CPT202 Assignment 1
Group Report for Software Engineering Group Project

2022/2023 Semester 2
<Pet Shop Appointment System>
<2023/4/26>
Group number: <10>
Submitted by: Yuan, 2036501>

Group Members:
1. <Shuchen Yuan, 2036501 >
2. <Leduo Chen, 2035836 >
3. <Xi Kong, 2037023 >
4. <Yihan Zhou, 2033677 >
5. <Juntuo Wang, 2036505 >
6. <Suqi Zhang, 2033871 >
7. <Yuzhen Chen, 2036328 >
8. <Yuyi Yang, 2036676 >
9. <Bowen Li, 2033827 >

CONTENTS

I. Introduction

A. Problem Statement

1) Current State

Customers and pet grooming businesses are facing inconvenience and inefficiencies challenges from many perspectives. Detailed speaking, pet owners often encounter difficulties in scheduling and tracking appointments smoothly, finding suitable groomers, and comparing services. Meanwhile, shop managers struggle with appointment tracking, resource management, and maintaining accurate customer information as well as finding decision support based on data analysis.

2) Desired State

The ideal scenario involves a user-friendly platform where customers could smoothly make wanted appointments as well as get detailed information about their booked appointment and the shop manager could efficiently manage the shop and allocate the resources of the shop. In addition, the system should provide some decision support based on data analysis and help the manager make appropriate sales strategies.

B. Aim of the Project

This project aims to develop an efficient online appointment system called iPet that addresses technical challenges such as security access, appointment management, and customization tool for making upselling and cross-selling strategies. It features a user-friendly appointment interface, detailed service descriptions, appointment management tools, data analysis capabilities, and multi-platform adaptability. By providing a comprehensive solution, we aim to promote seamless interactions between pet owners and grooming businesses, ultimately improving the overall experience for both parties.

C. Project Scope

1) Scope Description

The iPet project focuses on delivering a web-based platform that offers customers secure registration and login access, smooth appointment booking and management, current appointment status tracking, and appointment history searching for the customer. Add-on features like personalized account management, visual display of store content which includes both groomer ranks and service details, as well as pricing transparency is provided as well for them. Additionally, for shop managers, the system will enable master file maintenance and data visualization for which they can make modifications to critical system data such as the information of the customers, groomers and service, while also providing detailed information of recent appointments, business analytics and reporting features in specific time period. Finally, the iPet system will feature an ability to edit upselling and cross-selling strategies, ensuring the shop manager timely develop the selling strategies adaptive to marketing fluctuations and changing needs of customers. The primary goal is to enhance the user experience for pet owners and streamline business processes for grooming service providers. The project does not cover additional services such as pet boarding, training, or veterinary service.

2) Project Deliverables

- **User registration and account management**: Customers can successfully register an account and login to the website as well as log out from the website with security.
- **Appointment booking and management**: Customers can smoothly make appointments as well as pay for it and see their appointment history and status.
- **Groomer ranking and service pricing**: The website will display an intuitive ranking of groomer and service pricing to help them make appointment.
- **Sales strategies–upselling and cross-selling**: The shop manager will be able to upload and modify the information of upselling and cross-selling strategies.
- **Master file maintenance and management**: Shop manager can add, delete, search and modify the customers, groomers and services' data information after viewing.
- **Business analytics and reporting**: Shop manager can view the last 10 appointments of the shop as well as the annual and quarterly financial report since opening.

D. User Characteristics

1) Customers Characteristic

- Own or temporarily care for a pet and prefer online solutions for daily tasks and activities.

- Desire an easy-to-use interface that allows them to manage appointments, view order history, and update personal and pet information.
- Appreciate personalized recommendations, offers, and promotions based on their pet's needs and preferences.
- Prioritize convenience and efficiency in appointment booking and service selection.

2) Shop Managers Characteristic

- Responsible for overseeing the overall operations of the pet grooming business.
- Require a comprehensive system that enables them to manage and maintain master files, appointment schedules, and upselling and cross-selling policies.
- Value access to statistical reports and data analytics to make informed business decisions and strategies.
- Desire an intuitive interface that streamlines their daily tasks and responsibilities, improving overall business efficiency.

E. Assumptions and Dependencies

1) Assumptions

- **Human Resource Availability**: All project team members are available and in good conditions as well as have the skills and knowledge needed to carry out the project.
- **Scheduling Accuracy**: Deadlines and milestones set are achievable and the project can be completed on time.
- **Resource Support**: All necessary equipment and goods are available and in good condition when needed. The budget established is accurate and covers all project costs. Supports will be provided from project sponsor, stakeholder and project manager.

2) Dependency

- **Team Collaboration Platform establishment**: Our team work based on a remote code versioning platform. A repository of our team project is needed to be established at first.
- **Coding Versioning Control Commission**: Start to use Github for code versioning control of this project.
- **Development environment construction**: Define and configure the specific development environment together of the project, and all members maintain the consistency of the development environment. (Java 16.0.2 MySql 8.0.31)
- **Task Allocation**: Because in this coursework each member has to go through the complete development life circle, the task is allocation evenly to the group members.
- **Functional Requirement Development**: Develop the system's UI and functions based on user specifications. Define epics, features, PBIs, and tasks, saving completed functionality code to the Github repository.
- **Server Deployment:** Get a cloud server at first, then deploy and configure the server where we need to complete the installation of OS, Java, MySql and upload the source code of the project to the server. The codes uploaded should be run without bug and error.
- **Iteration/Update**: Iterations and improvements based on previously developed site models (if any) could be done. The old version of the system should be kept until the new version finished completely.

F. Project Risk

The following list the most likely encountered project risks.

1) Project Risks

- **Technical Difficulties**: Unanticipated technical issues may arise, affecting the project's timeline and cost due to capability mismatches between the project's demands and the development team's skills.
- **Scope Creep**: Unclear or evolving requirements can lead to increased complexity, delays, and higher costs.

2) Product Risks

- **Third-party Components**: Acquired components or software libraries might underperform, requiring additional resources to optimize or replace them and potentially causing development delays.
- **Security Vulnerabilities**: The iPet system could face security breaches or data leaks, which may compromise user data and harm its reputation, necessitating extra resources and time to address these vulnerabilities.

3) Business Risks

- **Competitor Advancements**: New products from competitors may impact iPet's market

share and adoption. Monitoring the competitive landscape and ensuring iPet remains innovative is crucial.

- **Market Adoption**: The iPet system might face challenges in attracting users and gaining market acceptance. Developing a strong marketing strategy and engaging with potential users to gather feedback can help mitigate this risk.

II. ARCHITECTURAL DESIGN

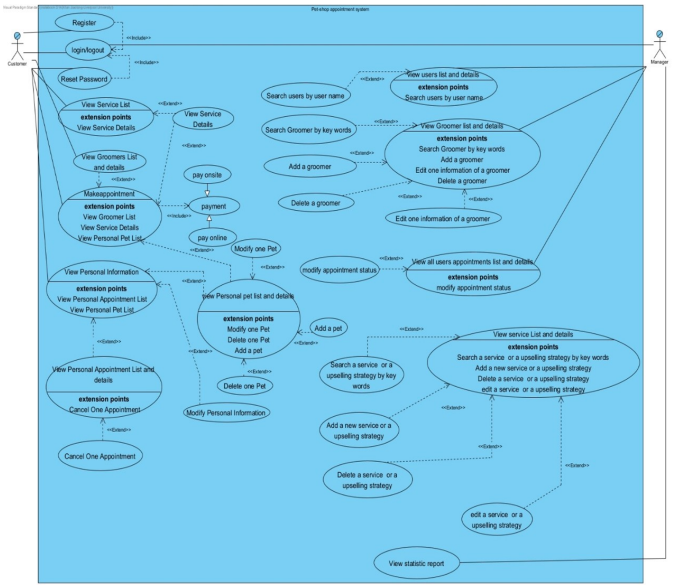


Fig. 1. Our Use Case Diagram

iPet is a WebApp based on MVC structure (Model-View-Controller) and B/S structure. These two structural theories guide the construction and construction of the entire system.

Because there are a large number of use cases concerned with Viewing Function, such as View Service List, View Groomers List, View Appointment List, View Pet List, and View User List. All of them need support from the database and the communication with the database is very frequent, which places demands on the design of our database. Meanwhile, there also exist connections and interactions between different lists. This situation requires that the system need to have low coupling and be easy to maintain the database. It is one of the reasons why we choose MVC as our system structure. Because the MVC structure divides the whole system into three parts: Model, View and Controller. The three modules are independent of each other, avoiding code logic clutter and making the system design more obvious. It organizes code in a way that separates business logic, data, and interface display, aggregates business logic into one part and does not need to rewrite business logic while improving and personalizing the interface and user interaction. And each View has its own corresponding Controller. Therefore, the system is able to isolate the design of the database, so that when our development team alter the database it would not influence other components directly. These features bring it a low coupling, high reusability and favourable software engineering management.

Furthermore, a lot of use cases in our system which try to edit our database, for example, a series of adding, deleting, modifying and checking operations in each page or list, would bring a considerable load to our database and server. In this case, our system server and its database must have strong maintainability to support its future works, which is the strength that makes MVC famous. At the same time, some extensive work in the foreseeable future will also require our system architecture design, which demands the reusability, life cycle costs, and modularization of the system. And this coincides with the structural advantages of MVC. Therefore, we choose MVC as our system structure.

On the other hand, our system is a WebApp, which means system users access it by a browse. And each use case needs a specific page to contain their view and information. Moreover, the website should be accessed at anytime and anywhere. Its characteristic command our system has excellent distribution. However, the traditional C/S structure requires users to download the specialized client installer and program and has poor compatibility, which does not satisfy the needs of our system. Compared with the traditional C/S structure, the advantages of the B/S structure in these aspects are prominent, which perfectly meets the needs of a WebApp. Thus, in order to have better performance, higher maintainability and compatibility, we choose the B/S structure instead of C/S.

A. System Architecture

1) MVC Pattern

As is illustrated in Fig. 2, The MVC pattern consists of three logical components that work together: ***Model, View, and Controller***. The Model handles

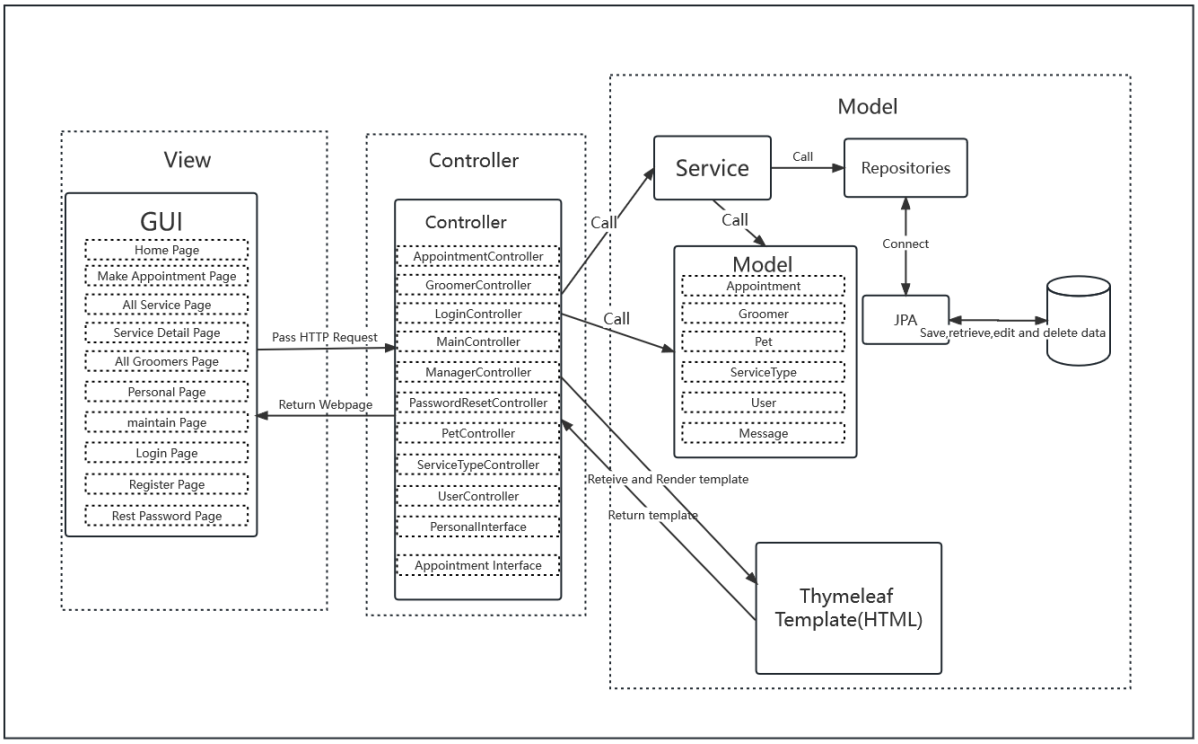


Fig. 2. The MVC Pattern

the data and its operations, the View determines how the data is presented to the user, and the Controller manages user interactions and communicates with both the Model and the View.

a) Model

In MVC pattern, the Model component represents the data and business logic of the system. It is responsible for managing and storing all information related to the pet grooming services, appointments, users, and other relevant data. It is made up of four modules: service, repositories, model and Thymeleaf template. We use the spring MVC to implement our system. In our project, we use Spring MVC to implement the MVC pattern.

In Spring MVC, model classes are used to represent the data of the application. They are often used in combination with the View layer to display data to the user. For example, a model object could contain data retrieved from a database and then be passed to a View for rendering the data in a web page. The Model object could also be used to validate input from the user before it is passed to the Service layer for processing. The service classes are a common pattern for implementing shared or reused business logic between different parts of an application, and they are often used to handle some cross-model business requirements. The repositories classes are used to manage data access and persistence. This kind of classes encapsulates the logic for accessing data from a database or other persistent storage which provide connection of JPA for the methods in service classes. In addition, Thymeleaf template is also a significant module in model component, which provide the HTML template to controller in order to transferring them to vision.

b) Vision System (Front-end)

The View component is responsible for presenting the data to the user and handling user interactions. It includes the user interface (UI) elements, such as buttons, input fields, and other graphical elements, that allow users to interact with the system Through HTTP request and response. The View will render the data retrieved from the Model through thymeleaf in a visually appealing and user-friendly manner. In the iPet system, the View will be developed using modern web technologies like HTML, CSS, and JavaScript, and will include web pages for user registration, login, appointment booking, appointment history, and more. In those pages,

the user interface (UI) will be designed using a responsive approach, ensuring compatibility across different devices, screen sizes, and resolutions. This will be achieved by employing CSS media queries and flexible grid layouts. In addition, A front-end framework, will be used to create reusable UI components and handle dynamic user interactions more efficiently. This will improve the overall development speed and enable smoother user experiences.

c) Controller

According to Fig. 1, the Controller component acts as an intermediary between the Model and View components, processing user input and requests, updating the Model, and updating the View accordingly. The Controller is responsible for handling user actions, such as booking an appointment or updating user information, and ensuring that the appropriate changes are made in the Model and reflected in the View. When a user interacts with the application, such as clicking a button or entering data in a form, the Controller receives a HTTP request as an input and then processes this input through mapping it into a method and determine the appropriate action to take.

d) Working process of our MVC architecture

The detailed information of the MVC pattern of fig. 1 is as followed. The user initiates interaction with the web page by clicking buttons, filling out forms, and performing various actions, which send an HTTP request to the Spring MVC application (Controller). Upon receiving the request, the front controller, selects the corresponding controller based on the URL. This controller is responsible for handling the request and invoking the service layer to perform the necessary business logic, such as handling database transactions, calling external services, and applying business rules. Once the service layer processes the business logic, it passes the data to the model layer for further manipulation. Comprising entity classes, data access objects (DAOs), and repositories, the model layer interacts with the database or other data sources and forwards the processed data to the view layer for rendering. Utilizing the Thymeleaf template engine, the view layer renders the data as HTML, CSS, and JavaScript, and returns the static content to the front controller. Finally, the front controller sends the rendered content back to the browser, where it is displayed to the user, ensuring a seamless and coherent user experience.

2) Justification

The iPet system has multiple ways for customers to view and interact with data, such as through forms, images, and drop-down list. Given the uncertain and variable nature of future requirements for data presentation and interaction, the system requires a flexible and adaptable architectural pattern. Although the application of MVC pattern will possibly increase the complexity of the code, MVC is an appropriate choice for the iPet system as it separates the system into three distinct components. This modular approach enables independent development and modification of each component, which allows for changes to one component without affecting the others. Furthermore, the use of MVC in the iPet system will provide benefits such as better code reusability, faster development, and more efficient debugging and maintenance. By isolating the Model, View, and Controller, developers can work on specific aspects of the system without impacting other areas. This not only saves time and effort but also makes the system more resilient to changes in user requirements and external factors.

3) Browser/Server Pattern

Browser/Server (B/S) pattern can be considered as an improved version of Client/Server (C/S) structure. The three-layer architecture of B/S mode adopted by iPet online system is shown in Fig. 3, which is split into client browser, web server and database server respectively. Users send requests by browser to the server, followed by web server requests database to retrieve required data. Then retrieved data are flowed from database to the server which then responds data back to the browser. Ultimately, the contents are displayed on user's screen.

a) Justification

B/S architecture offers several advantages that make it a suitable choice for web application development. B/S pattern allows users to access the application through a web browser conveniently since the client-side component of B/S architecture is implemented using a web browser, which means there is no need to configure or install specific software on user's device so that the cost of ownership for end-users is significantly reduced.

Additionally, B/S provides cross-platform compatibility by using web browsers as the client-side

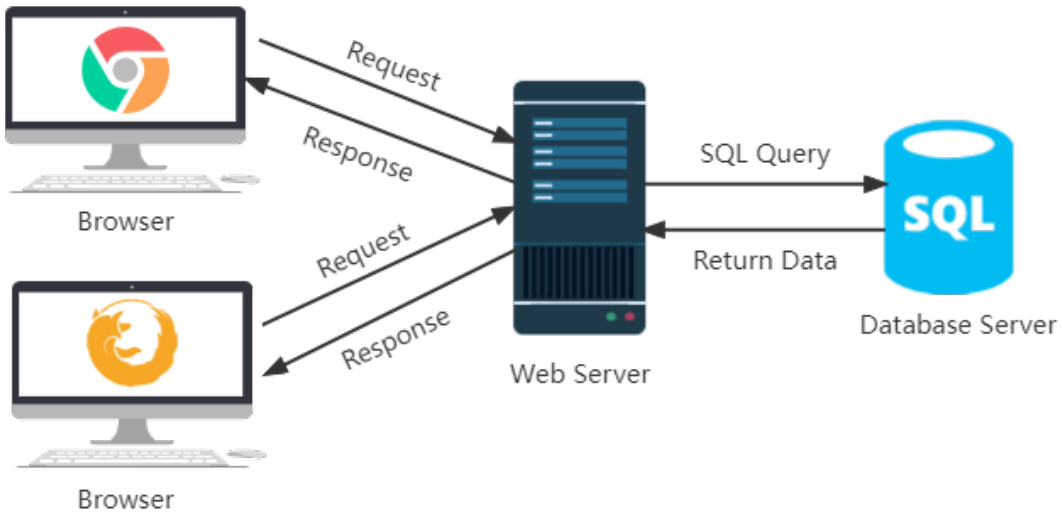


Fig. 3. The Browser/Server pattern

component of the application, which is available on a wide range of devices and operating systems, including Windows, Linux, iOS, etc. The function of application developed can be consistent across different operating systems and devices, making it easier for users to use, which is particularly important in today's world, where users use different devices and operating systems to access the internet. Furthermore, the architecture offers several security advantages. The server side of B/S structure can authenticate and authorize users to access certain parts of application. The web server can implement countermeasures toward common web attacks such as SQL injection. As shown in the Fig. 2, all requests and responses are processed through the web server, which can implement centralized logging and monitoring of security events. These advantages can help mitigate the risk of security incidents and protect the integrity and confidentiality of sensitive data.

B. System Module

The iPet Online Reservation System is broadly divided into five modules (User Authentication Module, Reservation Management Module, Service Display Module, Store Management Module and Data Analysis and Reporting Module, details in Fig. 4), each of which contains multiple components designed to be independent and Each module contains multiple components that are designed to be independent and encapsulated. This modular structure allows each component to operate autonomously while still functioning as an integral part of the overall system. This approach allows the system to be more maintainable, scalable, and upgradable.

The following is a more detailed decomposition of modularity and the rationale for each module:

a) User Authentication Module

- Components: ***Register, Login, Logout***.
- Rationale: This module concentrates on the authentication of users. Grouping all authentication-related functions within a single module can facilitate better integration and separation of concerns. These components are used to perform operations on user accounts, all of which relate to the core functionality of user authentication. By encapsulating the user authentication functionality in a separate module, we ensure that the security-related code is decoupled from the rest of the system. This separation not only facilitates updating or changing authentication methods without affecting other modules, but also allows for a more robust security implementation. As a result, developers can focus on enhancing user authentication without worrying about potential conflicts or dependencies with other parts of the system.

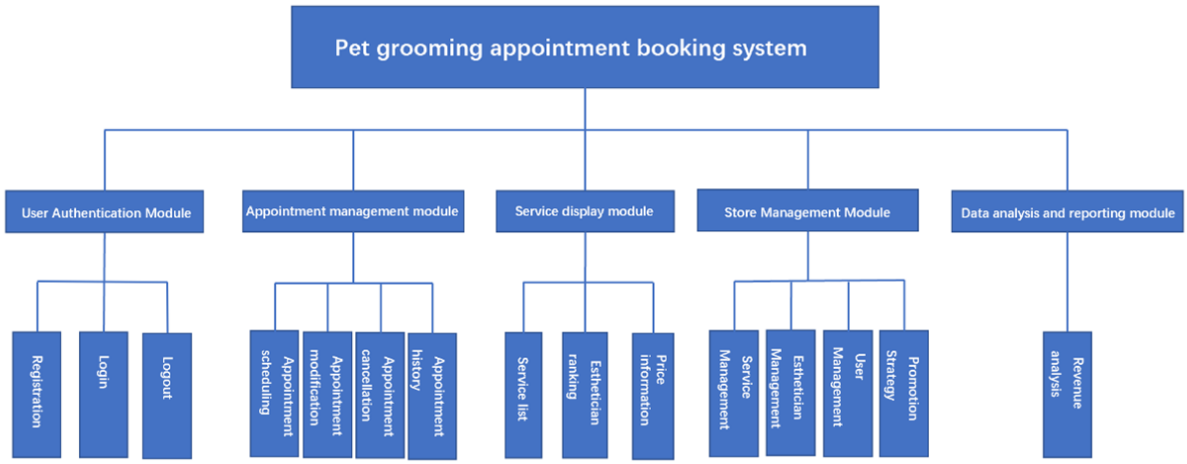


Fig. 4. System Module

b) Appointment Management Module

- Components: ***Appointment scheduling, appointment modification, appointment cancellation, appointment history***
- Rationale: This module concentrates on managing users' appointments. Grouping all appointment-related functions into a single module promotes better organization and separation of concerns. In addition, it simplifies the process of adding new appointment types, deleting existing appointment types, and modifying existing appointment rules, among other things. Furthermore, this encapsulation enables developers to work on appointment management features without affecting other modules, making the system more maintainable and efficient. Lastly, having a dedicated module for appointment management allows for streamlined updates and enhancements, ensuring a user-friendly experience and an easily adaptable system to cater to the ever-evolving requirements of the pet care industry.

c) Service Display Module

- Components: ***Service List, Groomer Ranking, Pricing Information***.
- Rationale: TThe Service Display Module is responsible for presenting the available services, groomer information, and pricing details to users. By maintaining the independence of this module, we can easily add, update, or remove services and groomer information without impacting other parts of the system. This separation not only ensures a clear division of responsibilities but also enhances the system's maintainability and adaptability. Additionally, having a separate Service Display Module enables the system to accommodate changing market trends or business requirements more efficiently. For instance, if a pet care business decides to introduce new services, update groomer qualifications, or modify pricing structures, these changes can be implemented in the Service Display Module without affecting the overall system stability.

d) Store Management Module

- Components: ***Service Management, Groomer Management, User Management, Promotional Strategies***
- Rationale: The focus of this module is to provide shop managers with tools to manage store information and promotional strategies. By separating these management functions from the user-facing modules, we can improve maintainability and allow for the addition of new management tools without impacting user functionality. This separation also streamlines the system architecture, ensuring that only relevant components are accessed by shop managers, thus enhancing system security and efficiency.

With a dedicated Shop Management Module, managers can easily monitor and adjust various aspects of the business, such as updating

service offerings, managing groomer schedules, handling user accounts, and tailoring promotional strategies to attract and retain customers. By encapsulating these management functions in a separate module, the system can better adapt to evolving business needs and seamlessly incorporate new features or requirements.

e) Data Analysis and Reporting Module:

- Components: ***Revenue Analysis***.
- Rationale: By modularizing data analysis and reporting functionalities, we ensure that data processing and analysis logic is separated from other modules. This separation enhances system performance, scalability, and maintainability, as changes to data analysis algorithms or reporting formats will not impact other parts of the system. The modular design of the Data Analysis and Reporting Module allows developers to focus on optimizing data processing and visualization techniques, resulting in more accurate and insightful analytics for shop managers. Furthermore, the encapsulation of this module makes it easier to update or add new analysis methods, data sources, or reporting templates, thus accommodating the changing needs of the business and industry trends. Additionally, having a dedicated Data Analysis and Reporting Module ensures the secure handling and storage of sensitive business data, as access to this module can be limited to authorized personnel only, reducing the risk of data breaches. In summary, by modularizing the software, we create a well-structured, organized system that improves maintainability, scalability, and extensibility. Modularity also encourages code reusability and allows concurrent development and integration of different modules, reducing interdependencies and improving overall development efficiency.

C. High-level database design

Our database design uses a relational database management system MySQL to ensure data integrity and consistency. The database contains multiple tables used to store different types of data. The relationships between these tables are maintained through foreign key constraints.

Fig. 5 below shows an entity-relationship diagram (ERD) of the overall database design:

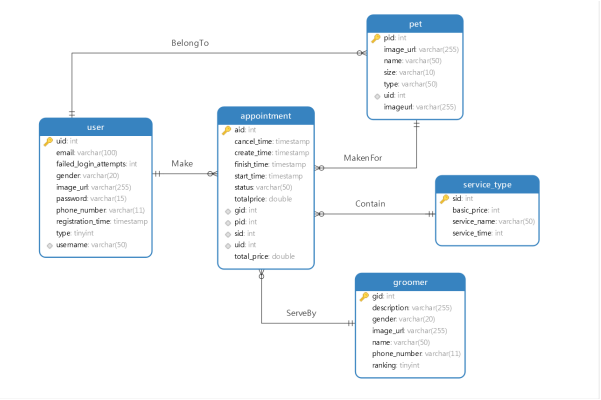


Fig. 5. The ER Diagram

1) Entity

Our database contains five entities: user, appointment, pet, serviceType, and groomer. The user entity represents the customers who book appointments, the appointment entity represents the appointments booked by customers for their pets, the pet entity represents the pets owned by customers, the serviceType entity represents the types of services available for pets, and the groomer entity represents the groomers who perform the services.

2) Attribute

Each entity in our database has its own set of attributes. The user entity has attributes such as uid, name, email, and phone. The appointment entity has attributes such as aid, date, time, notes, uid, pid, sid, and gid. The pet entity has attributes such as pid, name, species, breed, and uid. The serviceType entity has attributes such as sid, name, description, and price. The groomer entity has attributes such as gid, name, email, and phone.

3) Relationship

Our database has several relationships between the entities. The pet entity has a foreign key uid that references the uid field of the user entity, which represents the owner of the pet. The appointment entity has foreign keys uid, pid, sid, and gid that reference the uid, pid, sid, and gid fields of the user, pet, serviceType, and groomer entities respectively. These foreign keys allow us to track which user booked the appointment, which pet the appointment is for, which service was booked, and which groomer will perform the service. The pet entity also has a one-to-many relationship with the appointment entity, as one appointment can be for one pet, but one pet can

have multiple appointments. The serviceType entity has a one-to-many relationship with the appointment entity, as one service can be booked for multiple appointments, but one appointment can only have one service booked. Finally, the groomer entity has a one-to-many relationship with the appointment entity, as one groomer can perform services for multiple appointments, but one appointment can only have one groomer assigned.

III. Software Design

A. High-level Design

The iPet online appointment booking system is modularized of interconnected modules designed for ease of understanding, use, and maintenance while effectively implementing all critical features and functionalities in our system.

The following flow chart illustrates the relationship between each module and the workflow of the whole system:

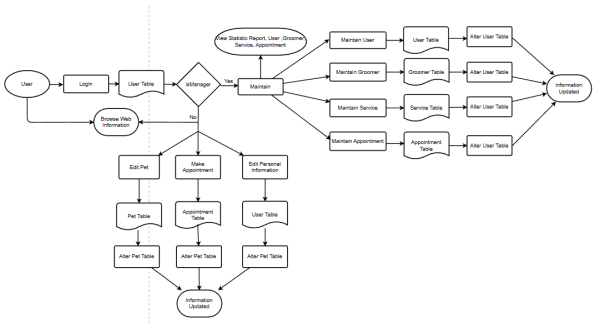


Fig. 6. Flow chart of the whole system

In Fig.6, the user initially enters the webpage and can navigate through all the web information available, including the various service types, details about each groomer, and any current promotions. To place an order, the user must log in, and the system will identify whether the user is an administrator or a customer based on their user type.

If the user is a customer, they can access the personal information page, the pet information page, and the order page. Here, the user can modify their personal information and pet information, as well as cancel orders. When placing an order, the user can select their pet, desired service, service time, and groomer. The user can choose to pay either online or offline. On the order page, the user can also view information about each groomer, and the system will recommend a groomer based on the user's previous order history.

If the user is a shop manager, they will enter the maintain page. Here, they can view all user, groomer, and pricing plan information, as well as appointment details. The administrator can modify, add, or delete groomers and pricing plans, as well as change order status. The administrator can also search for users and orders by username, groomers by name, and pricing plans by service name.

Any changes made are immediately synced to the server database and displayed on the page. The flow chart clearly shows all the processes of using this website, hoping to help you better understand the IPet system.

B. Software Support Service

To enhance functionality and user experience of the system. We provide support services both Internal and external to the system. For the internal support, the iPet system offers an intuitive and user-friendly interface, with well-structured webpage navigation services to guide users through the booking process. These services include clear menu options, breadcrumb navigation, and informative error messages to enhance the user experience.

Webpage navigation examples are shown in Fig. 7, Fig. 8:

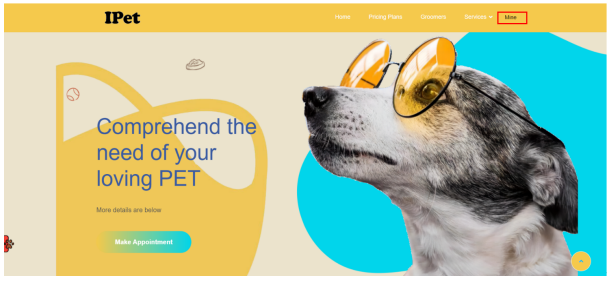


Fig. 7. The Navigation Menu on The Home Page

As it is illustrated, when users want to access their personal information, they can conveniently click on the "Mine" option in the navigation menu on the top. This action will swiftly direct them to their personalized "My Account" page, where they can securely view and manage their personal details. The user navigates very smoothly to the desired page in this case.

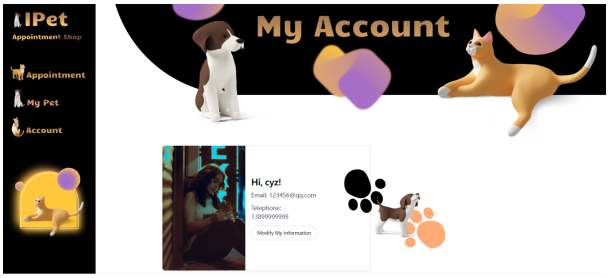


Fig. 8. The Navigation Result

Also, to protect user data and maintain system integrity, we have implemented advanced security measures such as encryption, authentication, access control and front-end form validation. These services safeguard sensitive information, ensuring secure transactions and data storage.

Webpage navigation examples are shown in Fig. 9:

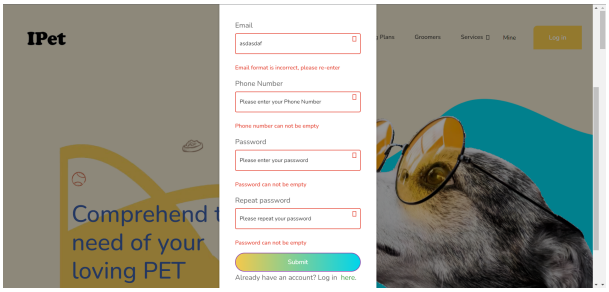
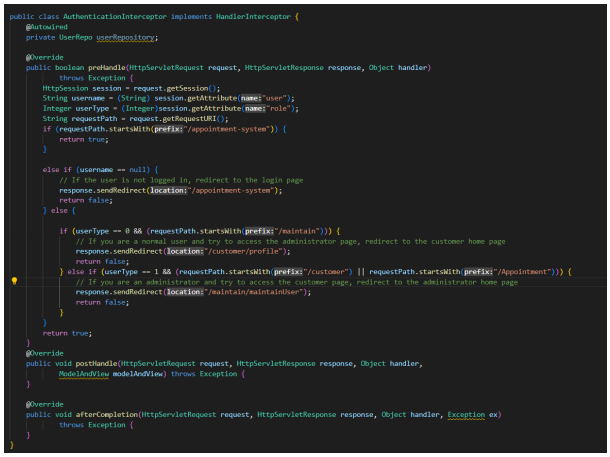


Fig. 9. The Form Validation for Registering

When registering, the user implemented that each input field in the form can have corresponding validation rules, the email address must conform to a valid email format, and the phone number need to be in 11-digit number and first digit must start with the number 1 and second digit can be varied from 3, 4, 5, 6, 7, 8 or 9. These digits indicate the different operators and types of services respectively. The rest nine digits can be any number, from 0 to 9. Also, the password must meet specific complexity requirements like password must contain letters and numbers, between 6-18 characters. In addition, because the password stage requires confirmation, the user needs to enter the password again to ensure that the two entries match. Any form input that does not match the requirements will be blocked. And the input box for the corresponding input field will be highlighted in red to remind the user to enter the correct information.

The interceptor example is shown in Fig. 10:



```
public class AuthenticationInterceptor implements HandlerInterceptor {
    @Autowired
    private UserRepo userRepository;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
            throws Exception {
        HttpSession session = request.getSession();
        String username = (String) session.getAttribute(name:"user");
        Integer userType = (Integer)session.getAttribute(name:"role");
        String requestPath = request.getRequestURI();
        if (requestPath.startsWith(prefix:"/appointment-system")) {
            return true;
        }

        else if (username == null) {
            // If the user is not logged in, redirect to the login page
            response.sendRedirect(location:"/appointment-system");
            return false;
        } else {

            if (userType == 0 && (requestPath.startsWith(prefix:"/maintain"))) {
                // If you are a normal user and try to access the administrator page, redirect to the customer home page
                response.sendRedirect(location:"/customer/profile");
                return false;
            } else if (userType == 1 && (requestPath.startsWith(prefix:"/customer") || requestPath.startsWith(prefix:"/Appointment"))) {
                // If you are an administrator and try to access the customer page, redirect to the administrator home page
                response.sendRedirect(location:"/maintain/maintainUser");
                return false;
            }
        }
        return true;
    }
    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
            ModelAndView modelAndView) throws Exception {
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
            throws Exception {
    }
}
```

Fig. 10. The Sample Code of Interceptor

The above code is an Interceptor class in the Spring Boot framework, used for authentication and permission control when users access the web application. Within the class, There are three override methods, preHandle(), postHandle(),afterCompleion().

PreHandle() is called before the controller processes the request, postHandle() is called after the controller method returns but before the view is rendered, and afterCompletion() is executed after the view has been rendered and is used to perform resource cleanup.

In the preHandle method, the username and user type information stored in the current session is obtained, as well as the URL path of the current request. If the request path starts with ”/appointment-system”, it means the user is accessing the login page and the request is released, return true. If the username is empty which means the session do not store user’s information, the request is redirected to the login page and return true that Force users to log in first. If the username is not empty, the user is logged in, determine the user type which includes managers and customers. If it is a customer (0) and trying to access the maintain page (starting with ”/maintain”), redirect the request to the client’s home page and return false. If user type is administrator (1) and an attempt is made to access a customer page or an appointment page (starting with ”/customer” or ”/Appointment”), the request is redirected to the administrator’s home page and false is returned. If none of the above

conditions are met, the request is released and true is returned.In summary, this method checks if a user is logged in when they access the web application and controls the user's access rights based on the user type and request path. This code fragment shows an interceptor method that keep customers and shop managers from jumping to the wrong pages.

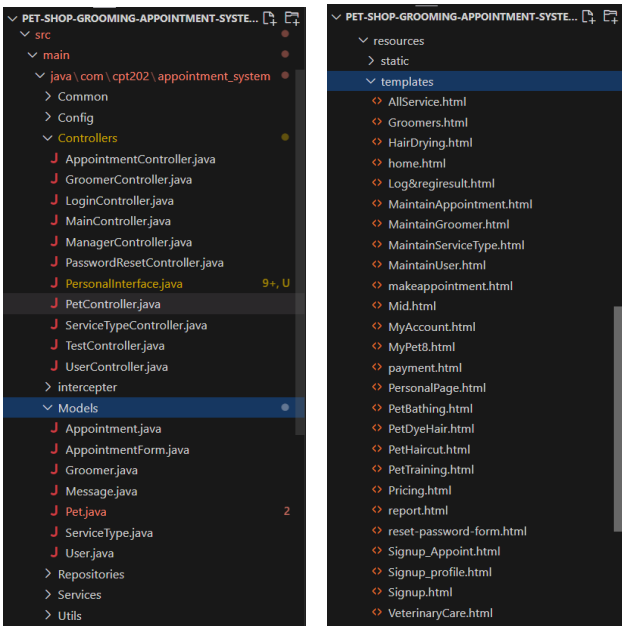
C. Code Structure and Convention

In the iPet online appointment booking system, we have adopted a clear and consistent code structure and convention to ensure that the project is easy to understand, maintain, and scale.

1) Code Structure

The code structure is based on the Model-View-Controller (MVC) architecture as we mentioned previously in Fig. 2, which separates the application logic into three interconnected components.

The MVC pattern can be used as a coding structure to help developers organize and manage code and improve maintainability and extensibility of code.



(a) Model and Controller (b) View

Fig. 11. code structure

- **In Model folder**: We have models for all the important entities such as appointment, groomer, message, pet and user which is shown in Fig. 11(a). It is directly mapping to the database and several tools is also provided.
- **In View folder**: Fig. 11(b) shows the structure of the view part. We provide the designed html file in the resource folder which is located in the same directory as the model folder.
- **In Controller folder**: In this project, all the models mentioned have a corresponding controller and for some functions that is not based on the model, we build an individual controller for them.

2) Code Convention

As for coding conventions, our team has established a set of rules and guidelines that all members are required to follow to ensure consistency across the codebase. These conventions include:

- **Naming conventions**: We use clear, descriptive, and concise names for variables, functions, classes, and files. Additionally, we follow the camelCase naming convention for variables and functions and PascalCase for classes.
- **Code formatting**: Consistent indentation and spacing are used throughout the code, with each level of indentation being four spaces.
- **Comments**: Inline comments are used to explain complex or non-intuitive code sections, while block comments are used to provide a high-level overview of a module or function.
- **Error handling**: Proper error handling is implemented with try-catch blocks, ensuring that the system remains stable even in case of unexpected errors or exceptions.

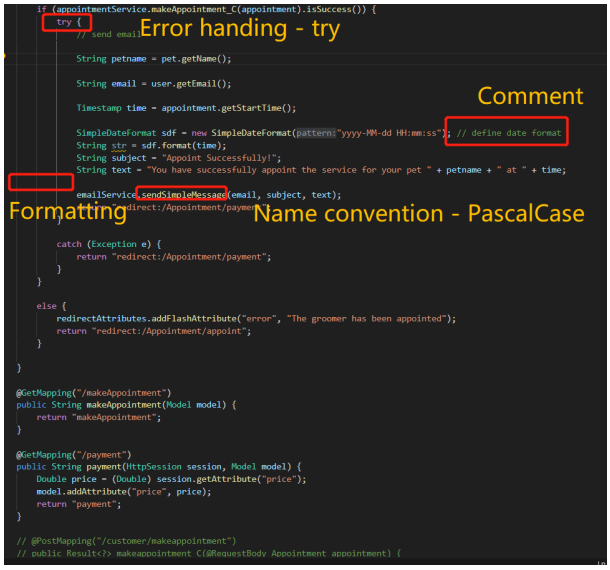


Fig. 12. The Form Validation for Registering

By adhering to these code structure and con-

vention guidelines, our team has developed a well-organized, easily understandable, and efficient codebase for the iPet online appointment booking system.

D. Software Configuration and Production Environment

1) Software Configuration

The software configuration consists of the following components:

- **Frontend**: HTML, CSS, and JavaScript are used to create a responsive and user-friendly interface for customers and shop managers. Java Script and Thymleaf templates are employed to enable asynchronous communication between the client-side and the server-side, enhancing the overall user experience.
- **Backend**: The application is built on the Spring Boot framework, utilizing the MVC architecture. This allows for the separation of concerns, making the codebase modular, maintainable, and scalable.
- **Database**: MySQL is chosen as the relational database management system to store and manage data efficiently. The database schema is designed to support the required functionality, such as user registration, appointment booking, and service management.

2) Product Environment

Server Configuration: The application is deployed on a reliable and high-performance server-AliyunServer to guarantee the system's availability and responsiveness. The server is configured to handle the anticipated load and traffic. The product environment is constructed on Java 16.0.2, MySql Community 8.0.30 on the 64-bit operating system - Windows Server 2022 Datacenter 21H2, with x64-based processor called Intel(R) Xeon(R) Platinum 8269CY CPU @ 250GHz2.50 GHz. The system also installed RAM 4.00 GB.

IV. Software Testing

The coverage of all the three kinds of tests we have done is 100% of this project. The pass rate is 100%. The following will be given an illustration of the three tests of our project. The detailed codes will be given in the **Appendix**.

A. Unit Testing

In the unit test of our project, we applied @SpringBootTest as the test framework, the Junit and Mockito for individual Java function as our tester.

1) Test Process

- Identify the critical functions within each module.
- Create test cases with specific input and expected output for each function.
- Execute the test cases and compare the actual output with the expected output.
- Document the test results and fix any identified issues.

2) Test Data Results

Below, shown in Fig. 13, is a sample unit test of the E-mail validation capability of the registration process

Test Case	Input	Expected output	Actual output	Result
Verify valid email format	Empty	Email cannot be empty	Email cannot be empty“	Pass
	12345@ **(wrong format)**	Email format is incorrect, please re-enter	Email format is incorrect , please re-enter	Pass
	usedEmail@used.com **(Email has been used)**	Email already exists	Email already exists	Pass
	rightEmail@right.com **(Right format, haven't been used before)**	The input box accepting email field turns green	The input box accepting email field turns green	Pass

Fig. 13. Test Data Results in Unit Test

In this example, we thoroughly tested the form validation for username input in the registration module by strictly following the test process. Same approach is applied to all functions of this project when conducting the unit test. The unit tests cover all the individual function in our project and the pass rate is 100%, ensuring the proper functioning of all system components and modules.

B. Integration Testing

In the integration test of our project, we applied Postman, Junit and Mockito as our tester for integration testing.

1) Test Process

- Identify the critical interactions and interfaces between modules.
- Develop test scenarios to verify the integration of the system components.
- Execute the test scenarios and evaluate the system's behavior.
- Document the test results and resolve any detected issues.

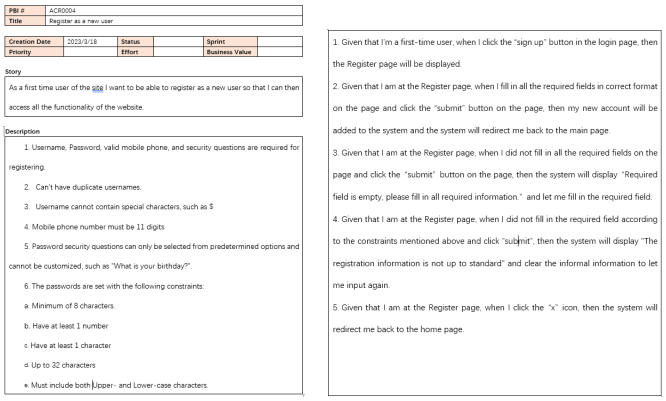
2) Test Data Results

Let us focus on the interaction between the appointment making process and the notification system as an example.

Test scenarios	Test inputs	Expected behaviors	Actual behaviors	Result
Create a new appointment and verify notification	Valid appointment details (date, time, service)	Notification is sent to the user and groomer upon booking	Notification is sent to the user and groomer upon booking	Pass
Update an existing appointment and verify notification	Updated appointment details (new date, time)	Updated appointment details (new date, time)	Updated appointment details (new date, time)	Pass
Cancel an appointment and verify notification	Click "Cancel" button	Notification is sent to the user and groomer upon cancellation	Notification is sent to the user and groomer upon cancellation	Pass
Verify notification is not sent for invalid appointment	Invalid appointment details (past date)	No notification is sent as the appointment is not created	No notification is sent as the appointment is not created	Pass
Verify notification is not sent for an appointment with an unavailable groomer	Appointment details with unavailable groomer	No notification is sent as the appointment cannot be created	No notification is sent as the appointment cannot be created	Pass

Fig. 14. Test Data Results in Integration Test

In this test, we've examined the integration of appointment booking and notifications, covering the creation, updates, cancellations, and invalid scenarios. The actual outcomes aligned with the expected, indicating successful integration testing.



(a) Part of Example PBI (b) Part of Example PBI

Fig. 15. Example PBI

C. Acceptance Testing

Acceptance testing evaluates the system's performance against the user requirements, ensuring that it meets the expectations of end-users and stakeholders.

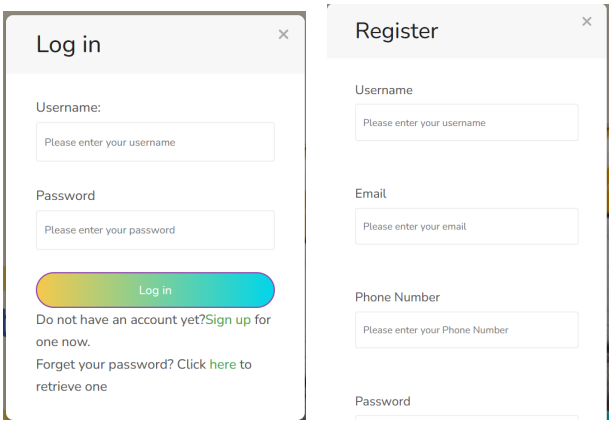
1) Test Process

- Develop test scenarios based on user requirements and use cases.
- Perform the test scenarios with the involvement of end-users or stakeholders, when possible.
- Collect feedback from the users or stakeholders and analyze the test results.
- Address any issues and improve the system based on the feedback.

2) Test Data Results

We will use the user registration as an example to show the results of our acceptance test. In this case, we will firstly show the PBI description and then do the acceptance test according to our PBI acceptance criteria, which you can find the details in Fig. 15.

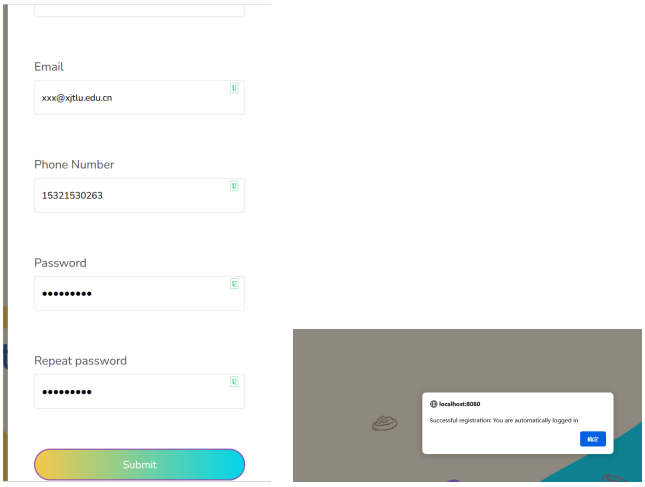
- Criteria 1 :Click sign up and jump to Register page.



(a) Criteria 1 Test - Login (b) Criteria 1 Test - Register

Fig. 16. Criteria 1 Test

- Criteria 2: Fill in all the field in correct format and submit.



(a) Criteria 2 Test - Register Form Validation (b) Criteria 2 Test - Register Result

Fig. 17. Criteria 2 Test

- Criteria 3: Do not fill in all the field.

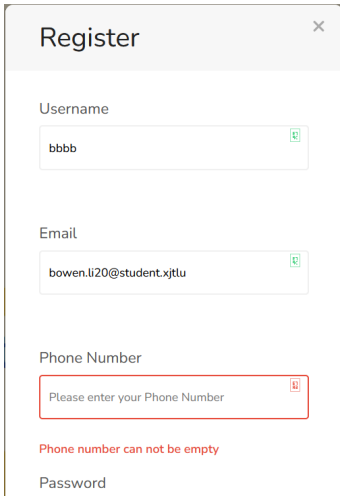


Fig. 18. Criteria 3 Test

- Criteria 4: did not fill in the required field according to the constraints mentioned above.

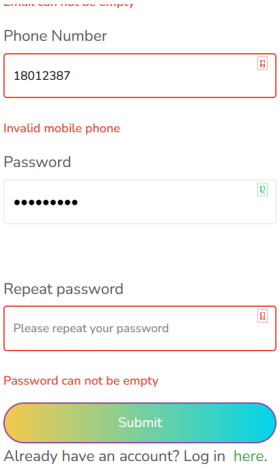


Fig. 19. Criteria 4 Test

- Criteria 5: click “x” icon and back to home.

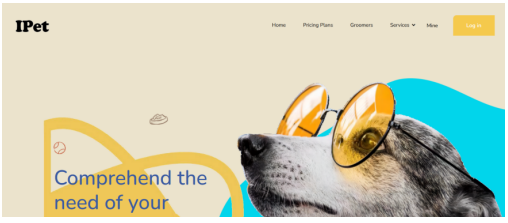


Fig. 20. Criteria 5 Test

The results of the test in Fig.21 show that the functionality of our system is fully compliant with the acceptance criteria of user registration as what is described in the PBI report we wrote previously.

Test scenarios	Test inputs	Expected behaviors	Actual behaviors	Result
Access registration page	Click on “sign up”	Registration form is opened	Registration form is opened	Pass
Register for an account	Name, Email, phone number,Password, Confirm Password in correct form	Account is created successfully and login to the home page directly	Account is created successfully and login to the home page directly	Pass
Didi not fill all the filed and click “submit button”	Name, Password, phone number Confirm Password in correct form without e-mail	Red warning hint will be displayed	Red warning hint will be displayed	Pass
Fill phone number in incorrect form	Name, Email, Password, phone number in incorrect way Confirm Password in correct form	Red warning hint will be displayed	Red warning hint will be displayed	Pass
Want to back to home page	Click “x” iccon	Back to home page	Back to home page	Pass

Fig. 21. Test Data Results in Acceptance Testing

The system full-fill the user requirement and the business expectation of the stake holders.

By conducting comprehensive testing and addressing the identified issues, the iPet system is continuously improved to deliver a reliable and efficient appointment management solution for the pet grooming industry.

Appendix

- **Sprint backlog**

Sprint 1 :
Period: 2023/03/20 - 2023/04/05

PBI 1	Registration		
Story	As a new user, I want to be able to register for an account by providing my personal information and creating login credentials, so that I can gain access to the system of being a registered user.		
Task #	Task Description	Effort Hours	Assign To
1	Design user registration interface	4	Leduo Chen
2	Implement user registration function	5	Leduo Chen
3	Verify user input email	3	Leduo Chen
4	Verify user input phone	3	Leduo Chen
5	Write test cases for user registration function	4	Leduo Chen

6	Test user registration function and fix issues	3	Leduo Chen

PBI 2	Login		
Story	As a registered user, I want to provide my login credentials to log in to my account, so that I can access the system and utilize the features and information specific to my account.		
Task #	Task Description	Effort Hours	Assign To
1	Validate user input username	3	Leduo Chen
2	Validate user input password	3	Leduo Chen
3	Write test cases for the user login function	2	Leduo Chen
4	User password retrieval function	4	Leduo Chen
5	Write test cases for the user login function	4	Leduo Chen
6	Test the user login function and fix problems	3	Leduo Chen

PBI 3	Retrieve password		
Story	As a registered user, I want to be able to retrieve/reset my password in case I forget it or need to change it for security reasons.		
Task #	Task Description	Effort Hours	Assign To
1	Design the password retrieval interface	3	Shuchen Yuan
2	Implement the password recovery function	4	Shuchen Yuan
3	Validate user input email	5	Shuchen Yuan
4	Validate user input phone number	4	Shuchen Yuan
5	Write test cases for the password retrieval function	3	Shuchen Yuan
6	Test the password recovery function and fix problems	2	Shuchen Yuan

PBI 4	Edit personal information		
Story	As a registered user, I want to have the ability to modify my personal information by accessing a dedicated page or form, so that I can update any changes to my personal details.		
Task #	Task Description	Effort Hours	Assign To
1	Design the interface for modifying personal information	5	Suqi Zhang
2	Implement the function to modify personal information	4	Suqi Zhang
3	Verify user input email	4	Suqi Zhang
4	Validate user input phone number	3	Suqi Zhang
5	Write test cases for the personal information modification function	2	Suqi Zhang
6	Test the function of modifying personal information and fix problems	3	Suqi Zhang

PBI 5	User View Pet List

Story	As a customer, I want to be able to view a list of my registered pets, so that I can easily manage and keep track of their grooming needs.		
Task #	Task Description	Effort Hours	Assign To
1	Design Pet List Interface	5	Juntuo Wang
2	Implementing the pet list function	3	Juntuo Wang
3	Write test cases for the pet list function	2	Juntuo Wang
4	Validate user input phone number	4	Juntuo Wang
5	Test the pet list function and fix problems	3	Juntuo Wang
6	Test the function of modifying personal information and fix problems	4	Juntuo Wang

PBI 6	Adding new pets by users		
Story	As a customer, I want to be able to add new pets to my account, so that I can manage and schedule grooming appointments for them.		
Task #	Task Description	Effort	Assign To

		Hours	
1	Designing the interface for adding new pets	5	Suqi Zhang
2	Implementing the Add New Pet function	4	Suqi Zhang
3	Validate user input (pet information)	3	Suqi Zhang
4	Writing test cases for adding new pets	2	Suqi Zhang
5	Test the add new pet function and fix problems	3	Suqi Zhang

PBI 7	Modifying pet information by users		
Story	As a customer, I want to have the ability to modify the information of my pets, so that I can keep their details up to date and accurately reflect any changes or updates.		
Task #	Task Description	Effort Hours	Assign To
1	Designing the interface for modifying pet information	5	Juntuo Wang
2	Implementing the edit function	4	Juntuo Wang
3	Validate user input (pet	3	Juntuo Wang

	information)		
4	Write test cases for the function of modifying pet information	2	Juntuo Wang
5	Test the function of modifying pet information and fix problems	3	Juntuo Wang

PBI 8	User Delete Pet		
Story	As a customer, I want to be able to delete a pet from my account, so that I can remove pets that no longer require grooming or are no longer in my care.		
Task #	Task Description	Effort Hours	Assign To
1	Design the interface of Delete Pet function	5	Yuzhen Chen
2	Implement the pet deletion function	4	Yuzhen Chen
3	Write test cases for the pet deletion function	3	Yuzhen Chen
4	Test the pet deletion function and fix the problem	2	Yuzhen Chen

PBI 9	User View Service List

Story	As a customer, I want to be able to view the list of available grooming services, so that I can explore the options and make informed decisions for my pets' grooming needs.		
Task #	Task Description	Effort Hours	Assign To
1	Designing the Service List Interface	2	Shuchen Yuan
2	Implementing the service list function	1	Shuchen Yuan
3	Write test cases for the service list function	4	Shuchen Yuan
4	Test the service list function and fix problems	5	Shuchen Yuan

PBI 10	Make appointment		
Story	As a logged-in customer, I want to make appointment at Make appointment page so that I can enjoy the service I need.		
Task #	Task Description	Effort Hours	Assign To
1	Find all pets owned by this user from the database	2	Bowen.Li
2	Render the size and name of each pet in the found petlist to the front-end pet drop-down menu	5	Bowen.Li
3	Find all existing service and package from the database	3	Bowen.Li

4	Render the basic price and name of each service in the found service list to the front-end service drop-down menu	4	Bowen.Li
5	Find all existing Groomers from the database	4	Bowen.Li
6	Render the rank and name of each groomer in the found groomer list to the front-end service drop-down menu	5	Bowen.Li
7	Fill in the time and calculate the finish time	2	Bowen.Li
8	Calculate the total price according to the pet size, service type, and groomer's rank.	4	Bowen.Li
9	Determine if the groomer has been booked at this time	3	Bowen.Li
10	Insert the newly made appointment to the database	2	Bowen.Li
11	Click make appointment button to call all the above function	6	Bowen.Li

PBI 17	View groomer list		
Story	As a shop manager, I want to have a groomer list, so that I can check the information of groomers quickly.		
Task #	Task Description	Effort Hours	Assign To
1	Design the groomer list part on the webpage	4	Leduo Chen
2	Implement the groomer list on the webpage	3	Leduo Chen
3	Get the groomer list from the	2	Leduo Chen

	database		
4	Render the groomer list on the html webpage	2	Leduo Chen

PBI 18	Add new groomer		
Story	As a shop manager, I want to be able to add new groomers to the system so that I can expand the number of staff.		
Task #	Task Description	Effort Hours	Assign To
1	Design the add groomer part on the webpage	5	Shuchen Yuan
2	Implement add groomer part on the webpage	4	Shuchen Yuan
3	Send the new groomer added to backend	3	Shuchen Yuan
4	Add the new groomer to database	2	Shuchen Yuan
5	Restrict that groomer name, gender, and rank to be must-fill	2	Shuchen Yuan

	field		

PBI 19	modify existing groomer		
Story	As a shop manager, I want to modify groomer information so that I can manage the information of a groomer.		
Task #	Task Description	Effort Hours	Assign To
1	Design modify groomer part on the webpage	5	Xi Kong
2	Implement modify groomer part on the webpage	4	Xi Kong
3	Restrict that all the fields of the updated groomer must be filled in the webpage	2	Xi Kong
4	Send the modified groomer to backend	2	Xi Kong
5	Update the groomer's information in the database	3	Xi Kong

PBI 20	Delete groomer

Story	As a manager, I want to be able to delete a groomer so that I can remove his/her record after his/her dismiss.		
Task #	Task Description	Effort Hours	Assign To
1	Design the delete groomer part on the webpage	4	Yihan Zhou
2	Implement delete groomer part on the webpage	4	Yihan Zhou
3	Send the groomer to be deleted to the backend	2	Yihan Zhou
4	Delete the groomer record in the database	3	Yihan Zhou

Sprint 2:
Period: 2023/04/06 - 2023/04/20

PBI 11	User View Groomer List		
Story	As a customer, I want to be able to view the list of available grooming services, so that I can choose the services that best suit the needs of my pets.		
Task #	Task Description	Effort	Assign To

		Hours	
1	Designing esthetician list interface	5	Xi Kong
2	Implementing the groomer list function	4	Xi Kong
3	Write test cases for the groomer list function	3	Xi Kong
4	Test the groomer list function and fix problems	3	Xi Kong

PBI 12	User view groomer details		
Story	As a customer, I want to be able to view detailed information about the groomers, so that I can learn more about their qualifications, experience, and specialties.		
Task #	Task Description	Effort Hours	Assign To
1	Designing groomer details interface	5	Xi Kong
2	Implementing groomer details	4	Xi Kong
3	Write test cases for the esthetician details function	3	Xi Kong
4	Test the Groomer detail function and fix problems	3	Xi Kong

PBI 13	User search for groomers based on keywords		
Story	As a customer, I want to be able to search for groomers based on specific keywords or criteria, so that I can quickly find groomers who specialize in certain breeds, services, or locations.		
Task #	Task Description	Effort Hours	Assign To
1	Design the interface of the groomer search function	5	Yihan Zhou
2	Implementing the groomer search function	4	Yihan Zhou
3	Test the groomer search function and fix the problems	3	Yihan Zhou

PBI 14	Cancel appointment by user		
Story	As a customer, I want to have the ability to cancel appointments that I have scheduled, in case of emergencies or changes in my schedule.		
Task #	Task Description	Effort Hours	Assign To
1	Designing the interface of appointment history cancel	5	Suqi Zhang

	function		
2	Implementing the cancel appointment function	4	Juntuo Wang
3	Writing test cases for the cancel function	3	Juntuo Wang
4	Test the cancel appointment function and fix problems	2	Suqi Zhang

PBI 15	See appointment status by users		
Story	As a customer, I want to be able to view the status of my appointments, so that I can stay informed about the progress and confirmation of my scheduled grooming appointments.		
Task #	Task Description	Effort Hours	Assign To
1	Designing the interface of the function of viewing appointment status	5	Yuzhen Chen
2	Implementing the function of viewing appointment status	4	Yuzhen Chen
3	Write test cases for the function of viewing appointment status	3	Yuzhen Chen
4	Test the function of viewing	2	Yuzhen Chen

	appointment status and fix problems		

PBI 16	User search for appointment history based on keywords		
Story	As a customer, I want to be able to search for my appointment history using specific keywords or criteria, so that I can quickly find and review past grooming appointments.		
Task #	Task Description	Effort Hours	Assign To
1	Implementing the search history function	5	Juntuo Wang
2	Implementing the function of viewing appointment status	4	Suqi Zhang
3	Test the search history function and fix the problems	3	Juntuo Wang
4	Test the function of viewing appointment status and fix problems	3	Suqi Zhang

PBI 11	Make appointment(add new task based on the previous to make some Optimisation the old ones that do not need to be modified aren't displayed)

Story	As a logged-in customer, I want to make appointment at Make appointment page so that I can enjoy the service I need.		
Task #	Task Description	Effort Hours	Assign To
1	Determine if the user has filled in the required fields and give a hint	3	Bowen.Li
2	The price calculation formula is displayed above the makeappointment button	5	Bowen.Li
3	When we press the make appointment button we are redirected to the payment page if the order is placed successfully.	4	Bowen.Li
4	When we press back button, it redirect me to the home page	3	Bowen.Li

PBI 21	Search groomer		
Story	As a shop manager, I want to be able to search a groomer by name so that I can quickly find him/her to get his/her basic information.		
Task #	Task Description	Effort Hours	Assign To
1	Design the search groomer part on the webpage	5	Yihan Zhou
2	Implement the search groomer part on the webpage	4	Yihan Zhou
3	Restrict that fuzzy searching	2	Yihan Zhou

	groomer using by the name		
4	Send the search key words typed in the search bar to the backend	3	Yihan Zhou
5	Get resulting groomers from database	2	Yihan Zhou
6	render the retrieved groomers from database on the webpage	1	Yihan Zhou

PBI 22	View customer list		
Story	As a shop manager, I want to have a list of all customers so that I can view both recently registered customers and customers who have signed up for a long time.		
Task #	Task Description	Effort Hours	Assign To
1	Design the customer list part on the webpage	4	Yuyi Yang
2	Implement the customer list on the webpage	3	Yuyi Yang
3	Get the customer list from the database	2	Yuyi Yang

4	Render the customer list on the html webpage	2	Yuyi Yang

PBI 23	Search customer		
Story	As a shop manager, I want to be able to search customer so that I can quickly find the customer who I want to look for.		
Task #	Task Description	Effort Hours	Assign To
1	Design the search customer part on the webpage	5	Yuyi Yang
2	Implement the search customer part on the webpage	4	Yuyi Yang
3	Restrict that fuzzy searching customer using by the username	3	Yuyi Yang
4	Send the search key words typed in the search bar to the backend	3	Yuyi Yang
5	Get resulting customers from database	4	Yuyi Yang
6	render the retrieved customers from database on the webpage	2	Yuyi Yang

PBI 24	View appointment list		
Story	As a shop manager, I want to have an appointment list so that I can monitor pet shop business conditions and maintain them in a timely manner.		
Task #	Task Description	Effort Hours	Assign To
1	Design the appointment list part on the webpage	5	Yuyi Yang
2	Implement the appointment list on the webpage	4	Yuyi Yang
3	Get the appointment list from the database	3	Yuyi Yang
4	Sort the appointment list by chronological order	3	Yuyi Yang
5	Render the appointment list on the html webpage	2	Yuyi Yang

Sprint 3:
Period: 2023/04/20 - 2023/05/03

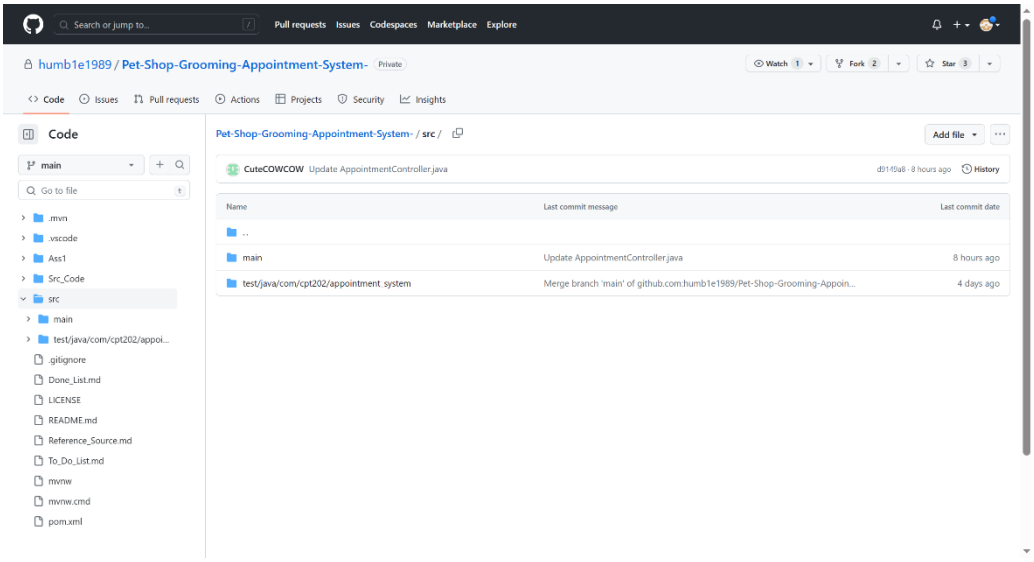
PBI 11	Make appointment (add new task based on the previous to make some Optimisation the old ones that do not need to be modified aren't displayed)		
Story	As a logged-in customer, I want to make appointment at Make appointment page so that I can enjoy the service I need.		
Task #	Task Description	Effort Hours	Assign To
1	Add new page at make appointement page	3	Bowen.Li
2	Display the total price directly at the button	2	Bowen.Li
3	Determining that the same pet cannot be served at the same time	5	Bowen.Li
4	Add a pop-up message for unsuccessful appointments	6	Bowen.Li
5	Recommend groomers based on rank	4	Bowen.Li
6	Recommend service base on you appointment amount	6	Bowen.Li
7			

PBI 25	Add new service		
Story	As a manager, I want to be able to add a new service so that I can maintain selling strategies as I want.		
Task #	Task Description	Effort Hours	Assign To

1	Design the add service part on the webpage	5	Yuzhen Chen
2	Implement add service part on the webpage	4	Yuzhen Chen
3	Send the new service added to backend	2	Yuzhen Chen
4	Add the new service to database	1	Yuzhen Chen
5	Restrict that service name, time, and price to be must-fill field	3	Yuzhen Chen

- **Detailed code**

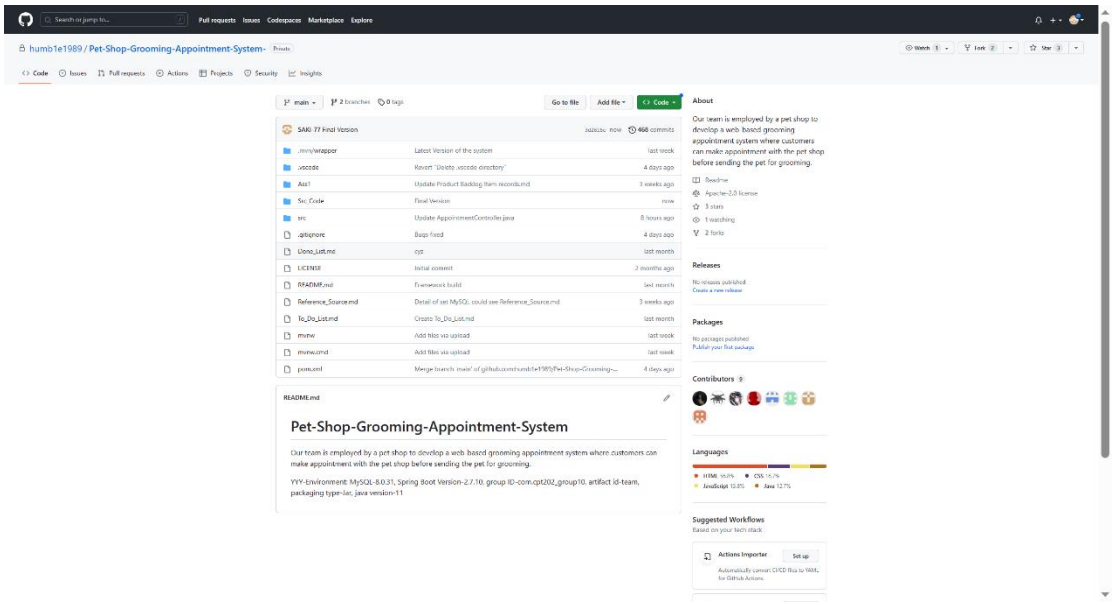
About the detail code of our project for developing and testing, you can find them on https://github.com/humb1e1989/Pet-Shop-Grooming-Appointment-System-/tree/main/src.



In this link, the “main” folder is the codes for developing, the “test” folder is the codes for testing.

- **Github link**

Our projects are all based on Github and the full picture of our project will be found here.
https://github.com/humb1e1989/Pet-Shop-Grooming-Appointment-System-



- **Individual contribution form**

CPT202 Assignment 1

Individual Contribution for Group Report and Presentation

Group Number:

Name	ID Number	Contribution (%)
1. **Xi Kong**	**2037023**	11.1%
2. **Shuchen Yuan**	**2036501**	11.1%
3. **Leduo Chen**	**2035836**	11.1%
4. **Yihan Zhou**	**2033677**	11.1%
5. **Juntuo Wang**	**2036505**	11.1%
6. **Suqi Zhang**	**2033871**	11.1%
7. **Yuzhen Chen**	**2036328**	11.1%
8. **Yuyi Yang**	**2036676**	11.1%
9. **Bowen Li**	**2033827**	11.1%

Signed by all members:

孔熙 Xi Kong.

周一涵 Yihan Zhou

王君拓 Juntuo Wang

张苏骑 Suqi Zhang

袁淑宸 Shuchen Yuan

陈乐多 Leduo Chen

陈宇臻 Yuzhen Chen

杨与懿 Yuyi Yang

李博文 Bowen Li