Swinburne University of Technology

School of Science, Computing and Engineering Technologies

ASSIGNMENT AND PROJECT COVER SHEET

Subject Code: SWE30003 Unit Title: Software Architectures and Design

Assignment number and title: 2, Object Design Due date: 11:59pm, 5th May 2024

Tutorial Day and Time: Wednesday 6.30pm Project Group: Wed 6.30pm – Group 3

Tutor: Mandeep Dhindsa

To be completed as this is a group assignment

We declare that this is a group assignment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

ID Number	Name	Signature
103795587	Jade Hoang	Jade Hoang
103795561	Henry Le	Henry Le
103541023	Thanh Nam Vu	Thanh Nam Vu
103844421	Dang Khoa Le	Dang Khoa Le

Marker's comments:

Total Mark:

OBJECT DESIGN – GROUP 3

Restaurant Information System

Jade Hoang, Henry Le, Thanh Nam Vu, Dang Khoa Le
SWE30003 – Software Architecture and Design

Table of Contents

1. Executive Summary

This document outlines the initial object-oriented design for the Restaurant Information System (RIS) of the Relaxing Koala café/restaurant. The RIS aims to modernise and streamline the day-to-day operations of the restaurant, including reservations, order management, invoicing, online menu access, and basic statistics generation.

Through a Responsibility Driven Design approach, this document includes a list of proposed classes, along with a graphical representation of their relationships in the form of a Unified Modelling Language (UML) class diagram. Additionally, it provides CRC (Class-Responsibility-Collaborator) cards for each chosen class, describing their responsibilities and collaborators.

The design incorporates the use of relevant design patterns and heuristics, which are documented and justified based on their applicability to the RIS domain. Moreover, the document illustrates the bootstrap (or initialisation) process of the system, highlighting the sequence in which classes are instantiated.

As part of the verification process, the document outlines four typical interaction scenarios. These scenarios aim to ensure the alignment of the design solution with the stated requirements.

Overall, this document serves as a comprehensive initial design for the RIS, providing a solid foundation for further development and refinement as the project progresses.

2. Introduction

Relaxing Koala, a growing café/restaurant, requires a robust system to manage its expanding capacity. This report introduces the initial object-oriented design for a Restaurant Information System (RIS) using the Responsibility Driven Design approach. The RIS aims to streamline operations such as reservations, orders, invoicing, and menu management.

This report details the following components of the design:

- **CRC cards**: High-level descriptions of classes and their respective responsibilities and collaborators within the system.
- **Class relationships**: Representation of class relationships through a UML Class Diagram.
- **UML sequence diagrams**: Illustration of four typical scenarios, showing how the design solution handles various system requirements as part of the verification process.
- **Quality of design solution:** Highlight design heuristics and design patterns used in the design.
- **Bootstrap process**: Highlight the sequence of instances of classes created.

3. Assumptions

- Each Order class instance will have a unique orderID assigned to facilitate easy reference and tracking.
- User roles within the restaurant are generalised into one category, "staff," without differentiation. All staff members will have access to the same set of functionalities, eliminating the need to manage various user roles.
- Each delivery will be associated with only one order.
- Staff information is generally not stored in the RIS, although staff will interact with the system.
- When a large group reservation spans multiple tables, the system will simplify the user experience by assigning a single table number or identifier to the reservation.
- A dedicated class will not handle statistical data about menu items. Instead, the system will calculate these statistics dynamically by analysing existing data stored within relevant classes like Order and Invoice. This approach promotes code reusability and avoids creating classes solely for calculations.

4. Problem Analysis

The Relaxing Koala restaurant is facing challenges due to its recent expansion. Their current manual processes for handling reservations, orders, communication with the kitchen, invoicing, and basic menu analysis are no longer efficient or scalable to support a larger capacity.

4.1 Functional requirements

The key functional requirements of the RIS identified in the Software Requirements Specifications (SRS) document are:

- Make reservations
- Place order
- Inform order information to kitchen
- Handle payment
- Generate receipt/invoice
- Provide the ordered menu item's statistic
- Provide an online menu
- Arrange delivery for takeaway orders

4.2 Quality attributes

Key quality attributes (non-functional requirements) for the RIS are also identified, including usability, performance, reliability, security, and scalability. These attributes ensure that the system is user-friendly, efficient, robust, secure, and capable of accommodating future growth and expansion.

4.3 Simplification

The following simplifications were made during the analysis phase of the RIS:

- Third-party integrations and external services are not explicitly represented as classes within this initial design solution.
- The RIS design simplifies user roles, consolidating all staff into one category without differentiation between roles such as waitstaff, kitchen staff, or managers.
- The design simplifies menu management by consolidating both takeaway and dine-in menus into a single class structure, eliminating the need for complex inheritance or separate divisions within the menu classes.
- The design simplifies large group reservations by assigning a single table number or identifier, rather than managing reservations across multiple tables.
- Split bills or partial payments are not allowed, simplifying payment processing to one payment per order.
- The process of sending order information to the kitchen will be simplified to display order details for kitchen staff visibility.

5. Candidate Classes

5.1 Candidate Class List

The following list presents all possible classes for the RIS:

- Restaurant
- Capacity
- Customer/People/Guest
- Kitchen
- Accounting
- Owner
- Staff
- InformationSystem
- Reservation
- Order
- Invoice
- Receipt
- Payment
- Menu
- MenuItem
- Statistic
- Table
- Offerings
- Delivery

5.2 Discarded Classes

- **Statistic**: Menu item statistics can be calculated dynamically by analysing existing data in other classes such as Order and Invoice. This keeps the system focused on its core responsibilities and prevents the proliferation of classes that represent operations rather than fundamental abstractions within the domain model.
- **Offering:** Overlaps with the Menu and MenuItem classes, which already represent the restaurant's available food and beverage options.
- **Capacity:** This is not a standalone entity with distinct attributes or responsibilities. Capacity information can be managed as an attribute within Table class.
- **Accounting:** This can be considered a responsibility rather than a class. Its functionalities can be incorporated into other classes like Receipt or Payment.
- **Owner:** Owner's information is not stored in the RIS and has no responsibilities/functions that need to be performed.
- **Staff:** RIS does not store staff information or require specific staff-related functions. Existing human resources systems can handle staff information and management.
- **InformationSystem:** This represents the RIS itself and is not necessary as a separate class within the system's design.
- **Restaurant:** This is too broad and encompasses the whole system, making it difficult to store information related to it.
- **Kitchen:** The Kitchen class might be unnecessary if the restaurant's kitchen operations are straightforward and do not require detailed management or tracking. Since the case study does not specify any complex kitchen management requirements, the responsibility of informing kitchen staff about orders can be handled by the Order class itself.

5.3 Chosen Classes

- **Customer:** This class is needed to encapsulate essential customer-specific data, including contact details and any dietary preferences/restrictions, and handles responsibilities such as making reservations and placing orders.
- **Reservation:** Represents bookings made by customers, storing information such as reservation details and table assignments while handling responsibilities such as managing table availability.
- **Table:** This class is essential for encapsulating key data related to seating arrangements such as table availability, table number and seating capacity.
- **Order:** Represents requests made by customers for food and drinks, storing information such as items ordered and quantity, and managing order-related functionalities.
- **Invoice:** Invoices need to store information such as the total amount and items ordered while responsible for generating, storing, and calculating invoice details.
- **Receipt:** This class is needed to store transaction details and payment status while managing responsibilities such as formatting and providing proof of payment.
- **Payment:** Even with an external payment gateway, the RIS needs a Payment class to interact with it. This class would handle payment transactions, while storing information about transactions, including payment method, amount, and reference number.
- **Menu:** This class is needed to manage the restaurant's menu, including the ability to add, remove, and update menu items, as well as display the menu for customers.
- **MenuItem:** This class represents individual menu items on the menu, including their names, descriptions, and prices.
- **Delivery:** This class represents the delivery service for takeaway orders and needs to store information such as delivery status, manages the delivery process for takeaway orders, including knowing delivery addresses, coordinating with third-party delivery services, and tracking delivery statuses.

5.4 UML Diagram

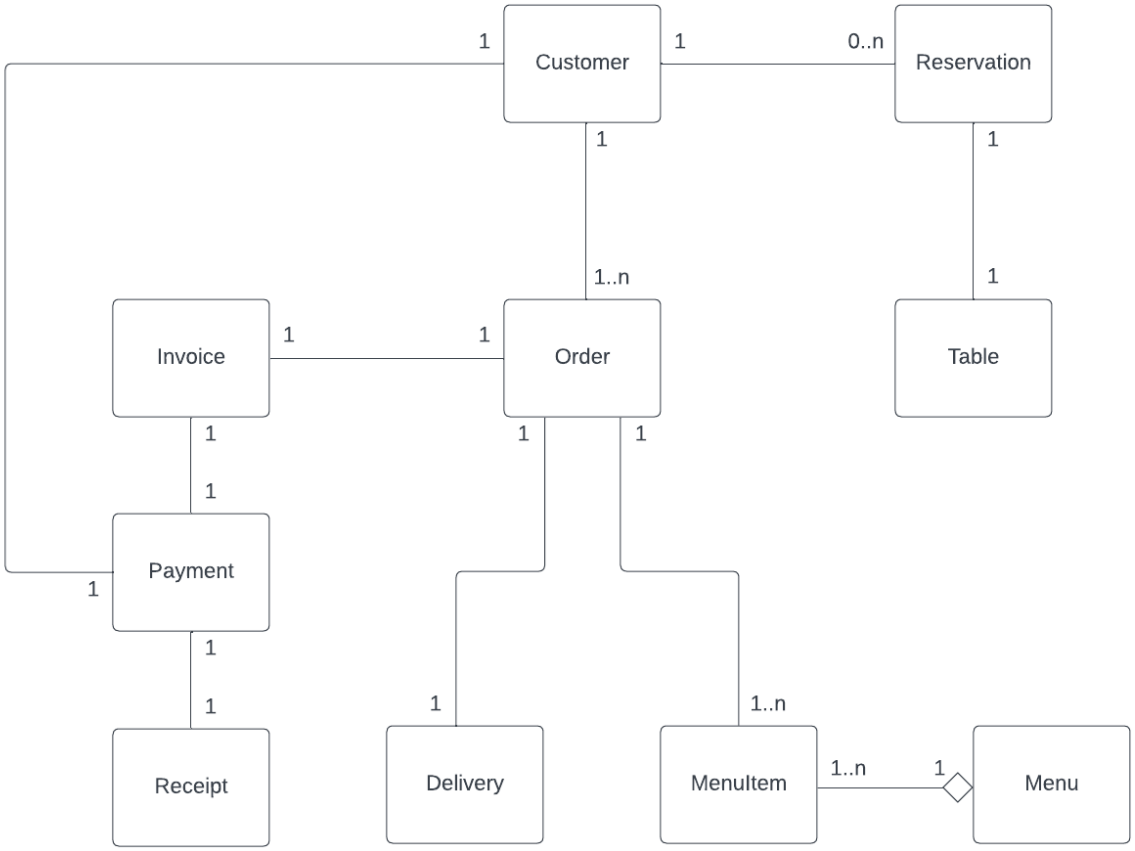

Figure 1 - UML Diagram

6. CRC Cards

6.1 Customer

Represents a customer of the restaurant, including their personal information and any dietary preferences/restrictions.

Class Name: Customer	
Responsibilities	Collaborators
Know customer's name, contact details, dietary preferences	N/A
Make reservation	Reservation
Place order	Order, Menu, MenuItem
Make payments	Payment, Invoice, Receipt

Table 1 - Customer CRC

6.2 Reservation

This class manages reservations made by customers within the RIS, including date, time, party size, and special requests. It ensures proper allocation of tables based on customer requests.

Class Name: Reservation	
Responsibilities	Collaborators
Know date, time, number of guests	N/A
Check table availability for requested date and time	Table
Assign table to reservation	Table

Table 2 - Reservation CRC

6.3 Table

This class manages table arrangements and information such as seating capacity, table number and table availability status.

Class Name: Table	
Responsibilities	Collaborators
Know table seating capacity (i.e., number of seats)	N/A
Know table number (i.e., tableID)	N/A
Update table availability status (occupied or available)	N/A

Table 3 - Table CRC

6.4 Menu

Manages the restaurant's menu, including adding, updating, and removing menu items. The Menu class manages a collection of MenuItem objects.

Class Name: Menu	
Responsibilities	Collaborators
Add menu item	MenuItem
Remove menu item	MenuItem
Update menu item	MenuItem
Display menu	MenuItem

Table 4 - Menu CRC

6.5 MenuItem

Represents an individual menu item within the menu, including its name, description, price, and category.

Class Name: MenuItem	
Responsibilities	Collaborators
Know item’s ID, name, description, category, and price	N/A

Table 5 - MenuItem CRC

6.6 Payment

Payment class manages the financial transactions initiated by customers. It interacts with an external payment gateway and generates receipts for successful transactions.

Class Name: Payment	
Responsibilities	Collaborators
Know payment method	N/A
Know invoice details	Invoice
Forward to external payment gateway	N/A
Generate receipt	Receipt

Table 6 - Payment CRC

6.7 Order

The Order class represents a customer's order within the RIS. It captures the details of the ordered items, their quantities, and facilitates communication with the kitchen.

Class Name: Order	
Responsibilities	Collaborators
Know orderID	N/A
Know special requests (e.g. allergy)	N/A
Know menu items	Menu, MenuItem
Associate a specific menu item with its quantity to the order	MenuItem
Send order information to kitchen	N/A
Track order status	N/A
Submit order details for delivery	Delivery

Table 7 - Order CRC

6.8 Invoice

Represents an invoice generated for a customer order, including order details, totals, discounts, and taxes. It serves as a record of the order details and the financial obligation.

Class Name: Invoice	
Responsibilities	Collaborators
Summarise order details	Customer, Order, MenuItem
Calculate the total amount	Order
Generate invoice	Customer, Order

Table 8 - Invoice CRC

6.9 Receipt

The Receipt class serves as a confirmation document for a successful payment within the RIS. It provides details about the completed transaction.

Class Name: Receipt	
Responsibilities	Collaborators
Know payment type, payment status	Payment
Know transaction details	Payment

Table 9 - Receipt CRC

6.10 Delivery

Represents the delivery service for takeaway orders and manages the delivery process. It acts as an intermediary between the RIS and a third-party delivery service

Class Name: Delivery	
Responsibilities	Collaborators
Know delivery address, delivery instructions	Order
Forward orders to third-party delivery services (to process delivery order)	N/A
Update delivery status	N/A
Track delivery status	N/A

Table 10 - Delivery CRC

7. Design Solution

7.1 Design Heuristic

- A class should capture one and only one key abstraction.
- Keep related data and behaviour in one place.
- Do not create god classes/objects in your system.
- Minimise the number of classes with which another class collaborates.
- Minimise the amount of collaboration between a class and its collaborators (i.e., the number of messages sent).
- Choose the containment relationship (aggregation) over association when given a choice.
- Do not turn an operation into a class.

7.2 Design Patterns

7.2.1 Behavioural Pattern - Observer pattern

Observer pattern is a behavioural design pattern that facilitates communication between objects. When a state change occurs in an observable class, it notifies its registered observer classes. This allows the observers to react accordingly, promoting loose coupling between classes. Within our design for the RIS, the Observer pattern is applicable in both order and reservation management.

Reservation management:

- The **Reservation** class could be observable. Upon making a reservation, the **Table** class, acting as an observer, would be notified. Consequently, the **Table** class could update its availability status to reflect occupied tables.

Order management:

- The **Order** class acts as an observable. Observer classes like **Delivery** and **Customer** could be notified whenever an order is placed, updated, or cancelled.
- **Delivery**: Upon notification, the **Delivery** class schedules delivery for takeaway orders.
- **Customer**: The **Customer** class receives updates on order status, such as "Preparing," or "Ready for Pickup".

7.2.2 Structural Pattern - Composite pattern

Composite pattern allows treating a group of objects as a single object. A composite object can contain other composite objects or leaf objects (i.e., objects without children). This pattern is useful for representing hierarchical structures.

In the context of the RIS, the **Menu** and **MenuItem** classes could be structured using the Composite pattern, where a **MenuItem** is a leaf object, and a **Menu** is a composite object containing other **Menu** objects (representing sub-categories) or **MenuItem** objects. This structure allows the building of a hierarchical menu with categories and subcategories.

8. Bootstrap Process

- The system starts by creating an instance of the **Menu** class.
- **Menu** creates instances of **MenuItem** class.
- Instances of the **Table** class are created to manage table arrangements.
- For each customer interaction, an instance of the **Customer** class is created.
- When a customer makes a reservation, **Customer** creates an instance of **Reservation** class.
- When the customer places an order, **Customer** creates an instance of the **Order** class.
- **Order** creates an instance of **Invoice** class.
- **Customer** creates an instance of **Payment** class.
- **Payment** creates an instance of **Receipt** class for the transaction.
- If the order requires delivery, **Order** creates an instance of **Delivery** class.

9. Verification

9.1 Make a reservation

In this scenario, the **Customer** initiates the reservation process by making a reservation with the **Reservation** class, providing details such as the *date*, *time*, and *party size*. The **Reservation** class then checks table availability by communicating with the **Table** class, which responds with a list of available tables. The **Reservation** class then selects a suitable table and assigns it to itself. Subsequently, the **Table** class updates the availability status of the assigned table. Finally, the **Reservation** class notifies the **Customer** of the successful reservation by providing the *tableID*.

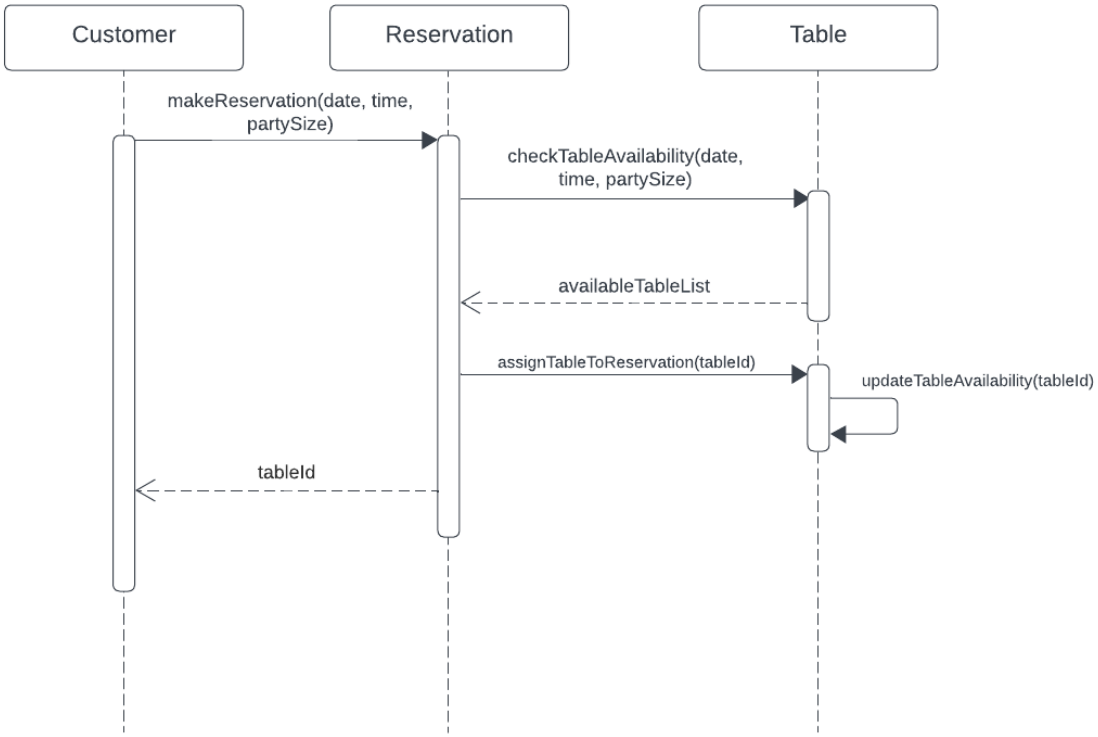

Figure 2 - Make a reservation UML sequence diagram

9.2 Place an order

Initially, the **Customer** initiates the order process by creating an order through the *placeOrder* method, which collaborates with the **Order** class. Subsequently, the **Order** class communicates with the **Menu** class to retrieve a list of available menu items using the *getMenu* method. The **Menu** class then interacts with the **MenuItem** class to retrieve a list of all menu items using the *getMenuItem* method. Upon receiving the menu items list, both the **Menu** and **Order** classes return it to the **Customer**. The **Customer** then adds selected items to the order by invoking the *addItemToOrder* method, specifying the *menuItemID* and *quantity*. Finally, the **Order** class confirms the order by returning the *orderID* to the **Customer**.

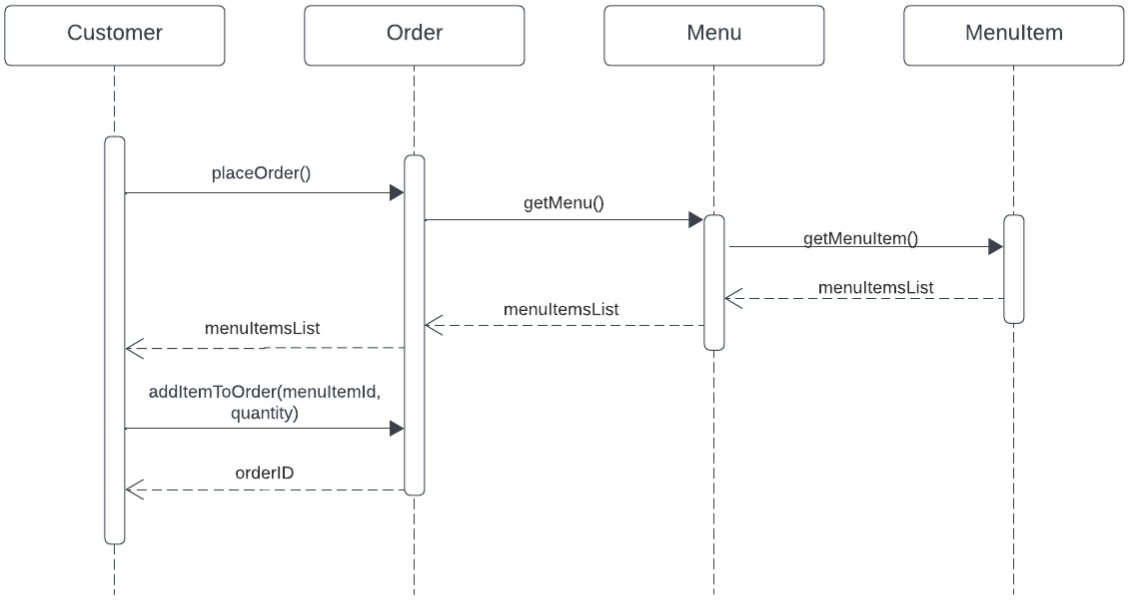

Figure 3 - Place an order UML sequence diagram

9.3 Make payment

To initiate a payment, the **Customer** triggers the process by invoking the *makePayment* method, proving *paymentMethod* and *orderID* to **Payment** class. The **Payment** class then communicates with the **Invoice** class to retrieve the invoice details associated with the order using the *getInvoiceDetails* method, with the *orderID* as a parameter. The **Invoice** class interacts with the **Order** class to gather the necessary order details, returning them to the **Invoice** class.

Using the obtained order details, the **Invoice** class calculates the total amount to be paid and generates an invoice accordingly. This invoice, along with its details, is then returned to the **Payment** class.

Subsequently, the **Payment** class forwards the invoice details to an external payment gateway for processing through the *forwardToExternalGateway* method. Once the payment is successfully processed, the **Payment** class generates a receipt for the transaction. This receipt is provided to the **Customer** by interacting with the **Receipt** class, thus completing the payment process.

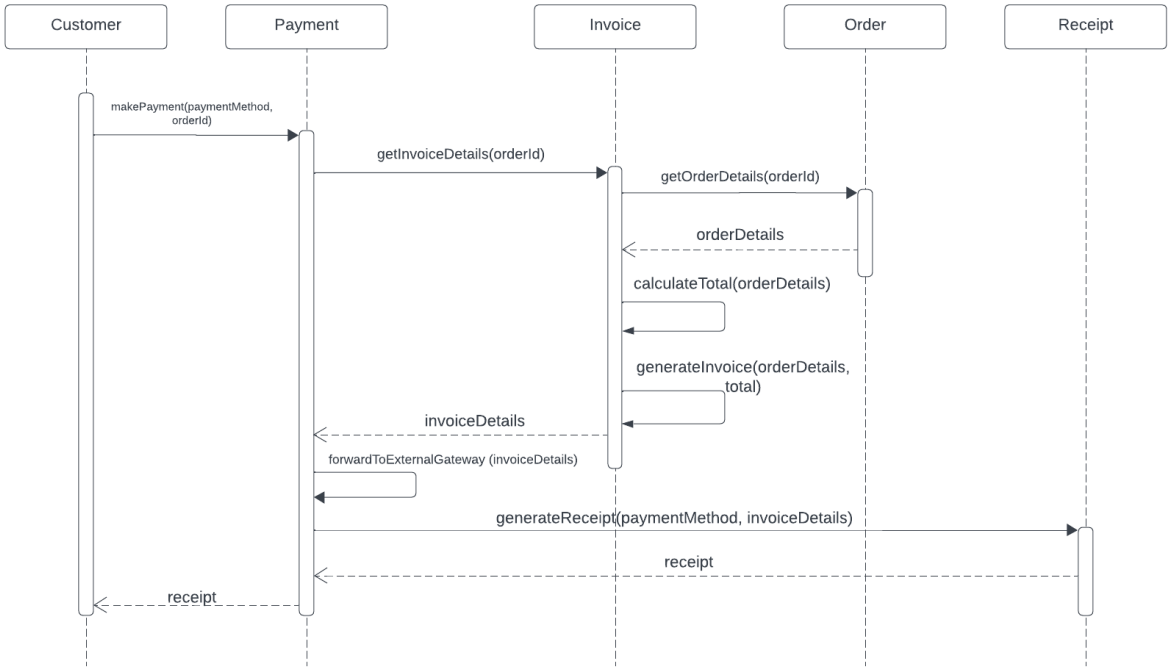

Figure 4 - Make payment UML sequence diagram

9.4 Handle delivery order

In this scenario, the **Customer** initiates the process by calling the *placeOrder* method on the **Order** class, specifying that it is a delivery order by setting the *isDelivery* flag to true. Once **Order** class get the order details (like the previous “Place order” scenario), it communicates with the **Delivery** class to submit order details using the *submitOrderDetails* method. The **Delivery** class then forwards the order details to a third-party delivery service for processing using the *forwardOrderToThirdParty* method. The **Delivery** class also updates the delivery status internally using the *updateDeliveryStatus* method. After processing, the **Delivery** class updates the order status and communicates it back to the **Order** class. Finally, the **Order** class relays the updated order status to the **Customer**.

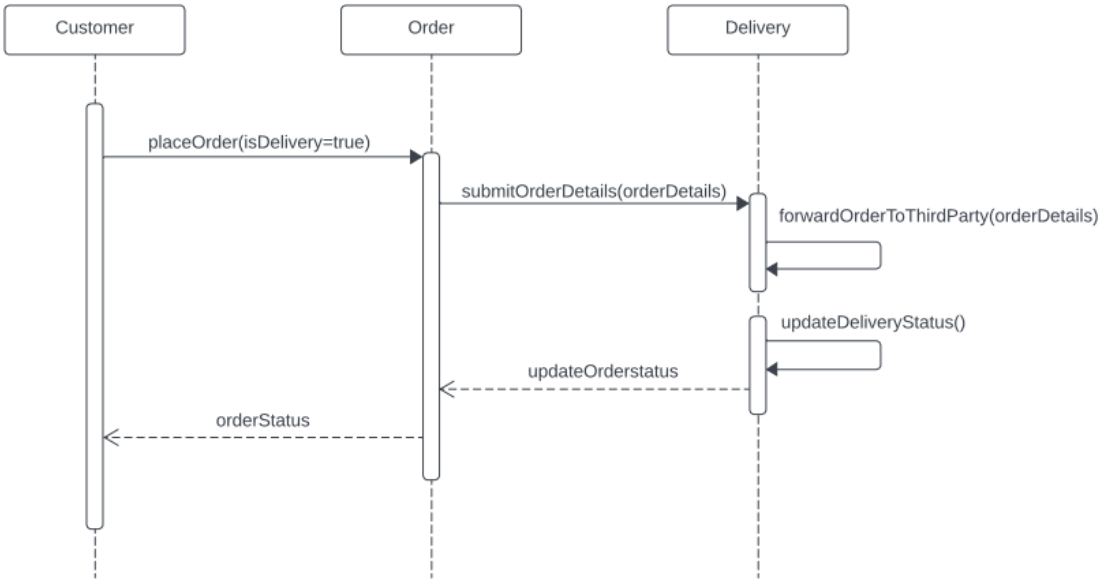

Figure 5 - Handle delivery order UML sequence diagram

10. Appendix

SOFTWARE REQUIREMENTS SPECIFICATION - Group 3

Restaurant Information System

Jade Hoang, Henry Le, Thanh Nam Vu, Dang Khoa Le
SWE30003 – Software Architecture and Design

Table of Contents

1. Introduction

This report is a Software Requirements Specifications (SRS) for a Restaurant Information System (RIS) for the Relaxing Koala restaurant located on Glenferrie Road. The system aims to automate various aspects of the restaurant's daily operations, including reservation management, order processing, order transmission to the kitchen, invoicing, and basic statistical analysis of menu item popularity.

This document includes functional requirements and quality attributes, domain-level requirements, a domain model diagram, and a workflow chart. The functional requirements will be broken down into user tasks and follow the Task and Support approach. Additionally, it covers other requirements such as product and design-level requirements, along with evidence of validation and verifiability.

2. Project Overview

The owners of Relaxing Koala have identified a need to automate some of their operations due to recent expansion. Their current manual processes for order management, kitchen communication, and accounting are no longer viable. To address this, they aim to implement a RIS, developed by Swinsoft Consulting, capable of handling reservations, orders, invoices, payments, and providing menu statistics. The overarching goal is to enhance efficiency, customer service, and scalability.

2.1 Domain Vocabulary:

- **Customer**: Individuals who visit Relaxing Koala to dine and make purchases.
- **Delivery**: Service of transporting orders to customers' locations.
- **Invoicing**: Generating bills for the orders placed by customers.
- **Menu statistics**: Gathering data on customer orders and preferences to analyse the popularity of menu items.
- **Online menu**: Digitally accessible version of the restaurant's menu that lists food and beverage options available at the restaurant.
- **Take-away menu**: Menu displayed for customers ordering food to be taken off-site.
- **Order transmission to the kitchen**: System for transmitting orders from the front-end to the kitchen staff.
- **Payments**: Processing transactions for the orders and services used by customers.
- **RIS**: Restaurant Information System, the proposed system aimed at assisting in the daily operations of Relaxing Koala.
- **Website**: Front-end service promoting the restaurant, accessible to the customers.

2.2 Goals:

- Streamline restaurant operations to improve efficiency and reduce manual efforts.
- Enhance customer experience, service quality, and accessibility through efficient reservation, ordering, and payment systems.
- Lay the groundwork for future scalability and increased capacity of the business.

2.3 Assumptions:

- The system integrates with an external payment gateway for managing customer payments online.
- Delivery services are outsourced to third-party providers. Therefore, RIS is responsible for arranging delivery but not managing the delivery drivers or logistics.
- If dine-in, customers pay for their meals after they have finished dining, allowing them to place multiple orders online during their dine-in experience and settle the bill once at the end of their visit.
- The system generates digital invoices or receipts and sends them to customers via their provided email address or phone number for convenience and accessibility.

2.4 Scope:

The project focuses on developing an information system for Relaxing Koala café/restaurant currently undergoing expansion. The information system should enable staff to handle reservations, take customer orders, transmit orders to the kitchen, generate invoices and receipts, and manage payments. Additionally, basic statistics about ordered menu items will be provided. The system will also make menus available online, allowing customers to view offerings, place orders for takeaways, and potentially arrange delivery.

2.5 Constraints:

The new system needs to integrate with or replace the existing low-tech, manual processes currently in place at the Relaxing Koala, which may pose challenges in terms of data migration, staff training, and process adaptation.

3. Problem Domain

3.1 Pain Points

- Manual order taking and processing.
- Low-tech accounting procedures.
- Can only support up to 50 customers.
- Lack of an online menu for informing potential customers about offerings.

3.2 Actors

- **Customer**: End-user uses the RIS to browse menus, place orders, make reservations, and make payments
- **Waiter**: Takes orders from customers (if not ordered directly through the system) and inputs them into the RIS. They may also access table information and reservations.
- **Cashier**: Handles payments if the RIS does not manage them directly. They might interact with the system to record cash or other offline payments.
- **Manager**: Use the RIS to access ordered menu item statistics.
- **Kitchen staff**: Receive order information and prepare the food accordingly.
- **External payment gateway**: Interact with the RIS to process customer transactions securely.
- **Third-party delivery service**: Receive order details from the RIS and handle deliveries to customer addresses.

3.3 Tasks

- Make reservations
- Place order
- Inform order information to kitchen
- Handle payment
- Generate receipt/invoice
- Provide the ordered menu item’s statistic
- Provide an online menu
- Arrange delivery for takeaway orders

4. Domain Model

4.1 Domain Entities

- **Customer**: Individuals who visit the restaurant to dine in, order takeaway, or request delivery services.
- **Order**: A record of selected menu items for consumption or delivery.
- **Menu**: Digital/physical listing of food and beverage offerings.
- **Reservation**: Booking for a specific table at a particular date/time.
- **Payment**: Financial transaction associated with an order.
- **Receipt**: Document summarising order details and payment upon purchase.
- **Invoice**: Detailed bill presented to the customer, often for takeaway orders.
- **Staff**: Restaurant employees who interact with the system.
- **Statistic**: Aggregated data from orders and menus for insights.
- **Table**: Physical dining area with seating capacity and location.
- **Kitchen**: Prepares food based on order information.
- **Delivery**: External service that delivers orders to customers.

4.2 Domain Model

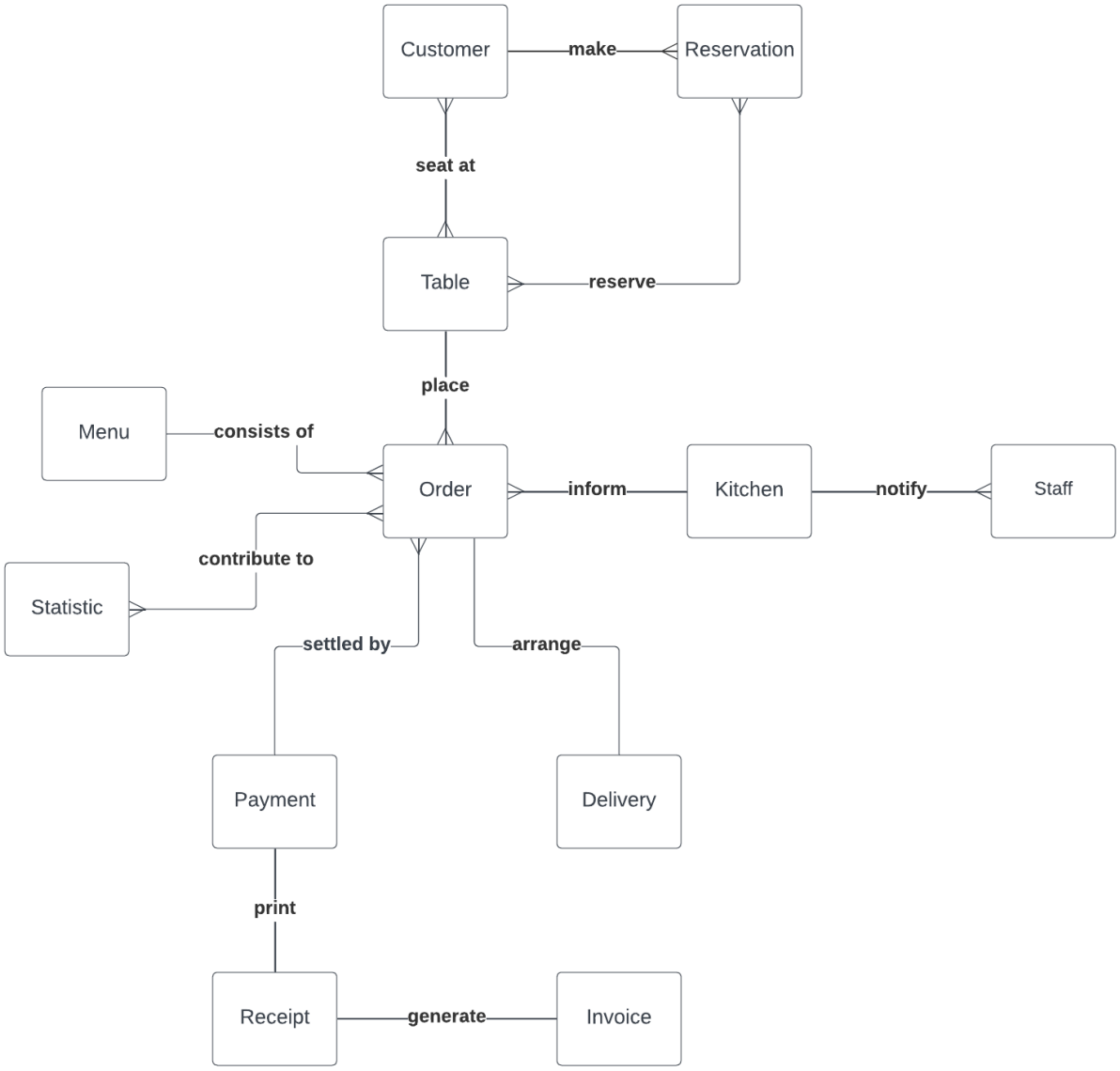

Figure 3 - RIS domain model

5. User Tasks

5.1 Make Reservation

Task:	Make Reservation
Purpose:	Allow customers to reserve a table online for a specific date and time. The system records the reservation details, marking the table as unavailable for that timeframe.
Trigger/Precondition:	Customers want to dine in without waiting for their seats.
Frequency:	Average 15 times/day on weekdays and 45 times on weekends.
Critical:	Influx of reservation requests during peak hours.
Sub-tasks:	**Example solution:**
1. Select preference time	System shows a list of timeslots to choose from.
2. Enter the number of customers	System displays a numeric input field labelled "Number of Customers" on the reservation interface.
3. Check availability	Determine the availability of tables based on the requested date, time, and party size.
4. Select table	System shows the table location on a floor plan, allows the customer to choose a specific table based on their selection.
5. Record customer information	System shows a list of input fields for customers to enter their details.
6. Confirm reservation	System shows the reservation has been successfully made message.
Variants	
3a. No available tables for the requested date and time.	The system informs the customer that no tables are available at the desired date and time. It suggests alternative options (e.g., waiting list, recommending closest time slots).
6a. Customer wants to modify their reservation	System offers alternative dates and times based on availability.
6b. Customer wants to cancel existing reservation	System prompts for reason (e.g., change of plans, emergency) and updates reservation status accordingly.

Table 11 - Make Reservation User Task

5.2 Place Order

Task:	Place Order
Purpose:	Allow customers to browse the restaurant menu, select items, and submit their order electronically.
Trigger/Precondition:	A customer visits the restaurant's physical location or accesses the restaurant's website and decides to order food for dine-in, delivery, or pickup.
Frequency:	Approximately 200 orders/day
Critical:	Large orders during high-volume periods.
Sub-tasks:	**Example solution:**
3. Access website	If dine-in, place QR codes at prominent locations on each table and direct customer to restaurant's order website upon scanning the QR code with a smartphone camera.
4. Browse menu **Problem:** Difficulty visualising dishes.	- System provides a digital menu accessible on the website. - Menu includes high-quality images and descriptions of dishes.
5. Select item **Problem**: Customer accidentally adds an incorrect item to their cart	- System offers a user-friendly interface for adding and removing items from the cart. - Provides clear visual confirmation of selected items before checkout. - System includes a "Remove" button next to each item in the cart and displays a confirmation prompt before finalising the order.
6. Customise order **Problem:** Customer has dietary restrictions (allergy, religious) and needs to customise multiple menu items.	System allows customisation options (e.g., remove sauce, extra vegetables, allergy) and a "Special Requests:" input field for each item.
7. Choose order type (Delivery/Pickup/Dine-in)	- System defines clear options to choose between "Dine-in" or "Delivery/Pickup" at the end of the ordering process. - If "Dine-in" is selected: System proceeds to the "Review and Confirm order" page. - If "Delivery" is selected: System provides an address selection process for delivery orders. - If "Pickup" is selected: Provide clear instructions and options for pickup orders (e.g., estimated waiting time).
8. Review and confirm order	- System displays a comprehensive order

	summary page for review (items, prices, and type with details). - Allows customers to edit or cancel items before finalising the order.
9. Submit order	System transmits the order to the kitchen and provides an immediate confirmation with an estimated preparation or delivery time.
Variants	

Table 12 - Place Order User Task

5.3 Inform Order Information to Kitchen

Task:	Inform order information to kitchen
Purpose:	Transmit the order to the kitchen staff for preparation
Trigger/Precondition:	Customer successfully places an order online through the restaurant's website.
Frequency:	Approximately 200 orders/day
Critical:	Influx of orders during peak hours
Sub-tasks:	**Example solution:**
1. Receive order details	System captures order details, including menu items, quantities, special requests, and table numbers.
2. Manage order queue	System organises orders in a queue based on order time or preparation time.
3. Display order on the screen	System displays incoming orders on the kitchen display screens
Variants	
1a. Special requests or instructions in the customer's order	System clearly displays any special instructions included in the customer's online order.

Table 13 - Inform Order to Kitchen User Task

5.4 Handle Payment

Task:	Handle Payment
Purpose:	Process financial transactions for orders placed by customers.
Trigger/Precondition:	Customer has finished their meal or confirmed their delivery/pickup order.
Frequency:	Approximately 200 times/day.
Critical:	
Sub-tasks:	**Example solution:**
1. Identify the table's bill (if dine-in) or identify customer's orders	System allows staff to search for a customer's bill by table number and displays a clear summary of all online orders placed for that table during the visit.
2. Review order details	System provides a detailed breakdown of each online order, including item name and quantity, price per item, etc.
3. Calculate total amount	System computes the total amount owed by the customer, including all orders and applicable taxes or fees.
4. Choose payment method	System shows available payment types such as: debit cards, credit card (Visa, Mastercard, etc.), and bank transfer.
5. Process payment	System interacts with the external payment gateway to securely process customer transactions
Variants	
3a. Customer requests to split the bill	System allows splitting the bill by item or equally amongst diners at the table. It recalculates the total amount for each diner accordingly.

Table 14 - Handle Payment User Task

5.5 Generate Receipt/Invoice

Task:	Generate receipt/invoice
Purpose:	Provide customers with transaction proof.
Trigger/Precondition:	Customers made payment
Frequency:	Average 200 times/day.
Critical:	
Sub-tasks:	**Example solution:**
1. Summarise order details	System automatically retrieves all relevant order information of the customer during their visit.
2. Generate invoice/receipt	System generates a clear and well-formatted electronic invoice/receipt
3. Deliver invoice/receipt **Problem**: Customer might not receive the invoice/receipt due to email delivery issues or incorrect phone number entry.	- System allows customers to choose their preferred delivery method (email or SMS with a link to the invoice). - System confirms customer-provided email addresses or phone numbers before sending the invoice/receipt. - Implements retry mechanisms for failed email deliveries.
Variants	
3a. Customer requests physical receipt.	System provides an option for customers to request a printed receipt and print the physical copy of the invoice/receipt.

Table 15 - Generate Receipt/Invoice User Task

5.6 Provide Ordered Menu Item Statistics

Task:	Provide ordered menu item statistics
Purpose:	Provide a clearer understanding of the types of menu items customers are ordering
Trigger/Precondition:	Customer completes an order, and the transaction is finalised
Frequency:	Daily, weekly, monthly or on-request
Critical:	
Sub-tasks:	**Example solution:**
1. Collect order data	System automatically collects data on all menu items included in fulfilled orders
2. Aggregate and analyse data	System aggregates order data to generate reports on the most popular menu items, least popular menu items, etc.
3. Generate reports	System generates reports in a user-friendly format (e.g., tables, charts) for easy analysis.
4. Save reports	System allows saving generated reports in a designated folder within the restaurant's database or cloud storage.
Variants	
4a. Manager wants to export reports in various formats	Provides options to export generated reports in different formats such as CSV or PDF.

Table 16 - Provide Menu Item Statistics User Task

5.7 Provide Online Menu

Task:	Provide online menu
Purpose:	Present the menu online and inform potential customers about the restaurant's offerings
Trigger/Precondition:	A customer visits the restaurant's website.
Frequency:	Always online and accessible
Critical:	
Sub-tasks:	**Example solution:**
1. Upload menu items **Problem**: Staff might upload inaccurate/outdated menu information	- System allows staff to upload the restaurant's menu items (including images, descriptions, prices, etc.) to the website. - System allows staff to edit or update the existing menu items information.
2. Display menu content	System renders the complete and up-to-date menu on the website.
Variants	
2a. Customer accesses the online menu on mobile/tablet	System implements a user-friendly and responsive website design.

Table 17 - Provide Online Menu User Task

5.8 Arrange Delivery for Takeaway Orders

Task:	Arrange Delivery
Purpose:	Facilitate customer takeaway orders and send them to third-party delivery services.
Trigger/Precondition:	Customer places a takeaway order and chooses the delivery option.
Frequency:	Average 40 times/day
Critical:	
Sub-tasks:	**Example solution:**
1. Receive delivery order	System accepts the order from the customer to initiate delivery with the third-party provider.
2. Order forwarding to third-party service provider	System automatically forwards takeaway orders with delivery selections to the chosen third-party providers.
3. Notify the customer regarding order dispatch	System implements automated email or SMS notifications to keep customers informed throughout the delivery process
Variants	

Table 18 - Arrange Delivery User Task

6. Workflows

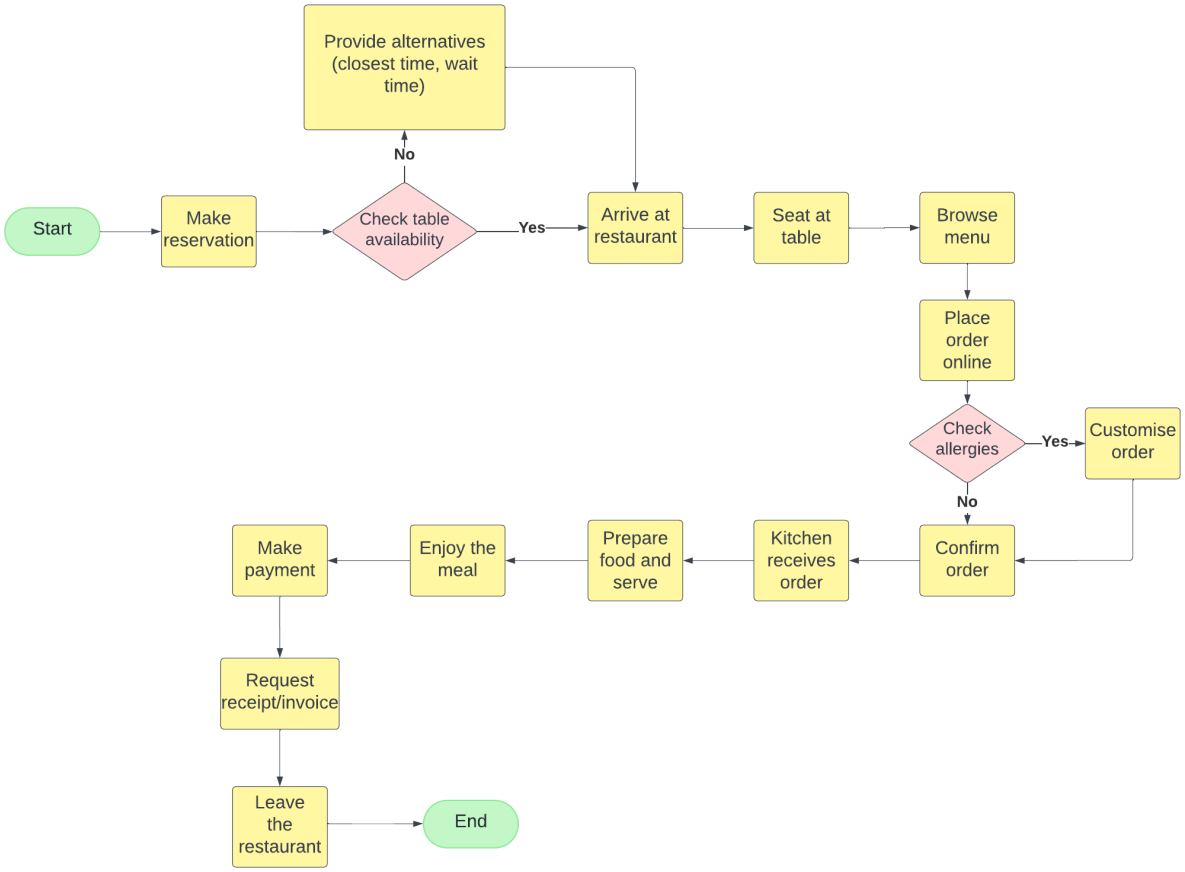

Figure 4 – RIS's Workflow

7. Quality Attributes

7.1 Usability

Usability is about how easy it is for users to learn, navigate, and interact with the RIS to achieve their goals (e.g., making reservations, placing orders, and viewing menu items).

This quality attribute is important because a user-friendly RIS allows customers of all ages, disabilities, and technical backgrounds, without any technical know-how to easily make reservations or place orders online without confusion or frustration. This is crucial for encouraging repeat visits and retaining a broad customer base for Relaxing Koala. Additionally, restaurant staff should be able to perform their tasks efficiently, minimising training requirements and reducing the risk of errors.

7.2 Performance

Performance implies the system's ability to respond quickly and handle a high volume of transactions and requests without significant delays or bottlenecks.

Fast performance is crucial for customer satisfaction, allowing customers to make reservations, and make payments quickly and efficiently. On the other hand, slow loading times or response times can lead to frustration and customers abandoning reservations or orders midway.

7.3 Reliability

Reliability means that the RIS is able to function correctly and consistently without failures or outages, even under varying loads and conditions.

The RIS will be responsible for critical business operations, such as order management, payment processing, and reservation handling. Any system failures or data loss could result in significant disruptions to the restaurant's operations, leading to customer dissatisfaction, lost revenue, and damage to the business's reputation.

7.4 Security

Security ensures that the RIS is protected against unauthorised access, data breaches and other malicious threats.

The RIS will handle sensitive customer information, such as contact details and payment data. Therefore, ensuring the security of this information is crucial to maintaining customer trust and complying with relevant data protection regulations. The software must be designed to prevent unauthorised access and store sensitive data with proper protection, ensuring data confidentiality and integrity.

7.5 Scalability

Scalability is the ability of the system to handle increasing workloads and accommodate future growth or expansion of the restaurant's operations.

As The Relaxing Koala has already expanded its capacity, the RIS should be designed with flexible scalability in mind. The system should be able to adapt to potential future growth, such as additional locations, increased customer volumes, delivery and the integration of new features or services without significant disruptions, significant cost adjustment or the need for complete system replacements.

8. Other Requirements

8.1 Product-level requirements

- The system shall provide a feature for customers to make reservations online.
- The system shall enable order placement and management for customers and staff.
- The system shall facilitate payment processing for customer orders.
- The system shall generate invoices and receipts for customer orders.
- The system shall track and provide menu item statistics.

8.2 Design-level requirements

- The system shall have separate user interfaces and access levels for customers, waiters, kitchen staff, and administrators.
- The reservation management interface shall provide a calendar view for staff to visualise and manage reservations effectively.
- The online ordering interface shall have an intuitive shopping cart, with options to add, remove, or modify items, and display a total amount before checkout.
- The website interface shall incorporate the business logo on every page to reinforce brand identity.

9. Validation

During the creation of the requirements specification for RIS, various validation steps were conducted to ensure the system aligns with Relaxing Koala's needs. Firstly, reviews and walkthrough sessions with key stakeholders (owners, managers, staff) to validate requirements against their needs and expectations.

This process includes:

- Identify and go through the user tasks outlined in the case study.
- For each task, brainstorm, and document at least 2-3 alternative solutions that could achieve the desired outcome.
- Reviewing existing task descriptions and assessing if they can be applied to each of the identified solutions. Refine the task descriptions if needed.

Furthermore, as part of the validation, a CRUD (Create, Read, Update, Delete) check was performed to validate that the system adequately supported these fundamental operations as outlined in the requirements.

Entity / **Task**	**Customer**	**Menu**	**Reservation**	**Payment**	**Staff**	**Order**	**Statistic**	**Invoice/ Receipt**	**Delivery**
Make reservations	C R U D		C U		R				
Place order	C R D	R				C R			
Inform order information to kitchen					R	R			
Handle payment	R U			C R	R	U		R	
Generate receipt/invoice	R			C R U	R	R U		C R	
Provide ordered menu item's statistic		R U			R	R	C R		
Provide online menu		R			R U				
Arrange delivery for takeaway orders					R	R			C R

Table 19 - CRUD Check

Note: "C" denotes Create, "R" denotes Read, and "U" denotes Update, "D" denotes Delete.

10. Verifiability

Verifiable requirements:

All identified functional requirements are verifiable through testing:

- **Make reservations**: perform a series of test cases to make reservations for different dates, times, and party sizes, including edge cases like fully booked scenarios or invalid inputs.
- **Place order**: test cases to verify a complete order placement process, from menu browsing, item selection, customisation, and order submission and measure the order success rate, error rates, response times, and accuracy of order data stored in the database.
- **Inform order information to kitchen**: place orders through RIS and verify that all orders are correctly displayed in the kitchen staff interface within a specified timeframe (e.g., 2 minutes).
- **Handle payment**: test the payment process using different payment methods for a range of order totals and scenarios (e.g., partial payments), record and measure the payment success rate and error rates.
- **Generate invoice/receipt**: generate receipts/invoices for a diverse set of order scenarios, including different item combinations, discounts, taxes, and payment methods. Verify the accuracy of receipt/invoice content, and formatting.
- **Provide ordered menu item statistics**: simulate a defined period of order data (e.g., 1 month) with known order patterns and menu item popularity. Generate reports and statistics for ordered menu items and verify the accuracy, completeness, and consistency of the reported data against expected values calculated from the simulated data.
- **Provide online menu**: conduct cross-browser and cross-device testing to verify the accessibility, responsiveness, and usability of the online menu interface and measure load times.
- **Arrange delivery for takeaway orders**: place a set of takeaway orders with delivery, simulating various delivery locations (e.g., nearby, far away, invalid addresses). Verify the accuracy of captured delivery addresses, order details, and successful delivery arrangement as well as their timeliness.

The majority of quality requirements can be verifiable:

- **Usability**: test with a sample group of users with varying technical skills (beginners, intermediate, advanced) attempting to make online reservations, browse the menu, and measure the time it takes them to complete tasks and collect user feedback through surveys and interviews. Using UI/UX tester tools (e.g., SolarWinds, testsigma) to detect problems and evolve users' need at early stages.

- **Performance**: use performance testing tools to measure page loading times and system response times for various user tasks.
- **Security (partially)**: simulate a hacking attempt on the RIS database to verify the system's ability to prevent unauthorised access attempts. Penetrates cybersecurity tester tools (e.g., Wireshark) to detect vulnerabilities in website's operation.

Non-verifiable requirements:

While many requirements can be verified through testing or analysis, the following requirements may not be fully verifiable due to external factors:

- **Security (partially)**: while security can be partially verifiable through penetration testing and security audits, this quality attribute depends on the evolving nature of external threats and potential hackers.
- **Reliability**: the system needs to encounter real-world usage over an extended period, and experience various usage scenarios and edge cases to fully verify reliability requirements.
- **Scalability**: this requirement cannot be tested until the system is deployed and experiences actual usage at scale. Therefore, its verifiability may be limited until real-world usage data is available.