

A series of five parallel white lines of varying lengths, slanted diagonally from the top right towards the center of the page.

Soft Engineering Coursework

Final Report

Group 102

Group 102 Members

Leader	Yixuan Lu	171043662/2017213202
Member	Yitong Luo	171043721/2017213237
Member	Tingyang Lu	171043651/2017213178
Member	Yujun Jin	171043329/2017213109
Member	Qizhou Zhang	171044429/2017213107
Member	Wenlong Liu	171043558/2017213143

1.Introduction

This semester, we developed an automatic order and management software system in the self-service machine for a ramen restaurant called Totoro-Ramen. During the development, we use Agile project management methods to organize and manage our software project. The Agile methods make our software development faster, better quality and more efficient. The software is suit well for the requirement. In addition, the release cycle of Agile development model makes us build the early version of the system and deliver it in the very early time, and gradually update more functions.

2.Project management

2.1 Choice of tool

The current special situation forces our group to collaborate online, we have thought about using IM applications to share files, but end up with using professional tools. SVN and git is our main option, considering the different learning curve and fault tolerance of these two tools, we decide to use git for

The screenshot shows the GitHub interface for the repository 'luty4ng / Totoro-Ramen'. At the top, there are buttons for 'Watch' (1), 'Unstar' (3), and 'Fork' (2). Below this is a navigation bar with links to 'Code', 'Issues' (0), 'Pull requests' (0), 'Actions', 'Projects' (0), 'Wiki', 'Security' (0), and 'Insights'. The repository name is followed by the description '【软工大作业】龙猫拉面点餐系统'. Below the description, there are statistics: '88 commits', '1 branch', '0 packages', '0 releases', and '2 contributors'. A 'Branch: master' dropdown and a 'New pull request' button are visible. On the right, there are buttons for 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The main content area shows a list of files and their latest commit information:

File	Commit Message	Time
bin	Update iteration 5	24 seconds ago
data	Update iteration 4	8 days ago
doc	Update iteration 5	24 seconds ago
lib	Update weekly survey function	last month
picture	Update iteration 4	8 days ago
src	Update iteration 4	8 days ago
.gitignore	Update information	last month
README.md	update README	last month

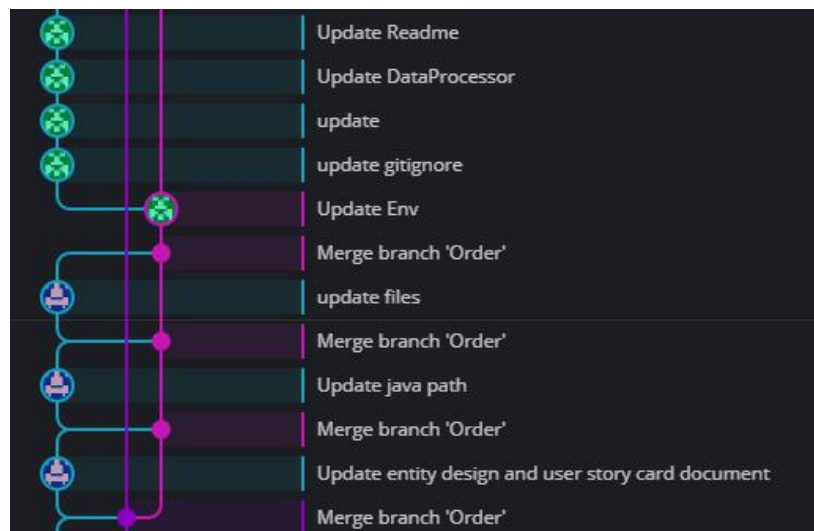
collaborative development and

Figure 1: *Our GitHub repository web page*

version management. Building a git server for our own is never an option, so we host our project to GitHub remote collaboration repository, and in order to simplify the operation we choose Gitkraken(an application for git) as a graphical tool for git.

2.2 Project collaboration

One of our group member is hosting the GitHub repository, and only maintain a master branch in remote. We require every group member to create a new branch in local when they start to develop their own part of function, who also needs to merge their work to local master branch then push them into remote master branch when they finished. In this way, we can iterate our projects step by step and



generate development logs automatically with the help of Gitkraken.

Figure 2: *Code management in Gitkraken*

2.3 Decision making

We host an online meeting on Friday or Sunday (depend on the schedule of team member) every week to discuss the existing problems and specify requirements. Our group members have multiple ways to feedback the their problem, we can discuss technical issues in Wechat or design problem through github message board. Our group leader will collect those problems and share with other member at weekly meeting, then we will decide the next iteration jobs together according to the backlog and tasks' priorities.

2.4 Planning, estimating and adapting

For the aspect of planning, we made plans based on the principle of *Arrangement for next iteration must be clear, for further iteration can be concise*. We have made a story list in the beginning that contains most arrangements for the project, which is finishing the noodle select and order functions. The GUI part of work was in developing at the same time since we need an interface to do human testing. The member function is the last because it is a kind of independent system. As for the

manager system, we keep this system matching with the order system. For the aspect of estimating and adapting, we firstly estimate whether the function was implemented, if so then we test it with some boundary cases to check its fault tolerance and stability. In the end we consider the coupling and cohesion of our project, which is means like, if a part of code is over coupled, we will decouple it before the next iteration. The main method we use to adapting the change is to divide every iteration into three steps: 1. fixing previous problems, 2. finishing temporary job, 3. creating extension interface. Step one is the way we adapting the changes, step three is the way we leave space for changes. By using this method, we have guarantee that our project can run correctly in each iteration.

3. Requirements

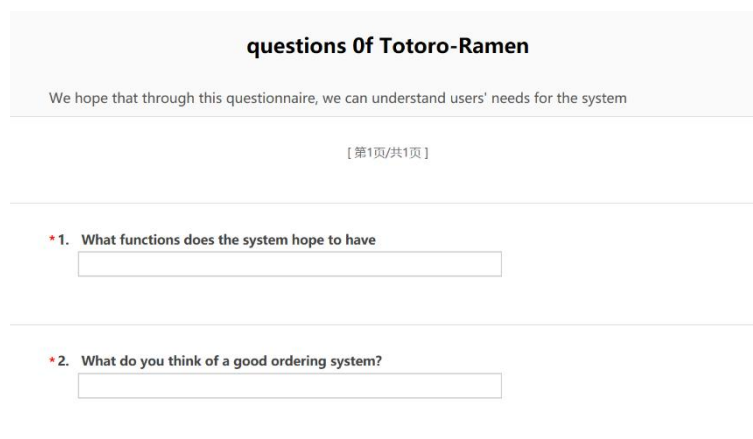
3.1 Requirements finding techniques

➤ Group discussion

We had a group meeting to discuss product requirements. We use Tencent meeting as a tool. In the meeting, each group member expressed his or her opinion about the requirements and give out the function we need to do. We drew up a preliminary list of requirements.

➤ Questionnaire

We designed a questionnaire to investigate the needs of users. To make sure the results are realistic and realistic, we surveyed people of all ages and groups. At the same time, in order to obtain high quality and sufficient quality questionnaire, we set up incentive system. Every user who fills out the questionnaire carefully will get a small reward.



questions Of Totoro-Ramen

We hope that through this questionnaire, we can understand users' needs for the system

[第1页/共1页]

*1. What functions does the system hope to have

*2. What do you think of a good ordering system?

Figure 3: *Some questions in the questionnaire*

➤ Investing current order system

We investigated the ordering system of KFC, McDonald's and Pizza Hut. We learn their UI and order flow, analyze what requirements the ramen customers want. In addition, interviewed our parents what kinds of requirements a ramen restaurant should be. These suggestions were helpful to our ordering system.

3.2 User story writing

User Stories are sentences written in everyday language or business language in software development and project management. We first discussed the overall workflow and arrangement through group meetings, and then broke the task into several epics. Then, divide epics into specific stories. In each iteration, we will modify, expand and adjust it according to the production schedule and the actual problems we encounter.

3.3 Priority and estimation rating

Based on MoSCoW theory, we prioritized the various parts to develop our production sequence and iteration plan.

- Must have: User interface; Ordering system; Payment system; The administrator modifies the data system; Administrator statistics system
- Should have: Membership registration system; Membership discount system; Dining style selection system; Ingredients are added to the system
- Could have: Detailed order feedback; Member modification information function; Analyze the sales status of ingredients
- Want to have: More optimized interface design

In the backlog, we point out every user story with a priority number and the estimating point.

Story ID	Story Name	Priority Number	Estimating story point
01a	System selection function When you start using the system	1	1
02a	Order ramen taste selection function	1	1
02b	Ingredient selection function	1	2
02c	Spiciness selection function	1	1
02d	Adding ingredients function for extra ingredient	1	3
03a	Dining way function	1	2
05a	Start Background management system	1	1
04a	Direct payment function	2	5
06a	Membership authentication function	2	5
06b	Intention selection function	2	2
06c	Registered function	2	3
06d	Login function	2	3
06e	Member information interface	2	2
04b	Virtual stamps payment functions Enough coupons	3	8
04c	Virtual stamps accumulation function	3	8
05b	View Statistical function	4	5
05c	Modify menu function	4	5
05d	Product sales statistics function	5	8

05e	Spiciness statistics function	5	8
07a	Interface optimization	5	2

Table 1: *Story with priority and estimating number*

3.4 Iteration Plan

In the iteration process, we put the core task of the program in the first iteration, and the work that has a greater impact on the subsequent work is also placed in the first iteration. We must formulate a clear and complete program framework at the beginning, and have a prototype of a program that can achieve basic functions, so as to complete the plan efficiently. In the following iterations, we will improve and modify our previous content based on the teachers' opinions and feedback, and incorporate the functions that we think the program should have in addition to the core functions. Fixing bugs and optimizing the user experience are also our work in the next few iterations.

➤ iteration 1(Mar 23 — Apr 3):

The establishment of the overall framework and the achievement of core functions, e.g., order ramen, price payment and modify menu price. Make sure that the first version can fit the fundamental requirement.

➤ iteration 2(Apr 6 — Apr 17):

Add VIP member functions to the system, including login and regist functions. Complete payment and manager interface.

➤ iteration 3(Apr 20 — May 1):

Finish the functions that related to virtual stamps. Add functions to examine the input format and warn the user to input the right data type.

➤ iteration 4(May 4 — May 15):

Add the complete version of modify menu function which contain modify price and modify ingredient remain.

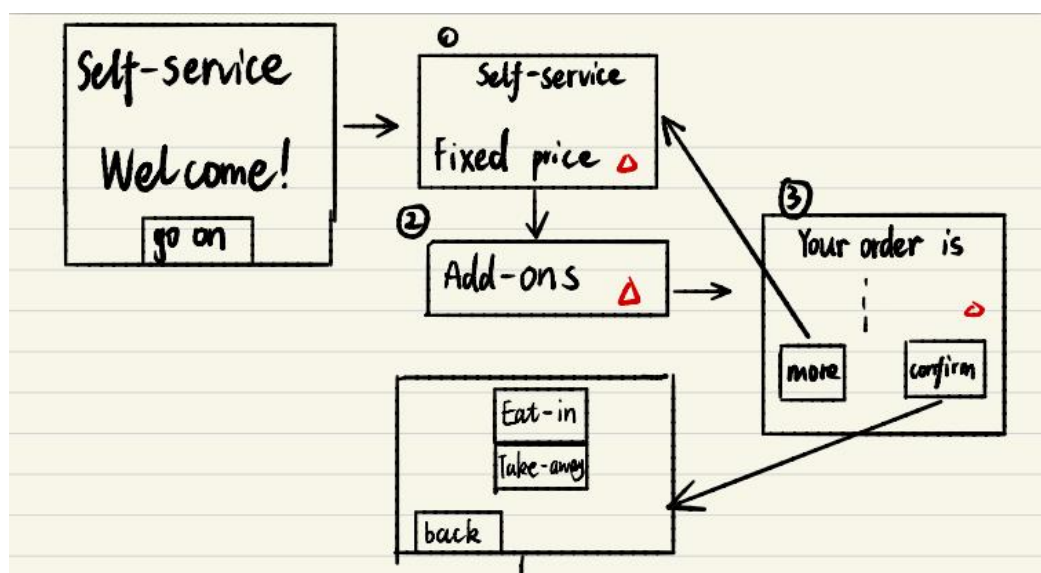
➤ iteration 5(May 18 — May 29):

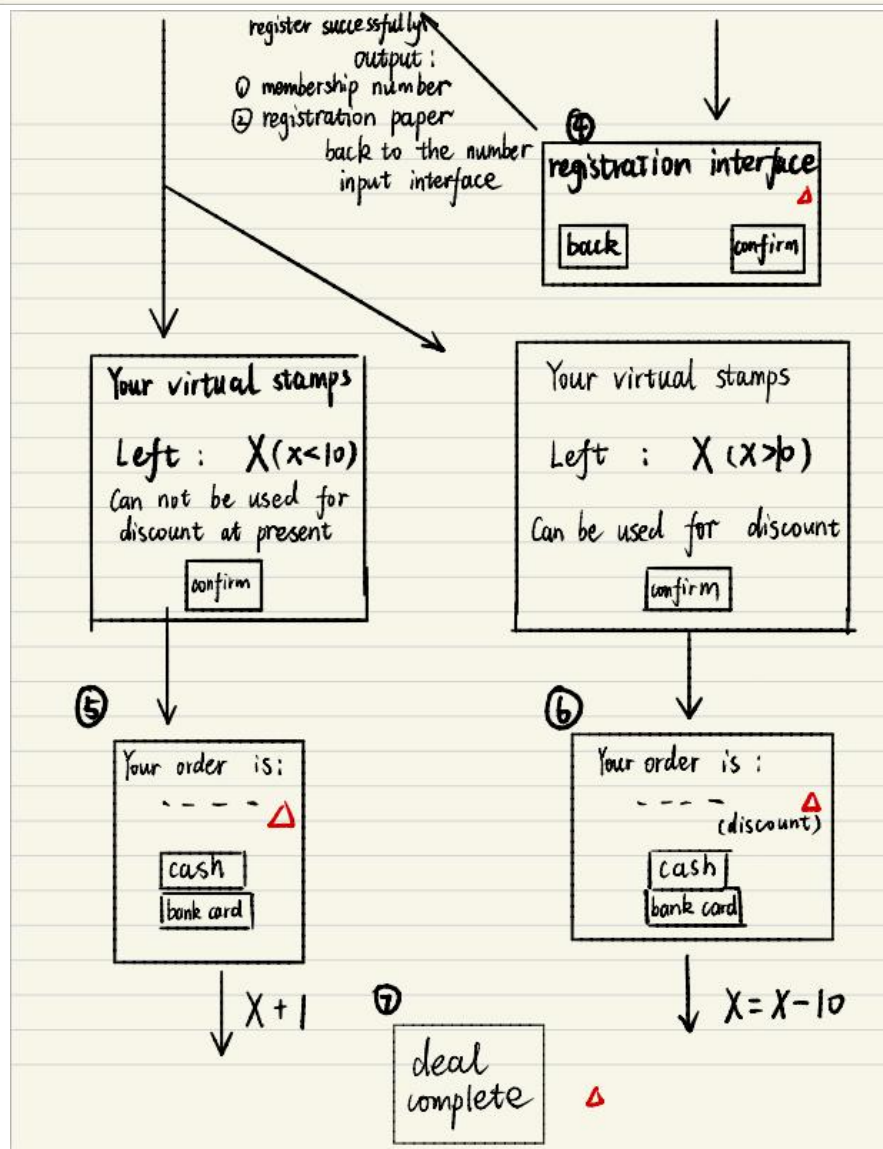
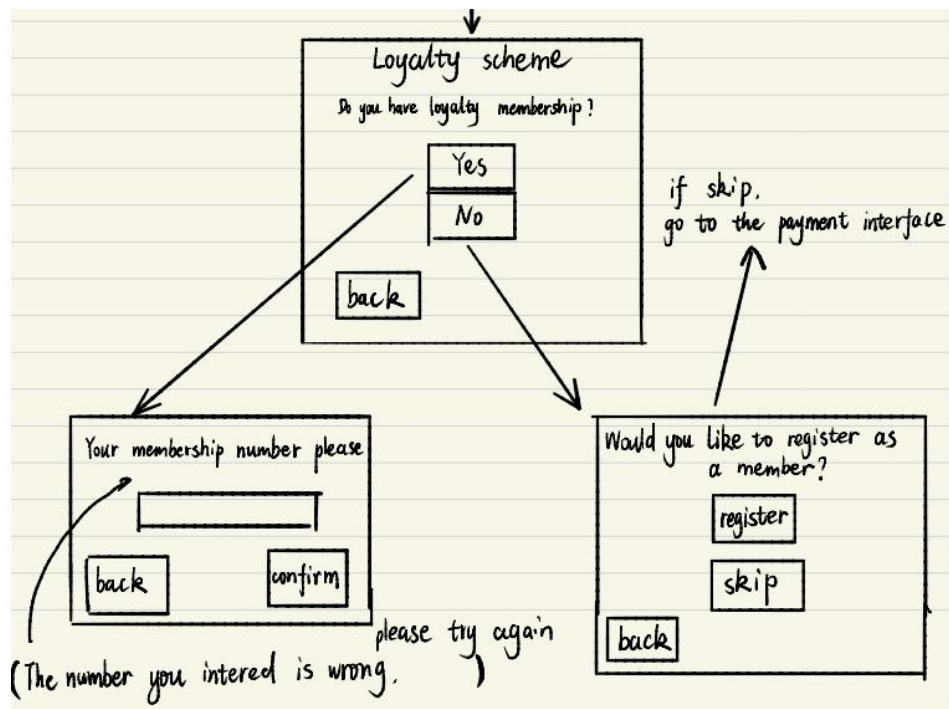
Add statistical functions to the system, and generate several statistic image for better user experience. In addition, do the UI optimization.

3.5 Paper prototype

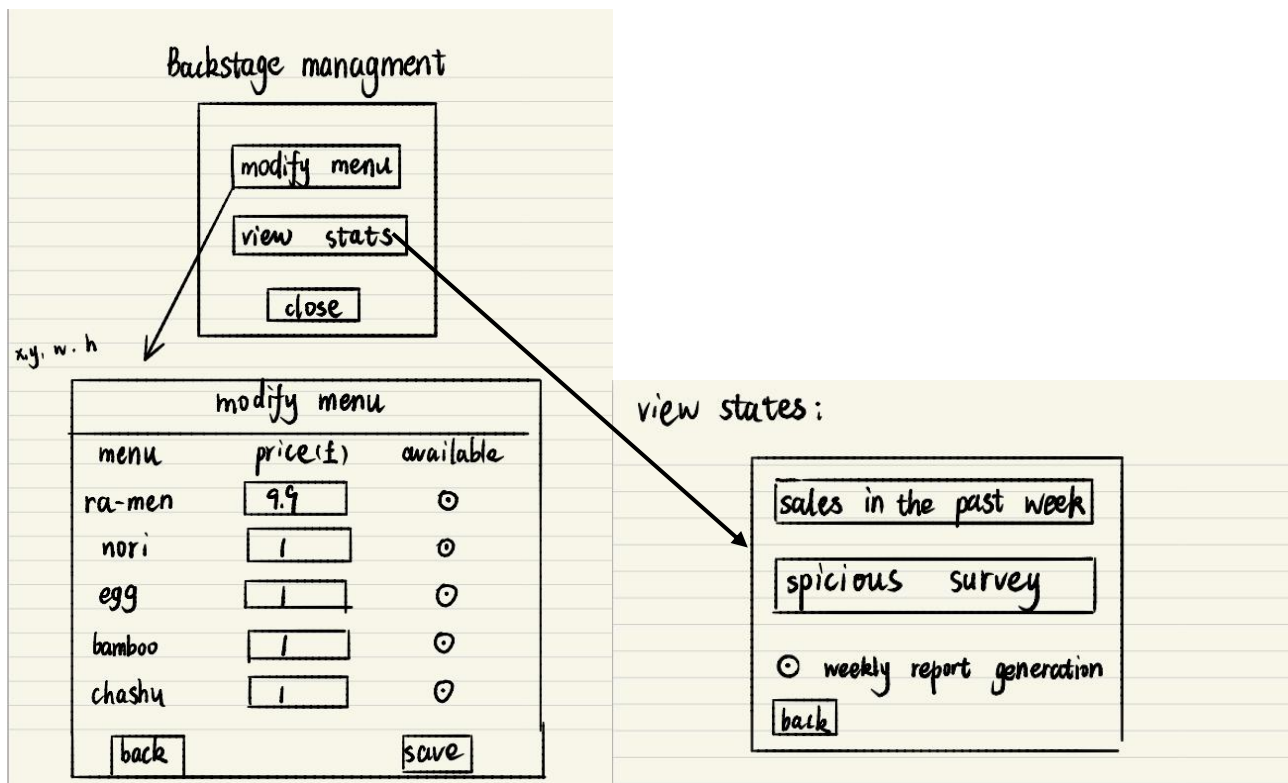
Full picture is shown in the “doc” file

➤ **order system**





➤ Management system



3.6 Adapt to changes

In every iteration, we will recap the system design and reanalyze the requirement from the user. Because we have well designed on the system. If the requirement changes, we can quickly adjust the requirement, add some user stories, postpone some unnecessary functions and plan a new schedule. If the changes is so big that all the design need to be changes, we will do the refactor to adapt the new requirement.

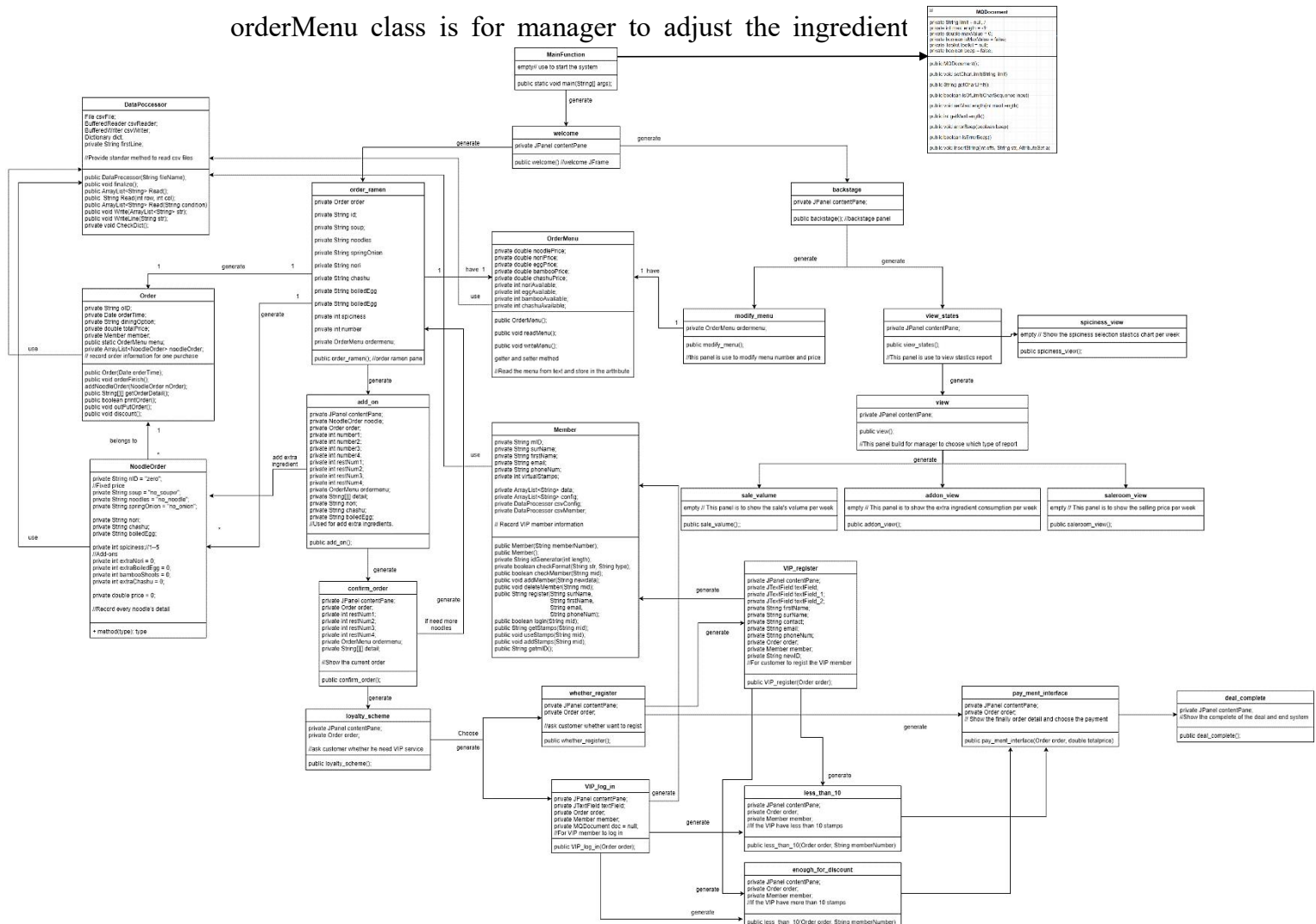
4. Analysis and Design

4.1 Overall design

Based on the requirements we analyze before, we design our software by using many software design principle and technique. We design each class specifically and separate them in three type boundary, control and entity. The UML diagram with all classes is shown in Figure 4. The UML pictures is in \Totoro-Ramen\doc\design doc file.

- Boundary classes: This type of classes contains all the UI classes. These classes get the instructions and input from users and show up the reaction of the program.
- Control classes: This type of classes is used to coordinate the information from boundary class and entity class. The main control unit to start the program.

- Entity classes: This type of classes is the classes we mainly focus on. We designed 5 classes to store information. Order class is to store order information e.g. order number, dining option, total price. From our assumptions, one person can order more than one bowl of noodles, we design a noodleOrder class to store the noodle information from the user. Then, the member class is used for VIP members functions. And orderMenu class is for manager to adjust the ingredient



price and remains. Finally is the dataProcessor class which obtain the method to read and write from the csv files.

Figure 4: UML diagram(Need to zoom in)

4.2 Additional assumptions

In addition to the requirements specified in the coursework question, we also add our own specific assumptions to make the requirements closer to the reality and simpler for user to understand. The first assumption is that one person can order more than one bowl of noodle. It's means that order class can have more than one noodleOrder class. This assumption is more conform to the order system in the real restaurant. Another assumptions is about virtual stamps. Because one order can have many noodles. If one VIP member have ten virtual stamps, from the requirement in the question, he can order as many noodles as he can and don't

need to pay for it. It's a little bit inappropriate to the shop owner. To avoid this happened, we change the requirement that if you have 10 virtual stamps, one noodleOrder in highest price will be free. And customer must consume this discount once they has 10 virtual stamps. He can't remain it is the virtual stamps is enough for discount. This assumption is more appropriate and closer to the reality situation.

4.3 Relationship to the files

To record the information of the order system, we use.csv file to record all the data that need to store. As you can see the right figure. The configure.csv file is use for record generate configuration of the order and member. Material.csv has store all the ingredients' price and remain number for menu modification. Member.csv record all the VIP members' information. Order.csv record all the history order which is prepared for statisitics. The orderForCustomer folder is used to store the bill print which the name is its order number(shown in Figure 5).

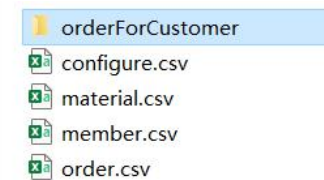


Figure 5: *bill's detail*

4.4 Quality of the code

In our code, we separate our code into very small method that only have one responsibility which is follow the SRP and DRY principle. Each method is very simple and easy to understand. We use these simple methods in the class to achieve our program. This can improve the reusability of our code. In addition, classes are independent with each other. It has high cohesive and loose coupling. We also overwrite a lot of code which the java already have e.g. toString(), equals(object)...

4.5 Extent to meets the main design principles

➤ Single Responsibility Principle (SRP)

Our code strictly follows this principle. We make our classes have low dependency to other classes. All the class has high cohesive and loose coupling.

➤ Open-Closed Principle (OCP)

Our entity class all follow this principle. If the requirement changes, we don't

need to change the code into the class. These class is “open for extension” and “closed for modification”. But the rest of classes don’t quite follow this principle

➤ **Don’t Repeat Yourself (DRY)**

We followed this principle very well in the code. When we realize the code need to use for many place, we will separate it into a simple method to reduce the repeat of the code.

➤ **Dependency Inversion Principle (DIP)**

We are not doing well in this principle. We link the variable directly to the other class without add abstract class. This will cause instability of the program.

➤ **Interface Segregation Principle (ISP)**

Because we don’t use interface to the class. We didn’t use principle in our code.

➤ **Liskov Substitution Principle (LSP)**

We follow the instructions of this principle, and have better performance.

5. Implementation and Testing

5.1 Implementation strategy

To implement the software, we followed some strategies:

- According to the UML class diagram, product backlog and user stories, we distributed our tasks to our group members with the consideration of everyone’s skills.
- We define the accurate relationship between each classes to help our group members fully understand the design and transform the design into code
- We separate the whole system into three subsystems which is VIP member subsystem, order ramen system and backstage management system. We Implement these subsystems in schedule and integrate them into a whole complete system.
- The names of functions and variables with regard to the main functionalities were decided after discussion so that we can read other members’ programs easily.
- We tried to use TDD mechanism in the development of some subprogram.
- We managed our codes with GitHub, which made our implementation well-organized and efficient.

5.2 Iteration plan

The specific iteration plan has been placed in **Section 3.4**. According to staged plans, each group member finished each task and the leader was in charge of integrating all subtasks, where the result of the iteration came out of.

At the end of each iteration, we arranged a meeting that we concluded the

completeness of iteration and discussed what need to pay attention to in next iteration. Reasonably adjust the backlog if there is a delay happen in last iteration.

5.3 Test strategy and test technique

➤ Unit test

Once one of the members complete the code, he will do the unit test before he updates the code to the main branch. If he found mistakes he will fix it right away. We force every member do the unit test before he finish the task and save his test file for further iterations. This test can make sure each task can run itself correctly.

➤ Black box test

We develop all the classes with the black box test including partition test, scenario test and integration test.

For partition test, we test the boundary of the input and record the possible output. Ensure that we would not ignore any type of mistakes. The detailed test information was recorded in the test matrix(see Figure 6). All basic and important functions were tested. And we also checked the methods of the class which the behavior meet their operation description.

For scenario test, we design several scenarios for the whole system and subsystem ,and test each scenario every iteration to make sure all the requirement is meet.

For integration test, we assign a group member to do this test specially. When a new function is integrated, he will do the test and make sure the system is running correctly.

(P.S. phone number length is 11 in China)↵

Test Cases↵	Description↵	Input↵	Expect output Pass/Fail↵
1.1↵	Test that the phone number null is right in the system↵	nothing↵	Fail↵
2.1↵	Test that the phone number length 10 is right in the system↵	1234567890↵	Fail↵
2.2↵	Test that the phone number length 11 is right in the system↵	12345678901↵	Pass↵
2.3↵	Test that the phone number length 12 is right in the	123456789012↵	Fail↵

Figure 6: Test matrix's detail of the user phone number input

➤ White box test

For each iteration, we used basis path testing to make sure all statements and conditions have been executed at least once. The technique we use is the Eclipse's coverage mode. This technique can help us to do the basis path testing

```
public void deleteMember(String mid)
{
    /**
     * delete a member
     * @param mid the member id of delete target
     * @return none
     */
    Iterator<String> d=data.iterator();
    while(d.hasNext()){
        String a=d.next();
        if (a.split(",")[0].equals(mid)) {
            d.remove();
        }
    }
    csvMember.Write(data);

    for(String i:config)
    {
        config.remove(0);
        config.add(i.split(",")[0]+", " + (Integer.toString(Integer.p
    }
    csvConfig.Write(config);
}
```

Figure 7: Eclipse's coverage mode

5.4 The using of TDD

We have tried to use TDD in entity classes. Before we start to write the class code, we discuss what should be test in the TDD and write the possible test class first in the meeting. We use the Junit technique to make this test run. We used TDD so that we wouldn't miss any functionalities and the mistakes could be found

- ▼ TDD
 - > add_onTest.java
 - > BackstagetestTest.java
 - > confirm_orderTest.java
 - > DataProcessorTest.java
 - > deal_completeTest.java
 - > enough_for_discountTest.java
 - > less_than_10Test.java
 - > loyalty_schemeTest.java
 - > MemberTest.java
 - > modify_menuTest.java
 - > MQDocumentTest.java
 - > NoodleOrderTest.java
 - > order_ramenTest.java
 - > pay_ment_interfaceTest.java
 - > viewTest.java
 - > VIP_log_inTest.java
 - > VIP_registerTest.java
 - > welcomeTest.java
 - > whether_registerTest.java

timely.

Figure 8: *TDD classes*

6. Conclusion

In this coursework, every members in our team show the excellent talent, collaboration capability and the communication skill. We use agile program process to develop a self-service machine system for a ramen restaurant. We meet so many difficulties and we use the technique and knowledge which we learn form the class to figure them out and make the program done. We have learned a lot about the software engineering and many design principles. This coursework help us a lot!

7. Reference

- [1] Martin, Robert, Agile Software Development: Principles, Patterns and Practices (1st edn)
- [2] The Art of Software Testing 3rd Edition by Glenford J. Myers (Author), Corey Sandler (Author), Tom Badgett (Author)

8. Appendix.

Main Screen Shots of the System:




(Welcome page)

Order a bowl of ramen

The price of a ramen is 9.9\$

You can choose the flavor!



Affirmatively Chosen Items

Soup	<input checked="" type="radio"/> Tonkotsu	<input type="radio"/> Shoyu	<input type="radio"/> Shio
Noodles	<input checked="" type="radio"/> Soft	<input type="radio"/> Medium	<input type="radio"/> Firm
Spring onion	<input checked="" type="radio"/> No please	<input type="radio"/> Just a little	<input type="radio"/> A lot!
Nori	<input checked="" type="radio"/> Yes	<input type="radio"/> No	
Chashu	<input checked="" type="radio"/> Yes	<input type="radio"/> No	
Boiled egg	<input checked="" type="radio"/> Yes	<input type="radio"/> No	
Spiciness	<input type="range" value="0"/> 0 1 2 3 4 5 Notice: 0 (No) and 5 (Max)		

Have Completed. NEXT

(ramen order)

Order add-ons for your ramen

You can order add-ons with extra payments.

Optional Items

	Unit Price	Number
Extra Nori	1.0	1 <input type="button" value="-"/> <input data-bbox="997 1435 1096 1471" type="button" value="+"/>
Extra Boiled Egg	1.0	1 <input type="button" value="-"/> <input data-bbox="997 1525 1096 1561" type="button" value="+"/>
Bamboo Shoots	1.0	1 <input type="button" value="-"/> <input data-bbox="997 1615 1096 1650" type="button" value="+"/>
Extra Chashu	1.0	1 <input type="button" value="-"/> <input data-bbox="997 1704 1096 1740" type="button" value="+"/>

Back **Have Completed. NEXT**

(add-on order)

Confirm your order

Confirm your order as follows:

Items	Number	Price
Tonkotsu noodle	1	9.9
—extra Nori	1	1.0
—extra boiled egg	1	1.0
—extra bambooShoots	1	1.0
—extra chashu	1	1.0

Dining options: ☒ Eat In ☐ Take Away

Total Price: 13.9


Add another bowl of noodles


Confirm!

(confirm order page)

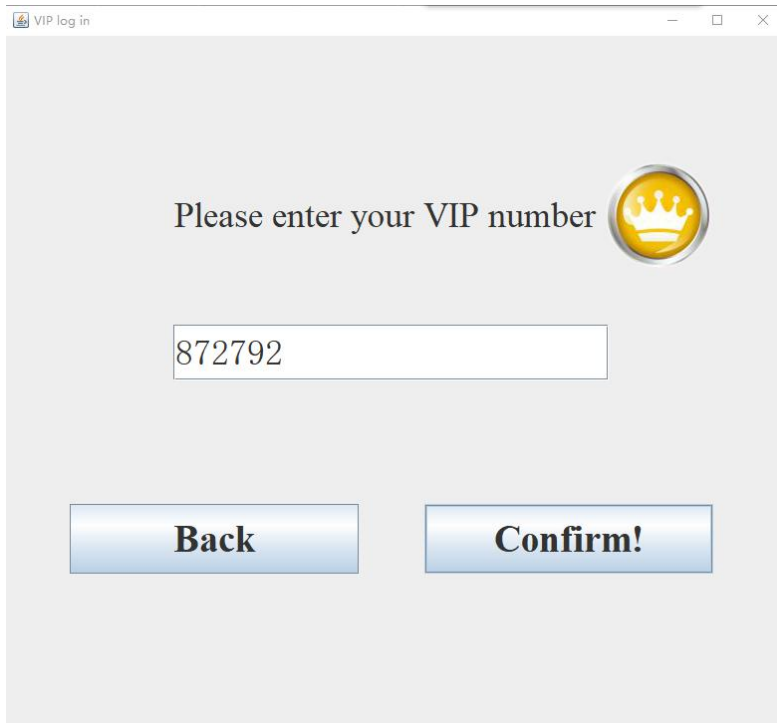
Are you a VIP member?

Are you a VIP member?






(VIP member asking)



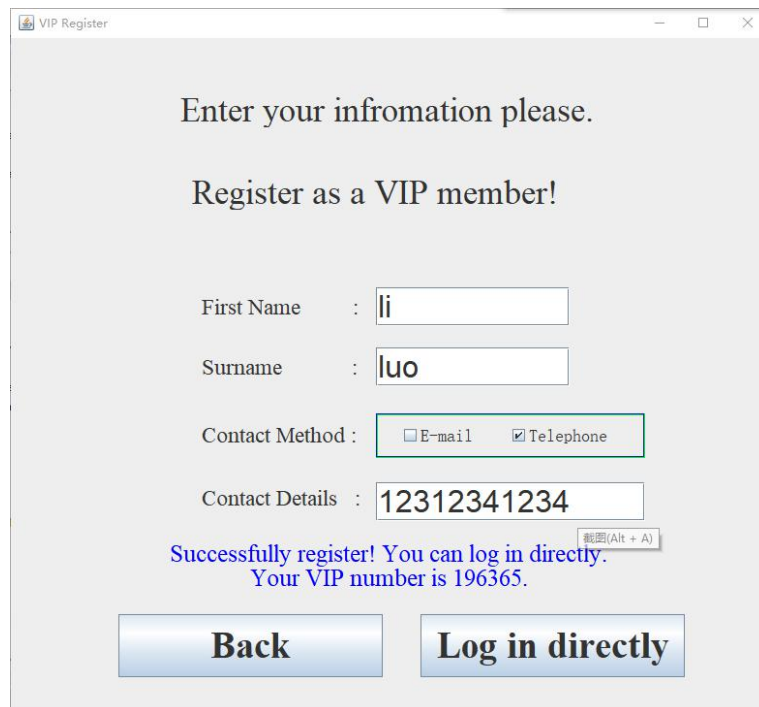
VIP log in

Please enter your VIP number 

872792

Back **Confirm!**

(VIP member login)



VIP Register

Enter your infromation please.

Register as a VIP member!

First Name : li

Surname : luo

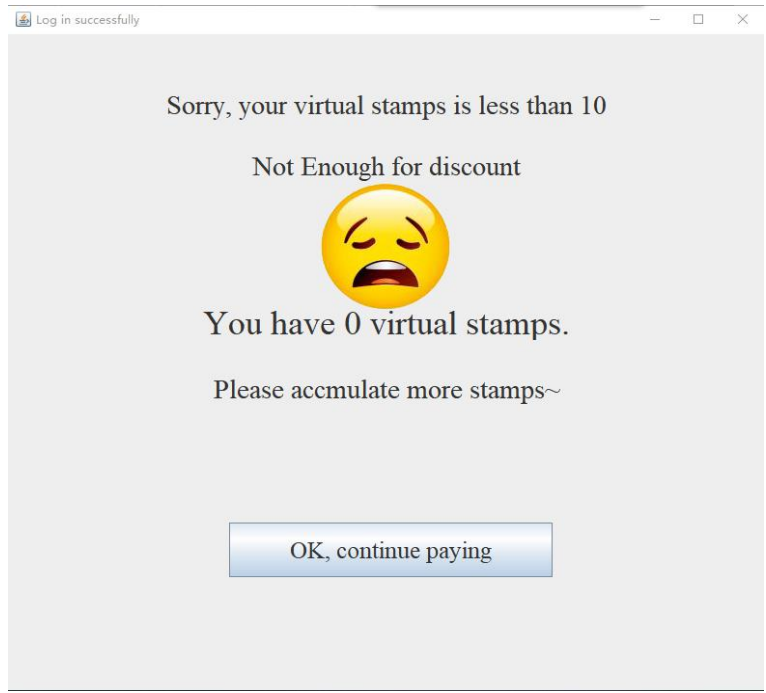
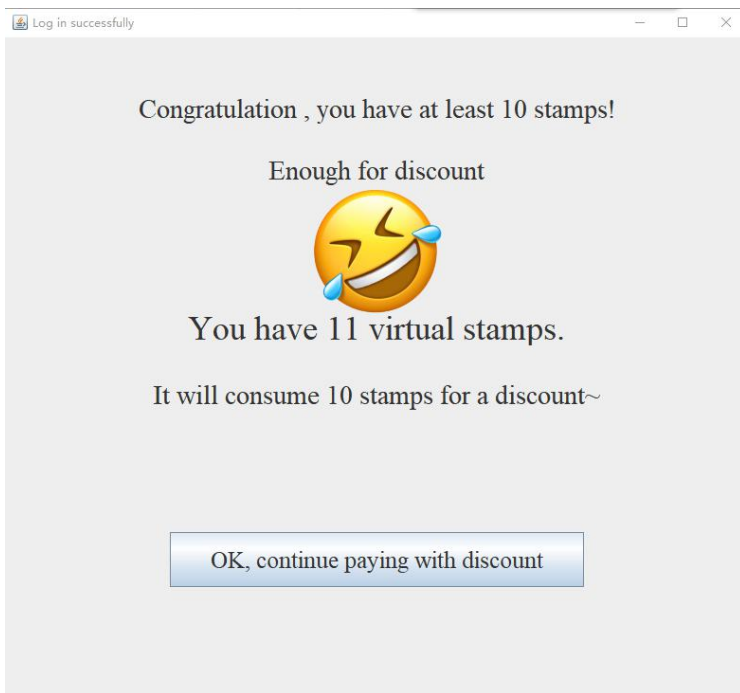
Contact Method : ☐ E-mail ☒ Telephone

Contact Details : 12312341234

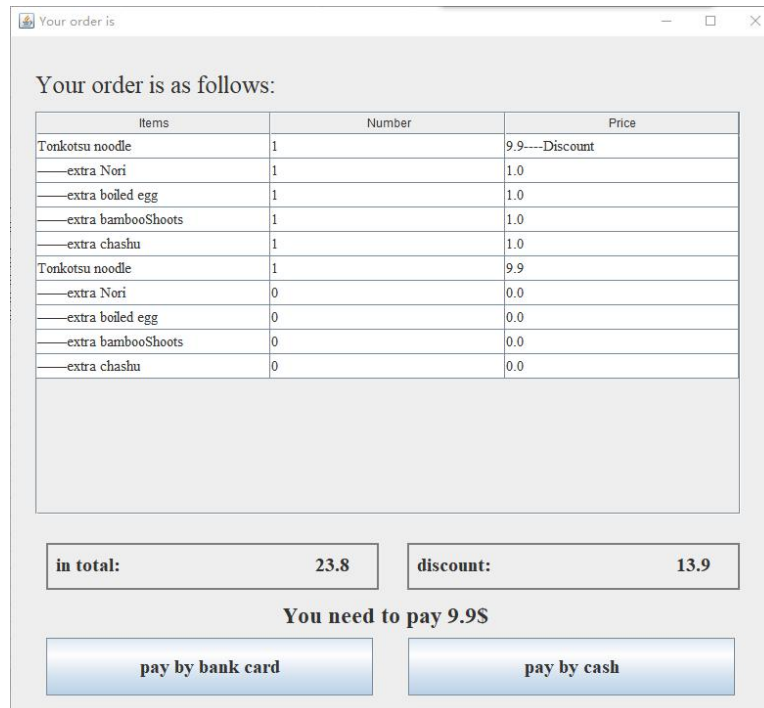
Successfully register! You can log in directly.
Your VIP number is 196365.

Back **Log in directly**

(register)



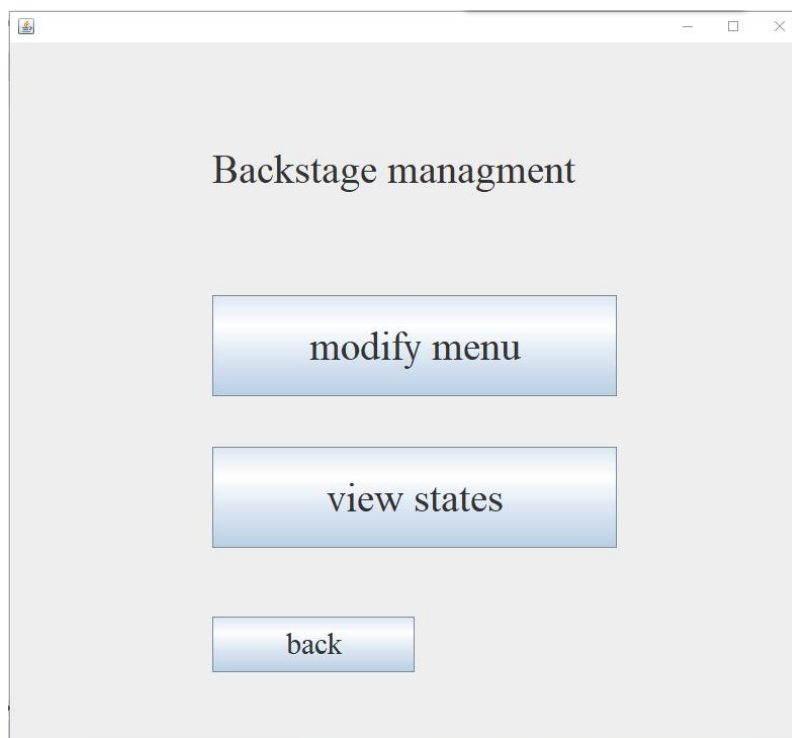
(virtual stamp alert page)



(order payment page)



(Thank you page)

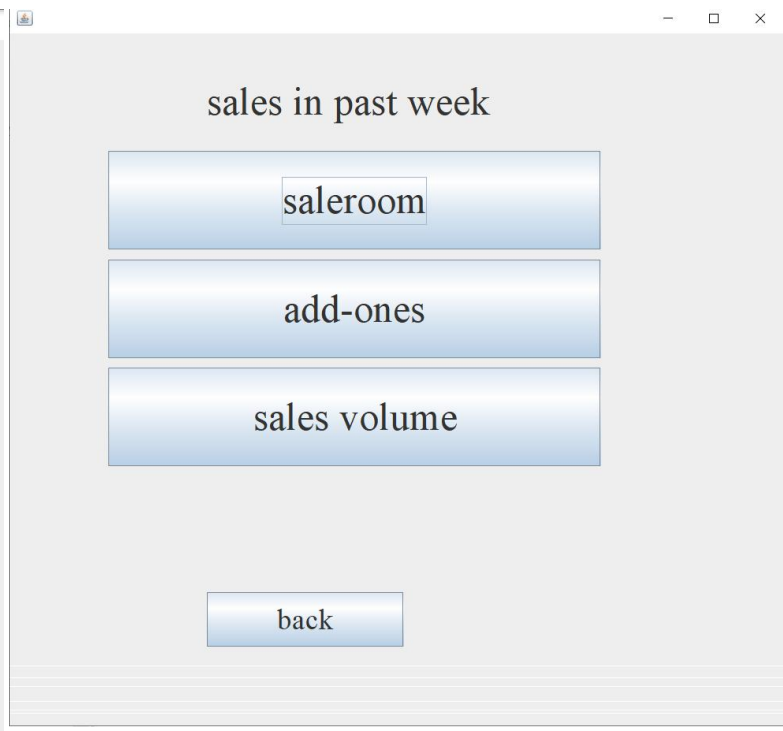
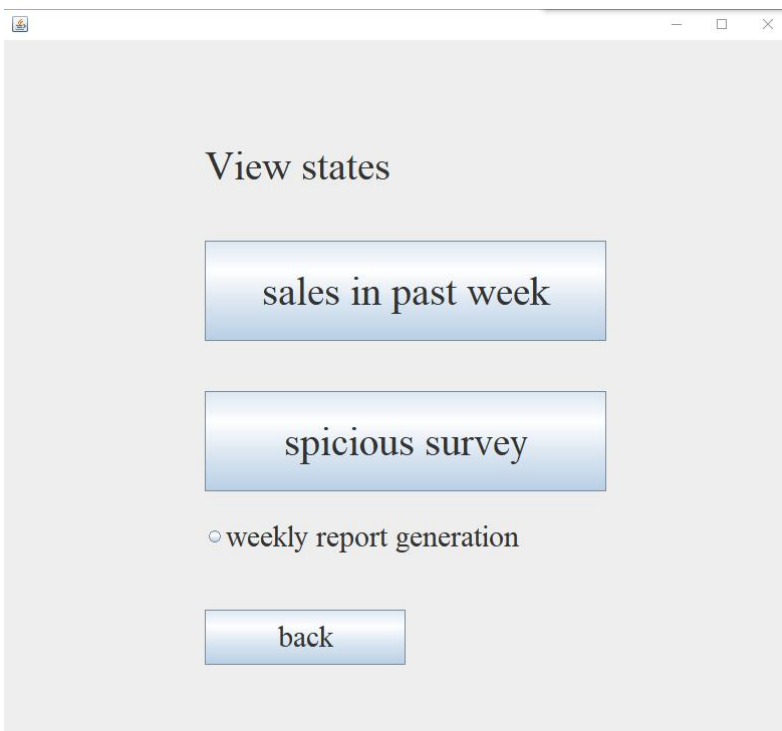


(back stage management option page)

Modify Menu

menu	Price		Stock	
	current	modify	current	modify
ramen	9.9	<input type="text"/>	Absolutely Enough	
nori	1.0	<input type="text"/>	8	<input type="text"/>
egg	1.0	<input type="text"/>	6	<input type="text"/>
bamboo	1.0	<input type="text"/>	3	<input type="text"/>
chashu	1.0	<input type="text"/>	8	<input type="text"/>
<input type="button" value="back"/>		<input type="button" value="save"/>		

(modify menu)



(view states)

