COMP246_008

Group 4 Project

Pet Food Info System

Alexis Potter (300805710)

Jaeuk Kim (301145308)

Ngoc Phuong Uyen Ho (301103427)

Content

Section 1: Problem Statement

1.1 Problem & Need

Pet owners have no way to know what pet food is good for their pet and what is not. Many pet food brands are recalled by the FDA and most owners will not know unless they are actively searching for food recalls. When an individual walks into a pet store they are often overwhelmed by the number of choices they could make with regard not only to brands, but to products within each brand. Having one place to review recall information, check pet food nutritional information, and see real customer reviews of the products would be invaluable for most pet owners.

Capabilities

- Scan pet food barcodes
- Search pet food brands and products
- Display information regarding pet food nutritional information
- Display recall information about specific product
    - Additional recall information about the specific brand, including the recall of any other products produced by that brand
- Display customer reviews of specific product
- Display alternative food options based on customer reviews + nutritional information
- Display top food products and brands based on customer reviews
- Set an alert for specific product or brand recalls

Benefits

- Pet owners have a single place to get all the necessary information they need regarding purchasing the best food product for their pet
- New pet owners can be confident they are purchasing a safe and healthy option for their pet
- Pet stores can use the software to check if a product they sell is on the FDA recall list, and proactively remove the product from their shelves
- Customers will not have to actively check whether their pet food has been recalled, the software will alert customer to any recalls of the specific product they are using, and the specific brand they are using
- Customers will be able to read the nutritional information about a product, with the ability to make the print bigger
    - Many pet food bags are heavy and awkward, so viewing the nutritional information can be difficult
- Customers can compare similar pet foods to pick the one that best suits their pets needs

1.2 Stakeholders

- Pet owners
- Pet store (Owners, employees, marketing)
- Pet food manufacturers (Marketing, CEO, chefs)
- Managers
- Developers

1.3 Identify the sub-systems of your application (What are its functional components)

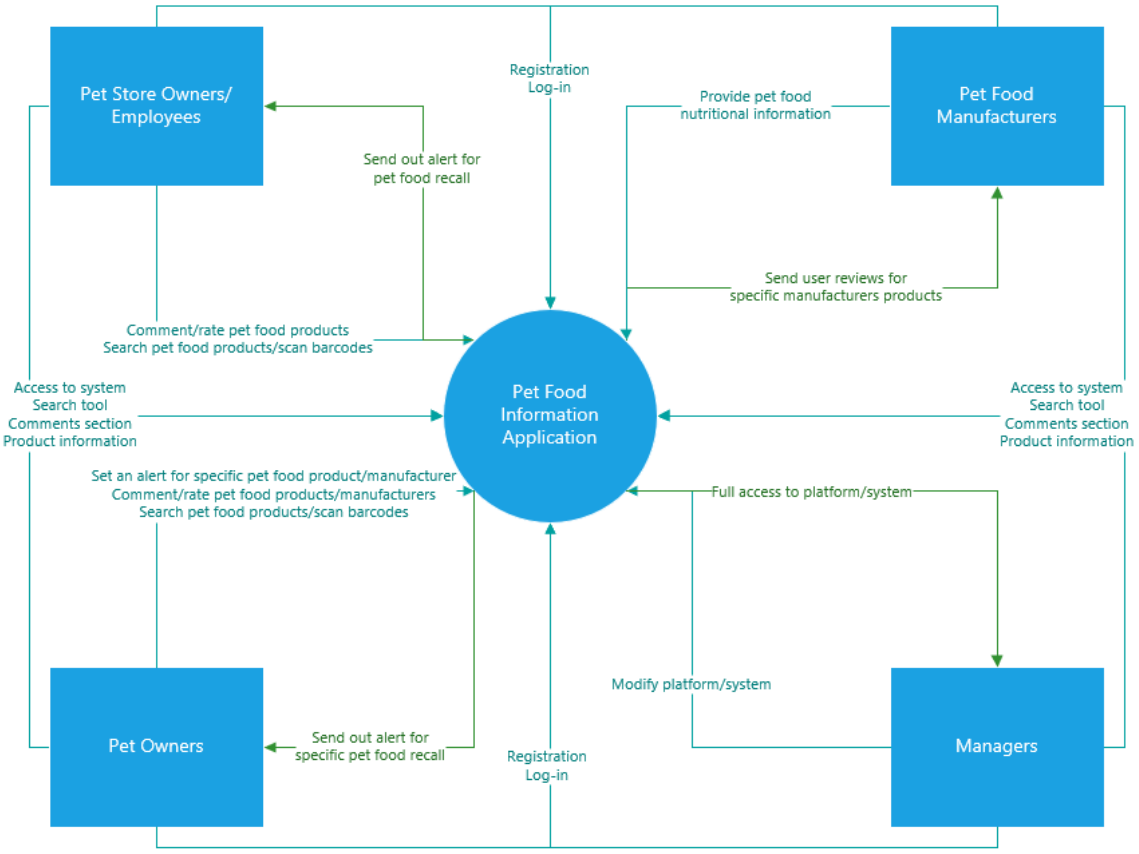
- Login subsystem + Registration subsystem
- Search subsystem
- Rating subsystem / Review(Comment) subsystem

1.4 Who are the intended users of the SRS documentation?

- Developers
- Managers
- Stakeholders

Section 2: A Context Flow – Structured Modeling

2.1 Context Flow Diagram

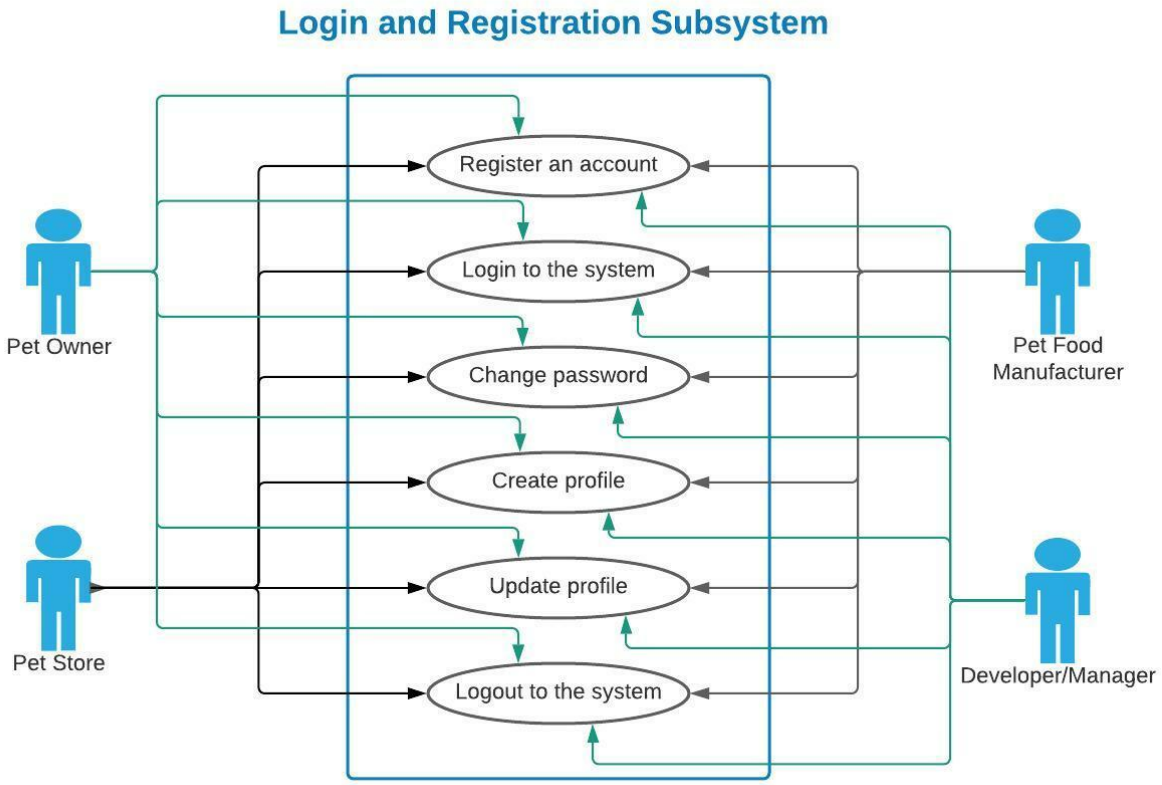


Section 3: Functional Requirements

3.1.1 Login and Registration Subsystem

FR#	Name (Goal Use case)	Role Player	Description
FR01	Register an account	All users	The users must create an account to login into the application.
FR02	Login to the system	All users	To log in to the system to type the user's username and password.
FR03	Change password	All users	The user can change the password in case the user forgot the password.
FR04	Create profile	All users	The user can create a new profile in the system.
FR05	Update profile	All users	The user can view and update a personal profile such as personal information, profile pictures.
FR06	Logout to the system	All users	The user can log out of the application anytime.

Use Case Diagram



User Story

1. As a user, I want to create an account and log in, so I can access the system.

   Acceptance Criteria:
   - Should be able to create a profile.
   - Should be able to select username and password.
   - Should be able to verify if the user has already existed
   - Should be able to see the last login of the user.
2. As a user, I want to change the password, so I can access the system if I forgot the password.
   Acceptance Criteria:
   - Should be able to change the user's password.
   - Should be able to give the one-time password to the user to change the new password.

3. As a user, I want to access my personal information, so I can update my personal profile.

   Acceptance Criteria:

- Should be able to update the user's profile pictures and personal information such as address and phone number.

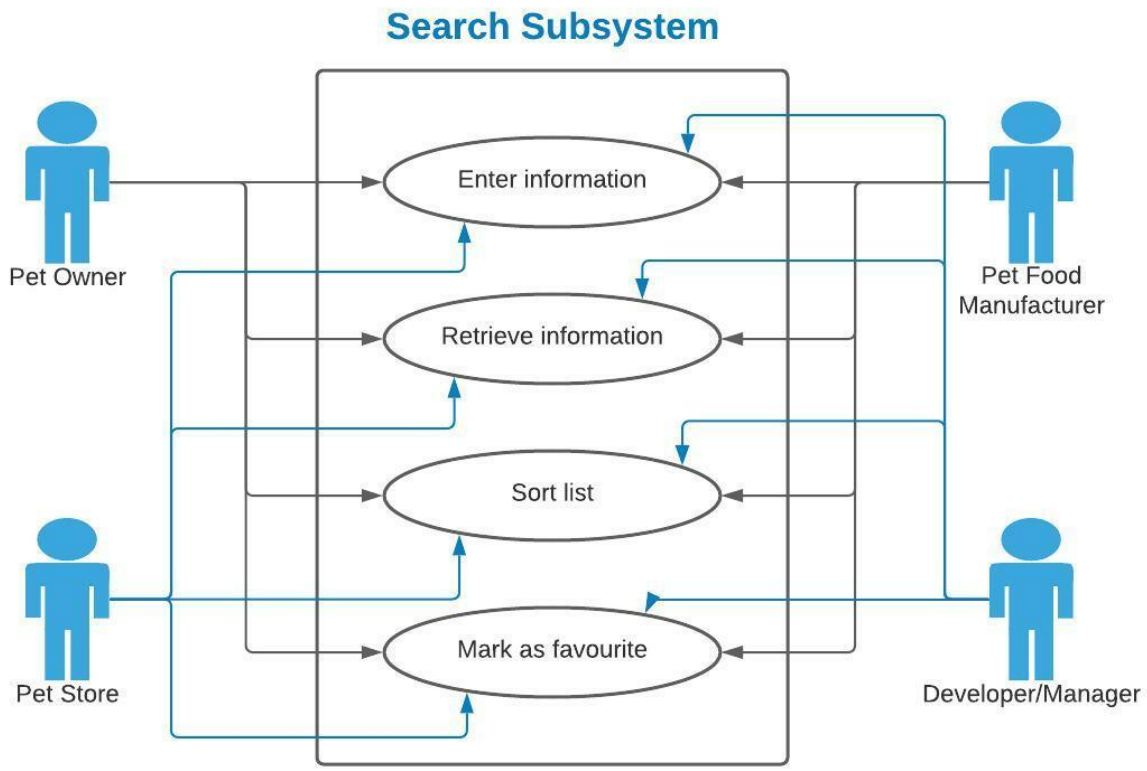
4. As a user, I want to be able to log out of the system, so I can log out of the system whenever I need.

   Acceptance Criteria:
   - Should be able to log out of the system after finishing the access.

3.1.2 Search Subsystem

FR#	Name (Goal Use case)	Role Player	Description
FR07	Enter information	All users	The user will enter the type of food, brand, rating, and price range.
FR08	Retrieve information	All users	The system displays a list of the available information in its database that matches the information provided by the user.
FR09	Sort list	All users	Allow the user to sort the list according to price range, rating, and brand.
FR10	Mark as favourite	All user	Allow the user to mark their favourite brands, and this data will be stored in their account.

Use Case Diagram



User Story

1. As a user, I want the best search tool, so I can insert specific details while searching.

   Acceptance Criteria:
   - Should be able to enter the type of food.
   - Should be able to enter brand.
   - Should be able to enter rating.
   - Should be able to enter price range.
2. As a user, I want a list of results, so I can find information to satisfy my needs.

   Acceptance Criteria:
   - Should be able to connect to the database.
   - Should be able to choose information based on the database to get the matching results.
   - Should be able to display results in the list format.
3. As a user, I want to sort the list, so I can filter my items more effectively.

Acceptance Criteria:

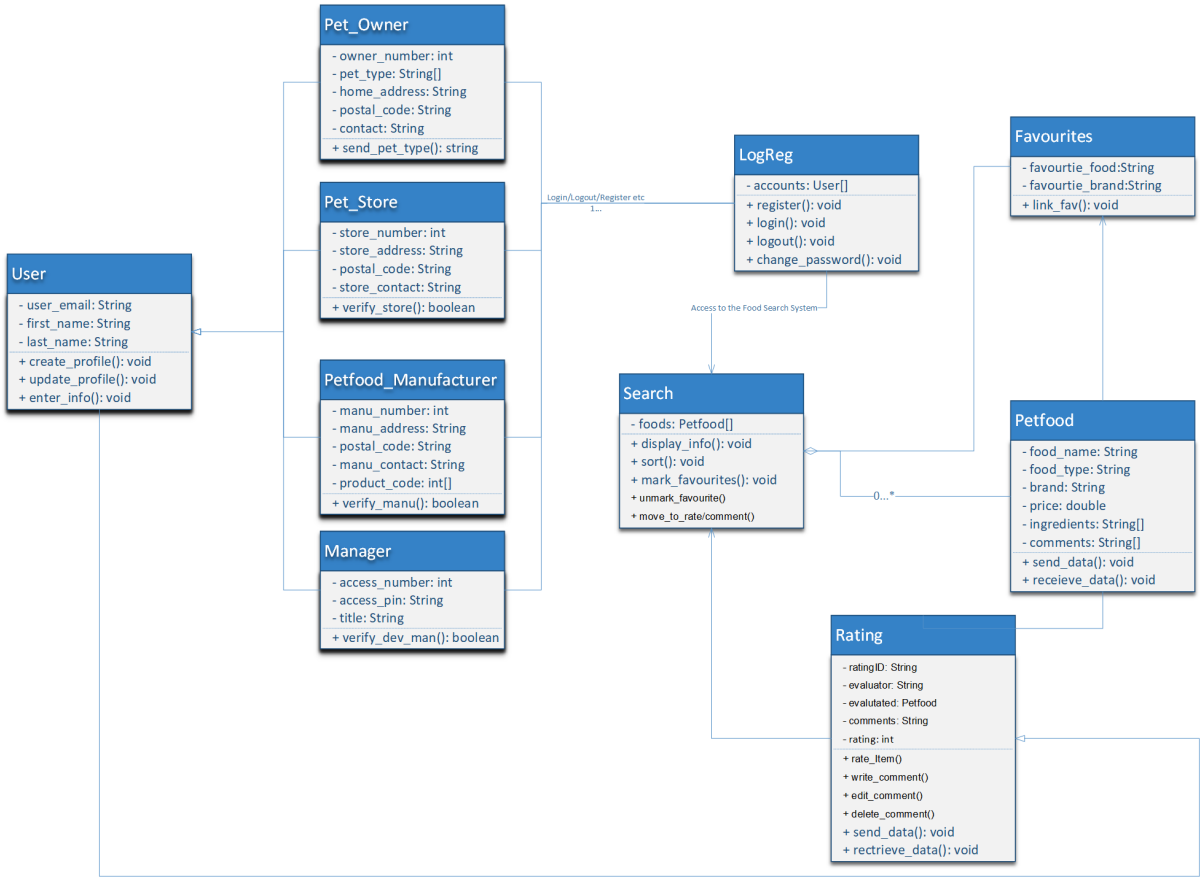
- Should be able to sort by brand.
- Should be able to sort by price range.
- Should be able to sort by rating.

4. As a user, I want to have a favourite list, so I will readily have the results I liked the most.

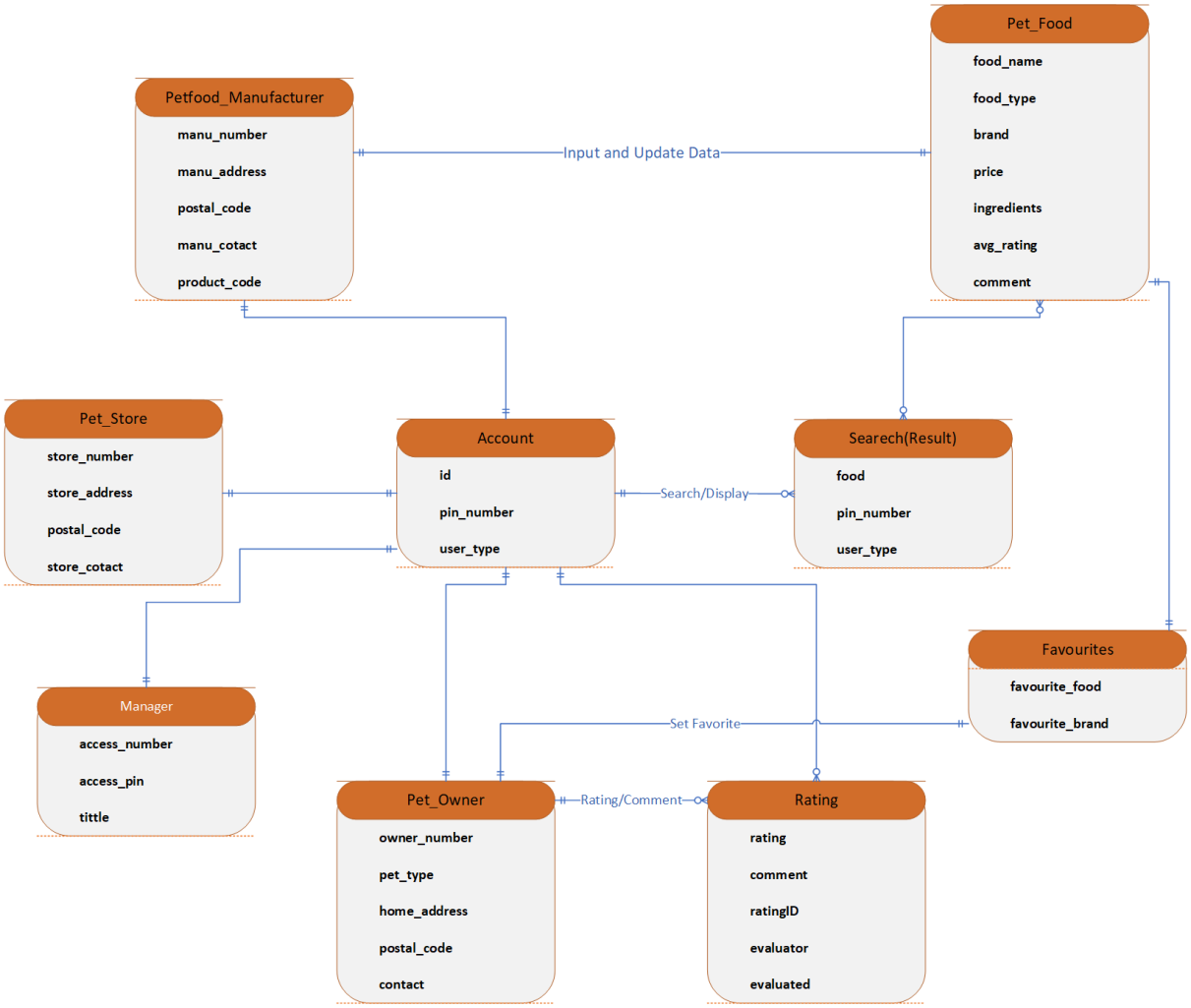
   Acceptance Criteria:

   - Should be able to mark favourite brand.

Section 4: UML Domain Class Diagram

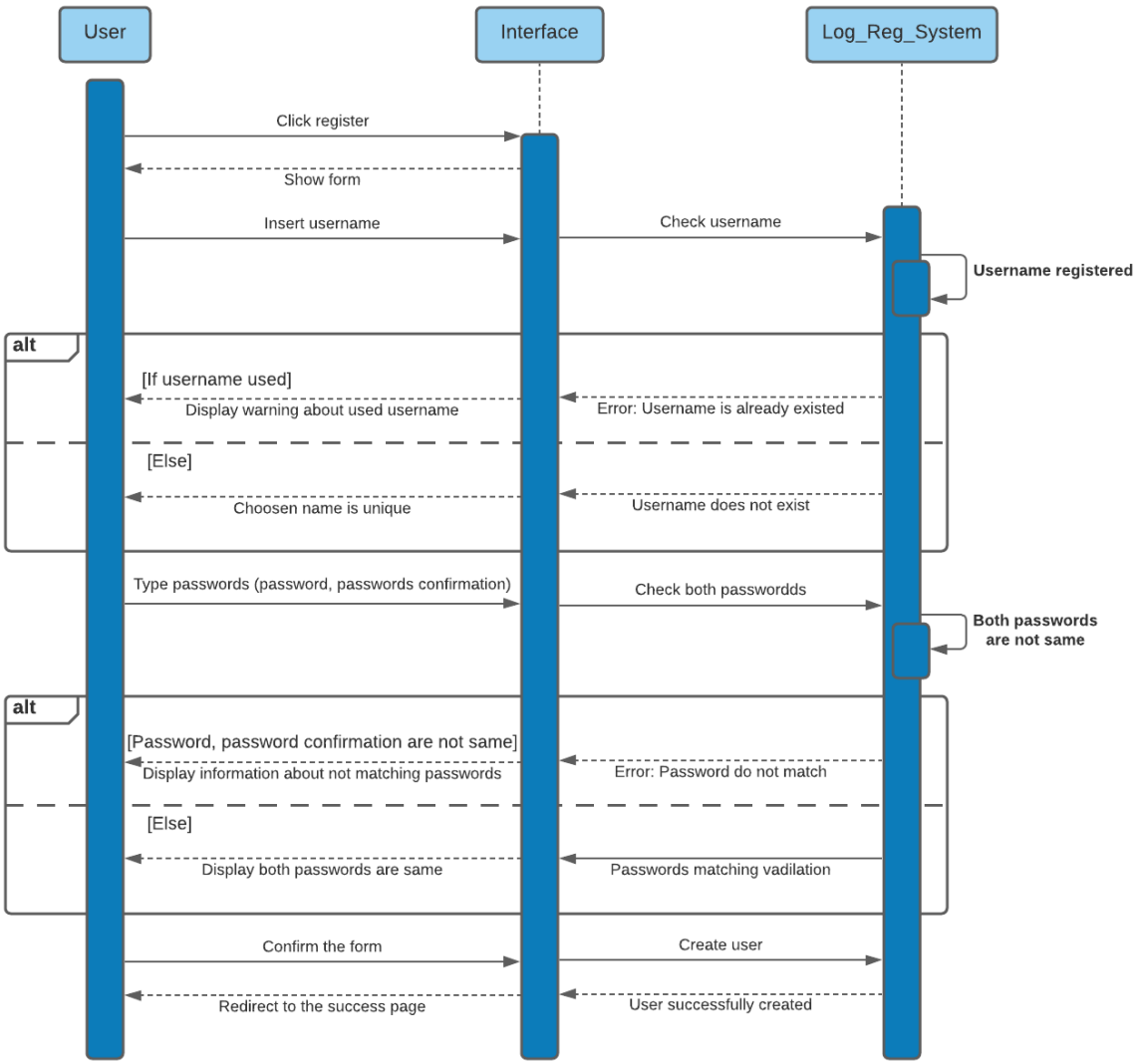


Section 5: ERD Model

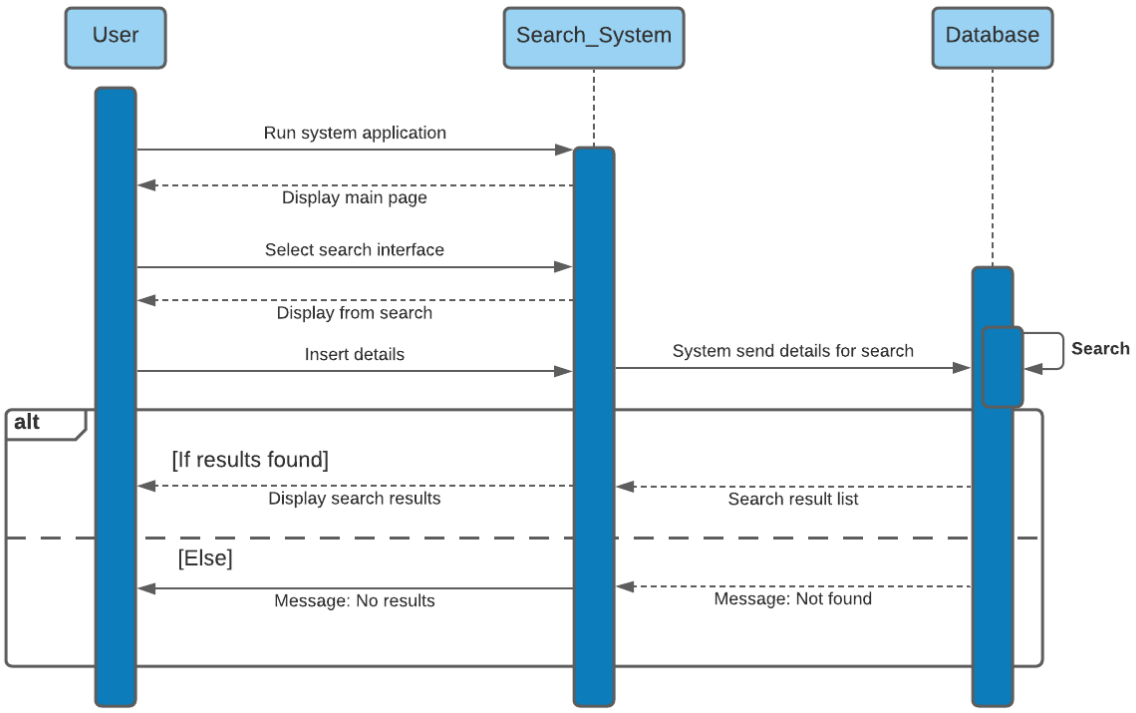


Section 6: Sequence Diagrams

6.1 Registration (use case: Registater to the system)

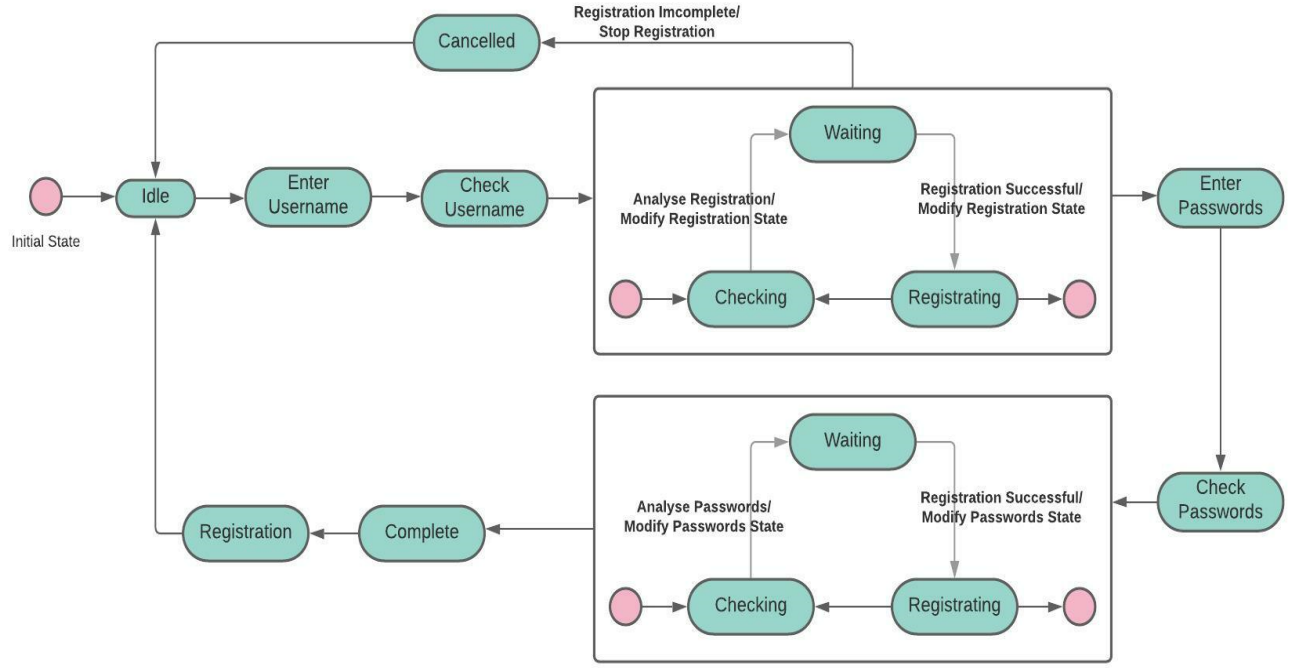


6.2 Search (Use cases: enter information and retrieve information)

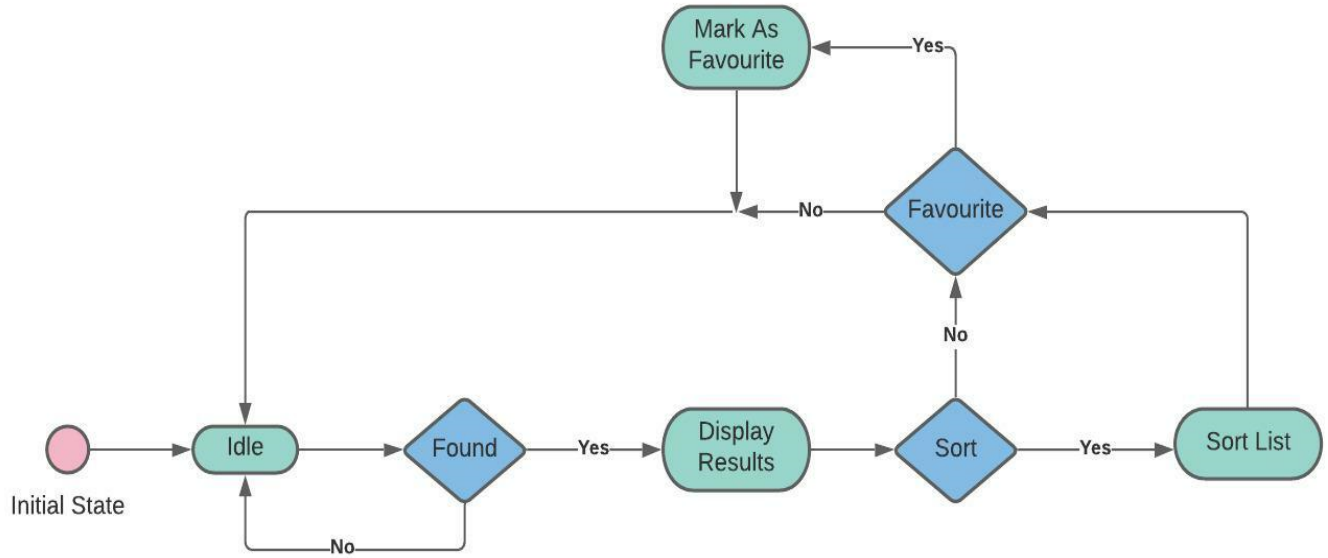


Section 7: State Diagrams

7.1 Registration (use case: Registater to the system)



7.2 Search system (Use cases: enter information and retrieve information)



Section 8: Technologies

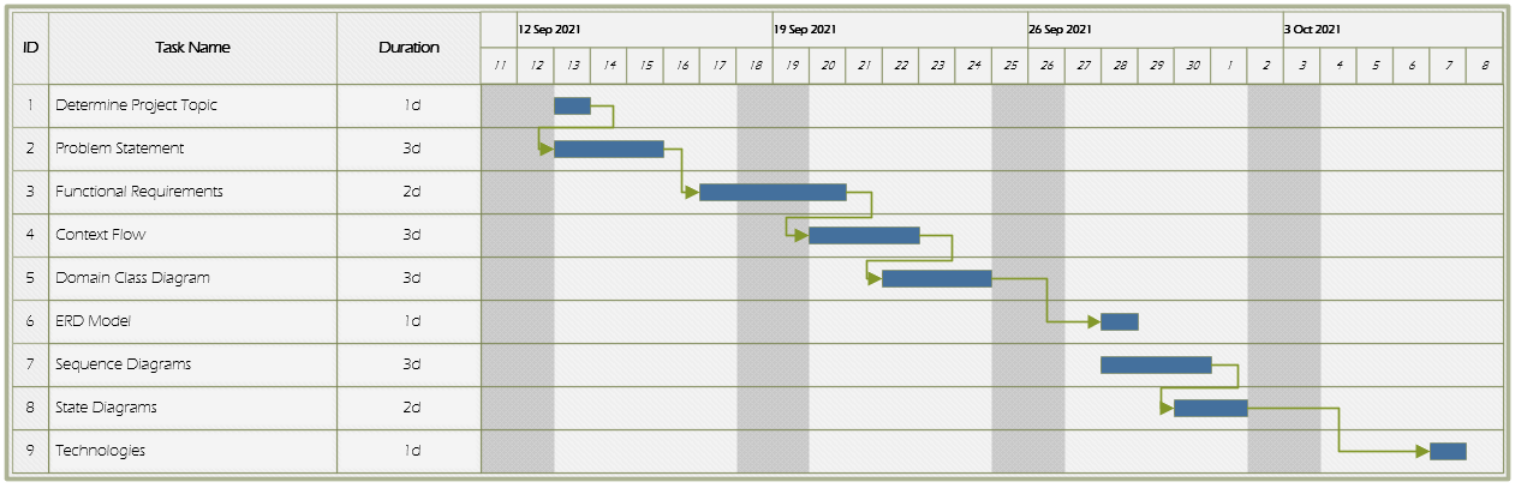
8.1 The development of application: Mobile

8.2 Front-end – GUI: Android Studio (Android system), Xcode (iOS system)

8.3 Middle layer – Class methods: HttpHandlers and HttpModules

8.4 Database: ASP.NET Core

Section 9: Gantt Chart



Part B: Software Design Architecture

Section 1: Requirements Edits to Part A

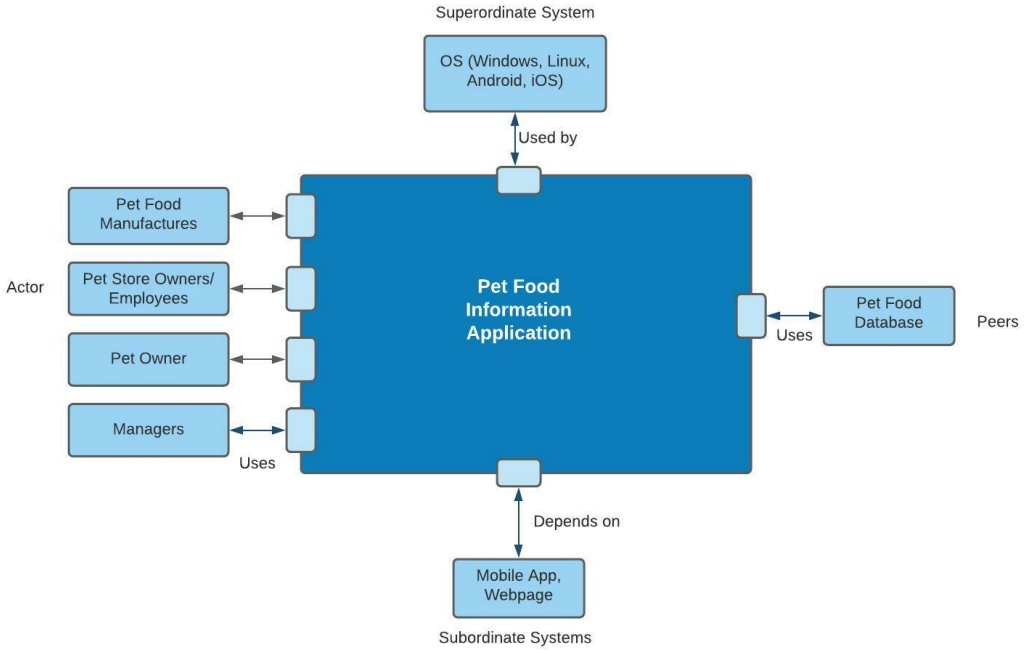
Section 2: Overview Model

2.1 Intended users of the SDD document

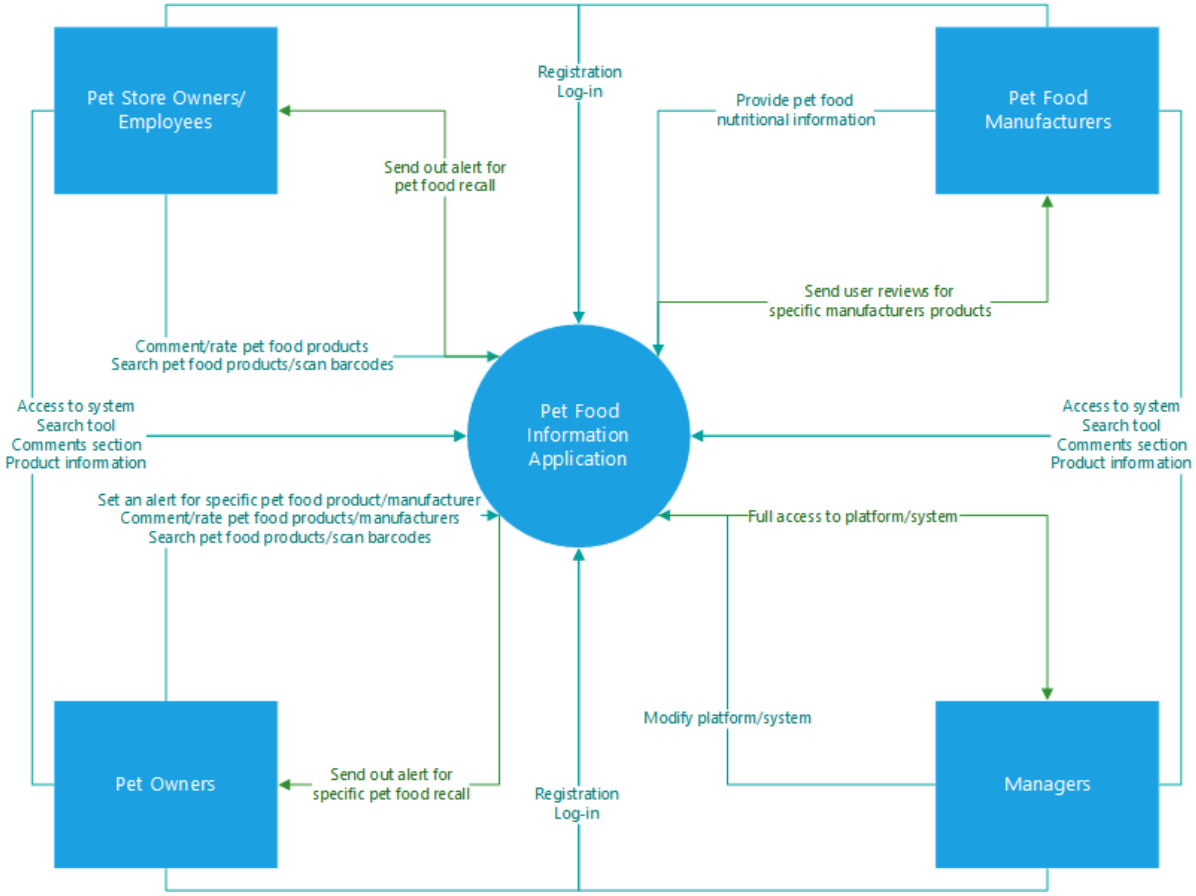
The purpose of the System Design Document (SDD) is to provide details of the description of the system design, so software developers will ultimately know what is necessary to build. In addition, some stakeholders including pet food manufacturers and pet food shops can use this document to understand the system well, which can lead to critical feedback. Therefore, the developers and managers responsible for the system development, and stakeholders use this document.

2.2 Architectural Context diagram (ACD) versus Context Flow Diagram (CFD)

2.2.1 Architectural Context diagram (HOW overview)



2.2.2 Context Flow diagram (WHAT overview)

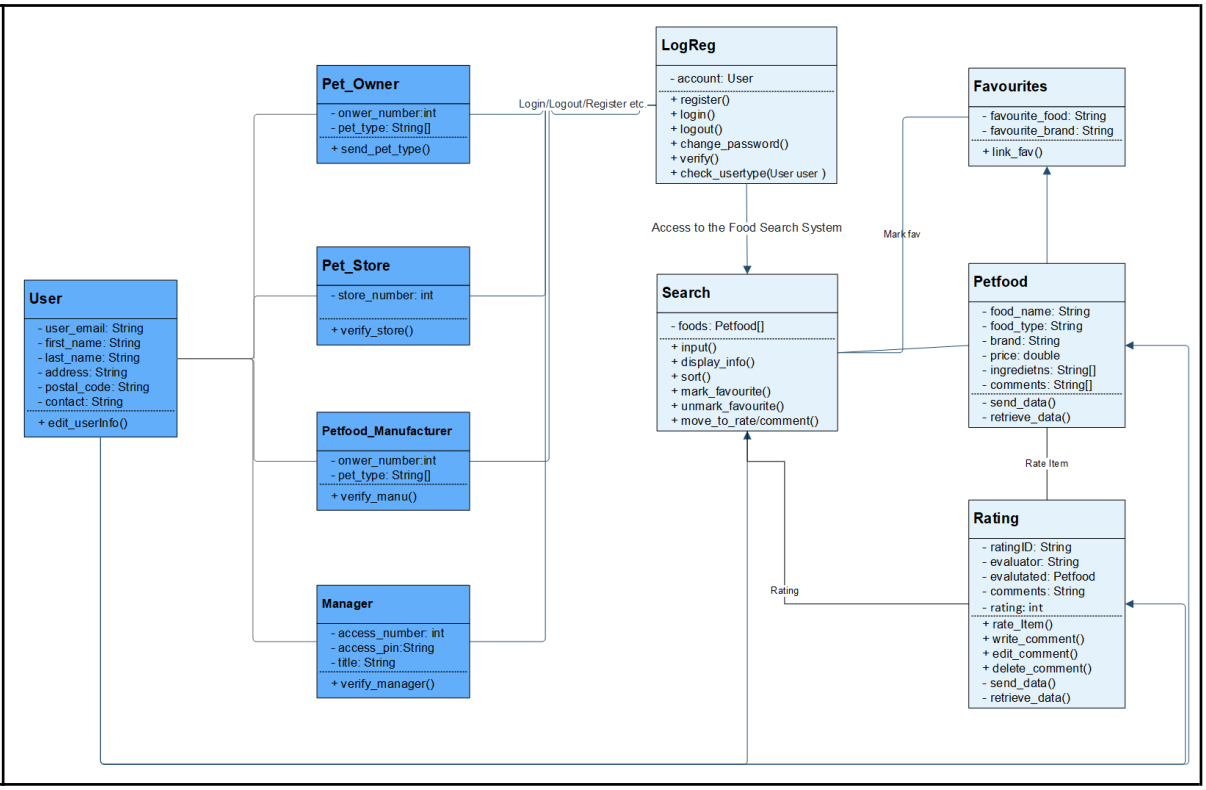


Section 3: Modularization

3.1 Partition the analysis model

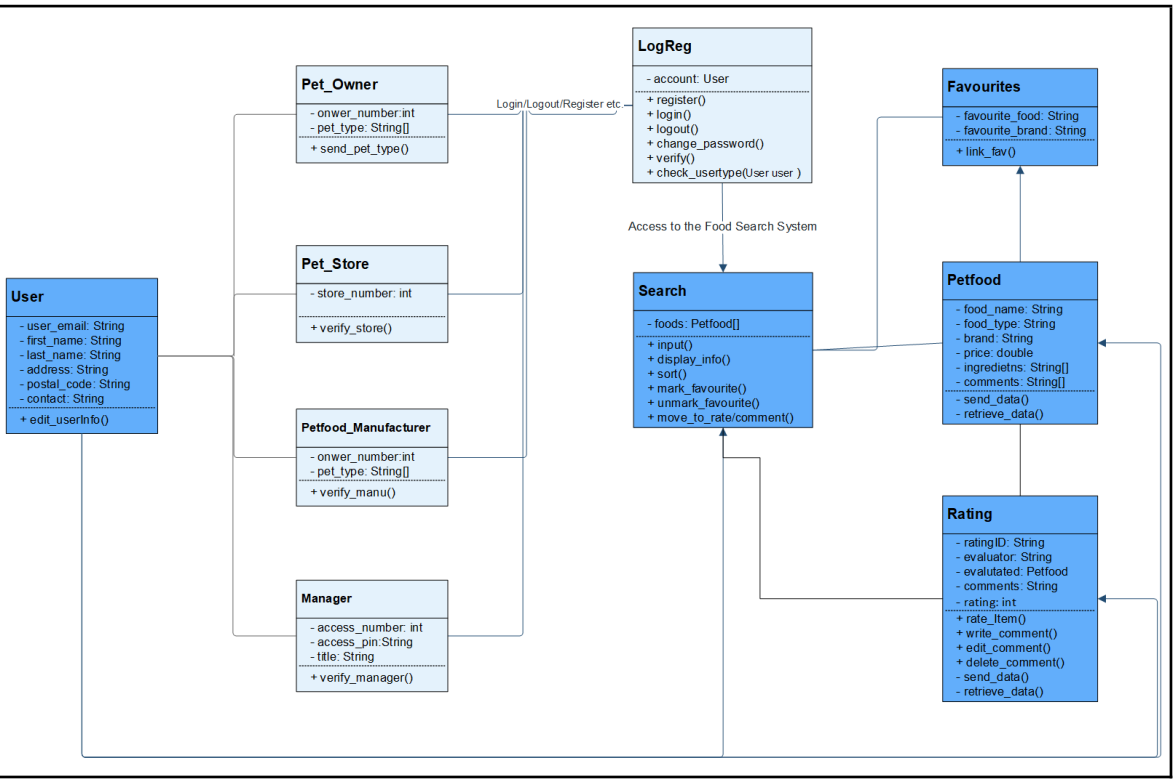
3.1.1 Login Subsystem

- User
- Pet Owner
- Pet Store
- Pet Food Manufacturer
- Manager
- Login and Register



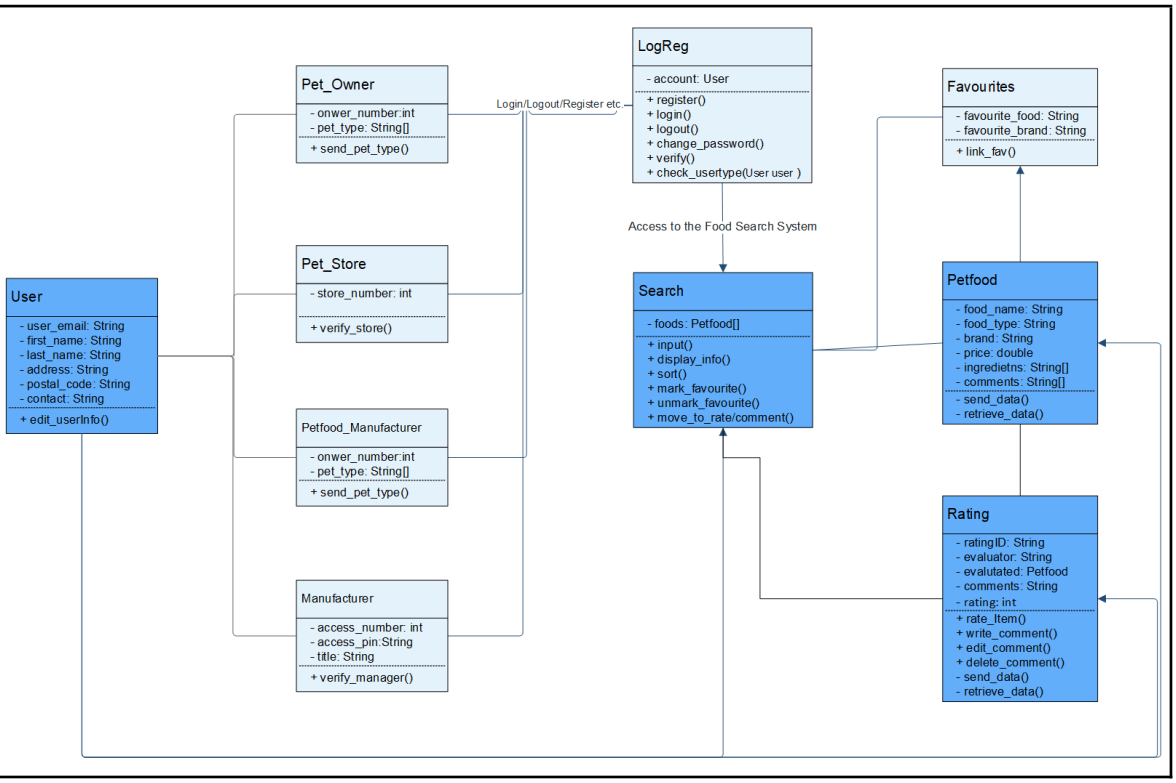
3.1.2 Search Subsystem

- User
- Search
- Pet food
- Favourites
- Rating



3.1.3 Rating Subsystem

- Rating
- User
- Search
- Petfood



3.2 Class Responsibility Collaboration (CRC)

3.2.1 Login Subsystem

User	
Super Classes:	
Sub Classes: Pet_Owner, Pet_Store, Petfood_Manufacturer, (System)Manager	
Description: Identifies users and stores user information	
Attributes:	
Name	**Description**
user_email	User's email address
first_name	User's first name
last_name	User's last name
home_address	Owner's address
postal_code	Owner's postal code
contact	User's phone number
Responsibilities:	
Name	**Collaborator**
edit_userinfo()	LogReg

LogReg	
Super Classes:	
Sub Classes:	
Description: Confirms user log-in information, registers new accounts	
Attributes:	
Name	**Description**
account	User obejct
Responsibilities:	
Name	**Collaborator**
register()	User / the subclasses
log-in/log-out	
change_password()	
verify()	
check_usertype()	

Pet_Owner	
Super Classes: User	
Sub Classes:	
Description: Stores information about the pet owner	
Attributes:	
Name	**Description**
owner_number	Unique ID number
pet_type	Types of pets the user owns
Responsibilities:	
Name	**Collaborator**
+ send_pet_type()	LogReg

Pet_Store	
Super Classes: User	
Sub Classes:	
Description: Stores information about the pet store	
Attributes:	
Name	**Description**
store_number	Unique store ID
Responsibilities:	
Name	**Collaborator**
verify_store()	LogReg

Manager	
Super Classes: User	
Sub Classes:	
Description: Stores information about the service manager	
Attributes:	
Name	**Description**
access_number	Unique access ID
access_pin	Unique access PIN code
title	Manager's title
Responsibilities:	
Name	**Collaborator**
verify_manager()	LogReg

Petfood_Manufacturer	
Super Classes: User	
Sub Classes:	
Description: Stores information about the manufacturer	
Attributes:	
Name	**Description**
manu_number	Unique manufacturer ID number
product_code	Manufacturer's product barcodes
Responsibilities:	
Name	**Collaborator**
verify_manu()	LogReg

3.2.2 Search Subsystem

User	
Super Classes:	
Sub Classes: Pet_Owner, Pet_Store, Petfood_Manufacturer, (System)Manager	
Description: Identifies users and stores user information	
Attributes:	
Name	**Description**
user_email	User's email address
first_name	User's first name
last_name	User's last name
home_address	Owner's address
postal_code	Owner's postal code
contact	User's phone number
Responsibilities:	
Name	**Collaborator**
edit_userinfo()	LogReg

Favourites	
Super Classes:	
Sub Classes:	
Description: Provides favourites for specific Users	
Attributes:	
Name	**Description**
favourite_food	Favourited Petfood
favourite_brand	Favourited Manufacturer
Responsibilities:	
Name	**Collaborator**
link_fav()	Petfood

Petfood	
Super Classes:	
Sub Classes:	
Description: Provides information about specific Petfood, interacts with ratings and favourites	
Attributes:	
Name	**Description**
food_name	Name of petfood
food_type	Type of petfood
brand	Petfood brand
price	Price of Petfood
ingredients	Array of petfood ingredients
comments	Array of User comments
Responsibilities:	
Name	**Collaborator**
send_data()	Rating
retrieve_data()	Rating

Rating	
Super Classes:	
Sub Classes:	
Description: Provides Petfood ratings to Users	
Attributes:	
Name	**Description**
ratingID	Unique rating ID
evaluator	The user who rates the product
evaluated	The product that is rated
rating	User rating
comments	User comments
Responsibilities:	
Name	**Collaborator**
rate_Item()	User, Petfood
write_comment()	User, Petfood
edit_comment()	User, Petfood
delete_comment()	User, Petfood
send_data()	Petfood
retrieve_data()	Petfood

Search	
Super Classes:	
Sub Classes:	
Description: Where users can enter search data and retrieve petfood information stored in the system	
Attributes:	
Name	**Description**
foods	Array of Petfood products
Responsibilities:	
Name	**Collaborator**
input()	
display_info()	Petfood
sort()	Petfood
mark_favourite()	Favourites
unmark_favourite()	Favourites
move_to_rate/comment()	Rating

3.2.3 Rating

User	
Super Classes:	
Sub Classes: Pet_Owner, Pet_Store, Petfood_Manufacturer, (System)Manager	
Description: Identifies users and stores user information	
Attributes:	
Name	**Description**
user_email	User's email address
first_name	User's first name
last_name	User's last name
home_address	Owner's address
postal_code	Owner's postal code
contact	User's phone number
Responsibilities:	
Name	**Collaborator**
edit_userinfo()	LogReg

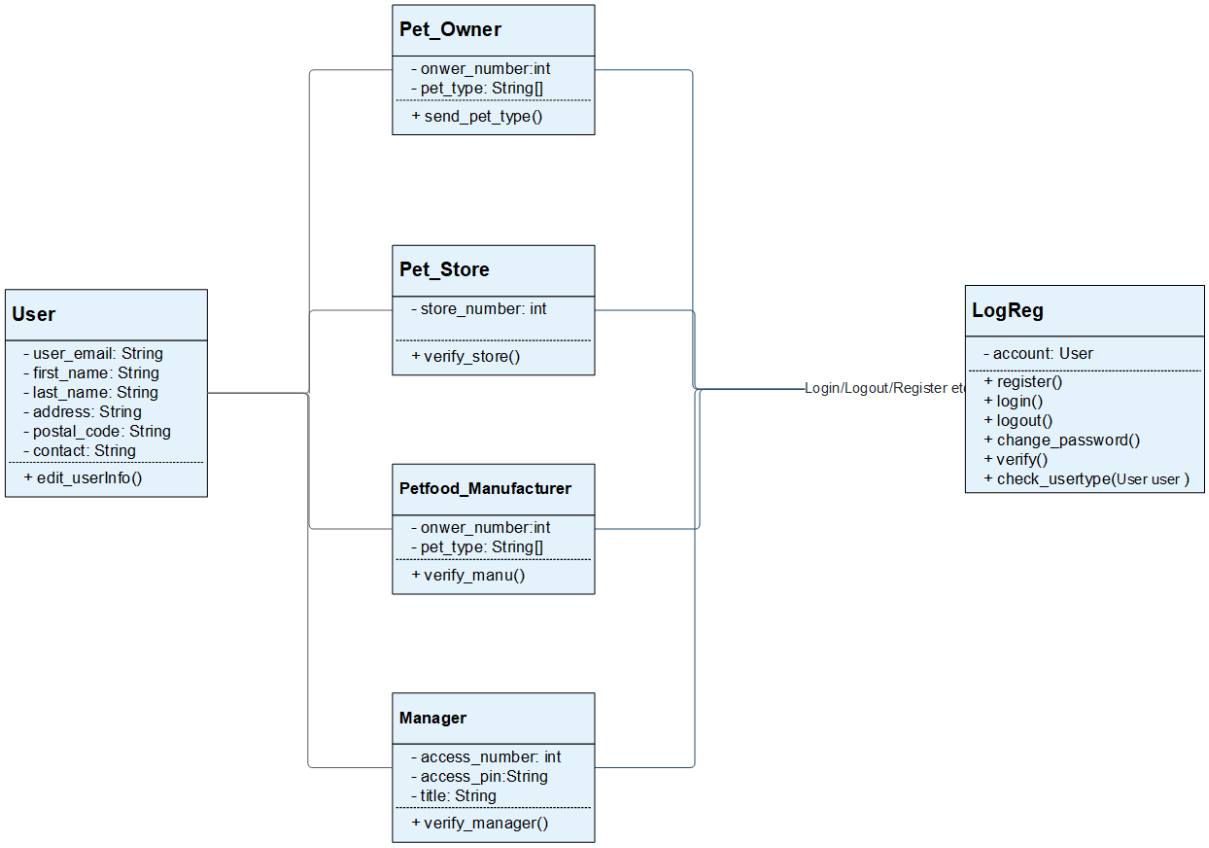
Rating	
Super Classes:	
Sub Classes:	
Description: Provides Petfood ratings to Users	
Attributes:	
Name	**Description**
ratingID	Unique rating ID
evaluator	The user who rates the product
evaluated	The product that is rated
rating	User rating
comments	User comments
Responsibilities:	
Name	**Collaborator**
rate_Item()	User, Petfood
write_comment()	User, Petfood
edit_comment()	User, Petfood
delete_comment()	User, Petfood
send_data()	Petfood
retrieve_data()	Petfood

Petfood	
Super Classes:	
Sub Classes:	
Description: Provides information about specific Petfood, interacts with ratings and favourites	
Attributes:	
Name	**Description**
food_name	Name of petfood
food_type	Type of petfood
brand	Petfood brand
price	Price of Petfood
ingredients	Array of petfood ingredients
comments	Array of User comments
Responsibilities:	
Name	**Collaborator**
send_data()	Rating
retrieve_data()	Rating

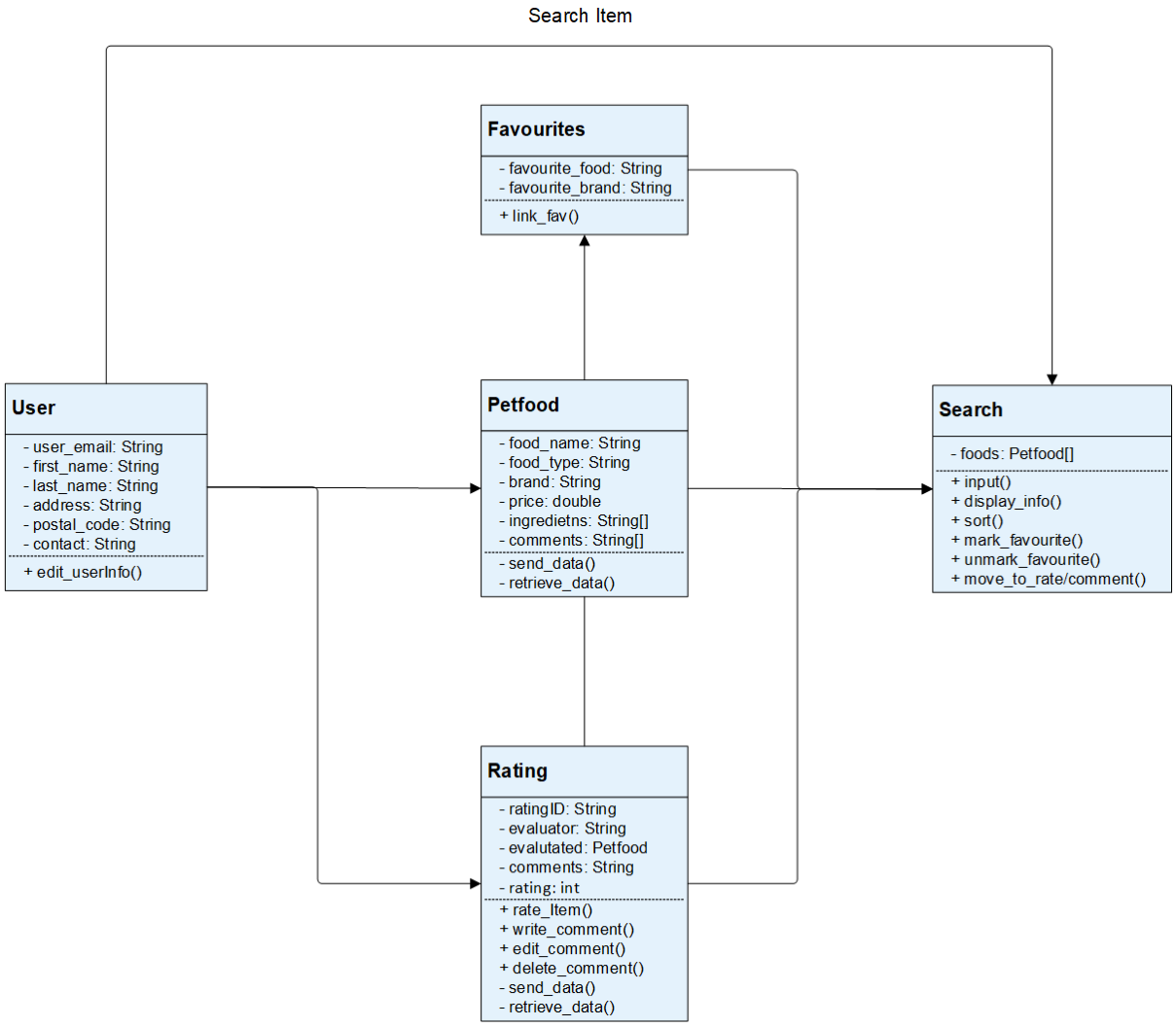
Search	
Super Classes:	
Sub Classes:	
Description: Where users can enter search data and retrieve petfood information stored in the system	
Attributes:	
Name	**Description**
foods	Array of Petfood products
Responsibilities:	
Name	**Collaborator**
input()	
display_info()	Petfood
sort()	Petfood
mark_favourite()	Favourites
unmark_favourite()	Favourites
move_to_rate/comment()	Rating

3.3 Design classes diagram

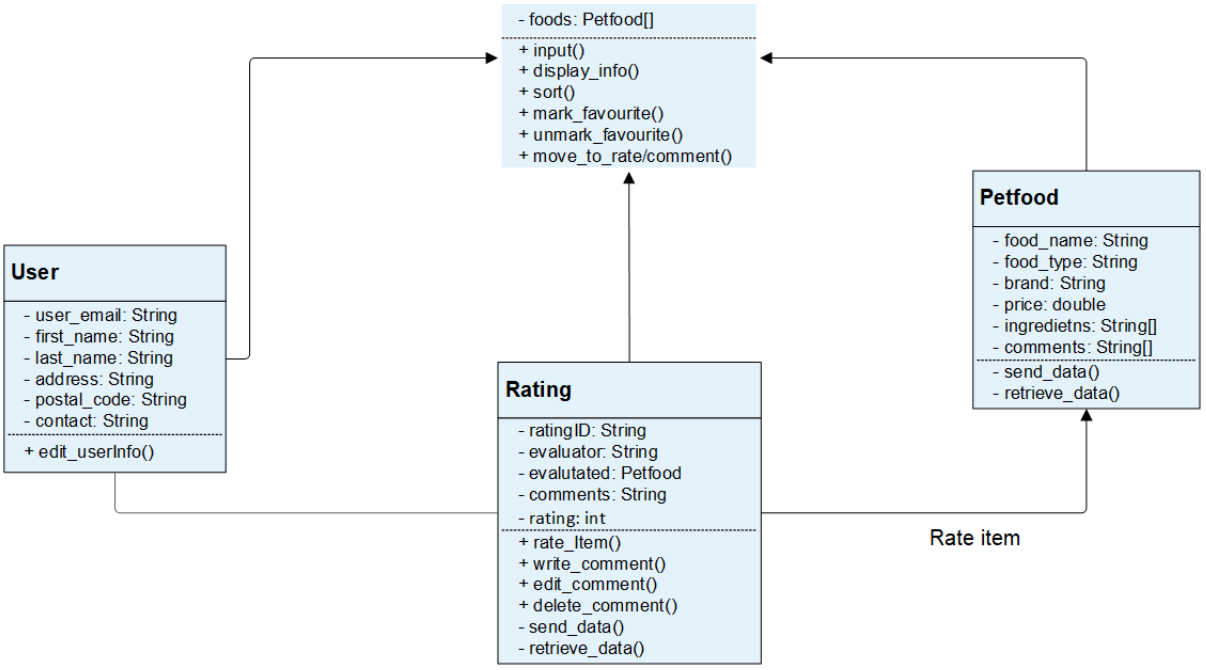
3.3.1 Login Subsystem



3.3.2 Search Subsystem



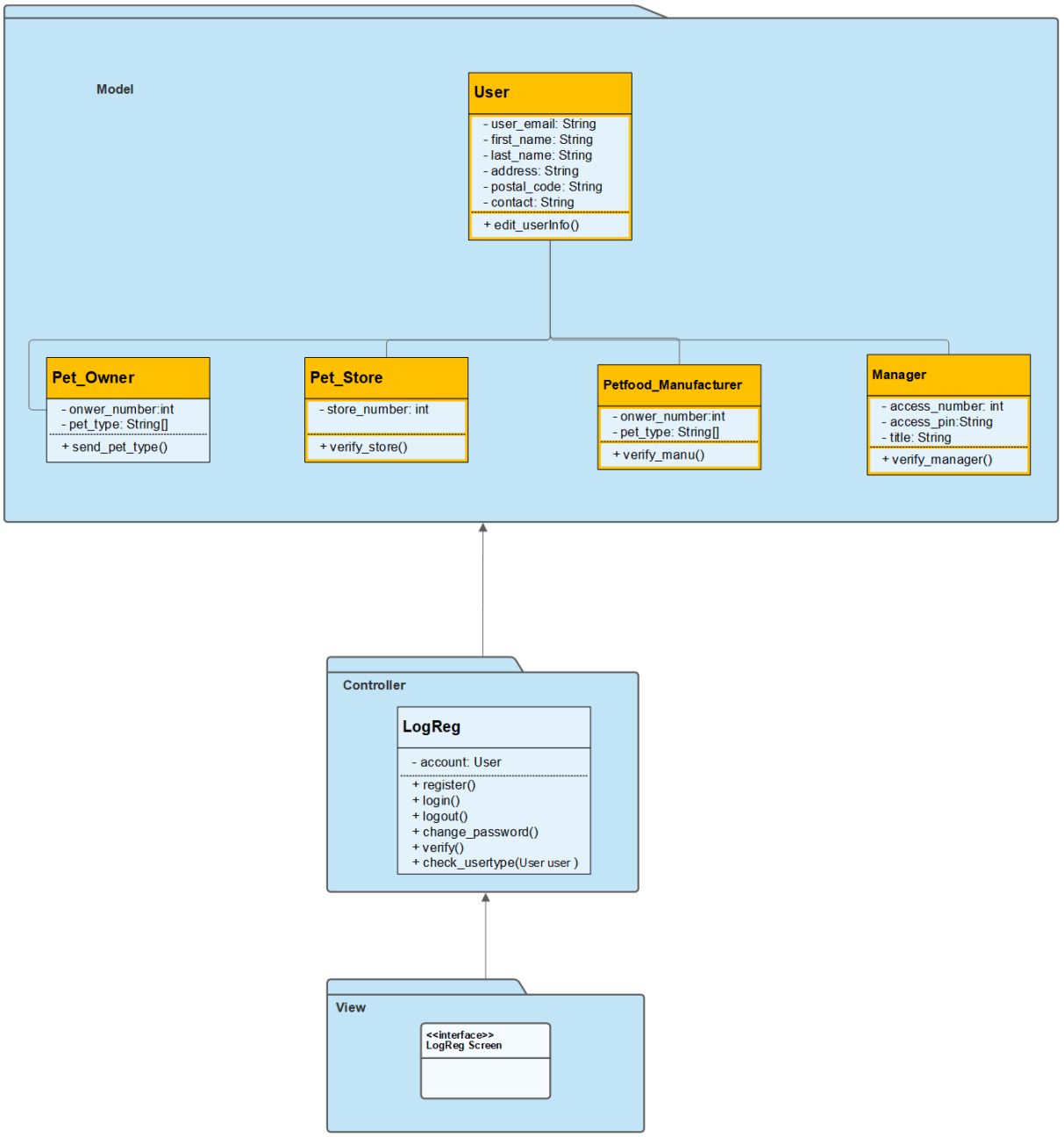
3.3.3 Rating Subsystem



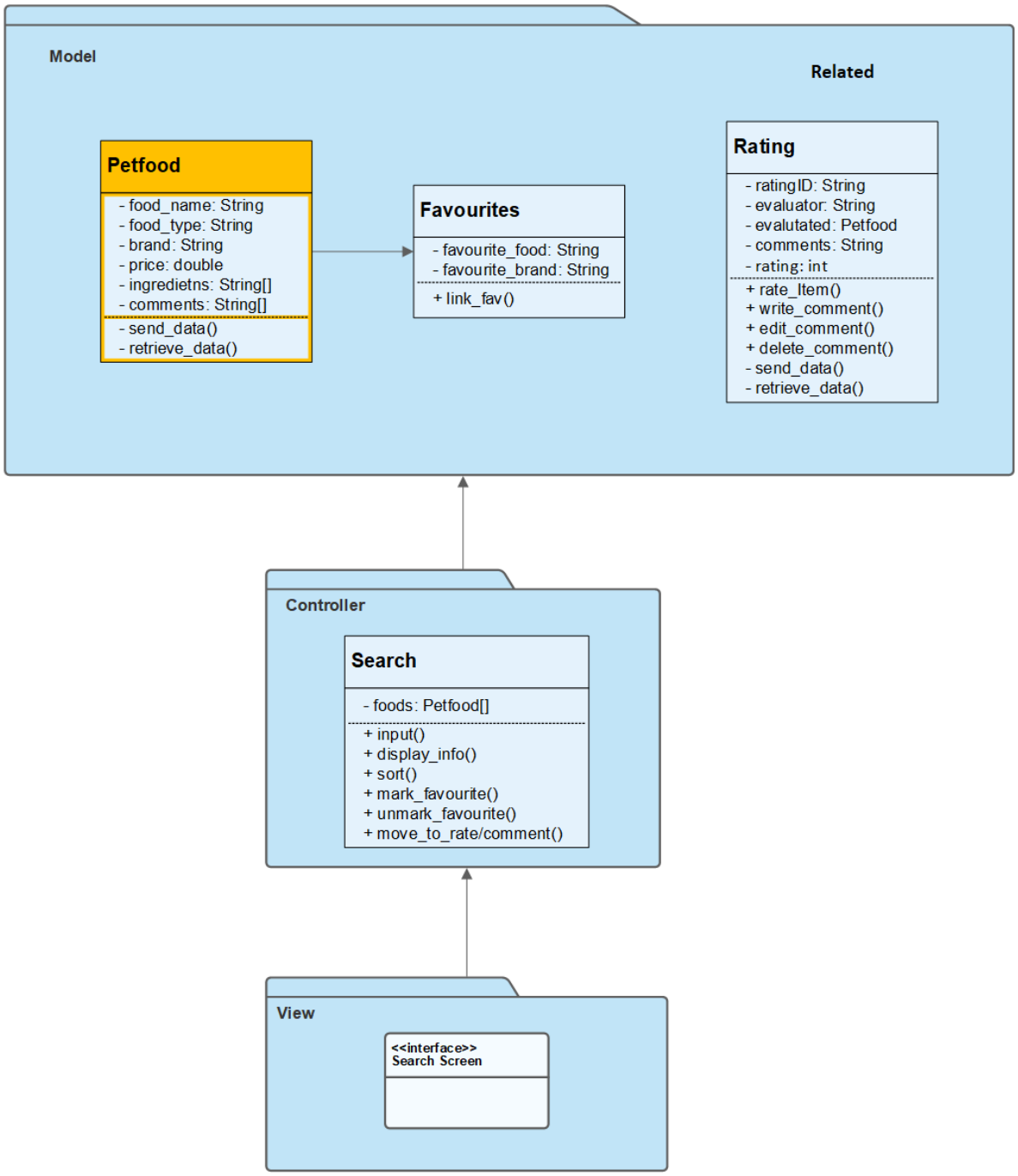
Section 4: Framework M(odel) V(iew) C(ontroller)

4.1 MVC pattern diagrams

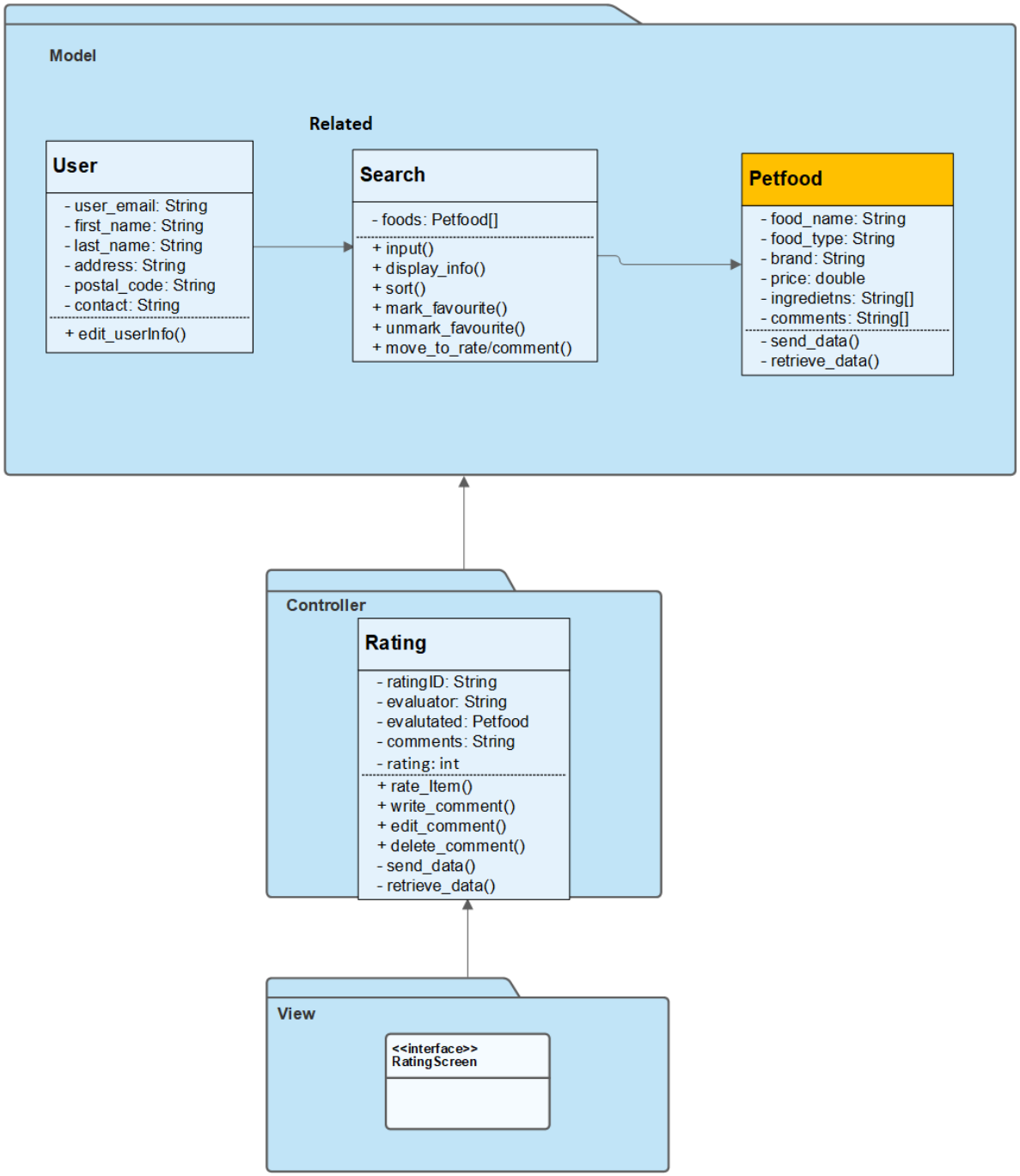
4.1.1 Login Subsystem



4.1.2 Search Subsystem

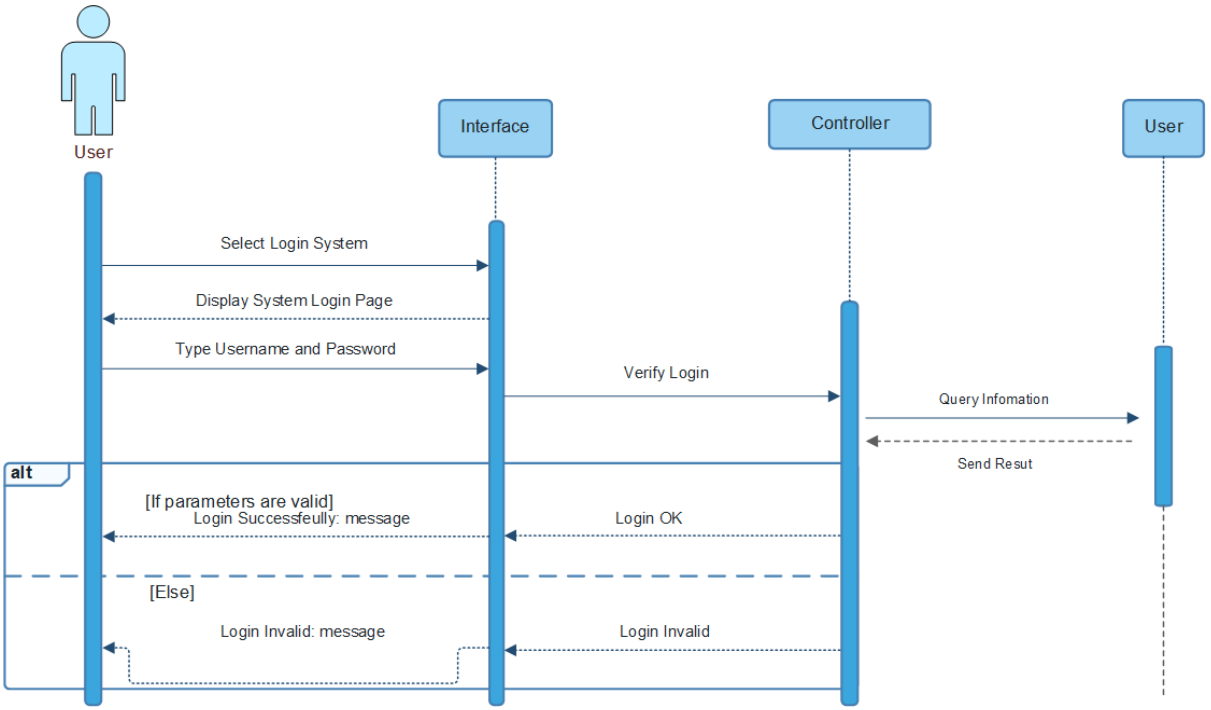


4.1.3 Rating Subsystem

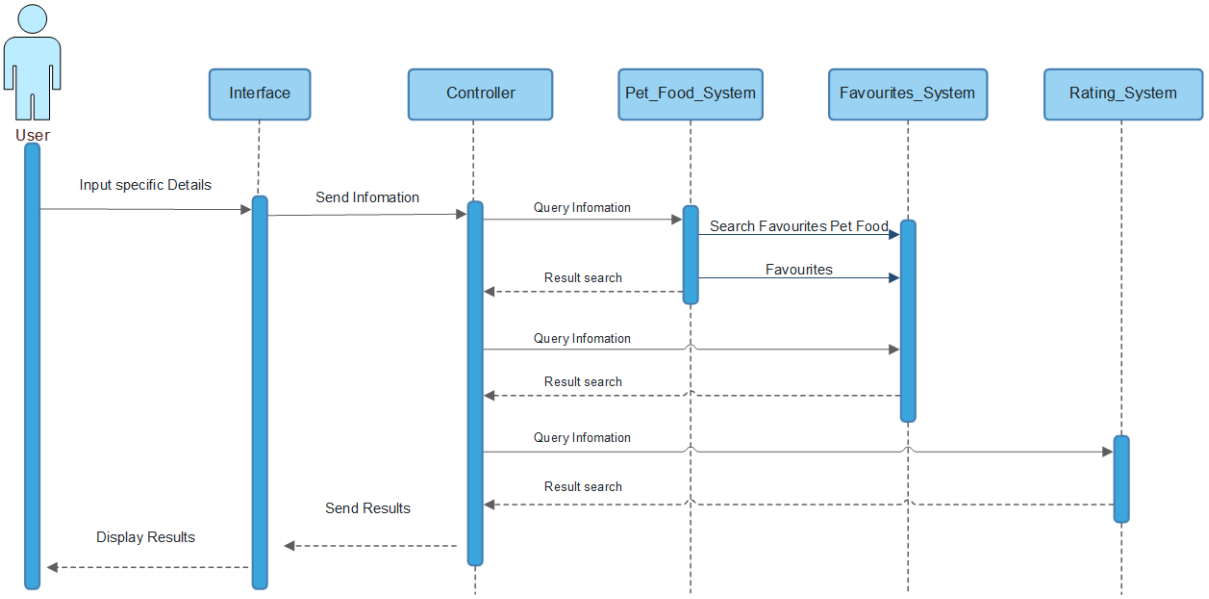


4.2 Full Sequence diagrams

4.2.1. Use case: Login (Login and Registration Subsystem)

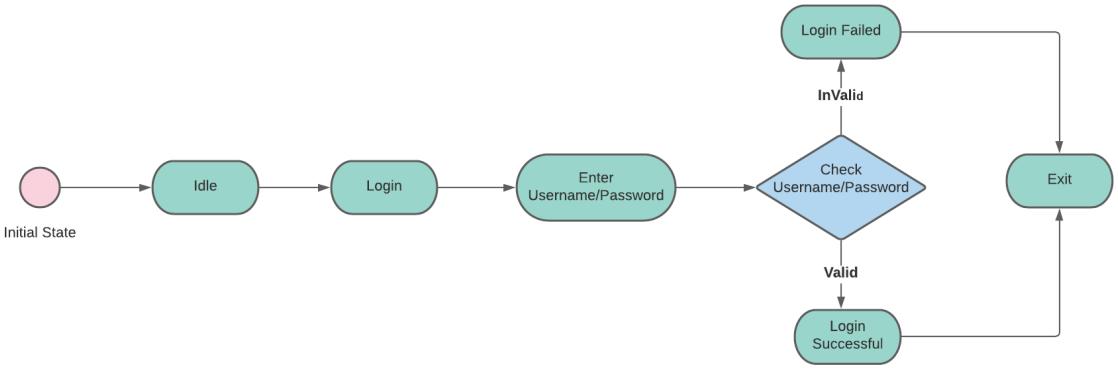


4.2.2. Use cases: Enter information and Retrieve information (Search Subsystem)

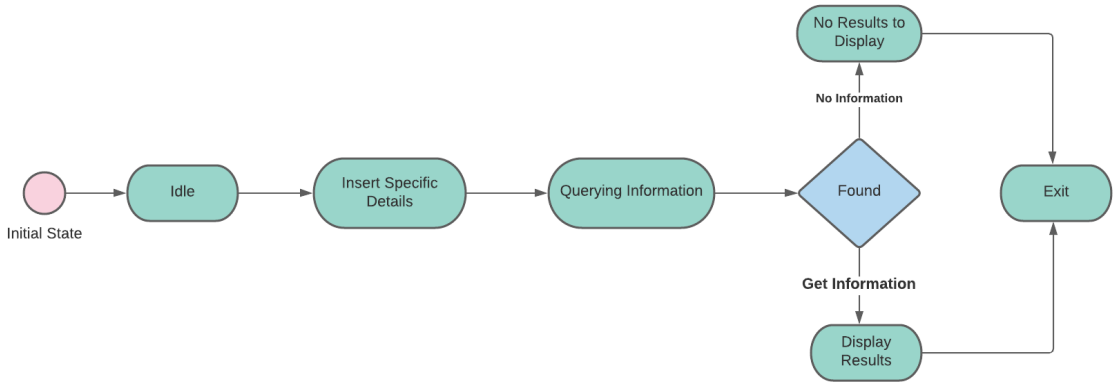


4.3 State Machine Diagrams

4.3.1. Objects: Login (Login and Registration Subsystem)

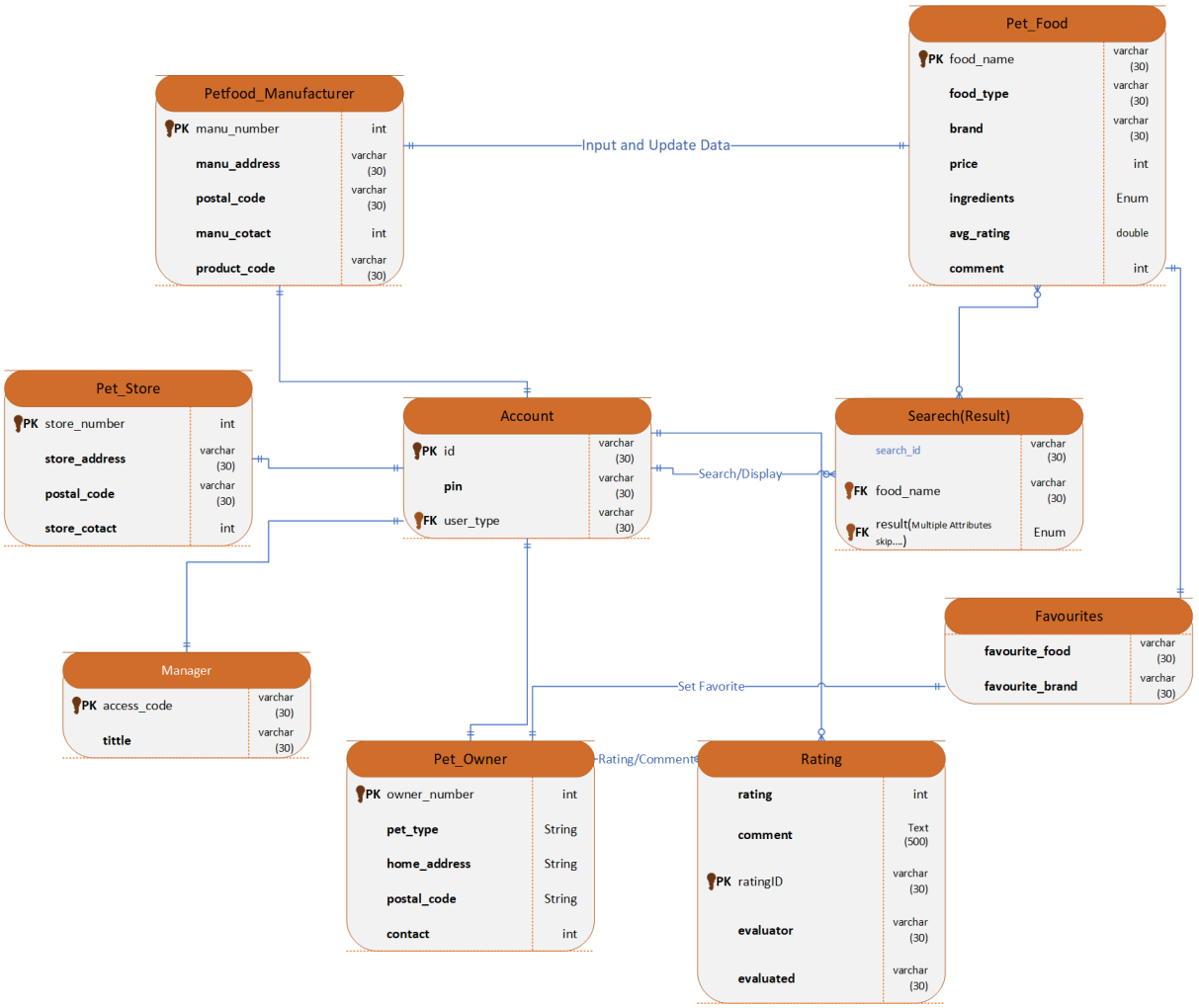


4.3.2. Objects: Search interface and Rating class (use cases: Enter information and Retrieve information)



Section 5: Data Layer

5.1. Database schema



5.2 Technology List Update

No updates required.

Section 6: Gantt chart update

ID	Task Name	Duration
1	Determine Project Topic	1d
2	Problem Statement	3d
3	Functional Requirements	2d
4	Context Flow	3d
5	Domain Class Diagram	3d
6	ERD Model	1d
7	Sequence Diagrams	3d
8	State Diagrams	2d
9	Technologies	1d
10	Intended Users of SDD	1d
11	Architectural Context Diagram	4d
12	Edits to Part A & Context Flow Diagram	1d
13	Analysis Models	5d
14	Class Responsibility Collaboration Cards	2d
15	Design Class Diagrams	5d
16	MVC Pattern Diagram	2d
17	Sequence Diagrams	2d
18	State Machine Diagrams	2d
19	Database Schema	3d
20	Technology List Update	1d

Part C

Section 1: Software Design Patterns

1.1 Facade Pattern

The facade pattern is used to protect subsystems from client access. Adding a facade interface to the software creates a type of go-between for the client and more complex parts of the internal software. We believe that the facade pattern would be helpful for our project because the subsystem would have one entry point.

1.2 Strategy Pattern

The strategy pattern allows the code to receive run-time instructions as to which algorithms or functions to use for a specific object. We believe the strategy pattern would be useful for our project regarding User subclasses. While all users have some similar attributes and functions, there are specific functions for each user that cannot be accessed by other users, for example Petfood_Manufacturer cannot "Rate" or "Favorite" Petfoods or Petfood_Manufacturers. By creating concrete strategies for each user class we would allow for future changes to these class functions and additional classes to be created without negatively impacting the original superclass.

1.3 Observer Pattern

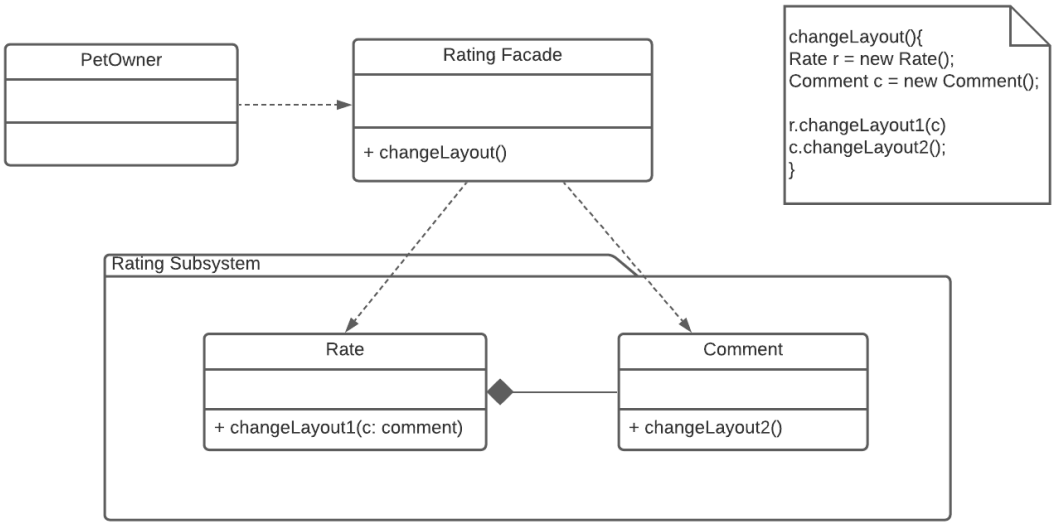
The observer pattern is used to notify all object dependents when the object state changes. It is used to simplify interactions between related objects by automatic notification. We believe that the observer pattern will be important for our project regarding petfood recalls. When a Pet_Owner "favourites" a Petfood product they should be automatically alerted when that Petfood product has been recalled, rather than having the Pet_Owner constantly check whether or not the product has been recalled. This could also be implemented for Pet_Store, but with all Petfood products available at the Pet_Store, rather than "favourited" Petfoods.

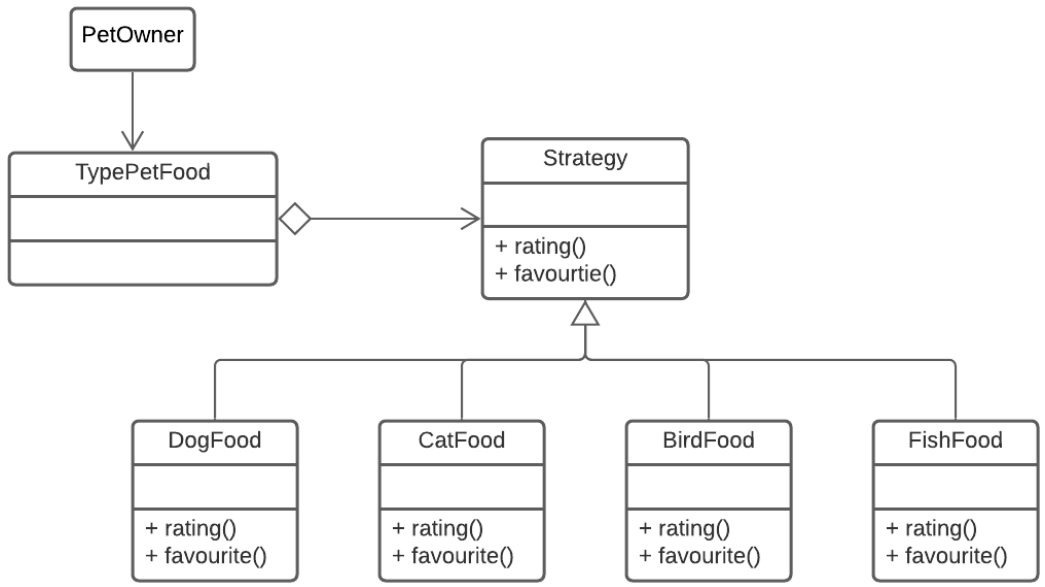
Section 2: Pattern-Organizing Table

Pattern-Organizing Table

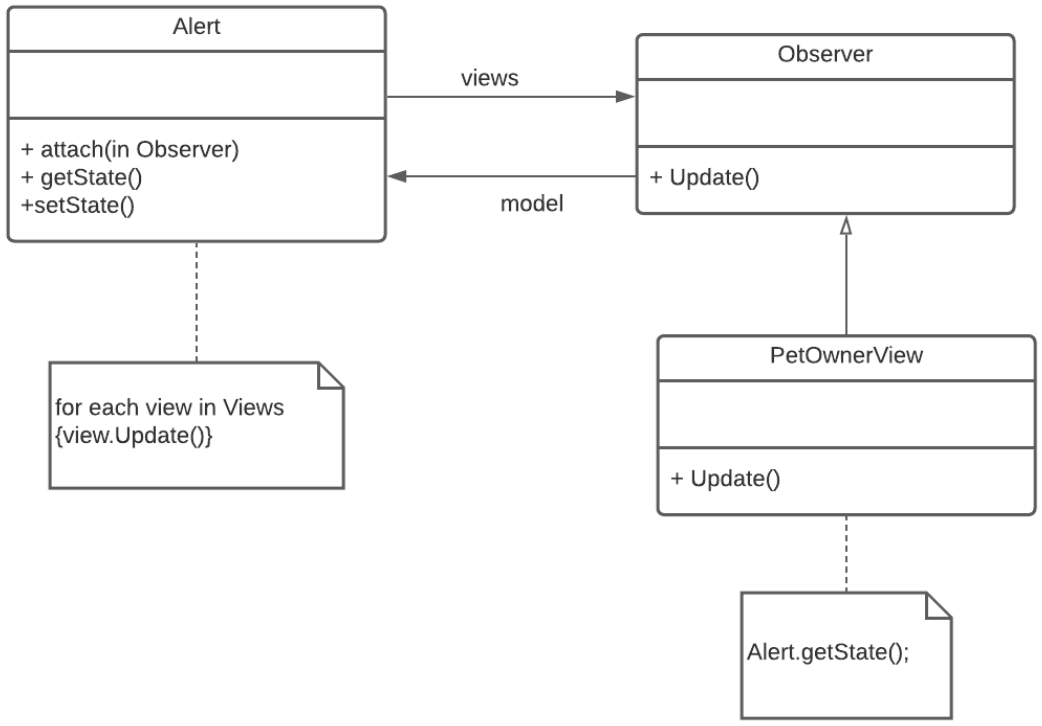
	Database	**Application**	**Implementation**	**Infrastructure**
Architecture				
Problem:		Facade Pattern	Facade Pattern	
Component-Level				
Problem:		Strategy Pattern	Strategy Pattern	

User Interface				
Problem: Pet owners need to observe the status or alert about food recalled	Observer Pattern	Observer Pattern		Observer Pattern

Name:	Facade Pattern
Problem:	A pet owner wants to change the pet food's rating layout; however, there are multiple classes involved in doing this.
Solution:	We use a facade pattern to create a wrapper around the classes needed to shield the pet owners from the complexity of commenting or rating an item.
Graph:	
Benefits and consequences:	It makes the subsystem easier to use by creating a wrapper class. One consequence is that the manufacturer will not directly access the Rating subsystem.

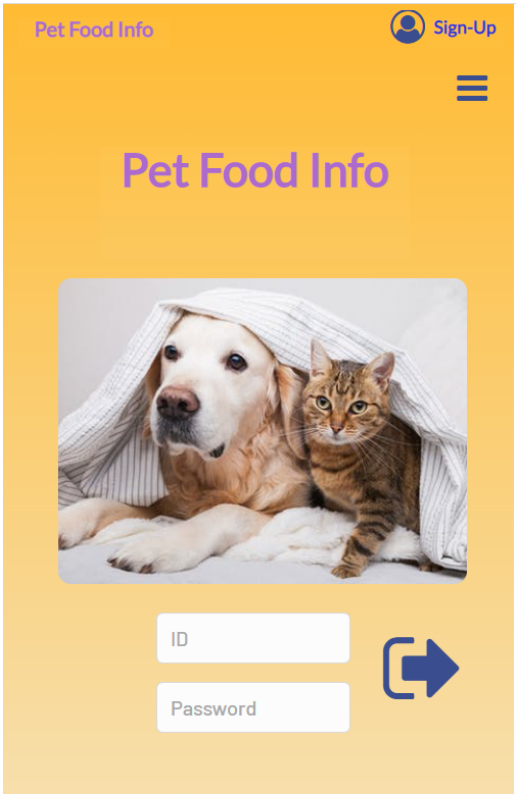
Name:	Strategy Pattern
Problem:	A pet owner wants to rate or favourite the pet food; however, there are multiple classes involved in doing this.
Solution:	We use a strategy pattern to separate types of pet food so pet owners can easily choose their favourite pet food or rate an item.
Graph:	
Benefits and consequences:	Pet owners will be easier to rate or favourite type of pet food by this subsystem. One consequence is that the manufacturer will not directly access the pet food subsystem to rate or favourite.

Name:	Observer Pattern
Problem:	Pet owners get notified which pet foods are recalled, these alert statuses will be automatically updated if recalled by the FDA.
Solution:	We can make use of subscription events pushed to the frontend using socket technology.

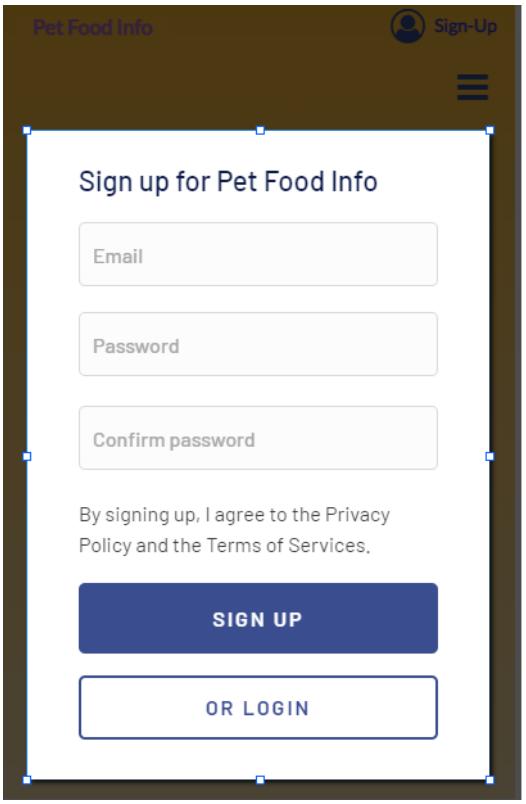
Graph:	
Benefits and consequences:	Pet owners will receive their information simultaneously, which makes it easier for them to choose safe food for their pets. By doing this, we can also use alerts so that the pet owners can be notified of which foods are recalled. One consequence of this is that it may require additional server load.

Section 3: UI/UX design

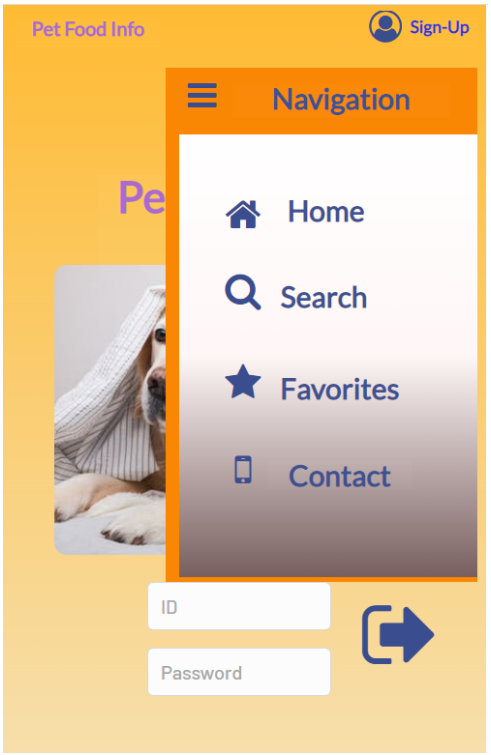
Home



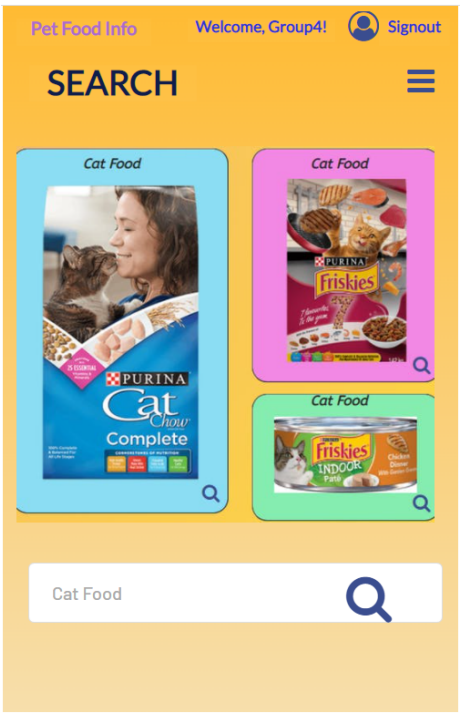
Sign Up Pop up



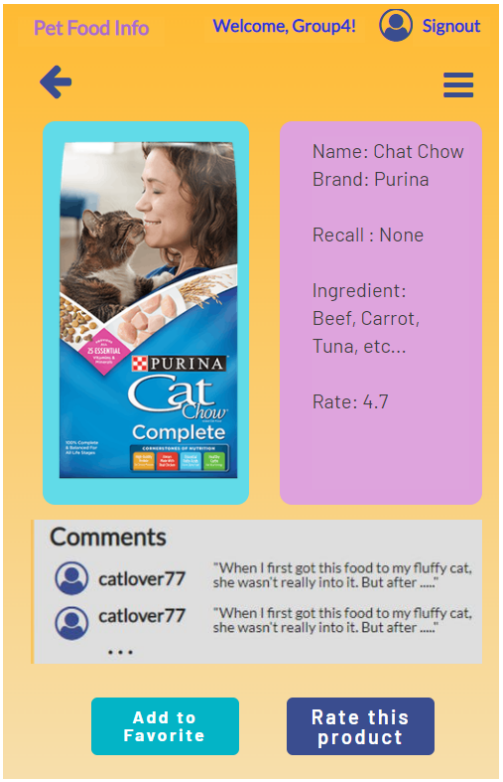
Navigation Bar



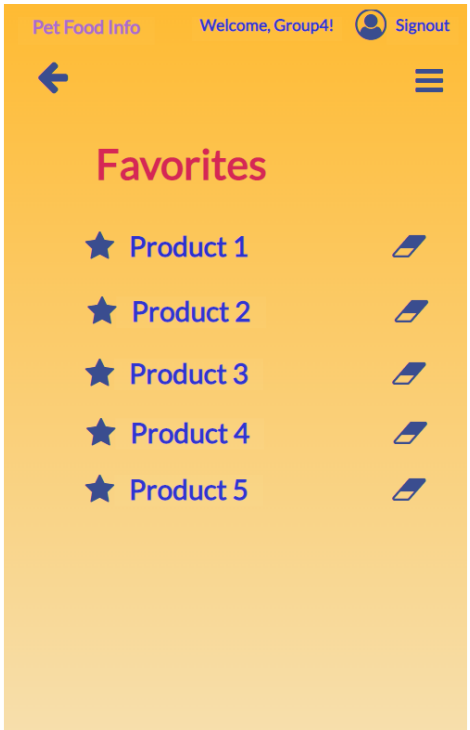
Search



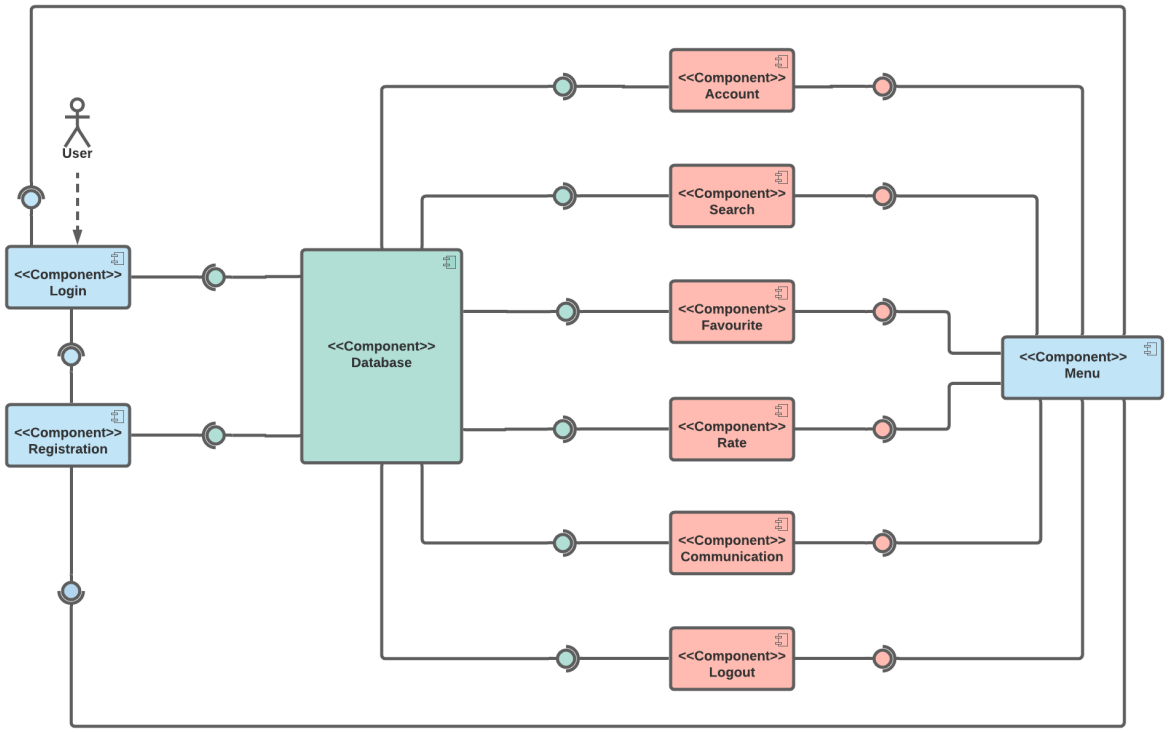
Detail (Each item in detail) with comments / Rates



Favourites



Section 4: High Level Component / Deployment Diagram



Section 5: Gantt chart update