



Catering Application

Author:

Yijie Chen
Jianing He
Muyao Li
Yangzi Li
Tania Turdean
Yaojue Xiong
Wenjin Yang

Author Email:

ucab171@ucl.ac.uk
jianing.he.22@ucl.ac.uk
muyao.li.22@ucl.ac.uk
yangzi.li.22@ucl.ac.uk
tania.turdean.22@ucl.ac.uk
yaojue.xiong.22@ucl.ac.uk
wenjin.yang.22@ucl.ac.uk

COMP0071: Software Engineering

March 22, 2023

Department of Computer Science
University College London

Contents

1	Introduction and Problem statement	3
1.1	Report Introduction	3
1.2	Project Background	3
1.2.1	Project Brief	3
1.2.2	Problem Statement	4
1.2.3	Project Scope	4
1.3	Project Approach	5
1.3.1	Development Process	5
1.3.2	Gantt Chart	5
1.4	Glossary	6
2	Requirements	7
2.1	Overview	7
2.2	Similar Product Analysis	7
2.3	MoSCoW Style Requirements	8
2.4	Domain Model	13
3	Use Cases	14
3.1	Overview	14
3.2	Use Case List	14
3.3	Use Case Diagram	16
3.4	Use Case Specifications	17

4	Object-Oriented Analysis and Design models	32
4.1	Overview	32
4.2	Object-Oriented Analysis	32
4.2.1	Class diagram	32
4.3	Object-Oriented Design	35
4.3.1	Class Diagram	35
4.3.2	Component Diagram	36
4.4	Deployment Diagram	37
4.5	Activity Diagrams	38
4.6	Sequence Diagrams	40
4.7	State Machine Diagrams	45
5	Application Mock-up	47
5.1	Overview	47
5.2	Mock Up	47
5.2.1	Use Case 1 - CustomerRegistration (Client)	47
5.2.2	Use Case 2&3 - CustomerLogin & AdminLogin (Clients&Admin)	48
5.2.3	User Case 7&8 - SeeOrderStatus & SeePreviousOrders (Clients)	49
5.2.4	Use Case 13 - EnactmentDelivery (Client)	50
5.2.5	Use Case 14 - EditPromotion (Admin)	50
5.2.6	Use Case 19 - ManageCustomerAccount (Admin)	51
5.2.7	Use Case 22 - OrderFoodFromOtherProviders (Admin)	51
6	Summary and Conclusions	52
6.1	Summary	52
6.2	Conclusion	52
	References	53
	Appendix A: Individual contribution summary	54

1 Introduction and Problem statement

1.1 Report Introduction

Our project aims to design takeaway software like Deliveroo, mainly for the university and the admin (catering company). The software needed to be web and app-based, flexible, scalable, easy to maintain and manage, and to meet design principles such as reliability, robustness, modifiability, ease of understanding, simplicity, testability, efficiency, standardization, sophistication, extensibility, and security.

For the university module, users need to be able to log in and register, create personal food preferences, search for food and filter food with filters on the main page, add food to the shopping cart, checkout with different payment options such as PayPal; after checkout, the user needs to receive an email confirmation (price, food ordered, date, etc.); the account page needs to see the past orders and export the PDF invoice information. For the administration module, the administrator needs to be able to see all users, e.g., what food was ordered on which day; to manage discounted fare, promotions, and food stocks; and order food for third parties; if the ordering period is one week, the order must be taken and prepared, and if the period is less than one week and there are not enough ingredients, a failed order message should be displayed.

Our report is divided into the following modules: *Introduction and Problem statement; Requirements; Use cases; Object-Oriented Analysis and Design models; Application models. Oriented Analysis and Design models; Application Mock-up; Summary and Conclusion; Who did What Summary and Appendices.* The final project report is completed to ensure the project's successful implementation and market launch.

1.2 Project Background

1.2.1 Project Brief

The purpose of developing this software is to provide universities with food order convenience. While the project brief is provided here, specific requirements would be introduced in the Chapter 2, which contains a MoSCoW to list all the compulsory and potential functions.

Brief: *An application to allow universities to order food during events such as conferences and workshops. The catering company offers a variety of menus and reductions, but also the possibility to deliver orders at specific dates and times. Universities can create a profile so they can keep track of their orders and preferences (vegetarian, vegan, etc). They can also export detailed invoices of their orders.*

1.2.2 Problem Statement

The universities are looking for an efficient method to order food from companies to prepare for the academic events beforehand.

Organizing food by universities during significant occasions is necessary to provide a satisfying and thoughtful circumstance for participants. However, the order tends to be huge and requires the merchants to purchase and cook in advance. Lacking of multi-functional ordering application leads to time-consuming and confusing experience for both universities and merchants.

To address the chaotic situation, an application is going to design for universities to purchase food from catering company easily. The most basic problem is how to provide a wide range of menus, reductions, and delivery options, which gives university staffs the possibilities to arrange well-pleasing feast. Another important aspect to consider is simplifying the ordering process and allow universities tracking the previous records. The application needs to be full-featured but concise.

1.2.3 Project Scope

This project aims to provide universities with streamlined and efficient ordering procedures to purchase food from companies. It considers the current situation listed in problem statement and thus the main users defined by this application are universities and catering companies.

- **University:** The main purpose of universities is ordering food conveniently. After registration and login, they could create personal profile to describe dietary preference. The main page are used for searching the preferred food and applying filters to select specific categories. The baskets allow university staffs to choose, change quantity of, and cancel food, as well as seeing the total price. When proceeding to checkout, universities can pick certain date and time to arrange delivery, and choose to use either card(Visa, MasterCard, etc.) or third party payment method. Once the booking has completed, the email confirmation for the order information would be automatically sent. Universities are also capable of visual the previous orders and export the relevant invoices.
- **Catering Company:** The basic duty of catering companies is preparing food and providing satisfying delivery services. They must be able to see all customers and their order history, including the ordered food and proposed delivery dates. In addition, the companies would manage the discounted food and edit promotions, which are displayed on the search page to attract customers. They are also required to create an inventory system that allows to check food availability, and if necessary, seek for external food providers to supply ingredients. One week in advance is necessary for companies to purchase ingredients from third party, otherwise the food should be unavailable to order on website.

1.3 Project Approach

1.3.1 Development Process

In the process of project development, a certain "agile" method is used to enhance teamwork, self-organization, continuous iteration and rapid feedback, so as to improve the efficiency and quality of software development, and also help to respond to market changes and customers more quickly demand, improve product quality and customer satisfaction (Schwaber, 2005).

The details of the application are as follows:

- **Sprint planning:** A sprint planning meeting is held, and each meeting selects some tasks with higher priority and assigns them to the next stage for development in the next sprint. Break stories down into smaller, actionable tasks that can be done during a sprint.
- **Daily Scrum:** A short Scrum meeting where each team member shares progress, obstacles, and plans for the day. This helps keep the team aligned and on track to complete the sprint goals.
- **Sprint review:** At the end of the sprint, a sprint review meeting is held with stakeholders. The team presents the completed work and collects feedback at the meeting, and after the meeting reflects on the effectiveness of the work in the previous stage and what can be improved for the next sprint.
- **User Testing:** Conduct user testing throughout the development process, collect feedback and make improvements based on user feedback.
- **Improve Product Backlog:** Improve the Product Backlog at the end of each sprint, incorporating feedback from sprint reviews and retrospective meetings.
- **Continuous Improvement:** Continuously collect user and stakeholder feedback and use this feedback to inform improvements and new features in future sprints.

1.3.2 Gantt Chart

Figure 1 shows our Gantt Chart. In order to ensure that the application meets the needs of customers, we will spend a week exploring the requirements and formulating MoSCoW in the early stage. The next two weeks will focus on analyzing user case lists and drawing various diagrams. Based on the above, we will spend a week on prototype production.

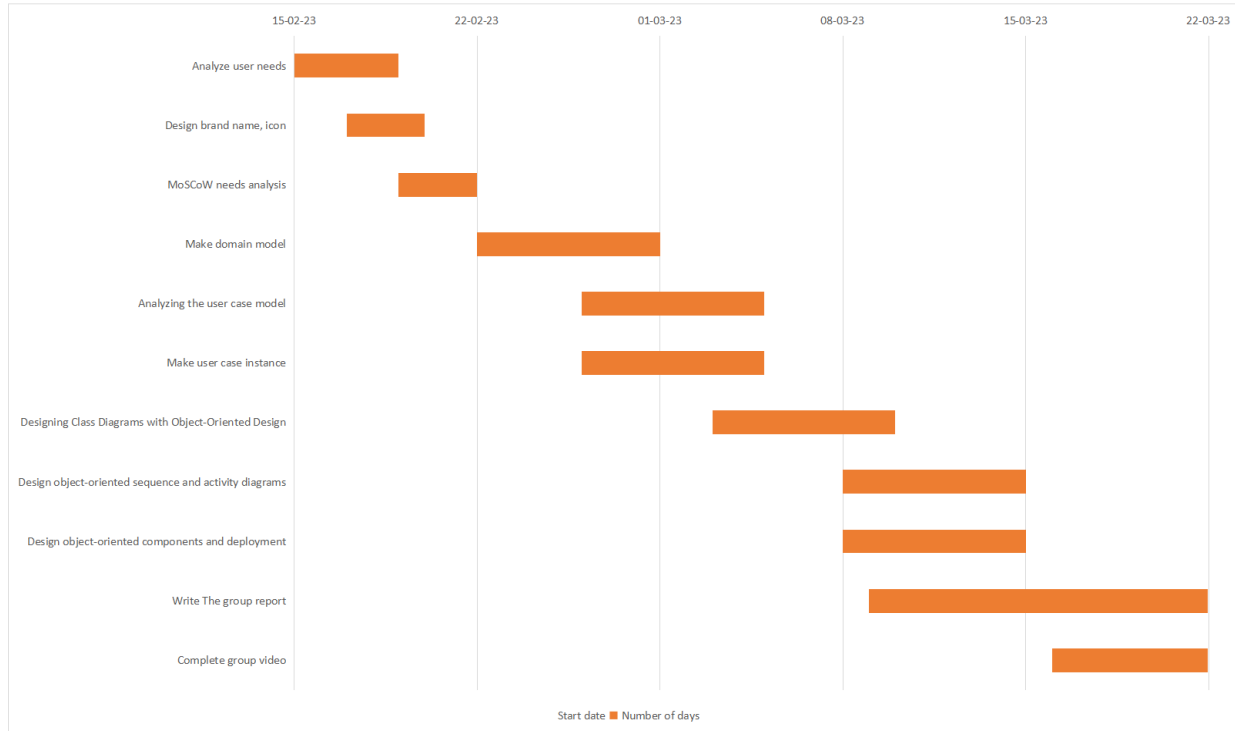


Figure 1: Gantt Chart- Outlining the development timeline for our application, with a focus on customer requirements exploration and diagram analysis in the early stages, followed by a week of prototype production.

1.4 Glossary

Term	Description
Client	User who has registered as a client and aims to view the website, purchase items and set up delivery.
Administration	The actors are in charge of supplying food and delivering it. They oversee the client accounts and make management if any illegal operation is found.
Third Party Payment Provider	This actor gives external payment options (such as PayPal) and transfer the money from clients to admins.
Third Party Food Provider	External food supplier that provide necessary ingredients for admins.

2 Requirements

2.1 Overview

Requirements are the cornerstone of product development. Only by determining a primary product requirements can we clear the research and development direction. Product requirements can provide a detailed definition for product planning and development (Sharp, Rogers and Preece, 2019), reducing our time on product design. In an application to allow universities to order food during events such as conferences and workshops, We first collected the stakeholders' requirements and discussed their feasibility in practical application. Then we proposed an iterative upgrade of the requirements based on the meeting content because the requirements were discovered iteratively, and knowledge learned from the lifecycle model could be supplemented during iteration (Sharp, Rogers and Preece, 2019). In addition, we also studied products similar to the application we wanted to do to find out whether requirements needed to be included and finally changed the requirements discussed into MoSCoW requirements according to importance. Finally, we used the domain model to introduce the entities of our application and the relationships between entities.

2.2 Similar Product Analysis

Name of Product	Features	Evaluation
CaterTrax	<ul style="list-style-type: none">- Allows universities to create custom menus and meal plans- Offers nutritional information for meals- Provides order tracking and updates- Offers online ordering and payment- Provides reporting and analytics for meal usage and spending	CaterTrax is a popular catering platform that is specifically designed for universities. The ability to create custom menus and meal plans can be helpful for universities with specific dietary needs, and the nutritional information can help students make informed decisions.
Fresh Ideas	<ul style="list-style-type: none">- Offers customizable meal plans and menus- Provides a dedicated account manager for universities- Offers nutritional information for meals- Provides online ordering and payment- Offers event catering	The ability to create custom meal plans and menus can be helpful for universities with specific dietary needs, and the dedicated account manager can add a personal touch. The event catering option is also a nice feature.

Sodexo	<ul style="list-style-type: none"> - Offers customizable meal plans and menus - Provides nutritional information for meals - Offers online ordering and payment - Provides event catering - Offers a range of other services such as facilities management and cleaning services 	The ability to create custom meal plans and menus is helpful, and the event catering option is a nice touch. However, some universities may find Sodexo to be somewhat inflexible in terms of menu options.
--------	---	---

2.3 MoSCoW Style Requirements

After careful analysis and evaluation with the customer, we finally identified 46 requirements that needed to be fulfilled, and by meeting these requirements, we could get a viable, fully equipped prototype. We use the MoSCoW method to determine requirements based on importance. First, if the application fulfilled all Must-Have and Should-Have requirements, we can see that our application has been completed to meet the stakeholder's demands, which means it is usable. If there is enough time, Could-Have requirements can be implemented to improve our application. However, Won't-Have requirements mean they cannot be implemented in this application. MoSCoW will help us determine which requirements are necessary for the application so that we can arrange our time suitably.

In our MoSCoW requirements, we divide them into functional and non-functional requirements, and each section is divided into different categories. For the functional part, we divide it into general, registration, login, accounts, the home page, checkout, and administration. For the non-functional part, we divide into performance and security.

We have a total of 44 requirements for our application. After careful consideration, we have divided these requirements into the following MoSCoW groups:

- **Must-Haves (36 requirements)** These requirements are necessary to the application and are mentioned directly by stakeholders. If these requirements are met, the application will become usable and complete. These requirements should be implemented in the first place.
- **Should have (6 requirements)** These requirements matter to the application's success rather than the key. These requirements can implement after the Must-Have requirements have been satisfied. Fulfilling these requirements will improve the performance and user experience of the application.
- **Could Have (2 requirements)** These are optional for completing the application. These requirements can contain if we have enough time, it needs to achieve after we satisfy the needs of Must-Have and Should-Have. Fulfilling these requirements will further improve the performance and user experience of the application.

- **Won't-Have (0 requirement)** These requirements do not need to realise in the application.

ID	Functional Requirements	Priority	Actors
General			
FR1	The application must have two types of accounts: clients and administrators.	Must	N/A
FR2	The application must have two types of interfaces: clients and administrators.	Must	N/A
FR3	The application must use the users' email address as their unique User ID.	Must	All
Registration			
FR4	The application must have a registration page for clients to register.	Must	Clients
FR5	The application must allow the client account to create a profile with dietary preferences.	Must	Clients
FR6	The application must have a function to verify the security of passwords	Must	Clients
Login			
FR7	The application must have a login page for clients and administrators to log in.	Must	All
FR8	The application must have an internal login page, which the client cannot see, for administrators to log in.	Must	Admin
FR9	The application should allow users to retrieve password	Should	All
Account			
FR10	The application must have a 'My Account' page for clients and administrators.	Must	All
FR11	The application must allow clients and administrators to modify their account details.	Must	All
FR12	The application must allow clients to modify their profile.	Must	Clients
FR13	The application must allow clients to see the status of their outstanding orders.	Must	Clients

FR14	The application must allow clients to export an order invoice in PDF format, including the food ordered, the price of each food, the total cost, and the delivery date.	Must	Clients
FR15	The application must allow clients to see their previous completed orders.	Must	Clients
FR16	The application must allow clients to cancel order.	Must	Clients
Main Page			
FR17	The application must have a main page allowing the users to view food by searching or applying the filter.	Must	Clients
FR18	The application must allow users to see the ingredients in each food.	Must	Clients
FR19	The application must allow users to see the price of each food.	Must	Clients
FR20	The application must allow users to select the amount of food to order.	Must	Clients
FR21	The application must allow users to add food into baskets, see the total price, and cancel the food.	Must	Clients
FR22	The application must allow administrators to check the total amount of available food.	Must	Admin
FR23	The application should allow administrators to manage the discounted food and edit promotions (displayed at the beginning of the main page).	Should	Admin
FR24	The application could have a function where food with missing ingredients will not appear on the main page if the order date is less than a week.	Could	Clients
Checkout			
FR25	The application must have a checkout page allowing clients to edit their order and checkout.	Must	Clients
FR26	The application must allow clients to use the third-party payment to pay.	Must	Clients
FR27	The application must allow clients to pick a delivery date.	Must	Clients

FR28	The application must allow clients to select a delivery address.	Must	Clients
FR29	The application must be able to send an email confirmation to the user after checkout, including price, ordered food, delivery date, and address.	Must	Clients
FR30	The application should allow clients to add a note to their order.	Should	Clients
Administration			
FR31	The application must allow administrators to manage all customer accounts.	Must	Admin
FR32	The application must allow administrators to see the details of each order.	Must	Admin
FR33	The application must have an inventory management system allowing administrators to check the total amount of available ingredients.	Must	Admin
FR34	The application should enable administrators to order food from the third-party provider.	Should	Admin

ID	Non-Functional Requirements	Priority	Actors
Performance			
NFR1	The application must respond to users within 5 seconds when users have some actions like checking out, refreshing pages and logging in to ensure the users have a pleasant ordering experience.	Must	N/A
NFR2	The application must be available at least 99% of the year to ensure users can use this application most of the time.	Must	N/A
NFR3	The application must be compatible with all browsers, operating systems and devices to ensure all users with different devices and different preference browsers can order.	Must	N/A
NFR4	The application must deal with 10,000 people visiting this application and orders simultaneously, without influencing response time.	Must	N/A
NFR5	The application should provide system reports on all orders and collect customer feedback to give suggestions on improving the application.	Should	N/A
NFR6	The application could provide multiple language selections on the interfaces to broaden users.	Could	N/A
Security			
NFR7	The application must have backups and recovery procedures in case of data loss.	Must	N/A
NFR8	The application must ensure user data security and support for security during payments.	Must	N/A
NFR9	The application must monitor all activities that happen within the application to find out security issues.	Must	N/A
NFR10	The application should use encryption technologies to protect sensitive data, including credit card information during payment and login credentials during login.	Should	N/A

2.4 Domain Model

The domain model is a diagram displaying all the static entities that the project consists of and the relationships between them. It serves as an initial blueprint for designing and implementing the software system and helps both the development team and the client to understand the underlying structure of the system(Boubekeur and Mussbacher, 2020). The process of creating the diagrams begins with extracting the entities (classes and user types) from the Functional Requirements list, then their relationships are described using arrows of different styles.

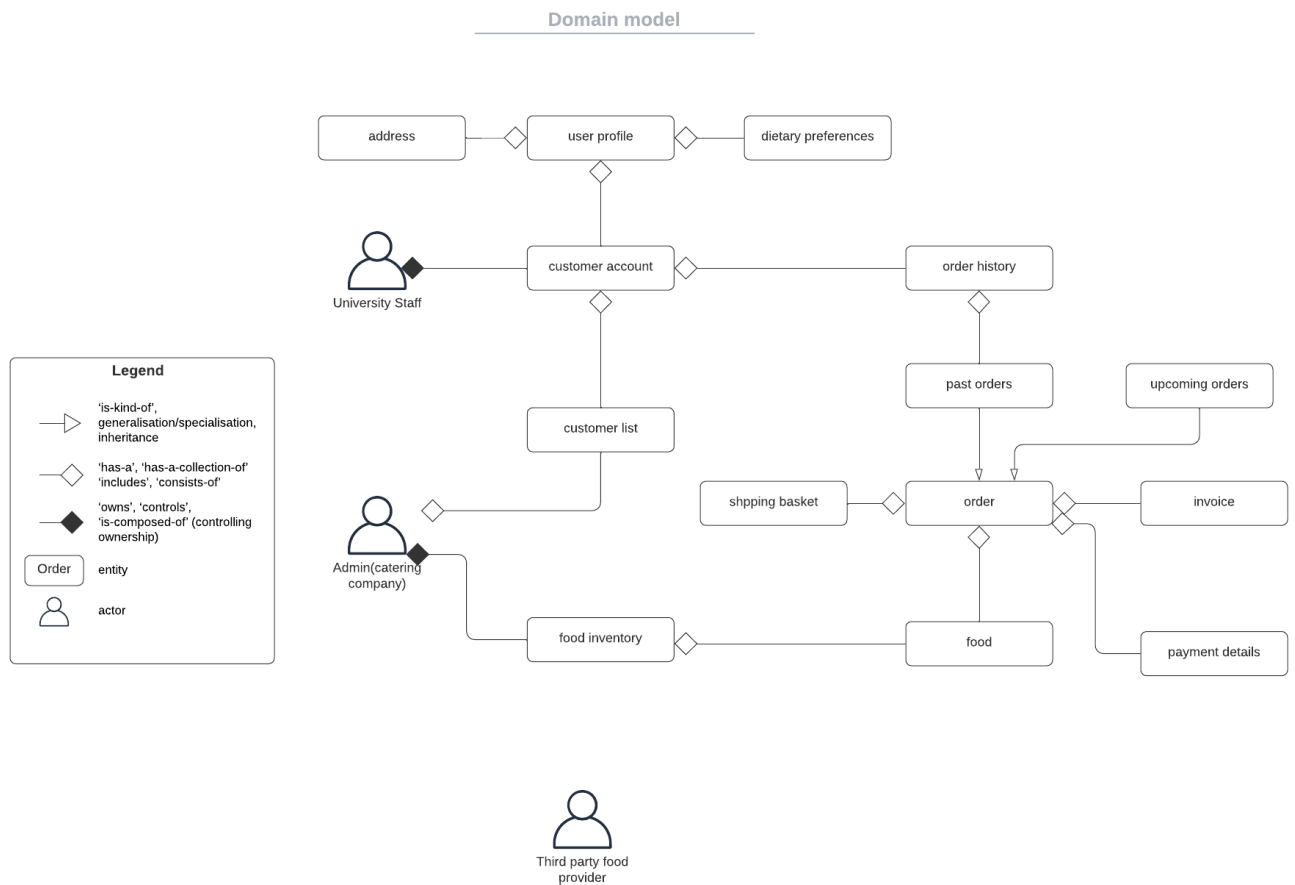


Figure 2: Domain model- displays the domain model for the delivery app, showcasing the static entities and their relationships. The diagram serves as a blueprint for designing and implementing the software system, aiding both the development team and the client in understanding the system's structure.

3 Use Cases

3.1 Overview

Use cases are derived from MoSCoW requirements in Chapter 2. Although MoSCoW requirements have described what we need to achieve in detail, it is still very abstract for users to realize these requirements, which makes the use cases very necessary. It helps users understand how to interact with the application. We have included in this chapter a use case list, use case diagrams, and use case specifications based on MoSCoW requirements. Using use cases can be crucial in extracting valuable insights from the software development lifecycle. This approach facilitates extracting pertinent information from crucial phases such as requirements gathering, modelling, testing, and maintenance. By utilizing use cases, developers can gain a deeper understanding of the intricacies involved in these critical stages, thereby improving the quality of the developed software (Tiwari and Gupta, 2015). Use case diagrams provide the interactions of various user roles with the application and the relationships between those interactions. Use case specifications describe the use cases summarized on each MoSCoW requirement, including its designed roles, conditions, and specific steps when the user executes it. Through the description of the application scenario, the users can feel the experience of using it. Use cases are necessary for the application development process, which can help to identify potential problems at the early stage of development and fix them in time. Moreover, through use cases, stakeholders can more intuitively feel how their requirements will be realized, which can help them put forward opinions more effectively.

3.2 Use Case List

The user case list is derived from the discussion of MoSCoW and the potential user for this website. The Glossary in Chapter 1 has defined the users and more detailed user list descriptions would be given in the following part. Unregistered users are able to browse the website but cannot place orders. The actors are listed as follows:

- Client
- Administration

ID	Use Case Name	Actor
UC1	CustomerRegistration	Clients
UC2	CustomerLogin	Clients
UC3	AdminLogin	Admin
UC4	RetrievePassword	Clients
UC5	ModifyCustomerAccount	Clients
UC6	ModifyAdminAccount	Admin
UC7	SeeOrderStatus	Clients
UC8	SeePreviousOrders	Clients
UC9	ViewFoodCatalogues	Clients
UC10	ViewFoodDetails	Clients
UC11	AddFoodIntoBasket	Clients
UC12	CancelFoodFromBasket	Clients
UC13	EnactmentDelivery	Clients
UC14	EditPromotion	Admin
UC15	Checkout	Clients
UC16	CancelOrder	Clients
UC17	OrderHelp	Clients
UC18	SendEmailConfirmation	Clients
UC19	ManageCustomerAccount	Admin
UC20	SeeOrderDetails	Admin
UC21	CheckIngredientAmount	Admin
UC22	OrderFoodFromOtherProviders	Admin

3.3 Use Case Diagram

The use case diagram Fig.3 describes how users interact with the application we design. The actors (unregistered guests, clients, and the admins) have different permissions and the various use cases are performed.

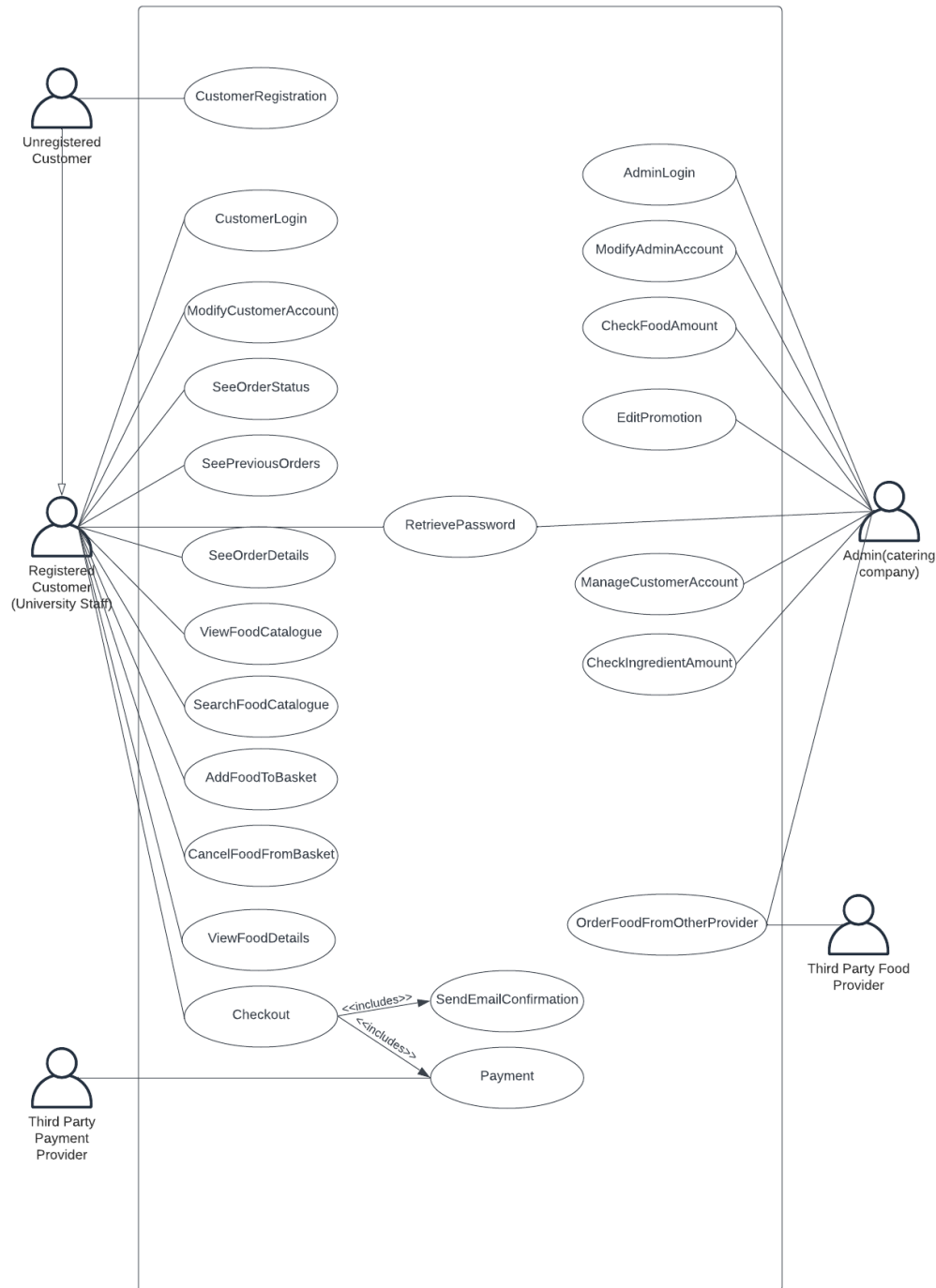


Figure 3: The user case diagram- Visualizing user interaction for our application's actors and permissions.

3.4 Use Case Specifications

Based on the user cast list, a series of user case specifications are formulated . This helps to better understand how users interact with the system, the scenarios they encountered and how the system should response.

Name:	CustomerRegistration
ID:	UC1
Brief Description	The customers can register
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	None
Main Flow:	<ol style="list-style-type: none">1. The use case starts when the user selects “Register”2. The system asks the user to set a username3. If the system finds a matching username then<ol style="list-style-type: none">3.1. User sees the error and is asked to set another username4. Else<ol style="list-style-type: none">4.1. The system asks the user to specify email address, phone number and dietary preferences4.2. The system asks the user to set a password4.3. If the client clicks the visibility icon then<ol style="list-style-type: none">4.3.1. Password is displayed4.4. Else<ol style="list-style-type: none">4.4.1. Password is masked4.5. If the password is weak then<ol style="list-style-type: none">4.5.1. User sees the error and instructions to set a stronger password4.6. Else<ol style="list-style-type: none">4.6.1. The system logs in the user
Postconditions:	The client has an account
Alternative Flows:	None

Name:	CustomerLogin
ID:	UC2
Brief Description	Login page for clients to login
Primary Actor(s):	Clients
Secondary Actor(s):	System
Preconditions:	Clients already registered and have an account
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the client clicks "login" 2. The system asks the client for username and password 3. If the client clicks the visibility icon then <ol style="list-style-type: none"> 3.1. Password is displayed 4. Else <ol style="list-style-type: none"> 4.1. Password is masked 5. If the username or password don't match with an existing account, then <ol style="list-style-type: none"> 5.1. The user sees the corresponding error ("Username/password is incorrect") 6. Else <ol style="list-style-type: none"> 6.1. The client gets logged into the his account 6.2. If "Remember me" is ticked then <ol style="list-style-type: none"> 6.2.1. The system memorizes the credentials
Postconditions:	The client is logged into his account
Alternative Flows:	None

Name:	AdminLogin
ID:	UC3
Brief Description	Internal login page for admins which clients cannot see
Primary Actor(s):	Admins
Secondary Actor(s):	System
Preconditions:	Admins are registered
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the admin clicks "login" 2. The system asks the admin for username and password 3. If the admin clicks the visibility icon then <ol style="list-style-type: none"> 3.1. Password is displayed 4. Else <ol style="list-style-type: none"> 4.1. Password is masked 5. If the username or password don't match with an existing account, then <ol style="list-style-type: none"> 5.1. The admin sees the corresponding error ("Username/password is incorrect") 6. Else <ol style="list-style-type: none"> 6.1. The admin gets logged into the his account 6.2. If "Remember me" is ticked then <ol style="list-style-type: none"> 6.2.1. The system memorizes the credentials
Postconditions:	Admin is logged into his account
Alternative Flows:	None

Name:	RetrievePassword
ID:	UC4
Brief Description	A system that allows users to retrieve password
Primary Actor(s):	Clients, admins

Secondary Actor(s):	None
Preconditions:	The user does not remember his password
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user clicks "retrieve password" 2. The user is asked for his username, phone number and email 3. The user receives an email that specifies the password
Postconditions:	The user is able to login to his account
Alternative Flows:	None

Name:	ModifyCustomerAccount
ID:	UC5
Brief Description	The client can edit their profile
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	The client must be logged into the system
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the client clicks "Modify Account" 2. The system displays a page showing the account information 3. The client selects which field they want to edit 4. When the client finishes, he clicks save and is returned to the home page
Postconditions:	The client's information is updated by the system
Alternative Flows:	None

Name:	ModifyAdminAccount
ID:	UC6
Brief Description	The admin can edit their profile
Primary Actor(s):	Admins
Secondary Actor(s):	None
Preconditions:	The admin must be logged into the system
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the admin clicks "Modify Account" 2. The system displays a page showing the account information 3. The admin selects which field they want to edit 4. When the admin finishes, he clicks save and is returned to the home page
Postconditions:	The admin's information is updated by the system
Alternative Flows:	None

Name:	SeeOrderStatus
ID:	UC7
Brief Description	Clients can see the current status of their order
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	<ol style="list-style-type: none"> 1. User must be logged in as a client 2. User must have a placed order currently pending delivery

Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when user clicks on an order which is pending delivery 2. The system navigates to a new page that displays the current status of the order (e.g. confirmed, in preparation, in delivery etc.)
Postconditions:	None
Alternative Flows:	None

Name:	SeePreviousOrders
ID:	UC8
Brief Description	Clients can see a list of their previous orders
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	<ol style="list-style-type: none"> 1. User must be logged in as a client
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user clicks on the “previous orders” option in the menu 2. The system checks for the users previous orders 3. If the system finds at least one previous order then: <ol style="list-style-type: none"> 3.1. The system displays a page containing the most recent orders placed by the user 3.2. For each order the system displays the name, date placed, date delivered, and price of the order 3.3. The user may choose to click on an order to view additional details 4. Else: <ol style="list-style-type: none"> 4.1 The system displays a message stating that there are no previous orders
Postconditions:	None
Alternative Flows:	None

Name:	ViewFoodCatalogues
ID:	UC9
Brief Description	Clients can search through the food catalogue using keywords and filters
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	1. User must be logged in as a client
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user navigates to the home page of the application 2. The user can scroll through the page to see a full listing of the currently available foods to order 3. The user can click on the search bar and type in a keyword to search for a specific type of food 4. The system will display all the food in the catalogue that matches the keyword the user entered 5. The user can also click on the “filter” icon to apply filters to the food catalogue such as (discounted, vegetarian, vegan, gluten-free, Halal) 6. The user can also click on the “filter” icon to sort food by price, name and popularity
Postconditions:	None
Alternative Flows:	None

Name:	ViewFoodDetails
ID:	UC10
Brief Description	Clients can see the full details of a particular type of food
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	1. User must be logged in as a client
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user clicks on a particular food in the catalogue 2. The system displays the name, pictures, basic ingredients, a brief description, and the price per portion of the food 3. The system displays the options to add the food to basket
Postconditions:	None
Alternative Flows:	None

Name:	AddFoodIntoBasket
ID:	UC11
Brief Description	The clients select food and add into baskets
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	Log in as a Client account
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user browse home page and choose preferred food. 2. The system shows a notification of the maximum stock, and asks the user to select food quantity.

	3. The food is added into basket and the systems shows the updated lump sum at the bottom of the page.
Postconditions:	None
Alternative Flows:	None

Name:	CancelFoodFromBasket
ID:	UC12
Brief Description	The clients cancel selected food in their baskets
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	Viewed and have selected food into basket
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user clicks "delete" near an item in basket. 2. The system shows a message to user to ask for their action confirmation. 3. When the user agree to approve action, the system shows the updated food basket.
Postconditions:	None
Alternative Flows:	None

Name:	EnactmentDelivery
ID:	UC13
Brief Description	The client chooses the specific time and date for delivery.
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	The client is directed to the checkout page

Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the client finished food browse and click "Delivery Details". 2. The system shows a calendar (with available dates in green and unavailable in red). 3. When the user select the preferred date, the system displays a valid timetable. 4. The user confirms the delivery date and time.
Postconditions:	The user proceed to payment.
Alternative Flows:	None

Name:	EditPromotion
ID:	UC14
Brief Description	The admins manage the discounted food and edit promotions.
Primary Actor(s):	Admins
Secondary Actor(s):	None
Preconditions:	Log in as an Admin account
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the admin navigates to the 'Food Details' page and choose to "Manage Promotion". 2. The system ask the admin to select particular food they want to add promotion to. 3. The admin changes or sets new discounted rate. 4. When the admin confirms the operation, he is returned to the home page,
Postconditions:	The system refreshes and displays the promotion on the beginning of search page.
Alternative Flows:	None

Name:	Checkout
ID:	UC15
Brief Description	The users pay for the order.
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	The client completed food and delivery date selection.
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the client confirmed delivery details and click "Payment". 2. The system shows kinds of payment methods. 3. If the user chooses to pay by card (Visa/Master-Card/American Express) then <ol style="list-style-type: none"> 3.1. The system asks the user to input information for card, including the owner name, card number, expiry date, and CVV. 4. Else if user select third party payment (PayPal/Apple pay) <ol style="list-style-type: none"> 4.1. The system is directed to third party page and requires user to complete payment.
Postconditions:	The order is completed.
Alternative Flows:	None

Name:	CancelOrder
ID:	UC16
Brief Description	The clients cancel ongoing orders.
Primary Actor(s):	Clients
Secondary Actor(s):	None

Preconditions:	The client has ongoing orders which is not already delivered.
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user navigates to the order page and select 'Cancel this order'. 2. The system requires the confirmation of this action and send the request to the related admin. 3. If the admin agrees <ol style="list-style-type: none"> 3.1. The system shows 'Already canceled' and processes refund to the original payment account. 4. Else <ol style="list-style-type: none"> 4.1. The system refuses the cancellation and displays the reason.
Postconditions:	The system refreshes and shows whether this order has been canceled.
Alternative Flows:	None

Name:	OrderHelp
ID:	UC17
Brief Description	Clients report errors in their orders.
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	The client has completed orders.
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user selects 'Order help' 2. The system displays various mistake types, including 'Order didn't arrive', 'Report missing', and 'Incorrect items'. 3. The user selects specific one and requires refund. 4. The system sends information to admins to process.
Postconditions:	The error in orders is reported.
Alternative Flows:	None

Name:	SendEmailConfirmation
ID:	UC18
Brief Description	Send a confirmation email to client
Primary Actor(s):	Clients
Secondary Actor(s):	None
Preconditions:	Checkout
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the user checkout 2. The system sends an email to the registered email address of the user, including the total price, ordered food, delivery date, and address.
Postconditions:	None
Alternative Flows:	None

Name:	ManageCustomerAccount
ID:	UC19
Brief Description	Admins manage the customers' account
Primary Actor(s):	Admins
Secondary Actor(s):	Clients
Preconditions:	Log in as an admin account
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the admin selects "Manage customers' account" 2. The system asks the admin to choose a specific customer's account 3. The system asks the admin to choose to deactivate, reactivate or delete the account
Postconditions:	None
Alternative Flows:	Cancel changes

Name:	SeeOrderDetails
ID:	UC20
Brief Description	Admins see the details of the customer's order
Primary Actor(s):	Admins
Secondary Actor(s):	Clients
Preconditions:	<ol style="list-style-type: none"> 1. Log in as an admin account 2. Some customers checked out
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when a customer checked out 2. The system sends details of the order to the admins, including the ordered food, delivery date, address, and the note written by the customer
Postconditions:	None
Alternative Flows:	None

Name:	CheckIngredientAmount
ID:	UC21
Brief Description	Admins check the amount of available ingredient
Primary Actor(s):	Admins
Secondary Actor(s):	None
Preconditions:	Log in as an admin account
Main Flow:	<ol style="list-style-type: none"> 1. The use case starts when the admins select to check the amount of available ingredient 2. If the admin is searching for a specific ingredient or applying a filter then <ol style="list-style-type: none"> 2.1. The inventory system shows the information for the specific ingredient or the relevant ingredient, including name, amount, and the date of last purchase

	<p>3. Else</p> <p>3.1. The inventory system shows all information for the available ingredient, including name, amount, and the date of last purchase</p> <p>4. If the amount of ingredients is less than 50, the line where it is located will be marked red as a warning</p>
Postconditions:	None
Alternative Flows:	None

Name:	OrderFoodFromOtherProviders
ID:	UC22
Brief Description	Admins purchase required ingredient
Primary Actor(s):	Admins
Secondary Actor(s):	None
Preconditions:	Log in as an admin account
Main Flow:	<p>1. The use case starts when the admin selects to purchase the required ingredient</p> <p>2. The system will show all ingredients and sort them in descending order according to the total amount</p> <p>3. The system asks the admin to select the type and amount of ingredients to buy</p> <p>4. The system sends the order to the third-party suppliers</p>
Postconditions:	None
Alternative Flows:	Cancel order

4 Object-Oriented Analysis and Design models

4.1 Overview

In this chapter, we will complete object-oriented analysis and design models, UML class diagrams, sequence diagrams, and other relevant diagrams and documentation. Unified Modeling Language (UML) diagrams are a powerful tool for expressing system requirements clearly and concisely. These diagrams contain various graphical symbols that enable developers to represent complex software systems accurately. By utilizing UML diagrams, developers can effectively convey various system components' structure, behaviour, and relationships, thereby enabling a better understanding of the software being developed (Ali, Shukur and Idris, 2007). Diagram the first one is the analysis class diagram, which is the initial version of the diagram and is easy to read and improve upon, building on the requirements and use cases from the previous chapters to complete the analysis class diagram; for the design class diagram, which is an improved version of the analysis diagram with more additional classes added to it. A set of sequence diagrams follows this to show use case scenarios and ensure that the design class diagram is usable; a state machine diagram, which considers the different states of the classes; then an activity diagram(s) to show business cases, which requires several use cases to demonstrate the application and variations of the actual scenarios; then a component diagram, which shows the components of the program and the excuses and separation features between them. Finally, there is the deployment diagram, which shows how the application should be deployed and how it should be connected. The following is a more precise table.

- Analysis class diagram
- Design class diagram
- Component Diagram
- Deployment Diagram
- Activity Diagram
- Sequence Diagrams
- State Machine Diagram

4.2 Object-Oriented Analysis

4.2.1 Class diagram

The class diagram is one of the building blocks of object-oriented modeling and is the main component of object-oriented modeling. The static structure of the model can show the class

information in the model, the internal structure of the class, and the association between classes. Class diagrams do not show transitory information. And class diagrams are the most commonly used UML diagrams to show interfaces and their static structure and relationships. They are used in the analysis and design phases of object-oriented software development. A well-structured class diagram has the following characteristics: some of the rules we will follow for class diagrams.

R1	Focus on sufficient instead of complete details
R2	Only include elements that are there for the people to understand
R2	Have enough information for the people to understand
R2	No error messages

Then is our class diagram:

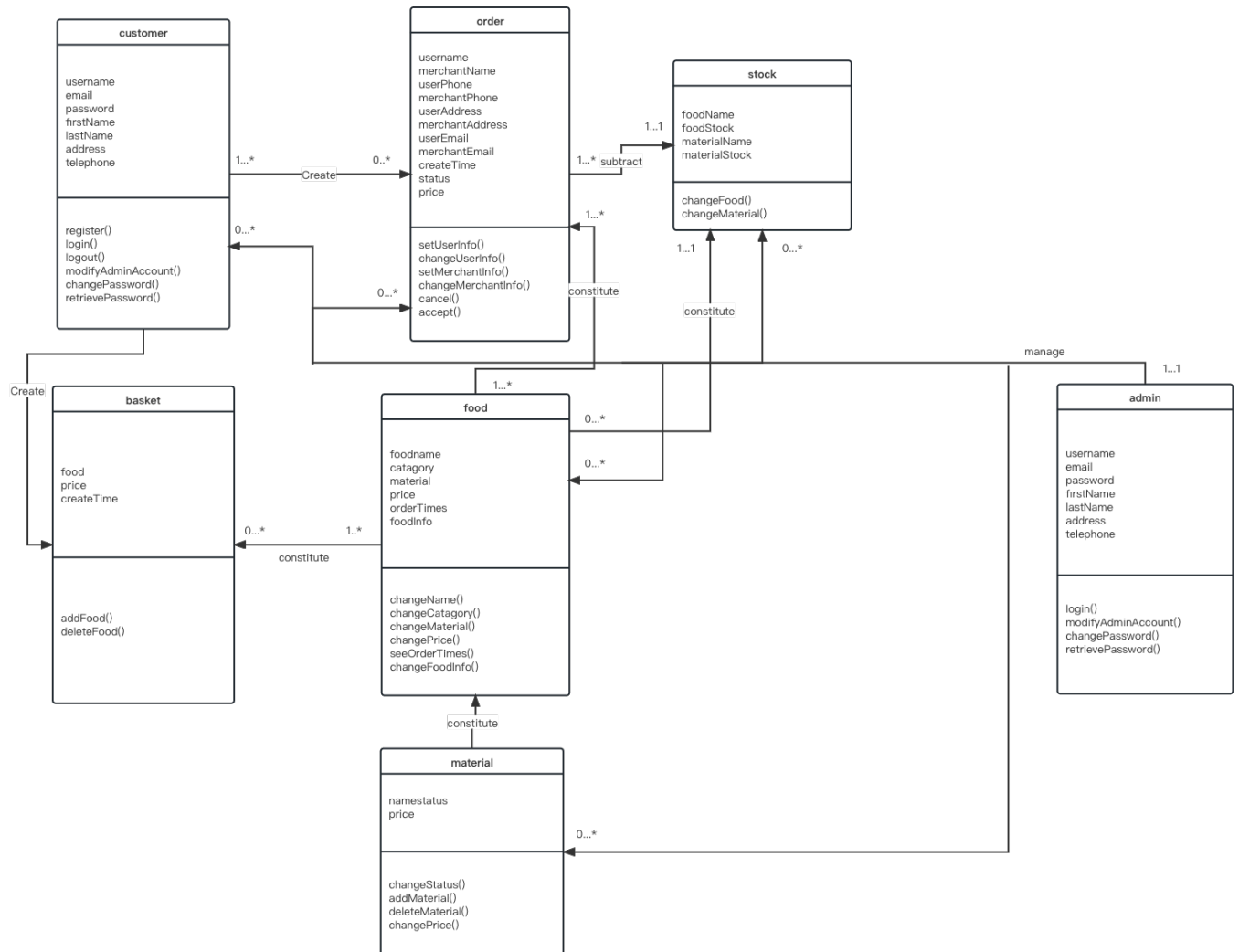


Figure 4: Class Diagram- For Static Structure and Relationships in Software Development, which has entities customer, order, stock, basket, food, admin, and material.

4.3 Object-Oriented Design

4.3.1 Class Diagram

Although both Class Diagram Design and Class Diagram Analyze are UML class diagrams, they have different emphases. The latter focuses more on requirement analysis, behavioral interaction, etc., and Class Diagram Design here demonstrates the conversion of requirements into code in detail (Mythily, 2018). transition framework.

- In addition to the basic needs of customers mentioned in Class Diagram Analysis, search and filter functions have been added to the product and shopping cart modules, so that schools can accurately search and compare when there are huge orders and many products to find the most Appropriate order items.
- In the food and material modules, add and delete functions have been added. Due to the large number of commodities and raw materials and various types, if only the administrator manages the items in this module, there may be problems such as omissions and incompleteness, so add them here In addition to adding the delete function, after the merchant submits the request, it needs to be reviewed by the administrator, and the modification can only be made after the review is approved.
- The inventory module also adds a food statistics function to facilitate merchants to count the remaining raw materials, and also enables the platform to obtain the quantity of food that can be sold.

Each attribute and function in Class Diagram Design is considered more carefully along with the relationship between each class, and the structure has also been adjusted to a certain extent. The specific and complete design diagram is shown in the following figure:

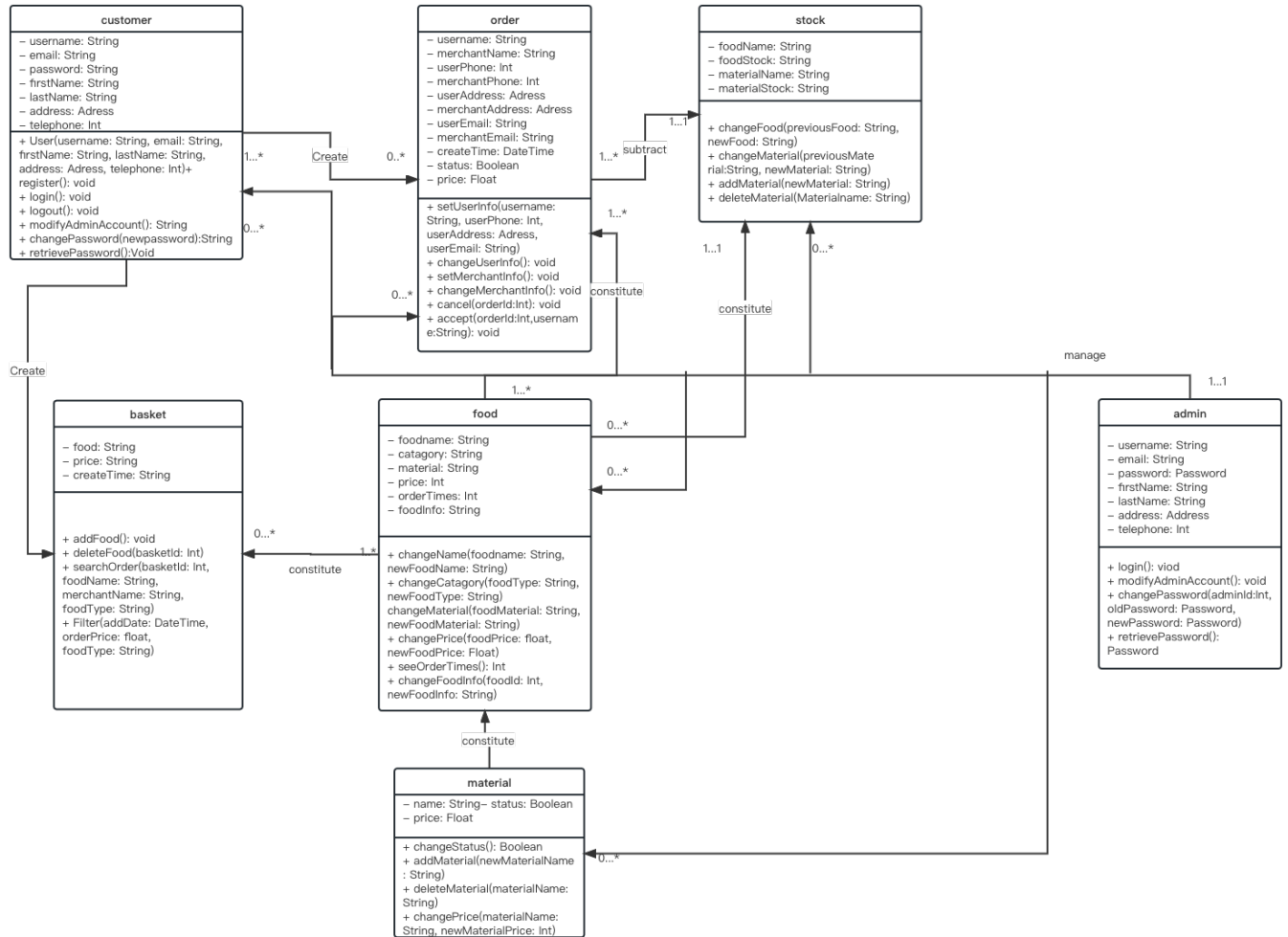


Figure 5: Class Diagram Design- a detailed one for software development, focusing on requirement analysis, behavioral interaction, etc., adding search and filter functions to the product and shopping cart modules.

4.3.2 Component Diagram

The component diagram in UML is mainly to show the components of each system more clearly, their interfaces and their dependencies and collaborations. A component means a module in the system, which can be reused, encapsulated and replaced. In the subsequent development process, the framework structure can be made clearer, and the subsequent system maintenance will be more convenient.

Components previously interacted through the interface (provided by the component itself, visualized by a ball) and the interface required by the component (visualized by a semicircle). The following image is the component diagram of this project:

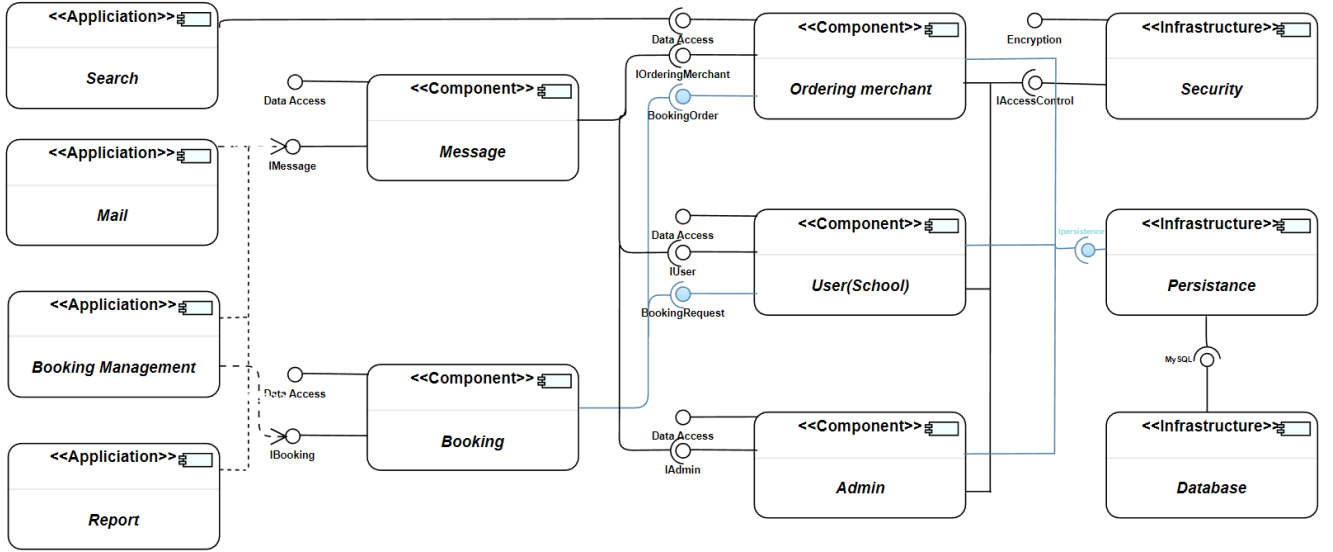


Figure 6: Component Diagram- Visualizing system modules and dependencies in UML for improved framework structure and maintenance.

4.4 Deployment Diagram

A deployment diagram is a diagram used to show the relationships between system or application components. It is often used to describe a software system's physical and logical deployment, such as the relationship between a server and an application, making the diagram of the components clearer. The relationship diagram between components displayed in the deployment diagram enables developers to understand the operating mode of the system better. It also displays the deployment information of the system so that administrators can better understand the deployment status of the system, and operation and maintenance personnel can better understand the system architecture and performance to manage and maintain the system (Mohammadi and Barforoush, 2014).

This deployment diagram shows that the user interfaces are connected to application servers through the web servers. The application server realizes most functions required by application operation, such as email notification and search order. These functions need to extract data from the database for matching. We connect the database and the application server through TCP. Here are the deployment diagram's details.

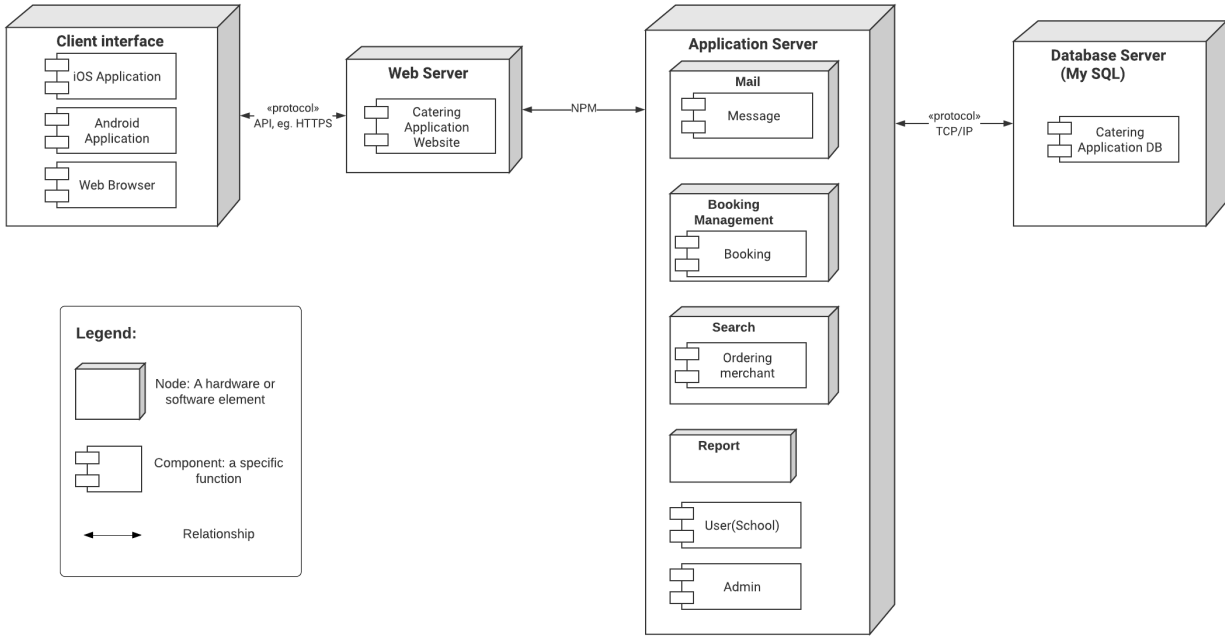


Figure 7: Deployment Diagram- visualizing system components and connections, which contain client interface, by using web server to connect with application server, and TCP connect database server with application server.

4.5 Activity Diagrams

An activity diagram is a UML diagram, a visual representation of the steps and operations involved in a series of product development processes, such as software engineering, indicating the association of specific activities (Eshuis and Wieringa, 2004). In our catering application, we created two activity diagrams: one to show the user's activity when registering and accessing the application and the other to show the process of the user ordering food. The user registration and access diagram will show the steps and actions involved in creating a new account, logging in, and accessing application features. The food ordering diagram will show the steps and actions involved in selecting items, customizing your order, and completing the checkout process.

Both diagrams are essential for understanding how the application works and how users interact. By visualizing these processes, we can simulate the user experience for improvement, and activity diagrams can easily present these activity processes to relevant personnel such as developers, designers, or analysts for analysis and discussion.

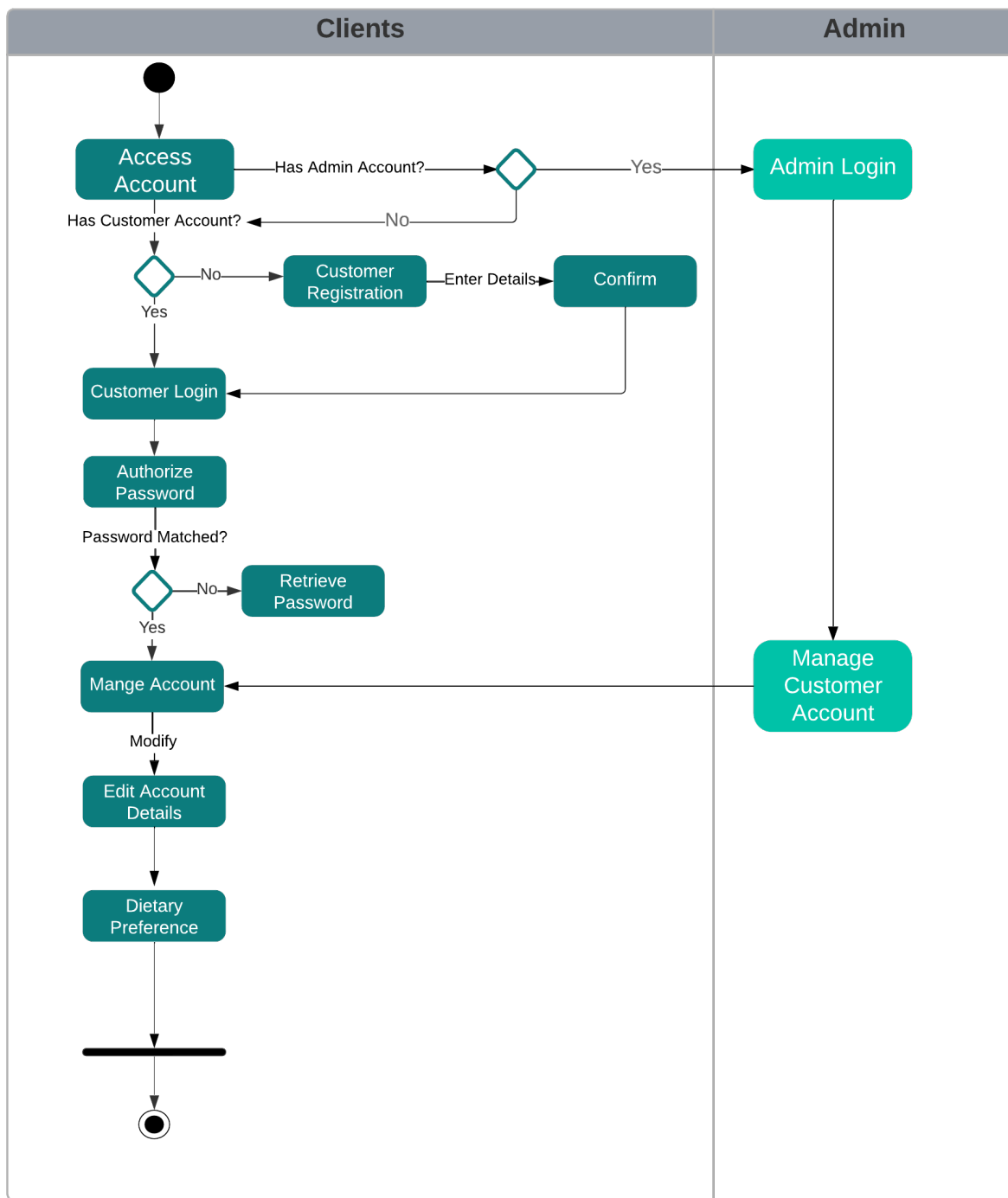


Figure 8: The user registration and access diagram- Show the steps and actions involved in creating a new account, logging in, and accessing application features.

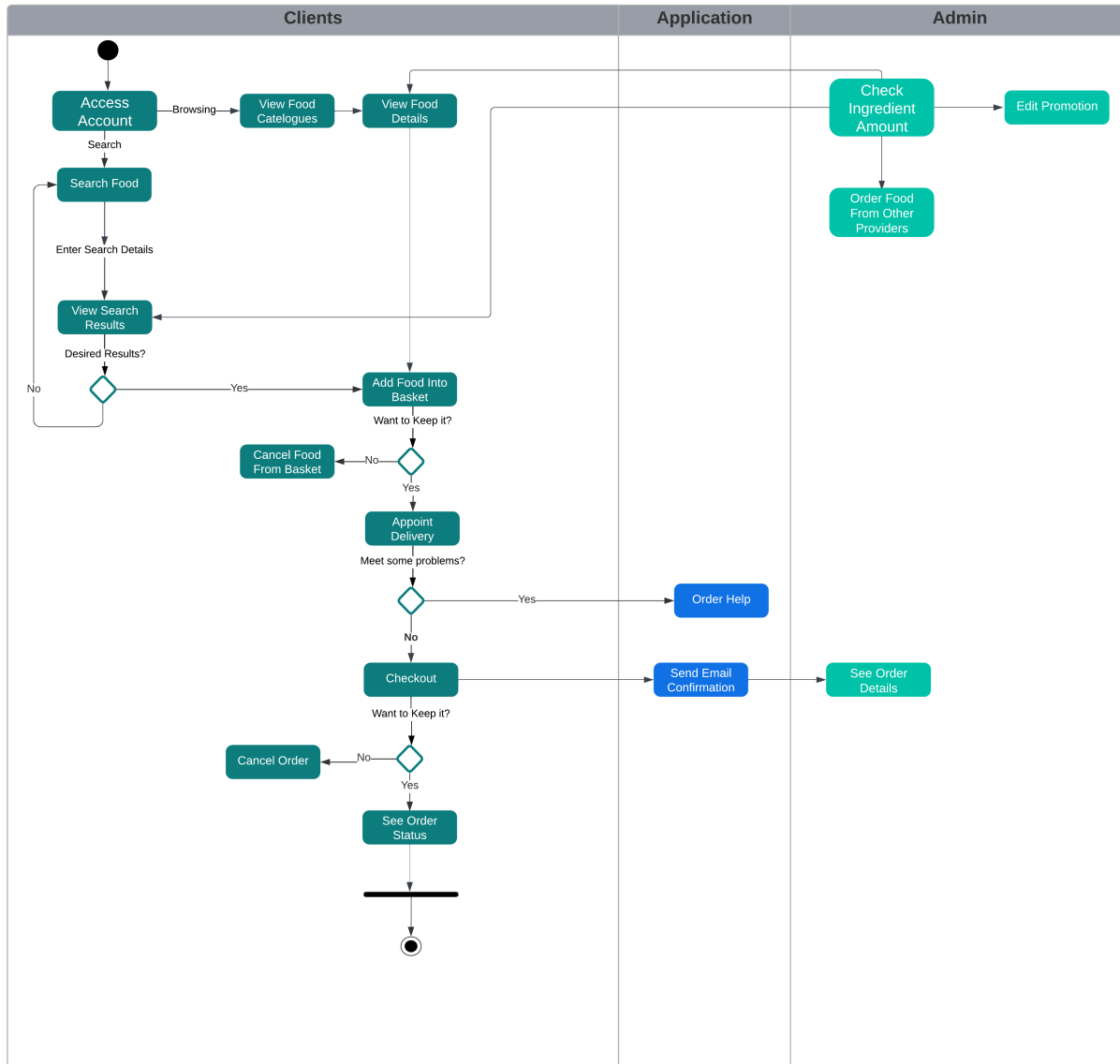


Figure 9: The food ordering diagram- Show the steps and actions involved in selecting items, customizing your order, and completing the checkout process.

4.6 Sequence Diagrams

The sequence diagram visualises the sequence of calls for our system, based on the use cases we have previously written, the use cases we have chosen are:

Use Cases(ID)	Figure	Title
CustomerRegistration(UC1)	Figure 10	Registration
CustomerLogin(UC3), ModifyCustomerAccount(UC6)	Figure 12	Login and modify
SeeOrderStatus(UC7), SeePreviousOrders(UC8), AddFoodIntoBasket(UC11), CancelFoodFromBasket(UC12), Check-out(UC15), CancelOrder(UC16), SendEmailConfirmation(UC18)	Figure 13	Order food
Edit Promotion(UC14), See Order Details(UC20),	Figure 14	Menu Display
Check Ingredient Amount(UC21), Order Food From Other Providers(UC22),	Figure 15	Manage Storage

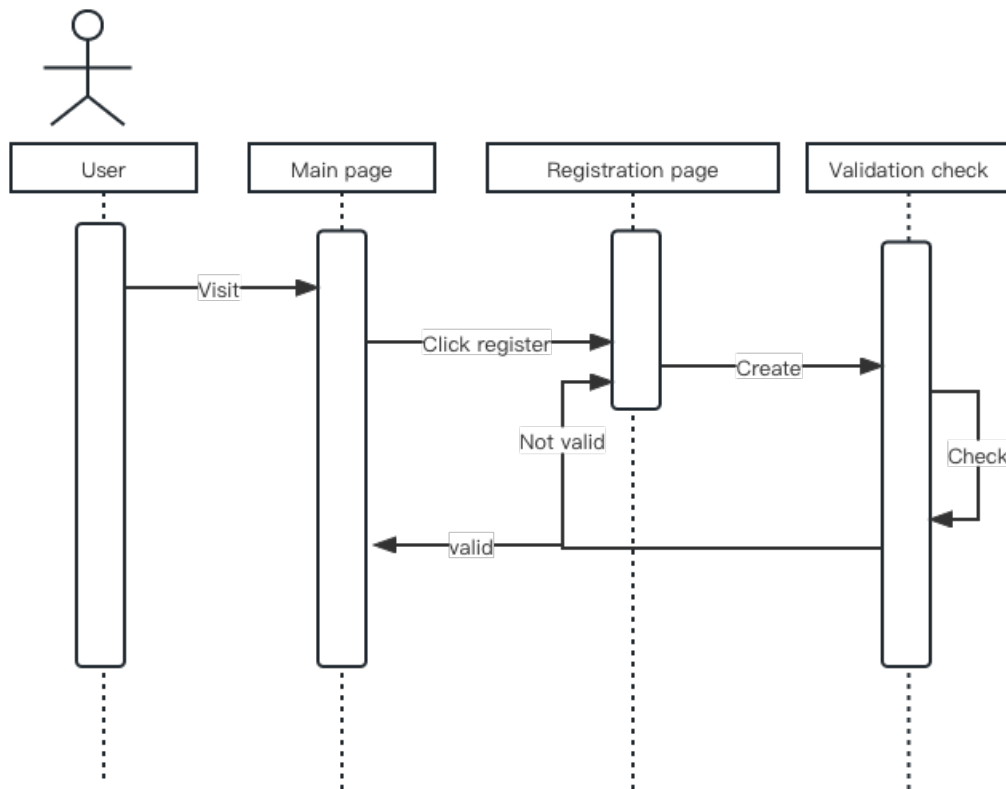


Figure 10: Sequence Diagrams of Registration- By visiting main page, user can fill in registration information, user can only have an account by pass the validation check.

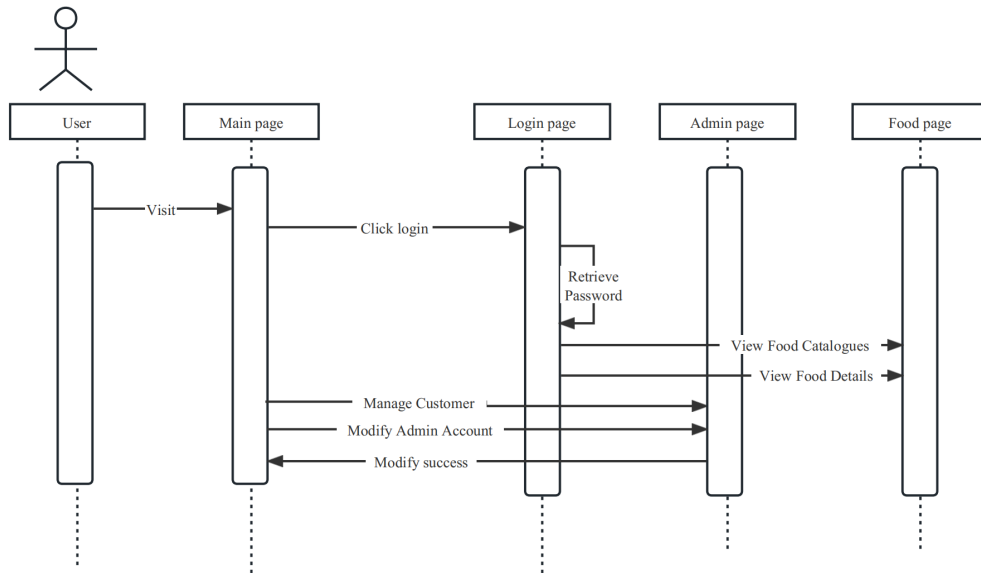


Figure 11: Sequence Diagrams of Admin management- By visiting main page, user can login to enter the login page, if they forget their password, retrieve password is needed. If login as admin, user go to admin page, then they can manage customer, modify admin account, and modify success, if login as customer, they will be directed to food page.

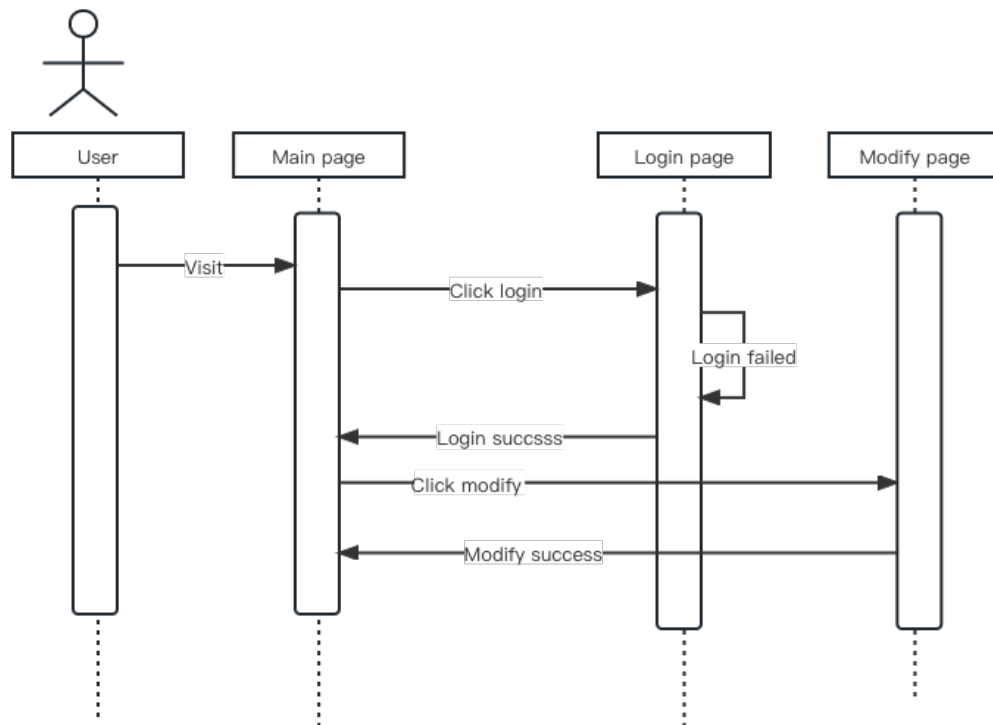


Figure 12: Sequence Diagrams of Login and modify- By visiting main page, and login successfully, user can modify information through modify page.

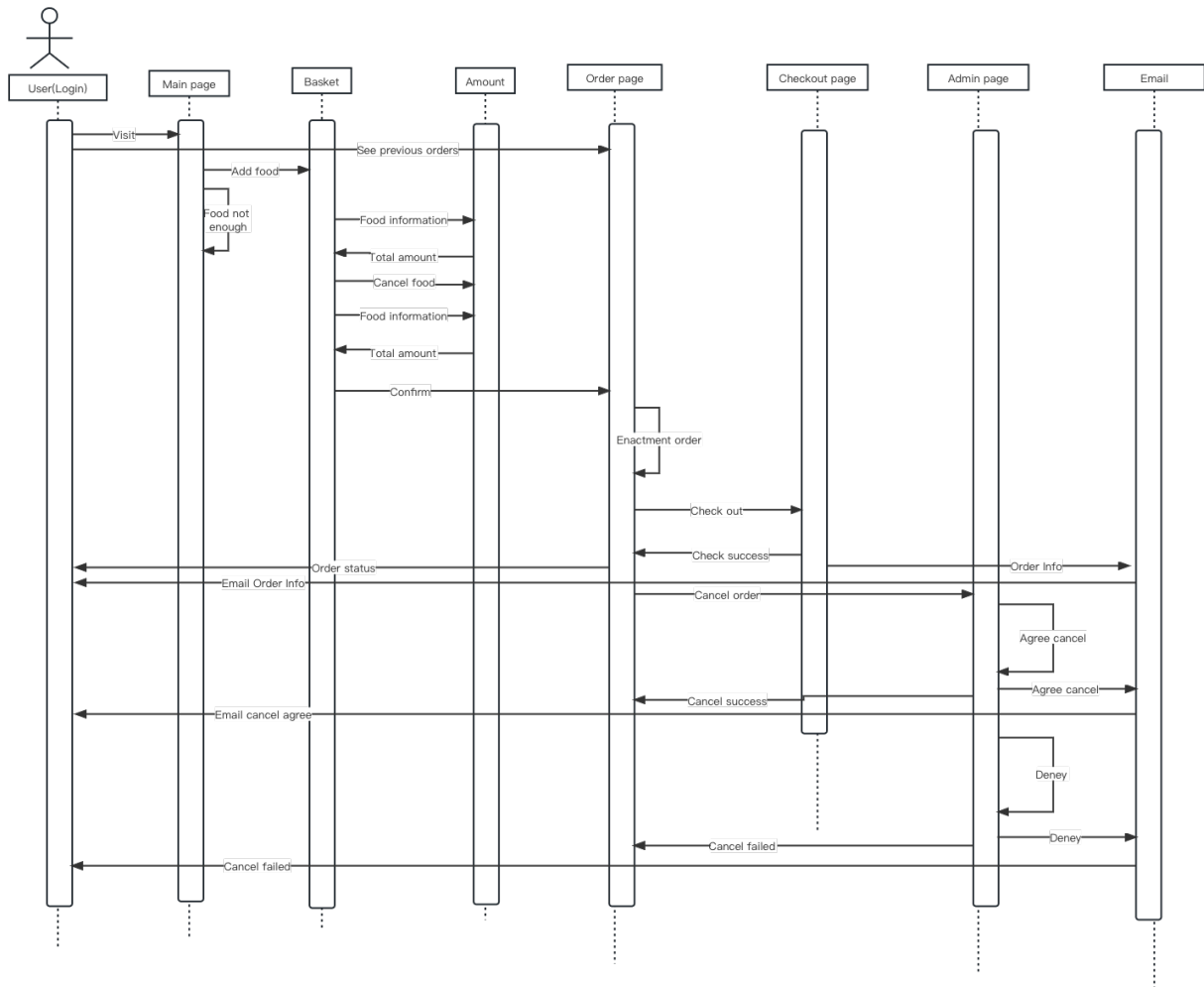


Figure 13: Sequence Diagrams of Ordering food- By visiting main page, and login successfully, customer can add food to basket and confirm amount of food, if everything is fine, they can click to order page to appoint delivery time, and go to checkout page to finish the purchase process, after these things done, a confirmation email will send to customer's email address. If customer want to cancel this order, they need to wait for the agreement of admin.

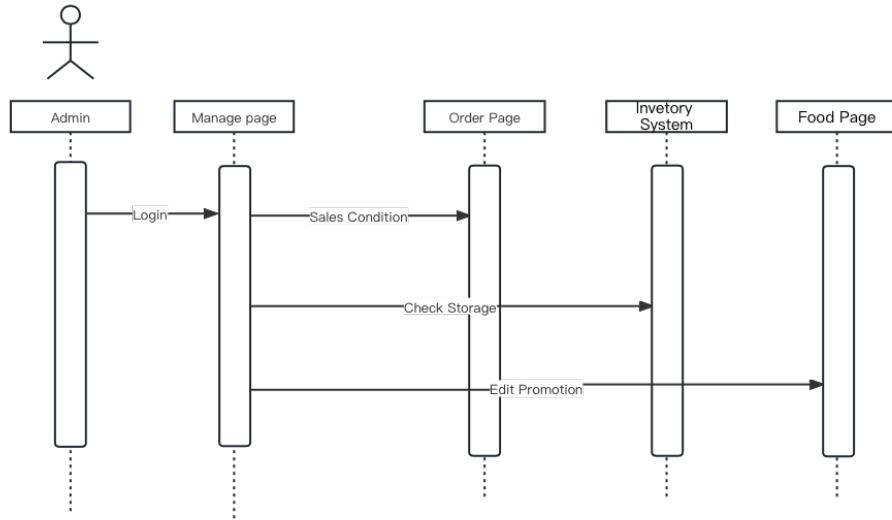


Figure 14: Sequence Diagrams of Menu Display- Admin can decide promotion intensity and select promotional products by viewing customer's order to find products sales condition, and check the inventory of ingredients to decide which menu to display on the food page, and select which menu to promote.

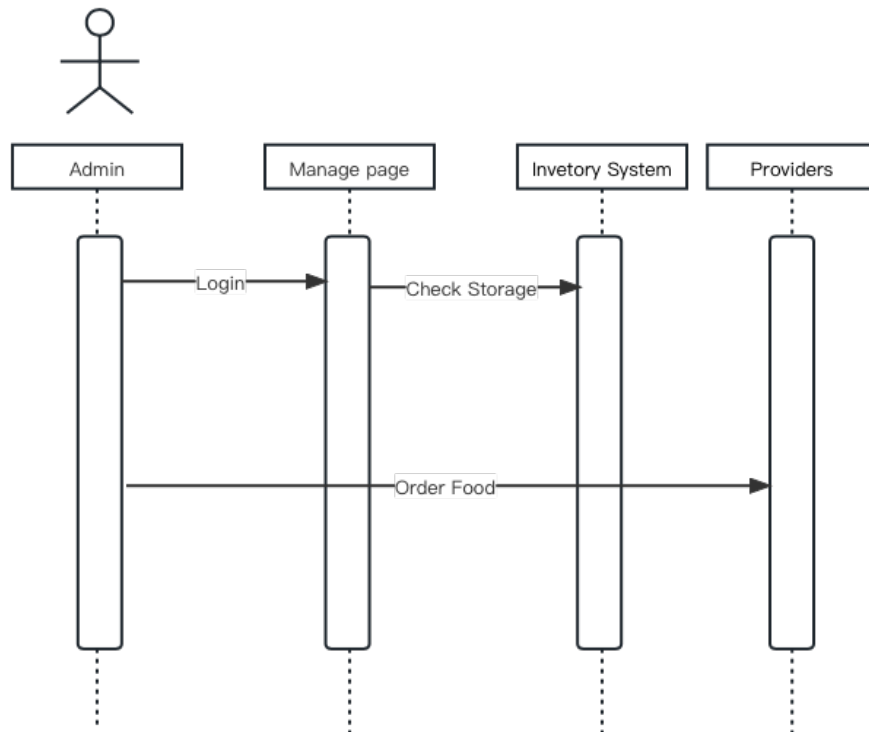


Figure 15: Sequence Diagrams of Managing Storage- admin need to check storage of ingredients regularly, if some Menus are ordered, which the storage is lack of ingredients, admin need to order them through other providers.

4.7 State Machine Diagrams

The state machine diagram mainly contains two modules: user account management and booking.

1. User account management: In this part of the user, the new user registers through Register(), and then the user logs in to check whether the account number and password are correct. After the verification is successful, the account status will be checked. If it is Inactive, the user will be asked to contact the administrator again for activation processing. If it is Active, the next step can be performed. In the above process, if the status of each step is determined to be False, it will return to the previous step. At the same time, after verifying that the login is successful, the user has the right to modify the account information and delete the account.

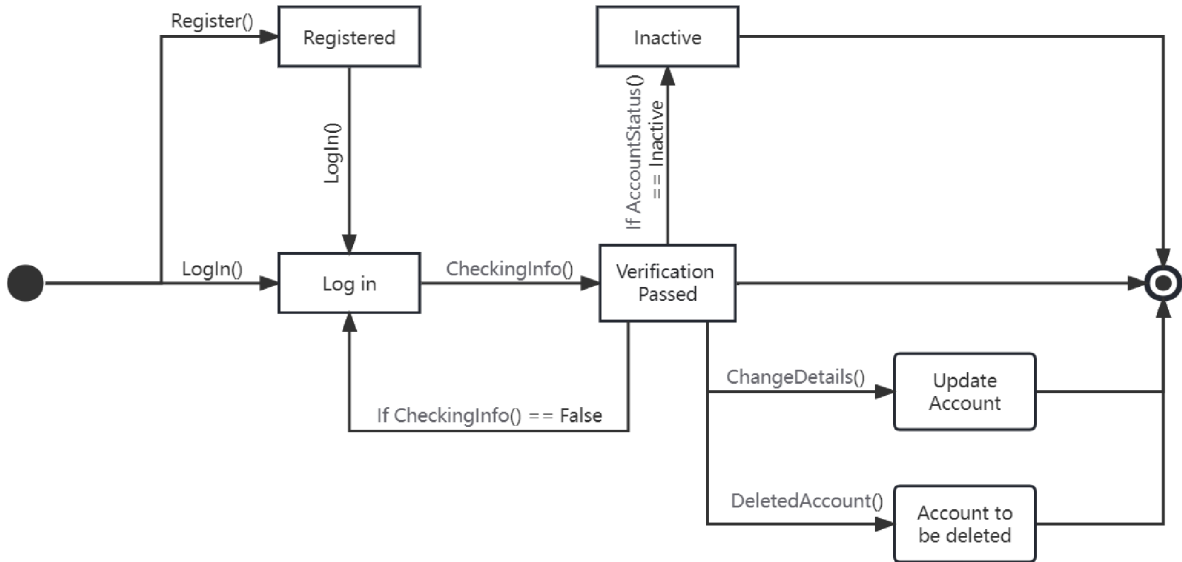


Figure 16: State Machine Diagram for User account management-Register, Login, Verify, and Modify Account Information.

2. In the Booking module, the school can directly create an order for payment, confirm the completion status after the order is completed, and give feedback and evaluation on the order. You can choose to cancel the order during the order creation and payment stages. Since the school order may have the problem of customizing the dishes, the Customization() function is specially designed to support the school to communicate with the merchant to customize the dishes. If the communication reaches an agreement, the order will be generated through the AcceptCustom() jump, and if rejected, the Rejected() cannot Generate orders. At the same time, because the school often needs the same order multiple times, the function of Create Another Same Order is set after the order is completed, so that the school can easily create another same order as before.

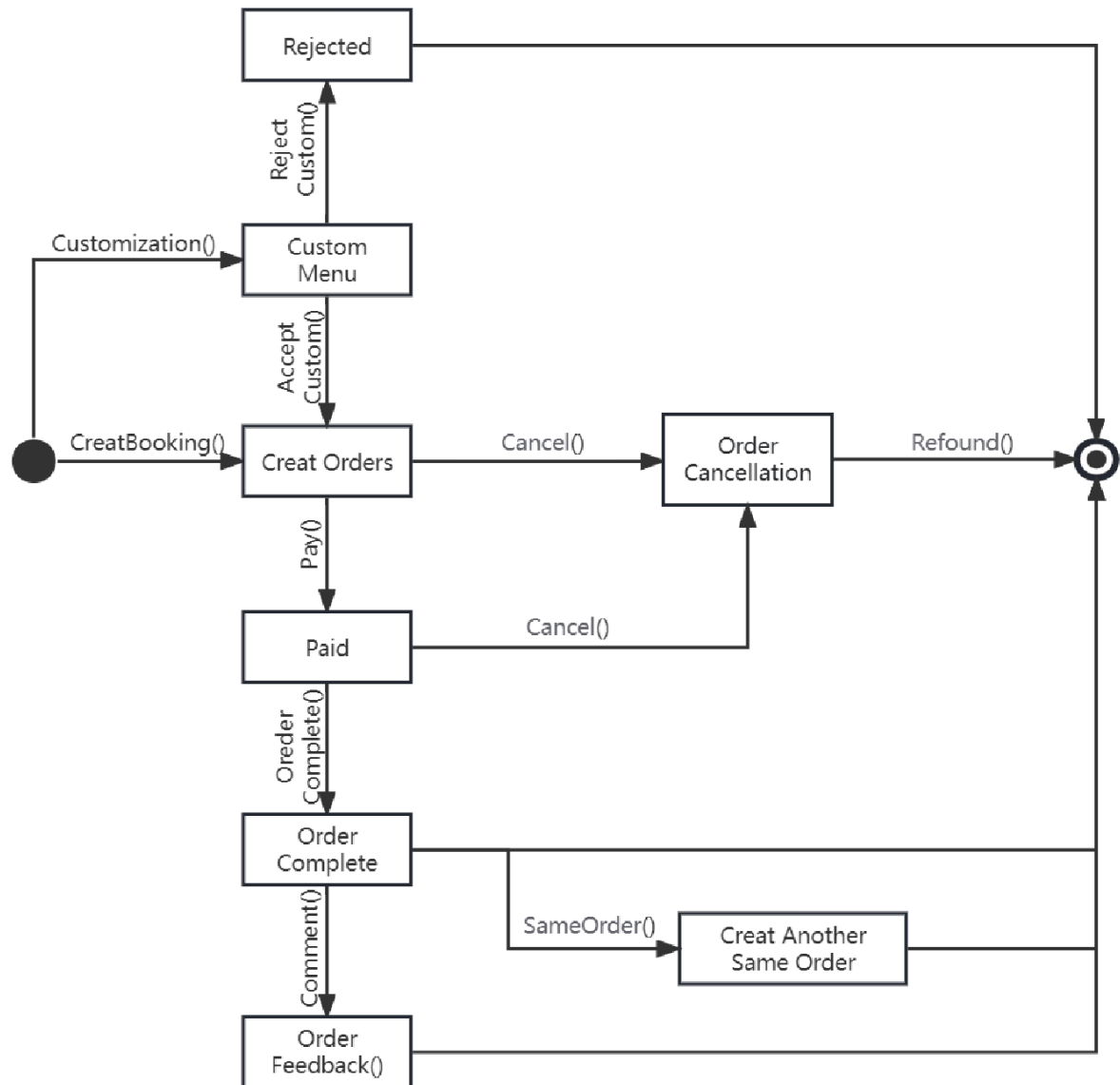


Figure 17: State Machine Diagram for Booking- Order, Payment, Customization, and Feedback with Flexibility.

5 Application Mock-up

5.1 Overview

A design prototype is a tangible representation that enables interested parties to engage with it and evaluate its appropriateness (Odom, 2016). In the following chapter, an illustration is provided for the prospective interface of Food Catering website. For the clients, a series of web pages are presented, from registration to selecting the delivery time of the food. It also shows the case lists of administrators to browse orders and manage users. The 7 pivotal usage scenarios provided later is as following:

- HomePage
- Use Case 1 - CustomerRegistration (Clients)
- Use Case 2&3 - CustomerLogin & AdminLogin (Clients&Admin)
- Use Case 7&8 - SeeOrderStatus & SeePreviousOrders (Clients)
- Use Case 13 - EnactmentDelivery (Clients)
- Use Case 14 - EditPromotion (Admin)
- Use Case 19 - ManageCustomerAccount (Admin)
- Use Case 22 - OrderFoodFromOtherProviders (Admin)

5.2 Mock Up

5.2.1 Use Case 1 - CustomerRegistration (Client)

The customer can register by specifying the required variables. The username and password fields check whether the username is in use or the password is weak. After clicking "Register", the client has an account and is logged in.

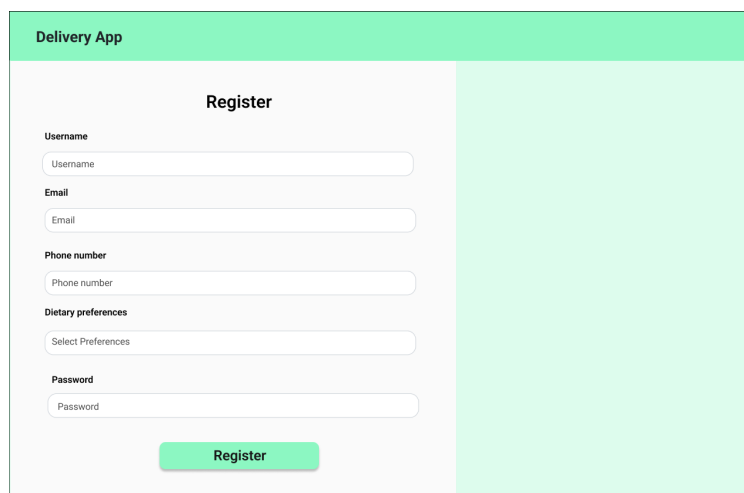
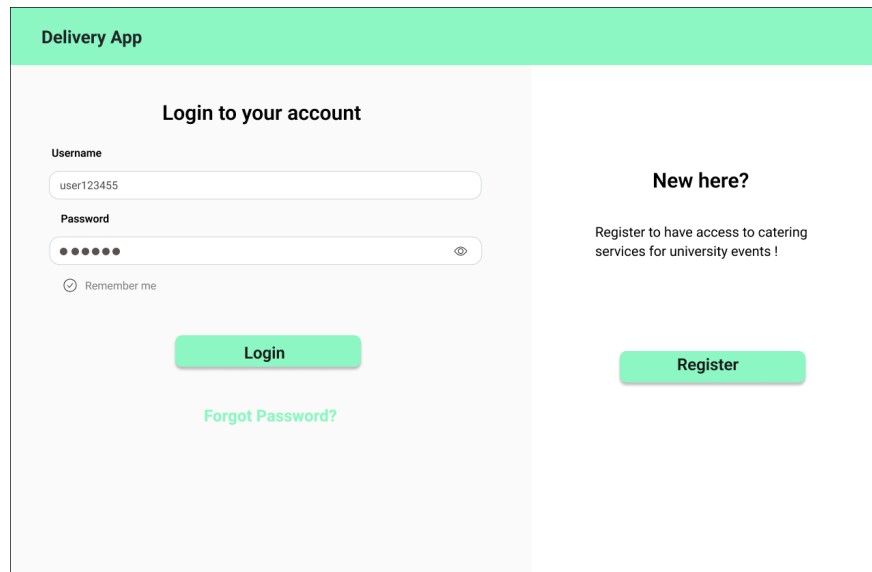


Figure 18: Mock-up for the Customer Registration page

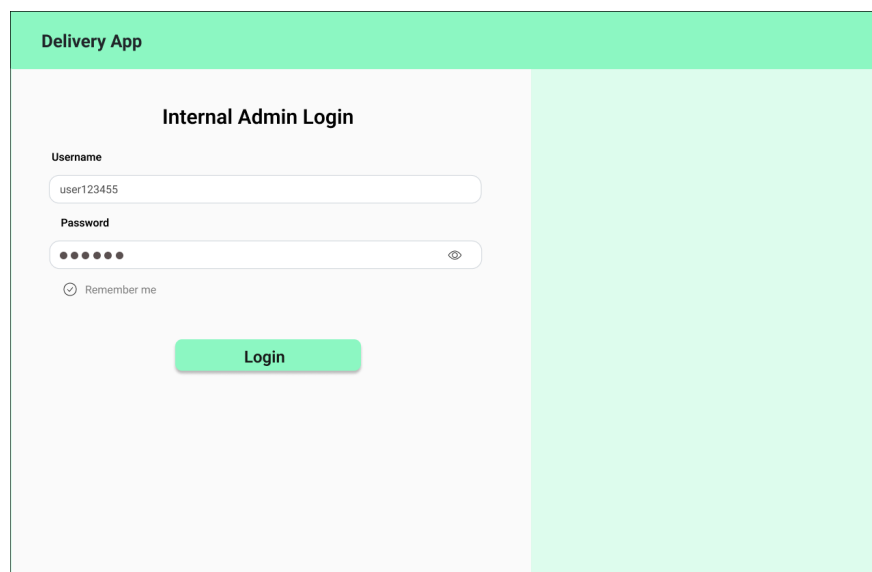
5.2.2 Use Case 2&3 - CustomerLogin & AdminLogin (Clients&Admin)

To log in, both the customers and the admins need to be registered with a username and password. The admin login is an internal page visible only to the admin. The password visibility can be specified by the user and there is a "Remember me" function. The system checks whether the username and passwords match and log in the user if they do, otherwise shows an error.



The mock-up for the Customer Login page features a green header bar labeled "Delivery App". The main content area is split into two columns. The left column, titled "Login to your account", contains a "Username" field with the text "user123455", a "Password" field with masked characters and a toggle icon, a "Remember me" checkbox, a green "Login" button, and a green link "Forgot Password?". The right column, titled "New here?", contains the text "Register to have access to catering services for university events !" and a green "Register" button.

Figure 19: Mock-up for the Customer Login page



The mock-up for the Admin Login page features a green header bar labeled "Delivery App". The main content area is split into two columns. The left column, titled "Internal Admin Login", contains a "Username" field with the text "user123455", a "Password" field with masked characters and a toggle icon, a "Remember me" checkbox, and a green "Login" button. The right column is a solid light green rectangle.

Figure 20: Mock-up for the Admin Login page

5.2.3 User Case 7&8 - SeeOrderStatus & SeePreviousOrders (Clients)

After customer checkout their order, they will be directed to the order's status page, which will show the details of the order. And it also allows the customer to check the status of the order, which includes four statuses, order placed, preparing, delivering, and completed. The colour of the circle will change from grey to green to indicate the status is completed. Also, if the order is not delivered, the customer can still cancel the order.

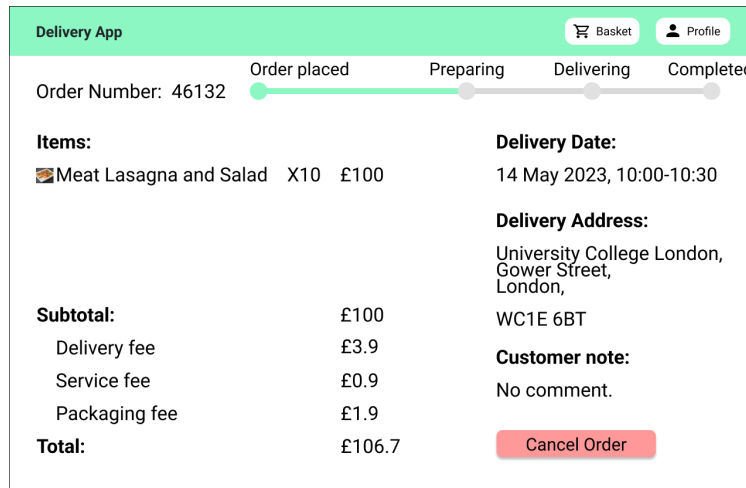


Figure 21: Mock-up for the customer views uncompleted order page

Once the order is completed, the customer can still view it through the previous order page. This page is similar to the order's status page, but the customer can place the order again by clicking the 'Order Again' button and can export the invoice in a PDF format.

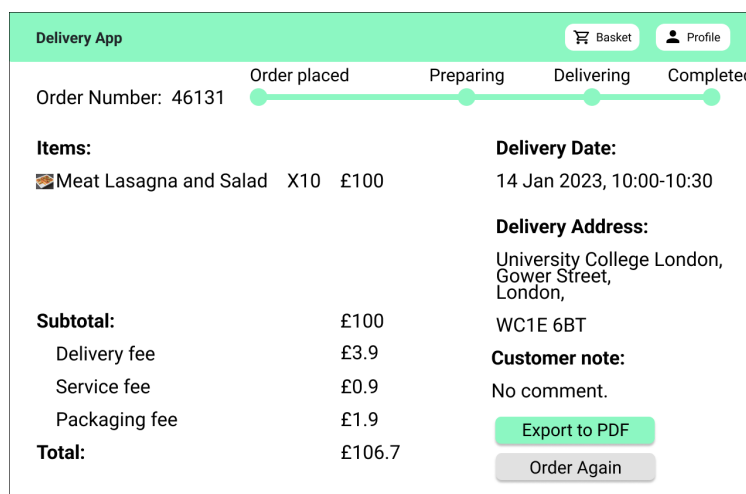


Figure 22: Mock-up for the customer views completed order page

5.2.4 Use Case 13 - EnactmentDelivery (Client)

After browsing and having added the preferred food, the user would be directed to the details confirmation page. This page enables users to select the expected delivery date and specific time during that day. The client can manage the amount of food and view the price and additional fees. Logged-out viewers are also free to view available delivery options, and they will be required to log in before making payment.

The mock-up shows a 'Delivery App' interface. At the top, there's a green header with 'Delivery App', a search bar, a 'Basket' icon, and a 'Login' button. Below the header, the 'Your basket' section displays 'Meat Lasagna and Salad' with a quantity of 10 and a price of £100. To the left, a calendar for May 2023 shows the 14th selected. To the right, 'Available Time' slots are listed: 10:00-10:30, 11:00-11:30, and 11:30-12:00. A 'Check Out' button is at the bottom right. A summary table on the right lists fees: Delivery fee (£3.9), Service fee (£0.9), Packaging fee (£1.9), and a Total of £106.7. A link 'See more available options' is also present.

Item	Quantity	Price
Meat Lasagna and Salad	10	£100
Subtotal		£100
Delivery date	May 2023 (14th selected)	
Available Time	10:00-10:30, 11:00-11:30, 11:30-12:00	
Fees		
Delivery fee		£3.9
Service fee		£0.9
Packaging fee		£1.9
Total		£106.7

Figure 23: Mock-up for the Checkout page for selecting the specific delivery date and time

5.2.5 Use Case 14 - EditPromotion (Admin)

Once the admin has logged in, they are able to add or remove promotions for foods in their canteens. This page provides information about original and current prices and allows the admin to edit promotions details. After clicking 'Confirm Change', the sales prices would be applied.

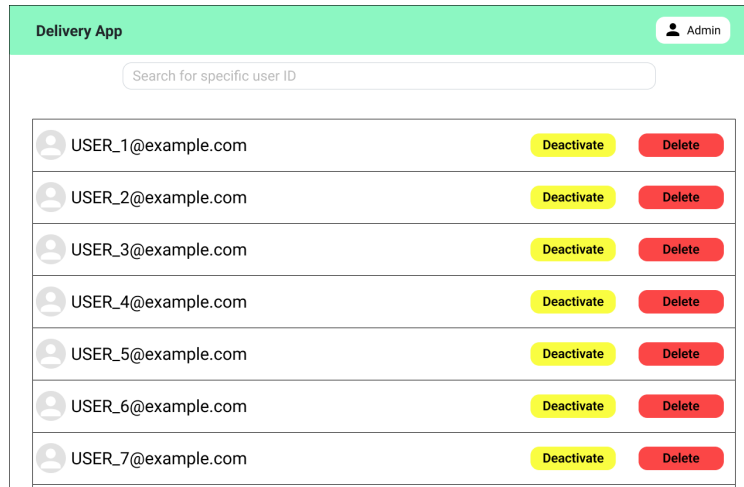
The mock-up shows an 'Admin' interface for 'Manage Promotion'. It features a grid of four food items: 'Meat Lasagna and Salad', 'Cumberland Sausages and Mash', 'Chicken with Mushroom Sauce and Potatoes', and 'Chicken Makhani and Basmati Rice'. Each item displays its 'Original price' and 'Current price', along with an 'Add Promotion' button (with a percentage input) and a 'Delete Current Promotion' link. A 'Confirm Change' button is at the bottom right. A link 'See more available food' is also present.

Food Item	Original Price	Current Price	Action
Meat Lasagna and Salad	£14	£10	Add Promotion: %percent, Delete Current Promotion
Cumberland Sausages and Mash	£13	£14	Add Promotion: %percent
Chicken with Mushroom Sauce and Potatoes	£15	£15	Add Promotion: %percent
Chicken Makhani and Basmati Rice	£15	£15	Add Promotion: %percent

Figure 24: Mock-up for the admin edits promotion page

5.2.6 Use Case 19 - ManageCustomerAccount (Admin)

Once the admin has logged in, they can manage the customers' account through this manage customers' account page. Admin can choose a specific customer account by using the search bar or finding it in the table directly. Then, the admin can choose to deactivate or delete the selected customer account, and the admin can also reactivate the account if the account is deactivated.

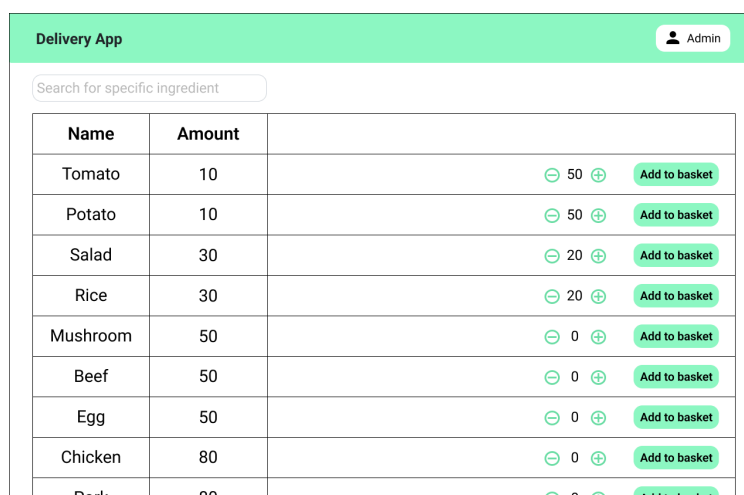


Delivery App		Admin
Search for specific user ID		
USER_1@example.com	Deactivate	Delete
USER_2@example.com	Deactivate	Delete
USER_3@example.com	Deactivate	Delete
USER_4@example.com	Deactivate	Delete
USER_5@example.com	Deactivate	Delete
USER_6@example.com	Deactivate	Delete
USER_7@example.com	Deactivate	Delete

Figure 25: Mock-up for the admin manages the customers' account page

5.2.7 Use Case 22 - OrderFoodFromOtherProviders (Admin)

Once the admin has logged in, they can check the amounts of ingredients and selects to purchase the required ingredients on this page, which will show all ingredients sorted in descending order according to the total amount. Also, the admin can search the specific ingredient by using the search bar. Then the admin can select the amounts of ingredients to add to the basket. After that, the system will send the order to the third-party suppliers.



Delivery App		Admin
Search for specific ingredient		
Name	Amount	
Tomato	10	⊖ 50 ⊕ Add to basket
Potato	10	⊖ 50 ⊕ Add to basket
Salad	30	⊖ 20 ⊕ Add to basket
Rice	30	⊖ 20 ⊕ Add to basket
Mushroom	50	⊖ 0 ⊕ Add to basket
Beef	50	⊖ 0 ⊕ Add to basket
Egg	50	⊖ 0 ⊕ Add to basket
Chicken	80	⊖ 0 ⊕ Add to basket
Pork	80	⊖ 0 ⊕ Add to basket

Figure 26: Mock-up for the page where the admin purchases ingredients from third-party suppliers

6 Summary and Conclusions

6.1 Summary

There is a substantial market demand for catering to universities, which often host academic conferences or significant events throughout the day. The report describes a software engineering project aimed at developing an app enabling universities to order food during events such as conferences and seminars. The software is designed to be Web - and application-based, flexible, scalable, easy to maintain and manage, and to meet design principles and security such as reliability, robustness, modifiability, ease of understanding, simplicity, testability, efficiency, standardization, complexity, scalability (Neill, Sangwan and Paulish, 2009). Catering companies offer a variety of menus and discounts, as well as the ability to deliver orders on specific dates and times. Colleges can create a profile and fill out their dining preferences to keep track of their orders and preferences (vegetarian, vegan, etc.) and even customize menus. They can also export the detailed invoice when they place the order.

The report comprises several sections: introduction and problem statements, requirements, use cases, object-oriented analysis and design models, and application and application models. The project approach uses agile methods to enhance teamwork, self-organization, continuous iteration, and rapid feedback. The report concludes with a summary and conclusion, a "Who did What" summary and an appendix. The project aims to provide universities with a simplified and efficient ordering process for purchasing food from universities and catering companies.

6.2 Conclusion

After this period of cooperation between our team and the client, this application about providing meal reservations for university events has gone smoothly. The client's needs are straightforward and clear, from sorting the needs into MoSCoW to showing the use case to client and then to prototype display. Each step is gradually improving our application. Everyone has a decent division of labour and clear goals, while we still met a problem on varying chart presentation requirements in this project. For example, some team members think that according to the guidelines, others have emphasized the importance of logical integrity in the process. Specifically, these members have suggested that we should include steps to allow unregistered clients to access our application. Finally, after everyone's unanimous discussion, some registration and login steps are reserved for showing the complete login process but do not pay too much attention to these steps. Decreases the readability of the report.

Through the UML diagram, we provide a complete application generation framework for the entire application, which reflects how each process in the application should be displayed, which saves us much time in drawing prototype diagrams. The prototype diagrams are modified through group discussions and discussions with the client.

Overall, the project is well done, and we look forward to your feedback.

References

- [1] Ali, N.H., Shukur, Z. and Idris, S. (2007). A Design of an Assessment System for UML Class Diagram. [online] IEEE Xplore. Available at: <https://ieeexplore.ieee.org/abstract/document/4301193> [Accessed 20 Apr. 2020].
- [2] Boubekur, Y. and Mussbacher, G. (2020). Towards a better understanding of interactions with a domain modeling assistant. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings.
- [3] Eshuis, R. and Wieringa, R. (2004). Tool support for verifying UML activity diagrams. IEEE Transactions on Software Engineering, 30(7), pp.437–447.
- [4] Mohammadi, R.G. and Barforoush, A.A. (2014). Enforcing component dependency in UML deployment diagram for cloud applications. 7'th International Symposium on Telecommunications (IST'2014).
- [5] Mythily, M., Valarmathi, M.L. and Durai, C.A.D. (2018). Model transformation using logical prediction from sequence diagram: an experimental approach. Cluster Computing, 22(S5).
- [6] Neill, C.J., Sangwan, R.S. and Paulish, D.J. (2009). 2.3.2 An Architecture-Centric Approach for Systems Design. INCOSE International Symposium, 19(1), pp.299–310.
- [7] Odom, W. et al. (2016). “From research prototype to research product,” Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.
- [8] Schwaber, K. (2005). Agile Project Management. Extreme Programming and Agile Processes in Software Engineering.
- [9] Sharp, H., Rogers, Y. and Preece, J. (2019). Interaction design : Beyond human-computer interaction. 5th ed. Indianapolis, In: Wiley.
- [10] Tiwari, S. and Gupta, A. (2015). A systematic literature review of use case specifications research. Information and Software Technology, 67, pp.128–158.

Appendix A: Individual contribution summary

Chapter	Yijie Chen	Jianing He	Muyao Li	Yangzi Li	Tania Turdean	Yaojue Xiong	Wenjin Yang
1.1 Report Introduction				✓			
1.2.1 Project Brief						✓	
1.2.2 Problem Statement						✓	
1.2.3 Project Scope						✓	
1.3.1 Development Process		✓					
1.3.2 Gantt Chart		✓					
1.4 Glossary						✓	
2.1 Overview							✓
2.2 Similar Product Analysis					✓		
2.3 MoSCoW Style Requirements	✓						✓
2.4 Domain Model			✓				
3.1 Overview							✓
3.2 Use Case List	✓						

Chapter	Yijie Chen	Jianing He	Muyao Li	Yangzi Li	Tania Turdean	Yaojue Xiong	Wenjin Yang
3.3 Use Case Diagram			✓				
3.4 Use Case Specifications	✓		✓		✓	✓	
4.1 Overview				✓			
4.2 Object-Oriented Analysis				✓			
4.3.1 Class Diagram		✓					
4.3.2 Component Diagram		✓					
4.4 Deployment Diagram							✓
4.5 Activity Diagrams							✓
4.6 Sequence Diagrams		✓		✓			✓
4.7 State Machine Diagrams		✓					
5.1 Overview						✓	
5.2.1 UC 1					✓		
5.2.2 UC 2& 3					✓		
5.2.3 UC 7& 8	✓						

Chapter	Yijie Chen	Jianing He	Muyao Li	Yangzi Li	Tania Turdean	Yaojue Xiong	Wenjin Yang
5.2.4 UC 13						✓	
5.2.5 UC 14			✓			✓	
5.2.6 UC 19	✓						
5.2.7 UC 22	✓						
6.1 Summary							✓
6.2 Conclusion							✓