

Pet Shop

Name: Laura-Luisa Voicu

Group: 30236

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

The project represents a pet shop application with additional functionalities such as adopting or giving for adoption a pet. The actors which interact with the system have different roles: Administrator, Sellers (of pet products), Customers, Pet fosterers. A user can have one or more of the roles enumerated.

**Functional requirements**

The application is provided with an authentication/authorization system for register or login, based on Spring Security framework. To note that passwords are stored in database using a hashing algorithm, increasing the security level.

Once a user is register, it has the default role of **Customer**. Customers are able to see all products available, add them to a virtual cart or a wish-list and buy them; they can filter the products by category, also having the possibility to see all categories available in the store. In addition, they can see pets given for adoption by types (a.k.a. breeds) and adopt them.

The **Seller** is also able to see all products in the store, create a product (put it up for sale), update product (for example change product description, quantity, category etc.) or delete its products (remove it from sale). It has the permission to see what categories exist in the store.

The **Pet-Foster** can give a pet for adoption (create it virtually), update the pet object, delete it (equivalent to giving it for adoption to another user of the application) and see all pets registered in application.

The **Administrator** is responsible with the creation, modification, deletion of product Categories and Pet-Types. Also, it has the all the permissions a Seller, Customer or Pet-Foster has (for testing purposes).

* **Future improvements** (as features):
  + all modifications made to a product (that implies all CRUD operations) must be approved by the **Administrator.**
  + a user has to provide a certificate approved by **Administrator** in order for it to be a foster / seller

## Use Case Model 1

### Use Cases Identification

Use-Case: Admin logs in, search for specific or all application users, select one and perform actions on it

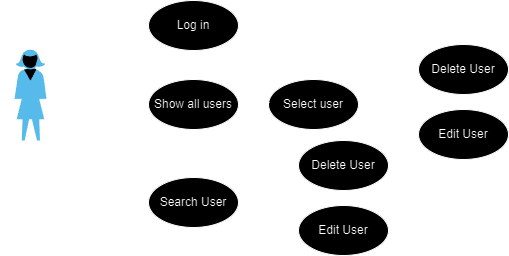
Level: Admin Goal

Primary Actor: Admin

Main success scenario:

* The administrator logs in using its credentials.
* After authorization, it is redirected to main page, where he can select from the top bar the actions on other users.
* If he chooses to show all the users registered to the application, it will be redirected to a page with a table with all information about those and a search box which improve the efficiency of getting a specific user.
  + He can select the row with specific users and perform CRUD operations on it.
* If it chooses to create a new user, it’s going to see a pop up with specific fields needed for registration.
  + It will be notified if the password is not strong (too short, no special characters, etc.), the email already exists in database, a mandatory field was not filled.
* If it chooses to remove a user, it will be able to do it eighter by selecting it from the table presended above, or it can choose **Remove User** from the tool bar and search it by name, username, email address.
* It it chooses to modify information of a user, it will be able to do it eighter by select it from the table presended above, or it can choose **Modify User** from the top bar and search it by name, username, email address.

Extensions:

* After log in, the administrator will be redirected to a Start page which consist in displaying personal information about the user,
* All pages have a top bar with all operations permitted for administrator.
* If the operations performed by the Admin on users are incomplete or wrong data is provided, a notify will pop up, letting to know what it did wrong.

Use-Case: Admin logs in, displays all application user-roles and creates new ones.

Level: Admin Goal

Primary Actor: Admin

Main success scenario:

* The administrator logs in using its credentials.
* After authorization, it is redirected to main page, where he can select from the top bar the actions on another user-roles.
* If he chooses to show all the users registered to application, it will be redirected to a page with a table with all information.
* If it chooses to create a new user, it’s going to see a pop up with specific fields needed for registration.
  + It will be notified if the password is not strong (too short, no special characters, etc.), the email already exists in database, a mandatory field was not filled.

Extensions:

* After log in, the administrator will be redirected to a Start page which consist in displaying personal information about the user,
* All pages have a top bar with all operations permitted for administrator.
* If the operations performed by the Admin on users are incomplete or wrong data is provided, a notify will pop up, letting to know what it did wrong

A diagram of a company

Description automatically generated with medium confidence

Use-Case: Admin logs in, review product, pet creation/modification/deletion or Seller/Adopter Priviledge Request and approve/deny it.

Level: Admin Goal

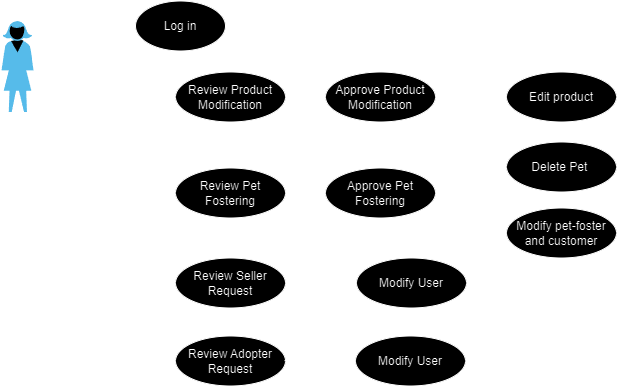
Primary Actor: Admin

Main success scenario:

* The administrator logs in using its credentials.
* After authorization, it is redirected to main page, where he can select from the top bar the actions on another user-roles.
* In the tool bar a notification button will pe displayed. By pressing it, the Admin will be redirected to a page where it can filter request (product or pet adoption )
* Product and/or Pet modifications/creations/deletions seller/adopter priviledge requestrequest will be shown.
* By selecting one of those requests, a pop-up page will apear with all details about the product, along with 2 buttons, Deny and Approve.
  + When one of those is selected, the pop-up will expand with a text field where the admin can specify the motivation of its decision.

Extensions:

* The notification button has a number on it, indicating the number of requests created.
* When the administrator is connected and another user is creating a new request in that moment, besides the fact that the number from notification button will be increased, a tiny pop-up will apear in the corner of the page, containing a summary of the request (type of request , product or pet adoption, and the creator)



Use-Case: Customer logs in or registers, buys products.

Level: Customer Goal

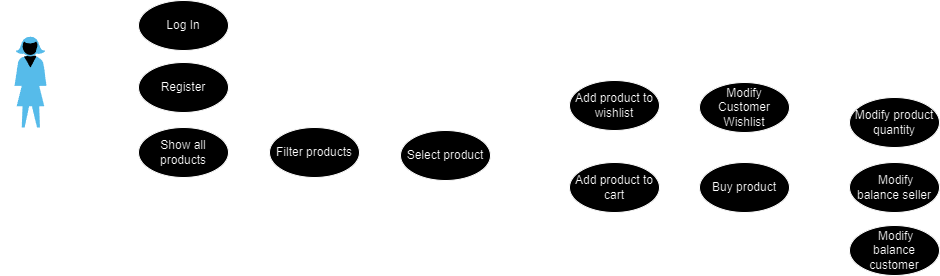
Primary Actor: Customer

Main success scenario:

* New customers hava to create an account by selecting “Register” Button from the top bar.
* If the customer already has an account, it can log in application via “Log In” button from the top bar.
* After authorization, it is redirected to the main page, where it can press a button “Shop Pet Products” to see all products available.
* If it wants a particular category of products, it can select desired Categories from the “Filter” button.
  + Here multiple filters can be added , including a search box with specification of products
* Products will be displyed in a table-like manner; by clicking on a product, the Customer is able to see details of it, including 2 buttons: “Add to Cart“ and “Add to wishlist”
* In the top bar will be 2 new buttons, unlike the Administrator page, designed to redirect the user to a new page with all Favorite Products or The Shopping Cart Page
  + The shopping cart page, besides the list with all the products added to the cart, will have a “Buy” button, that will redirect the Customer to a new page where he has to provide information about the Order

Extensions:

* The top bar can have “Frequent Questions”, “Forum” buttons.
* Besides information about the product, it can have a “Reviews” section, where other users can express their opinions about the product.
* When a product is added to Cart or Favorite list, a tiny notification will pop-up, letting the user know that the operation has succeeded.



Use-Case: Customer logs in or register, adopts a pet.

Level: Customer Goal

Primary Actor: Customer

Main success scenario:

* New customers hava to create an account by selecting “Register” Button from the top bar.
* If the customer already has an account, it can log in application via “Log In” button from the top bar.
* After authorization, it is redirected to the main page, where it can press a button “Adopt Pets” to see all pets given for adoption.
* If it wants a particular type of pet, it can select desired Types from the “Filter” button.
  + Here multiple filters can be added, including a search box with specification of pets (e.g. color, weight.)
* Pets will be displayed in a table-like manner; by clicking on a product, the Customer is able to see details of it, including 2 buttons: “Request Adoption”.
* In the top bar, besides the new buttons previously mentioned in the first use-case, there will be a third button indicating the list with adoptions.
  + The user is able to check the adoption status.

Extensions:

* The “Customer – Adopter” is able to chat with the Pet-Foster

A screen shot of a computer screen

Description automatically generated

Use-Case: Seller logs in or register, create/update a product.

Level: Seller Goal

Primary Actor: Seller

Main success scenario:

* In order to get priviledges of Seller, a simple customer must request Seller Roles from Administrator.
* After the prermissions are given to the customer, after login, the user must choose as what he wants to continue the login process: customer or seller.
* As seller, it is redirected to the main page, where it can select the action wanted to perform.
* If he chooses to show all the products sold by him, it will be redirected to a page listing all the products.
* If it chooses to create / update a new user, it’s going to see a pop up with specific fields needed.

Extensions:

A diagram of a product

Description automatically generated

Use-Case: Pet Foster logs in or register, create/update a product.

Level: Pet Foster Goal

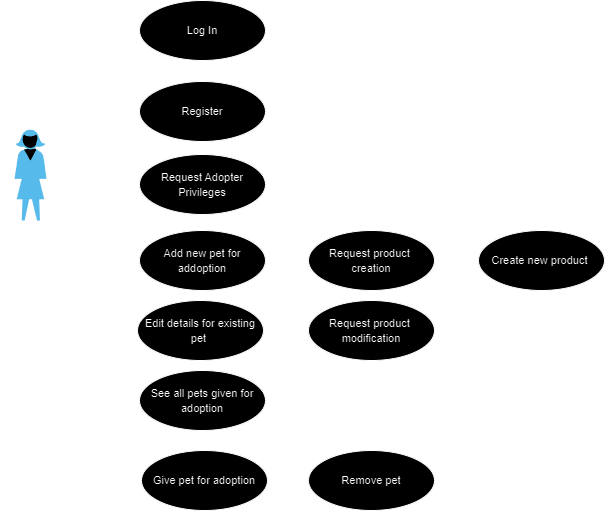
Primary Actor: Pet Foster

Main success scenario:

* In order to get the privileges of Pet Foster, a simple customer must request Pet-Foster Roles from Administrator.
* After the prermissions are given to the customer, after login , the user must choose as what he wants to continue the login process: customer or seller
* As pet-foster, it is redirected to the main page, where it can select the action wanted to perform.
* If he chooses to show all the pets given for adobption by him, it will be redirected to a page listing all the pets.
* If it chooses to create / update a new pet, it’s going to see a pop up with specific fields needed.

Extensions:

* The “Customer – Adopter” is able to chat with the Pet-Foster



### UML Use Case Diagrams

## Supplementary Specification

### Non-functional Requirements

[Choose 4 NF for your system, describe them and explain why these NF are suitable for your implementation. ]

### **Security** – The login/registration system within the application via the Spring framework ensures a high level of security. Moreover, by passing passwords through a hashing function instead of storing them in plain text, the application is protected against attackers.

### **Maintainability** – Through implementations that adhere to software design principles (OO – SOLID – DPs – architecture), stability and adaptability to changes are ensured, making the application easy to maintain.

### **Usability** – We want our system to be easy to use. That means making sure everything is clear and simple, like how you move around and what you need to do. We've made sure that each type of user has a clear role and knows what they can do. And when you use our system, it'll feel friendly and easy, no matter how much you know about technology.

### **Reliability** – We want our application to work smoothly without any big problems. So, we've checked it a lot to make sure it won't suddenly stop working. We keep an eye on it all the time to catch any issues quickly. And if something unexpected happens, we've set it up to handle it smoothly, so you won't even notice. This way, you can trust our application to work well every time you use it.

### Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

## Glossary

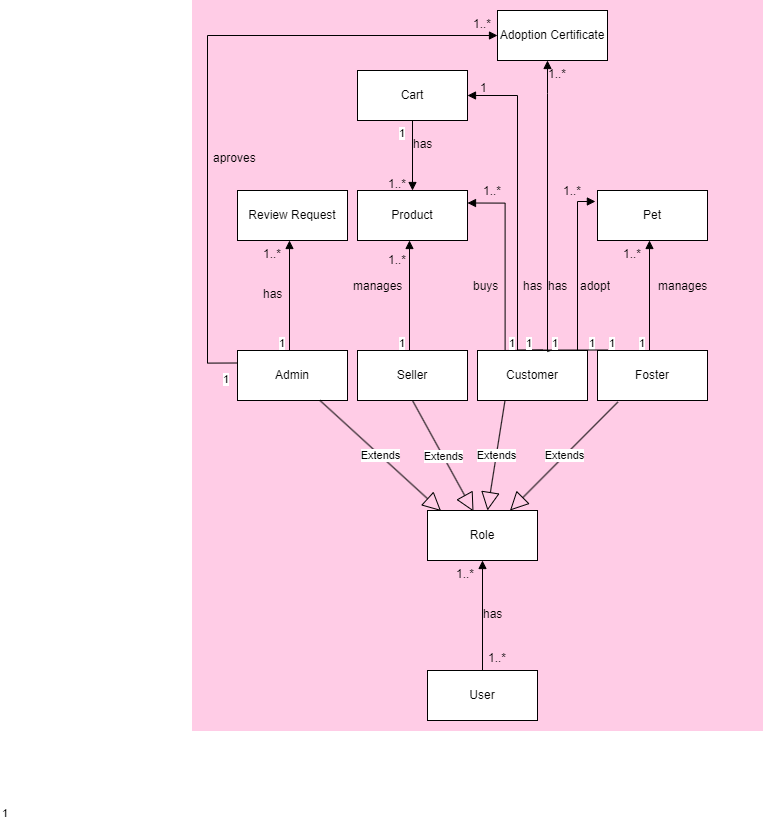
[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

# Deliverable 2

## Domain Model

[Define the domain model and create the conceptual class diagrams]

* Domain model is a structured visual representation of interconnected concepts or real-world objects that incorporates vocabulary, key concepts, behavior, and relationships of all of its entities
* Domain model is a structured visual representation of interconnected concepts or real-world objects that incorporates vocabulary, key concepts, behavior, and relationships of all of its entities
* Visual representation of conceptual classes or real-situation objects in a domain



## Architectural Design

### Conceptual Architecture

[Define the system’s conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

The most suitable architecture for a PetShop application is Layered Architecture because it provides a clear separation of concerns between presentation, business logic and data access layers. This separation enables easier mentenance and flexibility, allowing the programmer to manage pet adoption processes effectively.

Benefits of Layerd Architecture :

* Modular design : For a pet shop and adoption management app, you could have layers such as presentation/UI layer, business logic layer, and data access layer
* Flexibility: Layered architecture offers flexibility for adapting and evolving applications over time. Components within each layer can be modified or replaced without affecting other parts of the system. This flexibility is advantageous in dynamic domains like pet shop and adoption management, where requirements may undergo frequent changes.
* Security: Security measures can be implemented more effectively in a layered architecture by segregating concerns into distinct layers. For instance, access control and validation rules can be enforced within the business logic layer, while encryption and authentication mechanisms can be implemented within the presentation and data access layers
* Separation of responsibility: In a layered architecture, each layer concentrates on a specific aspect of the application. For instance, the business logic layer manages the application's rules and workflows, while the data access layer handles database interactions. This separation enhances the manageability, comprehensibility, and maintainability of the codebase.



### Package Design

[Create a package diagram]

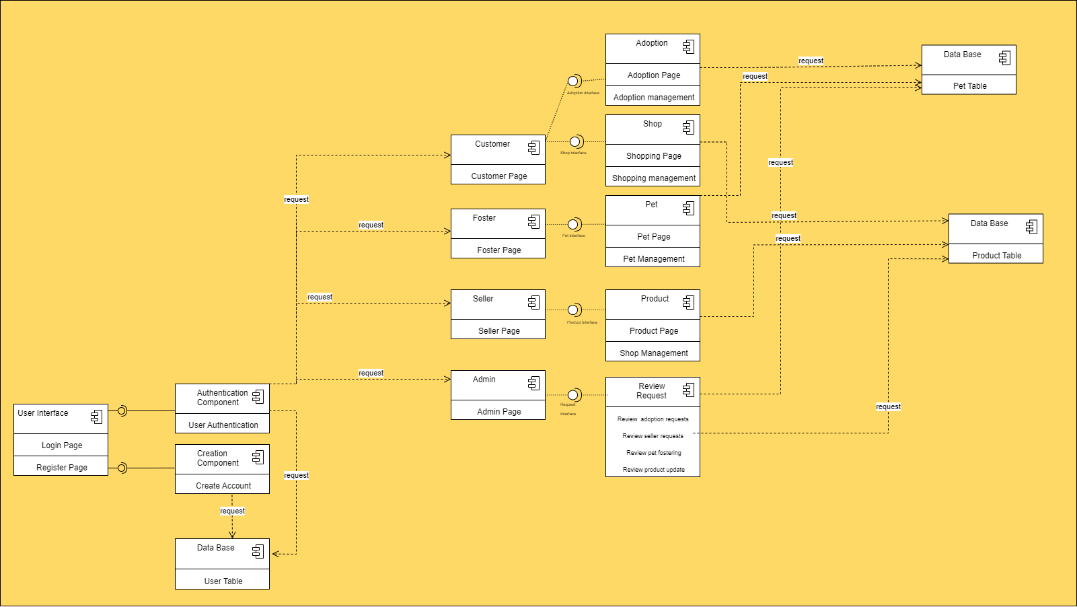
A screenshot of a computer diagram

Description automatically generated

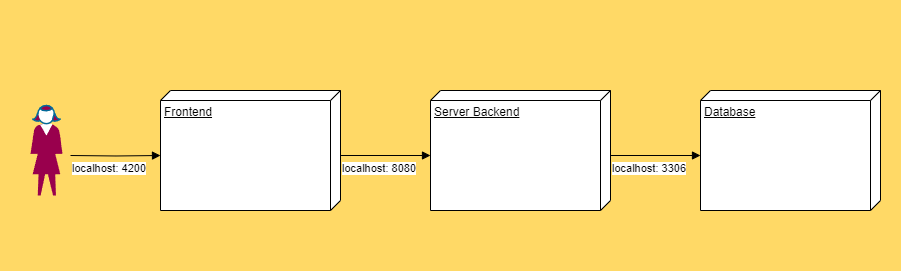
### Component and Deployment Diagram

[Create the component and deployment diagrams.]

* **Component diagram**



* **Deployment Diagram**



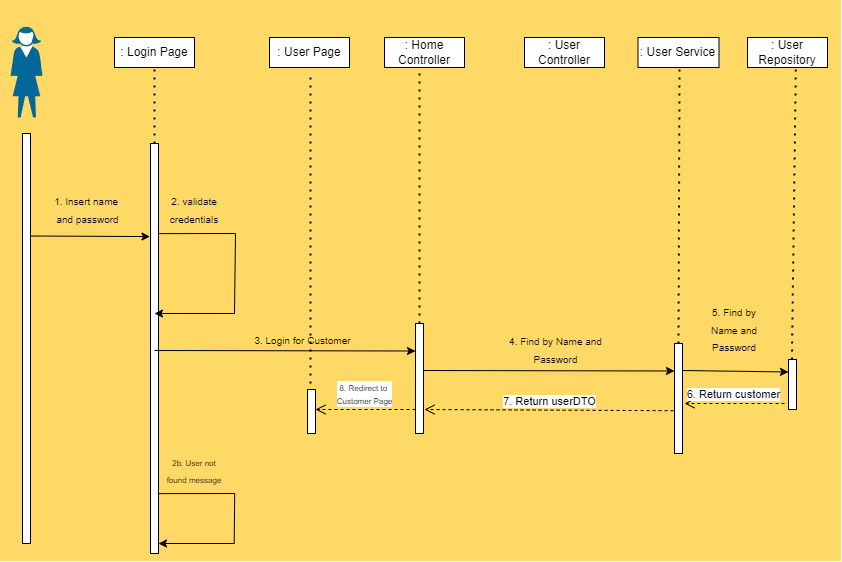
# Deliverable 3

## Design Model

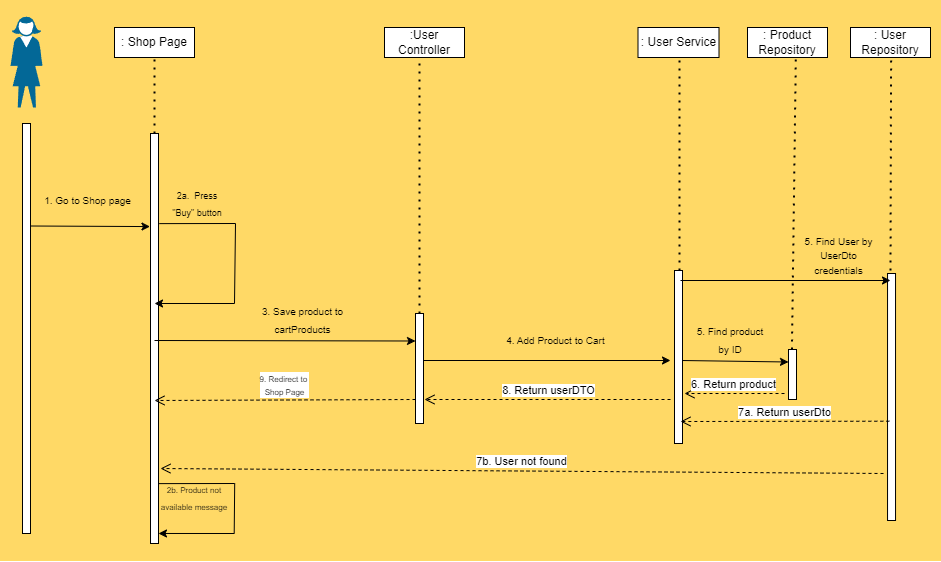
### Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

1. Login as Customer Scenario

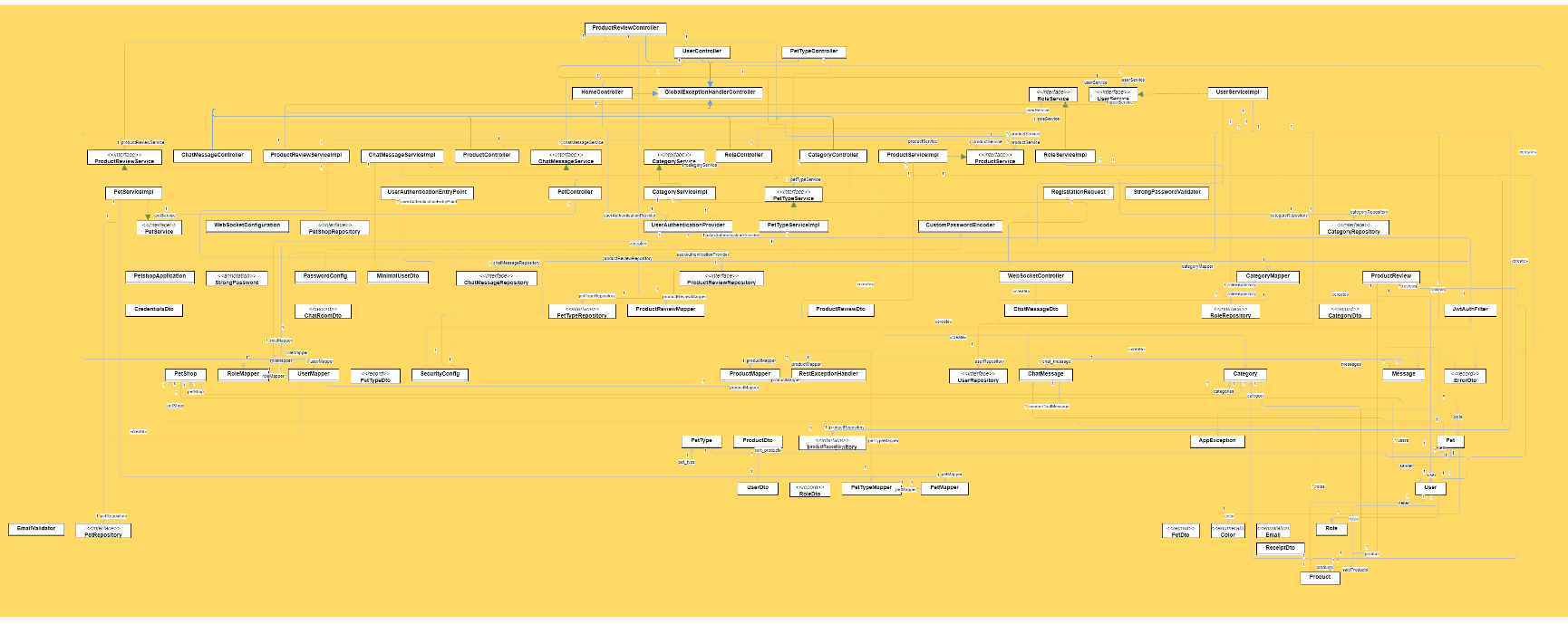


1. Shop Product Scenario



### Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]



## Data Model

[Create the data model for the system.]

A computer generated diagram of a computer

Description automatically generated with medium confidence

# System Testing

[Describe the testing methides and some test cases.]

* **Testing methods**
  + The test classes follow a unit testing approach using JUnit and Mockito frameworks. Unit testing involves testing individual units or components of software in isolation to ensure they function correctly.
* **Mocking Dependencies**: Mockito is used to mock dependencies such as services and providers. Mock objects simulate the behavior of real objects, allowing controlled interactions during testing.
* **Test cases**
  + **HomeControllerTest:**
    1. **testLogin**: This test case checks the login functionality of the HomeController. It verifies that the controller returns the expected UserDto object containing a token when valid credentials are provided. Mocks are used to simulate the login process and token generation.
  + **PetControllerTest:**
    1. **testGetAllProducts:** This test method ensures that the PetController correctly retrieves all pet products. It mocks the PetService to return a list of PetDto objects and verifies that the controller returns the expected response with the correct status code and pet details.
    2. **testCreateProduct:** This test case validates the functionality of creating a new pet product. It mocks the PetService to return a PetDto object upon successful creation and verifies that the controller returns the expected response with the correct status code and product details.
  + **ProductControllerTest:**
    1. **testGetAllProducts**: This test verifies that the ProductController retrieves all products successfully. It mocks the ProductService to return a list of ProductDto objects and checks that the controller returns the expected response with the correct status code and product details.
    2. **testCreateProduct:** This test case ensures that the ProductController correctly handles the creation of a new product. It mocks the ProductService to return a ProductDto object upon successful creation and verifies that the controller returns the expected response with the correct status code and product details.
  + **UserControllerTest:**
    1. **testGetAllUsers**: This test method checks if the UserController retrieves all users correctly. It mocks the UserService to return a list of UserDto objects and ensures that the controller returns the expected response with the correct status code and user details.
    2. **testDeleteUser**: This test case validates the functionality of deleting auser by ID. It mocks the UserService to return a user object when given an ID and verifies that the controller returns the expected response with the correct status code.

# Future Improvements

[Present some features that apply to the application scope.]

* **Enhanced Security with Two-Factor Authentication**: Strengthen user accounts with an extra layer of security. Users will receive a unique code via email or SMS, adding an additional step to the login process for increased protection against unauthorized access.
* **Pet Pop-ups for Fostering Opportunities**: Engage users with heartwarming pop-ups featuring pets in need of fostering. These delightful prompts will encourage users to consider providing a temporary home for pets awaiting adoption, promoting animal welfare and community involvement.
* **User Ban and Report Functionality**: Empower community members to maintain a safe and respectful environment. Users can report inappropriate behavior or content, prompting admin review and possible user banning if deemed necessary, ensuring a positive experience for all.
* **Animated Gifs for Interactive Experience:** Elevate user interactions with delightful animated gifs. From confirming purchases to completing actions, users will enjoy a dynamic and engaging experience with playful animations enhancing user engagement and enjoyment.
* **Secure Password Recovery via Verification:** Safeguard user accounts with a robust password recovery process. Users requesting password resets will verify their identity via email or phone number confirmation, ensuring secure access to their accounts.
* **Customizable Page Themes for Personalization**: Allow users to personalize their browsing experience. With customizable page themes, users can select their preferred color schemes and layouts, creating a personalized and visually appealing environment tailored to their tastes.
* **Unique Avatar Creation for User Profiles**: Foster individuality and creativity with customizable avatars. Users can create unique pet-themed avatars to represent themselves on the platform, adding a touch of personality to their profiles and interactions.

# Conclusion

# In summary, the pet shop application offers a user-friendly platform for customers, sellers, and pet fosterers. Robust authentication ensures security, while intuitive features enable seamless browsing, purchasing, and pet adoption. Future enhancements, like administrator approval for product modifications and user certifications, aim to elevate trust and accountability within the community, ensuring a safe and reliable experience for all users.

# Bibliography

1. [Implementing Token and Role Base Authentication Using Spring Boot + JWT+ MySQL - YouTube](https://www.youtube.com/watch?v=VZjs6-_yccw&list=PLZTETldyguF2bRz-ypCa3a8gumxeXB4pu&index=2&ab_channel=LearnProgrammingYourself)
2. [Secure Your Fullstack Angular - Spring Boot Application With the JWT Authentication - YouTube](https://www.youtube.com/watch?v=YUqi1IjLX8I&ab_channel=TheDevWorld-bySergioLema)
3. [Adding CORS in Spring Boot and Calling Signup API from Angular | Car Rental Project | Part 6 - YouTube](https://www.youtube.com/watch?v=ZI8nnzAdFAU&list=PLgYFT7gUQL8GNv-tqTsmwh6O_onYBFCiu&index=6&ab_channel=CodeWithProjects)
4. [Mastering JWT Authentication & Authorization with SpringBoot 3, Spring Security 6, Angular 16, MySQL - YouTube](https://www.youtube.com/watch?v=Ph_IO-Y_u74&t=3433s&ab_channel=CodeElevate)
5. [Mastering JWT Authentication & Authorization with SpringBoot 3, Spring Security 6, Angular 16, MySQL - YouTube](https://www.youtube.com/watch?v=Ph_IO-Y_u74&t=3433s&ab_channel=CodeElevate)
6. Email custom validation: <https://www.baeldung.com/java-email-validation-regex>
7. Spring Boot Testing: https://www.baeldung.com/spring-boot-testing
8. https://www.baeldung.com/java-password-hashing
9. Websockets Chat:
   * <https://github.com/neiserdeveloper/ms-chat-socket/tree/master>
   * <https://github.com/neiserdeveloper/ng-chat-socket/tree/master/src>
10. Angular Tutorial:
    * <https://www.youtube.com/watch?v=UIWK5_DrZ7w&list=PLpaspowtqj-dlt-C7FvRG42LNRFcXdRDD&ab_channel=CodeWithNasir>