VETERINARY PRACTICE SYSTEM

Object Oriented Software Engineering

Plan a project and apply UML diagrams to the resolution
of the Veterinary Practice

- Semester 1, 2022/23 -

TEAM K

NAKYUNG KIM (21191026)
DONGHYEOK LEE (21234175)
SHINGIRIRAYI MABIKA (21247544)

INDEX :

- Part A -

(Part A was also modified according to a professor's General feedback)

1. Introduction

1.1. Veterinary Practice System

The Veterinary Practice System is a web-based software that helps to store and manage information and procedures that employees must deal with and keep track of the customers, appointments, procedures, and treatments. The scope of veterinary system activities is for safety, quality, and efficacy in the clinic. The information in the systems allows the on-going assessments and future reference. The pages are made by HTML, the database is handled by MySQL, the software logic is controlled by PHP.

1.2. Project Goals

The goal of this project is to develop a veterinary practice system, a web-based software in Java, using object-oriented design of the Unified Modeling Language (UML). The UML helps to specify and visualize the software structure that meet our project requirements. Veterinary Practice Systems is a software that manages customer information and maintains records of various customers. The system provides an interface that allows users to manage information about veterinary operations.

2. Use Case Analysis

2.1. Overall Description

"Use cases are used to explain and document the interaction that is required between the user and the system to accomplish the user's task" (Dennis et al p. 147, 2015).

This software's actors are receptionist, customer, nurse, and veterinary. The customer who is actor is specialized into existing customer and new customer. Some major use cases are log-in system, create, modify, and manage of customer or employee information, create, and manage treatment record and make payment.

2.2. Use Case Diagram

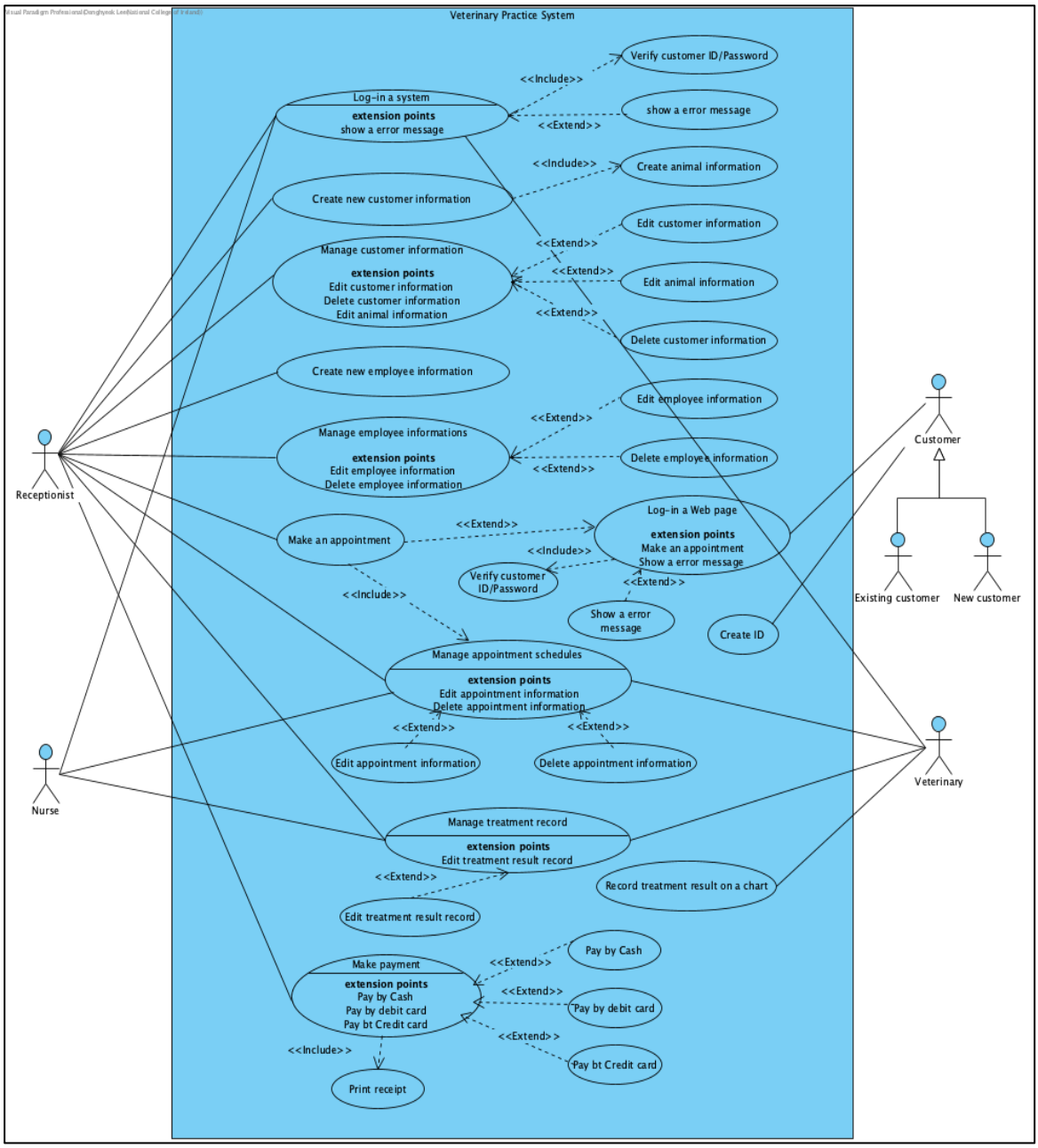


Figure 1. Veterinary Practice System Use Case Diagram

2.3. Use Case Specification

1) Manage information

<table>
<tr><td>Use Case Name: Manage information</td><td>ID:UC-1</td><td>Priority: High</td></tr>
<tr><td colspan="3">Actor: Receptionist</td></tr>
<tr><td colspan="3">Description:
This system allows to user Create and manage their customer information.
[Create new customer information: register new customer information in the system.
Create animal information: register new customer’s pet information in the system when receptionist create new customer information.
Modify customer information: modify existing customer’s information. Ex) Update information of owner, add description for customer s, etc..
Delete customer information: Delete existing customer’s information.
Modify animal information: modify existing pet’s information. Ex) add another pets, update information of pets, add description for pet, etc..]</td></tr>
<tr><td colspan="3">Trigger: Receptionist would like to manage an information.</td></tr>
<tr><td colspan="3">Preconditions:
1. There isn’t any issue in hardware.
2. All employee log-in their account correctly.</td></tr>
<tr><td colspan="2">Normal Course:
1. Receptionist creates new customer information.
2. Receptionist inputs requested information in the system.
3. Receptionist saves and create the customer information.
4. Receptionist creates new animal information.
5. Receptionist inputs requested information in the system.
6. Receptionist saves and creates the animal information.
7. Receptionist manages customer information in the system.
8. Receptionist modifies customer information.
9. Receptionist modifies customer’s information and save.
10. Receptionist searches animal information in the system.
11. Receptionist modifies animal information.
12. Receptionist modifies animal’s information and save.
13. After save, other actors (Veterinary, nurse) are enable to access these information.</td><td>Information for Steps:</td></tr>
<tr><td colspan="2">Alternative Course:
1. System error is occurred.
1a. Receptionists Copies and pastes the customer information. And save in other tool (word, notepad), after changed the information on the system when it fixed.
2. Receptionist Inputs existing customer’s information in create customer information. (duplicate information)
2a. Receptionist modifies existing customer's information.</td><td></td></tr>
<tr><td colspan="3">Postconditions:
1.Both of information is saved in the server.</td></tr>
<tr><td colspan="3">Exceptions:
1. New customer data are incomplete.
4. Animal’s species category doesn’t exist in the system.
4a. Receptionist selects any specie and record the description and save.
4b. Developer adds category in the system.
4c. Developer modifies category and erase the description.</td></tr>
<tr><td colspan="3">summary</td></tr>
<tr><td>Inputs</td><td>Source</td><td>Outputs</td><td>Destination</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

2) Make an appointment by customer

Use Case Name: Make an appointment	ID: UC-2	Priority: High
Actor: Customer		
Description: It shows a list of all available veterinarians and allows customers to make appointments to the veterinarian they want. The appointments can be booked, changed and deleted by 24 hours before the time of the treatment.		
Trigger: Customer would like to make an appointments through veterinary web page.		
Preconditions: 1. Customer logged in appointment manage system of the software.		
Normal Course: 1. The system displays the functions available to the customer on the screen (Making, change and delete Appointment). 2. The customer selects 'Making Appointment'. 3. The system displays a list of veterinarians available for appointment on the screen. 4. The customer selects the veterinarian he or she wants. 5. The system displays the available time for appointments on the screen. 6. The customer selects the time. 7. The customer clicks confirmation button of the appointment. 8. The system displays the customer's appointment information and exits.		Information for Steps: List of veterinarians Available time and date Appointment information
Alternative Course: 1.1. Accessing the system without logging in (Branch at step 1) 1. System shows error message that login is required 2. System forces to move to the login screen. 1.2. If the customer already has a medical appointment (Branch at step 2) 1. System shows the message that there is an existing appointment history. 2. System forces to go to the appointment change screen. 1.3. Exit (Branch at step 1-7) 1. At any time during making the appointment, the customer can exit the system by clicking exit button. 2. System shows a message to confirm whether to save the input data and exit. 3. System shuts down.		Existing Appointment information
Postconditions: 1. The appointment schedule is created.		
Exceptions: 1.1. No veterinarian available for appointment (Occurs at step 3) 1. The system shows the message 'No veterinarians are available'. 2. The system shuts down. 1.2. When the system is closed (24 hours before the time of the treatment) while using the reservation system (Occurs at step 3-6) 1. When approaching within 24 hours of treatment, messages are displayed 30 minutes, 10 minutes, and 1 minute before the system is closed. 2. When the deadline is reached, the system asks the customer whether to proceed with the new reservation or to shut down the system. 3a. The customer requests a new reservation. 4a. The system is re-run from Step 2. 3b. Customer requests shutdown. 4b. The system shuts down.		

summary

Inputs	Source	Outputs	Destination
List of veterinarians Available time and date	Appointment datastore Appointment datastore	Appointment information	datastore

3) Make an appointment by receptionist

Use Case Name: Make an appointment	ID: UC-3	Priority: High
Actor: Receptionist		
Description: It shows a list of all available veterinarians and allows receptionist to make appointments by customer's request. The appointments can be booked, changed and deleted before the time of the treatment.		
Trigger: Receptionist would like to make an appointments by customer's request.		
Preconditions: 1. Receptionist logged in the system of the software.		

Normal Course:	Information for Steps:
1. The system displays the functions available to the receptionist on the screen (Making, change and delete Appointment).	
2. The receptionist selects 'Making Appointment'.	
3. The system displays a list of veterinarians available for appointment on the screen.	List of veterinarians
4. The receptionist selects the veterinarian by customer's request.	
5. The system displays the available time for appointments on the screen.	Available time and date
6. The receptionist selects the time by customer's request.	
7. The receptionist clicks confirmation button of the appointment.	
8. The system displays the customer's appointment information and exits.	Appointment information
Alternative Course: 1.1. If the customer already has a medical appointment (Branch at step 2) 1. System shows the message that there is an existing appointment history. 2. System ask customer if you want to change existing appointment or not. 3. System forces to go to the appointment change screen. 1.2. Exit (Branch at step 1-7) 1. At any time during making the appointment, the receptionist can exit the system by clicking exit button. 2. System shows a message to confirm whether to save the input data and exit. 3. System shuts down.	Existing Appointment information
Postconditions: 1. The appointment schedule is created.	
Exceptions: 1.1. No veterinarian available for appointment (Occurs at step 3) 1. The system shows the message 'No veterinarians are available'. 2. The system shuts down.	

summary

Inputs	Source	Outputs	Destination
List of veterinarians	Appointment datastore	Appointment information	datastore
Available time and date	Appointment datastore		

4) Log in

Use Case Name: Log in	ID: UC-4	Priority: High
Actor: Receptionist, Nurse, Veterinary Doctor		
Description: User will be prompted to login with their Veterinary System account information before they can use the system. Therefore, the user has to have login details.		
Trigger: Customer would like to access to veterinary web page.		
Preconditions: 1. The user has to have a Veterinary System account that they can login with 2. The user tries to log in with their Veterinary System account 3. The user is logged into the system.		
Normal Course: 1.0 Users access the system URL 1. The system prompts the user for their Veterinary System account to put in credentials 2.The user enters their Veterinary System using his or her username and password. 3.The system authenticates the Veterinary System login username and password if it matches. 4.The user gains access to the systems functionality.		Information for Steps:
Alternative Course: 1.1 Invalid Veterinary System account user or password 1.User already logged in with Veterinary System account 2. User not registered with the Veterinary System account		
Postconditions: 1. The user is logged in to the system 2. The user has access to the functions of the system		
Exceptions: E1. Incorrect credentials		
summary		

Inputs	Source	Outputs	Destination
Username Password	Generated through registration	Access to system functionality	Veterinary System

5) Log out

Use Case Name: Log out	ID: UC-5	Priority: Medium
Actor: User		
Description: User clicks on “Logout” and their session is terminated.		
Trigger: Customer would like to log out from veterinary web page.		
Preconditions: 1. The user is logged into their Veterinary System account 2. The user has used the system 3. The user no longer wants to be logged in		
Normal Course: 1.0 User is done using the web application 1. The user clicks on the logout button 2. The system logs the user out and invalidates the session 3. The system redirects to the default Veterinary System logout page 4. The user no longer has access to the system’s functionality		Information for Steps:
Alternative Course: 1.1 N/A		
Preconditions: 1. The user is logged out of the system 2. The user no longer has access to the functions of the system 3. If the user wants to use the system again user has to login		
Exceptions : E1. N/A		
summary		

Inputs	Source	Outputs	Destination
Username Password	Generated through registration	Access to system functionality	Veterinary System

3. Class Diagram

3.1. Overall Description

The conceptual class diagram describes the relationship between classes in the system. In the system, key class is customer class. The customer class can make an appointment for animal. The employee in veterinary clinic can find the treat record about the customer's animal and old appointment schedules. When the customer requests to make an appointment, available nurse and veterinarian are listed to the appointment system. After treatment, new treat record is created by nurse or veterinarian and customer is requested to make payment.

3.2. Diagram

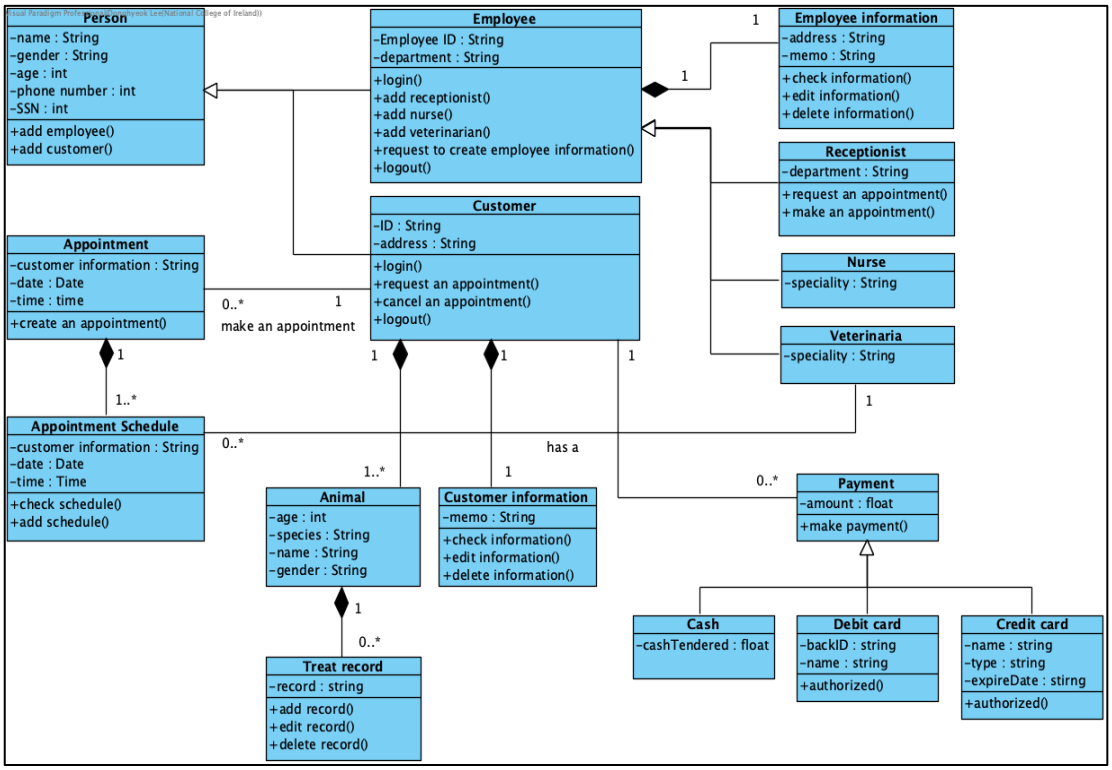


Figure 2. Veterinary Practice System Class Diagram

4. Sequence Diagram

4.1. Create information

1) Diagram

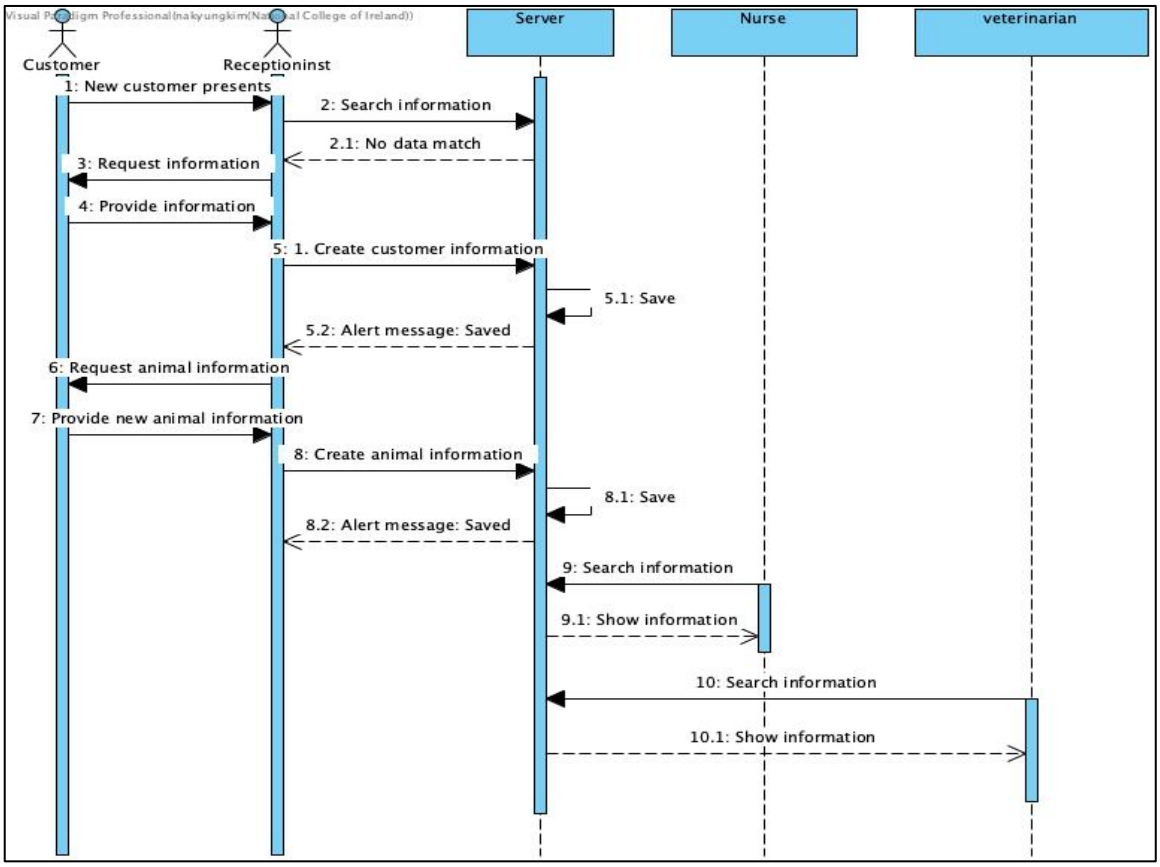


Figure 3. Veterinary Practice System Sequence Diagram – Create information

2) System Behavior – Contracts

Name	Create information
Responsibilities	- Record new customer and animal's information Customer: name, contact, address, date of birth, id, description. Animal: species, name, sex, date of birth, description. - Send this information to system to save.
Type	Management of information system.
Pre-conditions	- Receptionist verifies the customer existing or new. - Receptionist asks compulsory information to new customer. - Server verify the duplicate information.
Post-conditions	
- In case of new customer, Create new information and save. - New customer information is created (instance creation).	

4.2.Manage information

1) Diagram

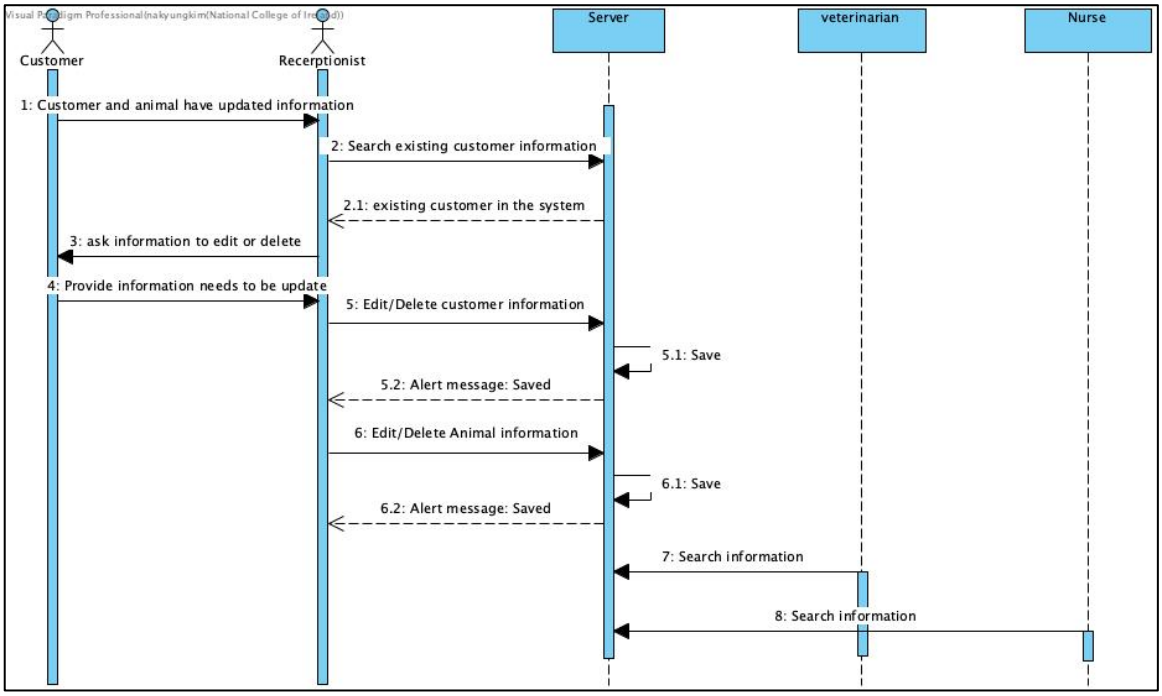


Figure 4. Veterinary Practice System Sequence Diagram – Manage information

2) System Behavior – Contracts

Name	Manage the information
Responsibilities	- Modify or Delete existing customer or animal's information. Customer: name, contact, address, date of birth, id, description. Animal: species, name, sex, date of birth, description. - Modify or delete this information from system.
Type	Management of information system.
Pre-conditions	- Customer states new information to receptionist. - Receptionist update/delete information
Post-conditions	
- Update information will be reflect immediately in the system. - if a new customer, Create new information.	

4.3.Make an appointment

1) Diagram

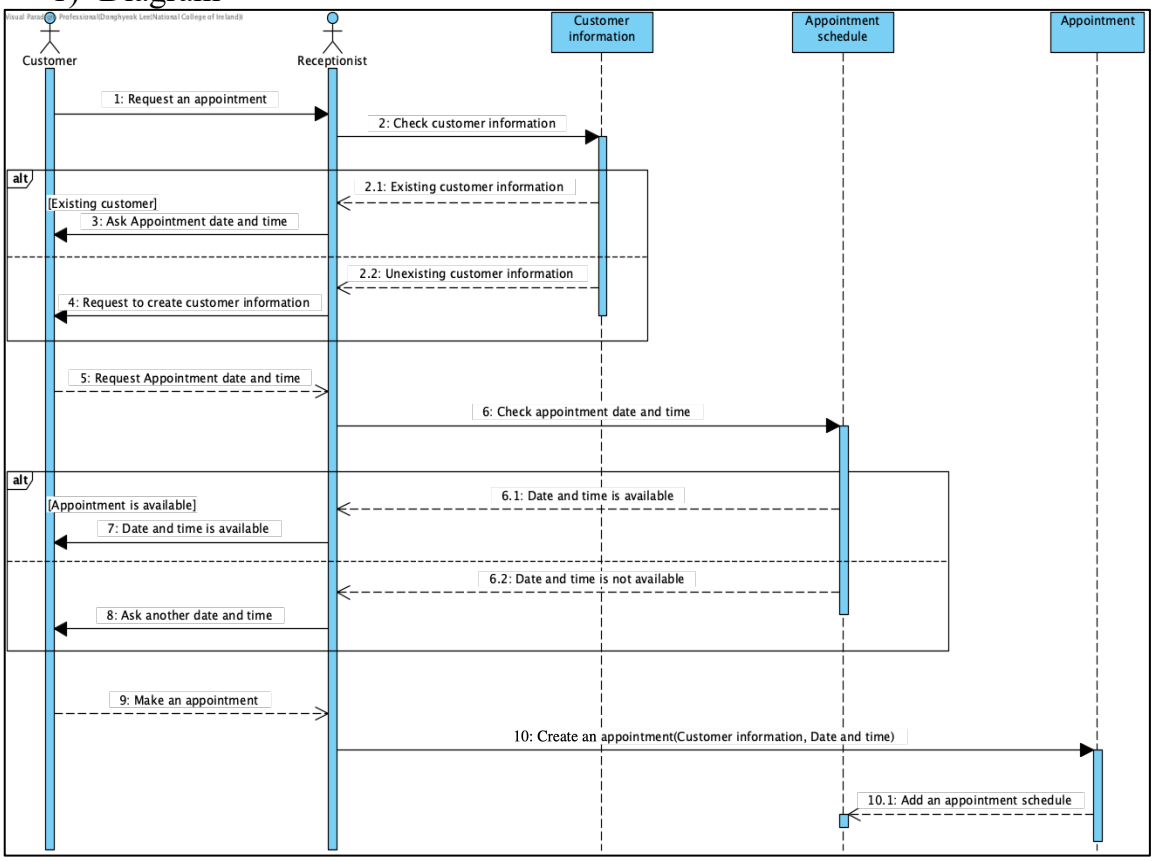


Figure 5. Veterinary Practice System Sequence Diagram – Make an appointment

2) System Behavior – Contracts

Name	Create an appointment
Responsibilities	- Record appointment's information (Customer information, date and time) - Send this information to appointment schedule class
Type	Appointment system
Pre-conditions	- Receptionist asks appointment date and time to customer. - Appointment schedule system confirms appointment information requested by customer is available.
Post-conditions	
- if a new appointment, appointment will be created (instance creation). - if a new appointment, the appointment will be added in the appointment schedule system. - New appointment schedule is created (instance creation).	

4.4.Log in

1) Diagram

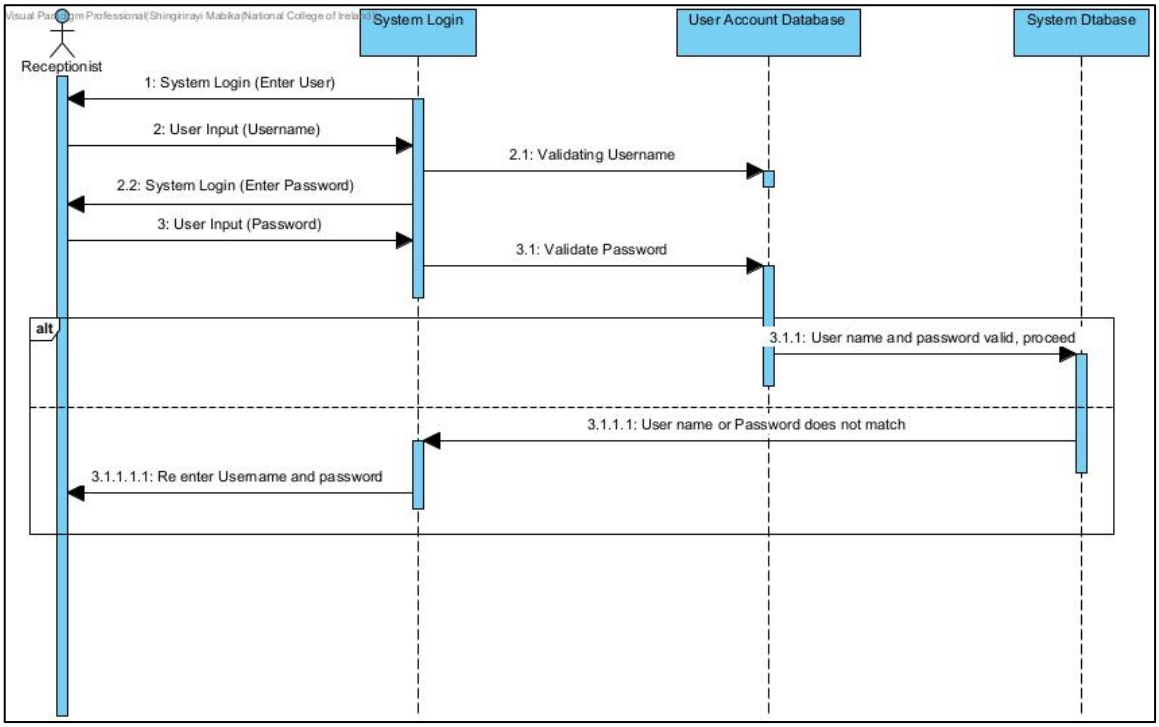


Figure 6. Veterinary Practice System Sequence Diagram – Log in

2) System Behavior – Contracts

Name	Log in
Responsibilities	Allows user to get in the system using system authentication like enter user name and password
Type	Login System
Pre-conditions	Should have been registered as a user Have Username and Password
Post-conditions	
- Make a booking - Check availability - Check employee on duty	

5. Communication Diagram

5.1.Create information

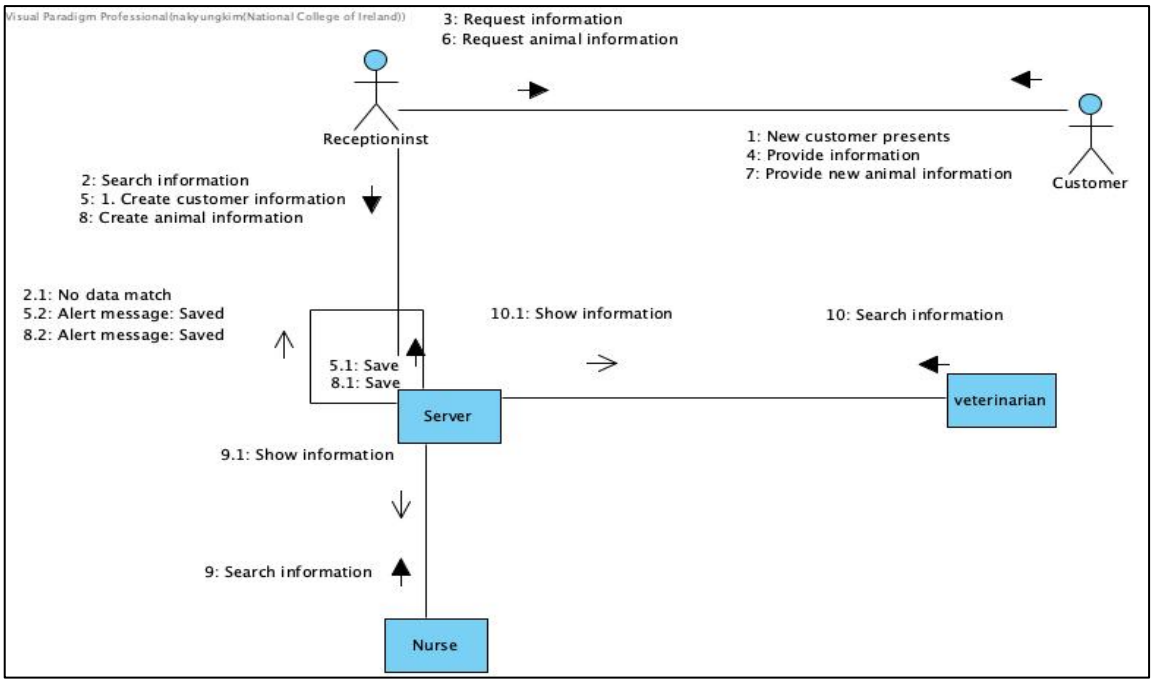


Figure 7. Veterinary Practice System Communication Diagram – Create information

5.2.Manage information

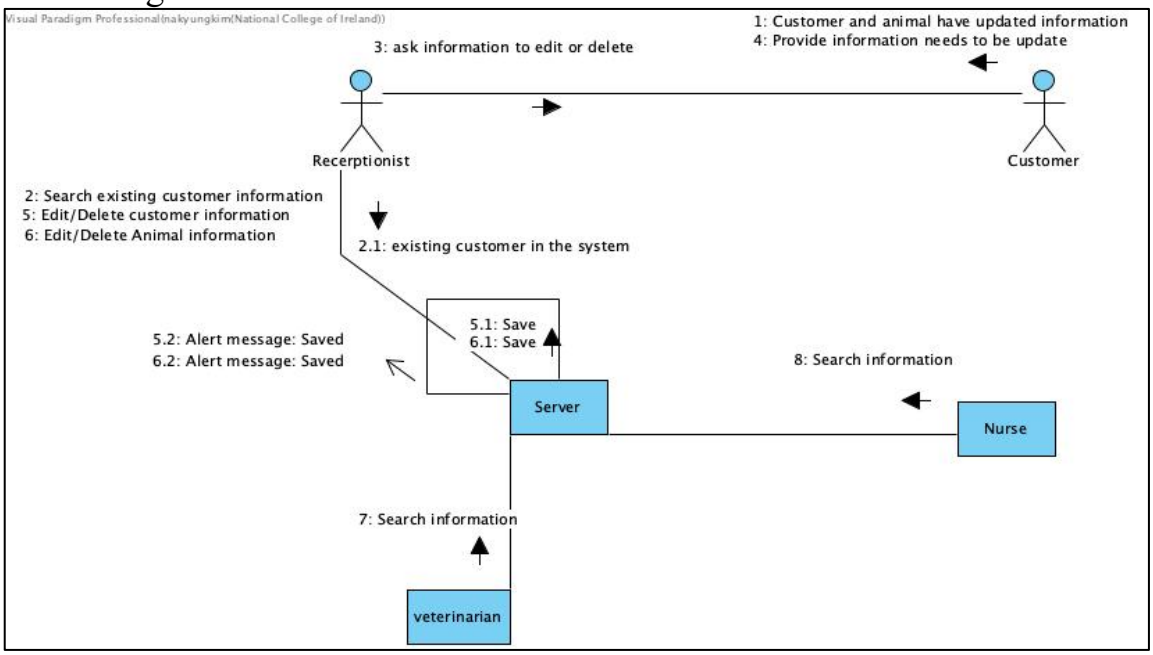


Figure 8. Veterinary Practice System Communication Diagram – Manage information

5.3.Make an appointment

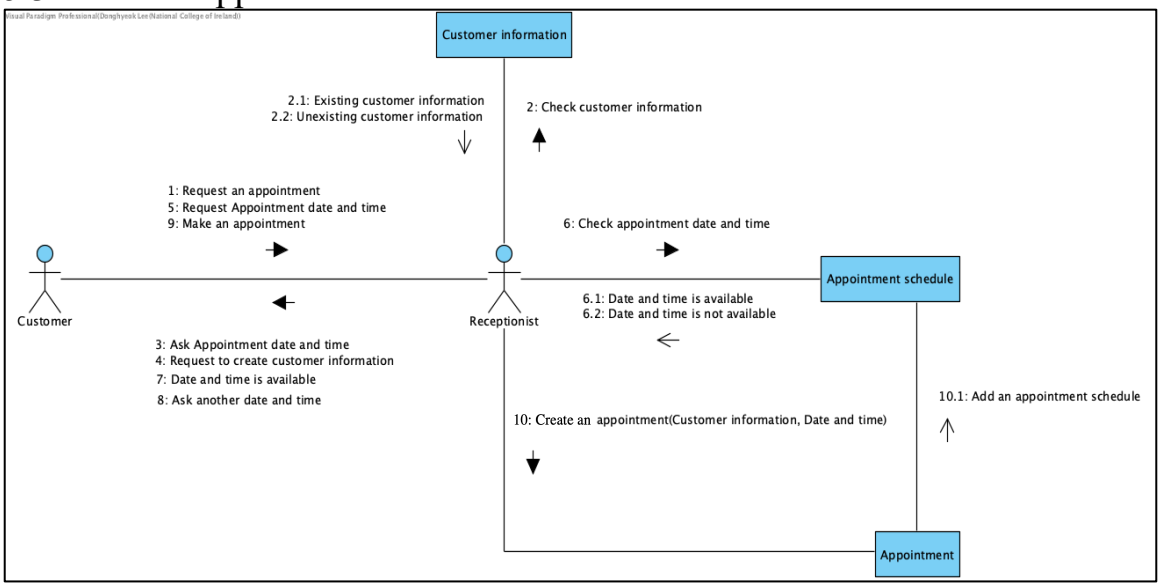


Figure 9. Veterinary Practice System Communication Diagram – Make an appointment

5.4.Log in

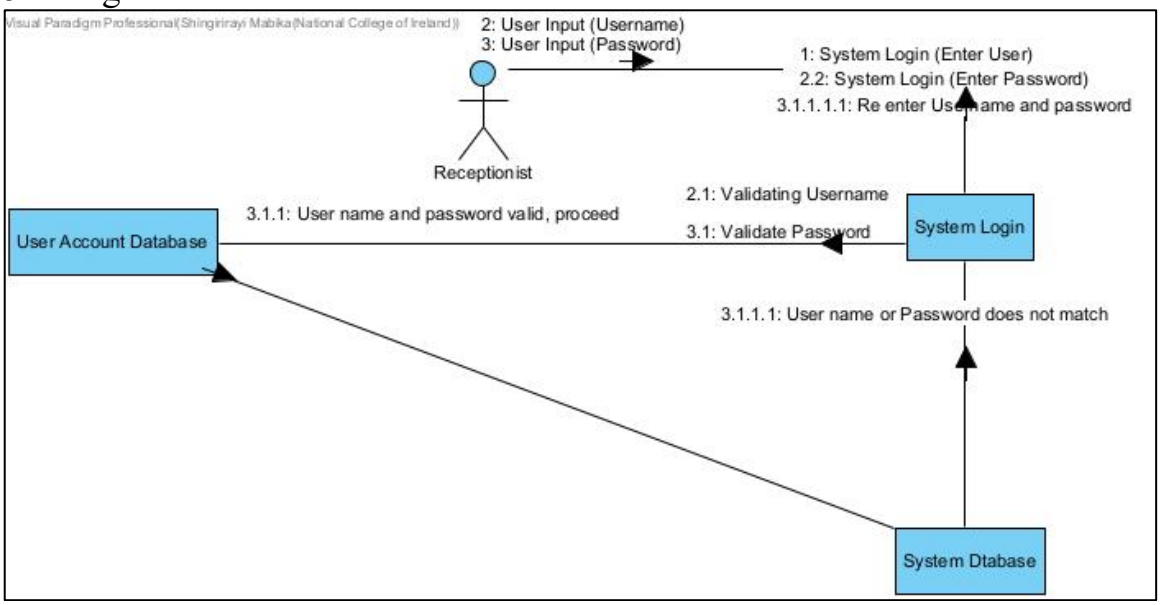


Figure 10. Veterinary Practice System Communication Diagram – Log in

6. Managing risk, quality, and communication in our project.

6.1. Risk and Risk Management

Risk is inevitable and, in all projects, a good project proposal will identify risks before the projects starts, they will try to put measures to cover the risk if it arises. There will always be room for risk to happen. Below will discuss the risks and how they will be resolved in case it happens.

a. Infighting/Conflict

In a project setup members can have different views of the project and the project manager might have a different perspective. That will lead to in fights/conflicts in the project, the Veterinary System implementation is no exception, in fights are prone to happen as members might tend to ignore other's opinion. Therefore, the Project manager and all other stakeholders will need to discuss before the kick of the project and as the project progress so that everyone is on the same understanding. If anyone feels different and puts the view it does not have to be dismissed but it has to be critically analyzed and accessed.

b. Scope creep

According to Dennis et al (p 57, 2012) Scope creep happens when new requirements are added to the project after the original project scope was defined and "frozen." Adding of new features or functions which were not in the initial scope of the project, this will lead to more financial cost and also more time which can even make the critical path longer therefore affecting the time of finishing project. This can be influenced by Project Manager or Project Sponsor who change often on what they want to do or what needs to be done. To manage this risk the new features being added should not be random and should be carefully analysed to see if it is really worth it to add as this might affect the Critical path of the project and affecting the project time length. There should be a formal procedure to follow if there is need to modify scope, this procedure should be known and followed by all members in the project team. This will delay the project pace of progress.

c. Incompetent staff

The staff might be incompetent due to several issues which can include lack of training and ambiguous requirements. All staff will be receiving adequate training and all additional request which were not on the initial scope will need a SWOT analysis to establish the weakness and threats it might bring to the project completion outcome. Also, all staff will have access to continuously check with the project manager if they are not sure of anything. Incompetent staff will delay progress and processing of the project. The staff need to understand the task so that the project is of high quality and will progress as desired.

d. Voluntary staff exodus

With the better offers in the competitive industry there is high risk of employees leaving voluntarily seeking better working environment and

salaries therefore for this project to be successful and not affected by voluntary staff exodus we shall have project members in pairs in each task that has to be done, also regularly feedback will be given to the superiors on a daily basis. There renumeration for the project will be as competent and attractive as possible to reduce staff exodus. The competent renumeration will be go to retain the desired staff members also to reduce brain drain in the project.

6.2. Quality Management

a. Agile Methodology

Therefore, having identified the above-mentioned risks the Agile methodology to ensure quality is maintained in the project. Agile methodology involve~~d~~ breaking down the Veterinary system project into several phases. It also enables stakeholders to constantly collaborate and discuss progress through the use of Teams and scrums. Scrums will be regular meetings held by the teams so as to get everyone up to date of what is going on the development circle of the project. This methodology allows interaction on all stages and encourages team participation. Expert~~s~~ advise and pointers will be easily available to everyone and discussions are ~~done~~ held among stakeholders.

b. System Test

Also, we will do a system test this will be done this is done with the support to the use of the test plan. It will be done as the project runs and will be in the project management plan so that at any given point every stakeholder to the project is sure of what the system offers and understands how the system works.

6.3. Communication Management

a. Training

All members working on the project will have a training program before the project commences. This training will be for everyone participating in the success of the project so that they all understand what will be done in the project.

b. Weekly meetings

There will be online meetings as per Agile method indicated as to maintain good quality and outcome. There will be interaction between project members. The meetings are held virtually and recorded for future referencing so that everyone is clear.

c. Top to bottom instruction

The project Manager will be responsible of giving direction and instruction of what has to be done. This will help manage conflict of ideas and it will know what exactly needs to happen. The project organogram will be available to everyone so that it is known who reports to who.

d. Open door policy

All Project stakeholders have the right to give their views and ideas which can be used in the project for the success of the project however the final decision will be given by project manager after careful analysis.

7. Agile Methodology

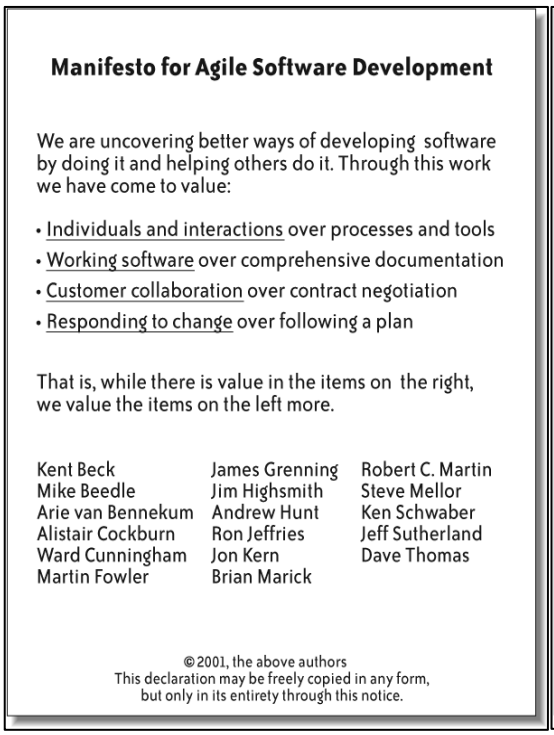
7.1. Justification for Agile Methodology

Agile project management enables cross-departmental teams to handle projects, resolve issues, and accelerate projects in shorter steps. In this way, teams can repeat faster and send and receive updates more often. According to the PWC report, Agile projects are 28% more successful than traditional methodologies. So, we followed Agile methodology, not the traditional methodology, in this project.

7.2. Overall Description

Our project is following Agile Methodology. Agile is a development methodology that focuses more on programming than on development methods that focused on document work and design. Agile is an adjective that means "quick" or "smart." Agile development methodology also follows the original meaning. It means a flexible approach based on the development cycle or software development environment, rather than just following a set plan.

In February 2001, seventeen software developers formed the agile alliance and began discussing faster and more flexible development methods. They have started various activities to define what Agile software development is and to encourage agile software development. At that time, the Agile alliance declared the following. a collection of 4 values (Figure 11-1) and 12 principles (Figure 11-2).



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

Figure 11-1. 4 values of Agile

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity, the art of maximizing the amount of work not done, is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 11-2. 12 principles of Agile.

7.3. Representative types of Agile methodology

1. **Extreme Programming, XP**
2. SCRUM
3. Crystal Family
4. Feature-Driven Development
5. Adaptive Software Development, ASD
6. Extreme Modelling

7.4. Extreme Programming (XP)

1) Overall description

Among them, extreme programming is considered to be the main thing in popularizing Agile development methodology. It can be called XP, it was created by Kent Beck who participated in the Agile Manifesto. Extreme programming considers communication and simplicity important values and is often used in projects where customer requirements change frequently.

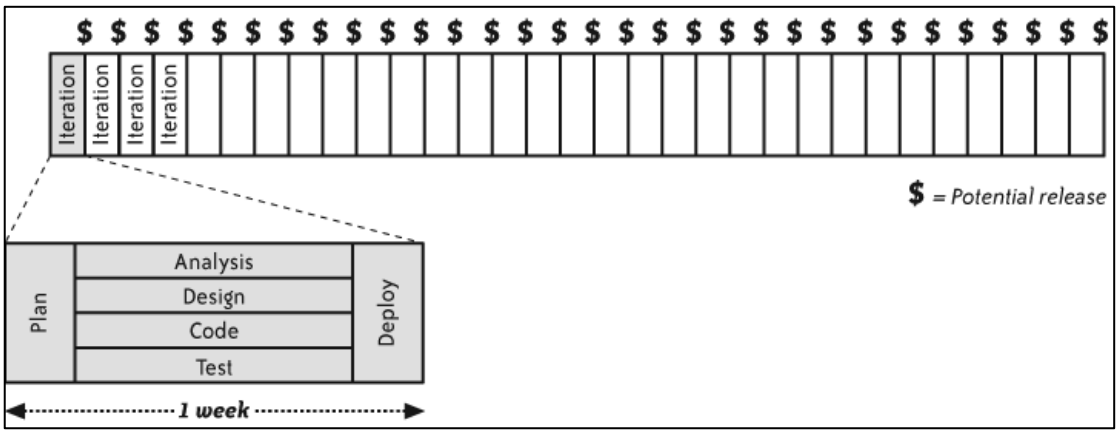


Figure 12. XP lifecycle

Shore and Warden (2014) suggest that Extreme Programming teams conduct almost every software development activity at the same time. Analysis, design, coding, testing, and even deployment perform with fast frequency There are many things to do at the same time. XP works through repetitive tasks (weekly tasks). Every week, the team plans for the release, designs, coding, tests, and more. To review, they distribute software in internal or to the customers.

2) Extreme Programming Core Practices

a. Fine scale feedback
: Two programmers collaborate on a single computer, switching all code coding and review roles (Pair Programming), Plan with players, rules, and goals like a game (Planning Game), Create and perform unit tests before writing actual code, and write code based on them (Test Driven Development), Customer is a project team member for development efficiency (Whole Team).

b. Continuous process rather than batch

: Always keep Deployment Available (Continuous Integration), eliminate complexity, improve communication, simplify, and flexibility without changing features (Design Improvement), Frequent releases at short intervals allow customers to see changes (Small Releases)

c. Shared understanding
   Create source code formats and rules according to standard (Coding Standards), The source code on the system can be modified at any time by any programmer on the team (Collective Code Ownership), Keep Design as Simple as Possible (Simple Design), Describe the structure of the system to be finally developed (System Metaphor)

d. Programmer welfare
   No more than 40 hours of work per week, no two consecutive weeks of overtime (Sustainable Pace)

- Part B -

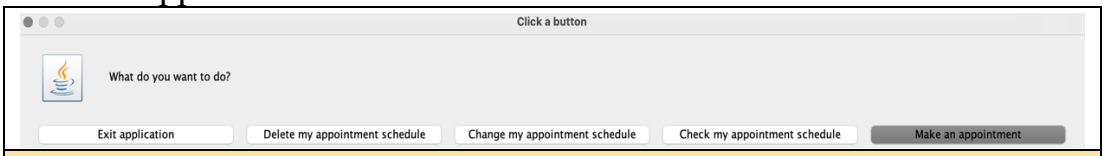
8. Fully develop ‘Make an appointment’ use case

(Source code was uploaded by zip.file on moodle)

8.1. Overall description

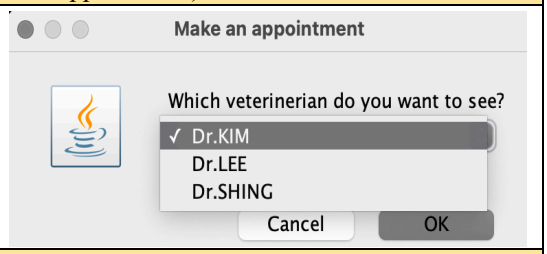
We fully developed the classes required to implement ‘Make an appointment’ use case. In addition, in this development, there are some additional functions (Check my appointment schedule, change my appointment schedule, and delete appointment schedule). Application’s screens and outputs are like below.

8.2. Make an appointment

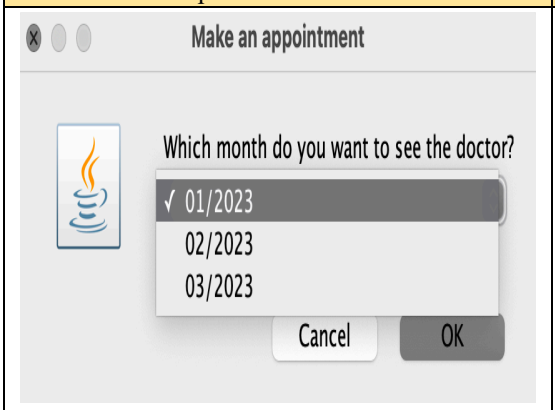


1.Select Menu (Make an appointment)

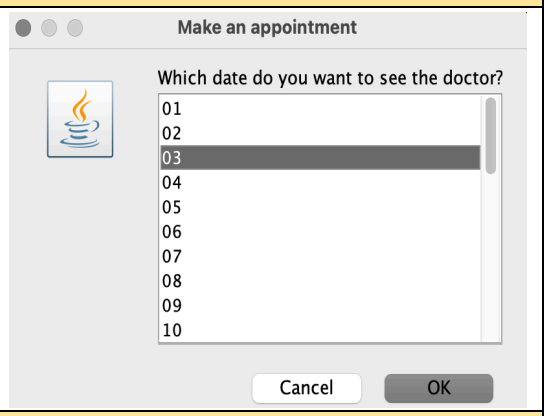
2.Input customer name



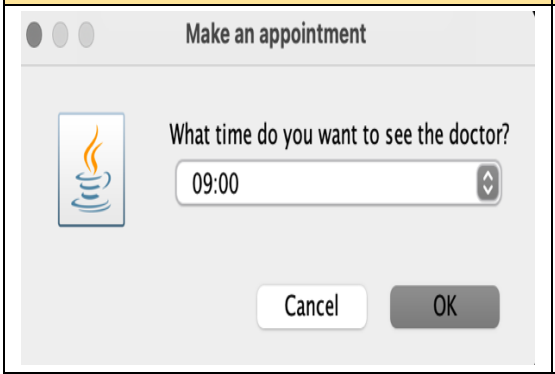
3.Select doctor customer want



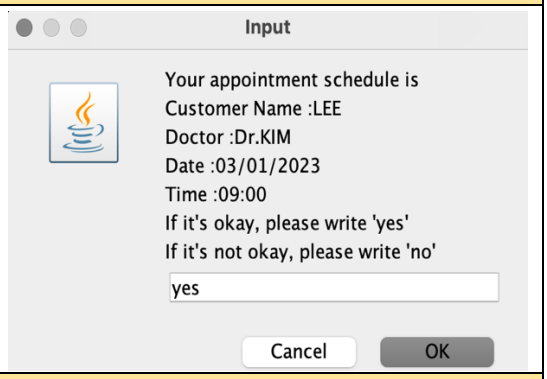
4.Select month customer want to see the doctor



5.Select date customer want to see the doctor

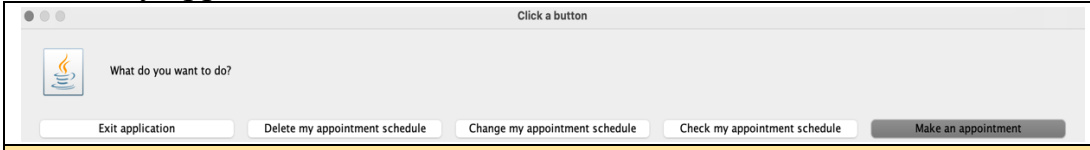


6.Select time customer want to see the doctor

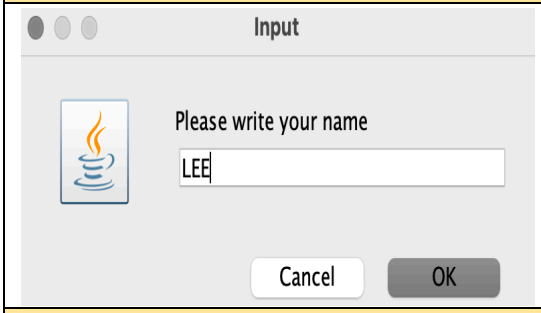


7.Confirm appointment schedule

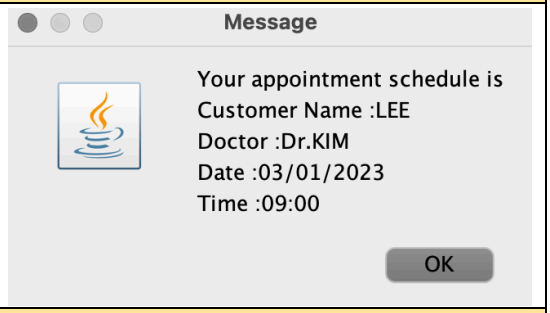
8.3.Check my appointment schedule



1.Select Menu (Check my appointment)

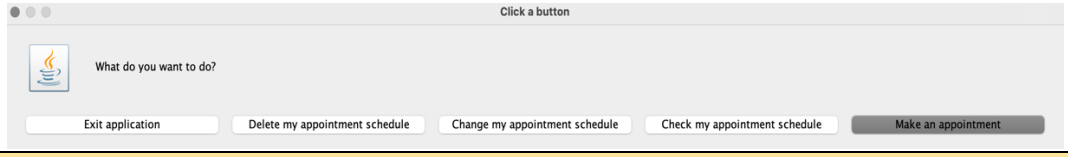


2.Input customer name



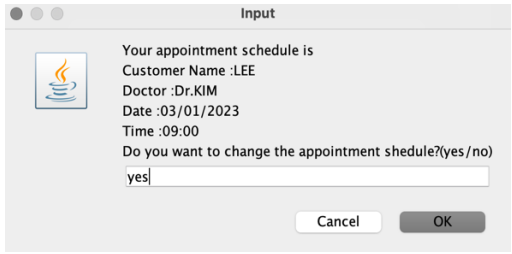
3.Check my appointment schedule

8.4.change my appointment schedule

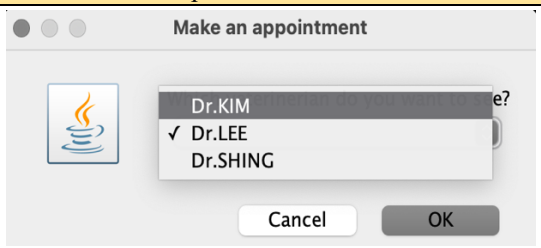


1.Select Menu (Check my appointment)

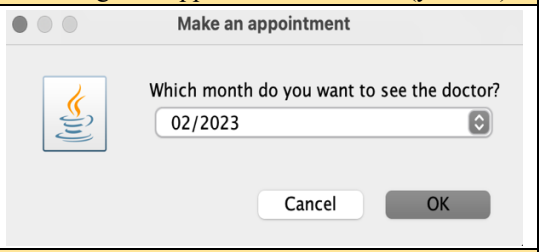
2.Input customer name



3.Change the appointment schedule (yes / no)

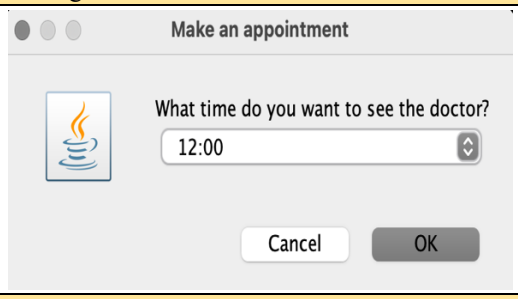


4.Change doctor customer want

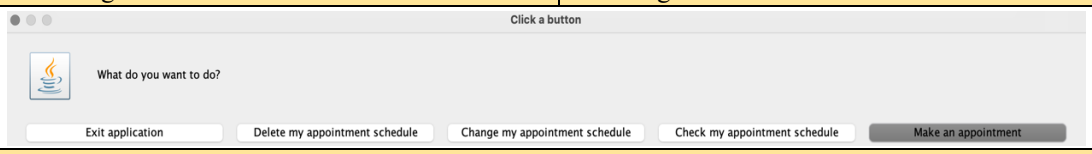


5.Chage month customer want to see the doctor

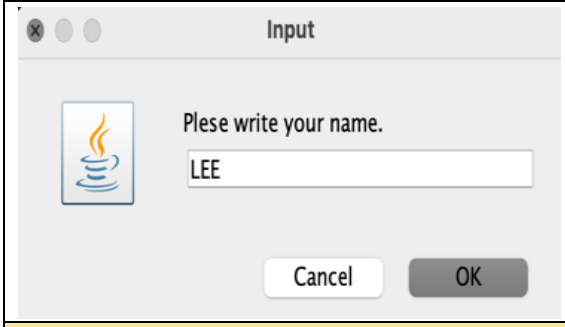
6.Change date customer want to see the doctor



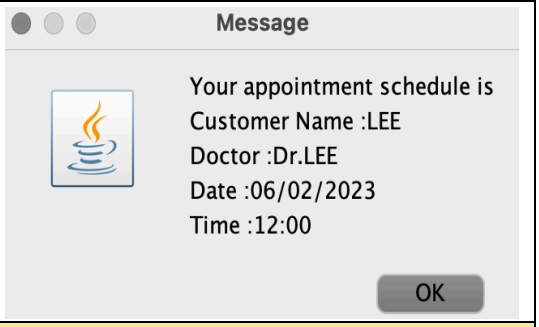
7.Change time customer want to see the doctor



8.Select Menu (Check my appointment)

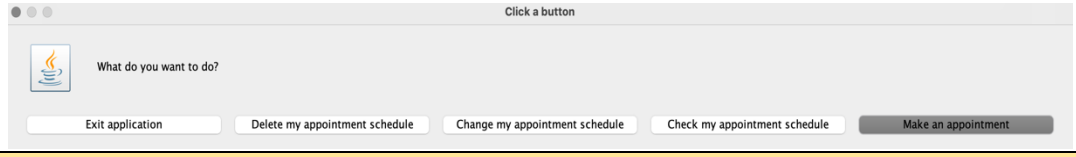


9.Input customer name

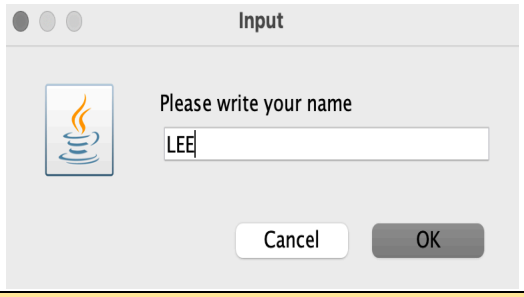


10.Check my changed appointment schedule

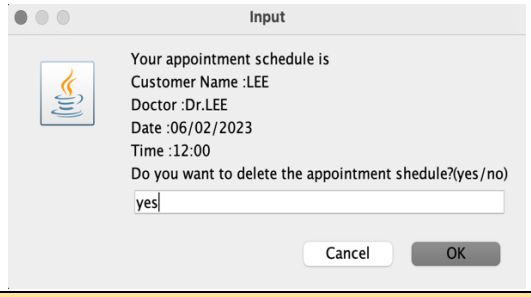
8.5.Delete my appointment schedule



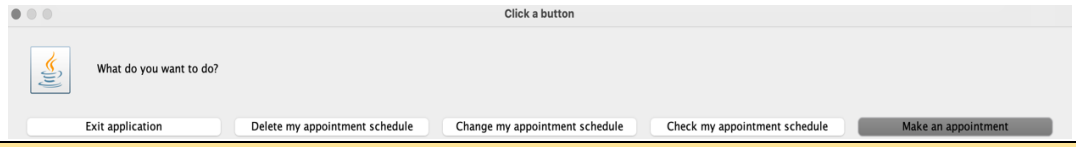
1.Select Menu (Delete my appointment)



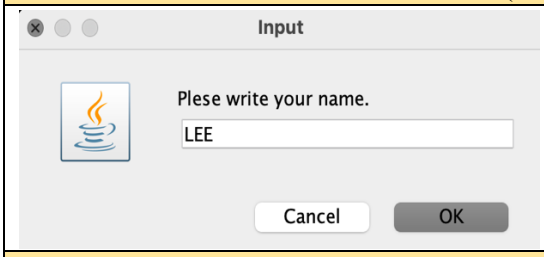
2.Input customer name



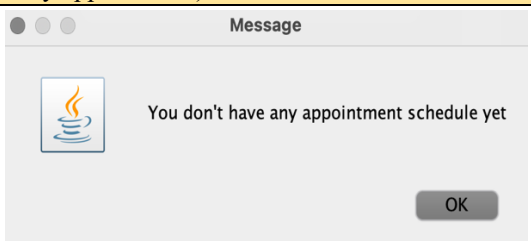
3.Delete the appointment schedule (yes / no)



4.Select Menu (Check my appointment)



5.Input customer name



6.Check my appointment schedule

9. Fully test the classes developed

Our program was fully tested by JUnit. Test screenshots are on **9.2 – 3)** Part.

9.1. Unit test

1) Overall Description

Test conducted to verify the correct implementation of the design and compliance with program requirements for one software element (e.g. unit, module) or a collection of software elements. Procedure for programming to create a test case for all functions and methods and verify that it works as intended. By dividing the program into small units and inspecting whether each unit operates correctly, the stability of the program is enhanced.

The following errors can be found when performing the unit test:

a. Misused data type
b. Incorrect logical operator
c. Undesirable consequences of algorithmic errors
d. Incorrect result from a miscalculated
e. Infinite repetition of loop statement.

2) Test driver and test stub

In the system, many modules are connected by exchanging information with each other. In other words, some modules call modules to be tested, while others to be tested call modules. Therefore, to test one module, it is possible to accurately test it only if all the upper and lower modules directly related to the module exist.

A virtual module that acts as an upper module is called a test driver, and its role is to call the module to be tested. In other words, the necessary data are passed through the factor, and the result value is received after the test is completed.

Conversely, a module that acts as a lower module is called a test stub. A stub module performs the function of carrying out the value received through the factor when the module being tested calls, and then passes the result to the module being tested. Therefore, the driver and stub module are simply implemented to provide only the necessary functions when test.

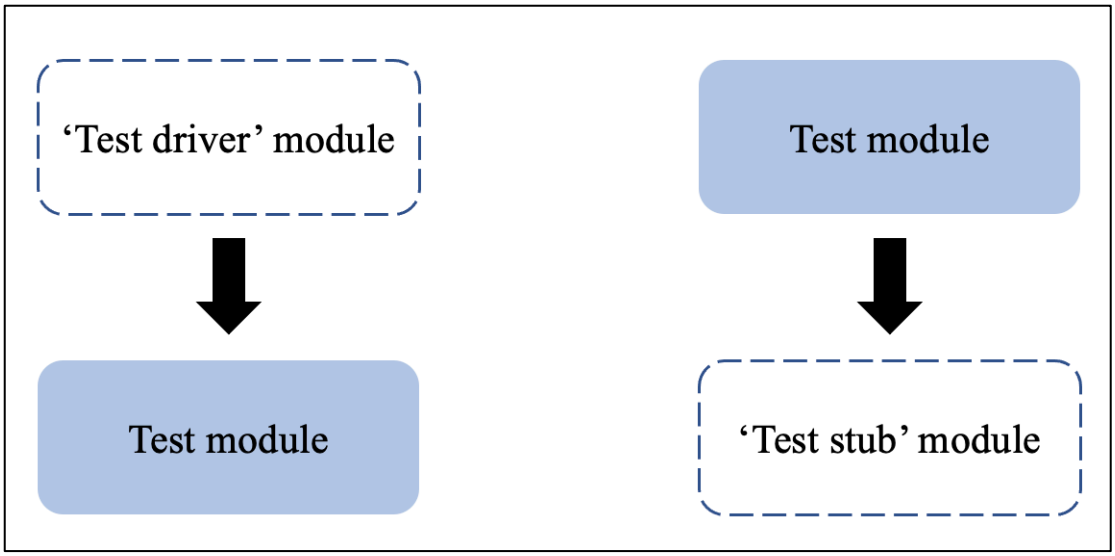


Figure 13.Relationship between test driver module, test stub module, and test module

3) Method of performance

a. Black box test

Black box test is a software test method that inspects the operation of a software without knowing its internal structure or operating principles.

Because the test is conducted mainly by entering the correct input and the wrong input to determine the correct output, information on the code, internal structure and development know-how of the software is basically not required for the test.

What is needed is external facts, such as features, requirements, and blueprints published for test, and the test is conducted focusing on features and requirements of the software, such as "What role should this software play. The test can be related to functions or to non-functions.

b. White box test

White box test is a method to test the source code inside the software source code. Software test methods include black box test and white box test. Black box test is a test method that checks the input and output values without looking inside the software and determines the validity of the function. The reason for conducting the white box test is that developers can track the process of internal source code, so they can examine not only the effectiveness of the process but also the procedure in which it is executed, unnecessary code or untested.

The white box test usually requires checking the execution path of the code, so many analysis tools on the market are used. White box test requires a lot of time and analysis compared to black box test, but they can be used very well to determine the location of defects where errors occur.

4) Advantage of the unit test

Since the unit test is conducted independently of the corresponding parts, it is possible to quickly check if which code is a problem.

a. Save time and money on testing.
b. It can be tested as soon as new functions are added.
c. Stability can be ensured during refactoring.
d. It can be a document about code.

9.2. Junit Test

1) Overall Description

JUnit is a unit test framework for Java programming languages. JUnit is important in terms of test-driven development. In programming at each given stage, we can realize we would like to add a code so that the program becomes better in functionality then the testing is re done to make sure that the code has not been disrupted and make sure it is working efficiently. Therefore, we adopted JUnit testing in this project.

2) Frequently used JUnit methods

a. assertEquals(a, b); = Verify that the values of objects a,b match
b. assertArrayEquals(a, b);= Verify that the values of array a,b match
c. AssertSame(a, b); = Verify that objects a,b are the same objects and verify that the references of the two objects are the same
d. assertTrue(a); = Verify that the condition a is true
e. assertNotNull(a); = Verify that object a is not null.

3) Junit Test for the classes developed
(Source code was uploaded by zip.file on moodle)

We implemented the Junit test code as below. We used the @BeforeAll and @AfteAll functions to indicate that our Junit test is started. Also used the @BeforeEach and @AfterEach functions to indicate that each method is started. This function contributed to increasing our test efficiency. To give an order to methods, I imported the Orderclass and related class.

Appointment.java | *Testing.java | Reservation2.java

```
1  /*
2   * Team K
3   * Junit code
4   * 21191026 NAKYUNG KIM, 21234175 DONGHYEOK LEE, 21247544 SHINGIRIRAYI MABIKA
5   */
6
7  package test;
8
9
10 import static org.junit.jupiter.api.Assertions.assertArrayEquals;
11 import static org.junit.jupiter.api.Assertions.assertEquals;
12 import static org.junit.jupiter.api.Assertions.assertNull;
13
14 import org.junit.jupiter.api.AfterAll;
15 import org.junit.jupiter.api.AfterEach;
16 import org.junit.jupiter.api.BeforeAll;
17 import org.junit.jupiter.api.BeforeEach;
18 import org.junit.jupiter.api.Disabled;
19 import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;
20 import org.junit.jupiter.api.Order;
21 import org.junit.jupiter.api.Test;
22 import org.junit.jupiter.api.TestMethodOrder;
23
24 import oose.Appointment;
25
26 @TestMethodOrder(OrderAnnotation.class)
27 class AppTest {
28
29     @BeforeAll
30     static void setUpBeforeClass() throws Exception {
31         System.out.println("Application started");
32     }
33
34     @AfterAll
35     static void tearDownAfterClass() throws Exception {
36         System.out.println("Application ended");
37     }
38
39     @BeforeEach
40     void setUp() throws Exception {
41         System.out.println("method started");
42     }
43
44     @AfterEach
45     void tearDown() throws Exception {
46         System.out.println("method ended");
47     }
48
```

Appointment.java

*Appointment.java *Testing.java Reservation2.java

```java
package oose;

import java.util.Arrays;

public class Appointment {

    public String arrCustomerName[] = new String[]{"AAA","BBB","CCC","DDD","EEE"};

    public String arrDoctor[] = new String[]{"Kim","Lee","Shingi","Kim","Lee"};

    public String arrDate[] =  new String[]{"2023-01-04","2023-01-28","2023-01-15","2023-01-07","2023-01-12"};

    public String arrTime[] = new String[]{"14:00","15:00","09:00","10:00","17:00"};


    public Appointment() {

    }

```

In this Unit test class, I created object 'aa' from the Appointment class. When I create the object, The four string arrays with values are created together. It's a kind of database that collected customers' appointment schedules. To obtain the clear results, I created this object in every testing method. Through this object, I will test function of each method.

a. Junit test method 1 - CheckAppointment

```java
    @Test
    @Order(1)
    void CheckAppointment() {
        Appointment aa = new Appointment();

         String arrCustomerName_check[] = new String[]{"AAA","BBB","CCC","DDD","EEE"};

         String arrDoctor_check[] = new String[]{"Kim","Lee","Shingi","Kim","Lee"};

         String arrDate_check[] = new String[]{"2023-01-04","2023-01-28","2023-01-15","2023-01-07","2023-01-12"};

         String arrTime_check[] = new String[]{"14:00","15:00","09:00","10:00","17:00"};

        assertArrayEquals(aa.arrCustomerName, arrCustomerName_check);
        assertArrayEquals(aa.arrDoctor, arrDoctor_check);
        assertArrayEquals(aa.arrDate, arrDate_check);
        assertArrayEquals(aa.arrTime, arrTime_check);

    }
}
```

In this Unit test method, I used a test tag to give an order. This method will be run first. I created the object 'aa' in the first line. After that, I declared four String arrays for the sample which has the same value as an object. For these arrays, I put the '_check' at the end of the name to verify easily. I implemented assertion to verify the value between the object and every sample array using the assertArrayEquals() function.

b. Junit test method 2 – ChangeAppointment

```
70    @Test
71    @Order(2)
72    void ChangeAppointment() {
73            Appointment aa = new Appointment();
74
75          String arrCustomerName_check[] = new String[]{"AAA","FFF","CCC","DDD","EEE"};
76
77          String arrDoctor_check[] = new String[]{"Kim","Shingi","Shingi","Kim","Lee"};
78
79          String arrDate_check[] = new String[]{"2023-01-04","2023-01-30","2023-01-15","2023-01-07","2023-01-12"};
80
81          String arrTime_check[] = new String[]{"14:00","15:00","09:00","10:00","17:00"};
82
83         for(int i = 0; i < 5; i++) {
84             if(aa.arrCustomerName[i] == "BBB") {
85                 aa.arrCustomerName[i] = "FFF";
86                 aa.arrDoctor[i] = "Shingi";
87                 aa.arrDate[i] = "2023-01-30";
88                 aa.arrTime[i] = "15:00";
89             assertEquals(aa.arrCustomerName[1], arrCustomerName_check[1]);
90             assertEquals(aa.arrDoctor[1], arrDoctor_check[1]);
91             assertEquals(aa.arrDate[1], arrDate_check[1]);
92             assertEquals(aa.arrTime[1], arrTime_check[1]);
93
94                 }
95
96             }
97
```

Due to the Order function, this method will be run as a second. I set the same condition as the CheckAppointment method. In this method, I will change the information of object's four arrays. I implemented four loop statement. In this data, if the appointment whose customer name is "BBB", It will change to "FFF". And also the Doctor, Date and Time will be changed. After changing this information, I will check the modification goes right. In this test, I used the assertEquals() Function to verify the value between the modified value and the sample value.

c. Junit test method 3 – DeleteAppointment

```
105   @Test()
106   @Order(3)
107   void DeleteAppointment() {
108       Appointment aa = new Appointment();
109
110       for(int i = 0; i < 5; i++) {
111           if(aa.arrCustomerName[i] == "AAA") {
112           aa.arrCustomerName[i] = null;
113           aa.arrDoctor[i] = null;
114           aa.arrDate[i] = null;
115           aa.arrTime[i] = null;
116           }
117
118           assertNull(aa.arrCustomerName[0]);
119           assertNull(aa.arrDoctor[0]);
120           assertNull(aa.arrDate[0]);
121           assertNull(aa.arrTime[0]);
122
123               }
124           }
125
```

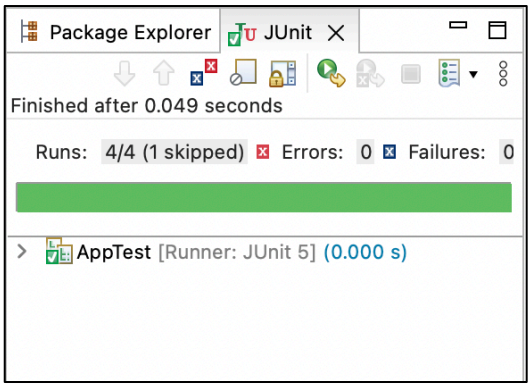
This method will be run as a third. In this method, I didn't declare sample arrays. Because I used the AssertNull() function to verify value is null or not. Using for statement, if the appointment whose customer name is "AAA" will be deleted. In the arrCustomerName, "AAA" is declared in the arrCustomerName[0]. So arrDoctor[0]'s value, arrDate[0], and arrTime[0] value will be modified as null.

d. Junit test method 4 – AddAppointment

```
127
128         @Disabled("Disabled until bug issue has been resolved")
129         @Test()
130         @Order(4)
131         void MakeAppointment(){
132             System.out.println("Disabled Method");
133         }
134
135 }//class*/
136
```

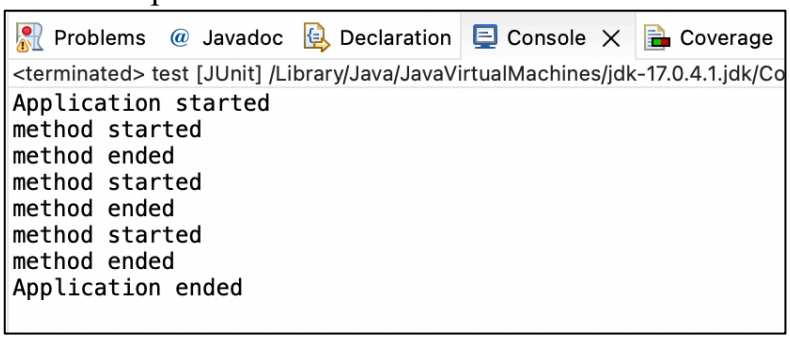
We developed the AddAppointment method but some bugs are issued. So we decided to disable this method until the bug issue has been solved.

e. Result



All test methods are passed in the Junit test. Modify, Delete, and Check functions in the application didn't have any bugs or errors.

f. Output



We can see that the @BeforeAll and @AfteAll functions are working successfully. Also, the @BeforeEach and @AfterEach functions worked correctly between each method run.

9.3. Integration Test

1) Overall Description

The meaning of the integration test is Integrate/combine unit test modules one by one and test their implementation in combined units. Generally, an integration test will be conducted after the 'unit test'. Once all units have been created and tested, combine the modules and integration test will be conducted. The main function or purpose of this test is to test the interface between every module.

In Unit Testing, codes used needs to be available, as it tests the written code, which is different in integration Testing, access to code is not needed, as it mainly tests the interactions and interfaces between modules. Software Developers do unit testing

2) Why we need to do integration test

a. When a program is developed, it is split into smaller modules and assigned one module to each developer. The logic implemented by one developer is very different from other developers, so it is important to ensure that the logic implemented by the developer meets expectations and render the correct values according to the specified criteria.

b. When moving from one module to another, the structure of the data is often changed. Some values are added or removed and subsequently the module has problems.

c. The module must also interact with some third-party tools or APIs and test as expected responses with accurate and generated data allowed by the API/tools.

3) Integration Testing Approach

a. Big Bang Approach

All components are integrated and tested at one at a time.

Advantages	Disadvantages
- Convenient for small systems	- Fault Localization is different. - It is easy to miss some interfaces links to be tested. - Integration testing can only be started after all modules have been designed, so the testing team has only a little time to run in the testing phase. - Because all modules are tested at once, it is difficult for high-risk core modules to be isolated and tested in order of priority. Peripheral modules handling user interfaces are not tested in order of priority

b. Incremental approach

Tests are performed by combining two or more logically related modules. Other modules involved for proper functionality are then added and tested. The process continues until all modules are successfully integrated and tested. These processes are performed using dummy programs called stubs and drivers. It can be divided into the following

[Bottom-Up approach]

In the Bottom-Up strategy, each module at a lower level is tested with a module at a higher level until all modules are tested. Need the help of a driver for the test.

Advantages	Disadvantages
- It is easy to locate the fault. - Unlike the Big Bang approach, there is no need to waste time developing all modules.	- Apps to control the flow of the module is the last time tested, and a defect tends to be. - it used to drive a prototype, it is difficult.

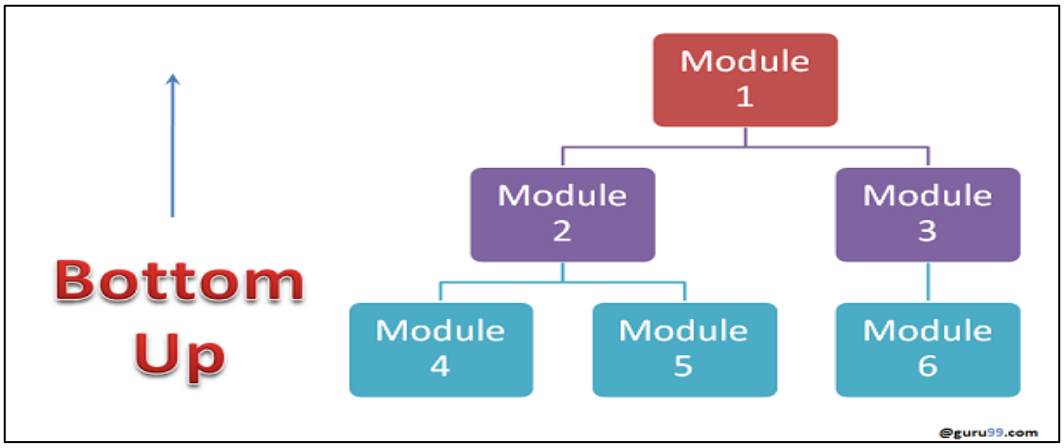


Figure 14. Bottom up approach

[Top-Down approach]

The control flow of the software system occurs from top to bottom. Get help from stubs for testing.

Advantages	Disadvantages
- It is easy to locate the fault. - There is a possibility that the prototype will be available as soon as soon as possible. - Critical modules can be tested in order of priority: critical design defects are found and may be corrected first.	- Need a lot of stubs. - The module is poorly tested at a low level.

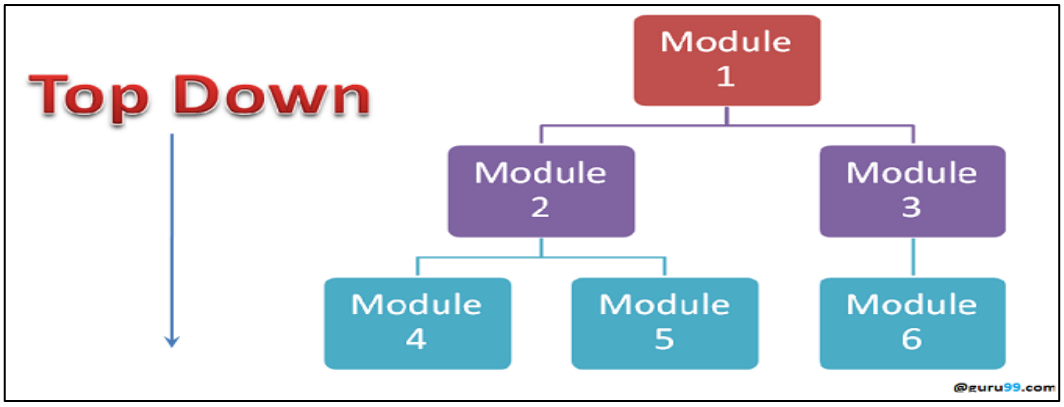


Figure 15. Top down approach

[Sandwich approach]

Combination of top-down and bottom-up approaches.

10. Artefacts of the agile methodology

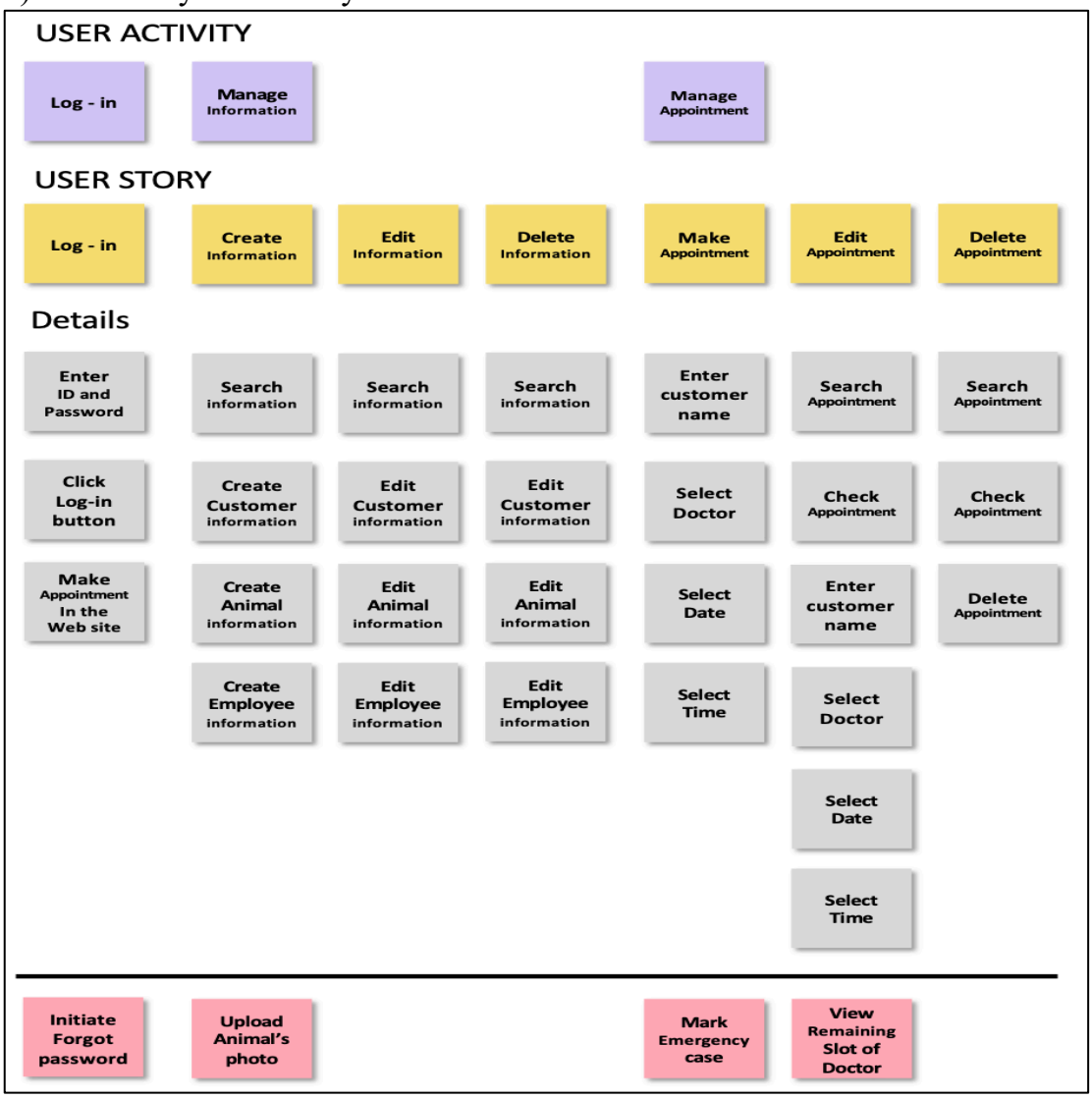
Artifacts are tools of scrum which will enable teams to manage their scrum process. They help team members communicate about the work they're doing and provide a record of what the team has agreed to do and what they have accomplished.

10.1. User stories

1) Overall Description

A user story a small unit in Agile framework and the purpose is to describe how the user or customer interprets the software features In Agile software development, user stories help the developer to understand what value a product feature can bring and understanding of why users want a certain functionality. It helps as communication tool they are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

2) Veterinary Practice System User Stories



10.2. Backlogs

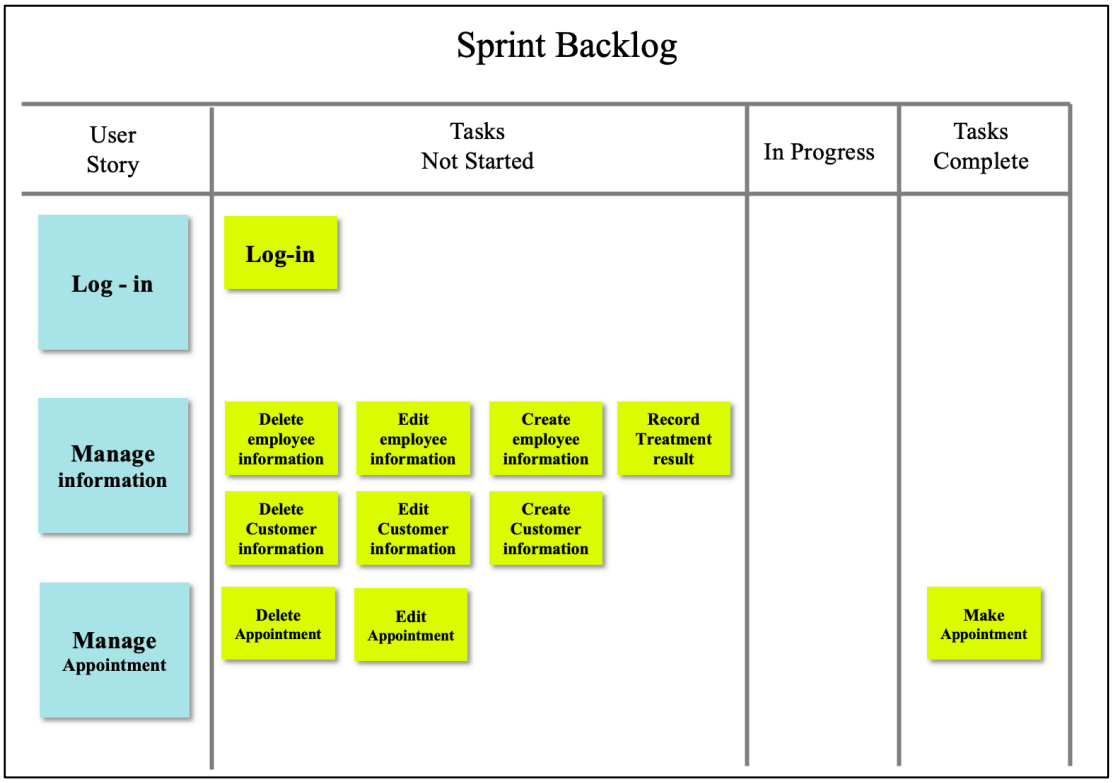
1) Overall Description

The product Backlog is an ordered list of everything that is known to be needed in. It is the single source of requirements for any changes to be made to the product. The product Owner is responsible for the product Backlog, including its content availability and ordering. We can identify the product Backlog is a list of requirements on which the Development team has to work. This list is maintained by the product owner.

But the product Backlog is not a simple list, it is a dynamic list. It evolves over time. It keeps changing as the product Owner and the Team discovers new information about the product . The product Backlog doesn't focus much on what the team is going to do in the distant future, it gives higher priority to the things which are to be done in the near future. This is because of the evolving nature of the Product Backlog. It is believed that the requirements and conditions will keep changing in the future, therefore focusing more on the things which are urgent saves a lot of time. The higher priority items are kept at the top of the list. First the work is done on these top tasks. The lower priority items at the bottom of the list are vague ideas as not much time is spent on the planning of those items. As these items keep moving towards the top of the list, more and more details are added to them. In this way, things get the focus of the team according to their importance.

2) Veterinary Practice System Product Backlog

Product Backlog					
ID	As a	Description	Estimated Time	Priority	Status
1	Receptionist	Make an appointment	8	Must	done
2	Member	Log-in	8	Must	To be started
3	Receptionist	Create new employee information	12	Must	To be started
4	Receptionist	Create new customer information	12	Must	To be started
5	Member	Manage appointment	20	Must	To be started
6	Veterinary	Record treatment result	12	Must	To be started
7	Receptionist	Edit employee information	8	Must	To be started
8	Receptionist	Edit customer information	8	Must	To be started
9	Veterinary, Nurse	Record treatment result	12	Could	To be started
10	Receptionist	Edit appointment	24	Should	To be started
11	Receptionist	Delete employee information	8	Should	To be started
12	Receptionist	Delete customer information	8	Should	To be started

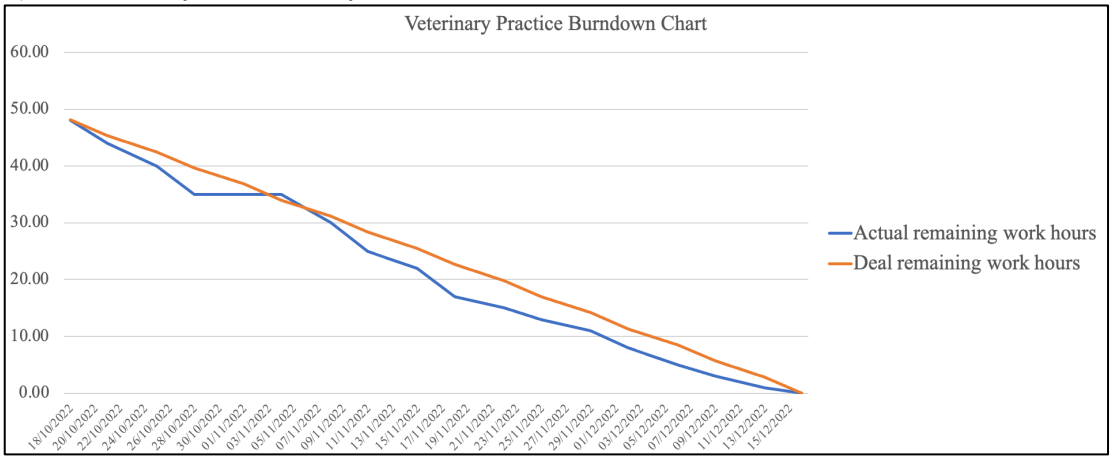


10.3. Burndown charts

1) Overall Description

In the Scrum framework of the Agile methodology, multiple cross-functional teams perform tasks, each of which is planned and executed on a short-term cycle basis of less than 4 weeks. These sprints or iterations allow teams to decide for themselves the amount of work to complete within a given cycle and collaborate no disruptively. Burndown charts provide a visual view of the progress of sprint units by drawing simple linear charts based on the time remaining and the amount of work remaining. The burn-down chart indicates whether the sprint is going as planned or if team-level coordination is required to achieve the stated goal.

2) Veterinary Practice System Burndown Chart



11. Glossary

No	Term	Category	Comments
1	System	Introduction	Implementation of the service based on customer's request. It include both program and work process.
2	Veterinary practice System.	Introduction	The system is used in veterinary clinic for manage information, manage appointment, and write medical record.
3	Customer	Actor	The owner of the animal
4	Animal information	Use case	Customer's pet information. It contains pet's species, name, sex, date of birth, description.
5	Account	Use case	An arrangement of user's information. It able to access to clinic's system by entering username and password.
6	ID	Sequence	The number on any official document or card that people use to prove their identity.
7	Username	Sequence	The account name created in the system for customer or employee
8	Treatment record	Sequence	The file that is recorded by employee. The employee records administration, injection, physical status of animal.
9	Compulsory information	Sequence	Mandatory information of customer. It includes name, contact, address, date of birth, id.
10	No data match	Sequence	The alert message when receptionist search customer's information on the system, but there isn't any matched data.
11	SWOT	Manage risk, quality	The method of analyze the good and bad features of situation or company. Strengths, weaknesses, opportunities, threats.
12	Critical path	Manage risk, quality	The process that allows to complete the work as shortest time and lowest cost.

12. Appendix (Source code and test code)

12.1. AppointmentApp.java

```
/*
 * Team K
 * AppointmentApp.java
 * 21191026 NAKYUNG KIM, 21234175 DONGHYEOK LEE, 21247544 SHINGIRIRAYI MABIKA
 */


import javax.swing.JOptionPane;

public class AppointmentApp {

    public static void main(String args[]) {
        // declare variables
        int x = 0;

        String arrCustomerName[];
        arrCustomerName = new String[100];

        String arrDoctor[];
        arrDoctor = new String[100];

        String arrDate[];
        arrDate = new String[100];

        String arrTime[];
        arrTime = new String[100];

        // declare objects
        Function myFunction;

        // create objects
        myFunction = new Function();

        while (x != 4) {

            String[] options = { "Make an appointment", "Check my appointment
schedule",
                    "Change my appointment schedule", "Delete my appointment
schedule", "Exit application" };

            x = JOptionPane.showOptionDialog(null, "What do you want to do?",
                    "Click a button",
                    JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE,
null, options, options[0]);

            // make an appointment

            if (x == 0) {

                // input
                myFunction.setCustomerName(arrCustomerName);
                myFunction.setDoctor(arrDoctor);
                myFunction.setDate(arrDate);
                myFunction.setTime(arrTime);

                // process
                myFunction.MakeAppointment();

                // check an appointment schedule

            } else if (x == 1) {

                // input
```

```
                myFunction.setCustomerName(arrCustomerName);
                myFunction.setDoctor(arrDoctor);
                myFunction.setDate(arrDate);
                myFunction.setTime(arrTime);

                // process
                myFunction.CheckAppointment();
            }

            // change an appointment schedule

            else if (x == 2) {

                // input
                myFunction.setCustomerName(arrCustomerName);
                myFunction.setDoctor(arrDoctor);
                myFunction.setDate(arrDate);
                myFunction.setTime(arrTime);

                // process
                myFunction.ChangeAppointment();

            }

            // delete an appointment schedule

            else if (x == 3) {

                // input
                myFunction.setCustomerName(arrCustomerName);
                myFunction.setDoctor(arrDoctor);
                myFunction.setDate(arrDate);
                myFunction.setTime(arrTime);

                // process
                myFunction.DeleteAppointment();

            }

            else if (x == 4) {
                JOptionPane.showMessageDialog(null, "Thank you! Bye.");
            }

            else {
                JOptionPane.showMessageDialog(null, "Please select menu again.");

            }
        }
    }
}
```

12.2. function.java

```
/*
 * Team K
 * function.java
 * 21191026 NAKYUNG KIM, 21234175 DONGHYEOK LEE, 21247544 SHINGIRIRAYI MABIKA
 */

import java.util.Arrays;

import javax.swing.JOptionPane;

public class Function {
    // data members
    private String arrCustomerName[] = new String[100];
    private String arrDoctor[] = new String[100];
    private String arrDate[] = new String[100];
    private String arrTime[] = new String[100];
    int i = 0;

    // constructor
    public Function() {
    }

    // set methods - one for every input
    public void setCustomerName(String arrCustomerName[]) {
        this.arrCustomerName = arrCustomerName;
    }

    public void setDoctor(String arrDoctor[]) {
        this.arrDoctor = arrDoctor;
    }

    public void setDate(String arrDate[]) {
        this.arrDate = arrDate;
    }

    public void setTime(String arrTime[]) {
        this.arrTime = arrTime;
    }


    // compute - process
    // make an appointment
    public void MakeAppointment() {
        String confirm = "no";
        while (confirm.equals("no")) {

            String name = "true";
            while (name.equals("true")) {
                arrCustomerName[i] = JOptionPane.showInputDialog(null, "Please
write your name");

                if (arrCustomerName[i] == null) {
                    JOptionPane.showMessageDialog(null,
                            "Please enter your name");

                } else {
                    name = "false";
                }
            }

            String doctor = "true";
            while (doctor == "true") {

                String[] selectionValues = { "Dr.KIM", "Dr.LEE", "Dr.SHING" };
                String initialSelection = "Veterinarian";
```

```
                Object selection = JOptionPane.showInputDialog(null, "Which
veterinerian do you want to see?",
                        "Make an appointment", JOptionPane.QUESTION_MESSAGE, null,
selectionValues,
                        initialSelection);
                arrDoctor[i] = (String) selection;

                if (arrDoctor[i] == null) {
                    JOptionPane.showMessageDialog(null,
                            "Please select the doctor you want");

                } else {
                    doctor = "false";
                }
            }

            String month = "true";
            while (month == "true") {

                String[] selectionMonth = { "01/2023", "02/2023", "03/2023" };
                String initialSelectionMonth = "Month";
                Object selectionMonths = JOptionPane.showInputDialog(null,
                        "Which month do you want to see the doctor?",
                        "Make an appointment", JOptionPane.QUESTION_MESSAGE, null,
selectionMonth,
                        initialSelectionMonth);

                if ((String) selectionMonths == "01/2023") {
                    String date = "true";
                    while (date == "true") {

                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10", "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24", "25",
                                "26", "27", "28", "29", "30", "31" };
                        String initialSelectionDate = "Date";
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);

                        if (selectionDates != null) {
                            arrDate[i] = (String) selectionDates + "/" + (String)
selectionMonths;
                            date = "false";
                            month = "false";

                        } else {
                            JOptionPane.showMessageDialog(null, "Please input the
date you want");
                            month = "false";
                        }
                    }

                } else if ((String) selectionMonths == "02/2023") {
                    String date = "true";
                    while (date == "true") {

                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10", "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24", "25",
                                "26", "27", "28" };
                        String initialSelectionDate = "Date";
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
```

```
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);

                        if (selectionDates != null) {
                            arrDate[i] = (String) selectionDates + "/" + (String)
selectionMonths;
                            date = "false";
                            month = "false";

                        } else {
                            JOptionPane.showMessageDialog(null, "Please input the
date you want");
                            month = "false";
                        }
                    }

                } else if ((String) selectionMonths == "03/2023") {

                    String date = "true";
                    while (date == "true") {

                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10", "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24", "25",
                                "26", "27", "28", "29", "30", "31" };
                        String initialSelectionDate = "Date";
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);

                        if (selectionDates != null) {
                            arrDate[i] = (String) selectionDates + "/" + (String)
selectionMonths;
                            date = "false";
                            month = "false";

                        } else {
                            JOptionPane.showMessageDialog(null, "Please input the
date you want");
                            month = "false";
                        }
                    }

                } else if (selectionMonths == null) {
                    JOptionPane.showMessageDialog(null,
                            "Please select the month you want");

                } else {
                    month = "false";
                }
            }

            String answer = "true";
            while (answer == "true") {

                String time = "true";
                while (time == "true") {

                    String[] selectionTime = { "09:00", "10:00", "11:00", "12:00",
"13:00", "14:00", "15:00",
                            "16:00", "17:00", "18:00" };
                    String initialSelectionTime = "Time";
                    Object selectionTimes = JOptionPane.showInputDialog(null,
                            "What time do you want to see the doctor?",
```

```
                            "Make an appointment", JOptionPane.QUESTION_MESSAGE,
null, selectionTime,
                            initialSelectionTime);
                    arrTime[i] = (String) selectionTimes;

                    if (arrTime[i] == null) {
                        JOptionPane.showMessageDialog(null,
                                "Please select the time you want");

                    } else {
                        time = "false";
                    }
                }

                if (i != 0) {
                    for (int z = 0; z < i; z++) {

                        if (arrDoctor[i].equals(arrDoctor[z]) &&
arrDate[i].equals(arrDate[z])
                                && arrTime[i].equals(arrTime[z])) {
                            JOptionPane.showMessageDialog(null,
                                    "The Doctor already has another appointment at
that time");

                        } else {
                            answer = "false";
                        }
                    }

                } else {
                    answer = "false";
                }
            }

            confirm = JOptionPane.showInputDialog(null, "Your appointment schedule
is"
                    + "\nCustomer Name :" + arrCustomerName[i]
                    + "\nDoctor :" + arrDoctor[i]
                    + "\nDate :" + arrDate[i]
                    + "\nTime :" + arrTime[i]
                    + "\nIf it's okay, please write 'yes'"
                    + "\nIf it's not okay, please write 'no'");
        }
        i = i + 1;
    }

    // check appointment
    public void CheckAppointment() {
        String CustomerName;
        CustomerName = JOptionPane.showInputDialog(null, "Plese enter your name.");
        boolean contains =
Arrays.stream(arrCustomerName).anyMatch(CustomerName::equals);

        for (int k = 0; k < arrDoctor.length; k++) {

            if (arrCustomerName[k] != null &&
arrCustomerName[k].equals(CustomerName)) {
                JOptionPane.showMessageDialog(null, "Your appointment schedule is"
                        + "\nCustomer Name :" + arrCustomerName[k]
                        + "\nDoctor :" + arrDoctor[k]
                        + "\nDate :" + arrDate[k]
                        + "\nTime :" + arrTime[k]);
                break;

            } else if (contains == false) {
                JOptionPane.showMessageDialog(null, "You don't have any appointment
schedule yet");
                break;
```

```
            }
        }
    }

    // chage appointment
    public void ChangeAppointment() {
        String CustomerName;
        CustomerName = JOptionPane.showInputDialog(null, "Plese write your name.");
        boolean contains =
Arrays.stream(arrCustomerName).anyMatch(CustomerName::equals);

        for (int m = 0; m < arrDoctor.length; m++) {

            if (arrCustomerName[m] != null &&
arrCustomerName[m].equals(CustomerName)) {
                String change;
                change = JOptionPane.showInputDialog(null, "Your appointment
schedule is"
                        + "\nCustomer Name :" + arrCustomerName[m]
                        + "\nDoctor :" + arrDoctor[m]
                        + "\nDate :" + arrDate[m]
                        + "\nTime :" + arrTime[m]
                        + "\nDo you want to change the appointment
shedule?(yes/no)");

                if (change.equals("yes")) {
                    String[] selectionValues = { "Dr.KIM", "Dr.LEE", "Dr.SHING" };
                    String initialSelection = "Veterinarian";
                    Object selection = JOptionPane.showInputDialog(null,
                            "Which veterinerian do you want to see?",
                            "Make an appointment", JOptionPane.QUESTION_MESSAGE,
null, selectionValues,
                            initialSelection);
                    arrDoctor[m] = (String) selection;
                    String[] selectionMonth = { "01/2023", "02/2023", "03/2023" };
                    String initialSelectionMonth = "Month";
                    Object selectionMonths = JOptionPane.showInputDialog(null,
                            "Which month do you want to see the doctor?",
                            "Make an appointment", JOptionPane.QUESTION_MESSAGE,
null, selectionMonth,
                            initialSelectionMonth);

                    if ((String) selectionMonths == "01/2023") {
                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10",
                                "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24",
                                "25",
                                "26", "27", "28", "29", "30", "31" };
                        String initialSelectionDate = "Date";
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);
                        arrDate[m] = (String) selectionDates + "/" + (String)
selectionMonths;

                    } else if ((String) selectionMonths == "02/2023") {
                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10",
                                "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24",
                                "25",
                                "26", "27", "28" };
                        String initialSelectionDate = "Date";
```

```
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);
                        arrDate[m] = (String) selectionDates + "/" + (String)
selectionMonths;

                    } else {
                        String[] selectionDate = { "01", "02", "03", "04", "05",
"06", "07", "08", "09", "10",
                                "11",
                                "12", "13", "14", "15", "16", "17", "18", "19",
"20", "21", "22", "23", "24",
                                "25",
                                "26", "27", "28", "29", "30", "31" };
                        String initialSelectionDate = "Date";
                        Object selectionDates = JOptionPane.showInputDialog(null,
                                "Which date do you want to see the doctor?",
                                "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionDate,
                                initialSelectionDate);
                        arrDate[m] = (String) selectionDates + "/" + (String)
selectionMonths;
                    }

                    String answer = "true";
                    while (answer == "true") {

                        String time = "true";
                        while (time == "true") {

                            String[] selectionTime = { "09:00", "10:00", "11:00",
"12:00", "13:00", "14:00",
                                    "15:00",
                                    "16:00", "17:00", "18:00" };
                            String initialSelectionTime = "Time";
                            Object selectionTimes =
JOptionPane.showInputDialog(null,
                                    "What time do you want to see the doctor?",
                                    "Make an appointment",
JOptionPane.QUESTION_MESSAGE, null, selectionTime,
                                    initialSelectionTime);
                            arrTime[m] = (String) selectionTimes;

                            if (arrTime[m] == null) {
                                JOptionPane.showMessageDialog(null,
                                        "Please select the time you want");

                            } else {
                                time = "false";
                            }
                        }

                        if (m != 0) {

                            for (int z = 0; z < m; z++) {

                                if (arrDoctor[m].equals(arrDoctor[z]) &&
arrDate[m].equals(arrDate[z])
                                        && arrTime[m].equals(arrTime[z])) {
                                    JOptionPane.showMessageDialog(null,
                                            "The Doctor already has another
appointment at that time");
                                } else {
                                    answer = "false";
                                }
                            }
```

```
                        } else {
                            answer = "false";
                        }
                    }

                    break;

                } else if (change.equals("no")) {
                    break;
                }

            } else if (contains == false) {
                JOptionPane.showMessageDialog(null, "You don't have any appointment
schedule yet");
                break;
            }
        }
    }

    // Delete appointment
    public void DeleteAppointment() {
        String CustomerName;
        CustomerName = JOptionPane.showInputDialog(null, "Plese write your name.");
        boolean contains =
Arrays.stream(arrCustomerName).anyMatch(CustomerName::equals);

        for (int j = 0; j < arrDoctor.length; j++) {

            if (arrCustomerName[j] != null &&
arrCustomerName[j].equals(CustomerName)) {
                String change;
                change = JOptionPane.showInputDialog(null, "Your appointment
schedule is"
                        + "\nCustomer Name :" + arrCustomerName[j]
                        + "\nDoctor :" + arrDoctor[j]
                        + "\nDate :" + arrDate[j]
                        + "\nTime :" + arrTime[j]
                        + "\nDo you want to delete the appointment
shedule?(yes/no)");

                if (change.equals("yes")) {
                    arrCustomerName[j] = "null";
                    arrDoctor[j] = "null";
                    arrDate[j] = "null";
                    arrTime[j] = "null";
                    JOptionPane.showMessageDialog(null, "Complete to delete!");
                    break;

                } else if (change.equals("no")) {
                    break;
                }

            } else if (contains == false) {
                JOptionPane.showMessageDialog(null, "You don't have any appointment
schedule yet");
                break;
            }
        }
    }
}
```

12.3. Junit Test code

```
/*
 * Team K
 * Junit code
 * 21191026 NAKYUNG KIM, 21234175 DONGHYEOK LEE, 21247544 SHINGIRIRAYI MABIKA
 */

package test;


import static org.junit.jupiter.api.Assertions.assertArrayEquals;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNull;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestMethodOrder;

import oose.Appointment;

@TestMethodOrder(OrderAnnotation.class)
class Testing {

	@BeforeAll
	static void setUpBeforeClass() throws Exception {
		System.out.println("Application started");
	}

	@AfterAll
	static void tearDownAfterClass() throws Exception {
		System.out.println("Application ended");
	}

	@BeforeEach
	void setUp() throws Exception {
		System.out.println("method started");
	}

	@AfterEach
	void tearDown() throws Exception {
		System.out.println("method ended");
	}

	@Test
	@Order(1)
	void CheckAppointment() {
		Appointment aa = new Appointment();

		 String arrCustomerName_check[] = new
String[]{"AAA","BBB","CCC","DDD","EEE"};

		 String arrDoctor_check[] = new
String[]{"Kim","Lee","Shingi","Kim","Lee"};

		 String arrDate_check[] = new String[]{"2023-01-04","2023-01-
28","2023-01-15","2023-01-07","2023-01-12"};

		 String arrTime_check[] = new
String[]{"14:00","15:00","09:00","10:00","17:00"};
```

```
            assertArrayEquals(aa.arrCustomerName, arrCustomerName_check);
            assertArrayEquals(aa.arrDoctor, arrDoctor_check);
            assertArrayEquals(aa.arrDate, arrDate_check);
            assertArrayEquals(aa.arrTime, arrTime_check);

      }


      @Test
      @Order(2)
      void ChangeAppointment() {
                  Appointment aa = new Appointment();

               String arrCustomerName_check[] = new
String[]{"AAA","FFF","CCC","DDD","EEE"};

               String arrDoctor_check[] = new
String[]{"Kim","Shingi","Shingi","Kim","Lee"};

               String arrDate_check[] = new String[]{"2023-01-04","2023-01-
30","2023-01-15","2023-01-07","2023-01-12"};

               String arrTime_check[] = new
String[]{"14:00","15:00","09:00","10:00","17:00"};

            for(int i = 0; i < 5; i++) {
                  if(aa.arrCustomerName[i] == "BBB") {
                        aa.arrCustomerName[i] = "FFF";
                        aa.arrDoctor[i] = "Shingi";
                        aa.arrDate[i] = "2023-01-30";
                        aa.arrTime[i] = "15:00";
                  assertEquals(aa.arrCustomerName[1], arrCustomerName_check[1]);
                  assertEquals(aa.arrDoctor[1], arrDoctor_check[1]);
                  assertEquals(aa.arrDate[1], arrDate_check[1]);
                  assertEquals(aa.arrTime[1], arrTime_check[1]);

                        }

                  }




         }


      @Test()
      @Order(3)
      void DeleteAppointment() {
            Appointment aa = new Appointment();

            for(int i = 0; i < 5; i++) {
                  if(aa.arrCustomerName[i] == "AAA") {
                  aa.arrCustomerName[i] = null;
                  aa.arrDoctor[i] = null;
                  aa.arrDate[i] = null;
                  aa.arrTime[i] = null;
                  }

                  assertNull(aa.arrCustomerName[0]);
                  assertNull(aa.arrDoctor[0]);
                  assertNull(aa.arrDate[0]);
                  assertNull(aa.arrTime[0]);

                        }
                  }
```

```
	@Disabled("Disabled until bug issue has been resolved")
	@Test()
	@Order(4)
	void MakeAppointment(){
		System.out.println("Disabled Method");
	}

}//class*/
```

12.4. Appointment.java (Junit testing java object code)

```
/*
 * Team K
 * Appointment.java
 * 21191026 NAKYUNG KIM, 21234175 DONGHYEOK LEE, 21247544 SHINGIRIRAYI MABIKA
 */


package oose;

import java.util.Arrays;

public class Appointment {

	public String arrCustomerName[] = new
String[]{"AAA","BBB","CCC","DDD","EEE"};

	public String arrDoctor[] = new String[]{"Kim","Lee","Shingi","Kim","Lee"};

	public String arrDate[] =  new String[]{"2023-01-04","2023-01-28","2023-01-
15","2023-01-07","2023-01-12"};

	public String arrTime[] = new
String[]{"14:00","15:00","09:00","10:00","17:00"};


    public Appointment() {

    }


	@Override
	public String toString() {
		return "Reservation [arrCustomerName=" +
Arrays.toString(arrCustomerName) + ", arrDoctor="
				+ Arrays.toString(arrDoctor) + ", arrDate=" +
Arrays.toString(arrDate) + ", arrTime="
				+ Arrays.toString(arrTime) + "]";
	}


	public void CheckAppointment() {
		for(int i = 0; i < arrCustomerName.length; i++) {
		System.out.println((i+1)+"\n"+"\nCustomer's name:
"+arrCustomerName[i]+"\nDoctor: "+arrDoctor[i]+"\nDate: "+arrDate[i]+"\nTime:
"+arrTime[i]+"\n====================");

		}

	}

	public void ChangeAppointment() {
		for(int i = 0; i < arrCustomerName.length; i++) {
			if(arrCustomerName[i] == "BBB") {
				arrCustomerName[i]= "FFF";
				arrDoctor[i] = "Shingi";
				arrDate[i] = "2023-01-30";
				arrTime[i] = "15:00";
				//System.out.println("\nCustomer's name:
"+arrCustomerName[i]+"\nDoctor: "+arrDoctor[i]+"\nDate: "+arrDate[i]+"\nTime:
"+arrTime[i]+"\n====================");

			}

		}
```

```
	}

	public void DeleteAppointment() {
		for(int i = 0; i < arrCustomerName.length; i++) {
			if(arrCustomerName[i] == "AAA") {
				arrCustomerName[i]= null;
				arrDoctor[i] = null;
				arrDate[i] = null;
				arrTime[i] = null;
				//System.out.println("\nCustomer's name:
"+arrCustomerName[i]+"\nDoctor: "+arrDoctor[i]+"\nDate: "+arrDate[i]+"\nTime:
"+arrTime[i]+"\n====================");

			}
		}
	}



}
```

Bibliography

Dennis, A, Wixom, B. H and Roth, R.M. (2012), System Analysis and Design 5th Edition, USA, John Wiley & Sons, Inc.

Guan, D. (2007) “Conflicts in project environment.” at 2007 PMI Global Congress – Asia Pacific, Hong Kong, People’s Republic of China Newtown Square, PA; Project management Institute

Dennis, A, Wixom, B and Tegarden, D. (2015), Systems Analysis and Design with UML, 5th edition, NJ, John Wiley & Sons, Inc.

Shore, J., and Warden, S. (2014), The Art of Agile Development, 1. USA, O'Reilly Media

Unit testing vs Intergration testing () Available at: https://circleci.com/blog/unit-testing-vs-integration-testing/ [Accessed 19 December 2022)

Wahid, M and Almalaise, A., "JUnit framework: An interactive approach for basic unit testing learning in Software Engineering," *2011 3rd International Congress on Engineering Education (ICEED)*, 2011, pp. 159-164, doi: 10.1109/ICEED.2011.6235381.

Team Member NAKYUNG KIM
- 2.3.1. Use Case specification – Manage information
- 4.1.1. Sequence Diagram – Create information - Diagram
- 4.1.2. Sequence Diagram – Create information – System Behavior - Contracts
- 4.2.1. Sequence Diagram – Manage information - Diagram
- 4.2.2. Sequence Diagram – Manage information – System Behavior - Contracts
- 5.1. Communication Diagram – Create information
- 5.2. Communication Diagram – Manage information

Team Member DONGHYEOK LEE
- 2.3.2. Use Case specification – Make an appointment by customer
- 2.3.3. Use Case specification – Make an appointment by receptionist
- 4.3.1. Sequence Diagram – Make an appointment - Diagram
- 4.3.2. Sequence Diagram – Make an appointment – System Behavior – Contracts
- 5.3. Communication Diagram – Make an appointment

Team Member SHINGIRIRAYI MABIKA
- 2.3.4. Use Case specification – Log in
- 2.3.5. Use Case specification – Log out
- 4.4.1. Sequence Diagram – Log in - Diagram
- 4.4.2. Sequence Diagram – Log in – System Behavior - Contracts
- 5.4. Communication Diagram – Log in

The others contents made by Team K all members together.