –** Manager

**Stakeholders and interests – (Inventory staff)** They will need to have access to a user-friendly system which shows them what items are available in stock, and to receive alerts when the stock is low. **(Managers)** They will be interested mainly in the report analysis of their stock usage so they can see how they are able to improve their efficiency.

**Preconditions:** The inventory system must be accessible and functional while accurate stock data must be maintained.

**Goal –** To put in orders for items which are low in stock and add them to them to the inventory records.

**Scenario –** The inventory staff has just received a notification that their stock is low. They are now looking to see which stock they need to reload so that they can input it into the system. After the making the order, they now need to input the new stock item into the system or make any necessary modifications in the system to ensure that the data is accurate. After a set period, a report can be generated to show an analysis of their stock usage.

**Constraints**: Most importantly in this use-case, data accuracy is the most important factor to ensure accurate levels of stock are displayed. This can lead to huge disruptions and any of the data is false. Alerts should be sent at a reasonable time and not too late or early as this can cause overstock or understock.

**Special requirements:** The system must integrate with order management for real time updates.

**Technology and data variation list:** Access stock through a desktop terminal or a tablet.

**Frequency of Occurrence:** Stocks will be checked daily which restocking will happen weekly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Functional Requirement | attributes |
| Order/inventory management | Kitchen staff | Manager | **(M)** **3F1-2** Add and Edit Stock Item Records  **(M)** **3F4** Display Stock Records  **(M)** **3F5** Monitor Stock Levels  **(M)** **3F6** Send Low Stock Notification Alert  **(M)** **3F7-10** Generate Analytic Reports | Stock ID, Item Name, Quantity, Threshold Level, Usage/Waste Logs. |

### Staff Scheduling for Managers

**Primary actor -** Manager

**Supporting actor –** staff

**Stakeholders and interests- (Managers)** They will need to ensure that they have enough staff needed to provide the necessary services and meet customer needs. A user-friendly system will allow them to do this and to also view any holiday request made from the staff members. **(Waitstaff)** They will also have an interest as they will need to be able to see what shift they are going to be working. The system will need to clearly list the day which they are going to be working to ensure that there isn't going to be any confusion.

**Preconditions:** The scheduling system must have accurate employee data.

**Goal** - To create staff rotas and handle any time off or shift change requests with the ability to change shifts around if necessary.

**Scenario –** It has now come to the end of the week and the manager wants to make the new weeks schedule before he heads off home. He goes on to the system and sees that a staff member has made a request for a holiday. He accepts it as he will still have enough staff available to work the following week and, it's also not too busy. After scheduling all the staff, he then submits it so that all staff are notified that a change has been made to their rota so they can go on and view their shifts

**Alternative scenario -** Whilst the manager creates the schedules, he notices that a lot of the staff members have requested time off for overlapping periods. This will result in inadequate staff during their peak periods. The manager resolves this by assessing alternative methods to ensure he has staff which are available for this. After this change has been made and the manager submits it, it will then be sent to the workers with a reason as to why the changes has been made and whether the current schedule is suitable for them. When it comes to customers modifying their reservations, there which be a certain time before the confirmed day of the reservation where they won’t be able to modify it.

**Constraints**: Managers must have full access to be able to create modify and manage their staff schedules. This will only be accessible for managers and not the staff members themselves. Staff will have to submit their holiday/time-off request before a certain time limit to avoid and last-minute disruptions. Alerts of over-lapping time off requests will need to be in place to prevent any understaffing and if necessary, a conflict management tool to be in place to resolve this.

**Special requirement:** The system must comply with labour laws. E.g. ensuring the staff are not being overworked and that they have 11 hours of rest between each working day.

**Technology and data variation list:** Desktop and mobile devices.

**Frequency of occurrence**: Schedules will me made every week.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Functional requirement | Attribute |
| Staff scheduling | Manager | staff | **4F1**: Receive Time Off Request.  **4F2**: Approve Time Off Request  **4F3**: Receive Shift Change Request  **4F4**: Approve Shift Change Request  **4F5**: Create Staff Schedule | Staff ID, Name, Position, Shift Times, Hours Worked. |

### Staff time-off requests

**Primary actor –** Staff

**Supporting actor –** Manager

**Stakeholders and interest – (Staff)** They will need a user-friendly system to view their shifts and to request time off or a shift swap. **(Managers)** They will need a user-friendly system to be able to view requests which staff have made and to accept it or reject it.

**Preconditions –** The request system must be accessible and functional.

**Goal –** Staff member needs to be able to access the system to view their shifts and to request time-off.

**Scenario –** A staff member has planned to go on holiday and has decided the dates which they would like to go. Before booking any flights, the send in a time-off request to see whether they’d be able to get time off from work. After inputting the dates and submitting it, the next day, their request gets accepted.

**Alternative scenario –** A few staff member has booked a certain day off which have already been approved of by the manager. Another worker, who is in aware of this, then proceed to book the same day not knowing a few workers are going to be off on that day. Their request gets rejected and the reason as to why is given to them. They are then offered with alternative days which they can choose whether there won’t be a shortage of staff.

**Constraints:** Staff should only be able to request time off and view shifts. They shouldn’t be able to modify theirs of anyone else’s shifts. The system should have a Holiday request calendar to staff are able to see when other staff members are on holiday to avoid any overlaps. If a staff member books a holiday which seems to overlap with a lot of other staff members holidays, the system should notify them and suggest rescheduling an alternative date. This is to prevent understaffing. Managers should review and respond to time request given by staff within a certain time frame e.g. 48 hours.

**Special requirements-** The system must adhere to HR policies.

**Frequency of occurrence-** This will be accessible whenever the staff needs it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use case | Primary actor | Supporting actor | Requirement analysis | Attributes |
| Staff time request | Staff | Manager | **6F4:** Submit Time Off Request  **6F5:** Submit Shift Change Request  **6F2**: View Records | Request ID, Staff ID, Name, Dates Requested, Status. |

## System Overview

A diagram of a restaurant system

Description automatically generated

**Reminder**: This Context Diagram give an overview of the whole Restaurant System, while showing all subsystems and brief explanation of the functionality carried out in each system.

## Sequence Diagram (UML)

This stage was helpful by allowing us to how objects within our system interact with one another, and how those interactions take place, and what order they take place in. This stage of our design gave us an opportunity to think about how we are going to code and build this system as the interactions in these diagrams are arranged in a timed system from top to bottom.

***\*For the staff scheduling and staff time off request, all the data is stored inside a single database, however the diagram illustrates three logical groupings to enhance the visual clarity and the separation of responsibilities within our system.***

### Inventory Management

This sequence diagram illustrates time ordered interactions between objects in our system. It visualises how data flows and which components are responsible for specific tasks, an example here is how stock is added and validated throughout the system.

#### A diagram of a company AI-generated content may be incorrect.

### Orders Management (Waitstaff)

A diagram of a workflow

AI-generated content may be incorrect.

### Orders Management (Kitchen Staff)

**A diagram of a process

AI-generated content may be incorrect.**

### Scheduling: Staff Scheduling (Managers Side)

A screenshot of a computer screen

AI-generated content may be incorrect.

### Scheduling: Staff Time off Request

A screenshot of a computer screen

AI-generated content may be incorrect.

### Reservation management: Customer

A diagram of a customer service

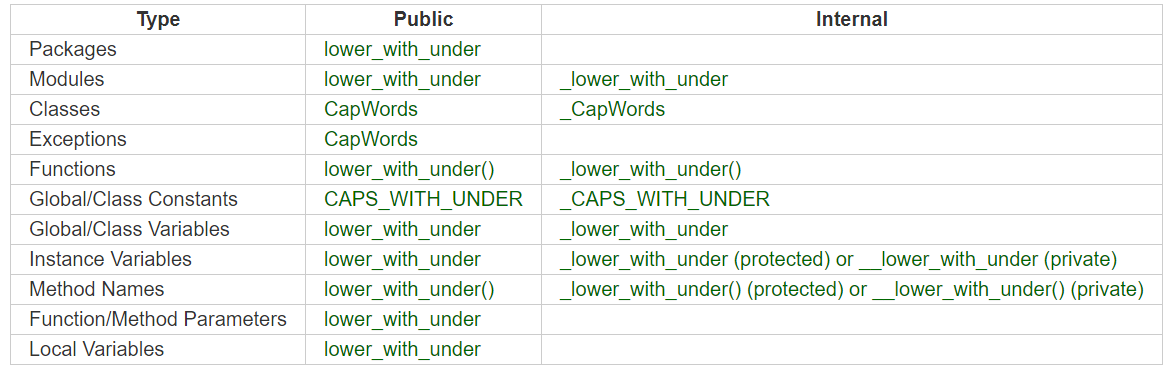
AI-generated content may be incorrect.

### Reservation management: Waitstaff

A diagram of a system

AI-generated content may be incorrect.

## Naming Convention



## Partial Class Diagrams

In this stage, we made partial class diagrams which allowed us to see the classes which exists within our system. This was beneficial as we could see the attributes and operations of each class. We also show the multiplicity between them by illustrating e.g. 1 to \* (one to many).

### Orders Management

A computer screen shot of several boxes

AI-generated content may be incorrect.A diagram of a computer

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

### Scheduling: Staff Time off Request

This partial class diagram demonstrates to us the core entities within our systems and their relationships. Each of the classes included relevant attributes and operations such as staff, shift change request and time off request. Multiplicity notations show how our entities are related which supports accurate data structure implementation.

A diagram of a computer

AI-generated content may be incorrect.

### Scheduling: Scheduling for Managers

A screenshot of a computer screen

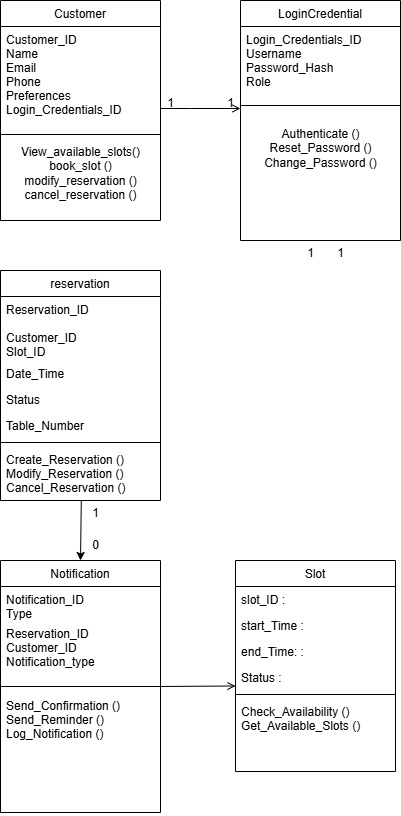
AI-generated content may be incorrect.

### Inventory Management

A close-up of a document

AI-generated content may be incorrect.

### **Reservation: Customer Reservation**



### **Reservation: Waitstaff Reservation**

A diagram of a server

AI-generated content may be incorrect.

## Full Class Diagram

In this stage, we developed on from our partial class diagram to illustrate a full class diagram of our systems classes. This was an important stage for us as this will assist us when we are coding as it shows the relationship between the classes within the system. Our key modules such as order management, staff scheduling and reservation handling, has been designed following object orientated principles with clear separation of their responsibilities and behaviours.

A screenshot of a computer

AI-generated content may be incorrect.

## 

## Component Diagram

In this stage, we identified the components within our system and how they connect to make up a system of sub-systems. We drew out each component and connected them using the lollipop and half lollipop to show what each component provides and what each component requires. This stage was useful for us to visualise how services such as [Reservation service], [Order Service] and [Notification Service] interact.

A diagram of a company

AI-generated content may be incorrect.

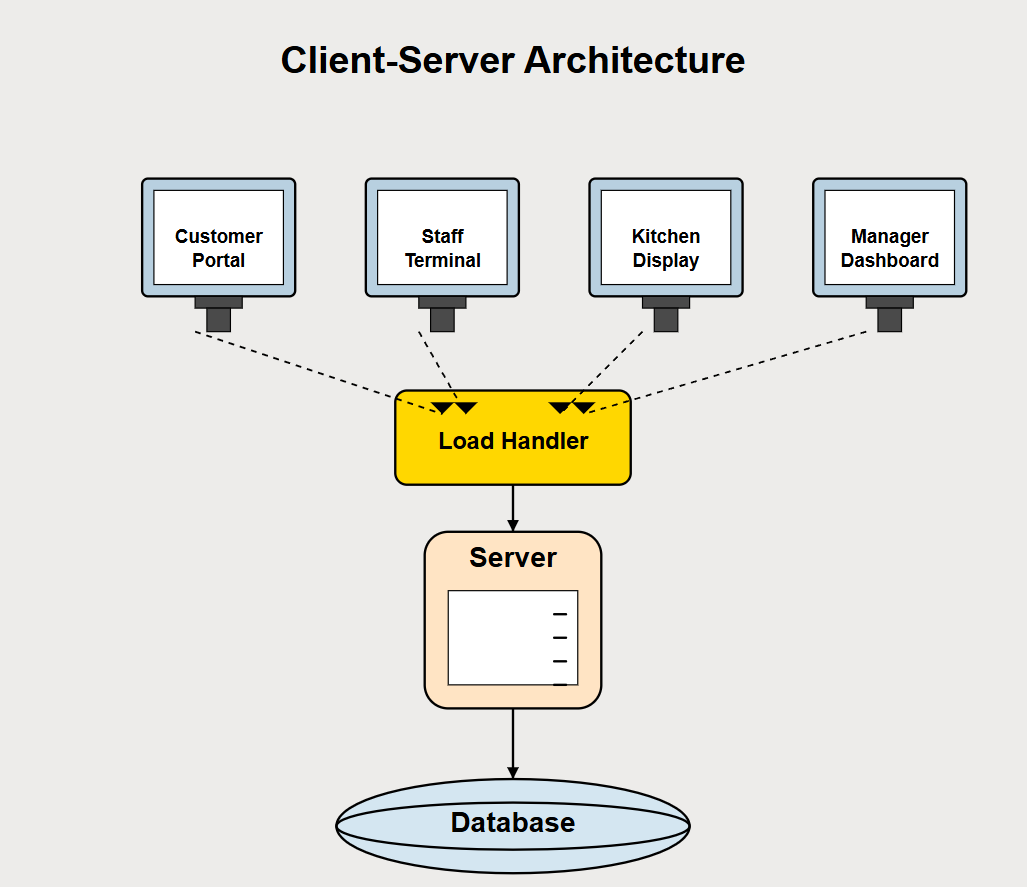
## Functional Requirements to Design Artifacts

To ensure a clear transformation from our requirements to our design, the table below represents each functional requirement in our fully dressed use case to its corresponding design artifacts. This demonstrates how each of our requirements have been considered and represented in out models such as sequence diagrams, class diagrams, and component diagram.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Functional Requirement | Function code | Design Artifact(s) |
| Customer Reservation | View slots with availability | 5F1-2 | Sequence Diagram – Customer Booking  Component Diagram – Customer Portal |
|  | Book reservation slot | 5F3-4 | Sequence Diagram – Customer Booking  Class Diagram – Reservation, Order |
|  | Receive confirmation notification | 5F5 | Sequence diagram |
|  | Send reminders | 5F6-8 | Sequence diagram |
|  | Modify booking | 5F9-10 | Sequence Diagram – Modify Reservation  Staff Class Diagram |
|  | Customer account |  | Class Diagram – Customer, Account |
| Waitstaff Reservation | View reservations | 1F1-2 | Sequence Diagram – View Reservations |
|  | Modify reservation | 1F3-4 | Sequence Diagram – Modify Reservation  Class Diagram – Reservation |
| Kitchen Order | Display orders with details | 2F5 | Sequence Diagram – Kitchen Order Flow  Component Diagram – Kitchen Display |
|  | Update order progress status | 2F6 | Sequence Diagram – Kitchen Order Flow  Class Diagram – Order |
| Staff Order | Display order with details | 2F2 | Component diagram – Staff portal  Sequence Diagram – Staff Order dashboard |
|  | Enter new order | 2F3 | Sequence Diagram – Staff Order Flow  Component Diagram – Order Service |
|  | Receive notification on ready orders | 2F4 | Sequence Diagram – Notification to Waitstaff |
| Inventory Management | Add/edit stock item records | 3F1-2 | Sequence Diagram – Inventory Flow  Class Diagram – Inventory Item |
|  | Display stock records | 3F4 | Class Diagram – Inventory Item  Sequence Diagram – Inventory Query |
|  | Monitor stock levels | 3F5 | Sequence Diagram – Stock Monitoring |
|  | Send low stock notification | 3F6 | Sequence diagram – Send low stock notification |
|  | Generate analytic reports | 3F7-10 | Component Diagram – Inventory Service  Sequence diagram – Inventory management  Class Diagram – Request Usage Report |
| Staff Scheduling (Mgr) | Receive & approve time-off requests | 4F1, 4F2 | Sequence Diagram – Manage Time Off Requests  Class Diagram – Time Off Request |
|  | Receive & approve shift change requests | 4F3, 4F4 | Sequence Diagram – Manage Shift Change Requests  Class Diagram – Shift Change Request |
|  | Create staff schedule | 4F5 | Sequence Diagram – Create Staff Schedule  Class Diagram – Schedule |
|  | View staff records | 4F6-4F7 | Sequence Diagram – View Staff Records  Class Diagram – Manager |
| Staff Time-Off | Submit time-off request | 6F4 | Sequence Diagram – Time-Off Submission  Class Diagram – Time Off Request |
|  | Submit shift change request | 6F5 | Sequence Diagram – Shift Change Request  Class Diagram – Shift Change Request |
|  | View time-off records | 6F2 | Sequence Diagram – View Records |

## Architectural Model – Client-Server Architecture

This stage was important for us as this is the structural foundation upon which our system will be built. *Because of its scalability, modularity, and support for multiple user interfaces, we have decided to use a client-server architecture for our restaurant management system. Through the separation of the frontend (client) and backend (server), this architecture enables various components, including the manager dashboard, staff terminal, kitchen display, and customer portal, to communicate with a central server using standardised requests. One significant benefit is that sensitive data and business logic stay safely on the server, enhancing security and maintainability. Additionally, it allows us to independently develop and update the server logic and user interface, which increases the system's adaptability over time. Furthermore, this method preserves a consistent backend while supporting a variety of access points, such as desktop, tablet, and mobile devices.*



## 

## Data model

The following database schema represents the data structure of our reservation system. In it, includes entities for our reservations, orders, staff schedules, inventory tracking, and time-off requests. I have included the foreign keys to ensure there is relational integrity within the system and that real time data access is supported for all components.

A screenshot of a computer program

AI-generated content may be incorrect.

## Design Patterns – Façade

For the design patterns, we have decided to choose façade as this allows us to simplify complex interactions between UI components and service layers. As you can see in the example below, it shows us how a client communicates with a subsystem by sending requests to a façade. As the façade knows which subsystem class are responsible for handling a request, it can delegate them appropriately within the system.

\***Note – The codes which you see below are not codes that we have used, rather they are example codes which we’ve put in.**

A diagram of a restaurant service

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

# Implementation Stage

**https://github.com/e4omar/implementation-swe-2uni**