**TABLE OF CONTENTS**

[CHAPTER 1: INTRODUCTION 3](#_Toc210581177)

[CHAPTER 2: SYSTEM ANALYSIS AND DESIGN 3](#_Toc210581178)

[2.1 Abc 3](#_Toc210581179)

[2.1.1 Functions 3](#_Toc210581180)

[2.1.2 User Interfaces 12](#_Toc210581181)

[2.1.3 Data Design 13](#_Toc210581182)

[2.1.4 Rest APIs Design 13](#_Toc210581183)

[2.2 Architecture Design 14](#_Toc210581184)

[CHAPTER 3: TEST PLAN 14](#_Toc210581185)

[3.1 Introduction 14](#_Toc210581186)

[3.1.1 Purpose 14](#_Toc210581187)

[3.1.2 Definitions, Acronyms, and Abbreviations 14](#_Toc210581188)

[3.1.3 References 14](#_Toc210581189)

[3.1.4 Background Information 14](#_Toc210581190)

[3.1.5 Scope of Testing 14](#_Toc210581191)

[3.1.6 Constraints 15](#_Toc210581192)

[3.1.7 Risk List 15](#_Toc210581193)

[3.2 Requirements for Test 15](#_Toc210581194)

[3.2.1 Test items 15](#_Toc210581195)

[3.2.2 Acceptance Test Criteria 15](#_Toc210581196)

[3.3 Test strategy 15](#_Toc210581197)

[3.3.1 Test Types 15](#_Toc210581198)

[3.3.2 Test Stages 15](#_Toc210581199)

[3.4 Resources 15](#_Toc210581200)

[3.4.1 Human Resource 15](#_Toc210581201)

[3.4.2 Test Management 15](#_Toc210581202)

[3.5 Test Environment 15](#_Toc210581203)

[3.5.1 Hardware 15](#_Toc210581204)

[3.5.2 Software 15](#_Toc210581205)

[3.5.3 Infrastructure 15](#_Toc210581206)

[3.6 Test Milestones 15](#_Toc210581207)

[3.7 Test Deliverables 15](#_Toc210581208)

[CHAPTER 4: TEST DESIGN 15](#_Toc210581209)

[CHAPTER 5: TEST CASES – TEST REPORT 15](#_Toc210581210)

[CHAPTER 6: CONCLUSION 15](#_Toc210581211)

# INTRODUCTION

**Potato** is anapplication designed to serve both web and mobile users. The backend is developed with **Spring Boot** and provides **RESTful APIs**, a robust and maintainable foundation for client applications. For the frontend, **ReactJS** is used for web interfaces, while **React** **Native** powers the mobile applications, providing an interactive user experience across platforms.

This application shows how to design and implement a scalable monolithic system using well-structured layers, separating concerns between the user interface, business logic, and data access, while enabling multiple client platforms to interact seamlessly via RESTful endpoints.

It is mainly designed by the **Spring** **ecosystem**, leveraging popular libraries that have been widely adopted by the community for years, combined with modern frontend technologies to provide a responsive, maintainable solution for real-world food delivery scenarios.

# SYSTEM ANALYSIS AND DESIGN

## Abc

### Functions

#### Business context

**Potato** covers the basic business scenarios for Merchant, Menu, Cart, Order, Payment, Rating and Access Control.

**With Merchant**, Customers can explore a list of merchants with options to search by name, filter and sort by type of food, rating, or location. They can also view detailed information of a selected merchant, including its name, introduction, type of food, rating, opening hours, estimated delivery time, and location. Merchant Admin can manage their own merchant details, while Admin has the ability to view all merchants, approve new ones, or deactivate existing ones.

With **Menu**, Customers can browse all the dishes offered by a specific merchant. They can also view detailed information of each available dish such as image, name, description and price. Merchant Admin has full control to managetheirmerchant’s menu.

With **Cart**, Customers can add any dish from themenu or from thedishdetailpage by choosing options provided by the Merchant, if needed. After adding, the dishes will appear in the cart along with their corresponding subtotal cost and a summary panel showing cart total cost. When Customers update the quantity of dishes from the cart, the summary panel and cart are updated accordingly. After all, Customers can proceed to checkout to place an order by clicking Checkout button from the cart view.

**With Order**, Customers can review the order details, including the list of dishes, quantity, price, delivery fee and total amount, before confirming. Customers can also track their order status after placing the order. Merchant Admin can confirm/cancel orders, update order status, and view **order statistics** within a defined period (today, last 7 days, last 30 days), including the number of orders, revenue, and average order value. System Admin can monitor all orders for administration and summary purposes.

With **Payment**, Customers can pay via supported methods (online payment or cash on delivery) and the order status will be updated accordingly.

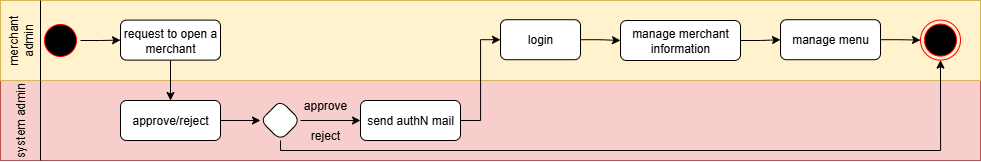
With **Rating**, Customers can rate for their orders after receiving them. Their rating will be used for calculating merchant’s average rating.

With **Access Control**, Customers, Merchant Admin, and System Admin can log in and log out the system. Besides, Customers and Merchant Admin can also sign up for a new account. After sign up as a Customer role, system will allow to login or to access protected resource. After sign up as a Merchant Admin role, the System Admin will decide to approve or reject the registration, if it is successful the Merchant Admin will be allowed to login. After login, Customers are directed to the home page, Merchant Admin to the merchant management page, and System Admin to the administration dashboard. When log out, Customers are directed to the home page, Merchant Admin and System Admin are directed to the login page.

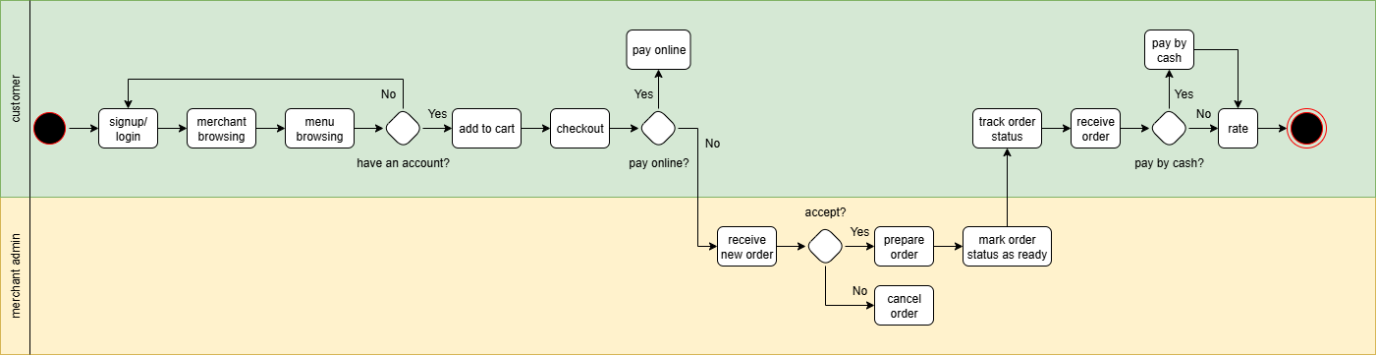
Some **one-off setup tasks** when the system starts include creating Admin users, sample Customers, Merchant Admin, Delivery Staff accounts, and populating sample data for merchants, dishes, orders, and deliveries.

#### Business Processes

* Merchant Onboarding Process



* Order Process



#### User Stories

1. **Merchant**

As a **Customer**, I want to **see a list of merchants with search, filtering and sorting** so that I can easily find merchants that match my preferences.

* Whenever searching by name in the search bar, then the list of merchants will narrow down with appropriate merchants.
* Whenever filtering with any name, type of food, rating, or location of the merchant, then the list of merchants needs to narrow down with appropriate merchants.
* Whenever sorting with descending on rating, or location, then the list of merchants needs to follow with this sorting.
* Whenever both filtering and sorting are in action, then the list of merchants will be effective by both of descriptions above.

As a **Customer**, I want to **view detailed information of a selected merchant** such as name, introduction, cuisine types, rating, opening hours, estimated delivery time, and location of this merchant.

As a **Customer**, I want to **browse all the dishes of a merchant** so that I can choose which one to order.

* The list of dishes will be displayed in sections based on category.
* Each dish will show at least its image, name, price and availability.

As a **Merchant Admin**, I want **to register a business** so that I can start selling products on the platform.

As a **Merchant Admin**, I want to **update my merchant’s information** such as name, introduction, cuisine types, opening hours and location.

As a **Merchant Admin**, I want to **manage my merchant’s menu** including manage categories, dishes, and options.

As a **System Admin**, I want to **approve business registration** so that only legitimate merchants are allowed to operate in the system.

As a **System Admin**, I want to **deactivate a merchant** so that inactive or violating merchants cannot continue serving customers.

1. **Cart**

As a **Customer**, I want to **view detailed information of a dish** and **add to my cart**.

* Whenever a Customer wants to add a dish to the cart, the system will display the details of that dish, along with a list of options assigned to it.
* Whenever a Customer adds a dish from the **menu interface**, then the system will add 1 of it into cart by **default.**
* Whenever a Customer adds a dish from the **dish detail interface**, then the system will allow the Customer to adjust the desired quantity.
* Whenever a dish is added to the cart (in both case), the menu interface will update to display the corresponding quantity of the selected dish. At the same time, the cart interface will show the selected dish with their quantity, and the system automatically calculates the subtotal and total amounts

As a **Customer**, I want to **update the quantity of a dish in the cart**.

* Whenever updating quantity of a dish happens, the cart will update accordingly and then the system will recalculate **subtotal** and total.

As a **Customer**, I want to **remove a dish from my cart** which I don’t want to order anymore.

* Whenever deleting quantity of a dish happens, the cart will update accordingly and then the system will recalculate **subtotal** and total.

As a **Customer**, I want to **view my cart** so that I know what I added and the current order value so far.

* Whenever the cart is opened, the list of selected dishes and their quantities will be displayed. The subtotal cost and the total cost will be also showed in the summary panel.

1. **Checkout & Payment**

As a **Customer**, I want to **proceed to checkout** from the cart so that I can place an order.

* Whenever a Customer proceeds to order, then the system will display the list of dishes, their quantities, unit price, subtotal, delivery fee, and total amount.
* Whenever a Customer confirms the order, then the system will save the order with status = “pending confirmation”.

As a **Customer**, I want to **pay for my order** via supported methods so that I can complete my purchase.

* Whenever a Customer selects online payment, then the system redirects them to the secure payment gateway.
* Whenever an online payment is successful, the order status will update to “paid”.
* Whenever an online payment is **unsuccessful**, then the order will remain in its current status, and the Customer will be notified of the failure.
* Whenever a Customer selects **cash on delivery (COD),** then the order will be marked as “COD”.
* Whenever a COD order is successfully delivered and confirmed, then the order status will be updated to “completed”.

1. **Order**

As a **Customer**, I want to **track my order status** after placing an order so that I know the progress of my order.

As a **Customer**, I want to **view my order history** so that I can review the details and track order status.

As a **Merchant Admin**, I want to **confirm an incoming order status** (e.g., delivering, completed) so that I can process customer requests properly.

As a **Merchant Admin**, I want to **view order statistics** within a defined period so that I can monitor business performance.

* Whenever Merchant Admin selects a time period (today, last 7 days, last 30 days), then the system will display the total number of orders, total revenue, and average order value in that period.

1. **Rating**

As a **Customer**, I want to **rate my order** (1 – 5 stars) after receiving it so that I can share my experience.

1. **Access Control**

As a **User** (Customer/Merchant Admin /Admin), I want to **log in** so that I can access my view.

* Whenever a user with a Customer role does a login, then I will be brought to the home page/merchant page.
* Whenever a user with a Merchant Admin role does a login, then I will be brought to the merchant management page.
* Whenever a user with a System Admin role does a login, then I will be brought to the administration page.

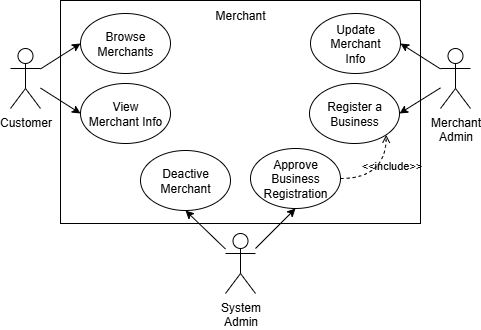
As a **User** (Customer/Merchant Admin/System Admin), I want to **log out** so that I can securely end my session.

* Whenever a User clicks “Logout”, then the system will terminate the current session.
* After logout, Customer will be directed to the home page (public).
* Whenever a Customer tries to access a protected page after logout, then the system will redirect them back to the login page.
* After logout, Merchant Admin and System Admin will be directed to the login page.

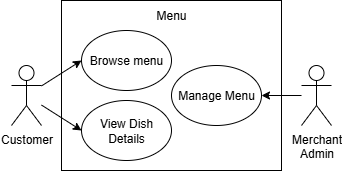
As a **Customer**, I want to **register for an account** so that I can log in to the system

#### Use case Summaries

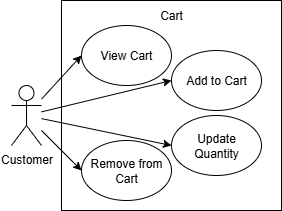
**Merchant**

****

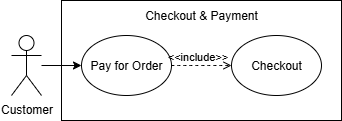
**Menu**

****

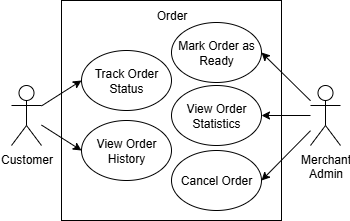
**Cart**

****

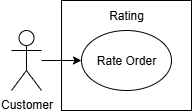
**Checkout & Payment**

****

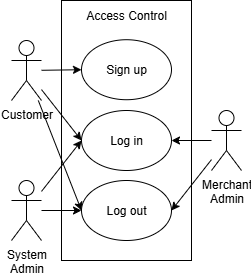
**Order**

****

**Rating**

****

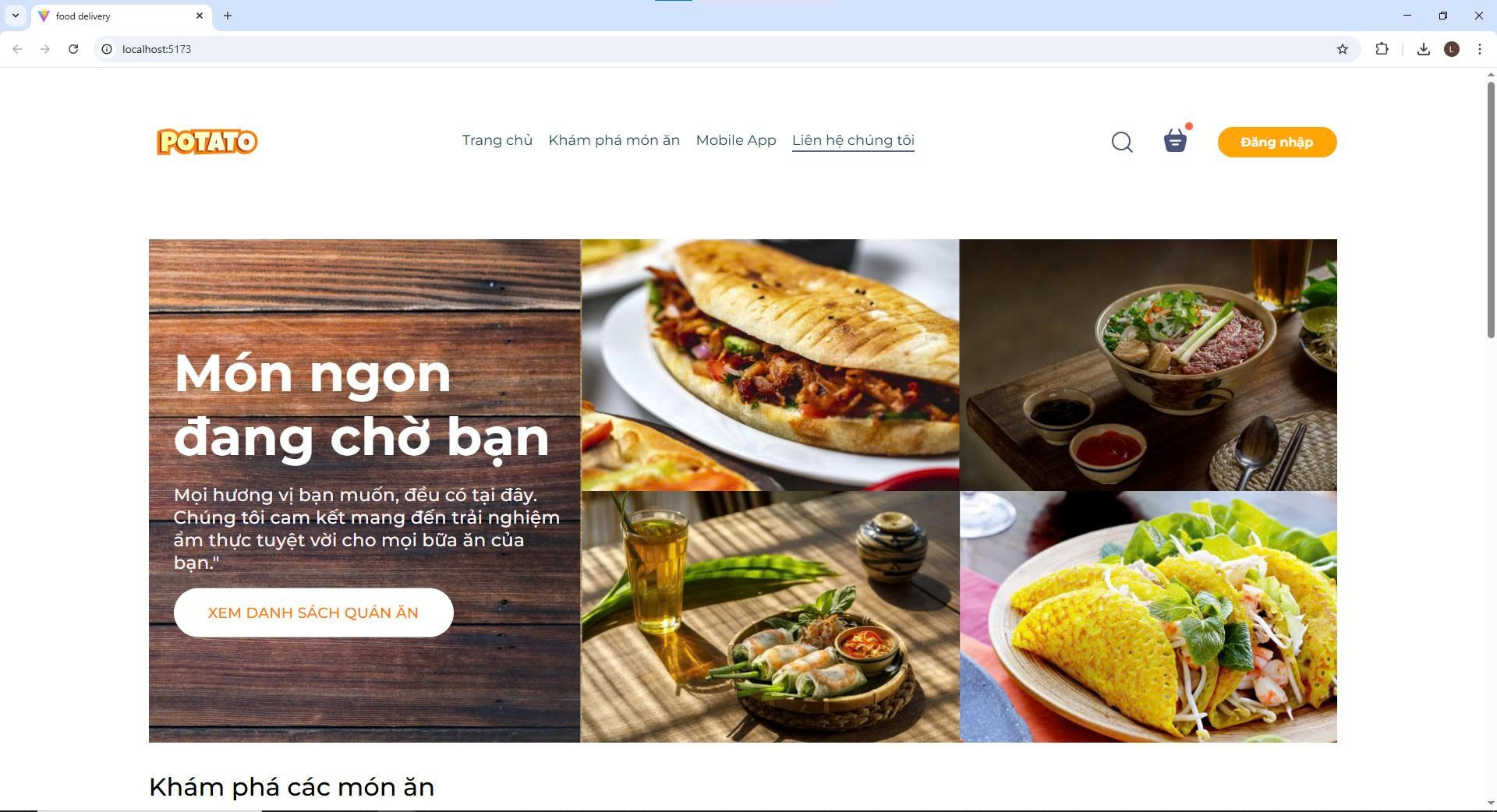
**Access Control**

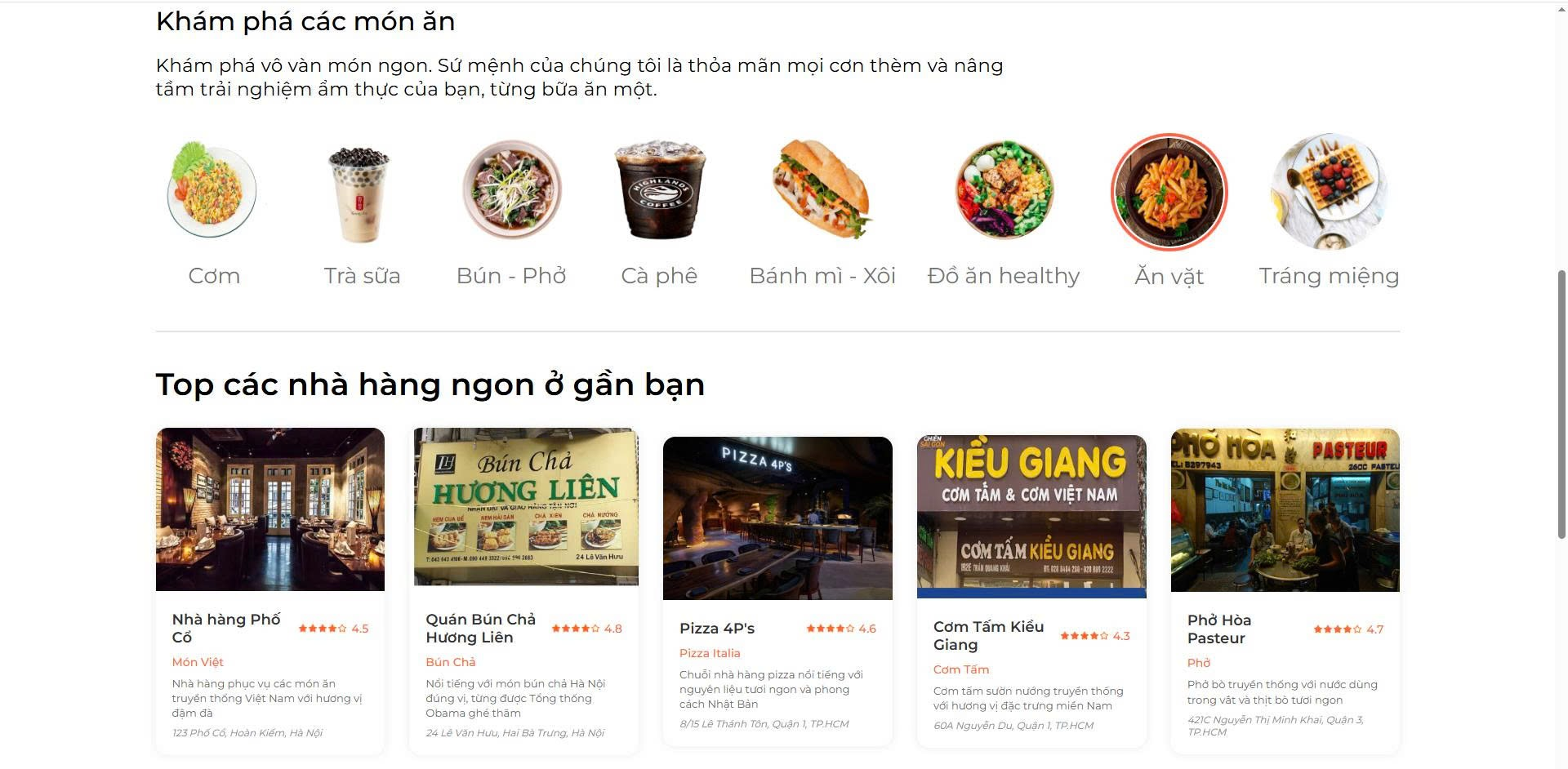
****

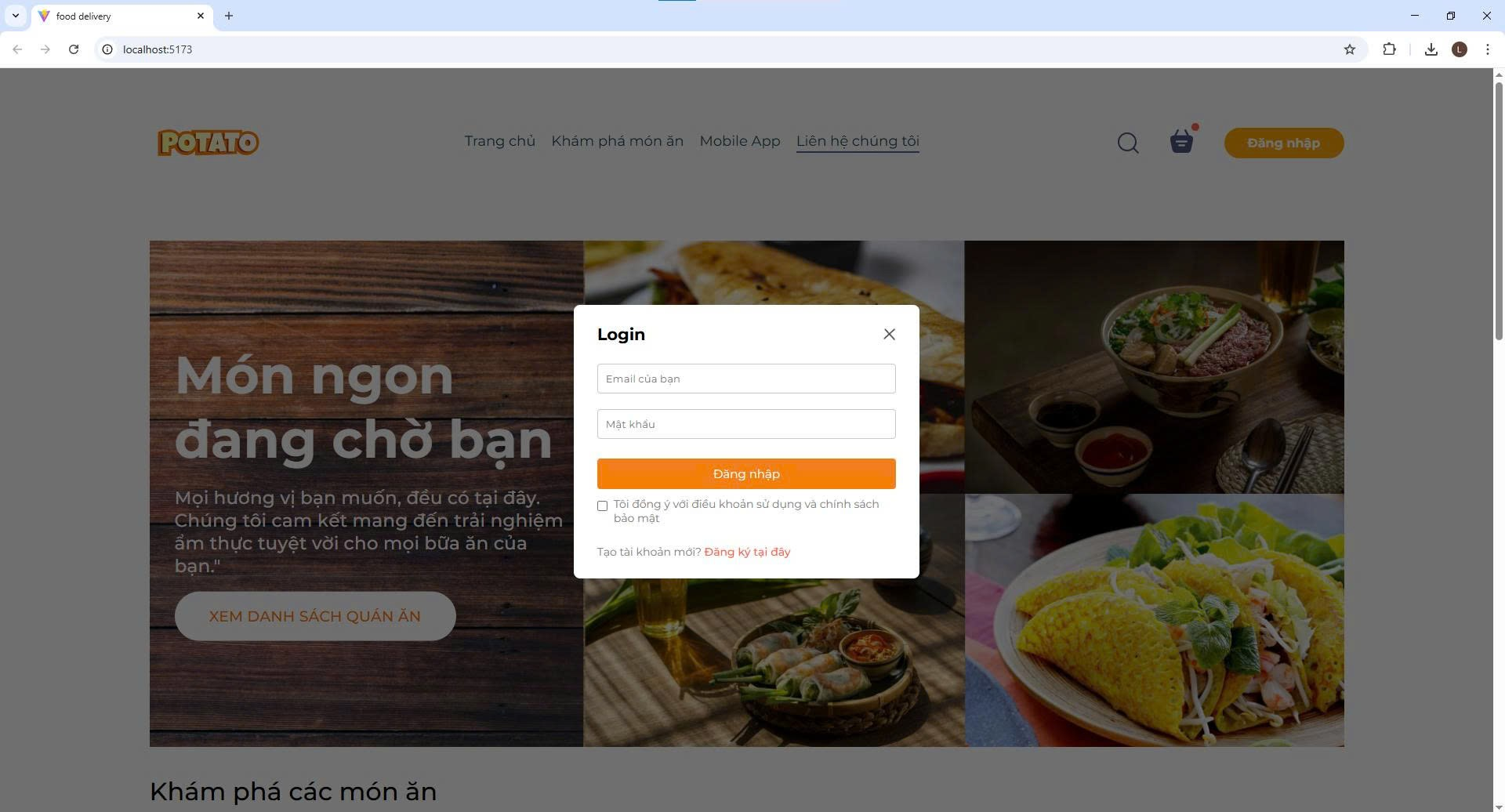
### User Interfaces

**Home page**

These are some user interfaces for user has role Customer, meeting several requirements: searching, cart, filtering with cuisine type, list of active merchants, sign up/log in

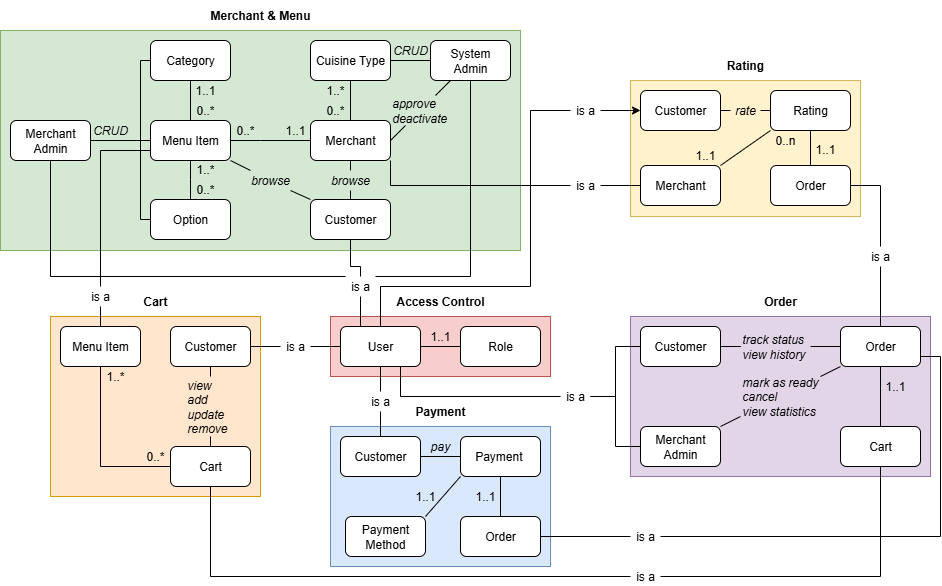






### Data Design

#### Conceptual Model



#### Object Model

#### Data Model

### Rest APIs Design

**Merchant**

* Customer see a list of merchants with search, filtering and sorting 🡪 get merchants by criteria (GET)
* Customer view detailed information of a selected merchant 🡪 get details of a specific merchant (GET)
* Customer browse all the dishes offered by a specific merchant 🡪 get all dishes (GET)
* Customer view detailed information of a dish 🡪 get details of a specific dish (GET)
* Merchant Admin update merchant’s details 🡪 update (PUT)
* Merchant manage merchant’s menu -> create (POST), retrieve (GET), update (PUT/PATCH), delete (DELETE)
* System Admin view all merchants 🡪 get all merchants (GET)
* System Admin approve new merchants 🡪 create merchant (POST)
* System Admin deactivate existing merchants 🡪 update merchant’s status (PATCH)

**Cart**

* Customer add a dish to cart 🡪 create a cart with selected dish (POST)
* Customer update quantity of a dish 🡪 update (PUT)
* Customer remove a dish 🡪 delete (DELETE)
* Customer view cart 🡪 get cart (GET)

**Checkout & Payment**

* Customer proceed to checkout 🡪 create a pre-order (POST)

**Order**

* Customer track order status 🡪 get details of an order (GET)
* Customer view order history 🡪 get list of order (GET)
* Merchant Admin confirming an incoming order status 🡪 update status of an order (PUT)
* Merchant Admin view order statistics within a defined period 🡪 get order statistics (GET)

**Rating**

* Customer rate order 🡪 create rating for an order (POST)

**Access Control**

* User (Customer/Merchant Admin/System Admin) log in (POST)
* User (Customer/Merchant Admin/System Admin) log out (POST)
* User (Customer) register (POST)

## Architecture Design

# TEST PLAN

## Introduction

### Purpose

### Definitions, Acronyms, and Abbreviations

### References

### Background Information

### Scope of Testing

* In Scope
* Test Stages

The following table lists the various stages of Potato application testing and the team member responsible for it.

|  |  |
| --- | --- |
| Stage | Team Responsible |
| Unit Testing | Development Team |
| Integration Testing | Testing Team |
| System Testing | Testing Team |

* Test Types

Potato application will undergo the following types of testing. All types of testing are explained in detail under Test Strategy section.

|  |  |  |
| --- | --- | --- |
| Type | Stage & Team Responsible | Description |
| Functionality Testing | Integration/System – Testing Team |  |
| Database Testing | Integration/System – Testing Team |  |
| Security Testing | Integration/System – Testing Team |  |
| GUI & Usability Testing | Integration/System – Testing Team |  |
| Code Testing | Unit Testing – Development Team |  |
| Smoke Testing | Unit Testing – Development Team |  |
| Regression Testing | Integration/System – Testing Team |  |
| Defect Validation Testing | Integration/System – Testing Team |  |
| Compatibility Testing | Integration/System – Testing Team |  |
| Interface Testing |  |  |

* Out of Scope
* Capacity planning and scalability beyond defined load/volume limits.
* Non-functional requirements not explicitly mentioned (e.g., disaster recovery).
* Testing on legacy browsers or operating systems not listed in the compatibility matrix.

### Constraints

### Risk List

## Requirements for Test

### Test items

### Acceptance Test Criteria

## Test strategy

### Test Types

### Test Stages

## Resources

### Human Resource

### Test Management

## Test Environment

### Hardware

### Software

### Infrastructure

## Test Milestones

## Test Deliverables

# TEST DESIGN

# TEST CASES – TEST REPORT

# CONCLUSION