CPT202

Final Report for Software Engineering Group Project

2022/2023 Semester 2

<Pizza Ordering System>

<2023.5.19>

Group number: <24>

Student Name: Changqing Lin

Student ID: 2039153

Project URL: http://121.199.161.166:8080/pizzaOrderingSys/login

Table of Content

Introduction

Up to Sprint3, the completed functions of the project include user registration, account management, order and pizza management for shop managers, and order placement and tracking for customers. The system also incorporates data statistics for analysis and requires approval from the software management monitoring center.

User Registration and Account Management: Both customers and store managers can register on the website and obtain different role accounts. The system distinguishes between customers and shop managers through their accounts and guides them to separate pages.

Shop Manager's Function:
Order Management: Shop managers can check order details, including order status and outstanding deliveries.
Pizza Management: Managers can manage pizza information by adding, deleting, and modifying pizza details, including sales status. They can also manage specific attributes such as size, toppings, and categories.
Data Statistics: Shop managers can view data statistics, such as revenue and customer comments, which help in analyzing issues and changing business strategies.

Customer's Function:
User Registration: Customers can register as new users by providing personal information such as phone number and address.
Order Placement: After logging in, customers can select pizzas that are available for sale, choose size and toppings, and add items to the shopping cart. They can confirm the ordered items and view the total amount before submitting the order.
Order Tracking: Customers can view the status of their current orders and access their order history to track previous purchases.
Payment and Delivery: Customers can choose their preferred payment methods during the order confirmation process. However, the actual payment connection is handled by a third party. Similarly, while the system tracks the order status, the actual delivery process is managed by a separate delivery service platform.
System Requirements: The system should be designed to handle a surge in the number of orders and prioritize smooth and fast ordering processes. The interface and operations should be user-friendly and easy to understand, catering to both the young and the elderly.

Approval Process: The completed system needs to be approved by the software management monitoring center before it can be implemented.
The solution involves various software modules and architectural design considerations. The software modules include user management, pizza management, menu management, shopping cart, order management, and statistical reports. The system architecture incorporates tools and frameworks such as JPA with JDBC, Spring Boot

with Spring MVC, and Thymeleaf for the frontend. JPA with JDBC is used for object-relational mapping and database connectivity. Spring Boot with Spring MVC provides a flexible web framework, and Thymeleaf is used for templating and frontend rendering.

Strengths of the system include:

The system addresses the needs of both managers and customers, providing comprehensive functionality for order management, customization, and reporting.
The use of JPA with JDBC and Spring Boot with Spring MVC allows for efficient data management and a scalable web application.
The inclusion of a centralized database enables efficient storage and retrieval of relevant data for both managers and customers.
The use of a custom authentication system with SMS verification enhances the security and reliability of user authentication.

Weaknesses or considerations of the system may include:

The reliance on third-party services for payment processing and delivery may introduce dependencies and potential integration challenges.
The text does not provide information about the technology stack, platform, or programming languages chosen for development, which could impact the system's implementation and scalability.
The text mentions risks related to data privacy, security, and legal compliance, but it does not elaborate on specific measures or protocols to address these risks. Ensuring data security and compliance with relevant regulations should be a crucial consideration.
The system's performance and scalability may be affected by the increasing number of orders and users, requiring continuous monitoring and optimization.
Overall, the provided solution outlines a comprehensive web-based pizza ordering system that caters to the needs of both managers and customers. It leverages modern technologies and frameworks to provide a user-friendly interface, efficient data management, and secure authentication. However, further considerations and implementation details are necessary to address potential challenges and ensure the system's success.

This report will focus on my implementation of 5 PBI, including all the function for customers to select items on the menu and editing selected items in shopping cart before they place an order. The report's structure is as follows. First, there is a discussion of my understanding of scrum and how I doing in scrum with my team members. Second, this report will explain and justify the PBI I have individually developed. Third, this report will show how to handle the change of requirement with examples of my individually developed PBI. The last section discusses legal, social, ethical, and professional issues, looks for innovative ways to advice better performance of this system. Finally, there is a conclusion about what I learned and what I could improve in the future.

Software Development Process

From my perspective, Scrum is an effective way to manage software development projects due to its flexibility, transparency, and focus on delivering value to the customer. It not only structures the development process but also enhances communication among the team and stakeholders, leading to better products and higher team satisfaction. Moreover, it requires discipline, open communication, and a willingness to adapt to change for it to work effectively.

Taking my responsible part of design and implementation the function of menu and shopping cart for customer in our Pizza Ordering System project as example, on the very beginning of planning stage, our group consider that one of the most important parts of the requirement from product owner needs the system could provide the function for their customer is customers are able to place an order. However, customers should able to choose the product they want to buy before they place an order, therefore, with the Sprint Planning in our group, a single Menu page and a check out function in my part got the highest priority. Then, our group just get started with building the minimal feature set we build what was planned next we test and review the result, after we keep doing this about 3weeks, a potentially shippable product in the first Sprint. And then up to the second Sprint, the whole process last about 2 weeks, we hold daily scrum meeting every day to discuss what we have completed, everyone's working process, and anything that might be blocked or need help; During that time, our group got the problem of fail to insert data into shopping cart because customer in our system did not have shopping cart at that time, and to deal with this problem we need to realize the function of generate a shopping cart when a customer successful registration, then I asking help for the team member who in charge of customer registration and user part to solve it. And when going into the third Sprint, during that time our team working on what we can improve our process, and setting the service to launch our project.

During the processes of scrum, I found the reason why scrum enhance the efficiency could be following reasons: 1. Our group maintains and prioritizes the Product Backlog, which is the single source of truth for everything that needs to be done. This constant reprioritization ensures that the team is always working on the most valuable features. 2. Scrum emphasizes active collaboration among team members and stakeholders. Everyone is encouraged to actively participate, share their opinions, and work together to solve problems, which can lead to more creative solutions. 3. Regularly held meetings such as daily stand-ups, sprint reviews, and sprint retrospectives facilitate ongoing communication and feedback. This not only ensures alignment among team members but also allows the team to continuously learn and improve. 4. Scrum encourages transparency by maintaining a visible backlog of tasks, and progress is tracked publicly using tools like a Scrum board or a burn-down chart. This provides everyone a clear understanding of what is being worked on and the status of each task.

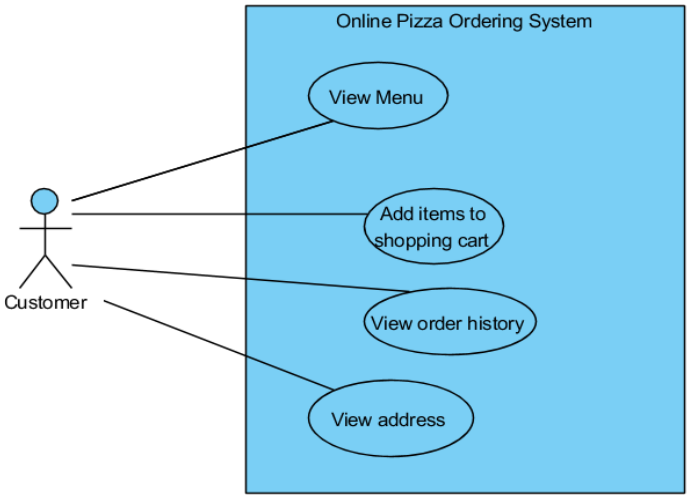
In summary, Scrum facilitates software development processes by promoting

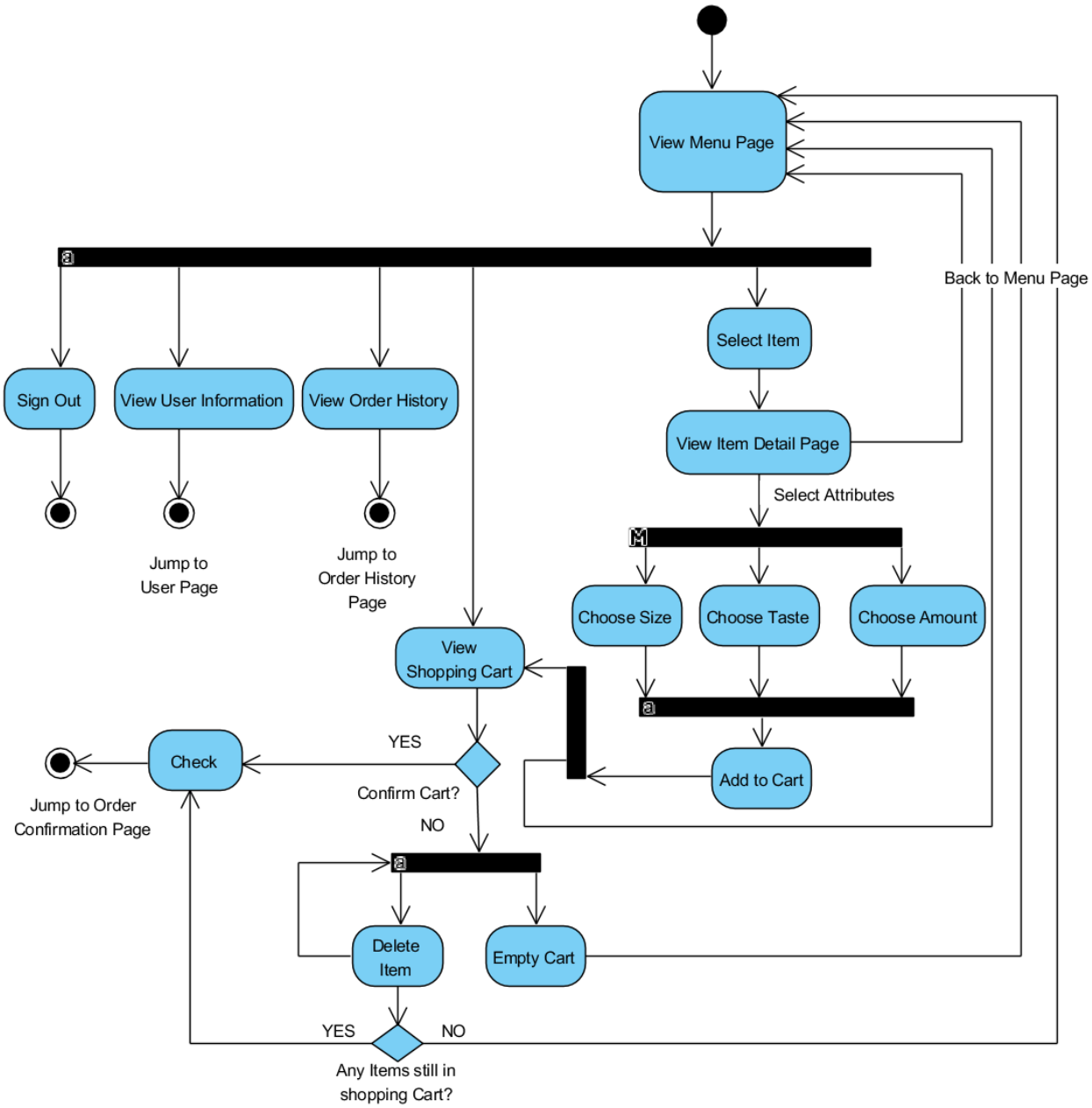
transparency, collaboration, and feedback, while empowering the team. These aspects of Scrum lead to effective communication, improved productivity, and better-quality software products.

Software Design

Here are 5 PBI I have individually developed:
USER1003 Pizza Menu
USER1004 View Details
USER1005 Select Attributes
USER1006 Check Shopping Cart
USER1007 Delete Item

The activity diagram and use case diagram shows the scope and process of these PBIs.



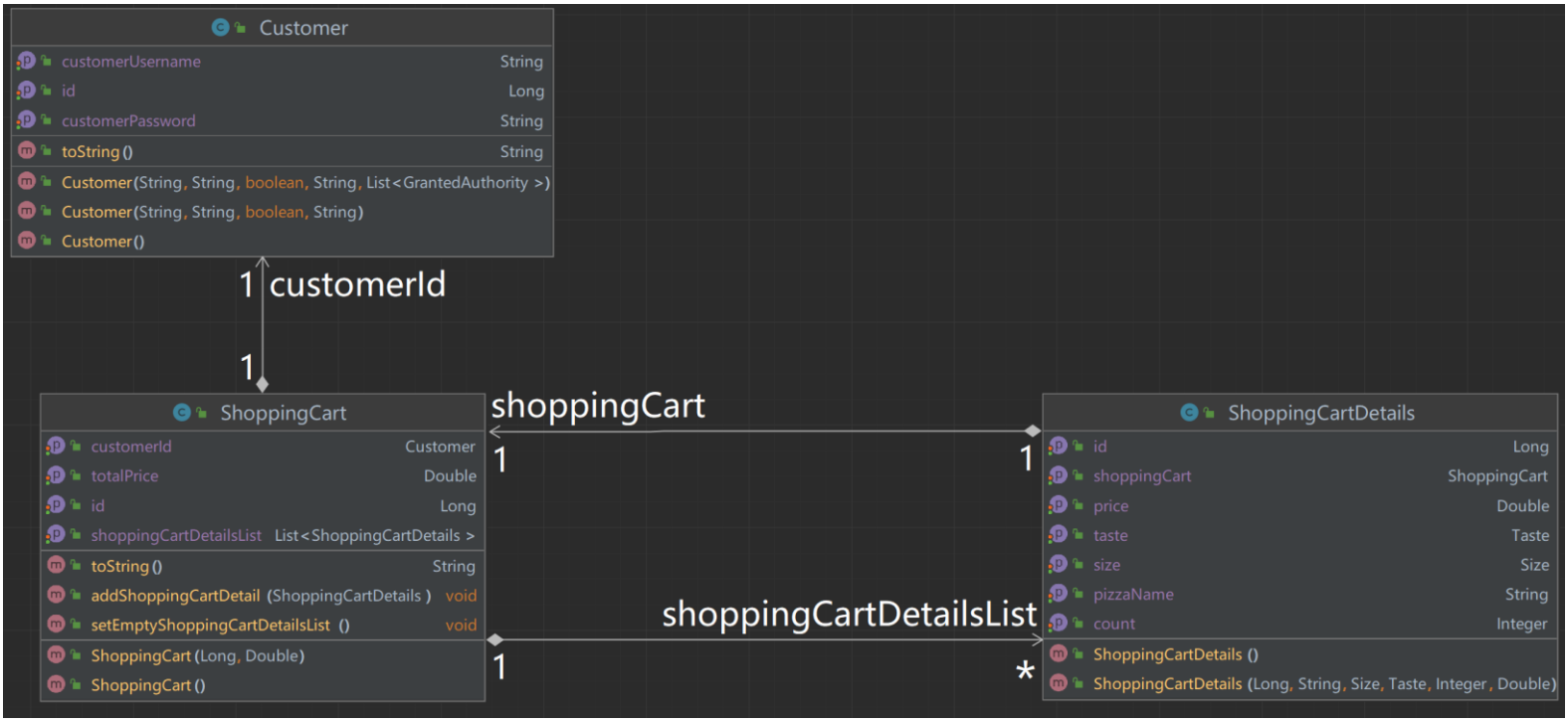


For PBI USER1003 Pizza Menu, USER1004 View Details, and USER1005 Select Attributes, I consider the menu for customer to choose items should have the following characteristics:

1. Easy Navigation: The menu should be designed with a clear and simple navigation system. It should be easy for users to find specific items, go back to the previous pages, or jump to different sections of the menu.
2. Visual Appeal: Make use of high-quality images and possibly short videos for each dish. A picture is worth a thousand words, and it's particularly true when it comes to food. This helps the user in making decisions.
3. Customer satisfaction: ① In Control: The process should be intuitive and user-driven. They should be able to easily navigate the menu, select items, and easily jump to other page for the function they need such as editing their address or view their order history. ② Good using experience: To make it more convenient for the user, I design the item detail page and the shopping cart page could display in the menu page at the same time to ensure that the experience for customer could be enjoyable and efficient, leading to a satisfying transaction.

Therefore, for the implementation of front-end page design of the menu, I used flex layout to divide the page into two major parts. The left part is for the user to select the product category and place the logout button, while the right part is divided into three areas, the top part place the "User Information" button and the "View Order History" button, and the middle part is for the display of the dishes, using warp arrangement. This is in order to load data from the back-end (those item data which are already input into database) to display in menu page can be neatly arranged. I also set each item as a button, all buttons on the Menu Page and Item Detail Page are setting it to hover style in CSS, so that when the user mouses over the dish, he or she can intuitively understand which button he is going to click on and which attribute he has selected.

For the implementation loading item information from the back-end to Menu Page and Item Detail Page, I realize through JavaScript ajax function, which is an asynchronous mode of communication between a browser and a server. The transmission is done through json data, and write the required data in the database to html via a for loop. And I use ".innerHTML" method in a defined function to print HTML on the middle area of Menu Page. Thus, for those data have already input into the database (back-end) can be displayed on the menu page according to whether the dish's statue is on sale or not. And these function could be executed through the controller on the back-end when customer login into the system.



For PBI USER1006 Check Shopping Cart and USER1007 Delete Item, I consider the shopping cart for customer to check or edit their chosen items and check out should achieve the following characteristics:

1. Order Review and Modification: Users should be able to easily review all items they've selected, along with their details such as quantity, customization options (if any), and prices. They should be able to modify these details – add or remove items, change quantities, or adjust customization options.
2. Transparent Pricing and Information: The cart should display key information of chosen items such as item name, tastes, size, amount and price. And the cart should show a clear breakdown of costs including each item costs and total price. The total price must be able to change in real time as customer add or subtract items.
3. Easy Navigation: Users should be able to seamlessly navigate between the cart and the menu. If they want to add more items or make changes, they should be able to do so without losing their current selections. The goal is to make the journey from menu browsing to order checkout as seamless, intuitive, and enjoyable as possible. A well-designed shopping cart plays a crucial role in achieving this.

Therefore, for the implementation of front-end page design of the shopping cart, I use the hidden element of HTML to realize that I can hide Shopping Cart Page behind the Menu Page. And I also realize that when customer click on “Shopping Cart” button on the menu, then the Shopping Cart Page will be displayed, and when customer click again, then the Shopping Cart Page will disappear; I realize this function through a “.style.display” method in a hidder function of JavaScript on the HTML.

For the implementation the function of display and listing key information of chosen items such as item name, tastes, size, amount and price and able to change in real time when customer add or subtract items, the back end is design is: for one customer possess one unique shopping cart, and all the details in each shopping cart will all collect and store into one table name “shoppingCartDetails”, do in this way is design for easier to editing the items in the shopping cart when customer doing operation on the web page.

For example, when customer want to delete one item with same item name and item based price but with different topping, then it is easier to delete the item based on the id of data in the table in “shoppingCartDetail”.

Therefore the implementation are as follow steps: ①Create the Entity of shoppingCart and shoppingCartDetails, shoppingCart only store the customer id, name and total price, each shoppingCart possess its unique id; shoppingCartDetails will store all relevant information of chosen items and possess its unique id either. ②Write the repository of shoppingCart and shoppingCartDetails, each shoppingCartDetails are able to find, and delete by certain id of shoppingCart while each shoppingCart are able to updated and find by customer name. ③Write the service of shoppingCart and shoppingCartDetails (the code is show in appendix), the price of one type of item display in the shopping cart is count as: (base-price+taste)*size*discount*amount, this calculation will doing in shoppingCartDetailsService, and the result will extract by shoppingCart. ④Write the controller of shoppingCart and shoppingCartDetails, to create the APIs to respond the operation of customer done on the front-end. The process need the APIs are: Ⅰ API (Get the current customer's shopping cart information from the backend); front pass customer_name, then the data returned by the back-end is the List of data in shoppingCartDetails and total price. Ⅱ API (Add the product to the shopping cart).
The parameters passed by the front end are user id, product id number of products product size product taste, then input these data into database on the back-end. Ⅲ API(Delete the specified products in the shopping cart). Front-end pass user_name, cart_id of each line (shopping_Cart_Detail_id), then the backend receives and deletes. ⅣAPI(Empty the shopping cart). Front-end pass user_name, and then backend empty data in the shoppingCart and relevant shoppingCartDetails data.

Change Management

Changes in requirements are a common occurrence in Scrum projects. To handle these changes, the following steps can be followed:
1. Understand the need to change: First, you need to understand the specifics of the change and the reason for the change. You need to communicate with the stakeholders to understand whether the new requirements or the existing requirements need to be modified.
2. assess the content of the shadow, need to assess the change for the project schedule and the mission to consider the priority of the new requirements, workload, time, feasibility and other factors.
3. Consult with the product owner and group member: Before handling the change, you need to consult with the product owner. It is important to clarify the impact of the new change on the Sprint objectives, all elements of the work that the team needs to complete during the Sprint cycle, and the time and resources required to support it.
4. Update The list of tasks in the Sprint Backlog needs to be updated as deemed necessary by Sprint. This requires the team to recalculate task points and time estimates and determine how the newly scheduled requirements will be achieved.
5. Inform Stakeholders: When changes are identified, they need to be communicated to

stakeholders to keep them in sync and keep them properly informed. All changes should be documented for review.
6. Summarize at the end of the iteration: If the changes reprioritize the Sprint or adjust the schedule of tasks, this should be summarized at the end of the iteration and subsequent Sprint cycles and requirements plans can be readjusted.

Taking my project as example: In the very beginning, we decide to manage each item in the Menu Page has its relevant Item Detail Page, and the Shopping Cart Page are also separated with Menu Page, which may cause the overload of sever when the project running with many customers and it is negative for the using experience for customer. Therefore, based on above analysis of negative impact with my team members, we decided to reschedule the Sprint of PBI USER1003 and USER1006 update the list of tasks in the Sprint Backlog, and finally summarized at the end of the iteration and subsequent Sprint cycles and requirements in our project.

Legal, social, ethical, and professional issues

Here are issues we concerned on Pizza Ordering System as follows:
Data Privacy and Security: Customers' personal and financial data will be processed and stored. How this data is collected, stored, used, and shared should be done in compliance with relevant laws. Failure to adequately secure this data could result in financial losses for customers and serious legal consequences for the company.
Ethical Sourcing and Sustainability: If your software includes features that show information about food sourcing or sustainability practices, these should be accurate and verifiable. Misrepresentation can have ethical and legal consequences.
Professional Ethics: Developers should adhere to professional codes of conduct, which include honesty, integrity, and competence. This means creating software that works as advertised, that doesn't include hidden features or 'spyware,' and that respects the rights and expectations of users.
False Advertising: All the information presented in the software, like pricing, deals, or delivery time, should be accurate. Misrepresentation could lead to consumer frustration and possible legal action.
Equality and Non-discrimination: Offering services without discrimination based on race, gender, religion, or other protected characteristics is both a legal and ethical obligation.
Intellectual Property: Use of third-party software, components, images, or other intellectual property should be properly licensed and credited.
Consumer Protection Laws: The software should align with applicable consumer protection laws, which cover things like the right to a refund or the right to cancel an order within a certain time frame.
Cultural Sensitivity: The software should be mindful of cultural differences and nuances, which could influence menu item descriptions, user interface design, or customer interactions.

Here are the possible solutions:

For Legal Issues:
Data Privacy and Security: Comply with data protection laws. Solution: Secure data storage and transfer, implement a comprehensive data protection policy, and ensure users' consent.
Consumer Protection Laws: Ensure products are as described, prices are accurate. Solution: Maintain truthful advertising, clearly present all charges before checkout.
Intellectual Property: Avoid unauthorized use of third-party assets. Solution: Always acquire the necessary licenses for any third-party software or images.
Social Issues:
Cultural Sensitivity: Respect cultural differences. Solution: Use neutral descriptions and ensure localized content is culturally sensitive.

For Ethical Issues:
Transparency: Be clear about data use. Solution: Provide a clear privacy policy and keep users informed about any changes.
Ethical Sourcing: Ensure truthful representation of sourcing practices. Solution: Source from verified suppliers and be transparent with customers about it.

For Professional Issues:
Quality Assurance: Ensure the system functions as advertised. Solution: Implement a rigorous QA process and conduct regular audits.
Updates and Maintenance: Keep the system secure and functional. Solution: Regularly schedule system maintenance and updates, and communicate effectively with users about them.

Conclusion

The project journey spanning six weeks and three sprints has been incredibly fruitful and enlightening, especially for me first time to act in role as a member in a scrum team. I'm pleased with the progress and results of the tasks completed, which have significantly bolstered my practical development skills and code-writing abilities.

Additionally, adaptability was a key learning point, given that swiftly adjusting to new environments and team members is crucial in a project setting. It was also apparent that the scrum process needs to be tailored specifically to a team's dynamics rather than strictly adhering to textbook methods. Decisions on the duration of a sprint, the number of PBIs to complete in a single sprint, etc., should be customized to a team's specific needs and capabilities.

After going through the whole process of complete front-end and back-end development, I have deeply explored various aspects of front-end and back-end technologies, including the use of Thymeleaf, HTML, CSS and JavaScript on the front-end, and learned how to develop software based on MVC pattern under springboot framework with JPA. Initially, deciding on the appropriate tools and how to use them

was a challenge. However, consulting with teachers and experienced colleagues, coupled with various online resources like Bilibili and CSDN, significantly helping my learning process.

Apart from technical skills, a critical takeaway was the value of cooperation in project development. It became clear that individual strengths vary greatly, and it's essential to leverage these diverse capabilities effectively. For instance, while some are naturally gifted in management, others may excel in hands-on development. Initially, roles may be assigned based on interest. Still, if performance isn't up to the standard, the flexibility to switch roles and ultimately finding the best fit is crucial for overall project success.

In conclusion, this project has enriched my learning experience and reinforced the importance of agile processes, technical skills, and collaborative efforts in successful project development. I look forward to implementing these insights in my future career and work.

Appendix

<You could include various screenshots and diagrams here>

```
2 个用法   Mr.kient
public ShoppingCart newShopCart(ShoppingCart shopCart) {
    System.out.println("new shopcart with Id: "+shopCart.getId());
    return shopCartRepo.save(shopCart);
}
7 个用法   Mr.kient
public ShoppingCart loadShopCartByCustomer(Customer customer) {
    if(!shopCartRepo.findShoppingCartByCustomerId(customer).isPresent()) {
        throw new NullPointerException("There is no existing shopcart");
    }
    return shopCartRepo.findShoppingCartByCustomerId(customer).get();
}

1 个用法   Mr.kient *
@Transactional
public void updateShopCartByCustomer( ShoppingCartDetails shoppingCartDetails,Customer customer) {
    Double olderPrice = shoppingCartDetails.getShoppingCart().getTotalPrice();
    Double newPrice = olderPrice + shoppingCartDetails.getPrice();
    shopCartRepo.updateShoppingCartByCustomerId(newPrice,customer);

}
2 个用法   Mr.kient
@Transactional
public void setTotalPriceZero(ShoppingCart shoppingCart) {
    Customer customer = shoppingCart.getCustomerId();
    shopCartRepo.updateShoppingCartByCustomerId( price: 0.0,customer);
}
```

```java
public List<ShoppingCartDetails> getShoppingCartDetailsList() { return shopCartDetailsRepo.findAll(); }


1 个用法  Mr.kient
public ShoppingCartDetails setShopCartDetail(Pizza pizza,
                                             Size size,
                                             Taste taste,
                                             Integer amount,
                                             ShoppingCart shoppingCart,
                                             Customer customer){
    ShoppingCartDetails shoppingCartDetails = new ShoppingCartDetails();
    shoppingCartDetails.setPizzaName(pizza.getName());
    System.out.println("ShopCartDetailService  Pizza name :"+pizza.getName());
    System.out.println(pizza.getPrice());
    System.out.println(pizza.getDiscount());
    System.out.println(size.getPrice());
    System.out.println(taste.getPrice());
    System.out.println(amount);

    double price= (pizza.getPrice() + taste.getPrice())
            * (size.getPrice()*0.01 +1) * amount * (pizza.getDiscount()*0.01);
    price = (double) Math.round(price * 100) / 100;
    shoppingCartDetails.setPizzaName(pizza.getName());
    shoppingCartDetails.setPrice(price);
    shoppingCartDetails.setSize(size);
    shoppingCartDetails.setTaste(taste);
    shoppingCartDetails.setCount(amount);
    shoppingCartDetails.setShoppingCart(shoppingCart);

    shopCartService.updateShopCartByCustomer(shoppingCartDetails, customer);

    ShoppingCartDetails res = newShopCartDetail(shoppingCartDetails);
    System.out.println("ShopCartDetailService  save shop cart detail :");

    return res;
}
```

```java
1 个用法  Mr.kient
@Transactional
public void deleteShopCartDetail(Long id) {
    shopCartDetailsRepo.deleteById(id);

}

3 个用法  Mr.kient
@Transactional
public void emptyShopCart(ShoppingCart shoppingCart) {
    shopCartDetailsRepo.deleteByShoppingCart(shoppingCart);
    shopCartService.setTotalPriceZero(shoppingCart);

}
```