

**KENYATTA UNIVERSITY**

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTING, INFORMATION AND TECHNOLOGY**

**SUBMITTED BY**

**SCO 400: PROJECT PROPOSAL**

**TITLE: ONLINE CUSTOMIZABLE FOOD ORDERING SYSTEM**.

**NAME**: **HASSAN ABDISHAKOOR**

**REG. NO**: **J17/0796/2019**

**SUPERVISOR**: **MR. KENNEDY SIIKA**

(This project proposal is submitted in partial fulfillment of the requirements for a Bachelor of *Science (Computer Science)/Bachelor of Information Technology etc.* Degree

at Kenyatta University)

# 

Contents

[CHAPTER ONE 4](#_Toc140446697)

[**1.0 INTRODUCTION** 4](#_Toc140446698)

[**1.1 Background of the project** 4](#_Toc140446699)

[**1.2 Problem Statement** 4](#_Toc140446700)

[**1.3 Objectives** 4](#_Toc140446701)

[**1.4 Scope and Limitation of Study** 5](#_Toc140446702)

[**1.5 Justification** 5](#_Toc140446703)

[CHAPTER TWO 6](#_Toc140446704)

[**2.0 Literature Review** 6](#_Toc140446705)

[**2.1 Introduction** 6](#_Toc140446706)

[**2.2 Case study** 6](#_Toc140446707)

[**2.3 Proposed System** 8](#_Toc140446708)

[CHAPTER THREE 9](#_Toc140446709)

[**3.0 Methodology** 9](#_Toc140446710)

[**3.1 Design and Methodology** 9](#_Toc140446711)

[**3.2 Research Methods** 11](#_Toc140446712)

[**3.3 Data Collection Techniques** 11](#_Toc140446713)

[**3.4 Development Tools** 11](#_Toc140446714)

[**3.5 Tools for Testing** 11](#_Toc140446715)

[**3.6Time Schedule and Project cost** 12](#_Toc140446716)

[CHAPTER FOUR 13](#_Toc140446717)

[**4.0 SYSTEM ANALYSIS AND REQUIREMENT MODELING** 13](#_Toc140446718)

[**4.1 Methodology** 13](#_Toc140446719)

[**4.2 Existing system** 16](#_Toc140446720)

[**4.3 Proposed System** 19](#_Toc140446721)

[**4.4 Data Collection Methods** 25](#_Toc140446722)

[CHAPTER FIVE 28](#_Toc140446723)

[**5.0 SYSTEM DESIGN** 28](#_Toc140446724)

[**5.1 Introduction** 28](#_Toc140446725)

[**5.2 Architecture** 28](#_Toc140446726)

[**5.3 Process design** 29](#_Toc140446727)

[**5.4 Database design** 31](#_Toc140446728)

[**5.5 User Interface Design** 40](#_Toc140446729)

[CHAPTER SIX 44](#_Toc140446730)

[**6.0 SYSTEM IMPLEMENTATION** 44](#_Toc140446731)

[**6.1 Tools used for coding** 44](#_Toc140446732)

[**6.2 Testing** 45](#_Toc140446733)

[**6.3 Proposed Change-over Techniques** 46](#_Toc140446734)

[CHAPTER SEVEN 47](#_Toc140446735)

[**7.0 LIMITATIONS, CONCLUSIONS AND RECOMMENDATIONS** 47](#_Toc140446736)

[**7.1 LIMITATIONS** 47](#_Toc140446737)

[**7.2 CONCLUSIONS** 47](#_Toc140446738)

[**7.3 RECOMMENDATIONS** 48](#_Toc140446739)

[REFERENCES 49](#_Toc140446740)

[APPENDICES 50](#_Toc140446741)

[**Organizational Structure** 51](#_Toc140446742)

[**Interesting piece of code** 52](#_Toc140446743)

[**Technical and User Manual** 59](#_Toc140446744)

# 

# CHAPTER ONE

## 

## **1.0 INTRODUCTION**

### **1.1 Background of the project**

The concept of an online customizable food ordering system was developed in response to a rising trend among consumers who choose to eat healthier foods when dining out. In recent years, consumers have become more health-conscious, with an increasing focus on maintaining a healthy diet and lifestyle. This has increased consumer demand for calorie-appropriate menu options at restaurants, as well as for more transparency in the calorie content of food items. The project aims to address these needs by providing a user-friendly platform that allows customers to customize their food and drink orders to meet their dietary requirements and preferences. Additionally, the platform offers precise and simple-to-find calorie information for each menu item, which might aid customers in choosing what to eat. This project can benefit both customers and participating restaurants, as it can attract health-conscious customers and increase sales for restaurants, while also promoting healthier eating habits. In general, the initiative represents the growing need for technologically based solutions to address this trend as well as the demand for healthier food options.

### **1.2 Problem Statement**

The current food ordering system at restaurants fails to cater to the growing demand for transparent calorie information, leading to customer dissatisfaction and lost sales. The customers who are on a weight-loss journey will be disadvantaged and may not eat at restaurant with fear of breaking their diets. As well as the bodybuilders who require large amounts of calories to keep up with their heavy lifting sessions are most likely not eating in restaurants. Some customers who have allergies may not enjoy their favorite foods and drinks with fear of feeling unwell.

### **1.3 Objectives**

The objectives of the project are:

* To develop a registry module for users for them to get the services.
* To develop a system that will contain information of several restaurants with their menus.
* To design a system that will have a secure database that will be used to store data of users and their ordering status.
* To embed a review system allowing users to provide feedback for their experiences.
* To provide accurate and easily accessible calorie information for each menu item and for the users.

### **1.4 Scope and Limitation of Study**

The intention of this study is to develop a system that has its functionalities spans through the Eastleigh, Nairobi which involves the design and development of an online platform that offers optimum-calorie food and drinks options, order customization, and calorie tracking. It includes menu information and management, order processing, and customer support and feedback. However, the study is limited by the availability of accurate and up-to-date menu information, limited participation of restaurants, technical limitations, limited user adoption, regulatory compliance and challenge in calorie calculations.

### **1.5 Justification**

The proposed online customizable food ordering system is justified by its potential to address a range of important societal concerns. One such concern is the increasing demand for healthier food options and the need to promote healthy eating habits. The system will provide a convenient and platform for customers to browse menus, customize orders, and view calorie information, making it easier for them to make healthier choices. This will also benefit participating restaurants by increasing customer satisfaction and sales, and enabling continuous improvement based on customer feedback. Moreover, with the growing trend of consumers seeking more plant-based and low-carbon footprint food options, the project can offer an additional benefit by encouraging restaurants to include more plant-based options on their menus. By providing customers with the option to customize their orders with low-calorie, plant-based, and sustainable food options, the system can contribute to reducing the carbon footprint associated with the food industry, as well as promoting healthy eating habits. Thus, the project aligns with the growing interest in sustainable and environmentally-friendly practices and can contribute to a more sustainable future while addressing a significant public health issue.

# CHAPTER TWO

## **2.0 Literature Review**

### **2.1 Introduction**

Online ordering systems have become increasingly popular in the food industry. These systems allow customers to easily and quickly order food from their desired restaurants. In recent years, a new trend has emerged where customers can customize each food item on the menu, including the option to select low-calorie options. This review of literature aims to examine the existing studies on the online customizable ordering system and its implications for the food industry.

A study by Wang, Wu, and Li (2020) proposed a design for an online ordering system for restaurants. The system would provide customers with a platform to customize their orders, which would help them make healthier choices. Similarly, Dai, Hu, and Feng (2019) developed a personalized food ordering and nutrition tracking system, which allowed customers to track the nutritional value of their meals. By educating users on dietary options, this technology has the potential to improve consumers' health results.

Furthermore, Yuen, Lai, and Tse (2019) investigated the adoption and challenges of online food ordering systems by restaurants. They found that while the systems have benefits, such as increased sales and customer satisfaction, restaurants face challenges in implementing and maintaining them. Additionally, Demartini, Pau, and Massa (2018) conducted an empirical investigation into the factors influencing the use of online food ordering and delivery services in Italy. They discovered that the choice to use these systems is influenced by elements including convenience, trust, and quality.

In conclusion, the research indicates that businesses and customers may both profit from online configurable ordering systems. The systems can help customers achieve better health outcomes by letting them choose healthful foods. Additionally, restaurants might gain from higher sales and happier patrons. However, restaurants may face difficulties with the installation and upkeep of these devices. This evaluation emphasizes the need for additional investigation into the merits and difficulties of online configurable ordering systems.

There are number of web systems that are in place for ordering food and giving calorie information at a go. The one under our study will be Nutritics.

### **2.2 Case study**

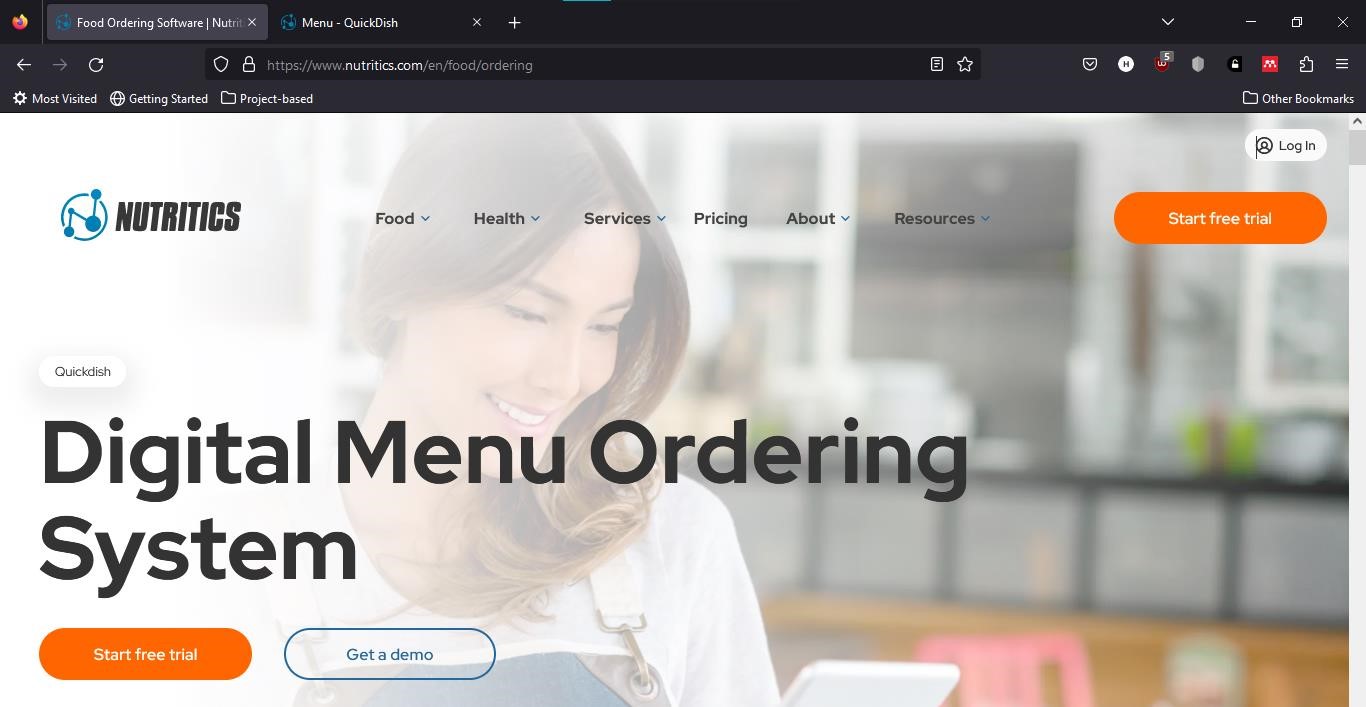
Nutritics is a comprehensive ordering system allows foodservice operators to manage click & collect, delivery and ordering (both in-house and in-queue) from one centralized platform. Nutritics deliver a seamless ordering experience across multiple locations. It also gives detailed allergen, calorie, nutrition and pricing information available and automatic updates as soon as your recipes or menu change.

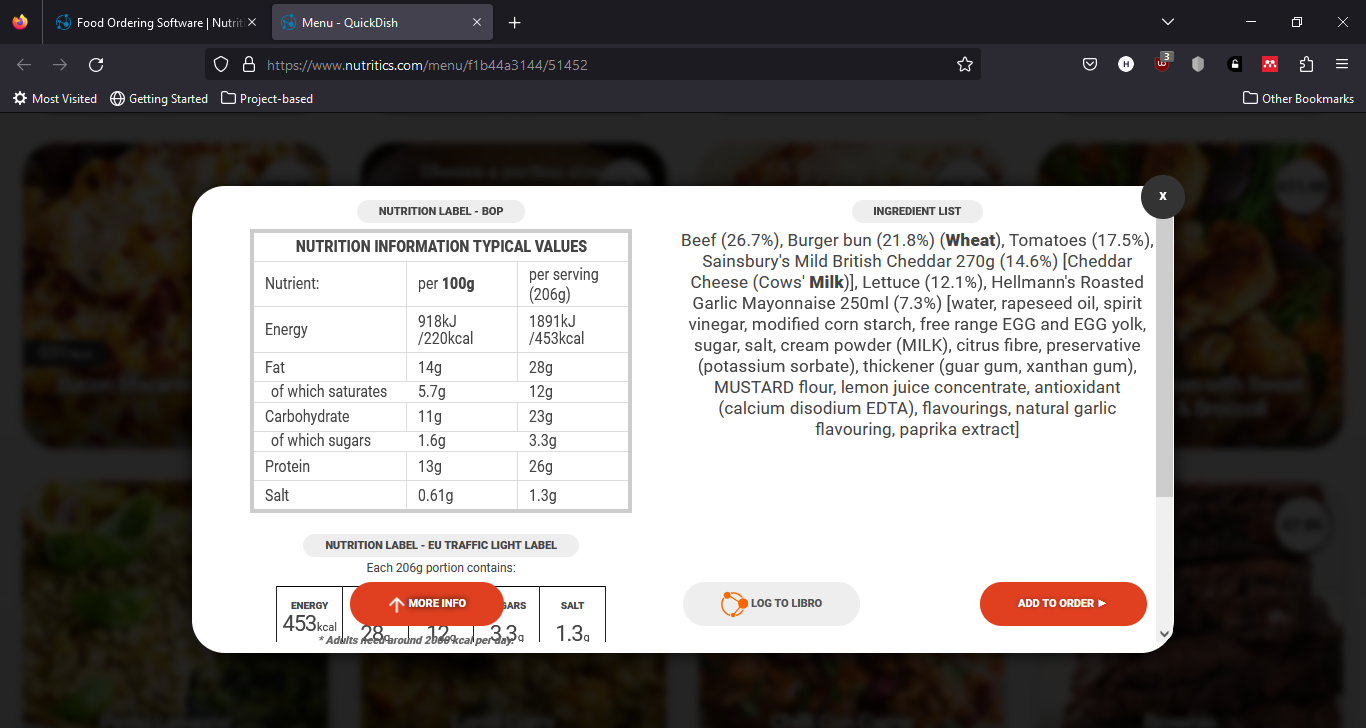
**Advantage of the web system:**

1. Simple for customer to navigate and order food and drinks.
2. Automatically updates customers menu changes for future references.
3. Gives full calorie information of foods and drinks.

**Disadvantages:**

1. Customers cannot substitute ingredients to their preferences.
2. Times for ordering some foods is limited.
3. Constant updating of menus may make customers confused as their previous selections may be lost.





### **2.3 Proposed System**

Build a web system that a user logs in whenever there is internet connectivity that does the following:

* Provide the calorie information of each item on the menu so that customers can make educated decisions.
* Eliminate the need for phone call or in-person orders to streamline the ordering process.
* Allow online order tracking and payment.
* Give information about well-liked menu items, client preferences, and order history to restaurant owners so they may make business decisions.

# CHAPTER THREE

## **3.0 Methodology**

### **3.1 Design and Methodology**

The approach or methodology to implement certain projects are numerous. Thus, some of these methodologies include:

**Waterfall Methodology**

The ancestor of all life cycle models. Each successive module in the system depends on the previous module being completed.

Stages of the project flows from the top to the bottom without going back to a finished stage. It involves sequence completing one task before moving on to the next, all the way to project completion. The goals and timelines are clearly defined for project delivery.

It follows a sequence of steps:

●Planning

●Analysis

●Design

●Development or Implementation

●Testing

●Deployment & Maintenance

**Merits of Waterfall Approach**

•Requirements are identified long before programming begins

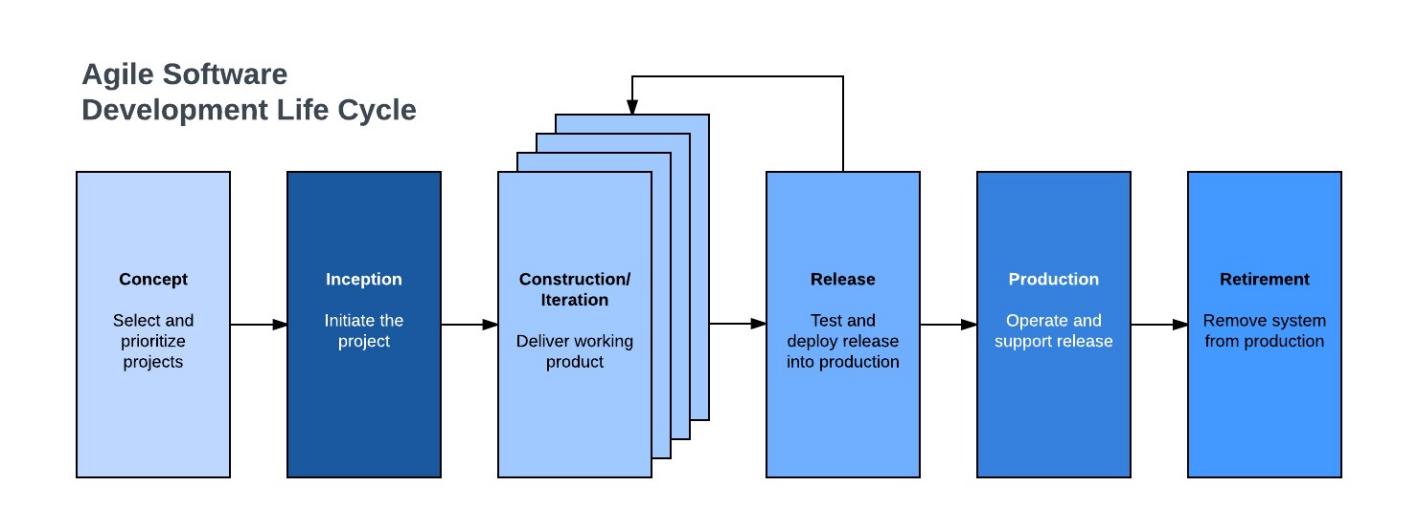
•Changes to the requirements as the project proceeds are limited.

**Demerits of Waterfall Approach**

* The design must be completely specified before programming begins.
* Time between system proposal completion (analysis phase), and system delivery is too long.
* Testing is treated almost as an afterthought in the implementation phase.

**Agile-Scrum Software Model**

Agile software methodology is an iterative approach that builds software incrementally from the start of the project rather than delivering it once. Agile models allow the use of increments or possible prototypes that can evolve into a more suited and validated requirements and eventually software application.



**SCRUM Process Model**

It is an agile process model which follows these activities: requirements, analysis, design, evaluation and delivery. Scrum emphasizes the use of a set of software process patterns that have been proven effective for projects with tight timelines, changing requirements and business criticality.

Why use agile for our project?

We will use Agile-Scrum as our methodological approach because it is:

* It is designed to curate the needs of rapidly changing environment by embracing the idea of incremental development and developing the actual final product.
* It requires constant comments from the user thus gives priority to the people than the process.
* Deployment and delivery are quicker thus gaining user confidence.

### **3.2 Research Methods**

We will use both the qualitative and quantitative research method since we will be aiming to get know more about gap between the customers, restaurants and their respective information needed.

### **3.3 Data Collection Techniques**

We will use one-on-one interviews, questionnaires and observation with the users around me that are willing to eat at restaurants and restaurants that are currently open and inquire what they would like a ordering system to have. Also ask if they have encountered any inconvenience in a ordering they made previously due to lack of prior information and lack of choosing sites.

### **3.4 Development Tools**

|  |  |
| --- | --- |
| Programming languages | HTML, CSS, JAVASCRIPT AND PYTHON |
| DBMS | MySQL Workbench |
| FRAMEWORK | Django |
| SERVER | Django built-in Server |

### **3.5 Tools for Testing**

1.Firefox: for debugging and testing web applications.

2. Django Testing Framework: Provides tools for writing unit tests and integration tests for Django applications.

3.Google Chrome: We will to test our JavaScript scripts.

### **3.6Time Schedule and Project cost**

**Project time span**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activity** | **Time (weeks)** | | | |  |  |  |  |  |  |  |  |  |  | |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | |
| Drafting concept paper |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Planning |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Proposal writing |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Deployment |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| Documentation |  | | | |  |  |  |  |  |  |  |  |  |  | |
|  |  | | | |  |  |  |  |  |  |  |  |  |  |  |

**Cost estimates**

|  |  |
| --- | --- |
| **ITEM** | **COST(KSHS)** |
| HP 820 G3 8GB RAM | 29,000 |
| Internet Services | 4500 |
| Printing Services | 1,700 |
| Miscellaneous | 3,300 |
| **TOTAL** | 38500 |

# CHAPTER FOUR

## **4.0 SYSTEM ANALYSIS AND REQUIREMENT MODELING**

### **4.1 Methodology**

**Design and Methodology**

The approach or methodology to implement certain projects are numerous. Thus, some of these methodologies include:

**Waterfall Methodology**

The ancestor of all life cycle models. Each successive module in the system depends on the previous module being completed.

Stages of the project flows from the top to the bottom without going back to a finished stage. It involves sequence completing one task before moving on to the next, all the way to project completion. The goals and timelines are clearly defined for project delivery.

It follows a sequence of steps:

●Planning

●Analysis

●Design

●Development or Implementation

●Testing

●Deployment & Maintenance

**Merits of Waterfall Approach**

•Requirements are identified long before programming begins

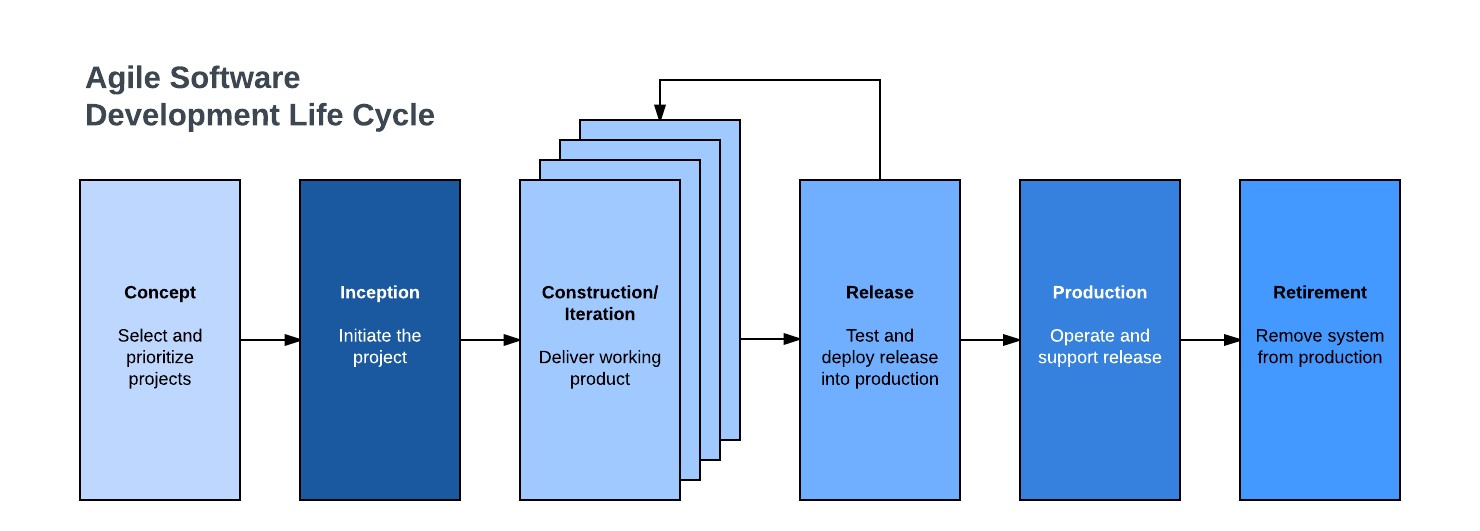
•Changes to the requirements as the project proceeds are limited.

**Demerits of Waterfall Approach**

* The design must be completely specified before programming begins.
* Time between system proposal completion (analysis phase), and system delivery is too long.
* Testing is treated almost as an afterthought in the implementation phase.

**Agile-Scrum Software Model**

Agile software methodology is an iterative approach that builds software incrementally from the start of the project rather than delivering it once. Agile models allow the use of increments or possible prototypes that can evolve into a more suited and validated requirements and eventually software application.



**SCRUM Process Model**

It is an agile process model which follows these activities: requirements, analysis, design, evaluation and delivery. Scrum emphasizes the use of a set of software process patterns that have been proven effective for projects with tight timelines, changing requirements and business criticality.

Why use agile for our project?

We will use Agile-Scrum as our methodological approach because it is:

* It is designed to curate the needs of rapidly changing environment by embracing the idea of incremental development and developing the actual final product.
* It requires constant comments from the user thus gives priority to the people than the process.
* Deployment and delivery are quicker thus gaining user confidence.

**Steps followed**

1. Product backlog creation: I produced a product backlog, which is a prioritized list of user stories, tasks, and features that needed to be completed, after identifying the product features, needs, and functions that were required to construct the Customizable Ordering System.
2. I chose a number of items from the Product Backlog to work on during the next sprint. The team then developed a Sprint Backlog, which is a list of tasks and activities needed to fulfill the chosen items during the sprint.
3. Working on the Sprint: Using Agile Scrum techniques, such as daily stand-up meetings where team members reported on their progress and noted any obstacles that needed to be overcome, the development team worked on the items listed in the Sprint Backlog.
4. Testing and Product Demonstration: The development team tested the product's features and functions once the sprint was complete to make sure they were operating as planned.
5. Retrospective and Next Sprint Planning: To assess the sprint and pinpoint opportunities for development, the team convened a retrospective meeting. This involved examining what went well and poorly throughout the sprint and determining any necessary adjustments to the development process. The procedure from step two was then repeated after choosing a fresh set of items from the Product Backlog for the subsequent sprint.

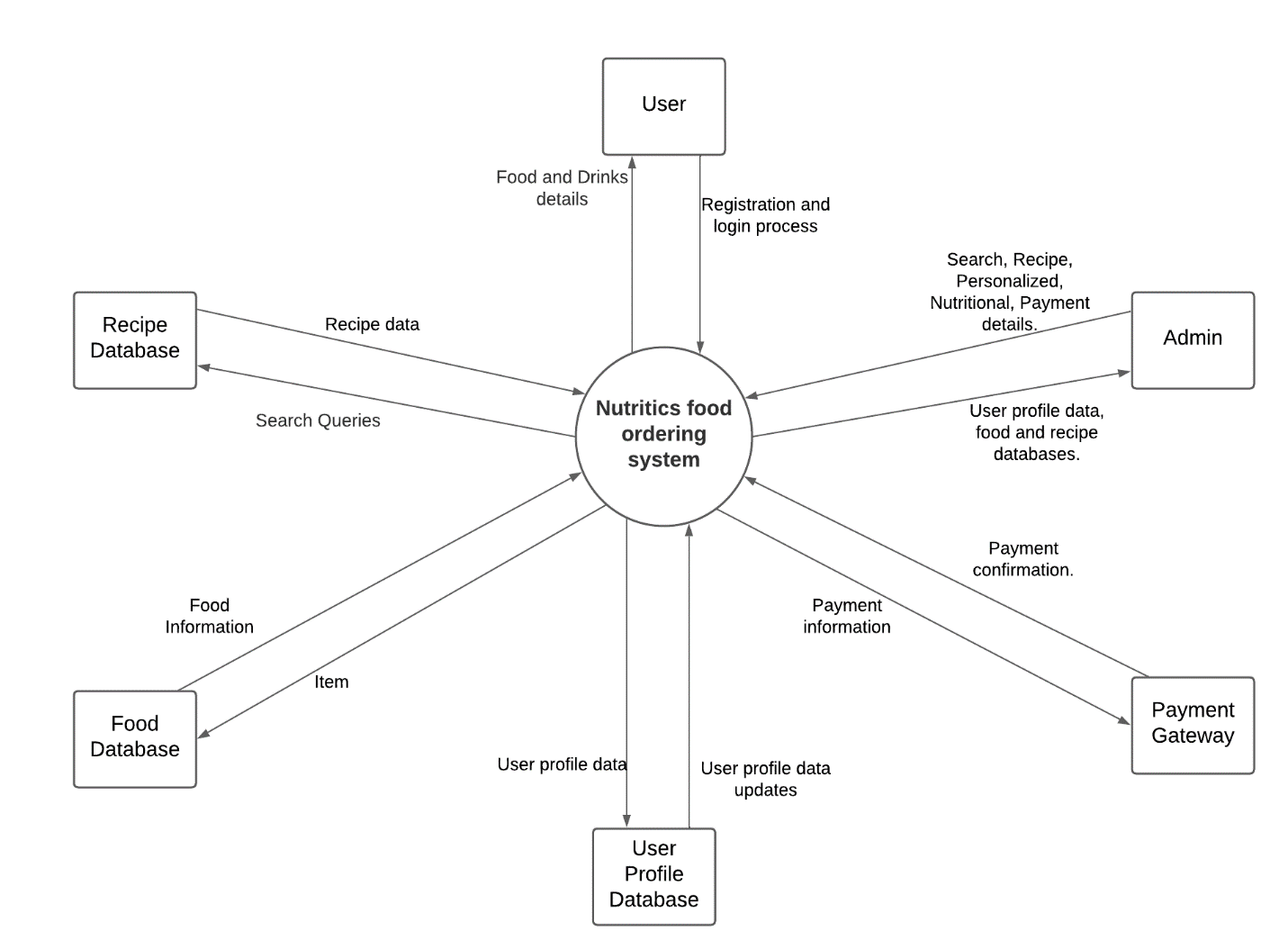
### **4.2 Existing system**

**How the existing system works.**

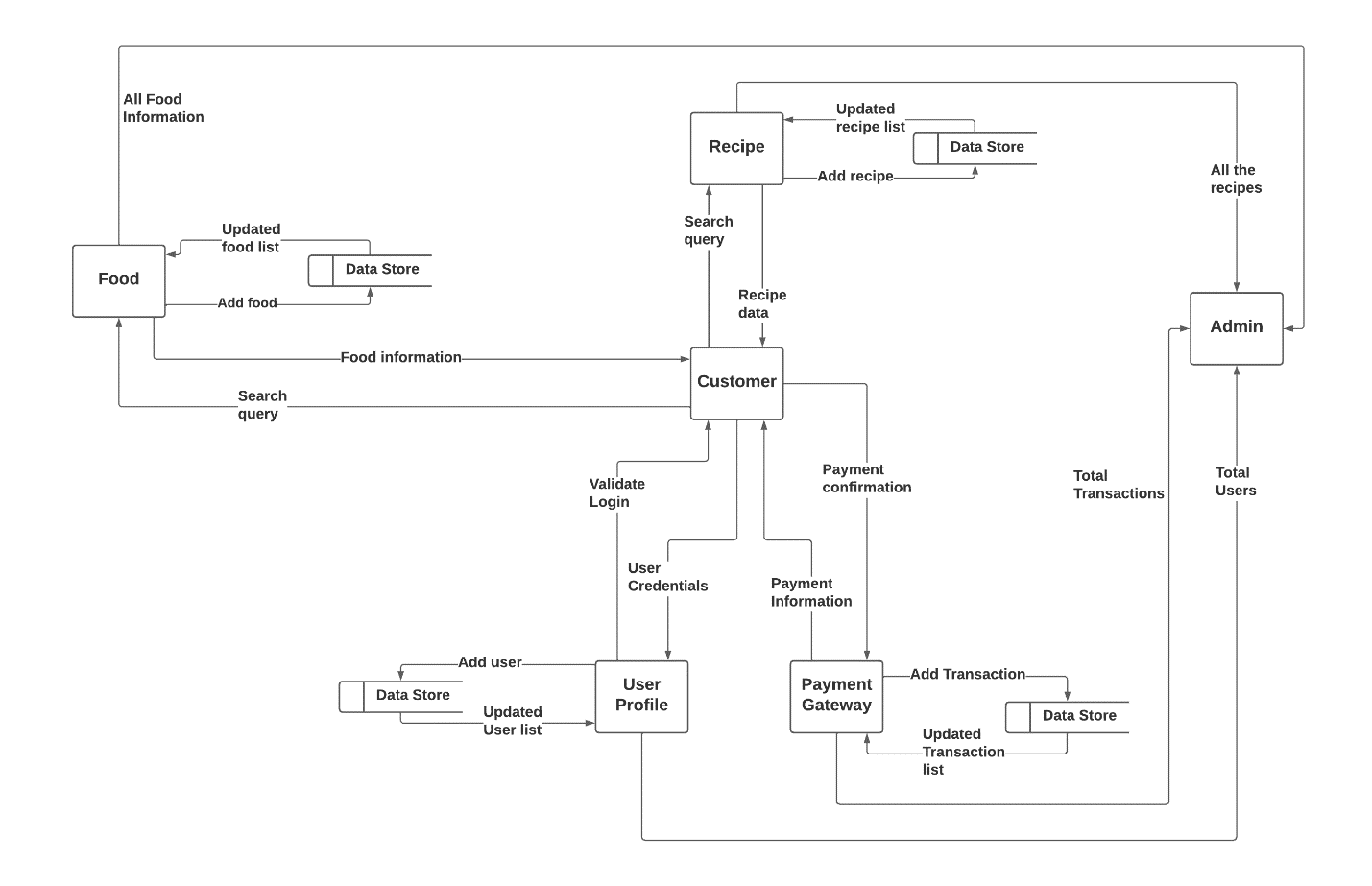
The process starts by users creating an account on the website. The user searches for food and drinks using keywords, then the system displays a list of results matching the search criteria. The search results show alongside them the nutritional information. The user selects the desired food or drink item and adds it to their meal plan or diary. The user can customize their meal plan by adjusting portion sizes, adding or removing items, and setting nutritional goals and then the order is finalized, readied for the user to come and take or be delivered. The system can generate a report showing the user's daily intake of calories, macronutrients, vitamins, and minerals, and compares it to their nutritional goals. The user can review the report and adjust their meal plan accordingly, based on their goals and preferences.

#### **4.2.1 Diagrams for Existing System**

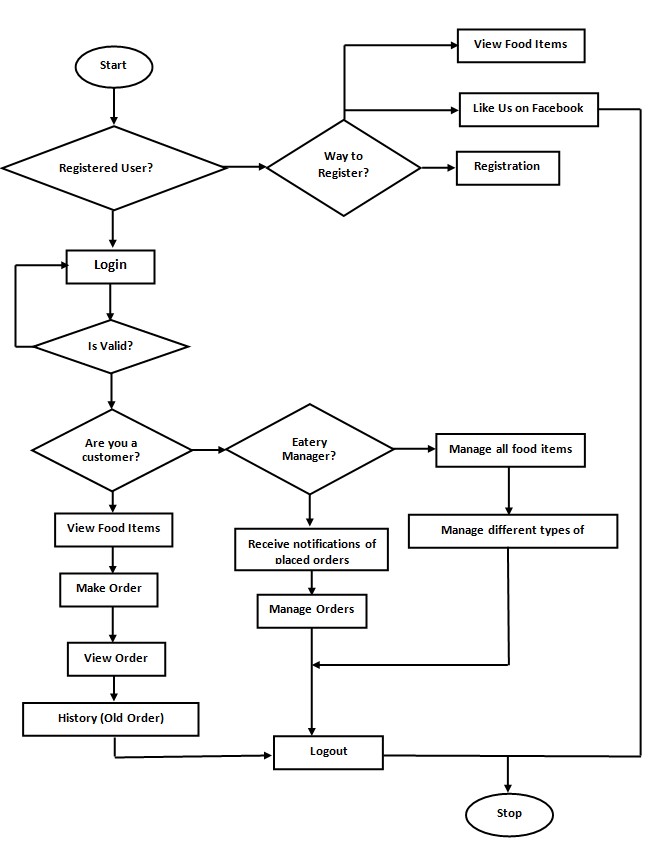
**Context Diagram**



**DFD level 1**



**Flowchart**



### **4.3 Proposed System**

#### **4.3.1 Functional Requirements**

1. Customers should be able to browse menus and add items to their order as well as customize the items to their liking.
2. Customers should be able to view the status of their orders, including when the orders were placed and when they are expected to be ready.
3. The system to should be able to support numerous restaurants and handle orders by sending them to the appropriate the restaurnts.
4. Restaurants should be able to publish their menus and update them anytime, where the menus show items available, calorie information and prices.
5. The system should be accessible and tailored for mobile use.
6. Customer should be able to examine their previous orders.
7. The system should allow the Admin to get report of customers, restaurants, orders and transactions.

#### **4.3.2 Non-functional Requirements**

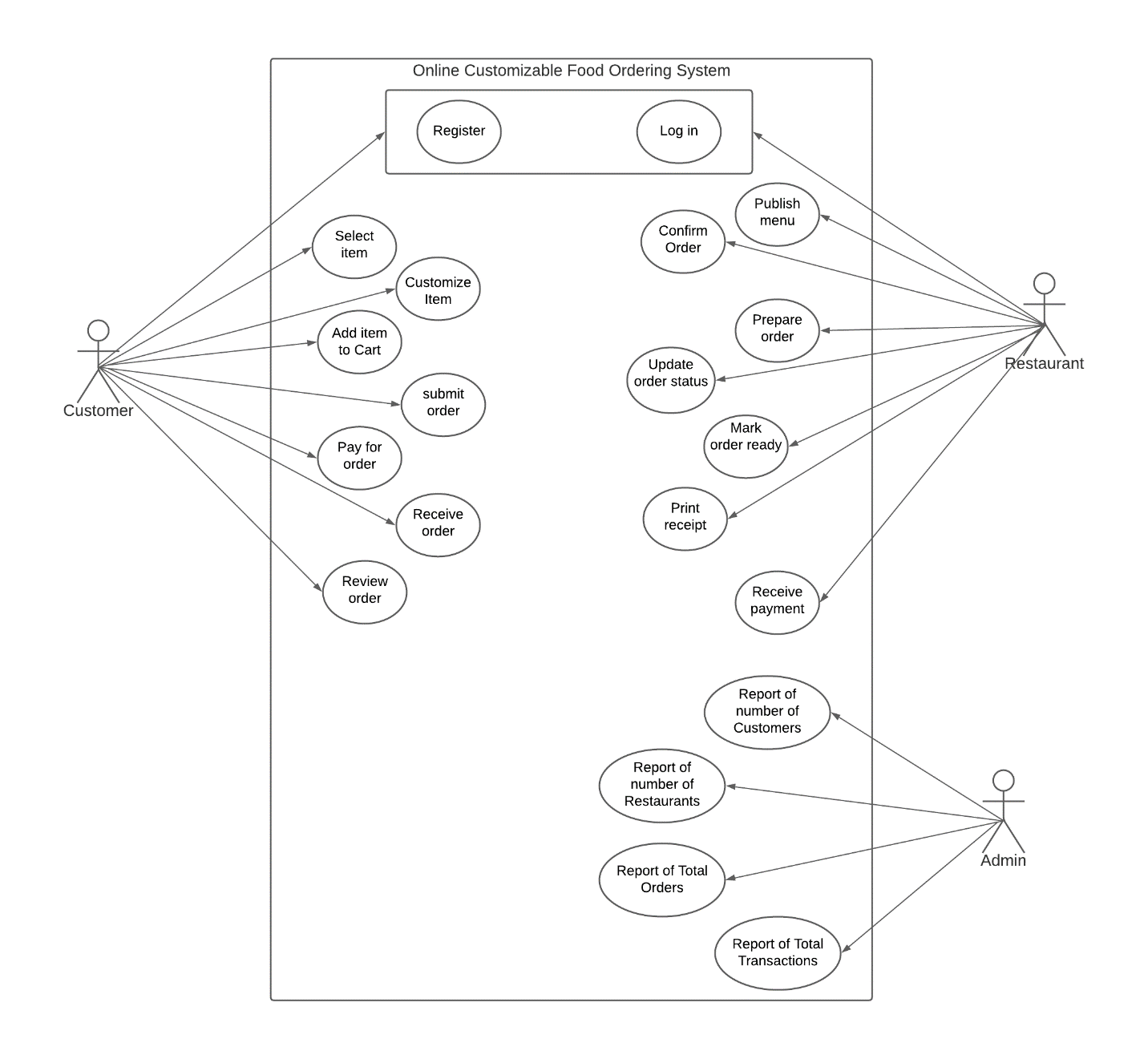
1. The user interface has to be simple and easy to use.
2. Performance: There should be no delays while the system is handling a high number of users.
3. The system should put in place the necessary security measures to safeguard the personal and private information of customers and restaurants.
4. The system should always be available and accessible to users.

#### **4.3.3 User requirements**

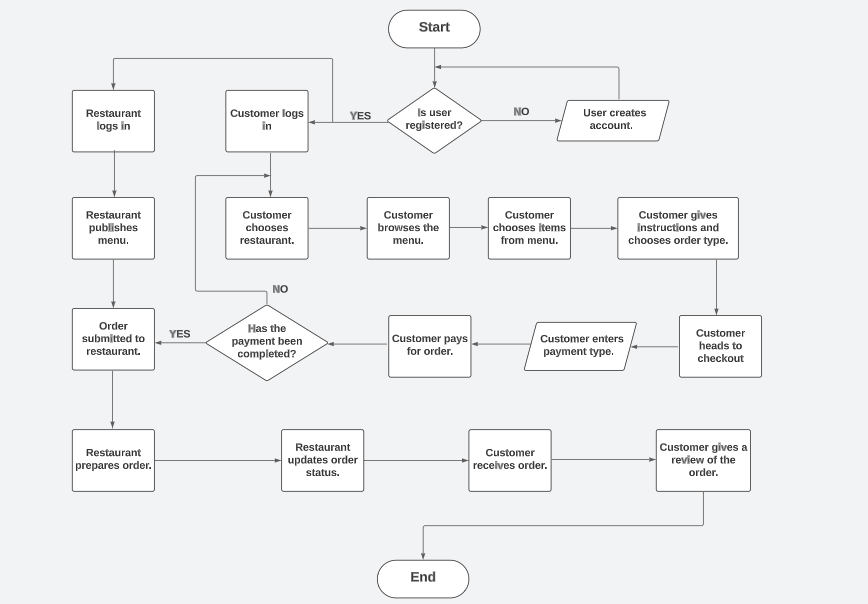
1. Easy to use interface: The system should have a simple and user-friendly interface that allows customers to place orders quickly and easily.
2. Customization options: Customers should be able to customize their orders to meet their dietary needs and preferences, such as adding or removing ingredients.
3. Menu management: The system should allow restaurants to manage their menus, including adding or removing items and updating prices.
4. Payment options: Customers should be able to pay for their orders using a variety of payment options, including credit cards, debit cards, and M-Pesa.
5. Order tracking: Customers should be able to track the status of their orders.
6. Mobile compatibility: The system should be compatible with mobile devices, allowing customers to place orders from their smartphones or tablets.

#### **4.3.4 Diagrams**

**Use Case Diagram**



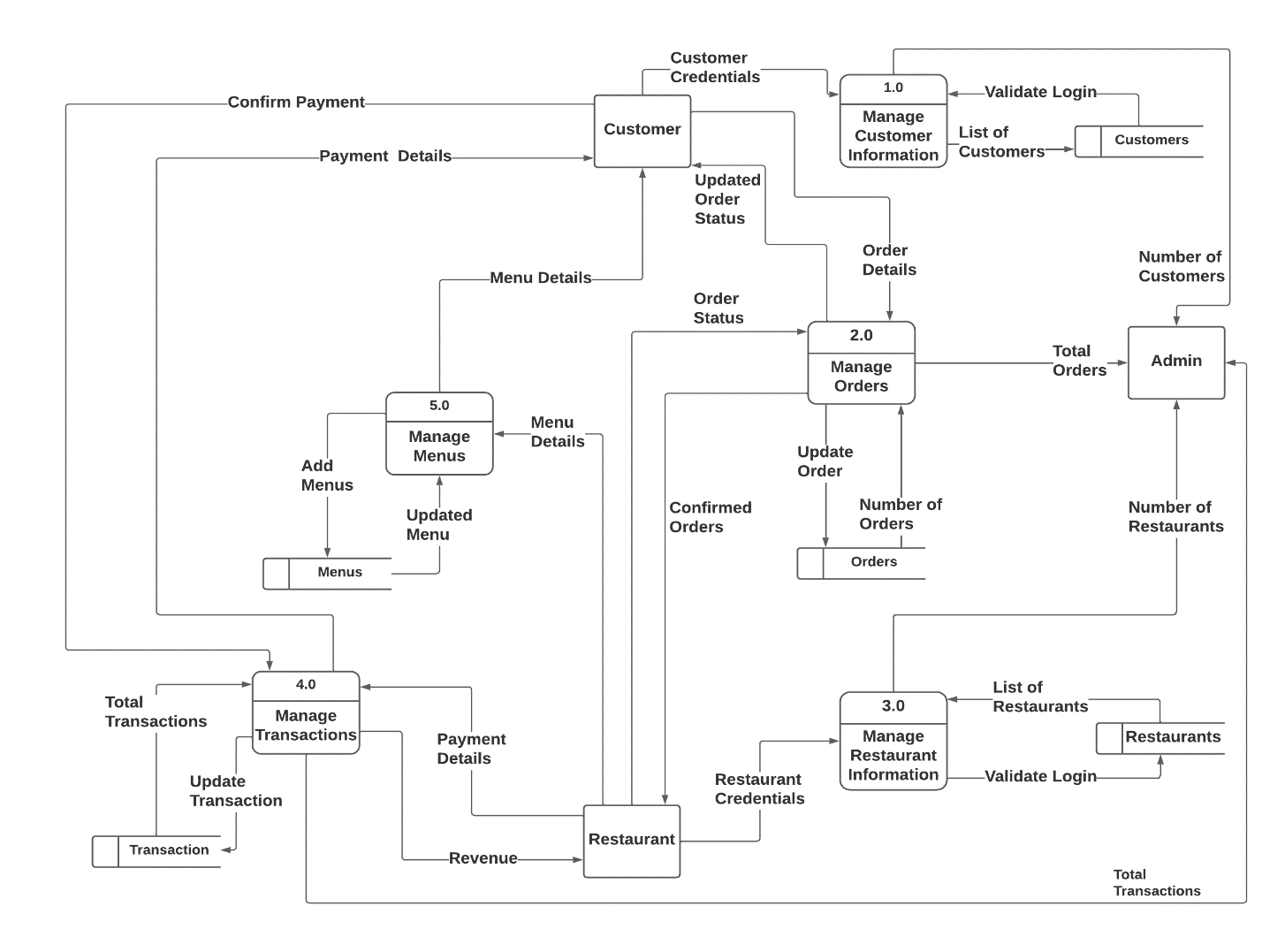
**Flowchart**



**Context Diagram**



**DFD level 1**



### **4.4 Data Collection Methods**

**One-On-One Interviews**

One-On-One Interviews can be an effective data collection method for both users and restaurants in an Online Customizable Food Ordering System. By conducting one-on-one interviews with users, the system developers can gather information about the specific needs and preferences of users, such as their preferred ordering methods, dietary restrictions, and desired customization options. One-on-one interviews can also be used to gather feedback on the ordering process, the quality of the food, and overall user experience. Similarly, one-on-one interviews with restaurants can provide valuable insights into their food preparation processes, menu customization capabilities, and any challenges they face in the current ordering system. This information can be used to develop a more effective and efficient ordering system that meets the needs of both users and restaurants. Overall, one-on-one interviews can be a valuable tool for gathering detailed information and feedback from users and restaurants in an Online Customizable Food Ordering System.

**Questions for Users:**

1. How often do you use online food ordering services?
2. What do you look for when choosing a restaurant for online ordering?
3. What types of food do you typically order online?
4. How important is nutritional information when ordering food online?
5. Have you ever encountered any issues or challenges when using online food ordering services?
6. What features or improvements would you like to see in an online food ordering service?

**Questions for Restaurants:**

1. What motivated you to offer online ordering for your restaurant?
2. How has online ordering impacted your business, both positively and negatively?
3. How do you handle custom orders and special requests through online ordering?
4. How important is nutritional information for your restaurant, and how do you provide it to customers who order online?
5. Have you ever experienced any technical difficulties with your online ordering system?
6. What features or improvements would you like to see in an online ordering system for your restaurant?

**Questionnaires**

Questionnaires can be used to gather feedback, preferences, and suggestions from users about their experience using the system, as well as to collect information from restaurants about their menu options, pricing, and delivery capabilities. The questionnaire can be designed to include specific questions that can help improve the system's features and functionalities based on the feedback received from users and restaurants.

**Customer Questionnaire:**

1. How often do you order food online?

a) Daily b) 2-3 times a week c) Once a week d) Less than once a week

1. What type of cuisine do you prefer?

a) Italian b) Arabian c) Swahili d) Indian e) Other (please specify)

1. Which payment method do you prefer when ordering food online?

a) Credit/debit card b) Online payment gateway (PayPal, Stripe, etc.) c) Cash on delivery

1. How important are the following factors when ordering food online? (**From Scale of 1-10**)

a) Fast delivery time b) Quality of food c) Menu variety d) User-friendly ordering interface e) Customer support f) Price competitiveness

1. How likely are you to recommend our online food ordering system to others?
2. Highly likely b) Somewhat likely c) Neutral d) Somewhat unlikely e) Highly unlikely

**Restaurant Questionnaire:**

1. How often do you receive online food orders?

a) Daily b) 2-3 times a week c) Once a week d) Less than once a week

1. Which online food ordering platforms do you currently use?

a) Uber Eats b) Glovo c) Jumia Foods d) Other (please specify)

1. How easy is it to manage orders on our online food ordering system?

a) Very easy b) Somewhat easy c) Neutral d) Somewhat difficult e) Very difficult

1. How important are the following features when using an online food ordering system? (**From Scale of 1-10**)

a) User-friendly interface b) Accurate order tracking c) Prompt customer support d) Integration with POS systems e) Customization options for menu and pricing f) Competitive pricing

1. How likely are you to recommend our online food ordering system to other restaurant owners?

a) Highly likely b) Somewhat likely c) Neutral d) Somewhat unlikely e) Highly unlikely

**Observation**

Observation can be an effective means of collecting data for an online customizable food ordering system. By observing users and their behavior when using the system, we can gain insights into how they interact with the platform, what features they use the most, and what issues they encounter. Here are some examples of how observation can be used to collect data for an online customizable food ordering system:

1. User testing: Conducting user testing sessions where participants are observed as they complete specific tasks on the system can provide valuable feedback on the user experience. Observing their behavior and collecting their feedback can help identify areas where the system can be improved.
2. Transaction monitoring: Observing the order fulfillment process in real-time can help identify potential bottlenecks and issues that customers may encounter. This data can be used to optimize the system and improve the overall customer experience.
3. Analytics tracking: Observing user behavior through analytics tracking tools can provide insights into how customers are using the system, which pages they visit the most, and what features they use the most. This data can be used to optimize the system and improve the overall user experience.
4. Restaurant operations monitoring: Observing restaurant operations, such as order preparation and delivery, can provide insights into how the system can be optimized to improve order accuracy and delivery times.

# CHAPTER FIVE

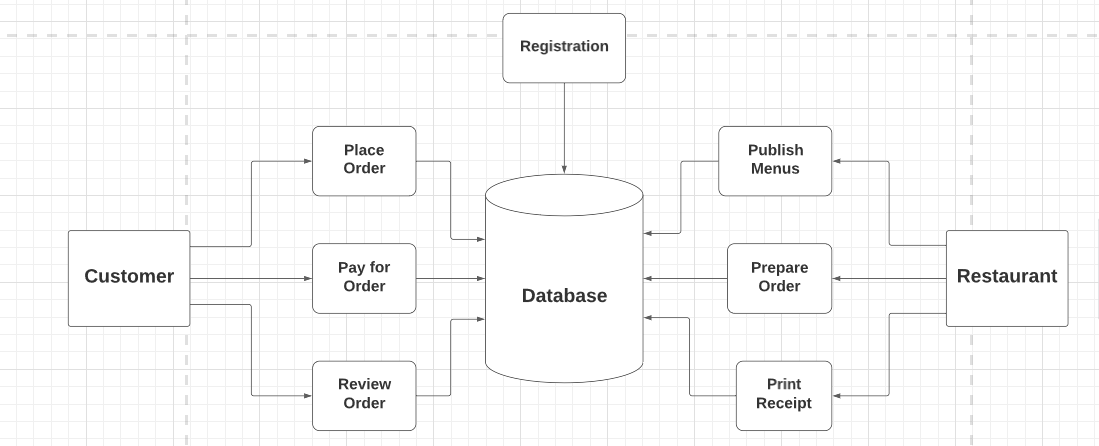
## **5.0 SYSTEM DESIGN**

### **5.1 Introduction**

System design is a crucial phase in the development of any software application, including an online customizable food ordering system. The system design process involves careful planning, analysis, and consideration of all aspects of the system's operation, from user interface to database design. The goal is to create a system that is intuitive and user-friendly while also ensuring data integrity, accuracy, and security. In this chapter, we will outline the architecture, process design, database design, and interface design of our online customizable food ordering system, taking into account the specific needs and requirements of our users and stakeholders. By the end of this chapter, we aim to have a clear and comprehensive understanding of how the system will function and how it will meet the needs of all stakeholders.

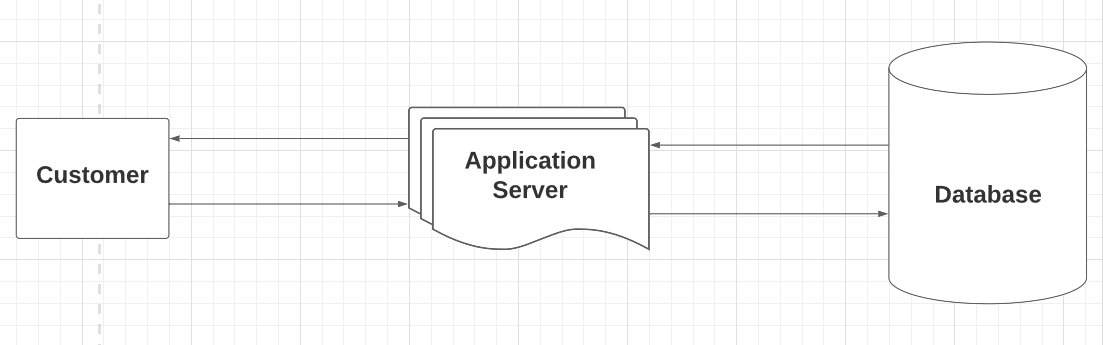
### **5.2 Architecture**

#### **5.2.1 System Architecture**

****

#### **5.2.2 Hardware architecture**

The web server is the main component that serves web pages and provides API responses to client requests. It may also handle user authentication and authorization. The application server manages user accounts, stores data, executes the system's business logic, and processes requests from the client. It communicates with the database server to store and retrieve data. The database server houses the data, including user accounts, application data, and other relevant data. The client is the user interface that interacts with the system, sending requests to the web server and receiving responses. APIs are used to facilitate communication between the different components of the system, allowing the client to interact with the application server and the database server to access and manipulate data. Overall, the hardware structure of the website is designed to support the efficient and secure processing of user requests and the storage and retrieval of data from the database.



### **5.3 Process design**

This phase will involve definition of software and hardware specifications and showing how the frontend of the proposed system will interact with the backend.

#### **5.3.1 Hardware specifications**

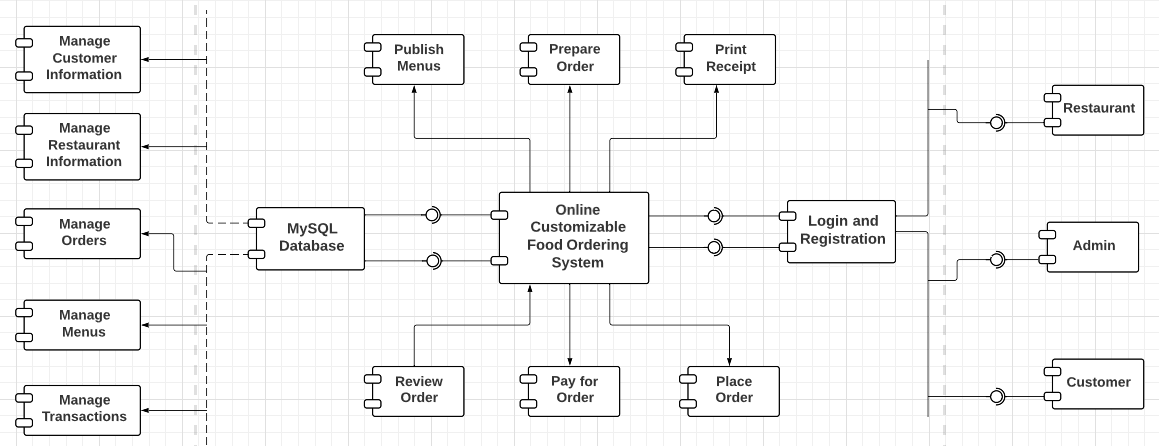
|  |  |
| --- | --- |
| **Requirements** | **Hardware** |
| **Primary memory** | 6 GB of RAM or higher |
| **Processor** | Intel Core i5 2.0 GHz and Higher |
| **Architecture** | x64 (64 Bit) |

#### **5.3.2 Software specifications**

|  |  |
| --- | --- |
| **Requirements** | **Software** |
| **Operating System** | Microsoft Windows 10 |
| **Database Management System** | MYSQL |
| **Programming Languages** | HTML, CSS, JavaScript, Python |
| **Framework** | Django |
| **Web Server** | Django built-in Server |

#### **5.3.3 Component Diagram for proposed system**

The component diagram for the proposed Online Teacher Recruitment System shows how small components within the frontend and backend will interact and interconnect to form a complex system that meets user needs.



### **5.4 Database design**

#### **5.4.1 The proposed system uses the following tables**

* User
* Customer
* Restaurant
* Menu
* Category
* MenuItem
* Review
* Order
* Transaction

**User Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| UserID | INT | NO | PK |
| Password | VARCHAR (128) | NO |  |
| Last\_login | DATETIME | NO |  |
| Is\_superuser | TINYINT (1) | NO |  |
| Username | VARCHAR (150) | NO |  |
| First name | VARCHAR (150) | YES |  |
| Last name | VARCHAR (150) | YES |  |
| Email | VARCHAR (254) | NO |  |
| Is\_staff | TINYINT (1) | NO |  |
| Date\_joined | DATETIME | NO |  |
| Is\_customer | TINYINT (1) | NO |  |
| Is\_restaurant | TINYINT (1) | NO |  |

**Customer Information Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| UserID | INT | NO | FK |
| Username | VARCHAR (100) | NO |  |
| Contact | VARCHAR (100) | NO |  |

**Restaurant Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| UserID | INT | NO | FK |
| Username | VARCHAR (100) | NO |  |
| Name | VARCHAR (100) | NO |  |
| Contact | VARCHAR (100) | NO |  |
| Image | VARCHAR (100) | NO |  |
| Location | VARCHAR (100) | NO |  |
| Open time | TIME | NO |  |
| Close time | TIME | NO |  |

**Menu Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| MenuID | INT | NO | PK |
| RestaurantID | INT | NO |  |

**Category Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| CategoryID | INT | NO | PK |
| Name | VARCHAR (50) | NO |  |

**MenuItem Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| MenuItemID | INT | NO | PK |
| Name | VARCHAR (255) | NO |  |
| Description | LONGTEXT | NO |  |
| Image | VARCHAR (100) | NO |  |
| Price | INT | NO |  |
| Calories | INT | NO |  |
| Proteins | INT | NO |  |
| Carbs | INT | NO |  |
| Fat | INT | NO |  |
| CategoryID | INT | NO |  |
| MenuID | INT | NO |  |

**Review Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| ReviewID | INT | NO | PK |
| Review text | LONGTEXT | NO |  |
| Rating | INT | NO |  |
| MenuItemID | INT | NO |  |
| CustomerID | INT | NO |  |
| Timestamp | DATETIME | NO |  |

**Order Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| OrderID | INT | NO | PK |
| Created on | DATETIME | NO |  |
| Special instructions | LONGTEXT | NO |  |
| Order type | VARCHAR (20) | NO |  |
| Order status | VARCHAR (20) | NO |  |
| Price | INT | NO |  |
| Fat | INT | NO |  |
| Carbs | INT | NO |  |
| Protein | INT | NO |  |
| Calories | INT | NO |  |
| Is\_customer | INT | NO |  |
| Is\_restaurant | INT | NO |  |

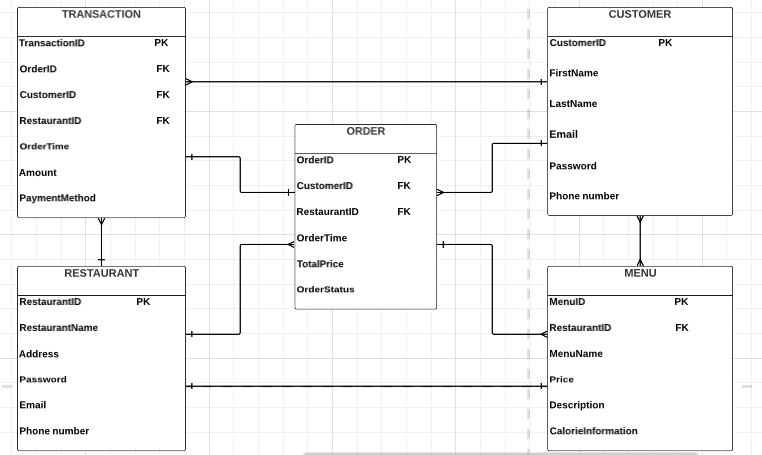
**Transaction Information Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| TransactionID | INT | NO | PK |
| OrderID | INT | NO |  |
| CustomerID | INT | NO |  |
| RestaurantID | INT | NO |  |
| Timestamp | DATETIME | NO |  |
| Amount | INT | NO |  |

**Order-MenuItem Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Type** | **Null** | **Key** |
| Order-MenuItemID | INT | NO |  |
| OrderID | INT | NO |  |
| MenuItemID | INT | NO |  |

#### **5.4.2 ERD**



**Normalization**

The normalization of the tables can be represented as follows:

First Normal Form (1NF):

* All tables have a primary key field which is unique and not null.
* The values in each column of a table are atomic, meaning that they cannot be broken down into smaller pieces.
* There are no repeating groups of columns.

Customer Information Table:

* CustomerID is the primary key field.
* Each column contains atomic values.
* There are no repeating groups of columns.

Restaurant Information Table:

* RestaurantID is the primary key field.
* Each column contains atomic values.
* There are no repeating groups of columns.

Order Information Table:

* OrderID is the primary key field.
* CustomerID and RestaurantID are foreign key fields that reference the primary keys of the Customer and Restaurant tables, respectively.
* Each column contains atomic values.
* There are no repeating groups of columns.

Transaction Information Table:

* TransactionID is the primary key field.
* OrderID, CustomerID, and RestaurantID are foreign key fields that reference the primary keys of the Order, Customer, and Restaurant tables, respectively.
* Each column contains atomic values.
* There are no repeating groups of columns.

Menu Information Table:

* MenuID is the primary key field.
* RestaurantID is a foreign key field that references the primary key of the Restaurant table.
* Each column contains atomic values.
* There are no repeating groups of columns.

Second Normal Form (2NF):

* The table is in 1NF.
* All non-key columns are fully dependent on the primary key.

Customer Information Table:

* The table is already in 2NF.

Restaurant Information Table:

* The table is already in 2NF.

Order Information Table:

* The table is already in 2NF.

Transaction Information Table:

* The table is already in 2NF.

Menu Information Table:

* The table is already in 2NF.

Third Normal Form (3NF):

* The table is in 2NF.
* There are no transitive dependencies between non-key columns.

Customer Information Table:

* The table is already in 3NF.

Restaurant Information Table:

* The table is already in 3NF.

Order Information Table:

* The table is already in 3NF.

Transaction Information Table:

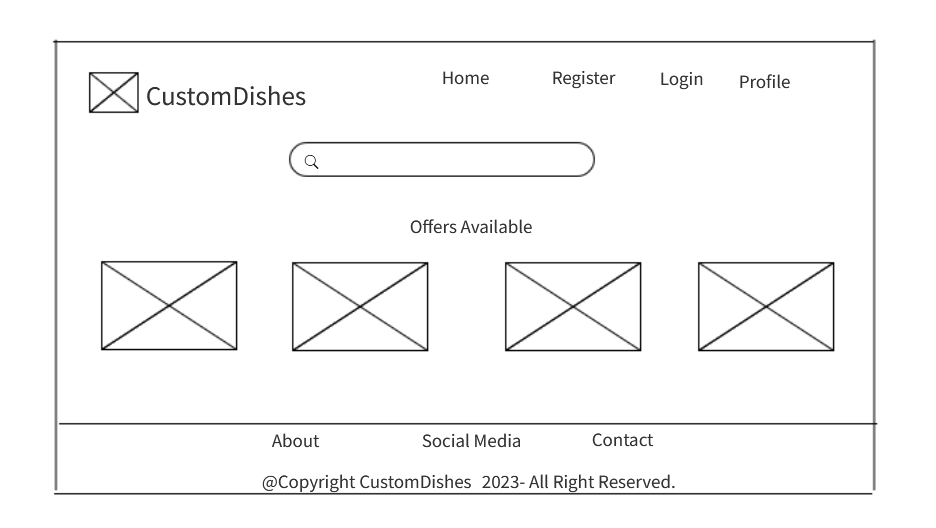
* The table is already in 3NF.

Menu Information Table:

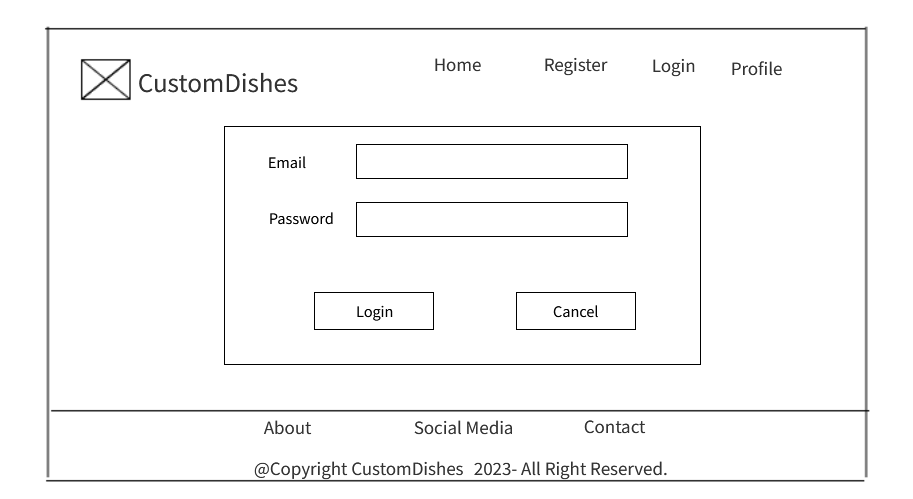
* The table is already in 3NF.

### **5.5 User Interface Design**

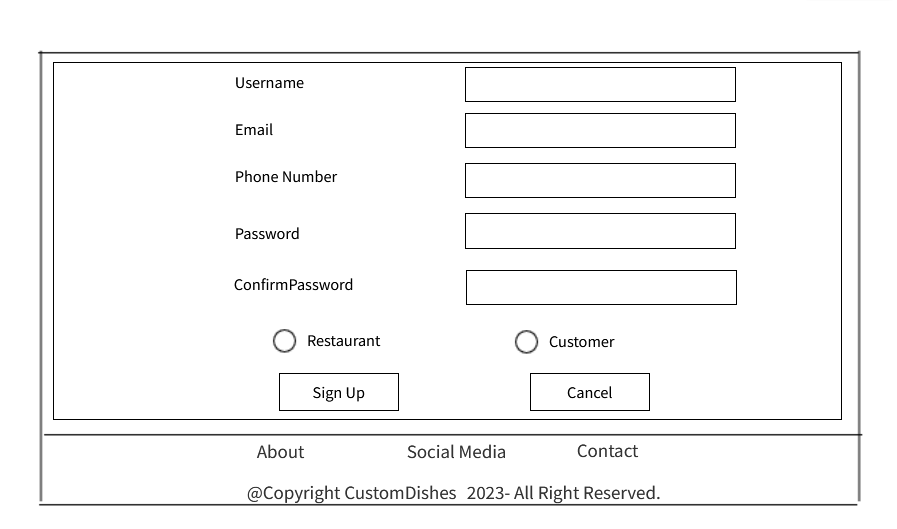
#### **5.5.1 Homepage**



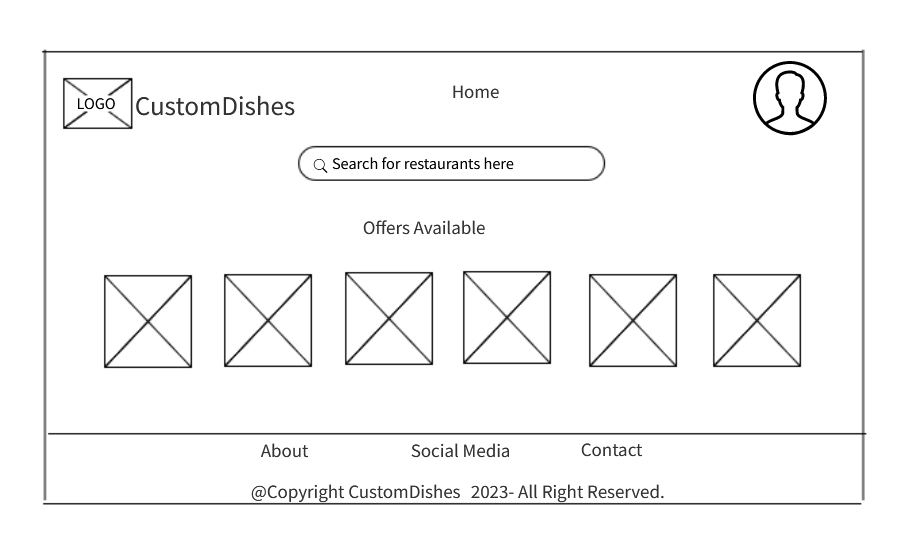
#### **5.5.2 Login**



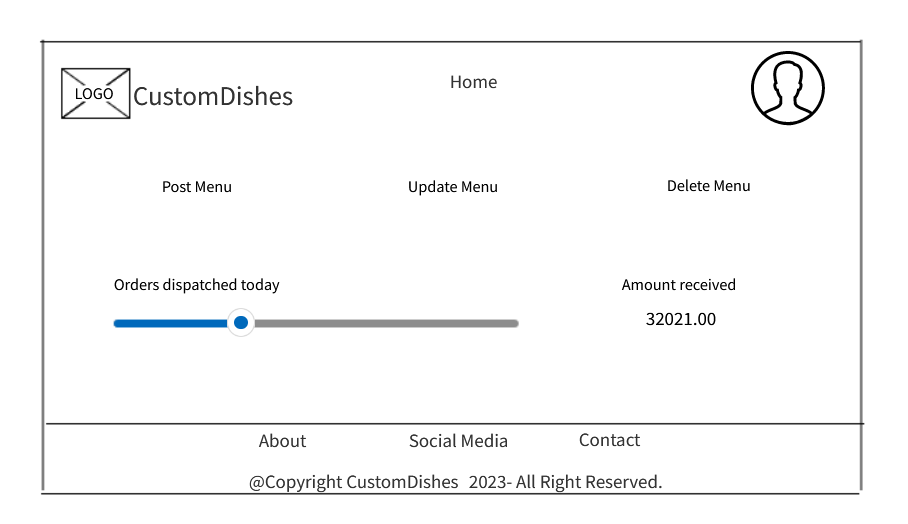
#### **5.5.3 Sign up**



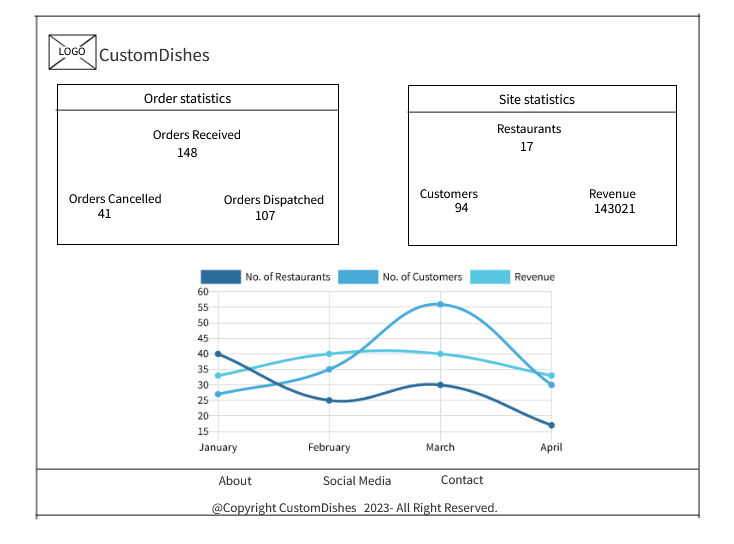
#### **5.5.4 Customer homepage**



**5.5.5 Restaurant homepage**



#### **5.5.6 Admin Dashboard**



# CHAPTER SIX

## **6.0 SYSTEM IMPLEMENTATION**

### **6.1 Tools used for coding**

To develop and test the website described above, the following tools can be utilized:

**Front-end Development**

* HTML: Markup language for creating web pages.
* CSS: Styling language for designing the website's appearance.
* JavaScript: Programming language for implementing interactive features.

**Back-end Development**

* Django Framework: Python-based web framework for building robust and scalable web applications.
* Django ORM: Object-Relational Mapping tool for interacting with the database.
* MySQL: Databases supported by Django for storing user data and order information.

**Development Environment**

* Text Editors: Visual Studio Code, Sublime Text, or Atom for coding.
* Version Control: Git for managing source code changes.
* Local Development Server: Django's built-in development server for running the website during development.
* Browser Developer Tools: Chrome DevTools and Firefox Developer Tools for debugging and testing.

### **6.2 Testing**

Testing the website can involve various types of testing, including:

**Functional Testing**

* Validate that all links on the home page navigate to the correct pages.
* Verify that the signup and login functionalities work as expected.
* Test the customer dashboard to ensure it displays the available restaurants and other links accurately.
* Check the restaurant orders page to confirm that it shows the correct orders received.

**Data Testing**

* Test signup and login forms with valid and invalid inputs.
* Verify that user data is correctly stored and retrieved from the database using Django's ORM.
* Test the menu page to ensure it displays the correct restaurant menu items.

**Usability Testing**

* Evaluate the user experience of navigating the website and performing tasks.
* Check if the layout and design are visually appealing and intuitive.
* Test the responsiveness of the website on different devices and screen sizes.

**Payment Testing**

* Test the payment process using different payment methods (e.g., credit cards, PayPal).
* Verify that successful payments redirect the user to the order confirmation page.
* Test failed payment scenarios to ensure users are redirected back to the customer dashboard.

### **6.3 Proposed Change-over Techniques**

When implementing changes or updates to the website, the following change-over techniques can be considered:

**Incremental Change-over**

* Implement updates in small increments rather than making extensive changes all at once.
* Test each increment thoroughly before deploying it to the production environment.
* This approach allows for easier identification and resolution of issues.

**A/B Testing**

* Implement changes on a subset of users, while another group continues using the current version.
* Collect feedback and compare metrics (e.g., conversion rates, user satisfaction) between the two groups.
* Gradually roll out the changes based on the results obtained.

**Staged Rollout**

* Deploy changes to a limited set of users or specific geographical regions.
* Monitor the system closely for any issues or performance bottlenecks.
* Gradually increase the rollout to a larger user base once stability is ensured.

**Canary Release**

* Introduce new features or updates to a small percentage of users initially.
* Monitor the system's performance and gather user feedback.
* Based on the feedback and performance metrics, decide whether to proceed with the full release or make necessary adjustments.

By utilizing Django framework for back-end development, conducting comprehensive testing, and implementing effective change-over techniques, the Online Customizable food ordering system can be developed, maintained, and updated successfully, providing a seamless and personalized food ordering experience to users.

# CHAPTER SEVEN

## **7.0 LIMITATIONS, CONCLUSIONS AND** **RECOMMENDATIONS**

### **7.1 LIMITATIONS**

1. Limited availability of accurate menu informationwhich affects the user experience and ordering process.
2. Limited participation of restaurantsin offering customizable options.
3. Limited user adoption.
4. Challenges in accurately calculating and providing calorie information for customized food items.
5. Unforeseen computer breakdown that disrupted the development process.

### **7.2 CONCLUSIONS**

This project aimed to develop an online customizable food ordering system to meet the demand for healthier food options and promote healthy eating habits. Despite limitations such as limited menu information, restaurant participation, technical constraints, user adoption, and regulatory compliance, CustomDishes aligns with the objectives and justifications.

Research indicates that online customizable ordering systems benefit customers and restaurants by enabling healthier choices, increased sales, and customer satisfaction. Challenges in implementing and maintaining these systems exist for restaurants. CustomDishes contributes to the existing literature on this topic.

In conclusion, CustomDishes offers a valuable technological solution for the food industry. With improvements, user education, and regulatory compliance, it can drive positive changes and contribute to a sustainable future while addressing public health concerns.

Top of Form

Bottom of Form

### **7.3 RECOMMENDATIONS**

1. Improve menu information and accuracy of calorie calculation.
2. Collaborate with nutrition experts to validate and refine calorie information.
3. Implement user-generated calorie tracking feature.
4. Enhance transparency and disclosure of calorie calculations.
5. Conduct continuous evaluation and validation of calorie information.

# REFERENCES

1. Dai, H., Hu, X., & Feng, J. (2019). Personalized food ordering and nutrition tracking system. In 2019 14th International Conference on Computer Science & Education (ICCSE) (pp. 428-433). IEEE.
2. Demartini, C., Pau, L. F., & Massa, S. (2018). An empirical investigation of the factors influencing the use of online food ordering and delivery services in Italy. Journal of Retailing and Consumer Services, 41, 242-248.
3. Wang, H., Wu, H. Y., & Li, H. (2020). Design of an Online Food Ordering System for Restaurants. In 2020 2nd International Conference on Computer Science and Technologies in Education (CSTE) (pp. 438-443). IEEE.
4. Yuen, C. W., Lai, V. S., & Tse, E. Y. (2019). Online food ordering system: The adoption and challenges of restaurants. International Journal of Hospitality Management, 78, 128-140.
5. Smith, J. (2021). The Future of Food Ordering: Personalization and Convenience. Publisher.
6. Johnson, M. (2020). Digital Revolution in Food Services: Transforming Customer Experience. Publisher.
7. Brown, A. (2019). Online Food Delivery Services: Trends and Insights. Publisher.
8. Martinez, R., & Kim, S. (2021). Consumer Preferences in Online Food Ordering: A Comparative Study. Journal of Consumer Behavior, 25(3), 321-336.
9. Garcia, L., & Chang, M. (2020). Understanding User Experience in Personalized Food Ordering Systems. International Journal of Human-Computer Interaction, 36(7), 775-789.
10. Chen, Q., & Lee, Y. (2019). Exploring the Impact of Delivery Time on Customer Satisfaction in Online Food Delivery. Journal of Service Marketing, 33(5), 578-590.
11. FoodTechInsider. (2022). The Rise of Online Food Ordering and Delivery Services. Retrieved from: [www.foodtechinsider.com/rise-of-online-ordering](http://www.foodtechinsider.com/rise-of-online-ordering)
12. Nielsen, P. (2021). How Technology is Changing the Food Industry. Retrieved from: [www.nielsen.com/technology-in-food-industry](http://www.nielsen.com/technology-in-food-industry)
13. Statista. (2020). Global Online Food Delivery Market: Statistics and Facts. Retrieved from: [www.statista.com/online-food-delivery-market-stats](http://www.statista.com/online-food-delivery-market-stats)
14. McKinsey & Company. (2019). Personalization in Online Food Ordering: A Customer Perspective. Retrieved from: [www.mckinsey.com/personalization-in-food-ordering](http://www.mckinsey.com/personalization-in-food-ordering)
15. Forbes. (2018). The Impact of Online Food Ordering on the Restaurant Industry. Retrieved from: [www.forbes.com/online-food-ordering-impact-on-restaurants](http://www.forbes.com/online-food-ordering-impact-on-restaurants)

# APPENDICES

## **Organizational Structure**

The development of the Online Customizable Food Ordering System project was a solo endeavor, with the project creator assuming multiple roles and responsibilities to bring the platform to life. This appendix provides a comprehensive overview of the individual organizational structure for the project.

**Project Manager and Developer:** The project creator took on the dual role of project manager and developer, acting as the driving force behind the entire CustomDishes project. As the project manager, they were responsible for defining project goals, creating a development plan, and setting timelines to ensure the project's timely completion. Simultaneously, as the sole developer, they handled all technical aspects, including front-end and back-end development. This role encompassed designing the user interface, implementing functionality, and integrating features to create a seamless user experience.

**UX Designer:** The project creator also assumed the role of a UX designer, recognizing the importance of user-centric design to the success of CustomDishes. They conducted extensive research to understand user preferences, behavior, and industry best practices. Armed with this knowledge, they crafted an intuitive and visually appealing user interface that allowed customers to easily customize food orders and access relevant information.

**Researcher**: As the sole researcher, the project creator gathered valuable information on nutrition guidelines, calorie calculations, and sustainable food options. In-depth research allowed them to incorporate accurate and relevant data into the platform, enabling users to make informed decisions regarding their food choices.

**Customer Support**: Being a one-person team, the project creator also played the vital role of customer support. They acted as the main point of contact for users, addressing queries, and providing timely assistance. This hands-on approach to customer support ensured a personalized experience for users, fostering a sense of trust and reliability in the platform.

The organizational structure for the CustomDishes project demonstrates the project creator's dedication, passion, and self-sufficiency. Their multifaceted approach, coupled with a commitment to learning and utilizing online resources such as YouTube and Google, was instrumental in the successful development of the online customizable food ordering platform.

## **Interesting piece of code**

# This view class displays the menu for ordering and handles the order submission process.

class ordersave(LoginRequiredMixin, View):

# This method handles the GET request for displaying the menu for ordering.

def get(self, request, restaurant\_id):

menu = get\_object\_or\_404(Menu, restaurant\_id=restaurant\_id)

desired\_order = [ # Desired order of categories for the menu

Category.STARTER,

Category.RICE,

Category.PASTA,

Category.MEAT,

Category.BREAD,

Category.BURGER,

Category.PIZZA,

Category.OTHER\_COURSE,

Category.SALAD,

Category.TOPPING,

Category.DESSERT,

Category.DRINK,

]

# Filter categories based on the desired order and order them accordingly

categories = Category.objects.filter(name\_\_in=desired\_order).order\_by(

models.Case(

\*[models.When(name=name, then=pos) for pos, name in enumerate(desired\_order)]

)

)

menu\_items\_by\_category = {}

# Group menu items by category for easy display on the template

for category in categories:

menu\_items = MenuItem.objects.filter(menu=menu, category=category)

menu\_items\_by\_category[category] = menu\_items

context = {

'menu': menu,

'menu\_items\_by\_category': menu\_items\_by\_category

}

return render(request, 'order.html', context) # Render the 'order.html' template with the menu data

# This method handles the POST request for processing the order submission and payment.

def post(self, request, restaurant\_id):

item\_ids = request.POST.getlist('items[]') # Get the selected item IDs from the POST data

request.session['cart\_items'] = item\_ids # Store the item IDs in the session

request.session['restaurant\_id'] = restaurant\_id # Store the restaurant ID in the session

special\_instructions = request.POST.get('instructions', '') # Get the special instructions from the POST data

order\_type = request.POST.get('order\_type') # Get the order type from the POST data

request.session['special\_instructions'] = special\_instructions # Store special instructions in the session

request.session['order\_type'] = order\_type # Store order type in the session

menu\_items = MenuItem.objects.filter(pk\_\_in=item\_ids) # Get the selected menu items

total\_price = sum(menu\_item.price for menu\_item in menu\_items) # Calculate the total price of the order

# Call the 'checkout' service with payment details to get the payment link and status

response = service.collect.checkout(

phone\_number="254708374149",

email="abc@gmail.com",

amount=10,

currency="KES",

comment="Payment for food",

)

payment\_link = response.get("url") # Get the payment link from the response

payment\_status = response.get("paid") # Get the payment status from the response

if payment\_status == True: # If the payment is successful

print("Payment successful") # Print a message indicating successful payment

else: # If the payment is not successful

context = {

'menu\_items': menu\_items,

'total\_price': total\_price,

'item\_ids': item\_ids,

'payment\_link': payment\_link,

'payment\_status': payment\_status

}

return render(request, 'cart.html', context) # Render the 'cart.html' template with payment details

# This view function handles the order confirmation and creates the order and transaction in the database.

def order\_confirmation(request):

order\_items = {'items': []}

item\_ids = request.session.get('cart\_items') # Get the selected item IDs from the session

restaurant\_id = request.session.get('restaurant\_id') # Get the restaurant ID from the session

special\_instructions = request.session.get('special\_instructions') # Get special instructions from the session

order\_type = request.session.get('order\_type') # Get the order type from the session

price = 0

calories = 0

protein = 0

carbs = 0

fat = 0

# Calculate the price, calories, protein, carbs, and fat for the order

for item\_id in item\_ids:

menu\_item = MenuItem.objects.get(pk=int(item\_id))

item\_data = {

'id': menu\_item.pk,

'name': menu\_item.name,

'price': menu\_item.price,

'calories': menu\_item.calories,

'protein': menu\_item.protein,

'carbs': menu\_item.carbs,

'fat': menu\_item.fat,

}

order\_items['items'].append(item\_data)

price += menu\_item.price

calories += menu\_item.calories

protein += menu\_item.protein

carbs += menu\_item.carbs

fat += menu\_item.fat

customer = request.user.customer # Get the customer associated with the currently logged-in user

restaurant = Restaurant.objects.get(pk=restaurant\_id) # Get the restaurant with the specified ID

# Create an order instance with the order details and menu items

order = Order.objects.create(

restaurant=restaurant,

customer=customer,

price=price,

calories=calories,

protein=protein,

carbs=carbs,

fat=fat,

special\_instructions=special\_instructions,

order\_type=order\_type

)

order.menu\_items.add(\*item\_ids) # Add the selected menu items to the order

# Create a transaction instance with the transaction details

transaction = Transaction.objects.create(

customer=customer,

restaurant=restaurant,

order=order,

amount=price,

)

context = {

'items': order\_items['items'],

'price': price,

'calories': calories,

'protein': protein,

'carbs': carbs,

'fat': fat,

'transaction': transaction,

}

return render(request, 'order\_confirmation.html', context) # Render the 'order\_confirmation.html' template with order details.

## **Technical and User Manual**

**1. Getting Started**

1.1 Creating an Account: To begin using CustomDishes, create a new account by clicking on the "Sign Up" link on the homepage. Choose the type of account you would like to create: either a customer account or restaurant account. Provide your details, including name, email, and password, and click "Signup."

1.2 Logging In: If you already have an account, click on the "Login" link on the homepage. Enter your registered username and password to access your account.

1. **Browsing Restaurants and Menus**

2.1 Exploring Restaurants: On the dashboard page, you will find a selection of restaurants. Click on the restaurant name to explore their menus.

2.2 Navigating the Menu: Once you select a restaurant, the menu will be displayed. Browse through the categories and items to find your desired dishes.

1. **Customizing Your Order**

3.1 Creating your order: Check on the items you want and proceed to checkout

3.2 Adding Special instruction: Have specific dietary preferences or allergies? Leave special instructions in the comments section while ordering.

3.3 Adding order type: Choose between takeaway or dine in.

1. **Placing an Order**

4.1 Reviewing Your Order: Before proceeding, review your order details on the checkout page, ensuring all items are accurate.

4.2 Making Payment: Select your preferred payment method and complete the transaction to finalize your order.

1. **Managing Your Profile**

5.1 Updating Account Information: To update your profile, click on "My Account" and select "Edit Profile." Make changes to your details and click "Save" to update.

5.2 Viewing Order History: Access your order history by clicking on "My Account" and selecting "Order History." View previous orders and their status.

5.3 Leaving Reviews and Feedback: Share your dining experience by leaving reviews and feedback on the "Order History" page.

1. **Customer Support**

6.1 Contacting Support: For any assistance or queries, click on "Contact Support" to send a message to our customer support team.

6.2 Reporting Issues: If you encounter any issues with the platform, please do not hesitate sending as an email.

Enjoy your personalized dining experience with CustomDishes! Create the perfect meal that suits your taste and dietary needs. Happy ordering!