OBJECT DESIGN: RELAXING KOALA RESTAURANT INFORMATION SYSTEM

Author: Mohammad, Orson, Matthew, Yasindu, Andy

MAY 5, 2024

GROUP 6 – SWE30003

Swinburne University of Technology

Contents

Executive Summary

This document presents a comprehensive analysis and design proposal for implementing a Restaurant Information System (RIS) for the "Relaxing Koala" restaurant. The RIS aims to streamline reservation management, order processing, and customer service, enhancing the dining experience for both customers and staff. The design phase outlines candidate classes with a UML diagram and CRC cards detailing responsibilities and collaborators. Design quality considerations, including design heuristics and patterns like Factory Method and Singleton, ensure a robust architecture. The bootstrap process defines initialization steps, including database setup and UI initialization. Verification scenarios validate the design's effectiveness. Overall, the proposed RIS design offers a user-centric approach, scalability, efficiency, and seamless integration, optimizing restaurant operations and enhancing service quality for the Relaxing Koala.

1 – Introduction

The following document presents a comprehensive analysis and design proposal for the implementation of a Restaurant Information System (RIS) for the "Relaxing Koala" restaurant. This system aims to streamline various aspects of restaurant operations, including reservation management, order processing, and customer service, to enhance the overall dining experience for both customers and staff members. Through careful problem analysis, consideration of key assumptions, simplifications, design justifications, and discarded class considerations, this document outlines a structured approach to addressing the restaurant's needs and challenges. By leveraging modern technology and adopting a user-centric design philosophy, the proposed RIS seeks to optimize efficiency, improve service quality, and ensure seamless integration into the existing restaurant environment.

2 – Problem Analysis

2.1 – Assumptions

In conducting the problem analysis for the Relaxing Koala restaurant, certain assumptions were made to facilitate the process and guide decision-making:

1. Seating Capacity: It is assumed that the Relaxing Koala restaurant has a seating capacity of 150 people at any given time. This assumption helps in determining the scale and scope of the system requirements, such as reservation management and table allocation.
2. Alcohol Service: The venue is assumed not to serve alcohol. This assumption influences various aspects of the system design, including menu management and payment processing, as alcohol-related functionalities are not required.
3. Staff Training: Staff members of the Relaxing Koala restaurant are required to receive training on the final product. This assumption ensures that staff members are equipped with the necessary knowledge and skills to effectively use the system, leading to smoother implementation and operation.
4. Staff Adoption: It is assumed that staff members will adopt the new processes and technologies introduced by the system. This assumption is essential for the successful integration of the system into the restaurant's workflow and ensures its acceptance and utilization by staff members.
5. Internet Connectivity: The restaurant is assumed to have reliable internet connectivity. This assumption is crucial for the system's functionality, as many aspects, such as online reservations and real-time updates, rely on internet access.

2.2 – Simplifications

During the problem analysis, several simplifications were applied to streamline the process and focus on essential aspects of the system:

- Scope Limitations: The analysis focused on core functionalities relevant to the restaurant's operations, avoiding overly complex or niche features that may not align with the project's objectives or constraints.
- Technical Constraints: Consideration was given to technical constraints such as time, resources, and technological capabilities, ensuring that the proposed solution remains feasible and realistic within the given constraints.
- User Requirements: Simplifications were made in user requirements to prioritize essential functionalities and user interactions, avoiding unnecessary complexity and ensuring a user-friendly experience.

2.3 – Design Justification

The design decisions made during the problem analysis phase were driven by the following justifications:

- User-Centric Approach: The design prioritizes user needs and preferences, aiming to enhance the dining experience for both customers and staff members.
- Scalability: Consideration was given to the restaurant's potential growth and future needs, ensuring that the system design can accommodate expansion and increased demand over time.
- Efficiency: The design focuses on optimizing restaurant operations and workflow, streamlining processes such as order management, table allocation, and payment processing to improve efficiency and reduce wait times.
- Integration: Compatibility and integration with existing systems and technologies were considered to minimize disruption and facilitate seamless implementation and operation of the new system.

2.4 – Discarded Class List

During the problem analysis, certain classes were considered but ultimately discarded for various reasons:

- Waiter / WaiterEvent / Tip / WaiterHandler / Database --> Waiters/WaiterEvents: Initially considered for tracking waiter-related data in the database, but deemed out of scope due to the focus on core restaurant operations and functionalities.

These discarded classes were deemed unnecessary or outside the scope of the project, leading to their exclusion from the final system design.

3 – Candidate Clases

3.1 – Candidate Class List

- Customer
- Product
- Table

- Booking
    - VenueBooking
- Order
- OrderType (enum)
    - DineIn
    - Takeaway
    - Delivery
- Invoice
- Payment (abstract)
    - PaymentCash
    - PaymentCard
- CustomerHandler
- OrderHandler
    - → Connects to an Order Printer peripheral.
- BookingHandler
- PaymentHandler
    - → Connects to a Payment Terminal peripheral.
- StatisticsHandler
- FrontOfHouseUI
- DataVisUI
- Database
    - ★ Customers
    - ★ Products
    - ★ Tables
    - ★ Bookings
    - ★ Payments

3.2 – UML Diagram

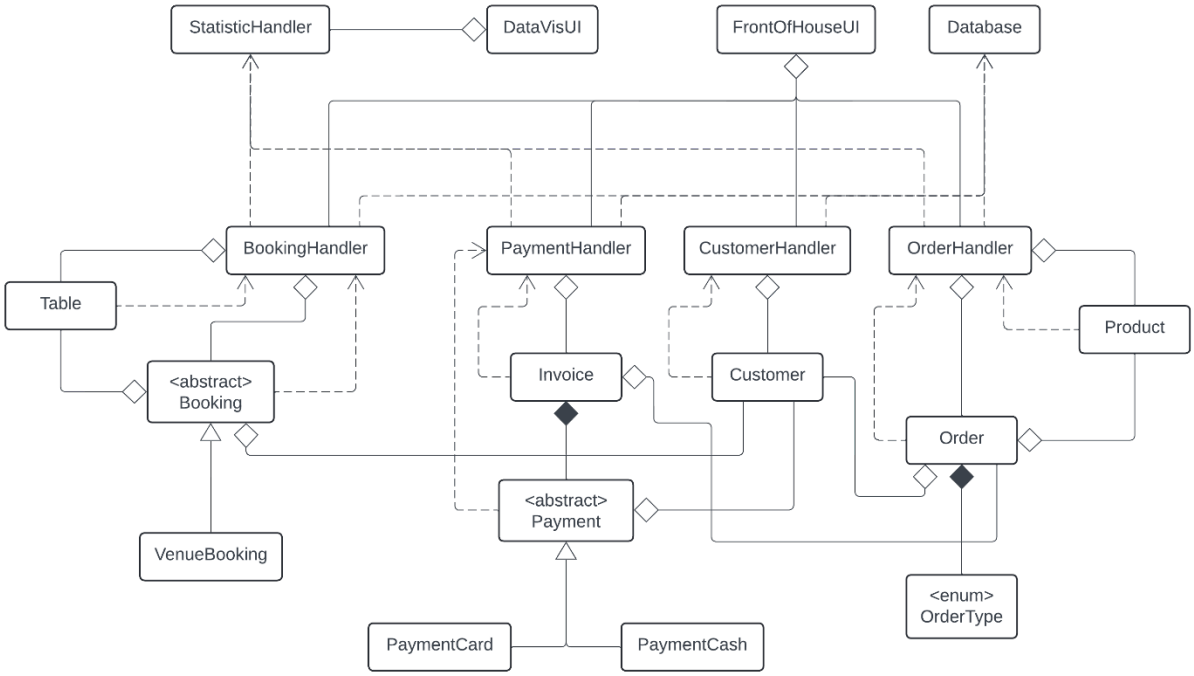


Figure 1: Class UML Diagram

After analyzing the system requirements and our identified classes we came up with the UML diagram to describe the system. It is separated into 4 main parts:

- The Database class which handles interaction with the database.
- 'Data' classes which represent database data.
- 'Handler' classes which create and manage 'Data' classes by interfacing with Database.
- 'UI' classes which use the information the 'Handler' classes can provide to handle program logic and UI.

The idea is to split the design into layers and to abstract actual data access and management behind the handler classes so that the user-facing programs can just request data instead of having to go through the database and deal with stuff like caching.

3.3 – CRC Cards

Class Name: Customer **Superclass:** -	
Records the identity of a single customer.	
Responsibilities	**Collaborators**
Knows customer ID.	Database
Holds customer name & phone number/s.	-

Class Name: Product **Superclass:** -	
Records a single product.	
Responsibilities	**Collaborators**
Knows product ID.	Database
Holds product name & current price.	-

Class Name: Table **Superclass:** -	
Records a single table at the restaurant.	
Responsibilities	**Collaborators**
Knows table ID.	Database
Holds table name & size.	-

Class Name: Order **Superclass:** -	
Records a Customer's purchase of products.	
Responsibilities	**Collaborators**
Knows order ID.	Database
Holds ordering Customer, ordering Table, list of ordered Products, OrderType, time ordered, & whether the order has been delivered or not.	Customer, Table, Product, OrderType

Class Name: OrderType (enum) **Superclass:** -	
Enum for the type of an order: DineIn, Takeaway, or Delivery.	
Responsibilities	**Collaborators**
-	-

Class Name: Booking **Superclass:** -	
Records a booking of a table.	
Responsibilities	**Collaborators**
Knowns booking ID.	Database
Holds Customer, booked Table, & timeframe booked.	Customer, Table
.toString()	Customer, Table

Class Name: VenueBooking **Superclass:** Booking

Records a booking of all tables in the restaurant.	
Responsibilities	**Collaborators**
-	-

Class Name: Invoice **Superclass:** -	
Invoice generated for completed orders at the restaurant, providing a detailed breakdown of items ordered and total cost	
Responsibilities	**Collaborators**
Knows invoice ID.	Database
Holds Payment, & list of related Orders.	Payment, Order
Calculates the total cost of all its related Orders on creation.	Order
.toString()	Payment, Order

Class Name: Payment (abstract) **Superclass:** -	
A payment made by a Customer.	
Responsibilities	**Collaborators**
Holds amount paid & Customer paid by.	Customer
.toString() (virtual)	Customer

Class Name: PaymentCash **Superclass:** Payment	
A payment made by a Customer via cash.	
Responsibilities	**Collaborators**
Receives cash payments	-
Handles change for customers cash payments	

Class Name: PaymentCard **Superclass:** Payment	
A payment made by a Customer via credit card.	
Responsibilities	**Collaborators**
Holds card number & transaction ID.	-

Class Name: CustomerHandler **Superclass:** -	
A class that deals with managing Customer data.	
Responsibilities	**Collaborators**

Holds loaded Customers.	Customer, Database
Loads in Customers from Database.	Customer, Database
Creates new Customers.	Customer, Database
Modifies Customer data.	Customer, Database
Provides utility functionality related to Customer data.	Customer
Interfaces with email library to send booking confirmation emails to customers.	

Class Name: OrderHandler **Superclass:** -	
A class that deals with managing Order and Product data.	
Responsibilities	**Collaborators**
Holds loaded Orders/Products.	Order, Product, Database
Loads in Orders/Products from Database.	Order, Product, Database
Creates new Orders/Products.	Order, Product, Database
Modifies Order/Product data.	Order, Product, Database
Provides utility functionality related to Order/Product data.	Order, Product
Interfaces with an Order Printer peripheral to provide orders to the kitchen.	Order, Product

Class Name: BookingHandler **Superclass:** -	
A class that deals with managing Booking data.	
Responsibilities	**Collaborators**
Holds loaded Bookings.	Booking, Database
Loads in Bookings from Database.	Booking, Database
Creates new Bookings.	Booking, Database
Modifies Booking data.	Booking, Database
Provides utility functionality related to Booking data.	Booking

Class Name: PaymentHandler **Superclass:** -	
A class that deals with managing Invoice/Payment data.	
Responsibilities	**Collaborators**
Holds loaded Invoices/Payments	Invoice, Payment, Database
Loads in Invoices/Payments from Database.	Invoice, Payment, Database

Creates new Invoices/Payments.	Invoice, Payment, Database
Modifies Invoice/Payment data.	Invoice, Payment, Database
Provides utility functionality related to Payment data.	Payment
Interfaces with Payment Terminal peripheral to retrieve and verify card payments.	Payment

Class Name: StatisticsHandler **Superclass:** -	
A class that deals with generating statistical data to for the system.	
Responsibilities	**Collaborators**
Collaborates with other 'Handler' classes to retrieve and parse statistical data.	CustomerHandler, OrderHandler, PaymentHandler, BookingHandler

Class Name: FrontOfHouseUI **Superclass:** -	
A class that handles the front-of-house program logic and UI.	
Responsibilities	**Collaborators**
Requests and retrieves data about Customers, Orders, Bookings, and Payments.	CustomerHandler, OrderHandler, BookingHandler, PaymentHandler
Allows the user to modify database data.	CustomerHandler, OrderHandler, BookingHandler, PaymentHandler
Facilitates front-of-house UI actions.	CustomerHandler, OrderHandler, BookingHandler, PaymentHandler
Manages front-of-house program UI.	-

Class Name: DataVisUI **Superclass:** -	
A class that handles the data-visualization program logic and UI.	
Responsibilities	**Collaborators**
Requests and retrieves statistical data.	StatisticsHandler
Presents statistical data to the user.	Customer, Product, Order, OrderType, Booking, Invoice, Payment
Manages data visualization program UI.	-

Class Name: Database **Superclass:** -	
A class that represents a connection to the database storing all the information	
Responsibilities	**Collaborators**
CRUD operations for all database data.	Customer, Product, Order, OrderType, Booking, Invoice, Payment.

4 – Design Quality

4.1 – Design Heuristics

H1 A class should capture one and only one key abstraction.

H2 All data unique to a class should be kept hidden and encapsulated within the class, adhering to the principle of encapsulation.

H3 All the data inside a base class should be kept and declared as private ensuring that they are only accessible within the class itself and not by external classes.

H4 An important aspect of programming is to have consistency. Consistency in naming conventions, diagrams, commenting, code formatting should be established to enhance code readability.

H5 Generated errors will be plain in language and proactively suggest a solution.

H6 A derived class should not be capable of overriding a base class method that returns a null operation.

H7 Do not create any ‘super-classes or a god class.

H8 Base classes should be declared abstract as they provide common functionality for the subclasses, they should be clearly marked as abstract classes.

H9 Abstract classes should only be base classes and serve as base classes only, by providing a template for the subclasses to implement specific behavior.

5 – Design Patterns

5.1 – Creational Patterns

5.1.1 Factory Method Design Pattern

The Factory Method design pattern is a creational pattern particularly useful for creating objects with similar but distinct roles and responsibilities. When we are looking at the Relaxing Koala Restaurants Information System, the Factory Method pattern is employed by various handlers to facilitate the creation of their handled classes. For example, OrderHandler parses database data to create Order and Product classes.

A variant of the Factory Method pattern is used for some handlers: the Abstract Factory pattern. This pattern creates a family of classes given abstract input data. For example, PaymentHandler parses database data to produce ‘Payment’ classes.

An advantage of this pattern is if more variants of classes were to be added, only the factory classes would need to be modified to allow the parsing of these new data formats.

5.1.2 Singleton Design Pattern

The Singleton design pattern ensures that a class has only one instance and provides a global point of access for that instance throughout the program. The Singleton pattern is very useful in the Relaxing Koala Restaurant information system, this is mainly because the Singleton pattern is used to make sure that certain essential components have a single, shared instance:

- **Database:** For instance, the database class is represented as Singleton. This ensures that there is only one connection to the database through the application event handlers. Having multiple different connections could lead to having inconsistency in data retrieval and storage.
- **Handlers:** Essential classes such as CustomerHandler, OrderHandler, and BookingHandler have also been implemented as Singletons. These classes are a crucial part of the system as they manage and enter critical data entries related to customers, orders, bookings. The main advantage of having this is that it ensures that each handler has centralized control and management of the associated data types. Helping to avoid inconsistencies that can arise from multiple instances.
- The Singleton pattern was also applied with **UI components** such as **FrontOfHouseUI** and **DataVisUI** to maintain a single interface instance.

5.2 – Behavioral Patterns

5.2.1 Strategy Design Pattern

The Strategy design pattern is a design pattern that defines a family of classes to solve one problem in multiple ways based on the class. This pattern is used minorly in the Booking and Payment classes, specifically in their '.toString()' methods so that they can be individually described on a per-class basis.

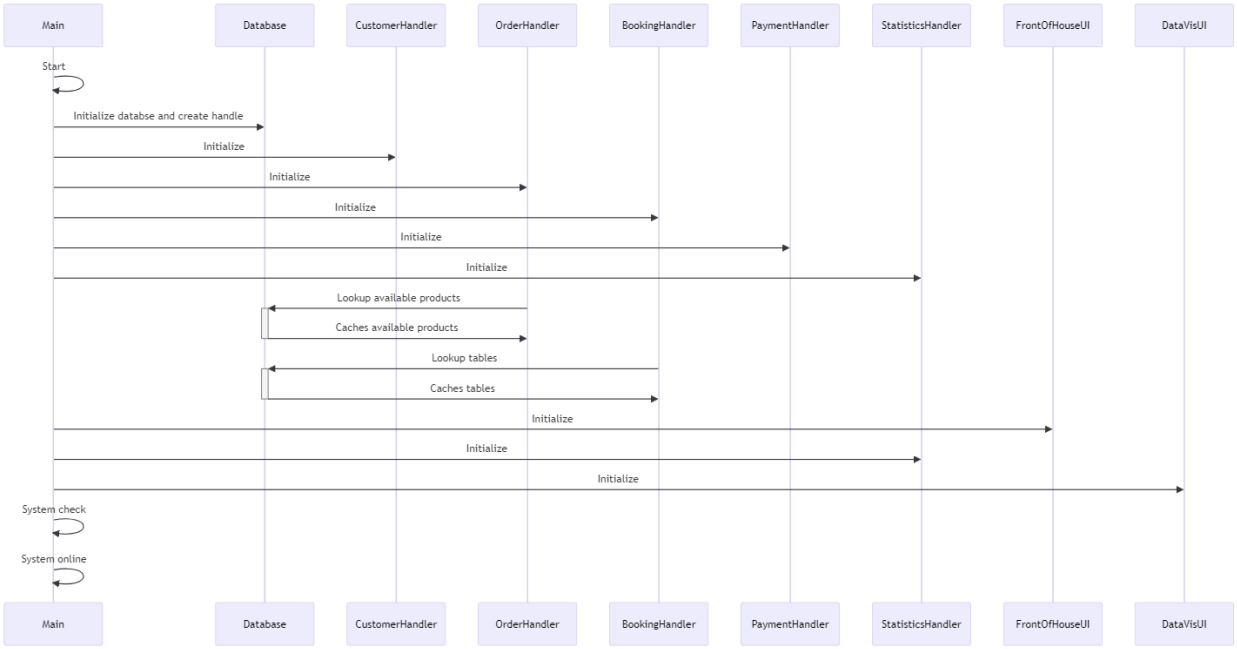
5.3 – Structural Patterns

5.3.1 Proxy Design Pattern

The proxy design pattern allows the creation of an intermediary class for an object. This pattern is used in the Database class and 'Handler' classes. The Database class acts as an intermediary for the database object, and the 'Handler' classes act as intermediaries for their controlled data.

The main advantage of the proxy pattern is that it allows control over the input and outputs of an object. For example, as the 'Handler' classes are the main point of access for their controlled classes in the database, they can cache database data, allowing quicker runtime access which doesn't need to go through the database connection.

6 – Bootstrap Process



Generic:

- The database is set up and filled with initial data (Products, Tables).
- Main creates an instance of the Database class.
- Initialize handler classes (CustomerHandler, OrderHandler, BookingHandler, PaymentHandler, and StatisticsHandler).
- OrderHandler fetches all available Products from the Database and caches them.
- BookingHandler fetches all currently existing Tables from Database and caches them.

FoH Program:

- FrontOfHouseUI is initialized.

Statistics Program:

- Initialize StatisticsHandler.
- DataVisUI is initialized.

Figure 2: Bootstrap Sequence Diagram

7 – Verification

7.1 – Bookings

This diagram depicts the interactions needed to create a booking.

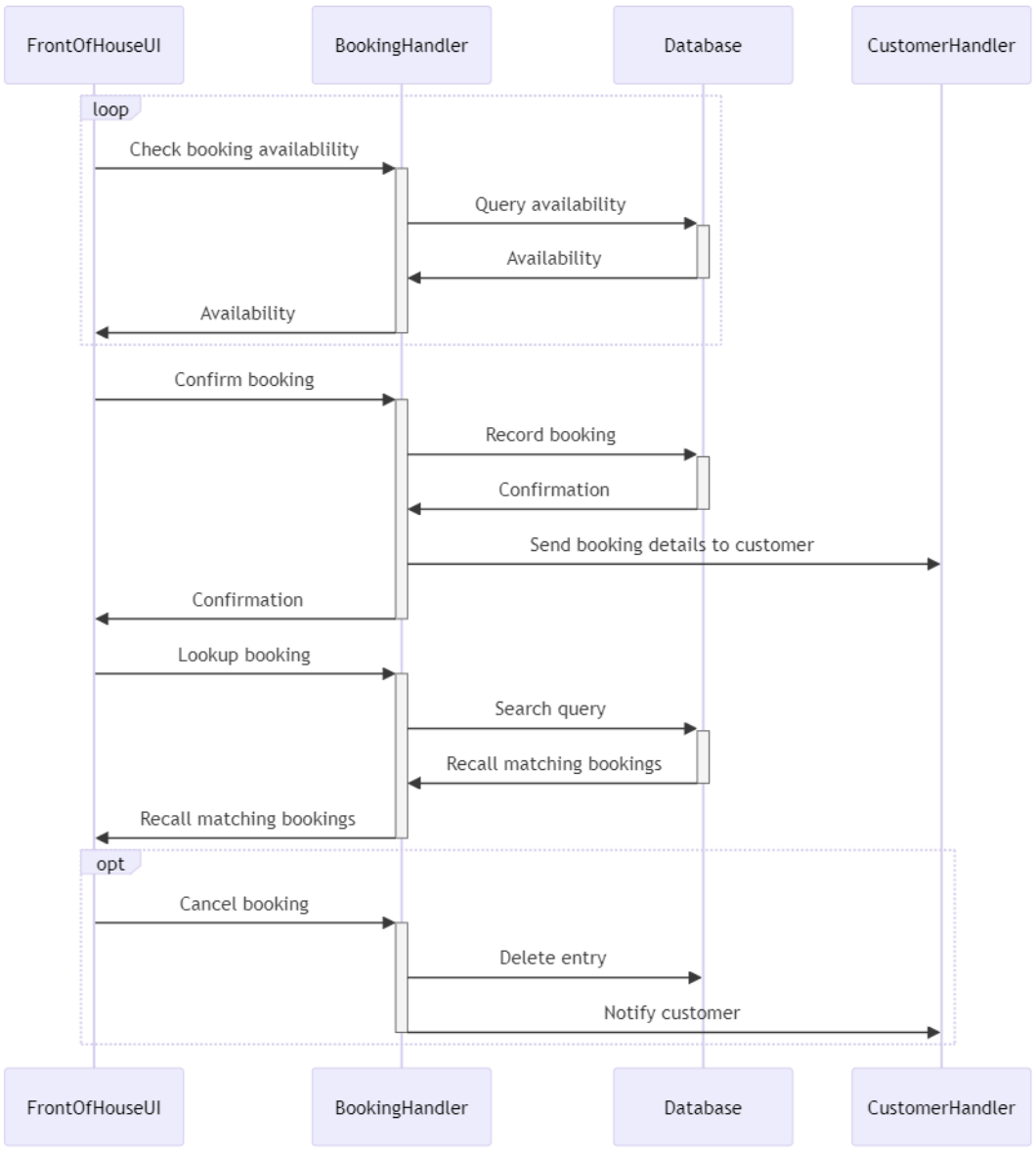


Figure 3: Booking Diagram

The booking handler makes sure the booking is available before letting the user confirm the booking. The booking is then entered into the database and can be retrieved later by the booking handler.

Bookings may also be deleted once they have been retrieved.

7.2 – Orders

This diagram depicts the interactions needed to create an order.

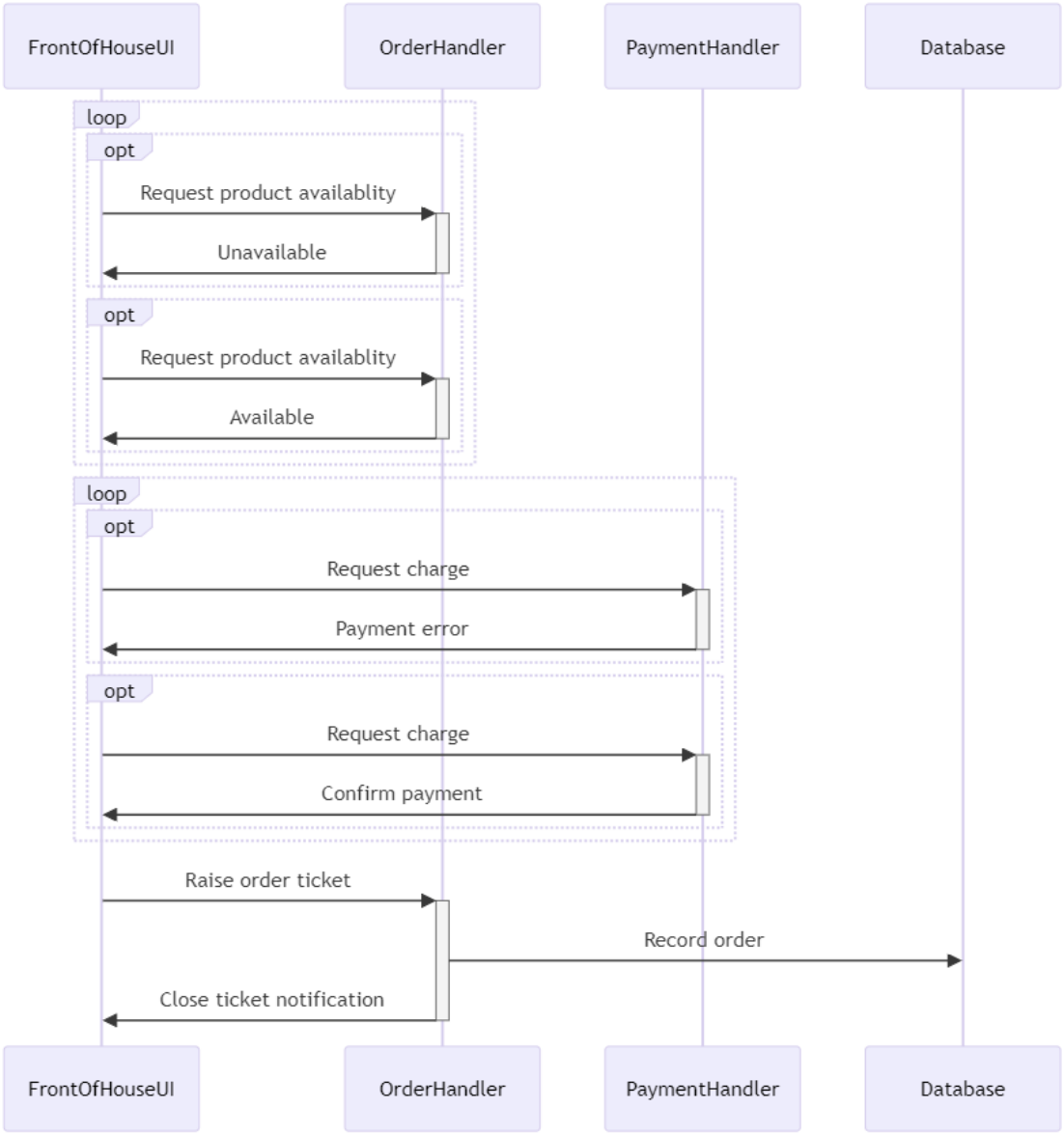


Figure 4: Orders Diagram

The order handler will only allow orders to be placed for products that are in stock and the payment handler will only let the order be placed when the payment goes through.

The order is then entered into the database by the order handler.

7.3 – Booking the Venue

This diagram depicts the interactions needed to book the entire venue for a day.

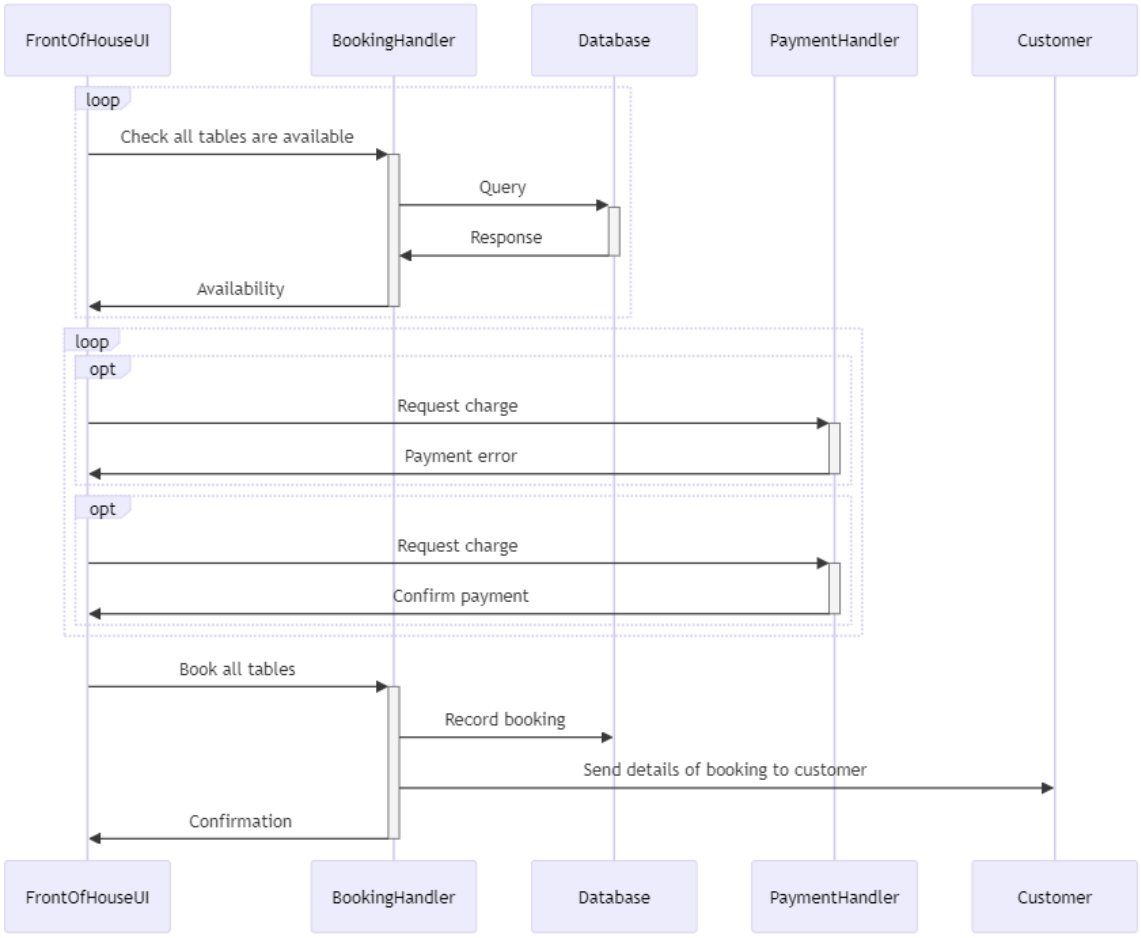


Figure 5: Booking the Venue Diagram

The front of house UI asks the booking handler to create a booking in the database and the handler checks that this is possible before making the booking.

The venue must be paid for upfront, so the payment handler checks the payment goes through and then the booking is entered into the database by the booking handler.

The customer is then notified of the booking so they have a record of the agreed date.

7.4 – Management Console

This diagram depicts the interactions needed to generate a statistical report.

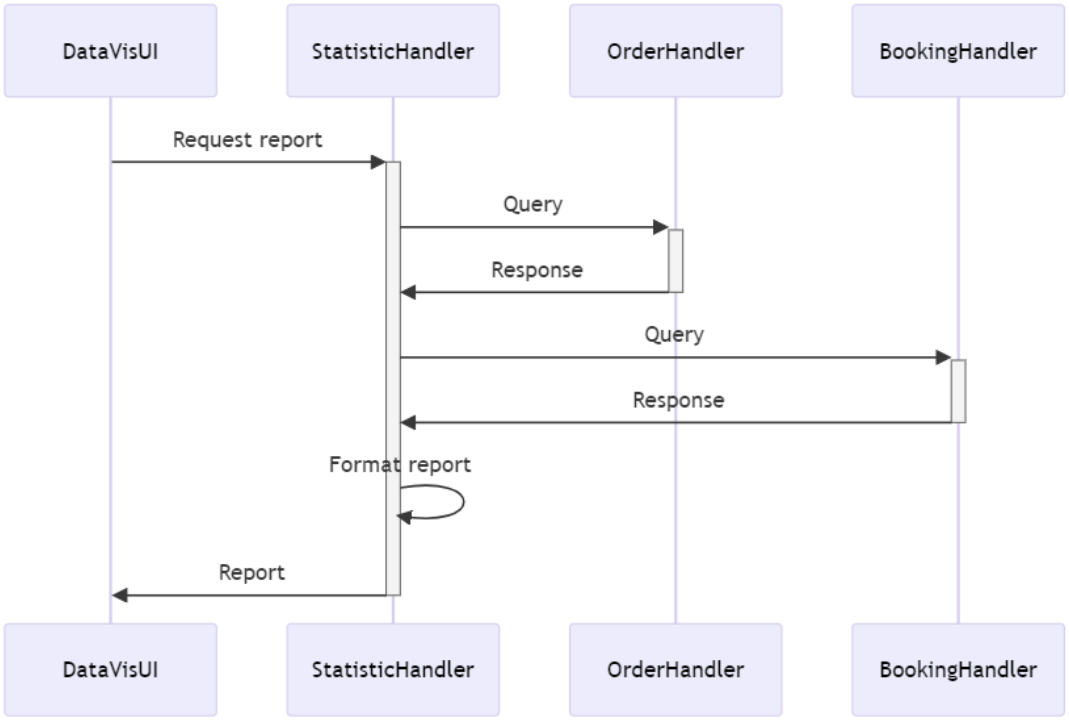


Figure 6: Management Console Diagram

The data visualization UI asks the statistics handler to fetch data from the database and generate a report.

The report is then displayed to the user.

8 – References

MEYER, B. (1992). APPLYING "DESIGN BY CONTRACT." COMPUTER (LONG BEACH, CALIF.), 25(10), 40–51. HTTPS://DOI.ORG/10.1109/2.161279

MARTIN, R. (2003). UML FOR JAVATM PROGRAMMERS (1ST EDITION.). PEARSON.

WIRFS-BROCK, R., & MCKEAN, A. (2003). OBJECT DESIGN : ROLES, RESPONSIBILITIES, AND COLLABORATIONS. ADDISON-WESLEY.

Budd, T. (2002). *An introduction to object-oriented programming* (3rd ed.). Addison-Wesley.

9 – Appendices

Appendix 1: Assignment and Project Coversheet Declaration

Cover sheet for submission of work for assessment

SWINBURNE UNIVERSITY OF TECHNOLOGY

UNIT DETAILS

Unit name	Software Architectures and Design			Class day/time	Tuesdays, 11:30AM
Unit code	SWE30003	Assignment no.	2	Due date	05/05/2024
Name of lecturer/teacher	Prof Jun Han				
Tutor/marker's name	Dr Mostafa Farshchi				

Office use only

Faculty or school date stamp

STUDENT(S)

	Family Name(s)	Given Name(s)	Student ID Number(s)
(1)	Mohammad	Hassanuzzaman	103820955
(2)	Routt	Orson	103575527
(3)	Matthew	Hulme	103112063
(4)	Yasindu	Ariyarathna	103494389
(5)	Andy	Ibrahim	100685973
(6)			

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)	Mohammad	(4)	Yasindu Ariyarathna
(2)	Orson Routt	(5)	Andy Ibrahim
(3)	Matthew Hulme	(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **www.swin.edu.au/student/**

Copies of this form can be downloaded from the Student Forms web page at **www.swinburne.edu.au/studentforms/** | PAGE 1 OF 1

APPENDIX 2: SRS: RELAXING KOALA RESTAURANT INFORMATION SYSTEM

SRS: REALAXING KOALA RESTAURANT INFORMATION SYSTEM

Mohammad, Orson, Matthew, Yasindu and Andy

APRIL 7, 2024
SWINSOFT GROUP-6
SWE30003

Table of Contents

1 - Introduction

This specification document is for a Restaurant Information System to assist the staff of the Relaxing Koala with basic daily tasks to do with reservations, customer ordering, and managerial tasks along with supporting statistical measurement of menu items and online menu availability.

This document aims to specify the requirements and quality attributes of the system by identifying tasks using the Tasks & Support approach.

2 - Project Overview

The relaxing Koala is a medium sized cafe/restaurant serving university students in Glenferrie Road. They are seeking to enhance their daily operations to boost their business by implementing information systems to enhance and grow their daily operations.

Some of the goals they have in mind to improve and streamline the overall operation is in taking orders from guests, passing the orders through to the kitchen team, reservations, payment handling, statistic tracking and having a place for customers to view their menu and order online.

The Existing manual operations of this cafe/ restaurants has been given to redevelop by Swinsoft consulting, and our goal is to implement a comprehensive Restaurant Information system (RIS) tailored to the needs of Relaxing Koala Cafe to improve and scale their business.

2.1 - Domain Vocabulary

POS – Point of Sale, where customer payment is processed.

Reservation – Customers may reserve a table for themselves ahead of time so that they do not need to wait upon arrival.

Dinner Service – Period of time during which the kitchen is open for dinner to be served.

FOH – Front of House, customer-facing staff such as wait staff and bartenders.

BOH – Back of House, non-customer-facing restaurant staff such as chefs.

FIFO – First in First Out, customers who are first to arrive are served first, stock that is the least fresh is used up before fresher stock.

Pushing a dish – When wait staff are instructed to recommend a dish due to excess in stock of a particular ingredient.

RSI – Restaurant Information System

UI – User Interface

UX – User Experience

Capacity – Number of customers the restaurant can serve per day.

Invoicing – Generating bills for customers.

Menu Items – food items available.

Orders – requests made by customers for a food item.

Statistics – Data analysis related to menu items ordered.

2.2 - Goals

Scalability

The main goal is to ensure the restaurant's daily operations can smoothly transition from its current capacity of 50 to 150 customers. The new system should be able to handle the increased volume of orders, reservations, and transactions efficiently.

Automating daily operations

The new system should automate the specified tasks such as taking orders, sending them to the kitchen, taking payments and generating invoices. Automating these tasks should improve the efficiency of the daily operations and help scale the business to accommodate the increased capacity.

Data Collection and Statistics

The system should collect data on menu item orders and generate basic statistics to outline information about purchases. This information will help the restaurant understand customer preferences, popular items, least popular items, enabling the business to make better decisions regarding menu items and prices.

Accounting

The goal for all accounting related processes at the restaurant is to streamline the transaction process for customers while providing invoices. This will include developing a system that will accept various payment methods, providing clear and detailed invoices for orders and recording transactions for the business' records.

2.3 – Assumptions

1. Staff Training and Adoption:

- It is assumed that staff members of the Relaxing Koala restaurant will receive adequate training and support to effectively use any new technology or systems implemented.
- The assumption is made that staff members are open to adopting new processes and technologies to enhance their workflow and improve customer service.

2. Internet Connectivity and Infrastructure:

- It is assumed that the restaurant has reliable internet connectivity to support cloud-based solutions and online functionalities.
- The assumption is made that the existing infrastructure of the Relaxing Koala restaurant, including network capabilities and hardware resources, can accommodate the proposed system requirements.

3. Data Security and Privacy Compliance:

- It is assumed that the implemented system will adhere to relevant data security and privacy regulations, ensuring the protection of customer information and transactional data.
- The assumption is made that necessary measures will be in place to safeguard sensitive data, such as encryption protocols and access controls.

4. Customer Engagement and Feedback:

- It is assumed that the Relaxing Koala restaurant management will actively seek customer feedback and monitor engagement with the implemented system to make necessary adjustments and improvements.

- The assumption is made that customer satisfaction and usability will be key metrics in evaluating the success of the system implementation.

5. Budget and Resource Allocation:

- It is assumed that the Relaxing Koala restaurant has allocated sufficient budget and resources for the implementation and maintenance of the proposed system.
- The assumption is made that any additional costs, such as hardware upgrades or staff training, have been accounted for in the project plan.

These assumptions provide a foundational understanding of the project context and help guide decision-making throughout the implementation process of the Restaurant Information System for the Relaxing Koala.

2.4 - Scope

The owners of Relaxing Koala intend to upgrade their business by integration of automation required for making reservations, taking orders from customers, informing the kitchen about orders, payment, creating invoice and receipts for customers. They also want an option to get basic statistics about ordered menu items. For Better informing to potential customers the owners also want to incorporate the menus online with the optionality of ordering take-away menus and possibly delivery arrangements.

3 - Problem Domain

3.1 - Pain Points

The current methods used by relaxing Koala has several pain points that need to be resolved.

1. Manual processes: Relaxing Koala relies on manual methods for order taking, managing reservations and processing payments which is not efficient to scale the business as it causes delays, errors, and inefficiencies.

2. No data insights: without a system for data collection and analysis, the business will struggle to gain insights into what the customers are ordering and popular menu items.

3. Customer experience: Service quality is subject to inconsistencies due to human error and communication gaps between the front of the house and back of the house.

3.2 - Domain Entities

- Customer
- Front-of-House
- Booking
- Order
- Kitchen
- Product
- Waiter
- Invoice
- Accountant

3.3 - Actors

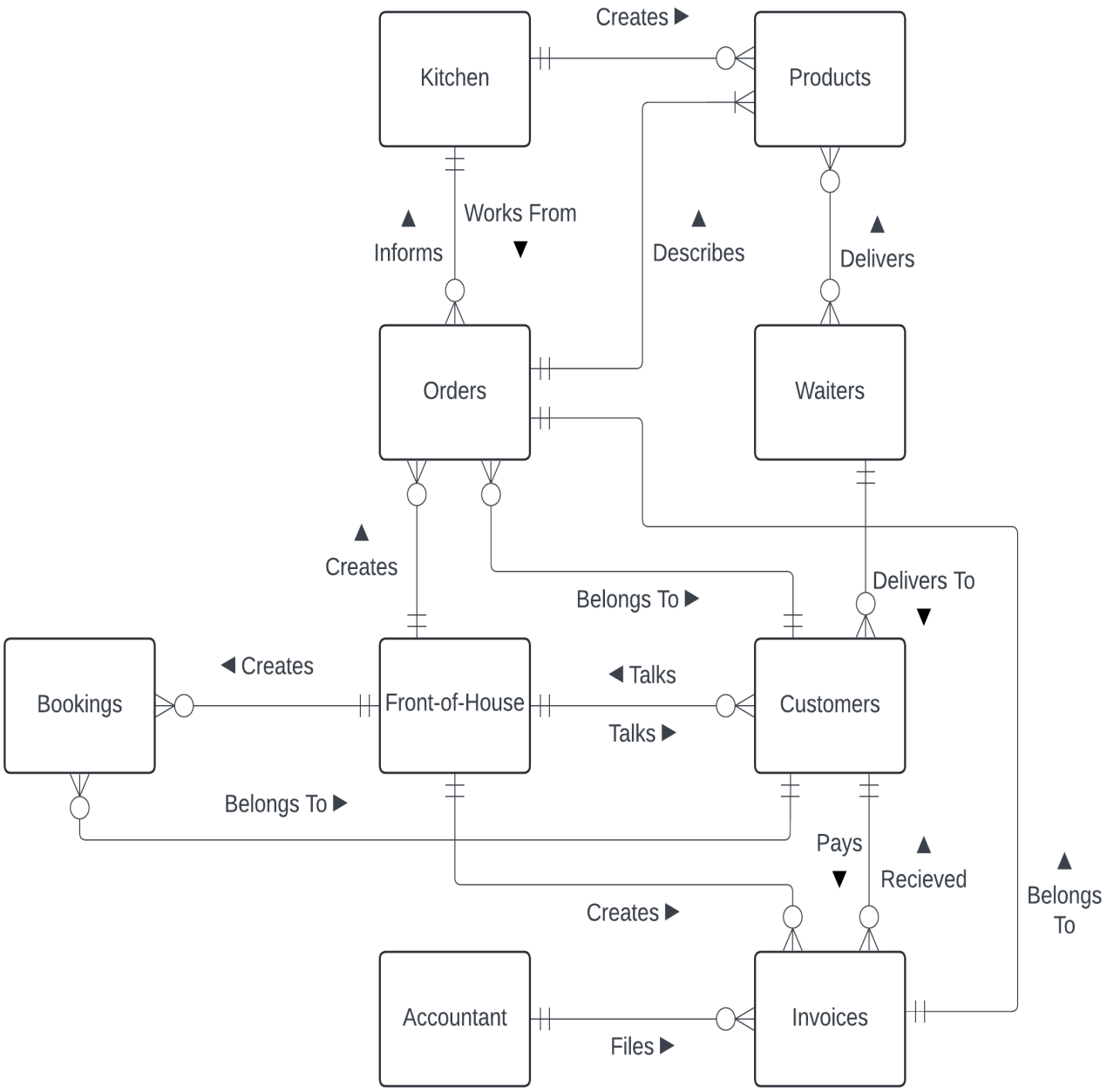
- Customers
- Waiters
- Kitchen Staff
- Accountant
- Manager

3.4 - Tasks

1. Taking Orders
2. Preparing Orders
3. Delivering Orders
4. Making Reservations
5. Creating Statistics Reports
6. Updating Customer Information

4 - Data Model

4.1 - Domain Model



4.2 - Entity Descriptions

Customer

A customer who wants to purchase from the restaurant. They can order in person or by phone, can choose to dine in or buy takeaway, and can make phone bookings to reserve seats in advance.

Front-of-House

The front-of-house aspect of the restaurant. Is responsible for talking to the customer, in person or on the phone, and handles all customer actions (ordering, booking, getting payment, etc.).

Order

An order made by a customer. Orders describe several Products a customer is purchasing, along with whether they are for takeaway or not.

Kitchen

The restaurant's kitchen. It is responsible for taking orders and preparing the products they describe.

Product

Something which the restaurant sells. Orders describe a list of these, may be prepared differently if the order is for takeaway (e.g., put in a takeaway container).

Waiter

A waiter working at the restaurant. Waiters are responsible for delivering products to customers in the restaurant.

Booking

A booking made by a customer. Reserves seating in the restaurant for a number of people at some time.

Invoice

An invoice for an order. To be paid by the order's customer either at the time of ordering (in person) or when the order is picked up (takeaway).

Accountant

An employee responsible for dealing with the restaurant's finances. Handles and files invoices.

5 - Functional Requirements and Task Descriptions

Task: 1. Taking Orders Purpose: Determine what a customer would like to purchase. Trigger: Customer has contact with business and can make an order. Frequency: Frequent (~100 per hour). Critical: A large number of orders (140+ per hour).	
Subtasks:	**Example Solution:**
1. Inform customers of any unavailable menu items.	Customers are warned of any unavailable menu items ahead of ordering so that they may ask for time to reconsider if they had intended on ordering those items.
2. Seat customer.	Customer seating is decided, and they are told where to sit.
3. Ask the customer what they would like to purchase.	Customer walks into business and up to counter. Front-of-house asks the customer what they would like to order.
4. Record customer order.	Customer tells front-of-house their order and it is recorded.
5. Take payment.	Customer is asked to pay and makes a payment. An invoice is generated. This order is logged in the database.
Variants:	
2a. Customer isn't dining in.	Customer will not be seated.
4a. Customer was not in business when ordering.	Payment will be collected when the customer is given their order.
5a. Customer ordered take-out.	Customer is told a time estimate for their order's completion and not seated.

Task: 2. Preparing Orders Purpose: Prepare products the customer has ordered. Trigger: An order has been created by the front-of-house. Frequency: Frequent (~100 per hour). Critical: A large number of orders (140+ per hour).	
Subtasks:	**Example Solution:**
1. Deliver order to kitchen.	After orders are created on the system, they are automatically printed as tickets for the kitchen. Order details are also logged into to the database.
2. Prepare order.	The kitchen reads the order ticket and prepares all relevant products.
3. Plate order.	Relevant products are plated.
4. Deliver order to front-of-house.	Relevant products are moved to a delivery area to be given to customers.
Variants:	
3a. An item/s is take-out.	Relevant products are put into take-out containers.
4a. An item/s is take-out.	Relevant products are put in a separate area and are packed into a bag/s.

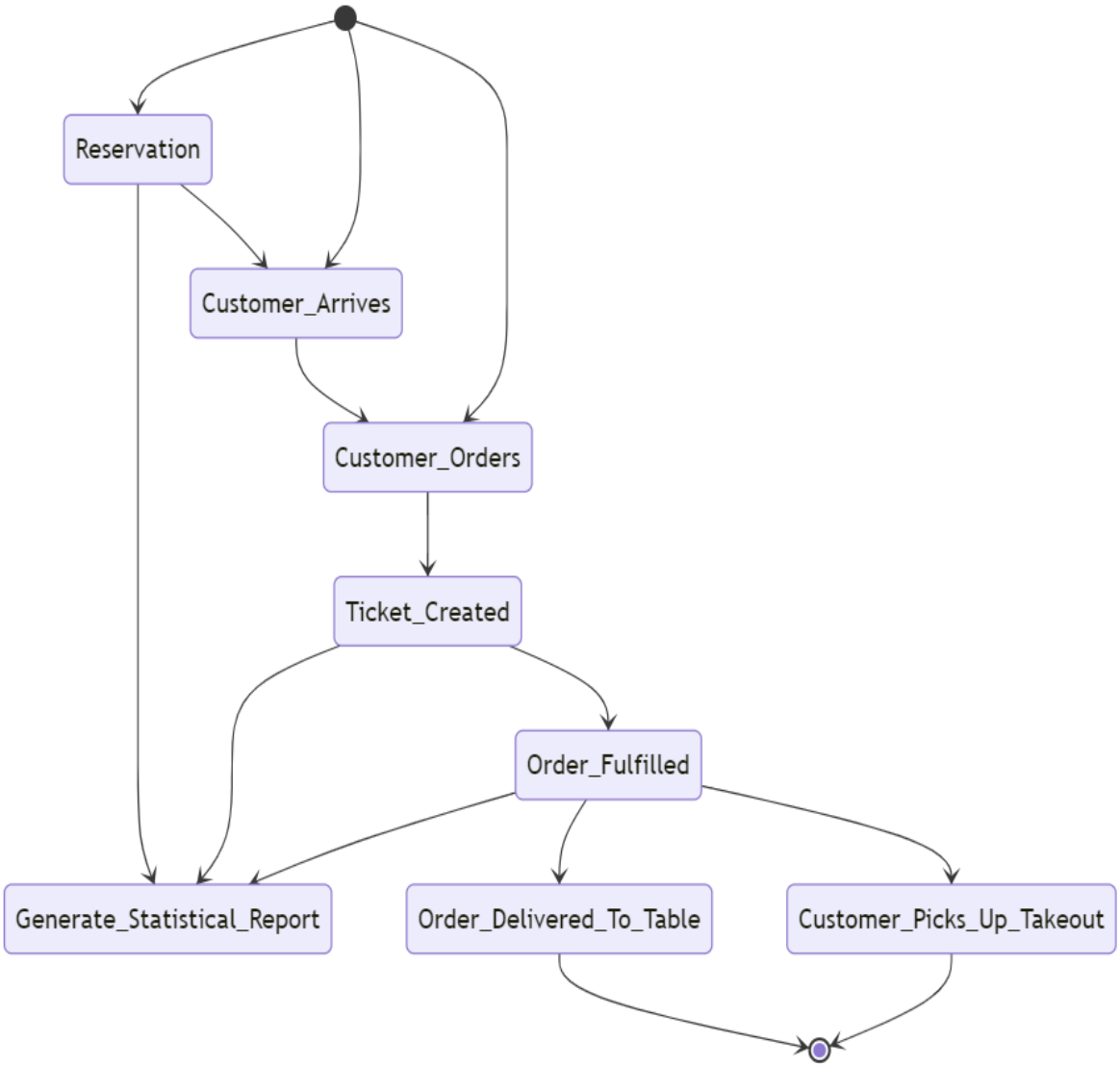
Task: 3. Delivering Orders Purpose: Deliver products to customers. Trigger: Order has been prepared, and customer can receive it. Frequency: Frequent (~100 per hour). Critical: A large number of orders (140+ per hour).	
Subtasks:	**Example Solution:**
1. Prepare order for delivery or pickup.	Orders are plated or packaged correctly for staff to identify.
2. Dispatch items to customers.	Waiters take plates/deliveries out to customers.
3. Confirm order.	Confirm order is received by customers.
Variants:	
2a. Items could be for delivery.	Items to be packed and sealed in a separate area.

Task: 4. Making Reservations Purpose: Allow customers to avoid queues and inform the restaurant how many patrons are expected to arrive. Trigger: Customer calls ahead and asks for a table reservation. Frequency: Occasional (~30 per day). Critical: The customer wishes to book the venue.	
Subtasks:	**Example Solution:**
1. Take reservation.	Ask the customer how many people will be dining and when.
2. Check availability.	Find out if a table is available at that time for that many customers.
3. Negotiate reservation details.	If the table is unavailable, negotiate an alternative with the customer.
4. Input reservation.	Input reservation details into computer system.
Variants:	
2a. Customer books venue.	All seating in the venue must be free on that day. Some payment must be made upon booking the venue.

Task: 5. Creating Statistics Report Purpose: To gather data and make analysis on menu item popularity and usage patterns. Trigger: At the end of each month. OR A statistics report is required prematurely. Frequency: Monthly. Critical: Data loss.	
Subtasks:	**Example Solution:**
1. Retrieve relevant monthly data.	When an order is placed, the system automatically records each menu item along with relevant details such as table, quantity, timestamp, and customer ID. The order is marked as completed when it is finished so that the business knows what the average preparation time is.
2. Perform statistical analysis of monthly data.	System can be made to analyze collected data (e.g. Gather insights such as which menu items' popularity is seasonal.).
3. Visualize monthly data.	System automatically graphs data on the dashboard to help management view performance.
Variants:	
N/A	

Task: 6. Updating Customer Information Purpose: To keep customer details up to date. Trigger: When customer details change. Frequency: Rare (average <1 per year per customer). Critical: Customer no longer lives in the country.	
Subtasks:	**Example Solution:**
1. Inform business about information change.	A customer may use a different phone number to call the business, front-of-house realizes this and notes it down.
2. Verify customer identity.	The customer arrives at the business and pays for their order, this verifies their identity.
3. Update relevant information.	Front-of-house updates the customer's phone numbers on the database to include the new one.
Variants:	
N/A	

6 - Workflow


Reservation
Customer_Arrives
Customer_Orders
Ticket_Created
Order_Fulfilled
Generate_Statistical_Report
Order_Delivered_To_Table
Customer_Picks_Up_Takeout

7 - Quality Attributes of System

When developing the new restaurant information system for the Relaxing Koala restaurant, quality attributes are a major factor for the success and seamless operations for expanding the business. There are several quality attributes we need to look at while being good software architectures. In this section we will be looking at the following quality attributes. Usability, security, reliability, performance, and portability. As the restaurant is making a transition from manual processing to a computer supported system, these attributes are essential in optimizing the efficiency and quality of the services provided by the system.

Security

Security is a paramount issue that concerns the Relaxing Koala restaurant. The new Restaurant Information System (RIS) should consider this. This is because of all the sensitive customer information that would be registered into the new system, such as contact details, phone numbers, bookings, online payments, and card transactions. Breaches and threats to this sensitive will be a major problem for the restaurant, it would cause financial lost, loss of trust and damage to the restaurant's reputation, thus ensuring the security and privacy of this data that the new systems gains are essential.

Therefore, the system must:

- Have access control, giving different users such as managers, waiters, and front-of-house different levels of access to this sensitive information.
- Store all data in a way that complies with Australian privacy law.
- Have strong user authentications and passwords such as multifactor authentication even though it can reduce the useability.
- Have a secure payment system that follows industry standards for things such as online payments and card payments.

Usability

From the case study provided we identified that the Relaxing Koala are using "very low- tech, mostly manual fashion" system within the restaurant. So therefore, the new system we are creating must be simple and easy for the staff and customers to use. The user interfaces we are creating, for e.g., the ordering system, online booking system must be straightforward with

minimum training for staff and customers to use. This would increase efficiency and optimize the daily operation conducted by the restaurant, helping them in their expansion.

Therefore, the system must:

- The user interface must be user-friendly and have consistency for staff and all types of customers to use with ease.
- Have minimum and easy steps for ordering.
- Have tooltips, popups, and accessibility features to help customers/staff to help them in certain scenarios.

Portability:

Portability is crucial for the new RIS as it allows for flexibility and accessibility across various devices, environments, and browsers. From the case study we know that Relaxing koala have the idea of expanding their menu online and having an online ordering system. Portability is a crucial step for this process as there would be different customers with a variety of devices trying to order from the new Online ordering platform seamlessly whether it's through an app or website using a smartphone or laptop. This would enhance customer engagement and improve the overall satisfaction of the customers, moreover it would allow the business itself to be competitive in the market.

Therefore, the system must:

- ✓ Have cross browser capability.
- ✓ Have cross platform capabilities with seamless user interfaces designed to behave identically on any device.

Reliability:

Having a reliable system is crucial for the Relaxing Koala Restaurant, we must ensure that the newly developed RIS system operates consistently and efficiently during the operation hours of the restaurant without fail. This will help minimize disruptions and issues that would arise in the restaurant like delays in ordering, booking malfunctions and system crashes. Therefore, avoiding negative experiences for both the staff and customers themselves. Moreover, a reliable system will be beneficial for the business itself as it would optimize daily operations compared to the existing old-fashioned system.

Therefore, the system must:

- ✓ Be available 100% of the time during operational hours.
- ✓ If any disruptions happen to the system a minimum down time of 5-8 minutes should be the time frame to get the system back in operations
- ✓ The system should be developed in a way that supports scalability, where if the operation wants to grow in the future no disruptions must occur.
- ✓ Have regular checks and tests to make sure the system is fully functioning and there are no faults or errors in the system.

Scalability:

Scalability will be a big factor for the Relaxing Koala restaurant, if they want to keep expanding their business. The newly built RIS system must be able to handle increasing demands in work and requests in a efficient manner without breaking the system. As the restaurant wants to expand even more, the system must be able to scale up to support the growing work without it affecting other quality attributes like reliability and usability.

Therefore, the system must:

- ✓ Be able to scale according to the changing workloads.
- ✓ Be able to consistently monitor the workload that the system is handling.
- ✓ Plan for future growth and expansion of the existing RIS system.
- ✓ Carry out testing to identify and mitigate potential scalability errors/issues.

8 - Other Requirements

In addition to supporting all the tasks outlined in section 4, the system shall meet the following requirements:

8.1 - Product Level Requirements

- ❖ **Performance:**
  - ➢ The system shall respond to user requests within 2 seconds under normal operating conditions.
- ❖ **Reliability:**

- The system shall have an uptime of at least 99% over a one-year period.

❖ **Scalability:**

- The system architecture shall support the addition of at least 20% more customers and orders without significant performance degradation.

❖ **Security:**

- User authentication and authorization mechanisms shall be implemented to ensure that only authorized personnel can access sensitive information such as customer data and financial transactions.

❖ **Data Management:**

- The system shall store and manage data input by users.
- Data entry shall undergo relevant validation procedures.
- Data validation shall adhere to SwinSoft internal data validation standards.

❖ **Information Display:**

- The system shall display relevant information/data upon user request.

❖ **Reporting Functionality:**

- The system shall generate and display relevant reports.

❖ **Document Generation:**

- The system shall print receipts and invoices.

8.2 - Design Level Requirements

❖ **Modularity:**

- The system shall be designed using a modular architecture, allowing for easy addition or modification of features in the future.

❖ **Flexibility:**

- The system design shall allow for customization of menu items, pricing, and other parameters to accommodate changes in the restaurant's offerings.

❖ **Database Design:**

- A relational database management system (RDBMS) shall be employed to store and manage data efficiently. The database schema shall be designed to optimize query performance and data integrity.

❖ **User Interface/Experience (UI/UX) Design:**

- The system shall adhere to the Swinsoft UI/UX design guidelines document for generic design decisions, including font sizing and color contrasts.

❖ **Special Algorithms:**

- The system shall incorporate special algorithms for data analysis.

❖ **Legal Documentation Display:**

- The system shall display Terms of Trade, Terms of Use, and Privacy Policy, and provide the option to download each document.

❖ **Branding Integration:**

- The system shall display the business logo on each screen.

❖ **Design Consistency:**

- The system shall follow the design guidelines of the business, including font styles and color palette.

9 - Validation of Requirements

9.1 - CRUD Check

Task / Entity	Customer	Order	Product	Booking	Invoice	Employee
1. Taking Orders	C, R	C, R	R		C, R	R
2. Preparing Orders		R	R			R
3. Delivering Orders	R	R	R		R	R
4. Making Reservations	C, R			C, R	C	
5. Creating Statistics Reports	R	R	R	R	R	R
6. Updating Customer Information	R, U, D					

10 - Possible Solutions

10.1 - Staff-Oriented Application-Based System

This approach involves staff members utilizing a computer-supported system to streamline restaurant operations while maintaining customer interaction. Key features include:

- Staff-initiated interaction with the system.
- Use of desktop computers and mobile devices (tablets) by restaurant staff.
- Automated processes for reservation management, order taking, and payment processing.
- Customizable options for menu items and customer details.
- Managerial access for generating statistical reports on sales data and trends.

10.2 - Customer-Oriented, Self-Service System

This approach empowers customers to engage with the restaurant's system independently, enhancing their experience and reducing staff involvement. Key features include:

- Self-service kiosks or mobile apps for customers to make reservations, place orders, and view menu items.
- Access to real-time information on table availability, wait times, and order status.
- An option for staff assistance if needed, maintaining a balance between self-service and personalized service.
- Benefits include improved efficiency during peak hours and enhanced customer satisfaction.
- Consideration of initial hardware costs and system responsiveness to ensure a seamless user experience.

10.3 - Cloud-Based Web Application

An alternative solution involves migrating the restaurant information system to a cloud-based platform, offering flexibility, scalability, and cost-effectiveness. Key features include:

- Hosting the system on cloud infrastructure such as AWS or Microsoft Azure.
- Scalability to accommodate fluctuations in demand and ensure optimal performance.
- Pay-as-you-go model, reducing upfront hardware costs and maintenance overhead.
- Integration of online reservation and ordering functionalities accessible via web browsers or mobile apps.
- Enhanced reliability and accessibility, enabling remote management and updates.
- Potential for expanding online presence and offering additional services such as delivery or online payments.

These solutions offer different approaches to modernizing the operations of the Relaxing Koala restaurant, considering factors such as staff efficiency, customer satisfaction, and technological advancements in the hospitality industry.