Fpt University

TRƯỜNG ĐẠI HỌC FPT

PET SHOP

Software Design Specification

– Hanoi, March 2023 –

RECORD OF CHANGES

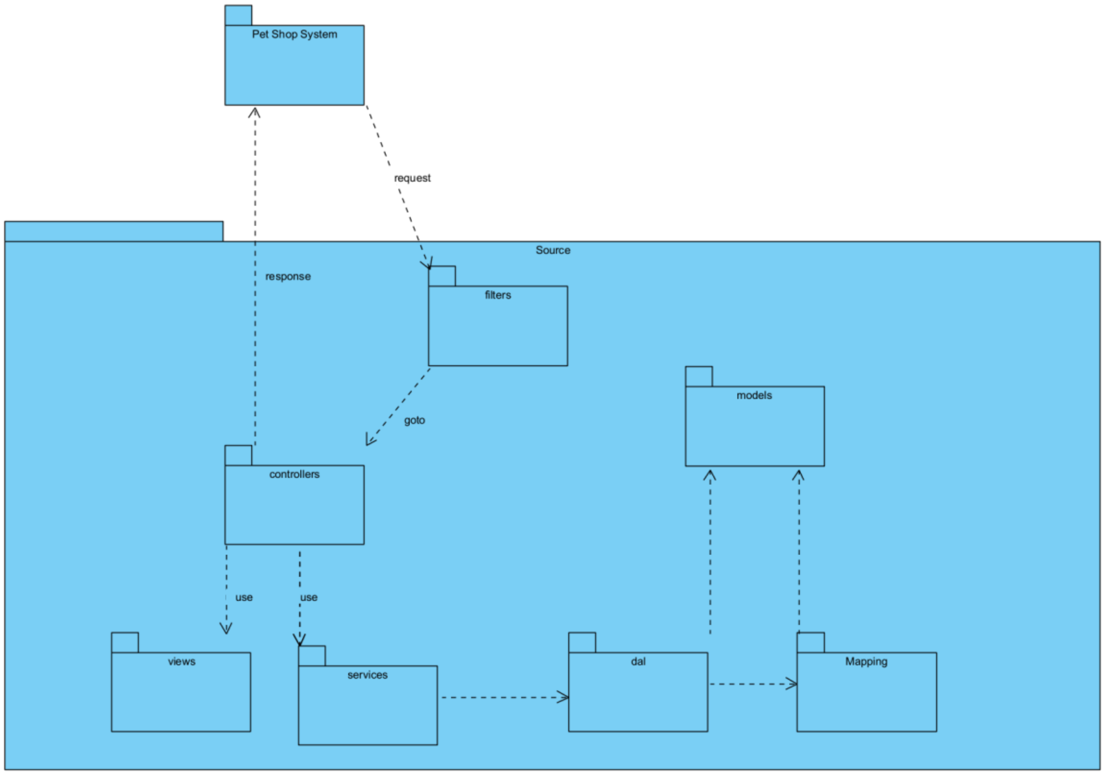
Date	A* M, D	In charge	Change Description
5/1/2023	A	Create new SRS	Create new
	A	Thành Long	User Register
	A	Minh Hoàng	Login
	A	Minh Hoàng	Logout
	A	Thành Lông	Forgot Password
	A	Thành Long	Change Password
	A	Anh Kỳ	Store,View, Filter, Search Product
	A	Đinh Sáng	Listing Category and SubCategory
	A	Đình Sáng	Home Page
	A	Minh Hoàng	Listing Product
	A	Đình Sáng	CRUD (create read, update, delete) product
	A	Đức Anh	View, Filter, Search Order
	M	Anh Kỳ	Product Detail, List Product Detail

*A - Added M - Modified D - Deleted

Table of Contents

I. Overview

1. Code Packages



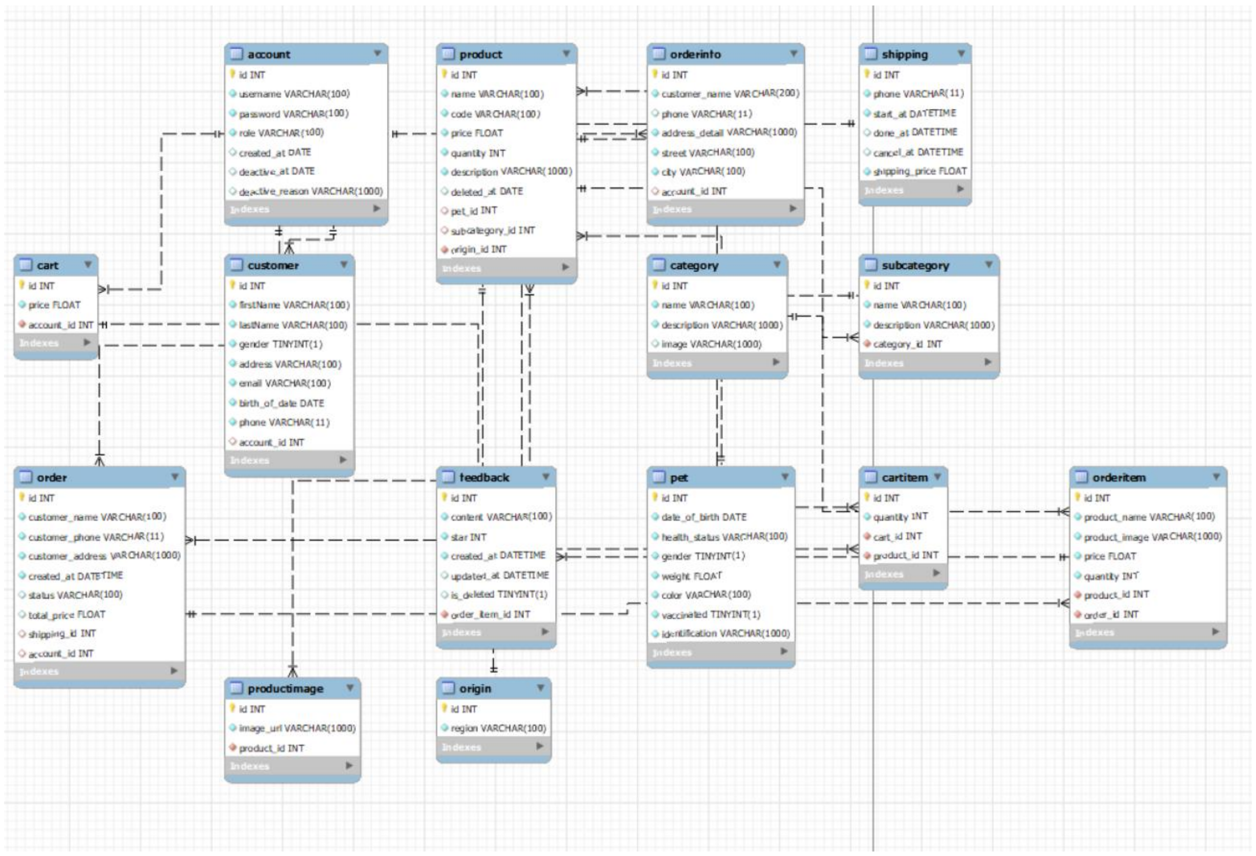
Package descriptions

No	Package	Description
01	AppConfig	This component is used to provides a convenient way to define and access session and cookie names in a web application
02	controllers	This component contains login, logout,register and display all front-end common features .
03	controllers.account	This component is used to change password of users
04	controllers.cart	This component provide functionality for managing a shopping cart
05	controllers.category	This component is designed to contain list category and subcategory
06	controllers.customer	This component is designed to provide functionality for managing self-user information
07	controllers.feedback	This component is designed to provide functionality for managing customer feedback
08	controllers.order	This component is designed to provide functionality for managing order

09	controllers.orderInfo	This component is designed to provide functionality for managing order of customer
10	controllers.product	This component is designed to display product detail
11	controllers.search	This component is designed to provide functionality for searching
12	controllers.store	This component is designed to provide functionality for display product
13	dal	This component is used to get data directly FROM the database
14	mapping	This component provides a way to map models to database records and vice versa.
15	models	This component includes interfaces used to receive data retrieved FROM the database and classes used to receive data
16	services	This component is used to process data after it is retrieved from the database. (dal)

2. Database Design

a. Database Schema



b. Table Description

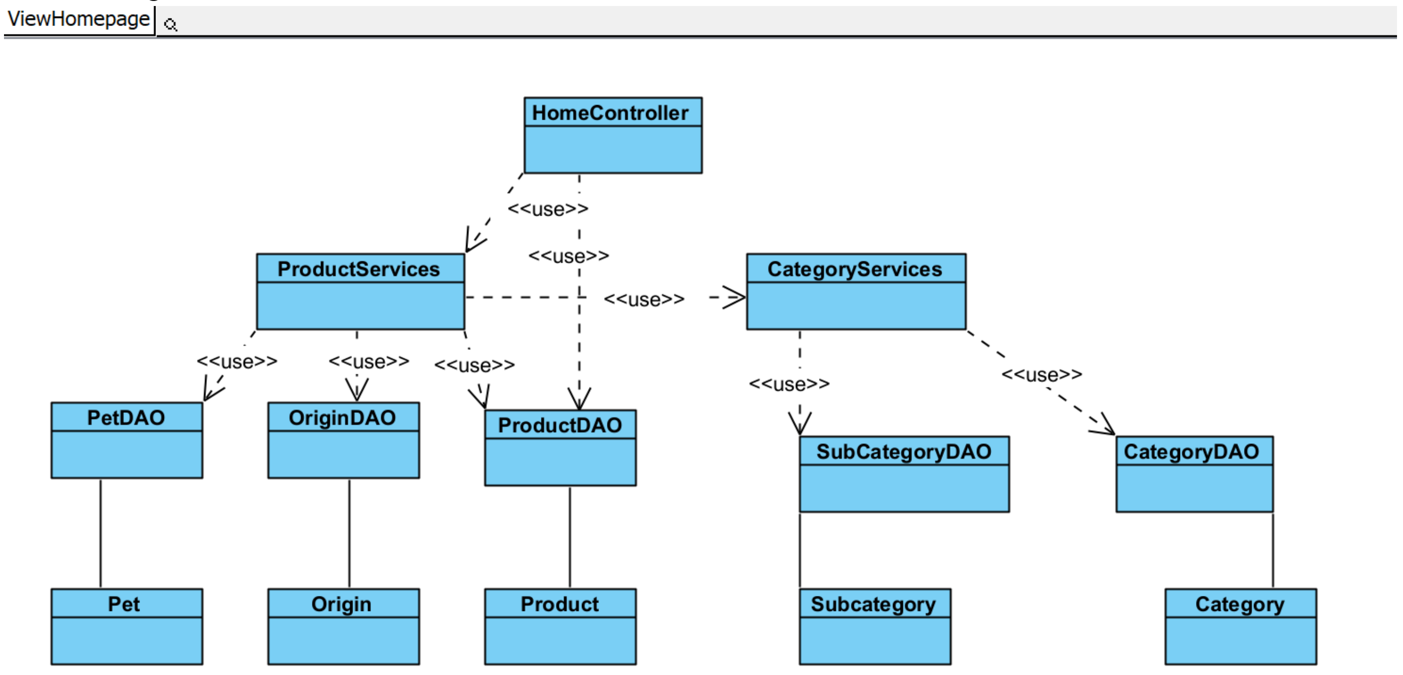
No	Table	Description
01	account	The registered Users, Employee, Admin of the system

02	order	Contains information about order
03	orderinfo	Contains information about order and customer
04	orderitem	Contains information about product customer buy
05	shipping	Contains information about shipping status
06	feedback	Contains information about feedback of customer after bought
07	customer	Contains information about Users that have an account
08	product	Contains information about product
09	origin	Contains information about region of that product
10	pet	Contains information about pet
11	productimage	Contains images of each product
12	cart	Contains information about cart
13	cart item	Contains item in cart
14	subcategory	Contains information about subcategory
15	category	Contains information on subcategory and category

II. Code Designs

1.View homepage

a. Class Diagram



b. Class Specifications

HomeController

No	Method	Description

01	doGet()	initializes some services and a data access object, receives and parses a parameter from the request, retrieves a list of products based on this parameter, retrieves a list of categories, sets some attributes to be used in the JSP page, and forwards the request and response objects to the home.jsp view.

ProductDAO

No	Method	Description
01	List<Product> getAllProduct()	Gets all products from a database table called "Product" using JDBC, maps the results to Product objects, and returns them as a List. Any errors are logged, and there's a commented out section to close the database connection after use.
02	pagingProductsByRandom(String code)	Takes in a String parameter code and returns a list of Product objects. The method connects to a database, executes an SQL query that selects products based on a code pattern, orders the results in a random manner and limits the number of results to 5

Category

No	Method	Description
01	Category()	Represents a category object.
02	toString()	Generating a string representation of the object.

Product

No	Method	Description
01	Product	Represents a product object.

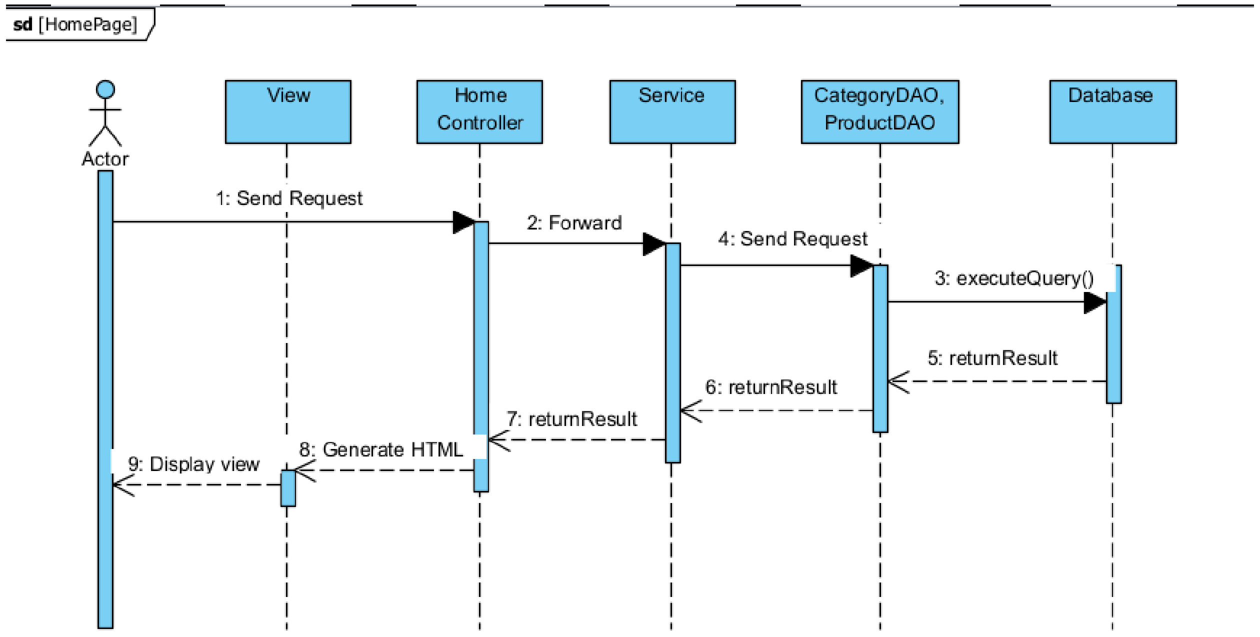
CategoryServices

No	Method	Description
01	getSubAndCategory()	uses two DAO objects to retrieve data from a database. The constructor initializes the two DAO objects.

ProductServices

No	Method	Description
01	getAllInforProductPet(int x)	The code retrieves information about a product and associated objects using DAOs and services. It sets the retrieved information into a Product object and returns it.

c. Sequence Diagram(s)

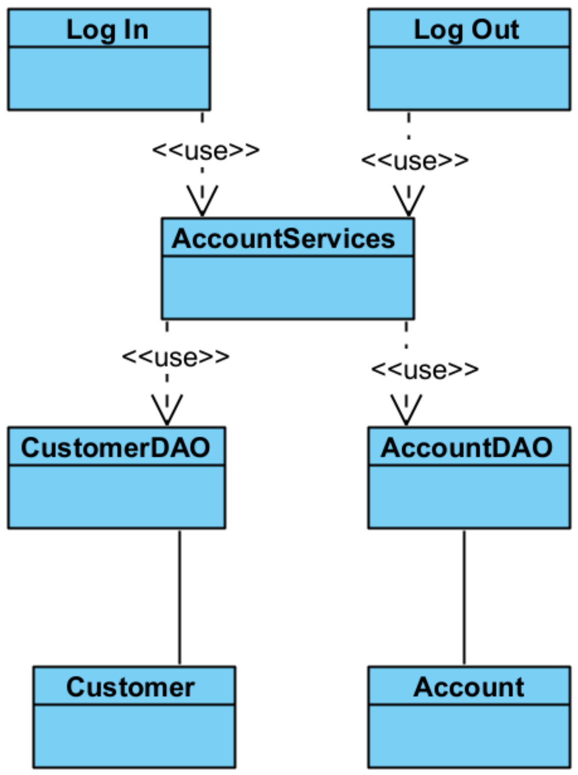


d. Database Queries

- select * from shop_pet.product spp inner join shop_pet.pet spp2 on spp.pet_id=spp2.id where spp.code like ? ORDER BY RAND() LIMIT 5
- select*from shop_pet.product spp inner join shop_pet.subcategory sps on spp.subcategory_id = sps.id inner join shop_pet.pet spp2 on spp.pet_id = spp2.id where spp.subcategory_id = ? and spp.id != ?
- select*from shop_pet.product spp inner join shop_pet.subcategory sps on spp.subcategory_id = sps.id where spp.subcategory_id = ? and spp.id
- SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where spc.id = limit ? offset ?
-

2.Login feature

a. Class Diagram



b. Class Specifications

Login

No	Method	Description
01	doGet()	Checks if a session attribute is present, and if it is, redirects the user to the home page. If the attribute is not present, the user is forwarded to the login page
02	doPost()	Handles a user login request, It retrieves the username and password from the request parameters, creates cookies with them. If the login is successful, it sets the user object in the session attribute and redirects the user to the home page. If there is an exception during the login process, it sets an error message and forwards the user to the login page.

AccountDAO

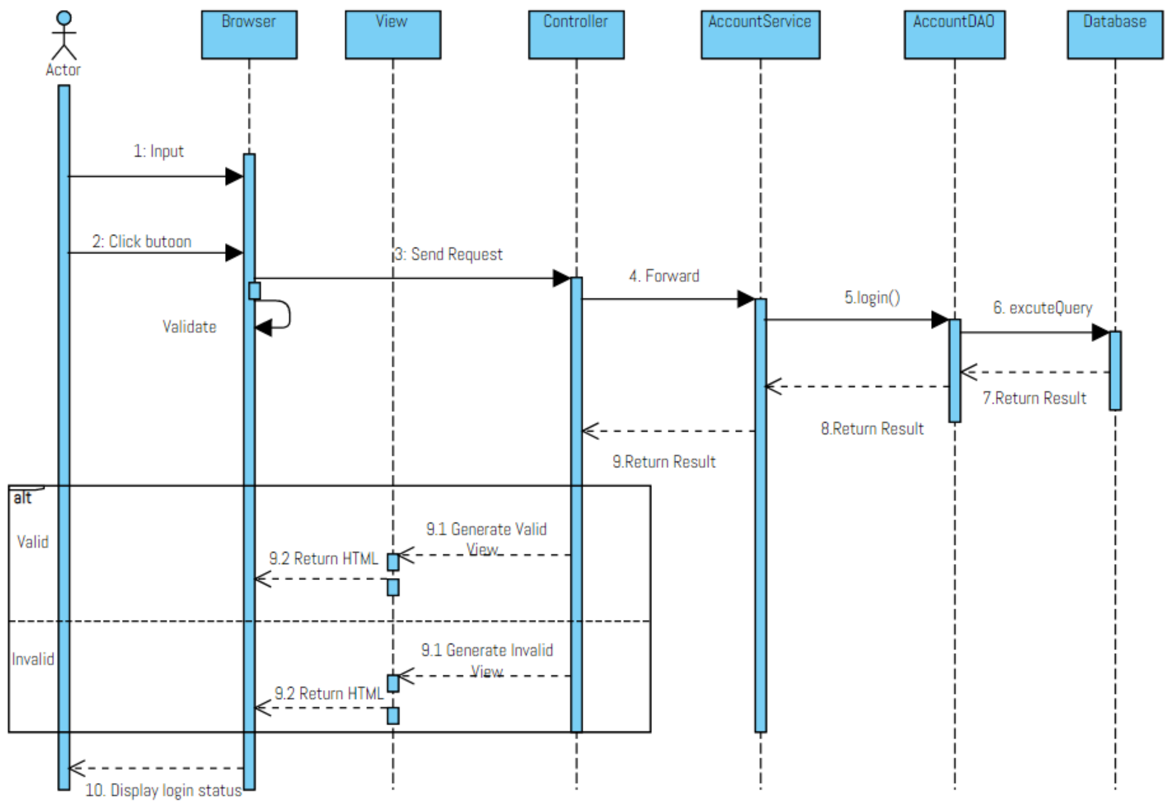
No	Method	Description
01	login(usename: String)	Attempts to log in an account by querying the database for a row where the username and password match the given

		parameters. If a matching row is found, it converts it into an Account object and returns it. If there is an exception during the process, it returns null.

Account

No	Method	Description
01	Account()	The Account class is a simple Java bean that represents an account in the system. It has private instance variables for the ID, username, password, role, created date, deactive date and reason, as well as a reference to its associated Customer object. The class provides getters and setters for all of its instance variables, allowing them to be accessed and modified from outside the class.
02	toString()	Generating a string representation of the object.

c. Sequence Diagram(s)

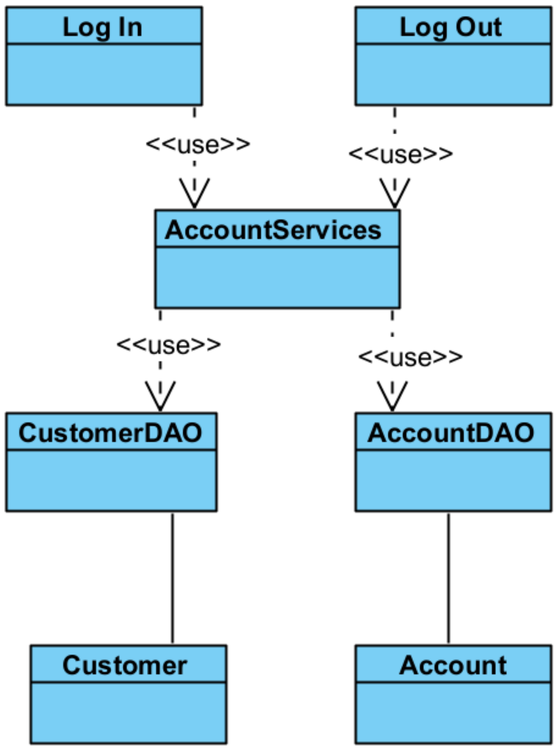


d. Database Queries

*SELECT * FROM Account WHERE username = ? AND password = ?*

3.Logout

a. Class Diagram

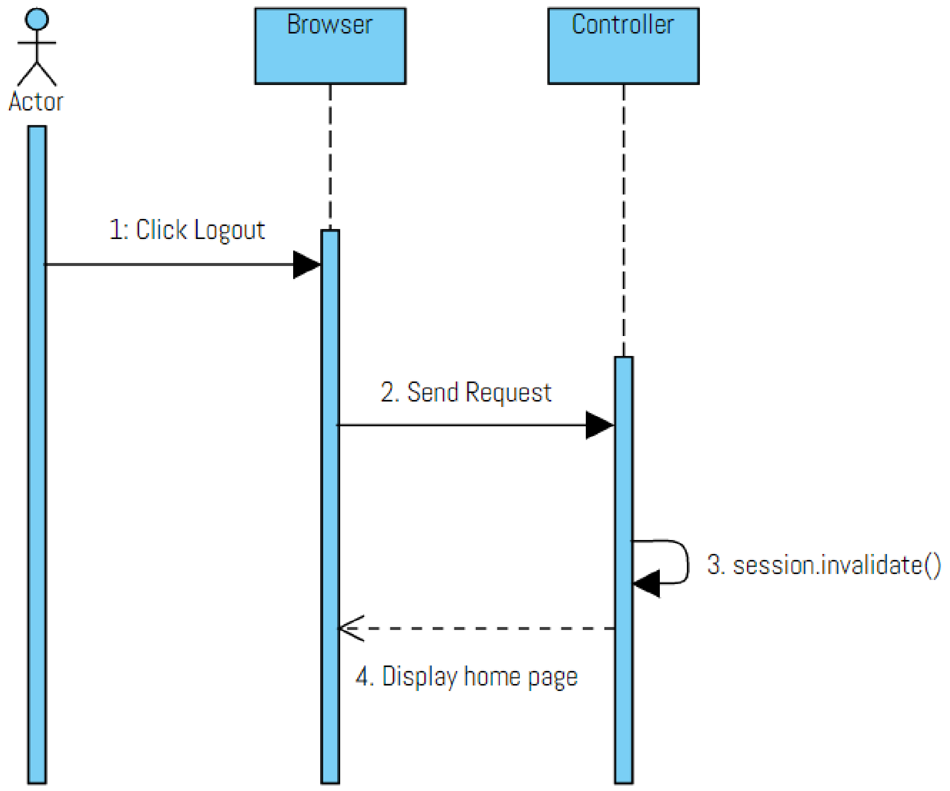


b. Class Specifications

Logout

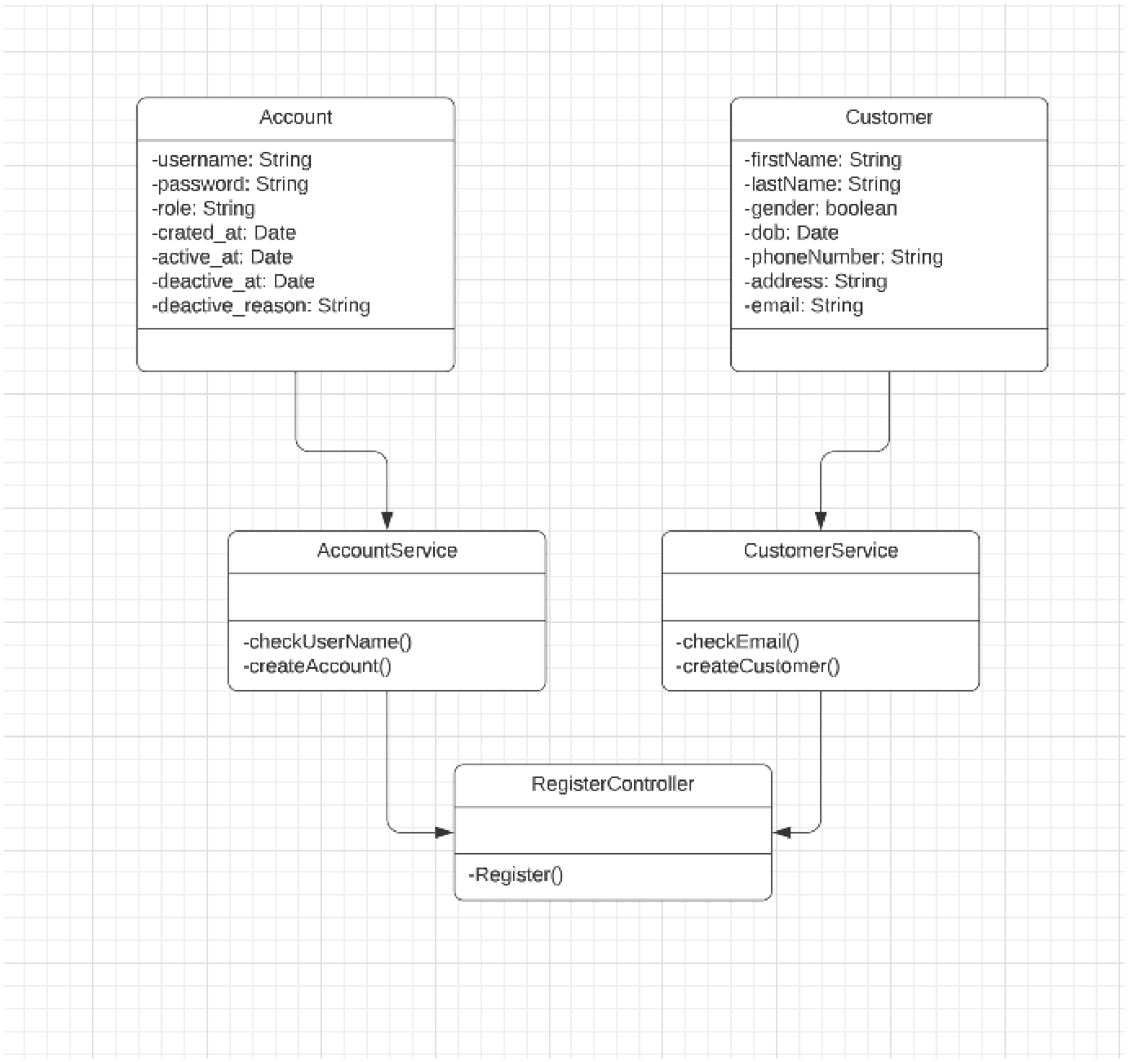
No	Method	Description
01	session.invalidate()	Destroys the current session and frees up any resources associated with it.

c. Sequence Diagram(s)



4. Register feature

a. Class Diagram



b. Class Specifications

RegisterController

No	Method	Description
01	doPost	handles registration form submissions. It checks for validation errors, creates new Account and Customer objects, and saves them to the database using AccountService and CustomerService. If there are errors, it displays error messages and prompts users to fix their inputs. Otherwise, it redirects users to the login page. Any errors encountered along the way are caught and printed to the response writer.
02	doGet	forwards the current request and response objects to the registration page named "register.jsp". The forward method is used to transfer control from one servlet to another during the execution of a request.

AccountService

No	Method	Description
01	create(Account account)	Creates a new record in a database for an input account object. It uses an instance of the "accountDAO" class to insert the account into the database and throws a SQLException if an error occurs.
02	boolean checkUserExist(String username)	Takes in a String parameter "username" and throws a SQLException. It calls the "checkExistUsername" method of an instance of an object named "accountDAO". The result of this method call is returned as a boolean value.

CustomerService

No	Method	Description
01	boolean checkEmailExist(String email)	Checks if an email exists in a database and returns a boolean value. It may throw a SQLException.
02	void create(Customer cus)	Takes in a parameter of type "Customer" named "cus" and throws a SQLException. It calls the "addCustomer" method of an instance of an object named "customerDAO".

CustomerDao

No	Method	Description
01	addCustomer(Customer customer)	Takes in a Customer object as a parameter and inserts its properties into a database table. The method uses a SQL query with placeholders for the values, sets the values from the Customer object to the prepared statement, and then executes the update. The code may throw a SQLException, but ensures that all resources are properly closed after execution.
02	boolean checkExistEmail(String email)	Takes in an email as a parameter and returns a boolean value. The method queries the Customer table to determine if the given email already exists and returns true if it does, otherwise false. The code may throw a SQLException, but ensures that all resources are properly closed after execution.

AccountDao

No	Method	Description
01	void insert(Account user)	Takes in an Account object as a parameter and inserts its properties into a database table. The method uses a SQL query with placeholders for the values, sets the values from the Account object to the prepared statement, and then executes the update. The code may throw a SQLException, but ensures that all resources are properly closed after execution.
02	boolean checkExistUsername(String username)	akes in a username as a parameter and returns a boolean value. The method establishes a database connection, queries the Account table to determine if the given username already

		exists, and returns true if it does, otherwise false. The code may throw a SQLException, but ensures that all resources are properly closed after execution.

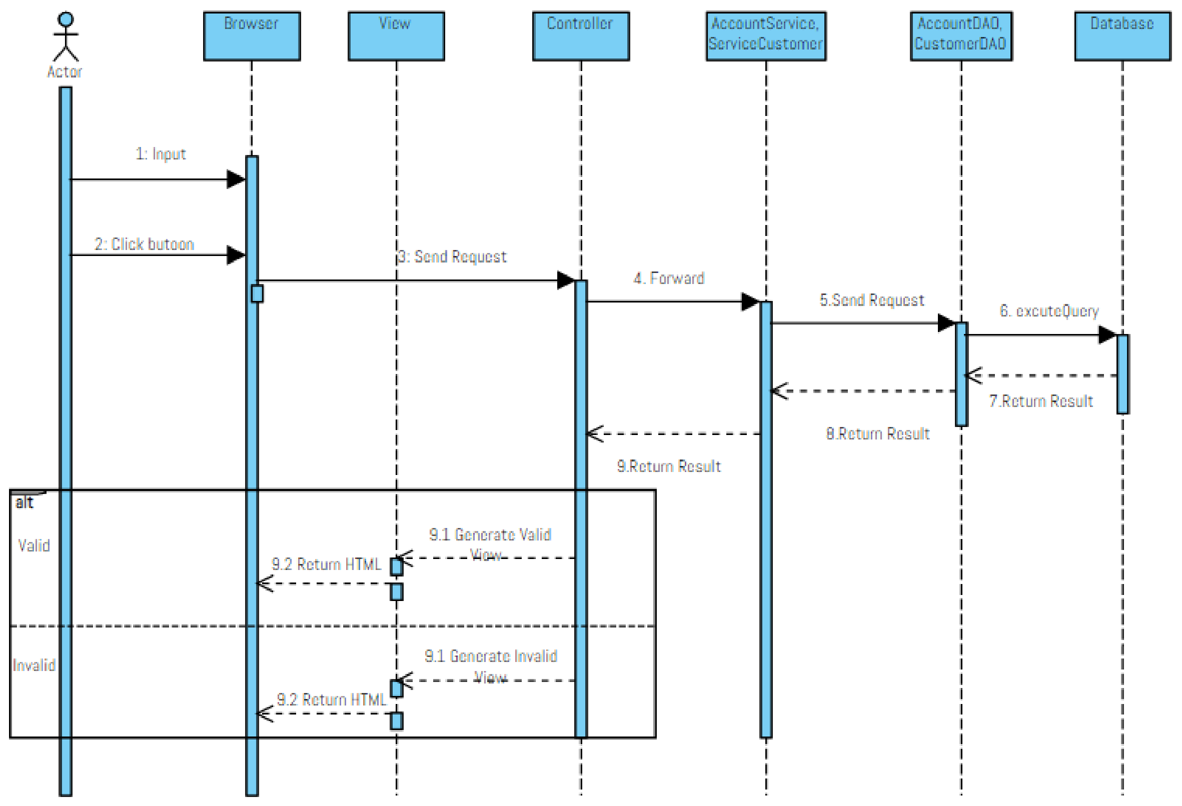
AccountMapping

No	Method	Description
01	daotoObject	Takes a ResultSet object as input and returns an Account object. It extracts data from the result set using various methods such as getInt, getString, getDate, etc., and initializes the fields of the Account object with the extracted values. Finally, it returns the Account object.
02	setFromObject	Takes an Account object and a PreparedStatement object as input and sets the values of the prepared statement using various methods such as setString, setDate, etc., with the values obtained from the fields of the Account object. Finally, it returns the PreparedStatement object.

CustomerMapping

No	Method	Description
01	daotoObject	Takes a ResultSet object as input and returns a Customer object. It extracts data from the result set using various methods such as getInt, getString, getBoolean, etc., and initializes the fields of the Customer object with the extracted values. Finally, it returns the Customer object.
02	setFromObject	Takes a Customer object and a PreparedStatement object as input and sets the values of the prepared statement using various methods such as setString, setBoolean, setDate, etc., with the values obtained from the fields of the Customer object. Finally, it returns the PreparedStatement object.

c. Sequence Diagram(s)



d. Database Queries

INSERT INTO Account (username, password, role, created_at, deactive_at, deactive_reason) VALUES (?,?,?,NOW(),?,?)

INSERT INTO shop_pet.Customer (firstName, lastName, gender, address, email, birth_of_date, phone,account_id) VALUES (?, ?, ?, ?, ?, ?, ?,?)

5. Forgot password

a. Class Diagram

b. Class Specifications

ForgotPasswordController

No	Method	Description

01	doGet	Forwards the request to the "forgotPassword.jsp" page for rendering in the user's browser.
02	doPost	Checks if the provided email exists in the customer database, generates a verification code, and sends it to the user's email address using the SendMail utility class.If the email fails, the user is redirected to an error page. If the email address is invalid or does not exist in the database, the user is informed through a message prompt on the forgotPassword.jsp page.

SendMail

No	Method	Description
01	boolean sendEmail(String email, String code, String key)	Sends an email to a given email address, containing a verification link with a code and key.

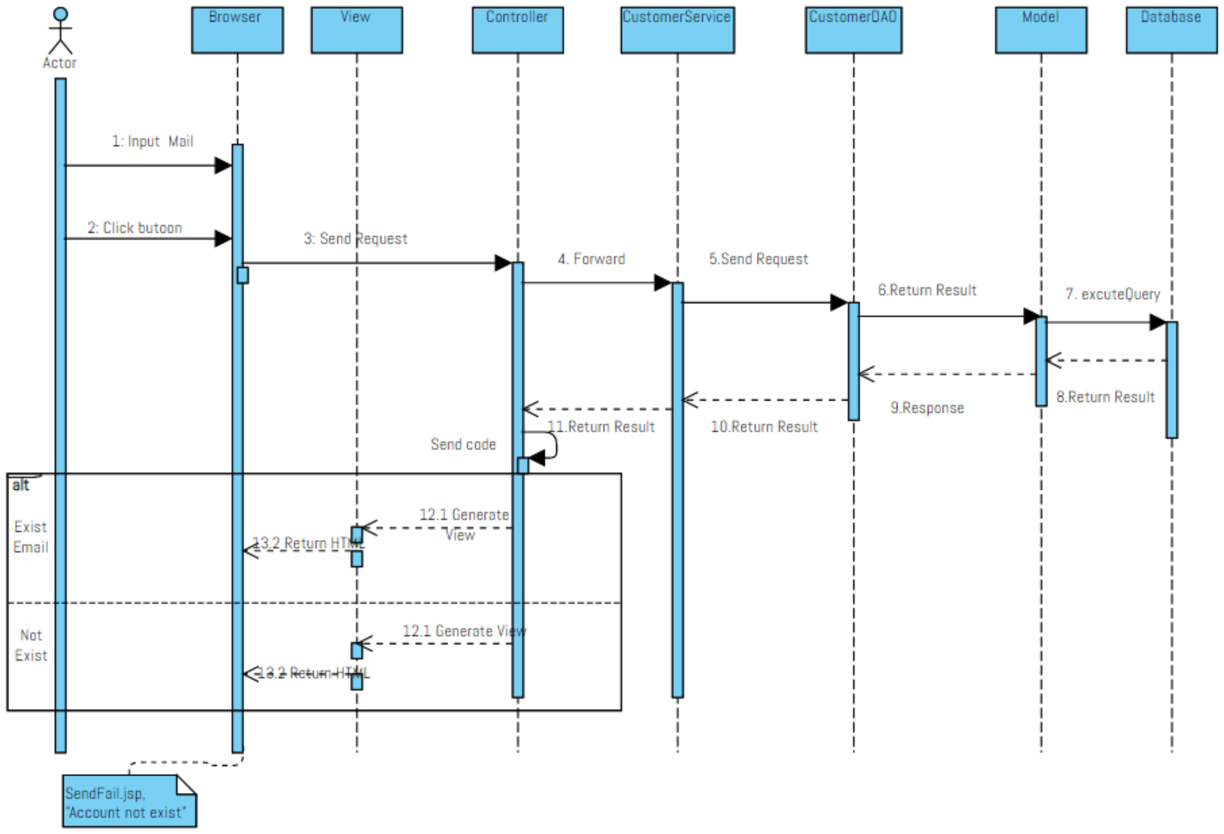
CustomerService

No	Method	Description
01	boolean checkEmailExist(String email)	Checks if an email exists in a database of customer records. It returns true if the email exists, and false if it does not exist.

CustomerDAO

No	Method	Description
01	boolean checkExistEmail(String email)	Checks if an email exists in a database of customer records. It returns true if the email exists, and false if it does not exist.

c. Sequence Diagram(s)

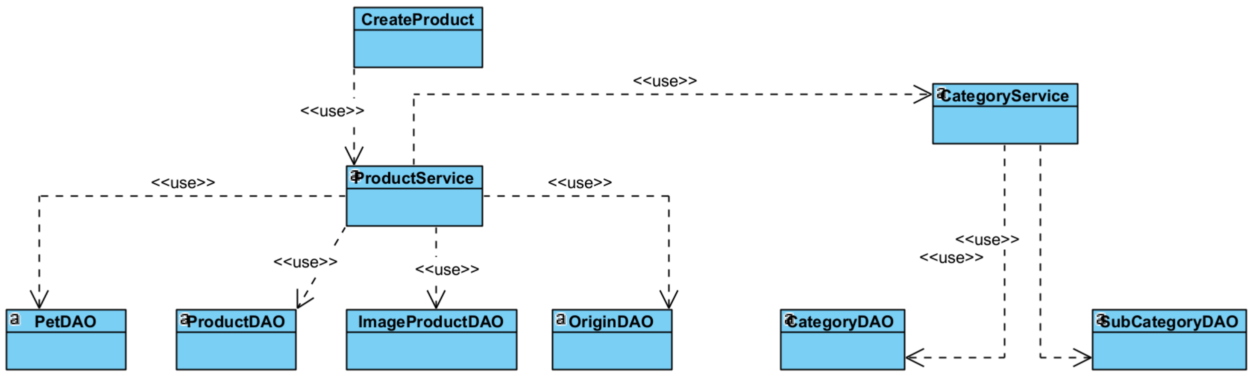


d. Database Queries

- SELECT COUNT(*) FROM shop_pet.Customer WHERE email = ?

6. Create Product

a. Class Diagram



b. Class Specifications

ProductDAO

No	Method	Description
01	void insertProductIsPet(int id, String name, String code, double price, int quantity, String description, int petId, int subId, int originId)	inserts a product into a "product" table in the "shop_pet" schema, with some additional fields , using a prepared statement to set fields with parameters passed to it. Any exceptions are logged by a built-in logger.
02	void insertProduct(int id, String name, String code, double price, int quantity, String description, int subId, int originId)	inserts a product into a "product" table in the "shop_pet" schema, using a prepared statement to set fields with parameters passed to it. Any exceptions are logged by a built-in logger.
03	int getCategoryIdfromproductId(int productId)	Takes an integer argument representing a product ID and returns the associated category ID by executing a SQL query that joins three tables (category, subcategory, and product). The retrieved category ID is stored in the variable subCategory, and if there is an error, it logs the exception using a logger object. Finally, it returns the retrieved category ID.

ImageProductDAO

No	Method	Description
01	void insertImgProduct(int id, String imgUrl, int productId)	Inserts a product image into a "ProductImage" table in the "shop_pet" schema, using a prepared statement to set fields with parameters passed to it. Any exceptions are logged by a built-in logger.

PetDAO

No	Method	Description
01	void insertPet(int id, Date dateOfBirth, String healthStatus, boolean gender, double weight, String color, boolean	Inserts pet information, such as ID, date of birth, gender, weight, color, vaccination status, and identification, into a SQL database table named pet. It establishes a connection to the database, creates a SQL query string, prepares a statement, sets parameter values, and executes an update. If there are any errors, it logs them to a Logger object.

	vaccinated, String identification)	

OriginDAO

No	Method	Description
01	void insertOriginProduct (int id, String region)	Inserts product origin information, such as ID and region, into a SQL database table named Origin. It establishes a connection to the database, creates a SQL query string, prepares a statement, sets parameter values, and executes an update. If there are any errors, it logs them to a Logger object.

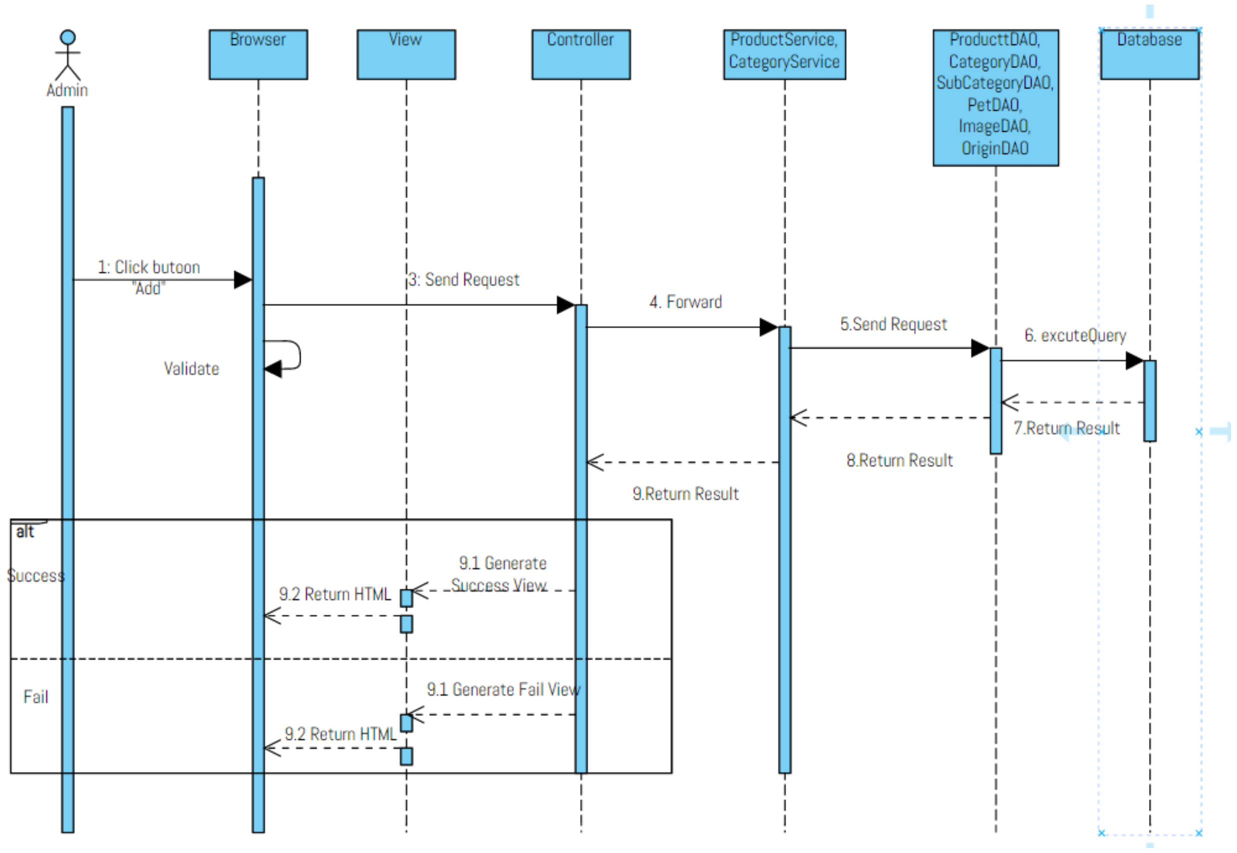
SubCategoryDAO

No	Method	Description
01	List<SubCategory> getNameSubCategory ByCategoryId(int categoryId)	Takes an integer argument representing a category ID and returns a list of SubCategory objects. It executes a SQL query that joins two tables based on their common column, converts the retrieved rows into SubCategory objects, adds them to a list, and returns the list. If there is an error, it logs the exception using a logger object.

ProductService

No	Method	Description
01	void InsertProduct(String name, String code, double price, int quantity, String description, int subId, int originId, List<ImageProduct> listImg)	Inserts a new product and image data into the database. It gets the maximum product and image IDs, increments them to create new IDs, and calls the appropriate insert methods using these IDs. If there is an error, it is not handled and will be thrown to the caller. Note that the method assumes that the listImg parameter contains at least one ImageProduct object.
02	void InsertProductIsPet(Product p, Pet p1, List<ImageProduct> listImg)	Inserts new product, pet, and image data into the database. It gets the maximum product, pet, and image IDs, increments them to create new IDs, and calls the appropriate insert methods using these IDs. If there is an error, it is not handled and will be thrown to the caller. Note that the method assumes that the listImg parameter contains at least one ImageProduct object.

c. Sequence Diagram(s)

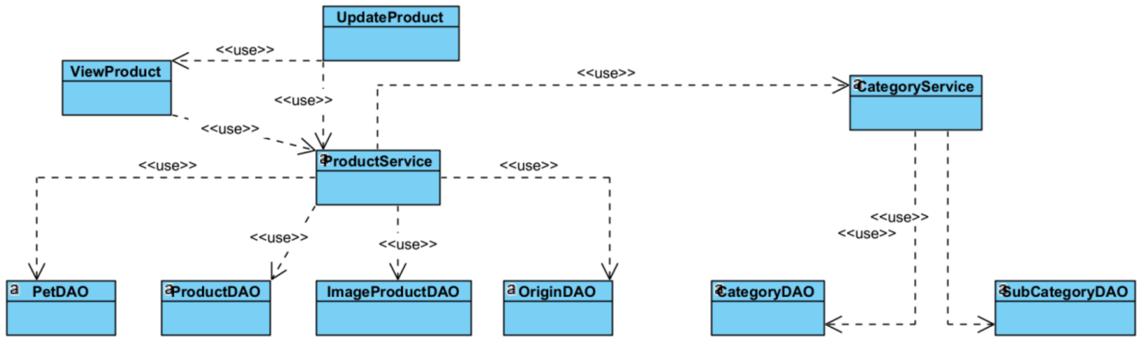


d. Database Queries

- INSERT INTO \`shop_pet\`.\`product\` (\`id\`, \`name\`, \`code\`, \`price\`, \`quantity\`, \`description\`,\`pet_id\`, \`subcategory_id\`,\`origin_id\`) VALUES(?,?,?,?,?,?,?,?,?)
- INSERT INTO \`shop_pet\`.\`product\` (\`id\`, \`name\`, \`code\`, \`price\`, \`quantity\`, \`description\`,\`subcategory_id\`,\`origin_id\`) VALUES(?,?,?,?,?,?,?,?)
- INSERT INTO \`shop_pet\`.\`ProductImage\` (\`id\`, \`image_url\`, \`product_id\`) VALUES(?,?,?)
- INSERT INTO \`shop_pet\`.\`pet\` (\`id\`, \`date_of_birth\`, \`health_status\`, \`gender\`, \`weight\`, \`color\`, \`vaccinated\`, \`identification\`) VALUES(?,?,?,?,?,?,?,?)
- INSERT INTO Origin (id, region) VALUES(?,?)
- select category_id from category c inner join subcategory as s inner join product p where c.id=s.category_id and p.subcategory_id=s.id and p.id=? ;
- select* from subcategory as s join category as c where s.category_id=c.id and c.id=?;

7. Update Product

a. Class Diagram



b. Class Specifications

ProductDAO

No	Method	Description
01	void updateProductIsPet(int id, String name, String code, double price, int quantity, String description, int subId, int originId)	Takes in several parameters including the ID of the product, its name, code, price, quantity, description, subcategory ID and origin ID. These parameters are used to update the corresponding fields in the 'product' table of the database using an SQL UPDATE statement with placeholders. Finally, the executeUpdate() method is called on the PreparedStatement object to modify the database, and any error that occurs is logged using the Logger class.
02	void updateProduct(int id, String name, String code, double price, int quantity, String description, int subId, int originId)	Uses an SQL UPDATE statement with placeholders to modify the fields of the 'product' table based on the input parameters passed into the method. Any error that occurs during this process is logged using the Logger class.

PetDAO

No	Method	Description
01	void updatePet(int id, Date dateOfBirth, String healthStatus,	Updates the details of a pet in the 'shop_pet' database. It uses an SQL UPDATE statement with placeholders to modify the fields of the 'pet' table based on the input parameters passed

	boolean gender, double weight, String color, boolean vaccinated, String identification)	into the method. Any error that occurs during this process is logged using the Logger class.

ImageProductDAO

No	Method	Description
01	void updateImgProduct (int id,String imgUrl)	Updates the image URL of a product in the 'ProductImage' table of the 'shop_pet' database. It uses an SQL UPDATE statement with placeholders to modify the 'image_url' field based on the input parameters passed into the method. Any error that occurs during this process is logged using the Logger class.

OriginDAO

No	Method	Description
01	int getIdOriginByName (String name)	Finds the ID of an origin from a "origin" table in a database where the region matches a given name. It returns the ID as an integer, or 0 if there is no match. The code sets up a connection, executes a SQL query using a prepared statement, and retrieves the ID value from the result set if it exists. Any exceptions are logged.

SubCategoryDAO

No	Method	Description
01	List<SubCategory> getNameSubCategoryByCategoryId(int categoryId)	Takes an integer argument representing a category ID and returns a list of SubCategory objects. It executes a SQL query that joins two tables based on their common column, converts the retrieved rows into SubCategory objects, adds them to a list, and returns the list. If there is an error, it logs the exception using a logger object.

ProductService

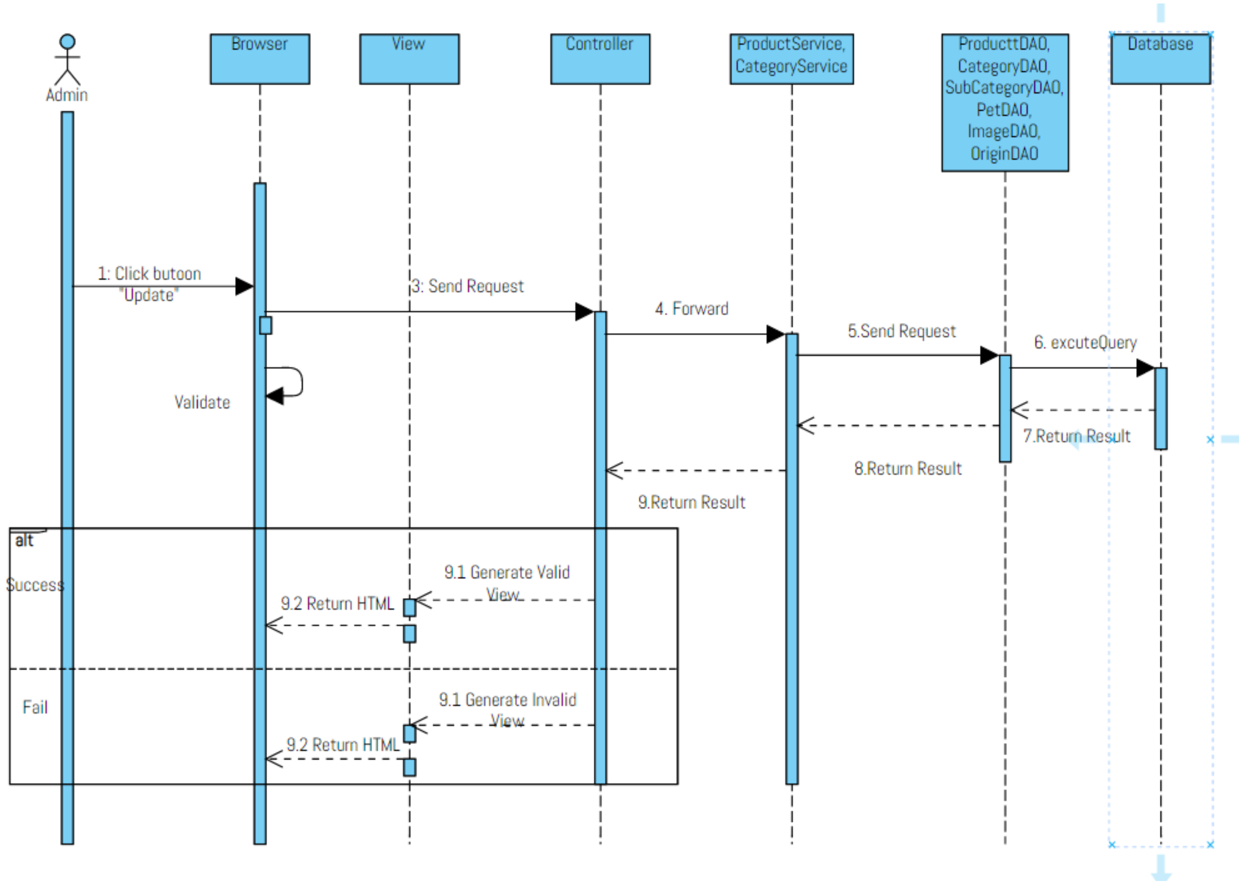
No	Method	Description
01	updateProductAccessory(int id, String name, String code, double price, int quantity, String description, int subId, int originId, String img)	Updates a product accessory's information and image in a database, using productDAO and ImageProductDAO objects to update the relevant fields.

02	void updateProductPet(int id, String name, String code, double price, int quantity, String description, int subId, int petId, int originId, Date dateOfBirth, String healthStatus, boolean gender, double weight, String color, boolean vaccinated, String identification, String img)	Updates a pet product's information and image in a database using productDAO, petDAO, and ImageProductDAO objects. It takes several parameters including the product ID, name, code, price, quantity, description, subcategory ID, pet ID, origin ID, date of birth, health status, gender, weight, color, vaccination status, identification number, and an image.

CategoryService

No	Method	Description
01	int getCategoryIdfromproductId(int productId)	Takes an integer argument representing a product ID and returns the associated category ID by executing a SQL query that joins three tables (category, subcategory, and product). The retrieved category ID is stored in the variable subCategory, and if there is an error, it logs the exception using a logger object. Finally, it returns the retrieved category ID.

c. Sequence Diagram(s)

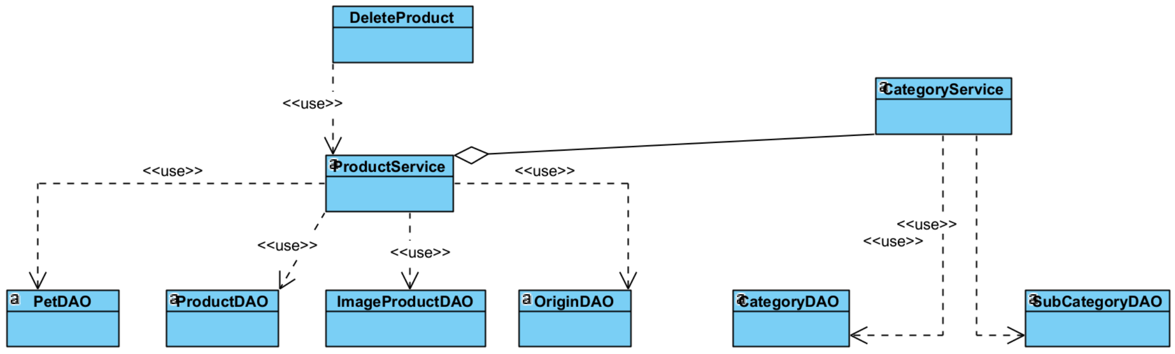


d. Database Queries

- UPDATE `shop_pet`.`product` set `name`=?, `code`=?, `price`=?, `quantity`=?, `description`=?,`subcategory_id`=?,`origin_id`=? WHERE id = ?
- UPDATE `shop_pet`.`product` set `name`=?, `code`=?, `price`=?, `quantity`=?, `description`=?,`subcategory_id`=?,`origin_id`=? WHERE id = ?
- UPDATE `shop_pet`.`pet` SET `date_of_birth`=?, `health_status`=?, `gender`=?, `weight`=?, `color`=?, `vaccinated`=?, `identification`=? WHERE id = ?
- UPDATE ProductImage set image_url= ? WHERE product_id = ?
- select id from origin where region like ?

8. Delete Product

a. Class Diagram



b. Class Specifications

ImageProductDAO

No	Method	Description
01	void deleteImgProduct (int id)	Deletes product images from a database based on the provided product ID. It uses a prepared SQL query to delete all rows of the "productimage" table where the "product_id" column matches the provided ID. If any exceptions occur during the deletion process, they are logged using a Java Logger object.

PetDAO

No	Method	Description
01	void deletePet(int id)	Deletes a row from a database table named "pet" where the value in the "id" column matches the provided ID parameter. If

		any errors occur during this process, it logs an error message with details about the exception using a Logger object.

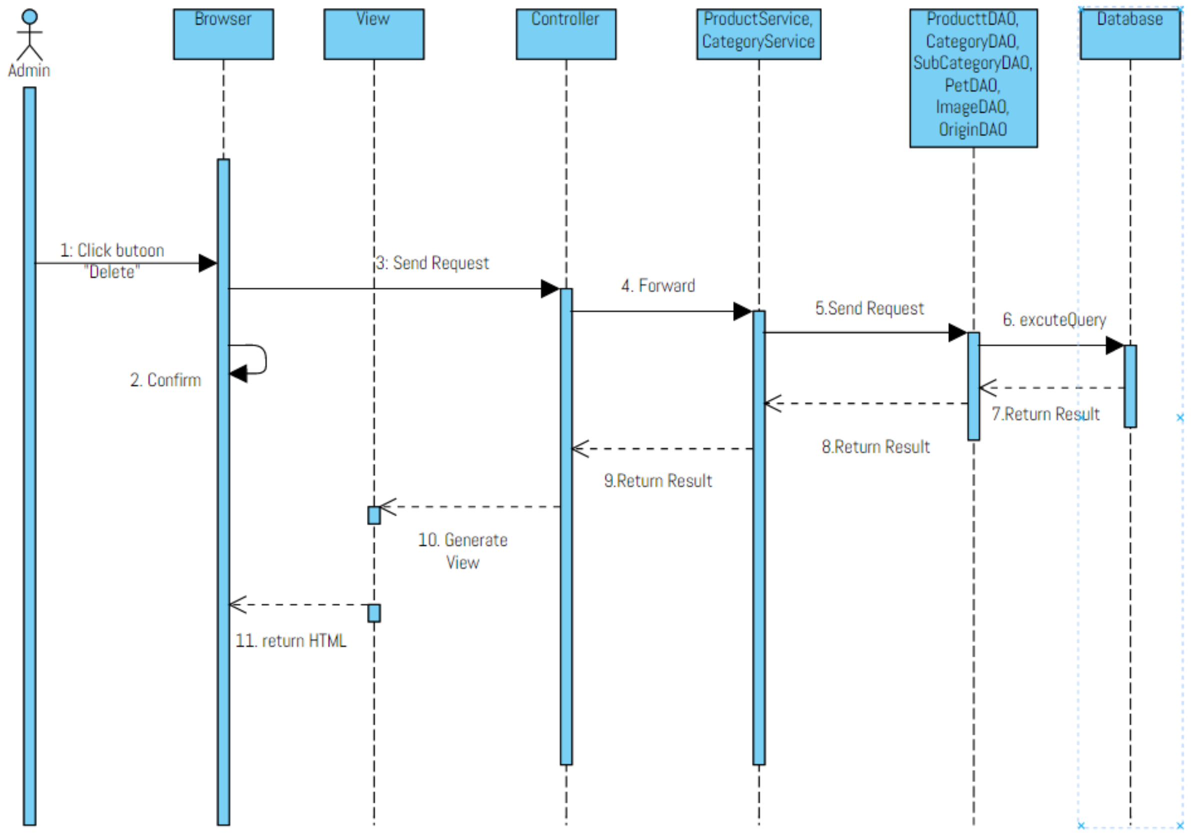
ProductDAO

No	Method	Description
01	void deleteProduct(int id)	Deletes a row from a database table named "product" where the value in the "id" column matches the provided ID parameter. If any errors occur during this process, it logs an error message with details about the exception using a Logger object.

ProductService

No	Method	Description
01	void deleteProductService(int id)	Deletes a product from a database. If the product has an associated pet, it also deletes the pet and any associated images. If not, it only deletes any associated images.

c. Sequence Diagram(s)

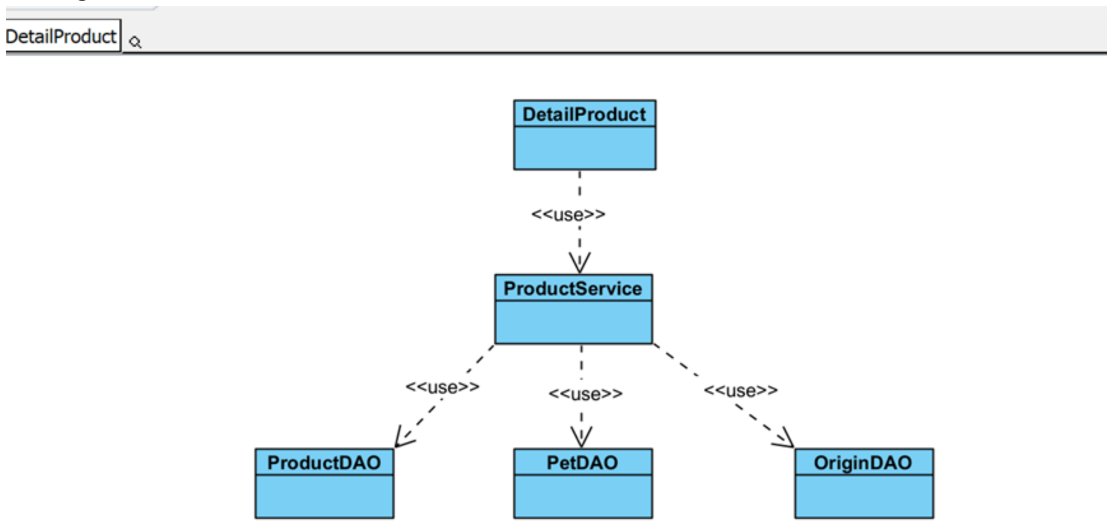


d. Database Queries

- DELETE from productimage WHERE product_id = ?
- DELETE from pet WHERE id = ?
- delete from product where id = ?

9. Product detail

a. Class Diagram



b. Class Specifications

ProductDAO

No	Method	Description
01	int getsubCategoryfromproductId(int productId)	Defines a method called "getsubCategoryfromproductId" that takes an integer parameter "productId". It retrieves the "subcategory_id" from a row in a database table named "product" where the value in the "id" column matches the provided productId parameter. If any errors occur, it logs an error message using a Logger object. The method returns the subcategory ID as an integer value.
02	int getCategoryIdfromproductId(int productId)	Defines a method called "getCategoryIdfromproductId" that retrieves the "category_id" from a database table named "product" by joining with the "subcategory" and "category" tables. It takes an integer parameter "productId" and returns the category ID as an integer value. If any errors occur, it logs an error message using a Logger object.
03	boolean getPetIdFromProductId(int productId)	Retrieves the pet ID associated with a given product ID from a database and returns a boolean value indicating whether

		or not the pet ID was found. It also contains commented out code related to closing the database connection.

CategoryDAO

No	Method	Description
01	List<Category> getAllCategory()	Retrieves all categories from a "category" database table and returns them as a List of Category objects using a prepared statement and ResultSet.

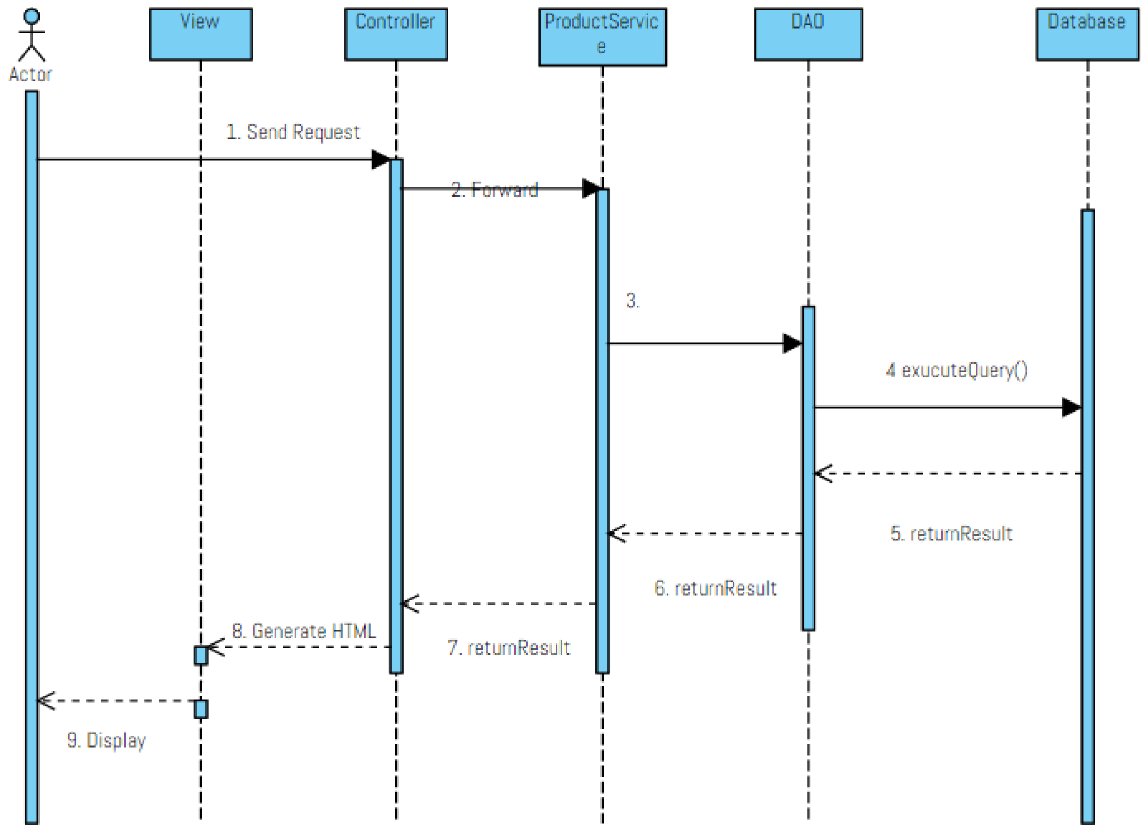
ProductService

No	Method	Description
01	Product getAllInforProductPet(int x)	Retrieves various types of information related to a product from a database using different DAO objects, populates a Product object with the retrieved data, and returns it. The method takes a product ID as input and is part of a larger program dealing with products and their associated information.
02	Product getAllInforProductAccessory(int x)	Retrieves information related to an accessory product from a database using different DAO objects, sets the retrieved information in a Product object, and returns it. The method takes a product ID as input and is part of a larger program dealing with accessories and their associated information.

CategoryService

No	Method	Description
01	List<Category> getAllCategory()	Retrieves a list of categories using a DAO and returns it. It's named "getAllCategory" and likely retrieves all categories stored in a data storage system.

c. Sequence Diagram(s)

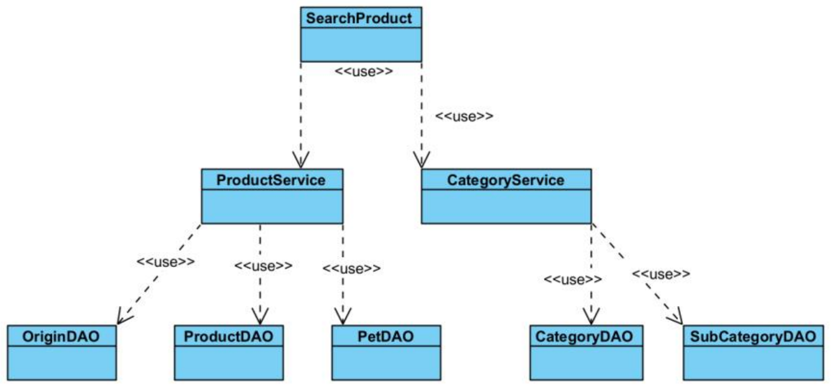


d. Database Queries

- *SELECT * FROM product WHERE id = ?*
- *SELECT * FROM `shop_pet`.`ProductImage` WHERE product_id= ?*
- *SELECT * FROM shop_pet.Pet WHERE id = ?*
- *SELECT * FROM shop_pet.Origin WHERE id = ?*
- *select * from subcategory where id = ?*
- *select * from category where id = ?*

10. Search

a. Class Diagram



b. Class Specifications

CategoryDAO

No	Method	Description
01	List<Category> getAllCategory()	Retrieves all categories from a "category" database table and returns them as a List of Category objects using a prepared statement and ResultSet.

ProductService

No	Method	Description
01	List<Product> getSearchAllProductbyCategoryID(int count, int index, int categoryId, boolean check)	Retrieves a list of products by category ID, along with their images, pet information, origin information, and subcategory information.
02	List<Product> filterColor(int categoryId, String color)	Filters a list of products by color within a given category. It retrieves the necessary data for each product and returns the filtered and enriched list.
03	List<Product> searchProduct(int categoryId, String search, boolean check)	Retrieves a list of Product objects based on the given categoryId, search string, and boolean value. It determines which DAO method to call based on the boolean parameter, then retrieves additional information for each product using other DAOs and services. Finally, it returns the list of Product objects with the added information.

ProductDAO

No	Method	Description

01	List<Product> SearchProductByCategoryPet(int categoryid, int count, int index)	Searches for products in a pet shop's inventory by category, with pagination.
02	List<Product> SearchProductByCategoryAsscessory(int categoryid, int count, int index)	Searches for products in a pet shop's inventory by accessory category, with pagination.
03	boolean getPetIdFromCategoryId(int categoryid)	Retrieves pet_id from a database table called "product" based on the given categoryid. It returns a boolean value indicating whether or not there are matching pet IDs in the result set of the query.
04	int getNumberProductsByCategoryID(int categoryId)	Counts the number of products associated with a given categoryId. It executes a SQL query that joins tables "category", "subcategory", and "product" to count the number of products associated with the given categoryId. The method returns the count or 0 if there are no results.

SubCategoryDAO

No	Method	Description
01	List<SubCategory> getNameSubCategoryByCategoryId(int categoryId)	Takes an integer argument representing a category ID and returns a list of SubCategory objects. It executes a SQL query that joins two tables based on their common column, converts the retrieved rows into SubCategory objects, adds them to a list, and returns the list. If there is an error, it logs the exception using a logger object.

PetDAO

No	Method	Description
01	List<Pet> getListPetByID(int id)	Retrieves a Pet object from a database using its ID. It establishes a connection to the database, prepares and executes a SQL query, then creates a new Pet object for each row in the ResultSet, adds it to an ArrayList, and returns the final list.

OriginDAO

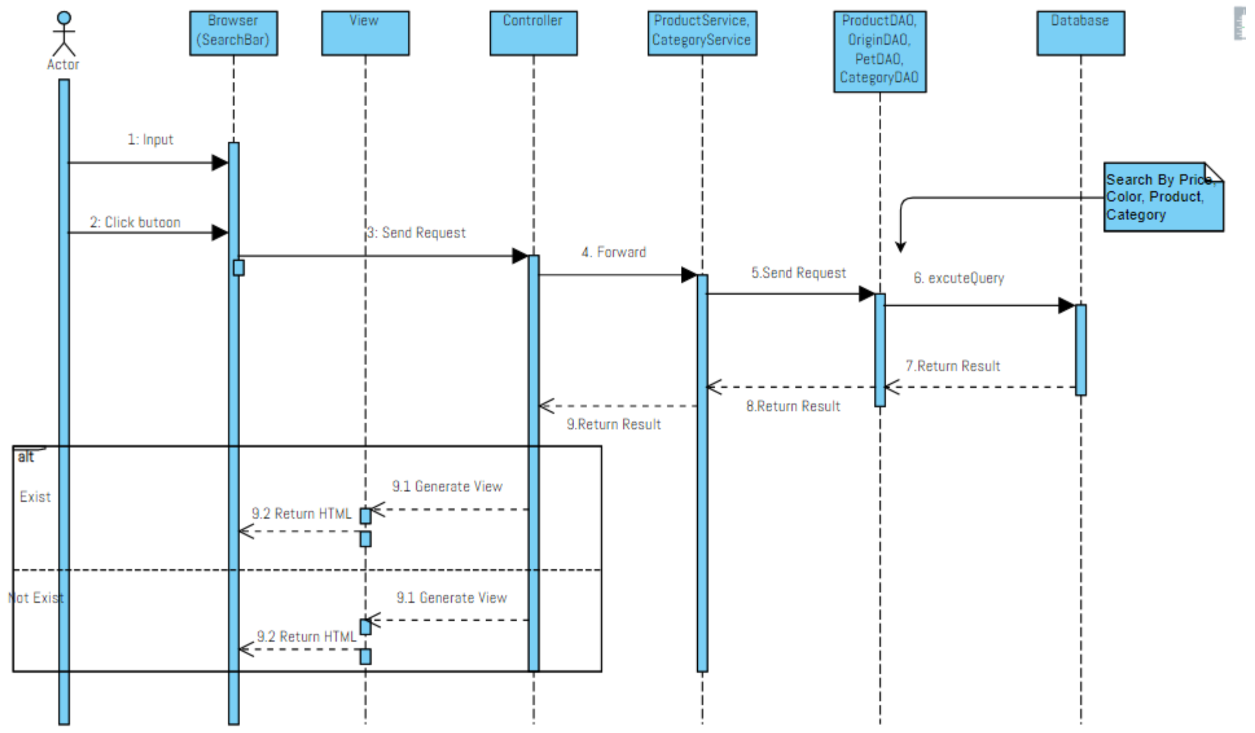
No	Method	Description
01	List<Origin> getListOrigins(int id)	Retrieves an Origin object from a database using its ID. It constructs a SQL query, establishes a connection to the database, executes the query with a PreparedStatement

		object, creates a new Origin object for each row in the ResultSet, adds it to an ArrayList, and returns the final list.

CategoryService

No	Method	Description
01	List<SubCategory> getSubAndCategoryByID(int x)	Retrieves a list of SubCategory objects and their corresponding Category objects from a database using a given ID, enriches the SubCategory objects with their associated Category objects, and returns the final list.
02	List<Category> getAllCategory()	Retrieves a list of categories using a DAO and returns it. It's named "getAllCategory" and likely retrieves all categories stored in a data storage system.

c. Sequence Diagram(s)



d. Database Queries

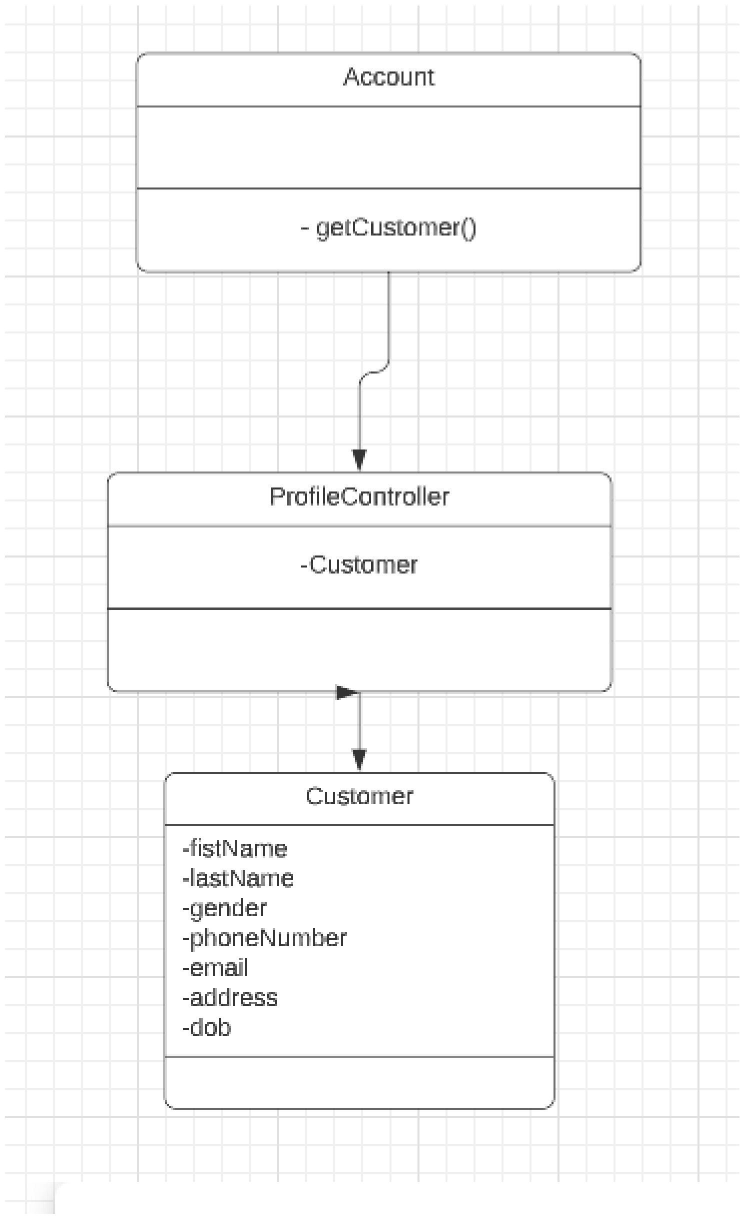
- *WITH x AS (SELECT @row := @row + 1 AS r, h.* FROM shop_pet.product h, (SELECT @row := 0) AS init WHERE name like ?) SELECT * FROM x WHERE r BETWEEN (? - 1) * 6 + 1 AND ? * 6*
- *SELECT * FROM `shop_pet`.`ProductImage` WHERE product_id= ?*
- *SELECT * FROM shop_pet.Pet WHERE id = ?*
- *SELECT * FROM shop_pet.Origin WHERE id = ?*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join*

shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1 = 1

- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1 = 1*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where spc.id = ? limit ? offset*
- *SELECT * FROM shop_pet.product spp2
  inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1=1*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1=1"*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1 = 1*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where sps.id=*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where sps.id= ?*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.Pet spp on spp.id=spp2.pet_id inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1 = 1*
- *SELECT * FROM shop_pet.product spp2inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1=1*
- *SELECT * FROM shop_pet.product spp2 inner join shop_pet.origin spo on spp2.origin_id=spo.id inner join shop_pet.subcategory sps on sps.id=spp2.subcategory_id inner join shop_pet.category spc on sps.category_id=spc.id where 1=1*
-

11 . View Profile feature

a. Class Diagram

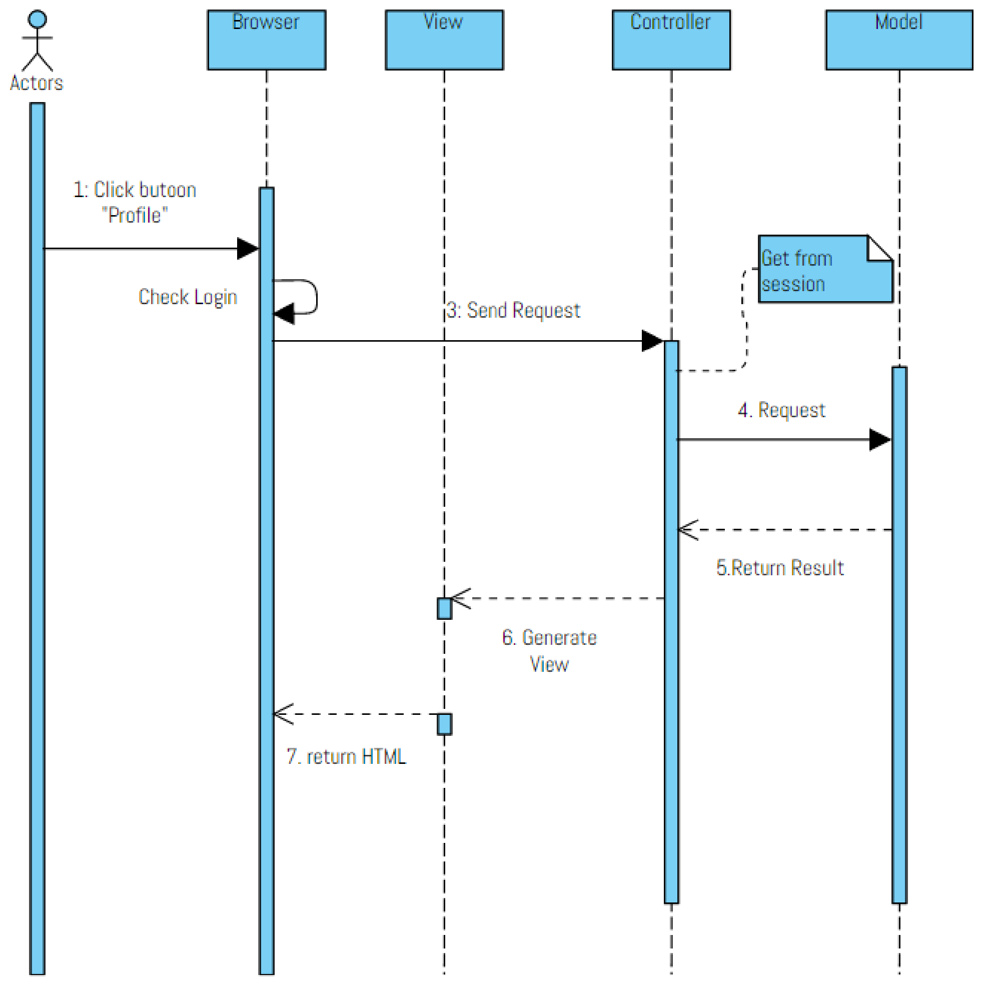


b. Class Specifications

ProfileController

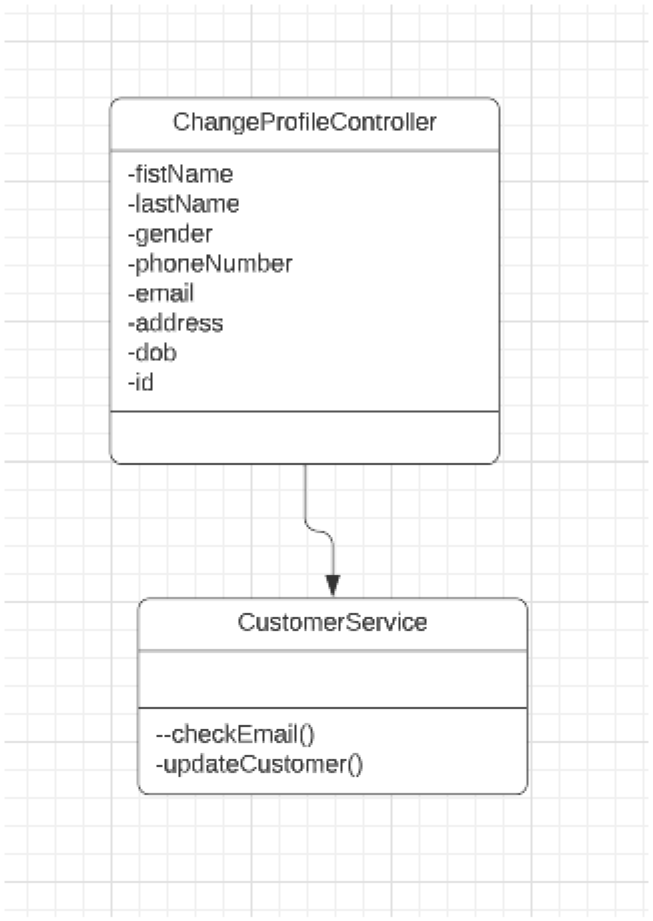
No	Method	Description
01	doGet	Retrieves the account information stored in the session, checks the role of the user, and accordingly forwards the request to the profile JSP file or logs out the user by removing the account information from the session. It also prints appropriate error messages and writes the account information to the response output stream.

c. Sequence Diagram(s)



12. Update Profile

a. Class Diagram



b. Class Specifications

ChangeProfileController

No	Method	Description
01	doPost	Retrieves customer information from the request object, creates a Customer object and sets its properties, then updates the customer and sets the account and customer session attributes. If an exception of type SQLException is caught, it prints the error message and forwards the request to the edit profile JSP file.

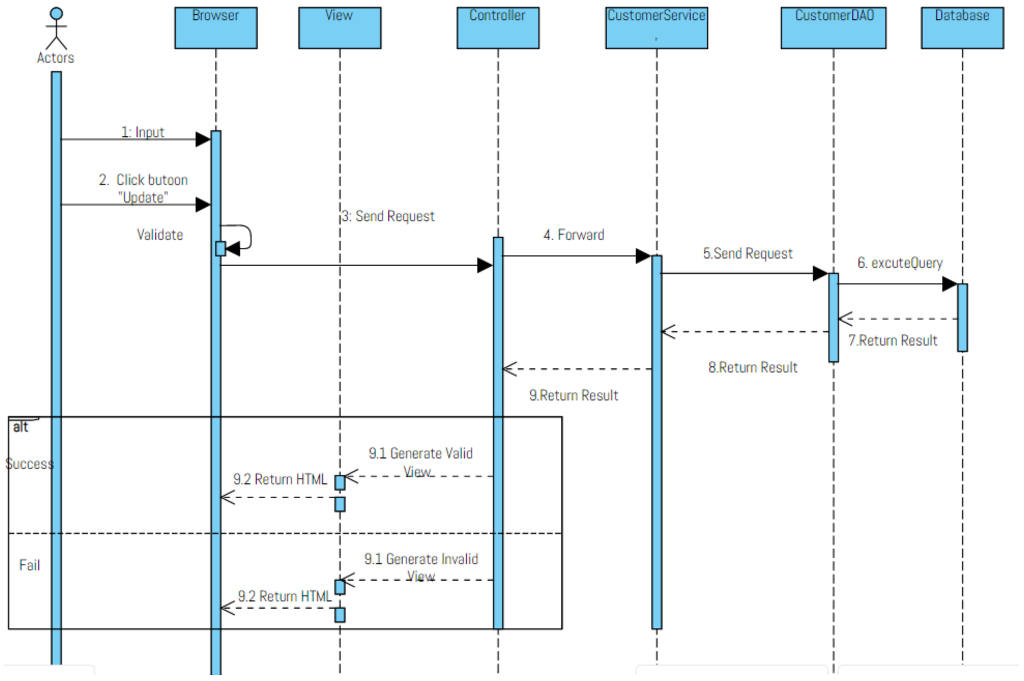
CustomerService

No	Method	Description
01	void update(Customer cus)	Updates the customer only if the email of the customer already exists or is the same as the one provided in the input, otherwise throws a SQLException.

CustomerDAO

No	Method	Description
01	boolean checkExistEmail(String email)	Checks the existence of the given email in the Customer table by executing a SQL query using a prepared statement, then returns a boolean value indicating whether the email exists or not.
02	Customer getCustomerById (int id)	Retrieves the customer record with the given id from the Customer table, sets the properties of the customer object using the CustomerMapping.daotoObject(rs) method, then returns the customer object.
03	void updateCustomer (Customer customer)	akes a Customer object as input and sets the values of the object's fields to corresponding columns in the table. The SQL query updates the rows in the table where the account ID matches the provided customer's ID. Finally, the method closes the database connection and statement objects.

c. Sequence Diagram(s)

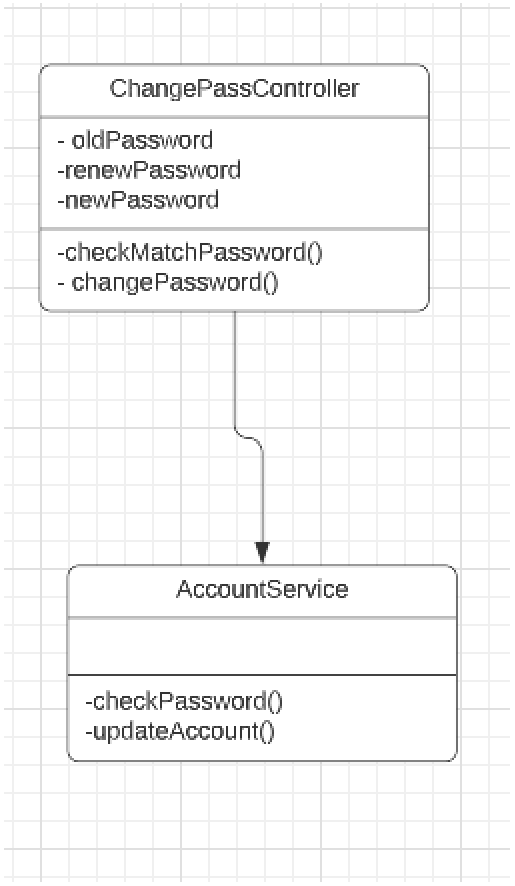


d. Database Queries

- SELECT COUNT(*) FROM shop_pet.Customer WHERE email = ?
- SELECT * FROM shop_pet.Customer WHERE id = ?
- UPDATE shop_pet.Customer SET firstName = ?, lastName = ?, gender = ?,address = ?,email = ?, birth_of_date = ?, phone = ? WHERE account_id = ?

13. Change Password

a. Class Diagram



b. Class Specifications

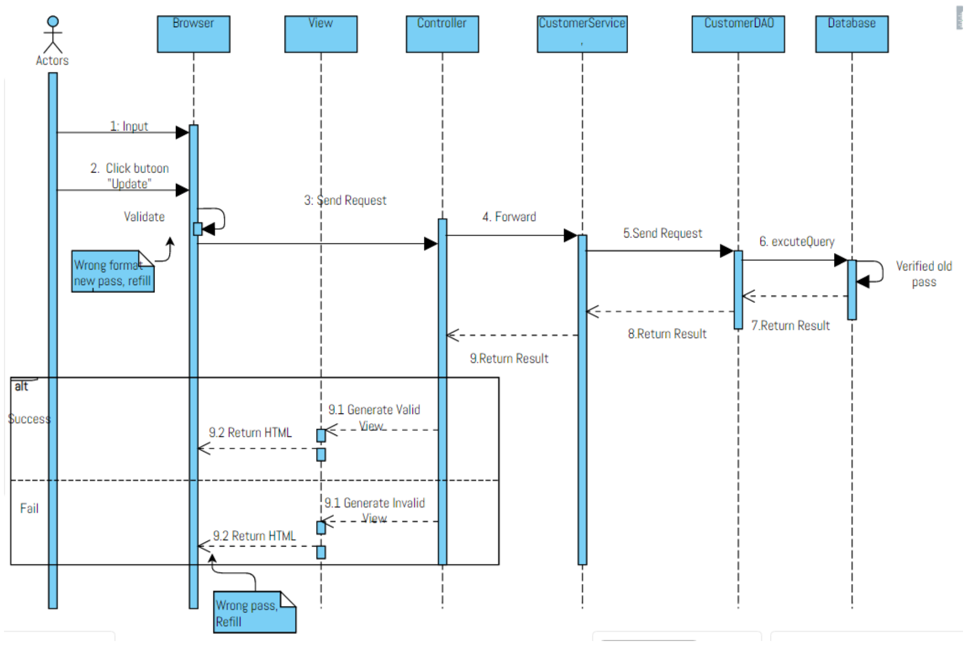
ChangePasswordController

No	Method	Description
01	doPost	Request to change the user's password, retrieves form field values, checks for errors, updates the user account, and forwards the request to a JSP page with success or error messages.

AccountService

No	Method	Description
01	void update(Account account)	Updates an "Account" object in a database if the account's username is unique or the same as the existing one. It retrieves all customers from the database and iterates through them to find the customer associated with the account being updated. If a matching customer is found, the method calls the "update()" method of the "accountDAO" object.
02	checkPassword()	New password and confirm password must equal

c. Sequence Diagram(s)

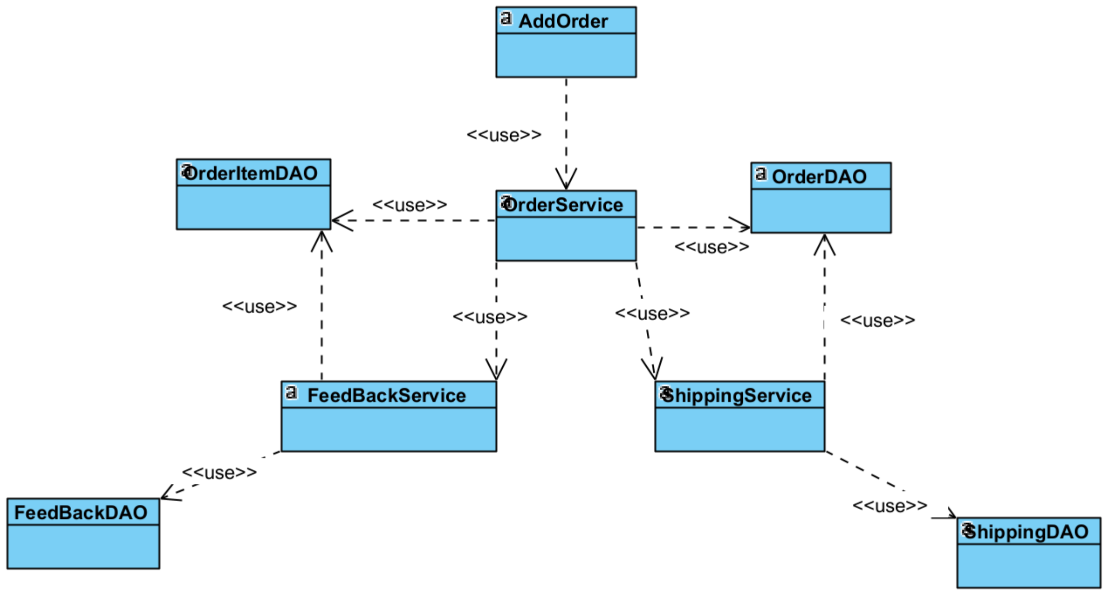


d. Database Queries

- UPDATE Account SET username=?, password=?, role=?, deactive_at=?, deactive_reason=? WHERE id=?

14. Create Order

a. Class Diagram



b. Class Specifications

OrderService

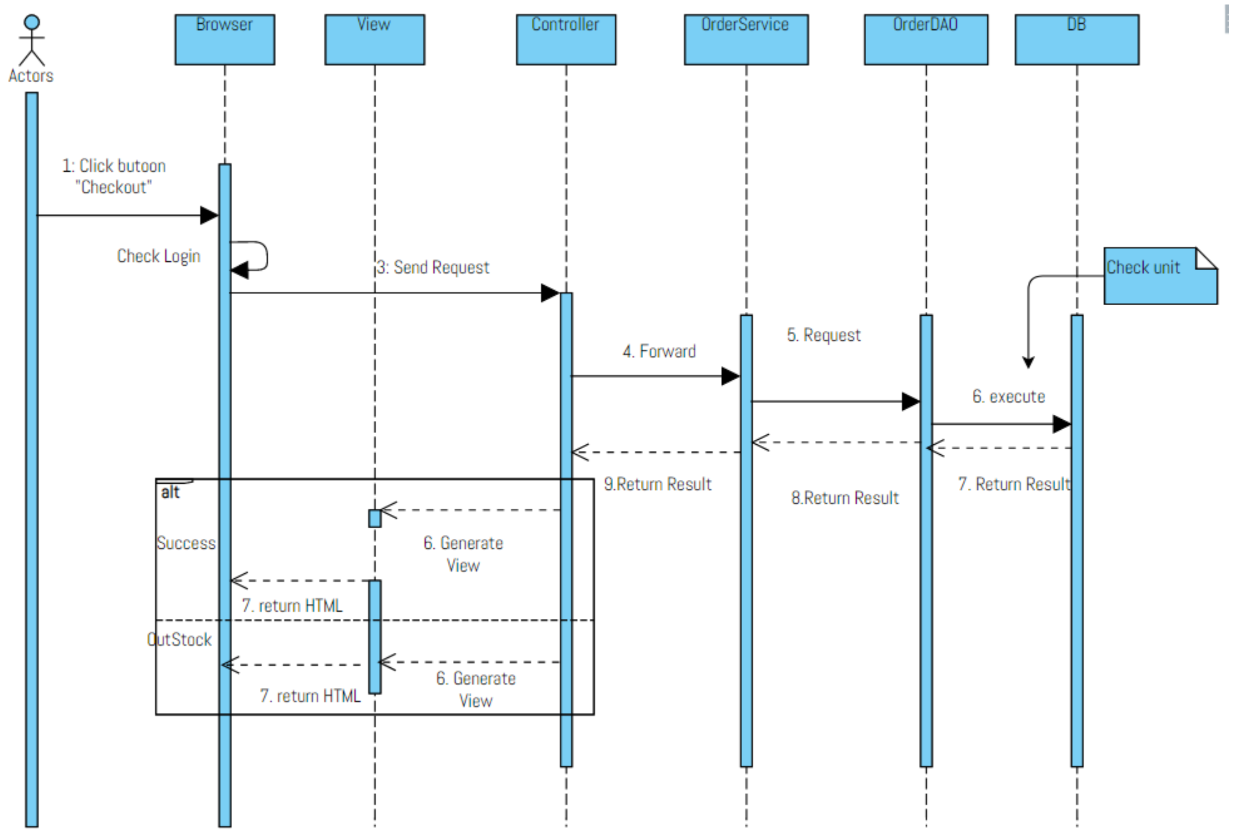
No	Method	Description
01	List<OrderInfo> getOrderInfo(int accountId)	Takes an accountId parameter and returns a list of order information objects for that account. It uses a DAO instance called orderInfoDAO to access the data source.
02	createOrder(Account account, String customerName, String customerPhone, String customerAddress)	Takes account, customer info, and creates an order with a list of items from the user's cart. It updates the product quantities and clears the cart. If the total price is zero or negative, it returns without creating the order.

OrderDAO

No	Method	Description
01	void createOrder(Order order)	Creates a new order in the database for the given Order object and its associated OrderItems. It executes SQL statements to create the order record with customer info, total price, and account ID, retrieves the generated key for the new order, and

		inserts each item into the orderitem table. If any exceptions occur, it rolls back the entire transaction.

c. Sequence Diagram(s)

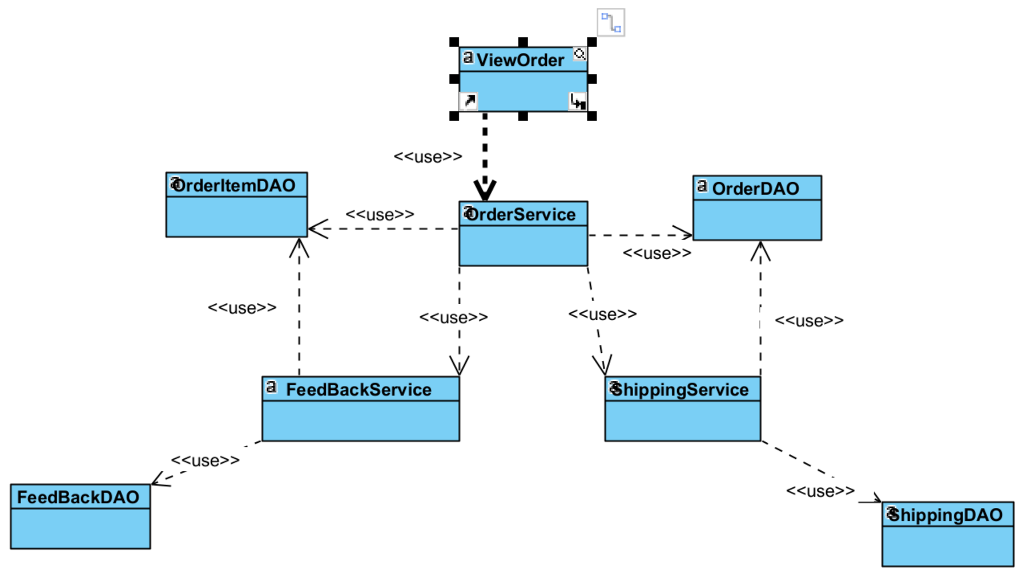


d. Database Queries

- select * from shop_pet.orderInfo where account_id = ?
- insert into shop_pet.`order` (`customer_name`, `customer_phone`, `customer_address`, created_at,`status`,`total_price` , account_id) values (?,?,?,now(), null, ?, ?)
- insert into shop_pet.orderitem (order_id, product_id, product_name,product_image, price, quantity) values (?, ?, ?,?, ?, ?

15. View Order

a. Class Diagram



b. Class Specifications

Order Class

No	Method	Description
01	*<method name>*	*<Description of the method, including the inputs, outputs & internal method processing>*

ABC Class

No	Method	Description
01	*<method name>*	*<Description of the method, including the inputs, outputs & internal method processing>*

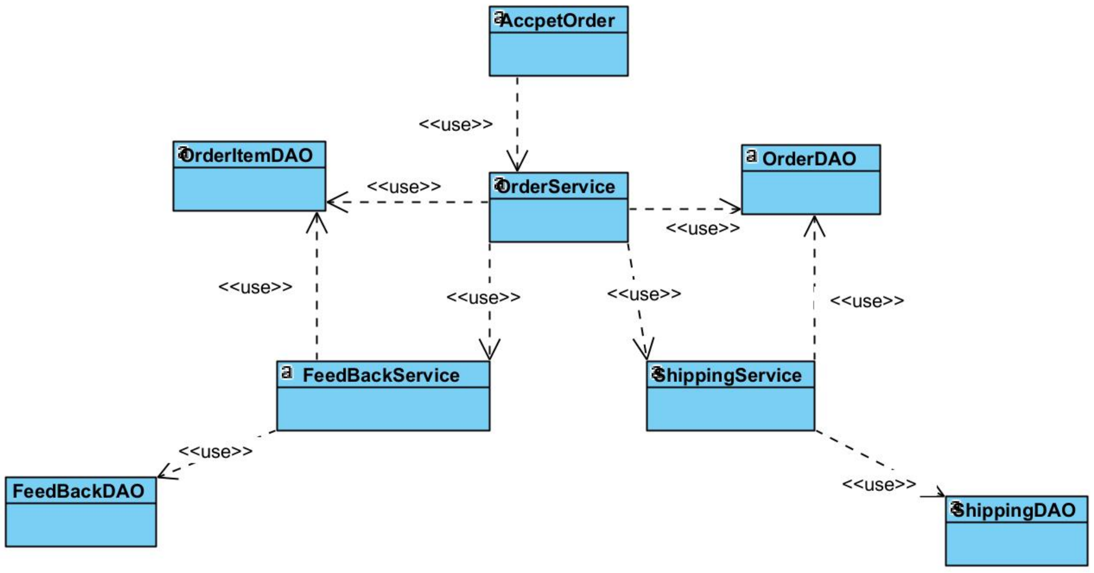
c. Sequence Diagram(s)

d. Database Queries

String sql = "select distinct o.* from shop_pet.`order` o left join shop_pet.orderitem oi on o.id = oi.order_id left join shop_pet.product p on oi.product_id = p.id left join shop_pet.shipping s on s.id = o.shipping_id where o.account_id = ? ";

16. Accpet_Deny Order

a. Class Diagram



b. Class Specifications

OrderService

No	Method	Description
01	getAllOrder(String customerName, String customerPhone, String customerAddress, String orderStatus, String shippingStatus, String dateFrom, String dateTo, int limit, int pageIndex)	Retrieves a list of orders with specific conditions, including associated shipping and order item data. It takes in parameters such as customer information, order and shipping status, and date range. The method retrieves the relevant data and adds it to each order before returning the modified list.

ShippingService

No	Method	Description
01	List<Shipping> getShippingsById(List<Integer> shippingIds)	Takes as input a list of shipping IDs, and returns a list of Shipping objects by calling the getShippingsById method from the shippingDAO object.

ShippingDAO

No	Method	Description
01	List<Shipping> getShippingsById(List<Integer> shippingIds)	Retrieves a list of Shipping objects by their IDs. It constructs and executes a dynamic SQL query using parameter binding, converts the resulting rows into Shipping objects, adds them to an output list, and returns this list.

OrderItem

No	Method	Description
01	List<OrderItem> getOrderItemsByOrderIds(List<Integer> orderIds)	Retrieves a list of OrderItem objects by their order IDs. It constructs and executes a dynamic SQL query using parameter binding to prevent SQL injection attacks. The resulting rows are converted into OrderItem objects which are added to an output list, before returning the list.

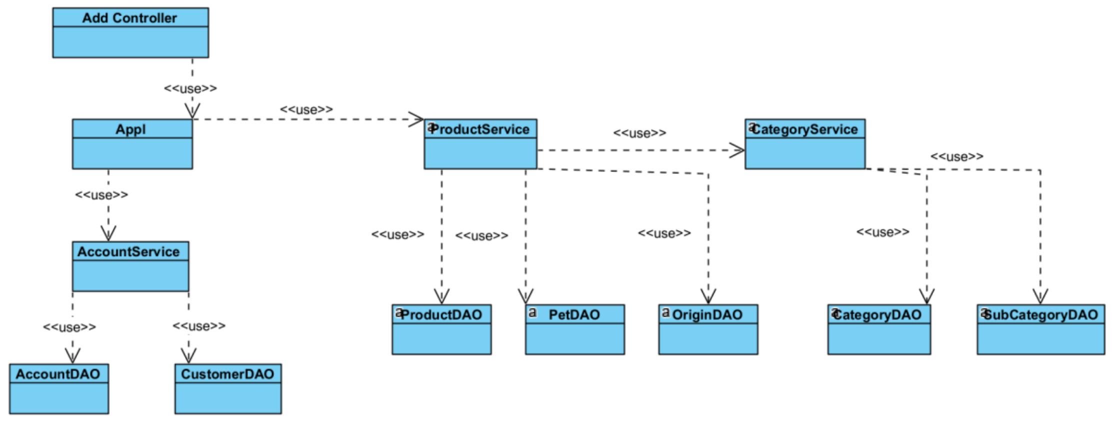
c. Sequence Diagram(s)

d. Database Queries

- select * from shop_pet.Shipping where id in (0
- select * from shop_pet.OrderItem where order_id in (0

17. Add to Cart

a. Class Diagram



b. Class Specifications

CartService

No	Method	Description
01	void addCart(int accountId, int productId, int quantity)	Adds a product to the cart for a given account ID and product ID with a specific quantity. It updates an existing cart item or creates a new one if it does not exist, ensuring that the quantity is within bounds. If the new quantity is negative or zero, the cart item is deleted.

AddController

No	Method	Description
01	addCartNotLogin(HttpServletRequest request, HttpServletResponse response, int productId, int quantity)	Adds a product to the cart for users who are not logged in. It updates the quantity of an existing cookie if the product ID matches, otherwise it creates a new cookie with the product ID and quantity. Finally, it adds the resulting cookie to the response object.
02	void addCartInLogin(int accountId, int productId, int quantity)	Adds a product to the cart for logged-in users by calling the addCart method of a CartService object with the accountId, productId, and quantity as parameters.

CartItemDAO

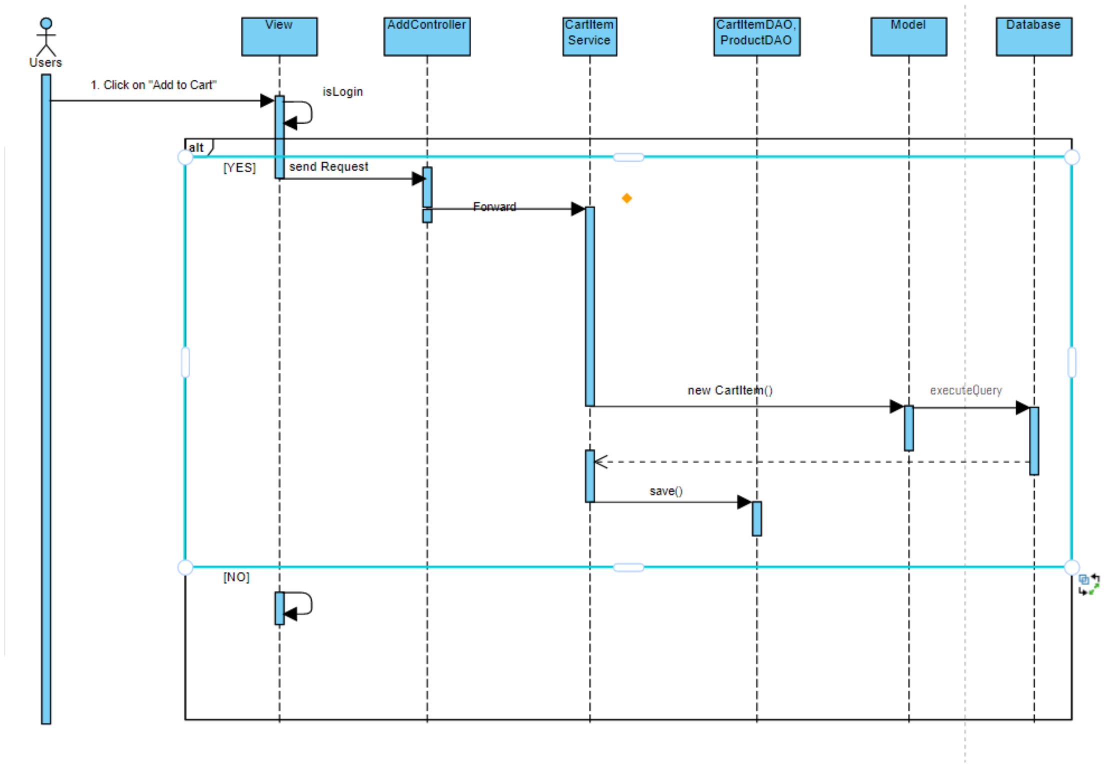
No	Method	Description
01	CartItem getCartItem(int	Retrieves a CartItem object given the cartId and productId. It constructs and executes a SQL query to retrieve the relevant row from the database, converts it into a CartItem object, and

	cartId, int productId)	returns this object. If an error occurs during execution, the method logs an exception and returns null.
02	void createCartItem(int cartId, int productId, int quantity)	Creates a new CartItem in the database using an SQL insert query with the cartId, productId, and quantity. If successful, the changes are committed to the database. Any exceptions cause a rollback. Finally, the method closes the database connection.

ProductDAO

No	Method	Description
01	Product getProductById(int id)	Retrieves a Product object given its ID. It constructs and executes a SQL query to retrieve the relevant product row from the database, converts it into a Product object, retrieves its associated image from the database, and returns this object. If an error occurs during execution, null is returned.

c. Sequence Diagram(s)

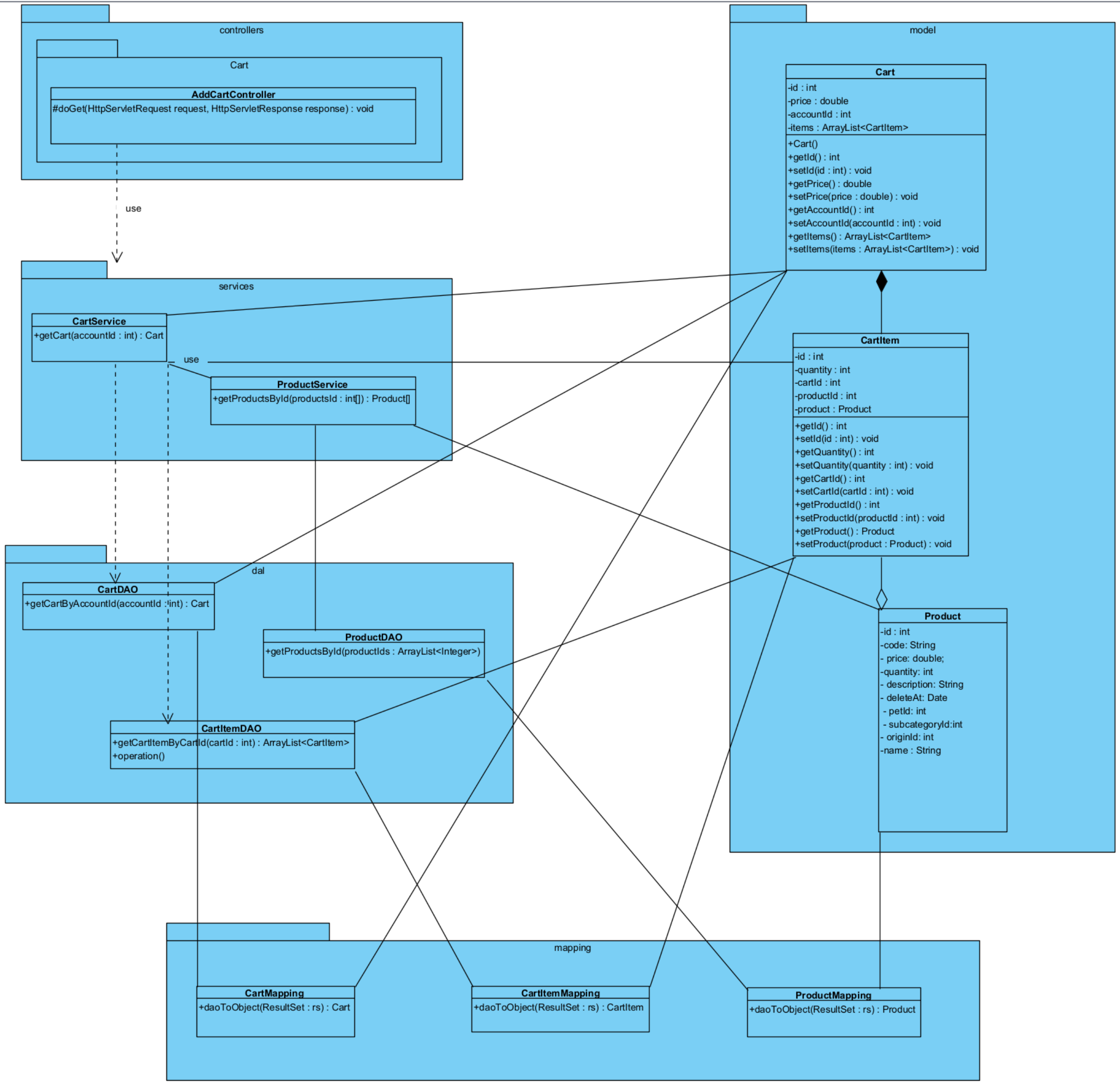


d. Database Queries

- SELECT * FROM shop_pet.cartitem where cart_id = ? and product_id = ?
- SELECT * FROM product WHERE id =
- insert into shop_pet.cartitem (quantity,cart_id,product_id) values (?, ?, ?);

18. View Cart

a. Class Diagram


controllers
Cart
AddCartController
#doGet(HttpServletRequest request, HttpServletResponse response) : void
use
services
CartService
+getCart(accountId : int) : Cart
use
ProductService
+getProductsById(productsId : int[]) : Product[]
dal
CartDAO
+getCartByAccountId(accountId : int) : Cart
ProductDAO
+getProductsById(productIds : ArrayList<Integer>)
CartItemDAO
+getCartItemByCartId(cartId : int) : ArrayList<CartItem>
+operation()
mapping
CartMapping
+daoToObject(ResultSet : rs) : Cart
CartItemMapping
+daoToObject(ResultSet : rs) : CartItem
ProductMapping
+daoToObject(ResultSet : rs) : Product
model
Cart
-id : int
-price : double
-accountId : int
-items : ArrayList<CartItem>
+Cart()
+getId() : int
+setId(id : int) : void
+getPrice() : double
+setPrice(price : double) : void
+getAccountId() : int
+setAccountId(accountId : int) : void
+getItems() : ArrayList<CartItem>
+setItems(items : ArrayList<CartItem>) : void
CartItem
-id : int
-quantity : int
-cartId : int
-productId : int
-product : Product
+getId() : int
+setId(id : int) : void
+getQuantity() : int
+setQuantity(quantity : int) : void
+getCartId() : int
+setCartId(cartId : int) : void
+getProductId() : int
+setProductId(productId : int) : void
+getProduct() : Product
+setProduct(product : Product) : void
Product
-id : int
-code: String
- price: double;
-quantity: int
- description: String
- deleteAt: Date
- petId: int
- subcategoryId:int
- originId: int
-name : String

b. Class Specifications

AddCartController

No	Method	Description
01	doGet(HttpServletRequest request, HttpServletResponse response) : void	Connect to web

Cart Service

No	Method	Description
01	getCart(accountId : int) : Cart	get Cart with CartItem by Account Id

Product Service

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

CartDAO

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

CartItemDAO

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

ProductDAO

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

CartMapping

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

CartItemMapping

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

ProductMapping

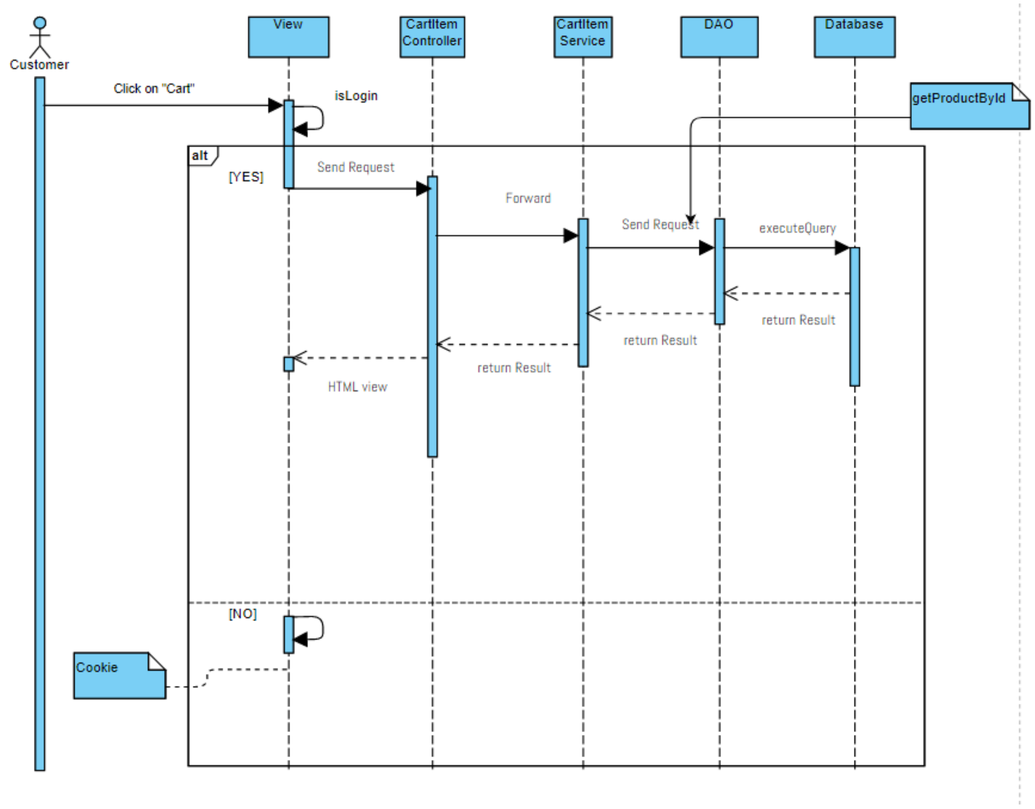
No	Method	Description

01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

Cart

No	Method	Description
01	getProductsById(productsId : int[]) : Product []	get Product to map with CartItem

c. Sequence Diagram(s)

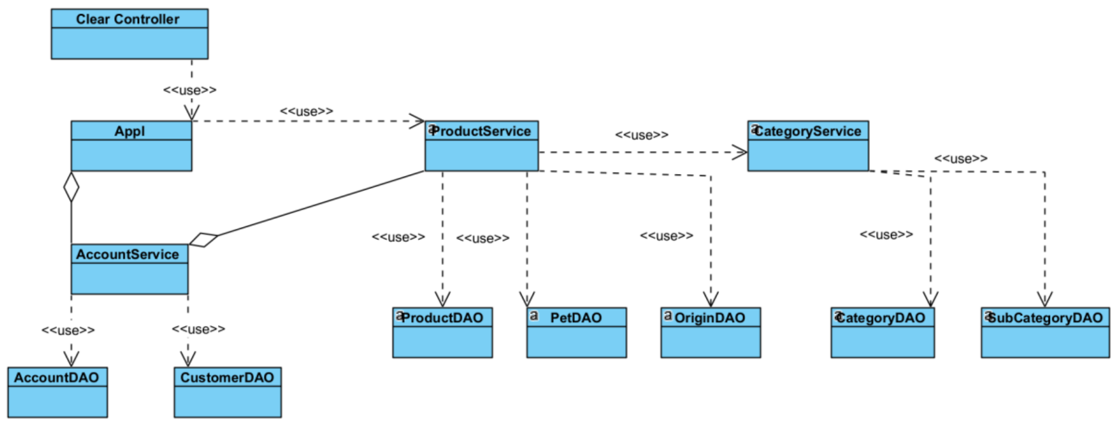


d. Database Queries

- select * from shop_pet.Product where id in (

19. Clear cart

a. Class Diagram



b. Class Specifications

Controller

No	Method	Description
01	clearProductCartNotLogin	Clears a specific product from a shopping cart cookie when a user is not logged in. It extracts the product ID from the cookie name, matches it with the given productId parameter, and deletes the corresponding cookie using setMaxAge(). Finally, the updated cookie is added to the response using addCookie().
02	clearCartNotLogin	Clears the contents of a shopping cart cookie when a user is not logged in. It iterates through each cookie in the request, identifies the shopping cart cookie by its name containing "productCartCooky",
03	doGet	Clears the shopping cart if the user is not logged in, and clears either the entire cart or a specific product's cart if the user is logged in. At the end of the code, it removes the shopping cart attribute from the current session and sets the total price to 0.

CartService

No	Method	Description
01	void clearProductCart(int accountId, int productId)	Clears a specific product from a user's cart by removing the corresponding cart item using the clearCartItemByProduct() method of the cartItemDAO object. It takes two integer parameters: accountId representing the user's account ID, and productId representing the ID of the product to be removed.
02	void clearCart(int accountId)	Clears a user's entire cart by removing all the cart items associated with their account using the clearCartItem()

		method of the cartItemDAO object. It takes one integer parameter: accountId representing the user's account ID.

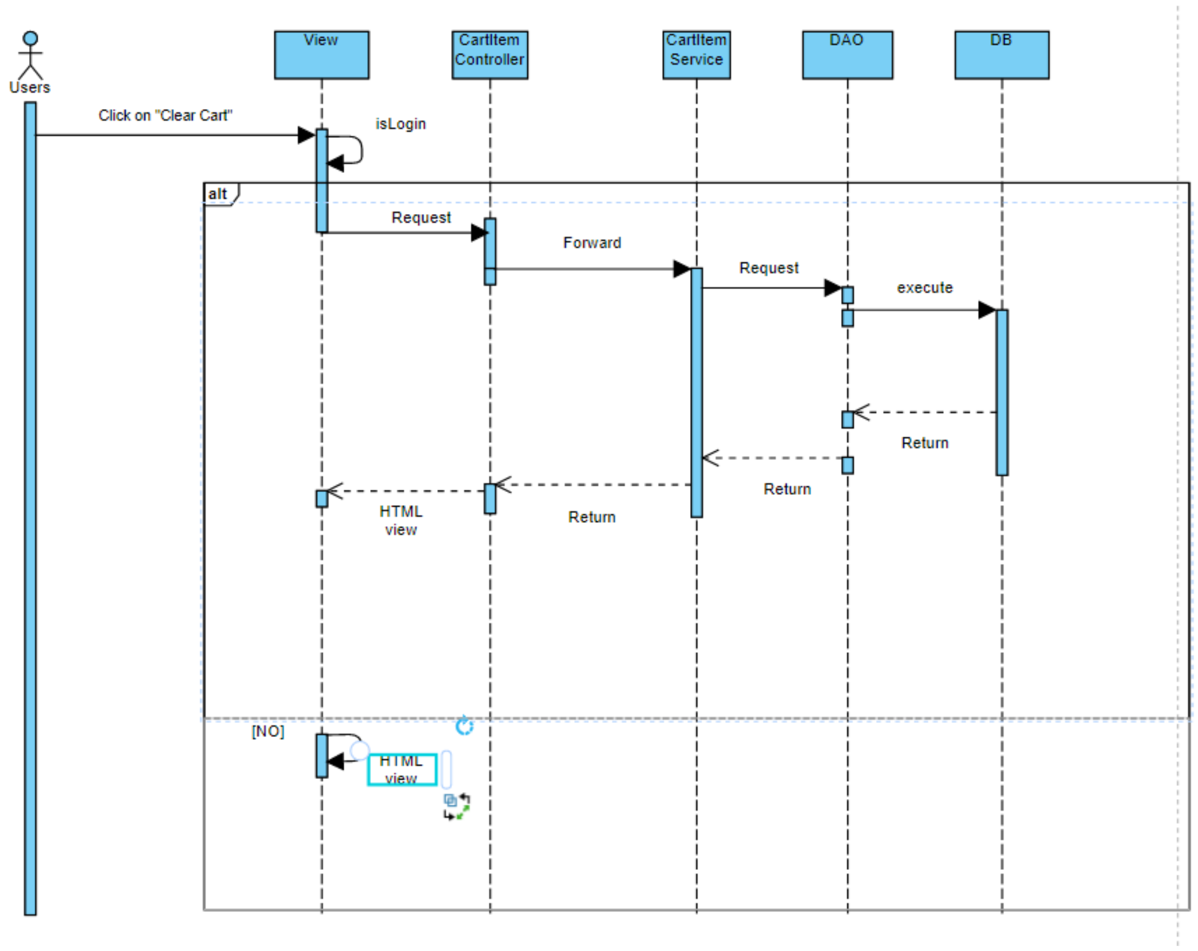
CartDAO

No	Method	Description
01	Cart getCartByAccountId(int accountId)	gets a Cart object from a database based on an account ID. It executes a SQL query to select matching rows, iterates through the result set to map each row to a Cart object, and returns the resulting object.

CartItemDAO

No	Method	Description
01	void clearCartItem(int cartId)	Clears all items from a Cart object in a database based on the cart ID. It executes a SQL query to delete all matching rows, and rolls back changes if any exceptions occur during the transaction. Finally, it resets auto-commit to true and closes the database connection.
02	void clearCartItemByProduct(int cartId, int productId)	Clears a CartItem object from a database based on the cart ID and product ID. It prepares a SQL query to delete matching rows, executes the update, and rolls back changes if any exceptions occur during the transaction. Finally, it resets auto-commit to true and closes the database connection.

c. Sequence Diagram(s)



d. Database Queries

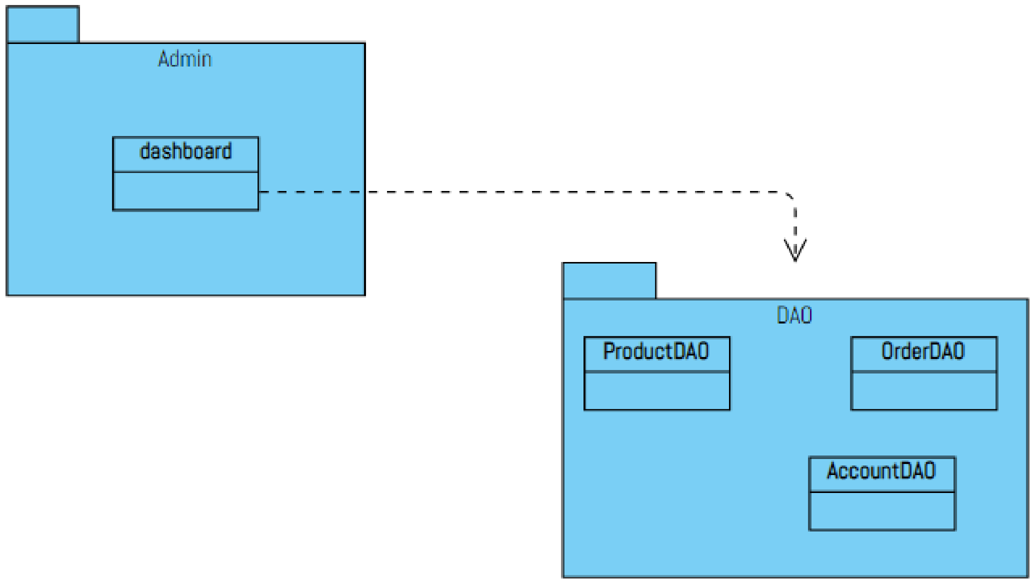
SELECT * FROM shop_pet.cart where account_id = ?

delete from shop_pet.cartitem where cart_id = ?

delete from shop_pet.cartitem where cart_id = ? and product_id = ?

20. Dashboard

a. Class Diagram



b. Class Specifications

DashBoardController

No	Method	Description
01	doGet	Request to a JSP page to show all accounts.

AccountDAO

No	Method	Description
01	int countAccount()	Counting the number of rows in the "account" table in a database.

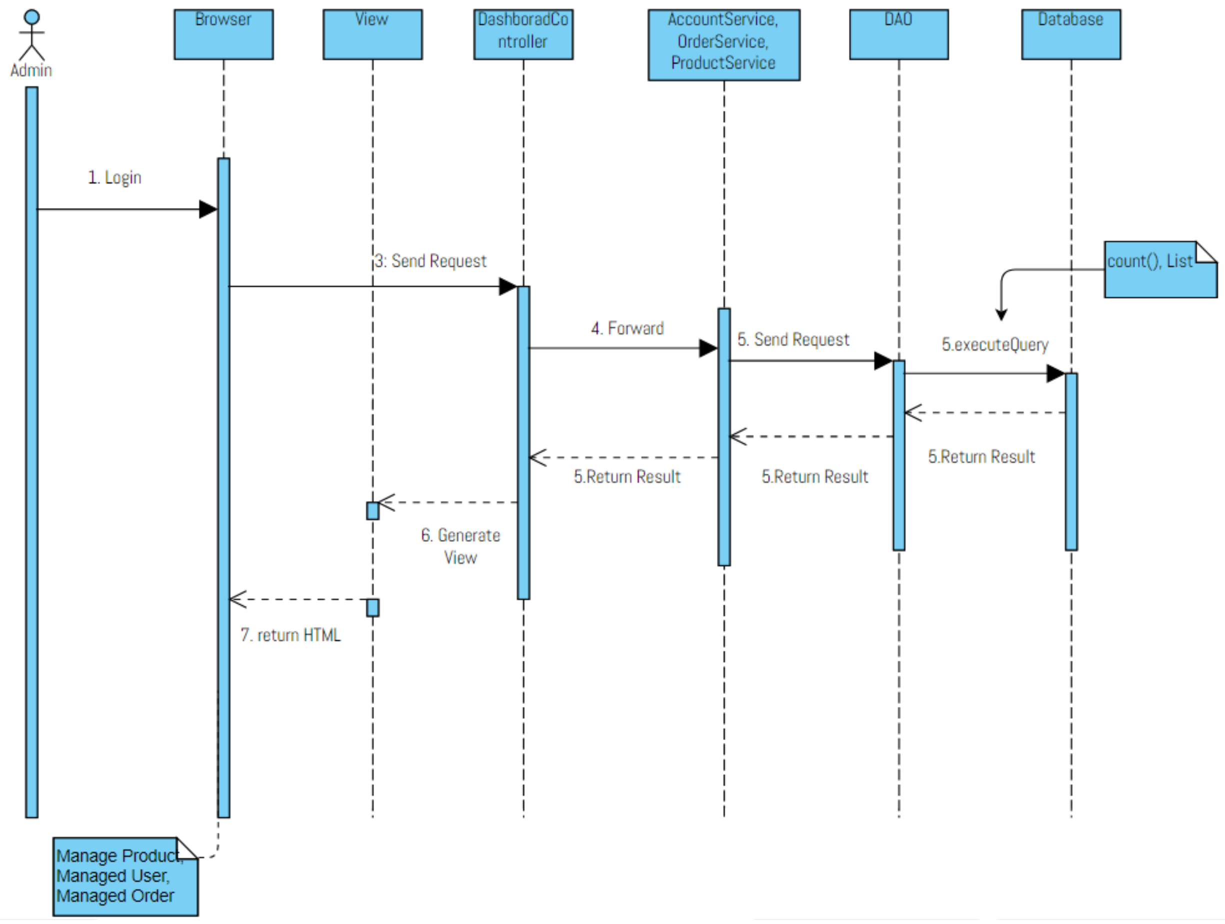
ProductDAO

No	Method	Description
01	int countOrder()	Counting the number of rows in the "order" table in a database.

OrderDAO

No	Method	Description
01	int countProduct()	Counting the number of rows in the "product" table in a database.

c. Sequence Diagram(s)

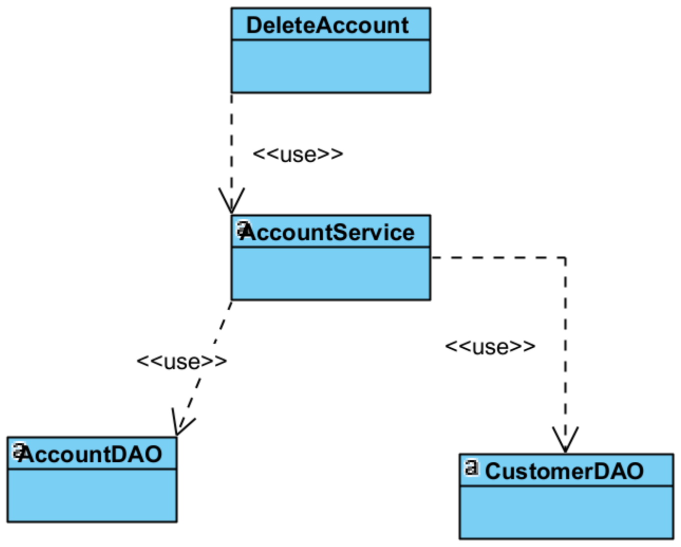


d. Database Queries

- *select count(*) as count from account*
- *select count(*) from product*
- *select count(*) from shop_pet.order*

21. Deactive/Active account

a. Class Diagram



b. Class Specifications

AccountService

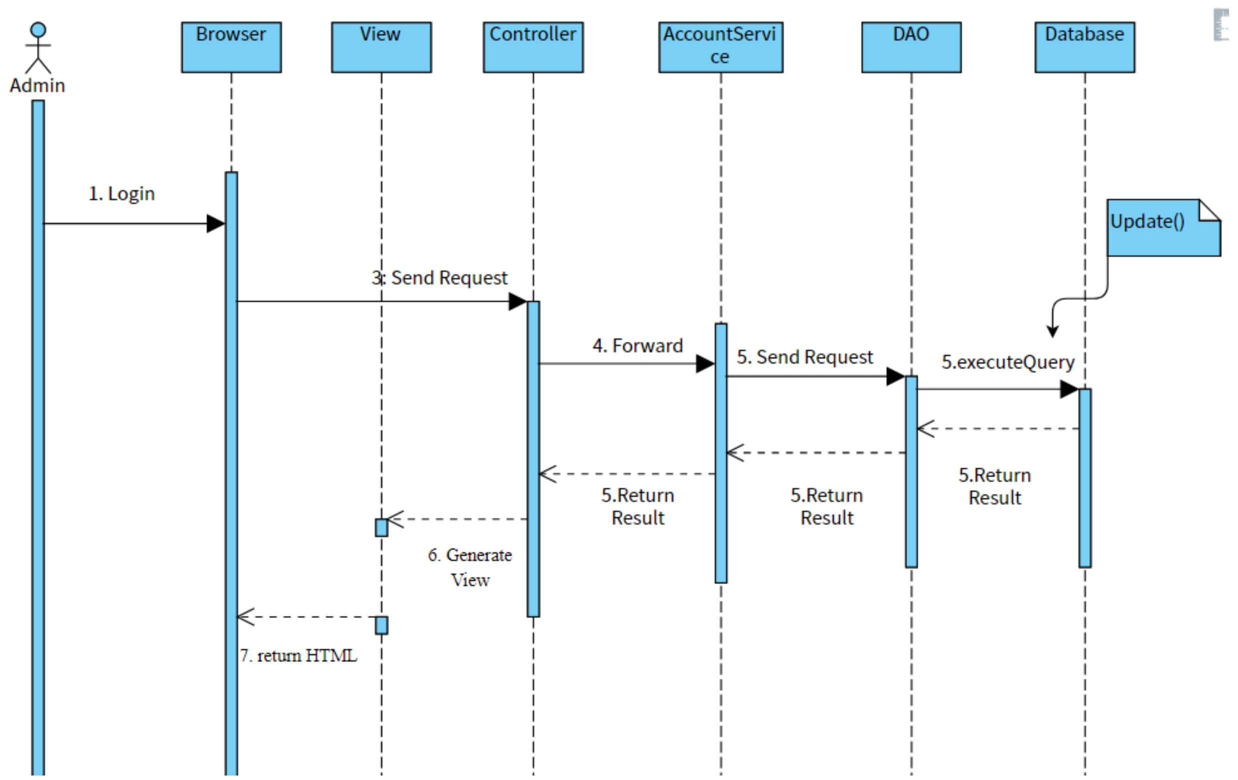
No	Method	Description
01	Update(Account account)	Method to update Account with Input being an Account found from request, it will check the account exist or not and the account must be a Customers' Account , if it satisfies the condition, the account will be updated
02	GetByID(Int id)	Method to find Account and fill them with input id from request service , method return a Account if they are found and be Customers' Account, otherwise return null
03	update(account)	Method to update Account with Input being an Account from AccountService.Update(Account account)

AccountDAO

No	Method	Description
01	getAccountByID(Int id)	Method to find Account with input id from AccountService.getByID(int id), method return a Account if they are found otherwise return null
02	checkExistUsername(account.getUsername())	Method to check the account exists or not with Input being an account.getUsername() from AccountService.Update(Account account), Method return True if the account exists otherwise return false

03	getAllCustomers()	Method to get List All Customer in the system, method will called by AccountService.getByID(int id) and GetByID(Int id) to filter the account

c . Sequence Diagram(s)



d. Database Queries

- SELECT * FROM Account WHERE id = ?
- SELECT COUNT(*) FROM Account WHERE username = ?
- SELECT * FROM shop_pet.`Account`