

Name: **Hameed Hassan**

E-mail: Hameed_20180216@fci.helwan.edu.eg

Product Review Analysis for Genuine Rating

PART 1: Overview & Software Requirements Specification

1) Introduction:

a) Purpose:

The purpose of this project is to help users and customers choose the right products, with the best prices, from the best service providers. The project (website) achieves that by providing rating for every product, and also provides a list of sellers for this specific product. For the best experience, user can see other users' reviews and ratings.

b) Project Scope:

As the scope of the project goes, User can see ratings of any product that is stored in the Database, also User can compare between two products to see the differences like (price, weight, scale, resolution, version, release date, ...etc.).

*User can give a feedback about his experience so far with the website, and admins can see this information for future upgrades, and this is a real-life application for **Software Evolving**.*

Admin is responsible for managing sellers' data, information. Admin can delete edit information about a specific seller. Beside that, admin can view users' profiles, ratings, and feedback about the website.

c) Glossary and Abbreviations:

-A-

Admin : The person who is responsible for daily management of the system and manage the data base , He can update the products details and see the user's feedback and review for the system and products.

AI: Artificial Intelligence .

App: stands for "Software application" .

-C-

CBO : Coupling between objects.

CCM : Cyclomatic Complexity Metric.

Compare: Comparing two products from the same category.

CSS: Cascading Style Sheets. A type of code used to control the layout of a web page in a browser.

-D-

Database : Contains the products stored in the system with their details Such as Product's names, Product's capabilities , Product's reviews and Product's price.

DBMS : Stands for "Database Management System" .

DIT: Stands for " Depth of Inheritance Tree".

-E-

ERD: Stands for "Entity Relationship Diagram".

-F-

Feedback: Recommendation from user to improve the system.

-H-

HTML : Stands for "Hypertext Markup Language", A form of code used to create web pages in browsers .

-L-

LCOM: Stands for "Lack of Cohesion of Methods".

LOC: Stands for "Lines Of Code".

-N-

NOC: Stands for " Number of children".

Notification: Message from the admin to all the users in the system such as adding a new product.

-R-

Rate : The user rating for the products and every product can be rated only once by every user from 1.0 to 5.0

Review : The user giving a review to any product by commenting .

RFC: Stands for "Response For Class".

-S-

SOW: Stands for "Statement Of Work", it includes things like:

- Purpose of system.
- Scope of the system.
- Resources.

SRP: Stands for "Software Requirements Pattern".

SRS: Stands for "System Requirements Specification".

SSD: Stands for "System Sequence Diagram".

-U-

User : The user of the system website can buy products and see their details , Can compare two products , giving feedback of the system and review the products.

UI: Stands for "User Interface".

-W-

WMC: Stands for "Weighted Methods Per Class".

d) The List Of Stakeholders:

- The Manager "Includes the partners".
- The Sponsors.
- The Sellers "Owners of products".
- The Developers.
- The Companies which deliver the products.
- The customer which buys products.
- Anyone benefited from the project.

e) References:

- (Book)Software requirement patterns: By Stephen Withall
- (Book, as quotes)The Timeless Way of Building (1977): By Christopher Alexander
- <https://www.wayferry.com/glossary>

2) Software Requirement Pattern (SRP):

Whenever we develop a software, the same problems, and obstacles. The same requirements recurrent over time, and we deduce from that that we must not reinvent the wheel or redo the same solutions from scratch, so we have to use some sort of patterns to use it whenever we face any of the selected problem.

Software requirement pattern (SRP) is a way of solving a specific common software problem that happens over and over and this solution works the best (so far) for this problem, so we can solve this problem no matter how many times it happened again, with the same time efficiency, and the same processes as any problem that uses this pattern.

3) Functional Requirements:

a) User Requirements Specification: The user requirements specification described the needs that the user wants from the system , It was written early by the system owners and end-users in the documentation before the system is created .

Example:-

- User can see other users' ratings of the same products.
- User can also writing his review and let other's know what he thinks of this product.
- User can compare products with the same category .

b) System requirements Specification: the system requirements specification is a document written to describe the purpose of the system and what the system will do and perform .

Example:

- *In our system the system automatically generates the feedback from the user as a report to be printed by the admin to read it and improve the system.*
- *The User must be have a email and password to log in the system if he doesn't have he must register a new account to be able to log in.*
- *The user can choose the method of delivery of the product.*

c) Requirements Priorities(using MoSCoW Scheme): MoSCoW Scheme is a technique for prioritization used in many fields such as :management , project management ,business analysis and software development to be understanding the requirements of the stakeholders.

*The term "MoSCoW" is derived from the first letter of each category of four prioritization categories (**Must have, Should have, Could have, and Won't have**).*

- **Must have:** *System must have a browser to view and interact with the web-app , and also it must have an internet connection so that the project can connect to server and fetch the data that users entered for the product, and the basic details of any product the user views.*
- **Should have:** *System should have a good stable internet connection, not just a working connection, otherwise the browser will take some time to fetch data and view it for the user, or (in some cases) the connection will not perform a proper connection to the serve, and data may be corrupted.*
- **Could have:** *System could have an updated browser, with the latest stable update so it can view any interface with the right design that it meant to be without any issues with the user-experience.*
- **Will not (Won't) have:** *System may have a good hardware device, to speed up the process as a fancy feature.*

4) Non-Functional Requirements:

a) the general types of non-functional requirements:

- **Reliability:** *Just keeping the quality of products review integrity by every-way possible, to provide a reliable source of thinking shape, and intuition making for users about some products.*
- **Manageability:** *For that reason, there are two point of views, to view different aspects of projects data. Those views are (User view, and admin view). Admin view will change some points in the architecture, like how to view data that is collected, and how admins manage everything that goes through the website, which of course acts as site-supervision, and user trust.*
- **Maintainability:** *We provided a dedicated section for user feedback, and comments to help us improve the service more and more. Of course, this helps maintain the website, and find minor issues and fix them as soon as possible when the user*

reports for an issue he faced and disrupted his experience. That changed some coding dimensions...we added this section in almost everywhere, by attached it to navbar, so we provide the user with the ability to give us any comment anytime.

b) Non-Functional Requirements Specification: it specifies the attribute's quality of a software system, they judge the software system depends on many non-functional standards that critical to success of the system such as : Responsiveness, Usability, Security, Portability and other non-functional standards.

c) The Fit-Criteria for every Non-Functional Requirement:

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

5) Domain Requirements:

- ***The validity of product:*** the product shown in the system must be in validity time and not expired.
- ***The extent of corporate credibility:*** the product shown in the system must be valid for use, effective, not used before, complete if it consists of many parts and formally documented.
- ***The product quality:*** the product shown in the system must be works very efficiently and there are no industrial defects.
- ***Availability quantity of product:*** You must make sure that there is a sufficient quantity of products to cover the requests and not to be there is shortening in the quantity Especially when there is a lot of demand for the product.

6) Design & Implementation Constraints:

- System is developed using (HTML5, CSS, JavaScript, bootstrap) as the tools for Frontend.
- Backend and data management is developed using pure PHP only, without any special, or additional framework to do the job.
- For databasing, and data management, we use SQL, why? Because data is a bit complicated to store as Non-SQL services or JSON data type for storing. For instance, users have reviews, reviews have comments or/and ratings. In some cases, we would like to search for a specific thing like comments review, or we can view a specific value to view, and of course this is relatively a bit complex data to store as something different than SQL.

7) System Evolution & Anticipated changes:

Our software is very able to change easily, especially because we kept in mind the manageability and reliability as a non-functional requirement, so we can make a good use of feedback section for updating UI, or even the functionality. Changing one of the main functionalities is an anticipated change, for instance, user might not like the way the rating process is happening, or even the way of suggesting products to them, some people might recommend to put a toggle for this feature in order not to collect any basic data about the user.

Any software updates in time, and this project is no special case. We can add a new way of taking review in addition to the classic way, and we knew that from the beginning so we made everything dynamic in backend, like the way we store the review for example.

Some changes in the future might not be handled with the current design. If we would like to make the commenting section on every product a real-time-changing element, unfortunately we will have to add more architecture to observe if any additional comment new to the section, or we might change the current architecture without adding new one, if we wanted to use SOCKETs for instance.

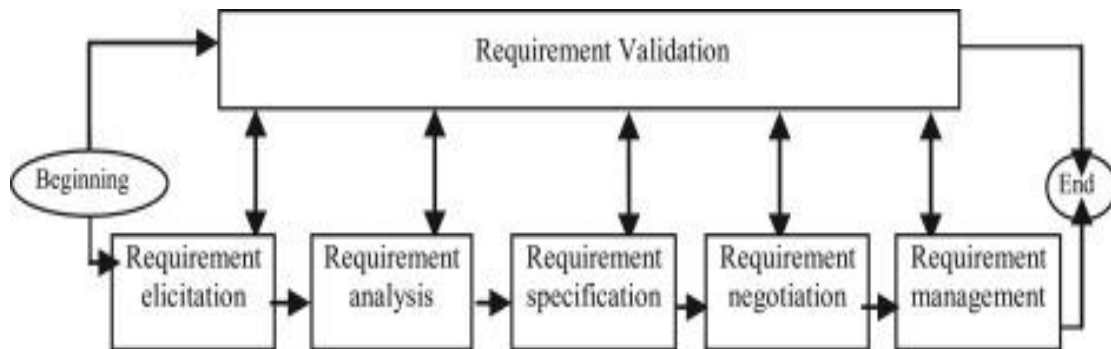
8) Requirements Discovery :

One of the approaches we used to get more and more requirements is “Brainstorming”. This is very useful for our team as a way of finding more problems to solve, and it’s very effective in our case.

For example, when we tried to find something that we can use to get more recommendations for user, and letting them expose edge-case bugs or problems. We found from meeting sessions that if user liked the user-interface or found it attractive to do something, then we can gently-force user to give us his feedback. We also found when we performed empathy on user cases, that if I found some sort of issue or bug or whatever, and I did not find a direct and easy way to report that event, simply I would do nothing positive for the website. Later we came up with the idea of making the navbar appear in almost every screen, and simply attach the “Feedback” button to it.

We are using “empathy” skills (that we learned through the project) too much, and we guess it’s super effective in the term of knowing what user would do in a certain situation. This also helped us to find the best way possible to design and implement a good website.

9) Requirements validation:



- **Requirements validation** is the process that check that the requirements defined for development and define the system that the customer really wants.
- We perform requirements validation to check the issues related to the requirements.
- The requirements validation is used to check the error at the initial phase of the development as the error may increase when detected later in the development process.
- The requirements must be based on the available budget.
- In the requirements validation process we perform a different type of test to check the requirements mentioned in the Software requirements specification (SRS).

There are several techniques that used either individually or with other techniques to check all the system or part of it:

Test-case generation: the requirements mentioned in the (SRS) document should be testable. Generally believed that if the test is impossible to design, this usually means that the requirements will be difficult to implement.

Requirements Reviews: The SRS is reviewed by a group of people; the reviewer will analyze the document to check the error and ambiguity.

Prototyping: in this validation technique the prototype of the implemented system is presented before the client or end-user, they experiment the prototype and check if it meets their need or not. The main goal of this type of model is collecting feedback from users about their requirements.

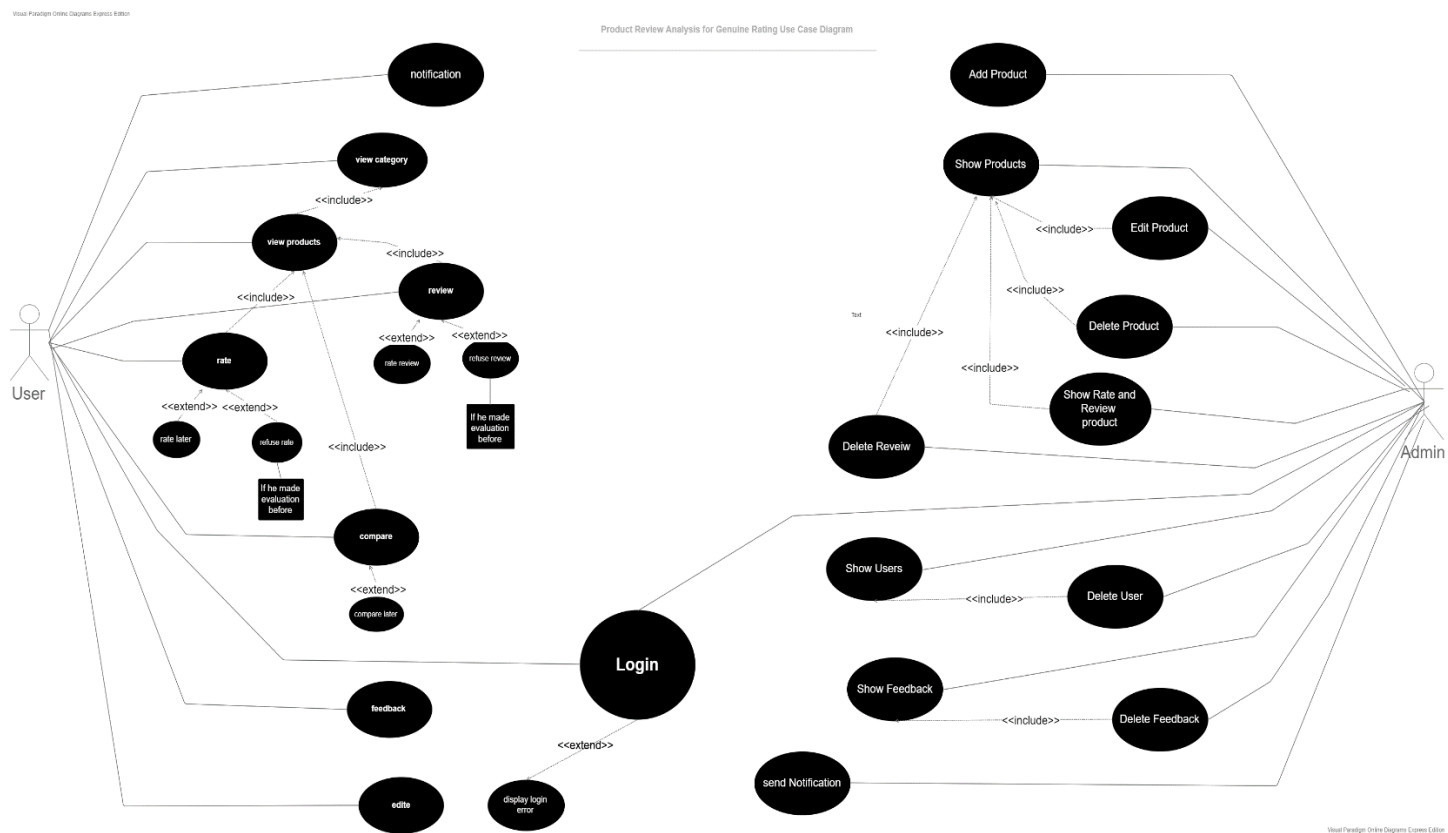
Examples

1. when the customer adds his e-mail, it should achieve the conditions e.g. (having the sign @ and .com)
2. when the customer adds the no of credit card it must has 16 numbers
3. When someone logs into the system, the system must determine whether it is the admin or the user
4. to complete the order, the user must specify one item at least if he clicks on (Buy Now)

PART 2: System Design & Models

10 Functional Diagrams:

a) Use-Case Diagram(s)



b) Detailed Use-Cases Description

1- Login

Uses case name

Actor

Pre-condition

Basic flow

Login

Admin, user

User or admin enter username and password

Open login page

Alternative flow	Enter username and password Click on login button Username or password is incorrect Re-enter username and password again
Post-condition	If login successfully user or admin can enter to this view in the system

2- Add product

Uses case name

Actor

Pre-condition

Basic flow

Alternative flow

Post-condition

Add product

Admin

Admin make login successfully

Click on add product in toggle bar

Enter product details

Click on add product button

Admin forget a field in form then enter data in this field and submit

If add product made successfully product will added in database

3- Show products

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

show products

Admin

Admin add product(s)

Click on show products in toggle bar

Admin show all product in the system which added by admin

4- Edit product

Uses case name

Actor

Pre-condition

Basic flow

Edit product

Admin

Admin list all products in the system

Click on show products

And click on Edit button to product which admin want to edit it.

Admin edit data he wants in update form after this click on update button

Post-condition

If edit product made successfully product will edited in database

5- Delete product

Uses case name

Delete product

Actor

Admin

Pre-condition

Admin list all products in the system

Basic flow

Click on show products

And click on delete button to product which admin want to delete it.

Post-condition

If delete product made successfully product will deleted from database

6- Show Rate and Review product

Uses case name

Show Rate and Review product

Actor

Admin

Pre-condition

Admin list all products in the system

Basic flow

Click on show products

And click on Rate and Review button to product which admin want to show it Rate and Review.

Post-condition

Show Rate and Review to this product

7- Delete Review

Uses case name

Delete Review

Actor

Admin

Pre-condition

Admin list all reviews to the product which admin want

Basic flow

Click on Rate and Review

And click on delete button to review which admin want to delete it.

Post-condition

If delete review made successfully review will deleted from database

8- Show users

Uses case name

show users

Actor

Admin

Pre-condition

User make registration on system

Basic flow

Click on show users in toggle bar

Post-condition

9-Delete user

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

10-Show feedback

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

11-Delete feedback

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

12-Send notification

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

13-View category

Uses case name

Admin show all users in the system

Delete user

Admin

List all users in the system

Click on show users in toggle bar

Click on delete button which admin want to delete.

If delete review made successfully review will deleted from database

show feedback

Admin

User make feedback in system

Click on show feedback in toggle bar

Admin show all feedbacks in the system

Delete feedback

Admin

List all feedbacks in the system

Click on show feedback in toggle bar

Click on delete button which admin want to delete.

If delete feedback made successfully feedback will deleted from database

Send notification

Admin

No pre-condition

Click on notification in toggle bar

Enter notification details

Click on send notification button

If send notification made successfully notification will added in database and send to user at the same time

View category

Actor

Pre-condition

Basic flow

Post-condition

14-View products

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

15-Rate

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

16-Review

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

17-compare

Uses case name

Actor

Pre-condition

Basic flow

user

User make login successfully

Go to our category in index page

Show all category in the system

View products

user

User choose category

Go to our category in index page

Click on view products button

Show all products in this category

Rate

User

The product exists

Go to our category in index page

Click on view products button

Click on product details button

Choose rate from 1 to 5

Submit rate by click on rate button

Rate added successfully in database

Review

User

The product exists

Go to our category in index page

Click on view products button

Click on product details button

Enter review

Submit review by click on review button

Review added successfully in database

Compare

User

More than one product in this category

Go to our category in index page

Post-condition

18- feedback

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

19- edit profile

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

20- notification

Uses case name

Actor

Pre-condition

Basic flow

Post-condition

Click on view products button

Click on product details button

Click on compare to another product

Choose second product of the same category to compare with first product

User show comparing between the two products

Feedback

User

User make login successfully

Go to footer in index page

Write feedback

Submit feedback by click on submit button

User feedback added successfully in database

Edit profile

User

User make login successfully

Go to nav bar in index page

Click on profile

Update data in form

Submit the new update

User details updated successfully in database

notification

User

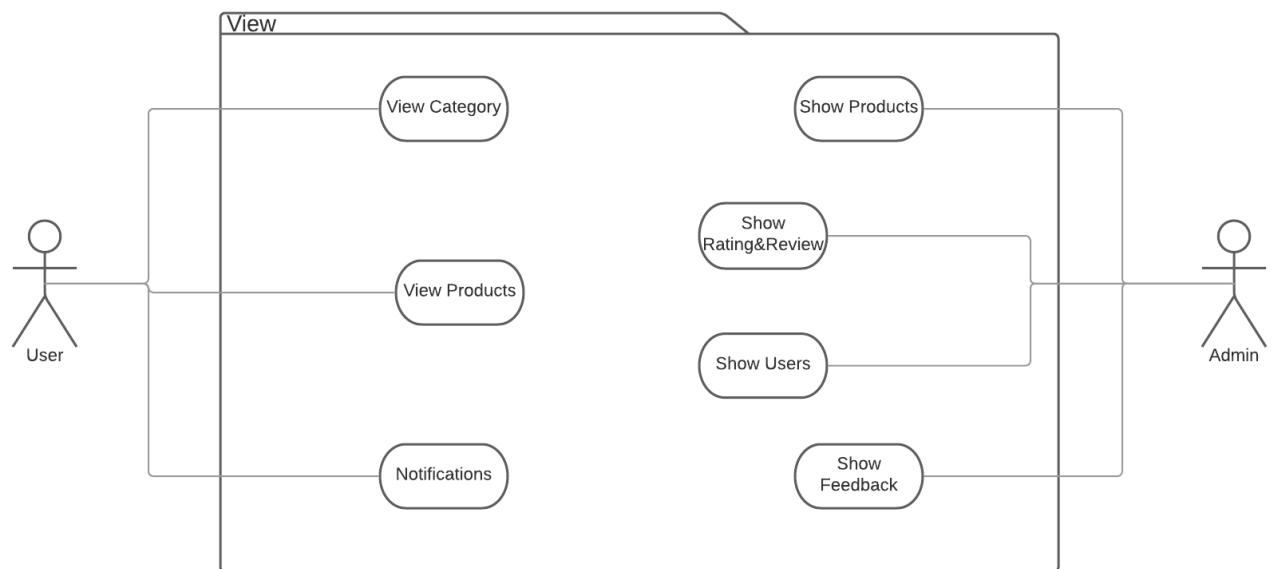
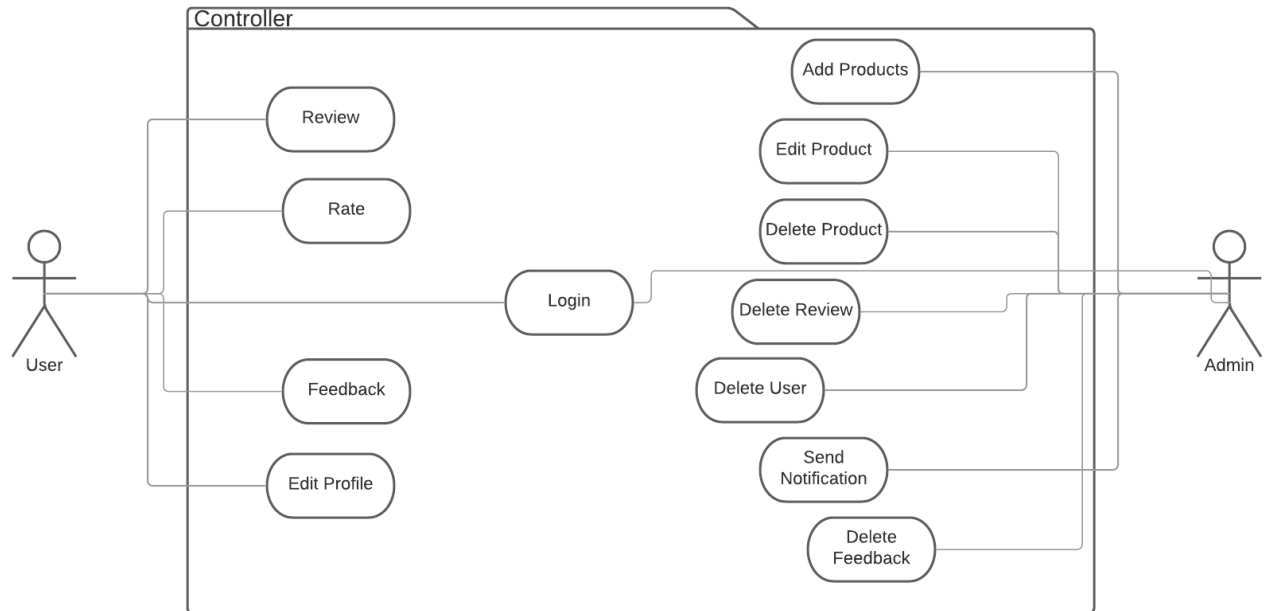
User make login successfully

Go to nav bar in index page

Click on notification

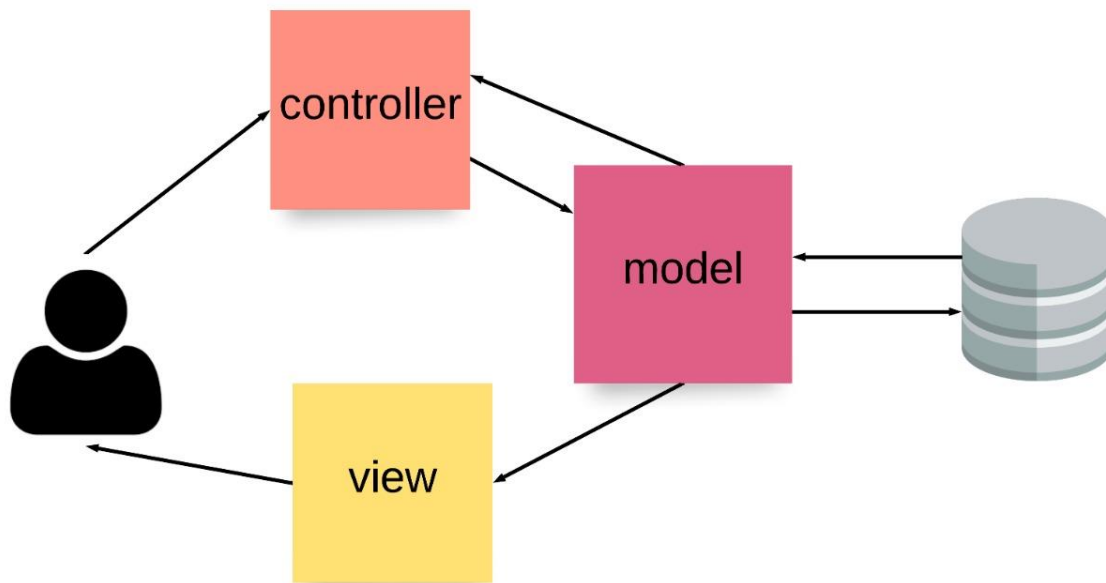
User show all notification that send by admin

c) Package Diagram grouping relevant Use-Cases into Packages.



11) Structural & Behavioural Diagrams:

a) System Architecture



Model

- interacts with database
- Executes business logic

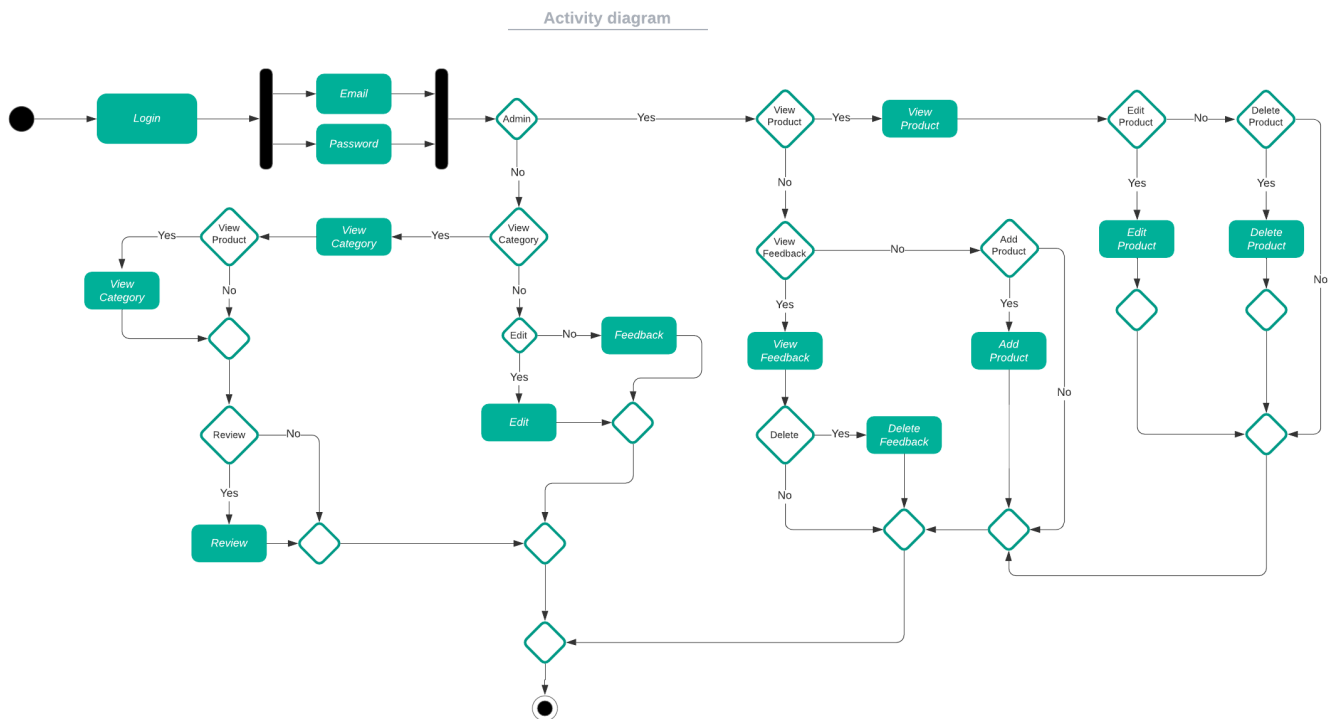
Controller

- Take user input as a (request parameters)
- interact with model and view

view

- what user sees on the screen
- generate UI for the user in a friendly way

b) Activity Diagrams



c) Based on the Activity Diagrams, the List of User Interfaces required for the System, and the corresponding users of each interface.

Interface

Login page

Send feedback (page to send feedback about website)

View feedback (show feedback which users sent them)

Add product page (page to add a new product)

View category (show category of products)

View product (page to show product)

User of interface

Admin, customer
customer

Admin

Admin

customer

Admin, customer

Edit page (page to update his/her customer data)

Edit product (page to edit details of product) Admin

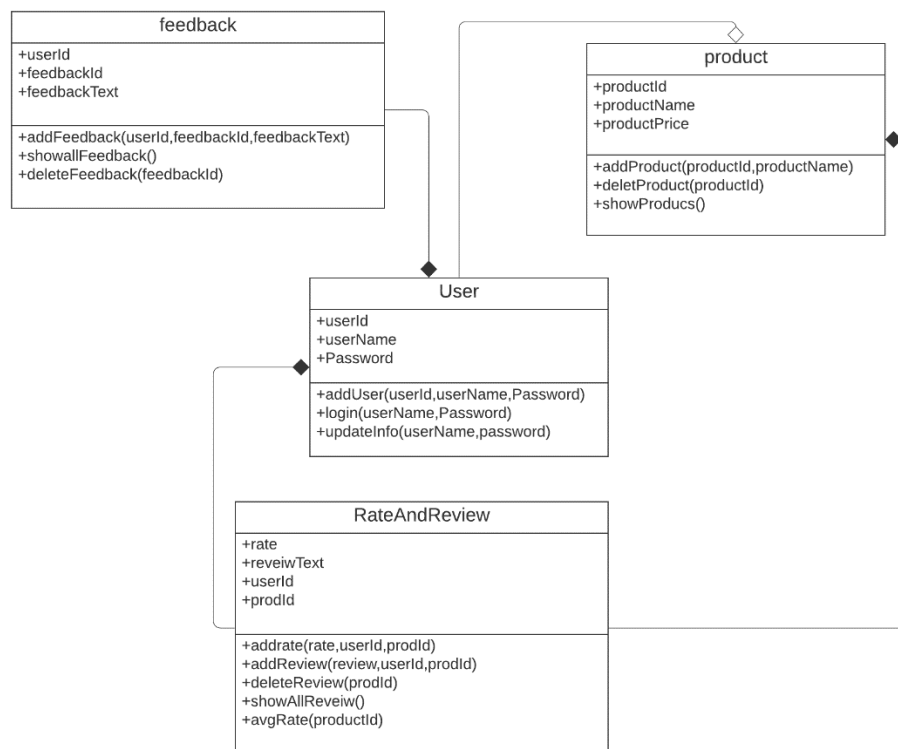
d) What is a Software Analysis Pattern? Apply at least one analysis pattern while analysing and designing your system.

The process of understanding and specifying in detail what an information system should do.

Analysis patterns are useful for discovering and capturing business processes. This helps to reuse the solution and OOPS design for similar instances of analysis patterns.

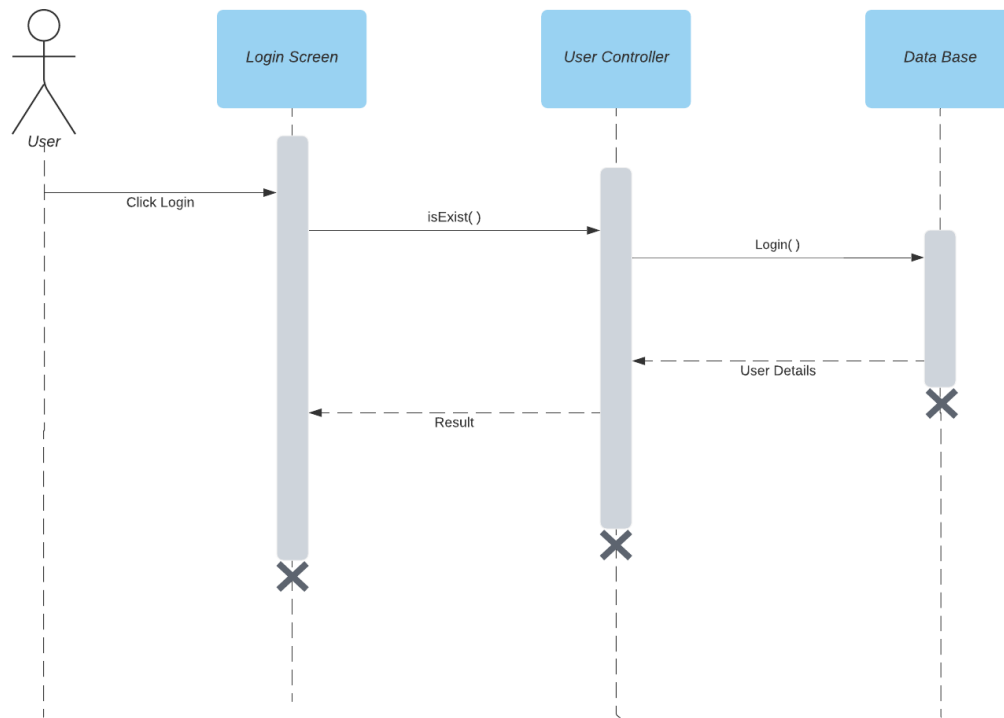
They are patterns that a business analyst or a systems analyst will encounter often.

e) Class Diagram 1:

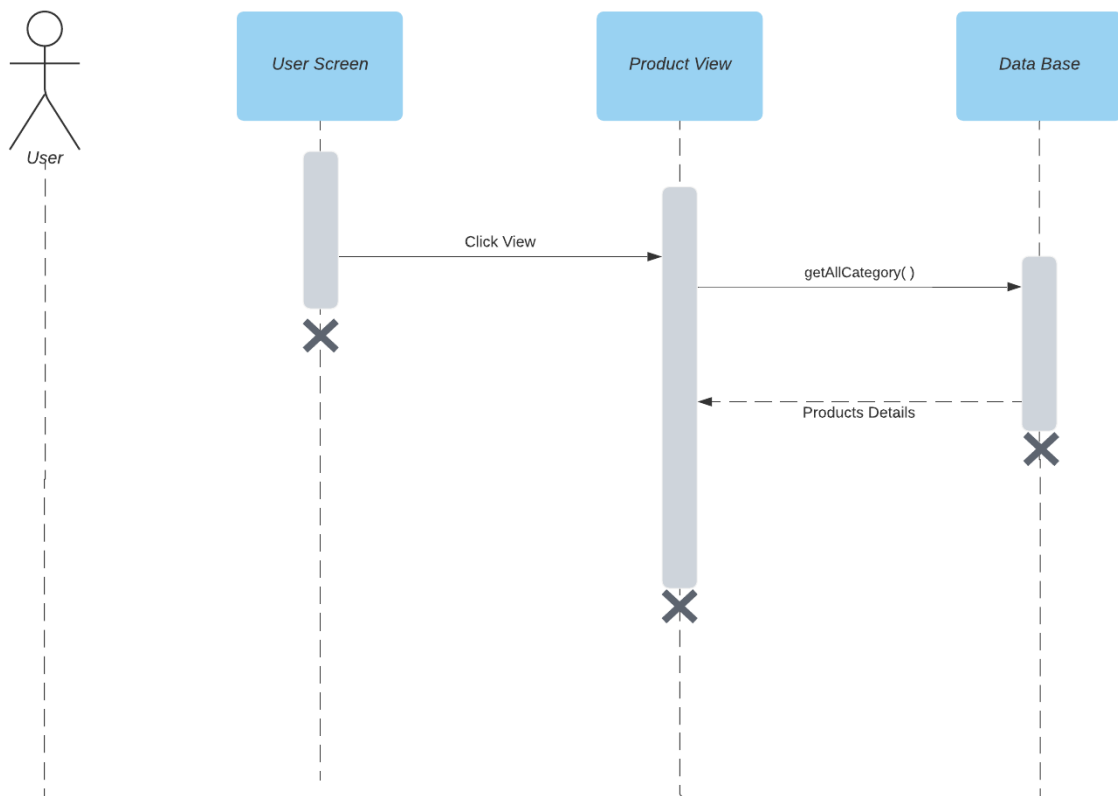


f) Sequence Diagram(s)

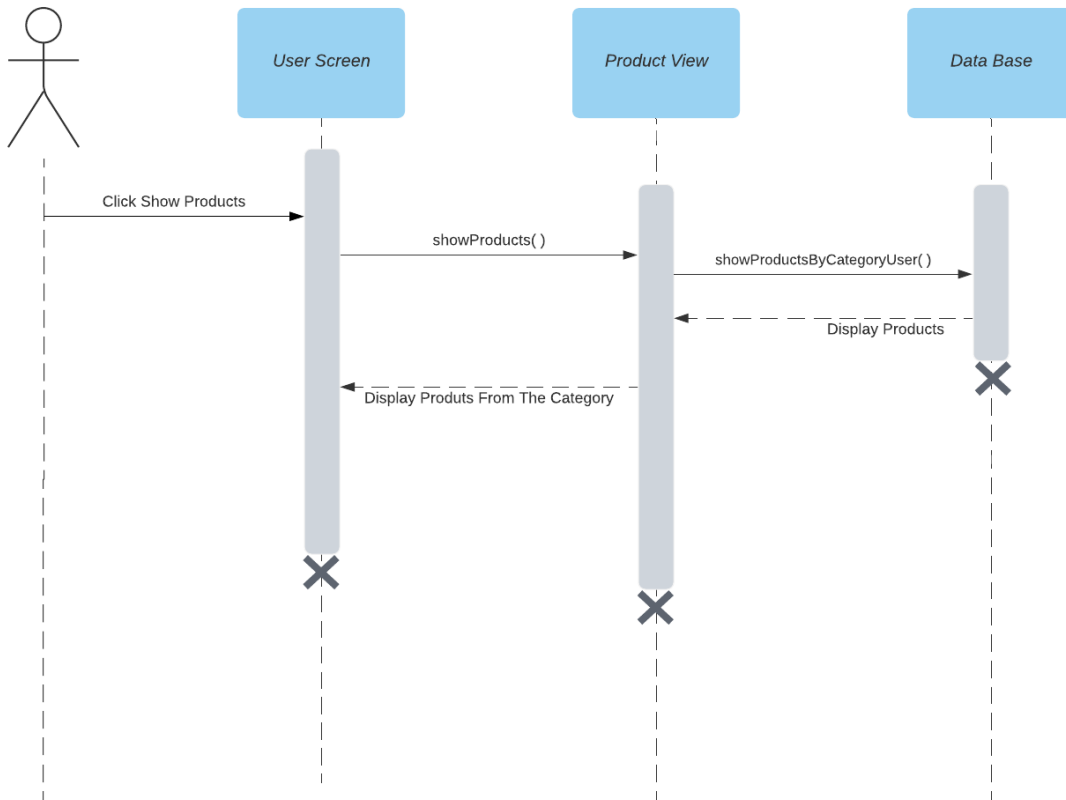
Login



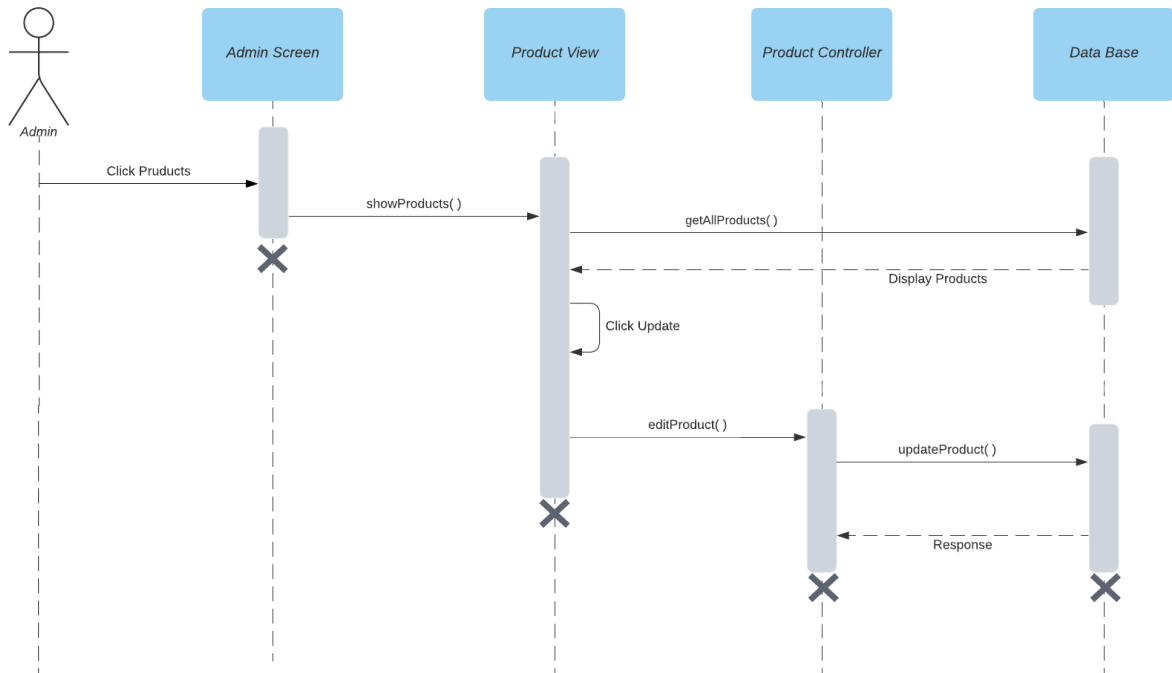
User View Category



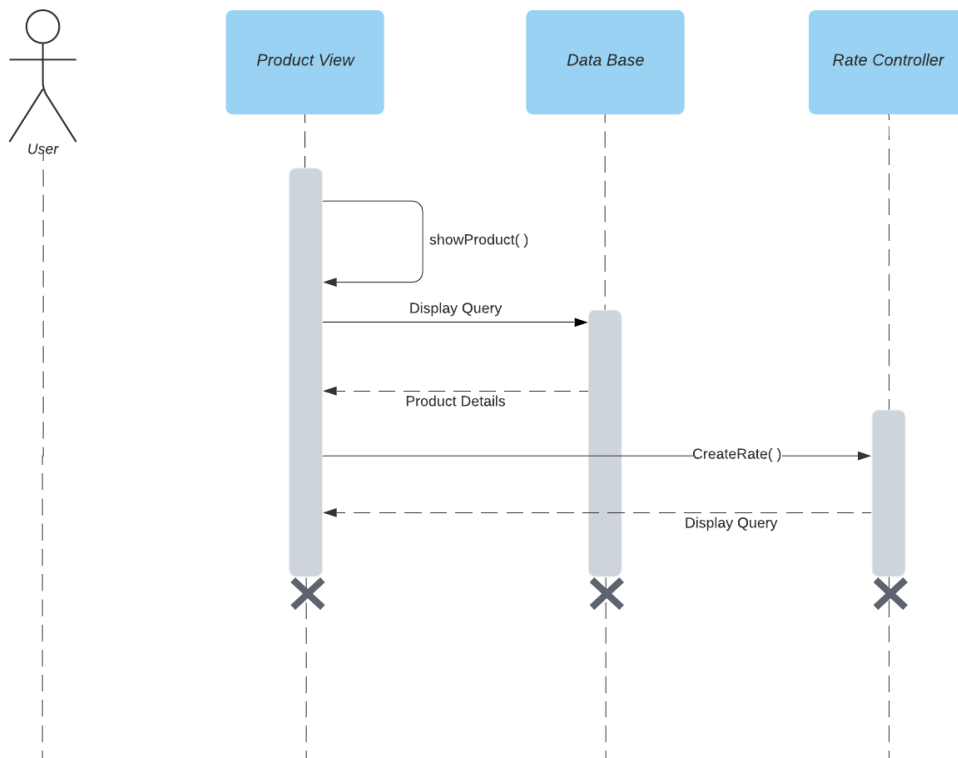
Show Products From a Category



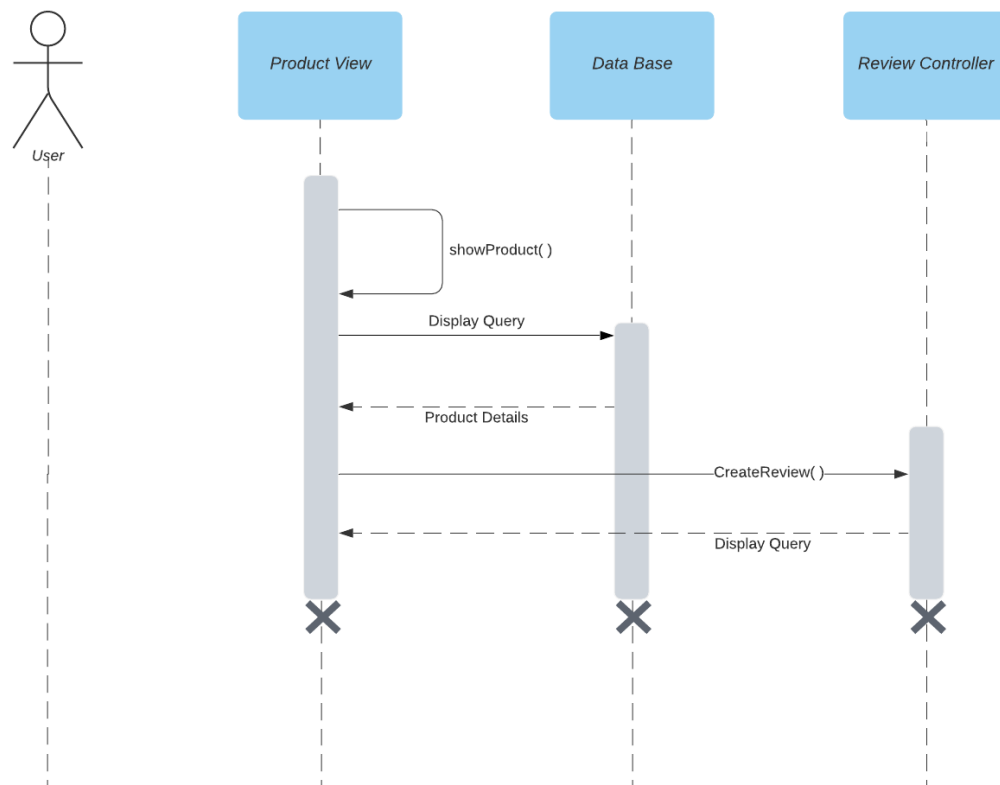
update Product



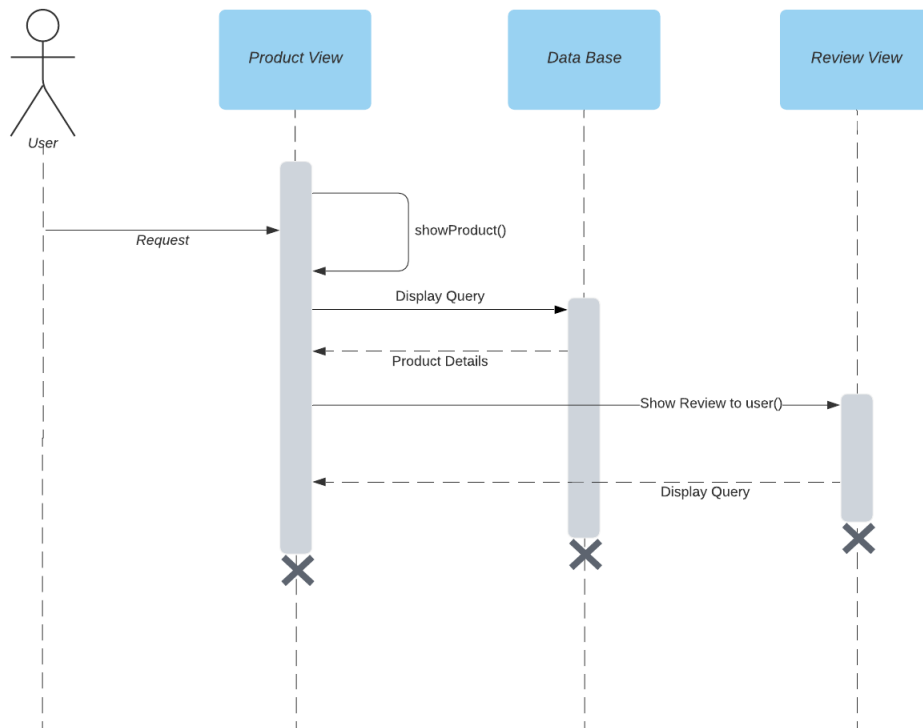
Rate a Product



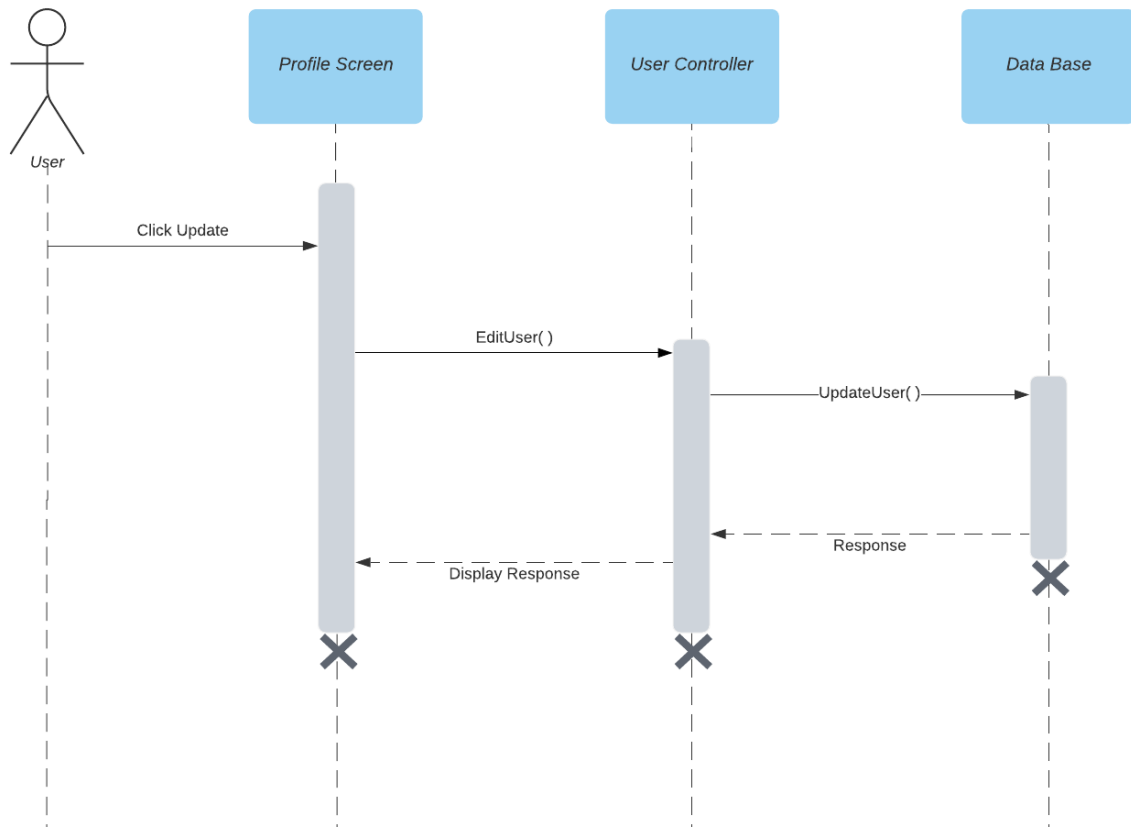
Review a Product



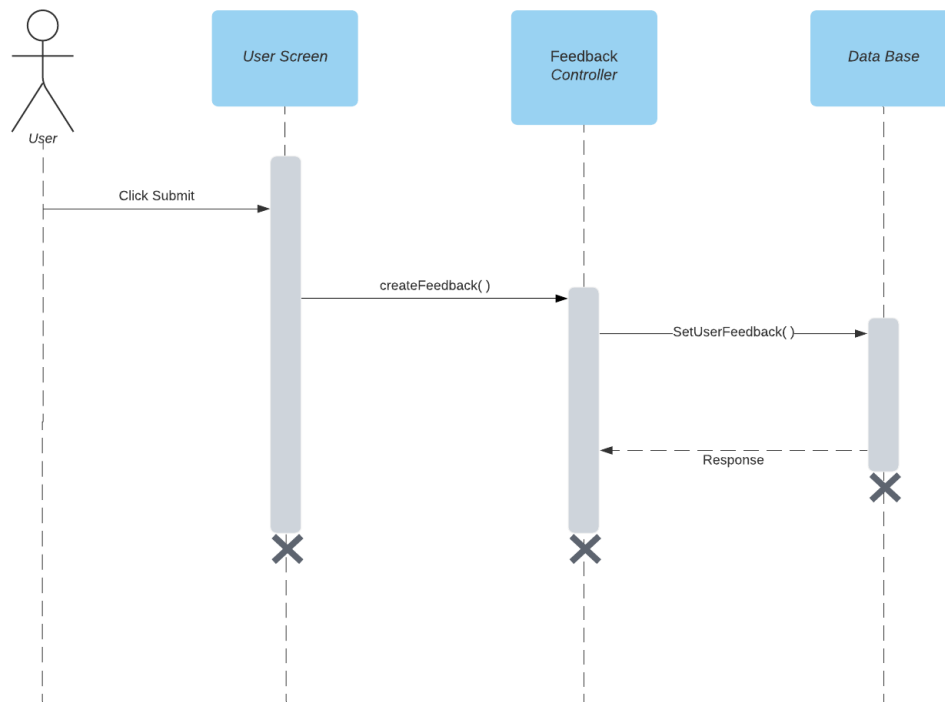
Show Review To User



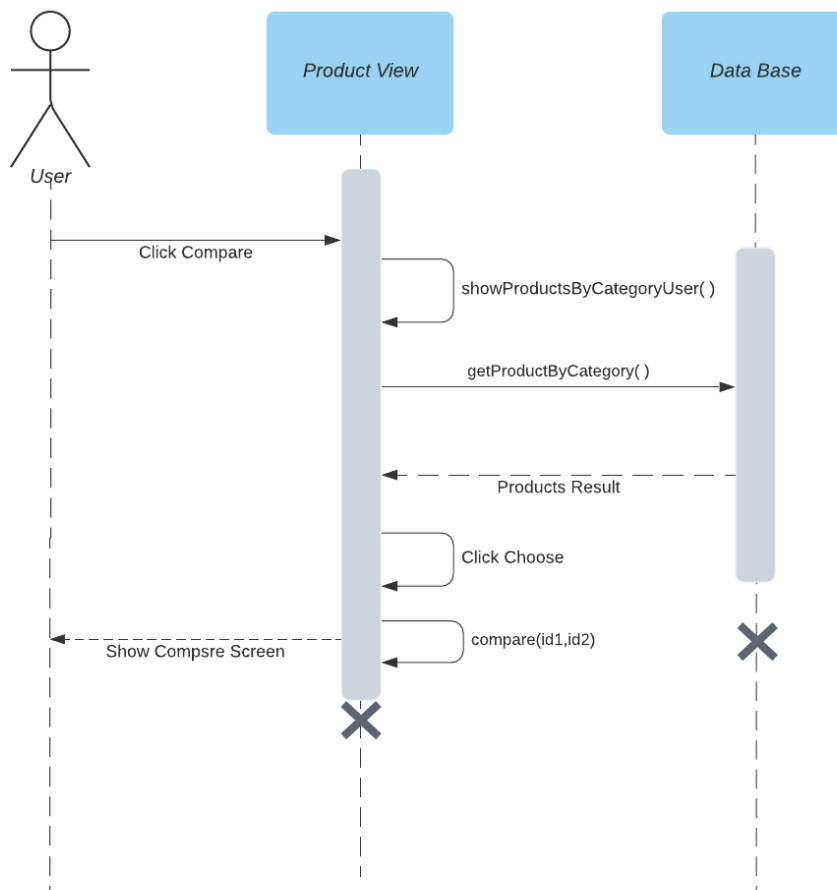
Edit User



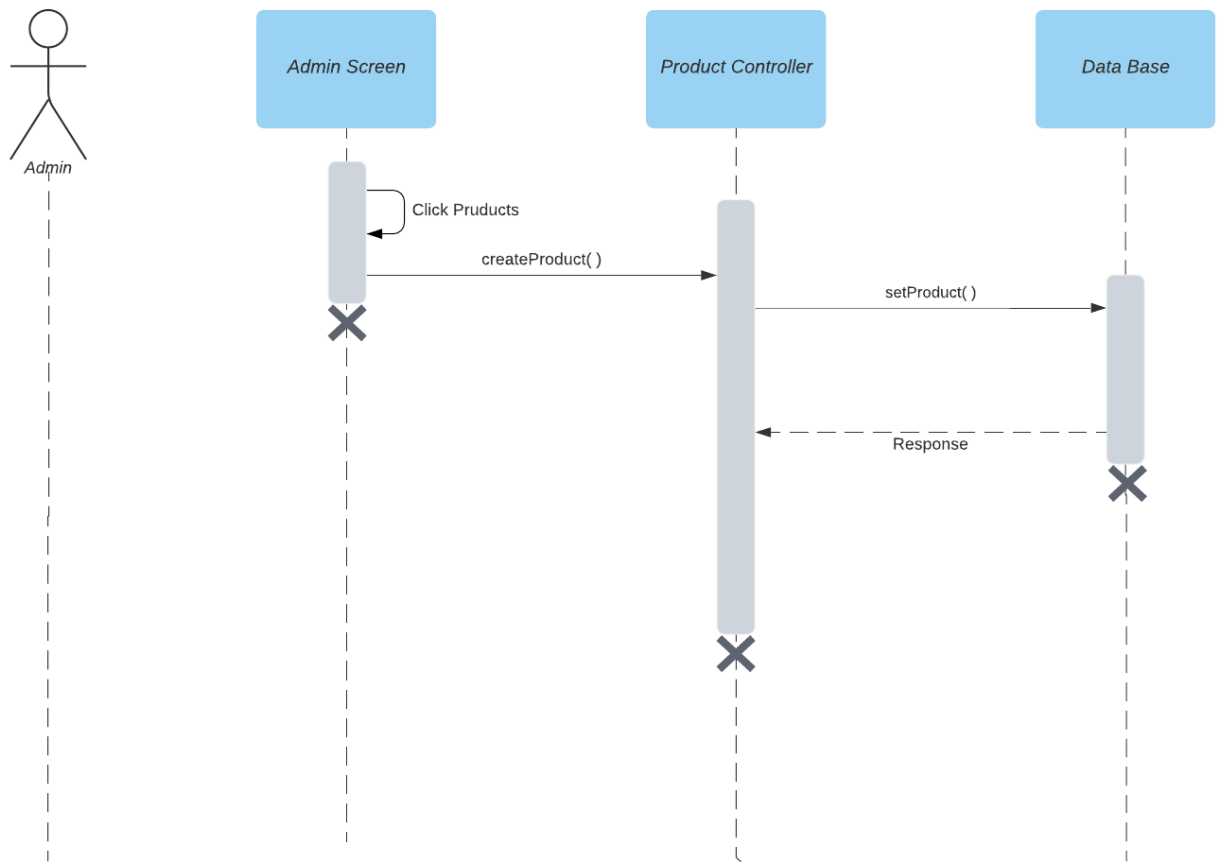
Give a Feedback



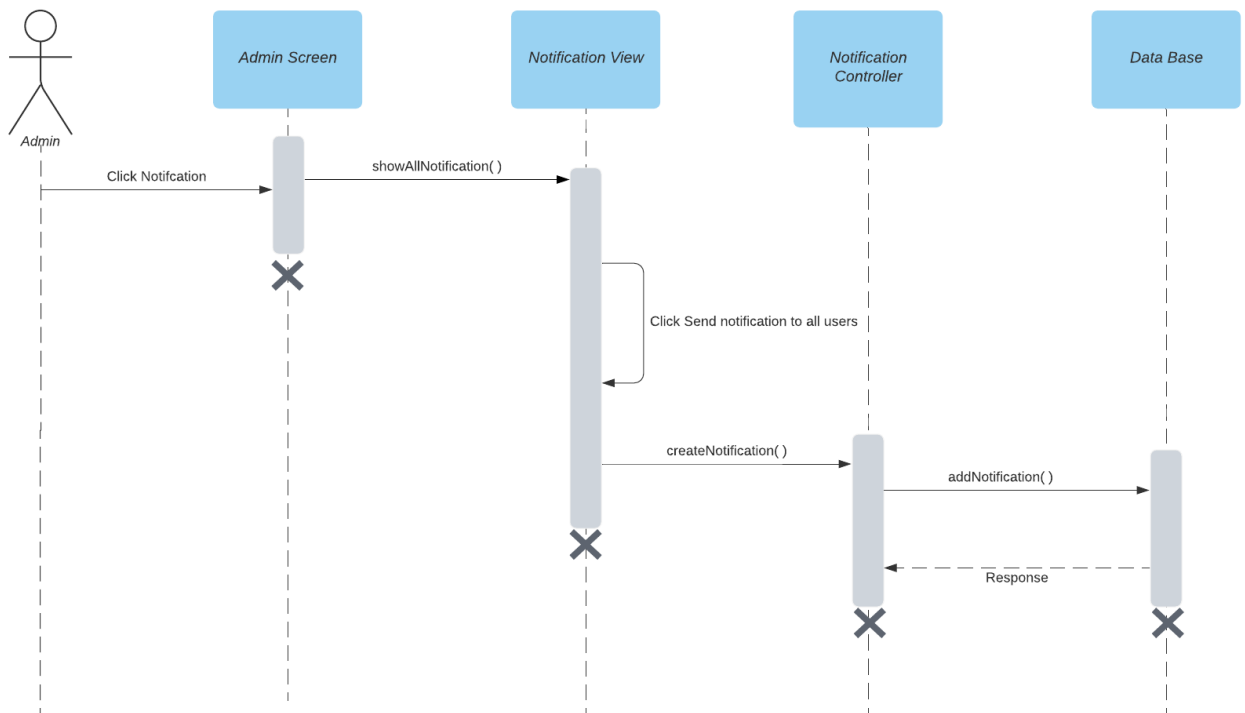
Compare



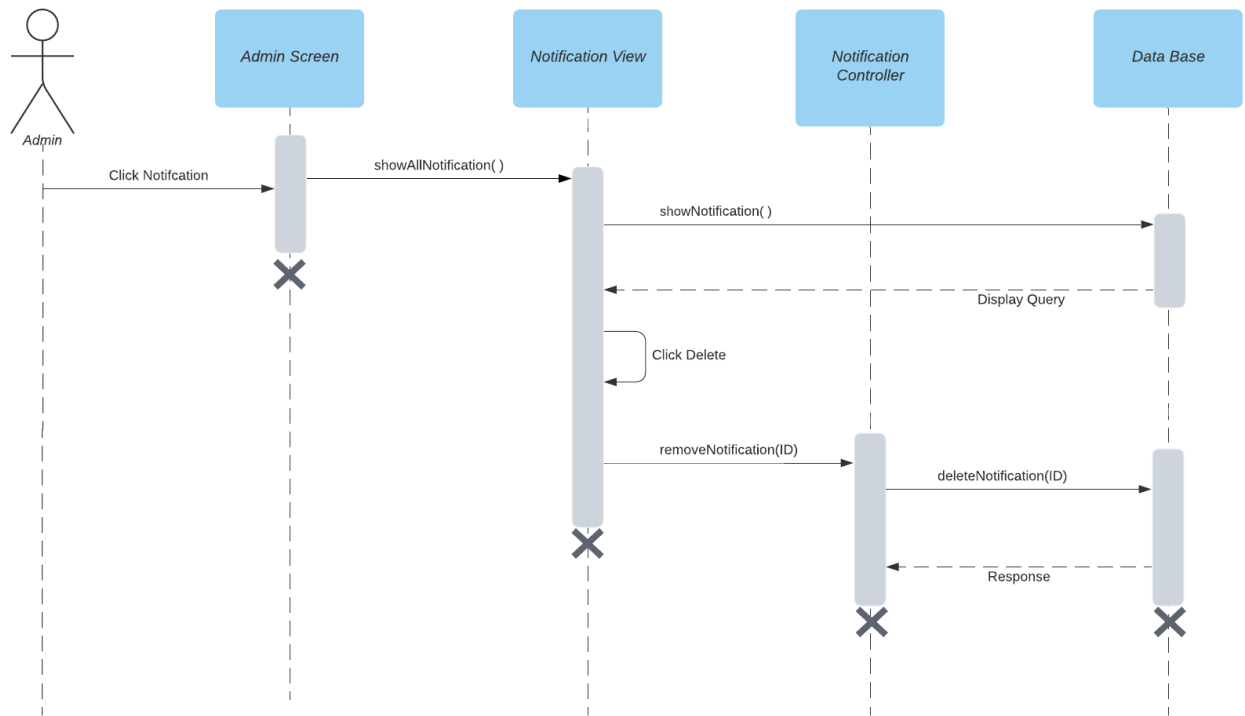
Add Product



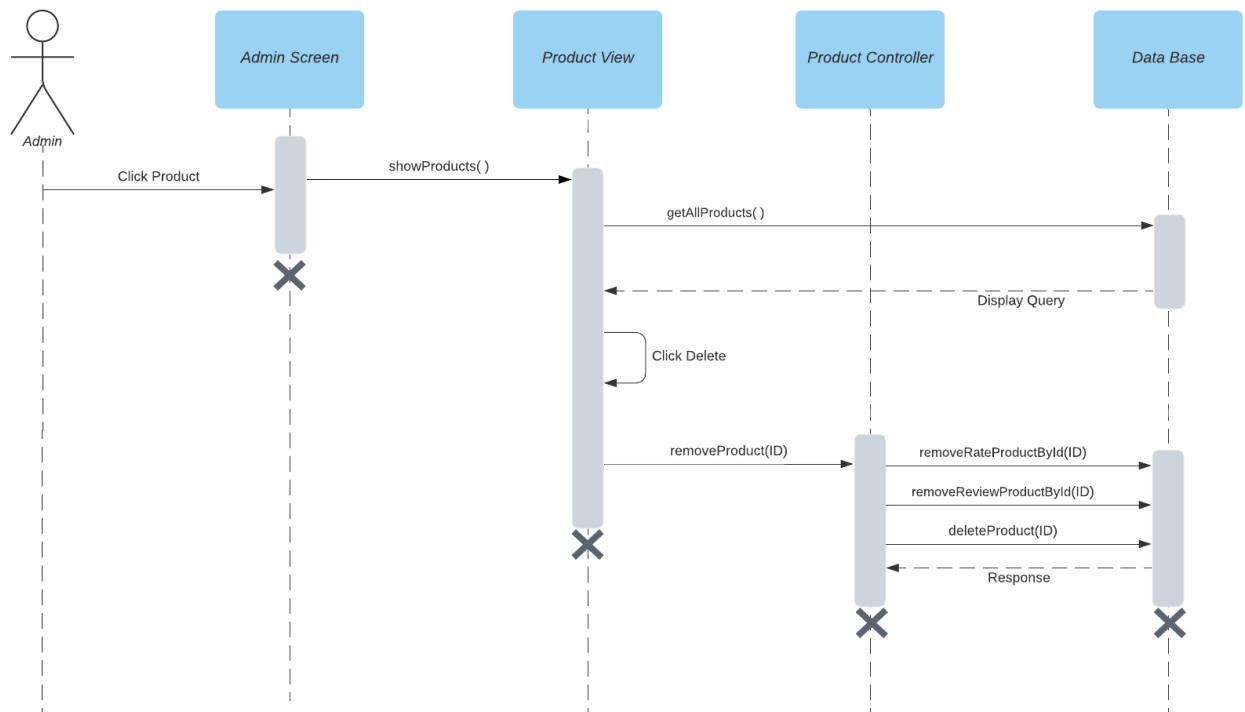
Send Notification



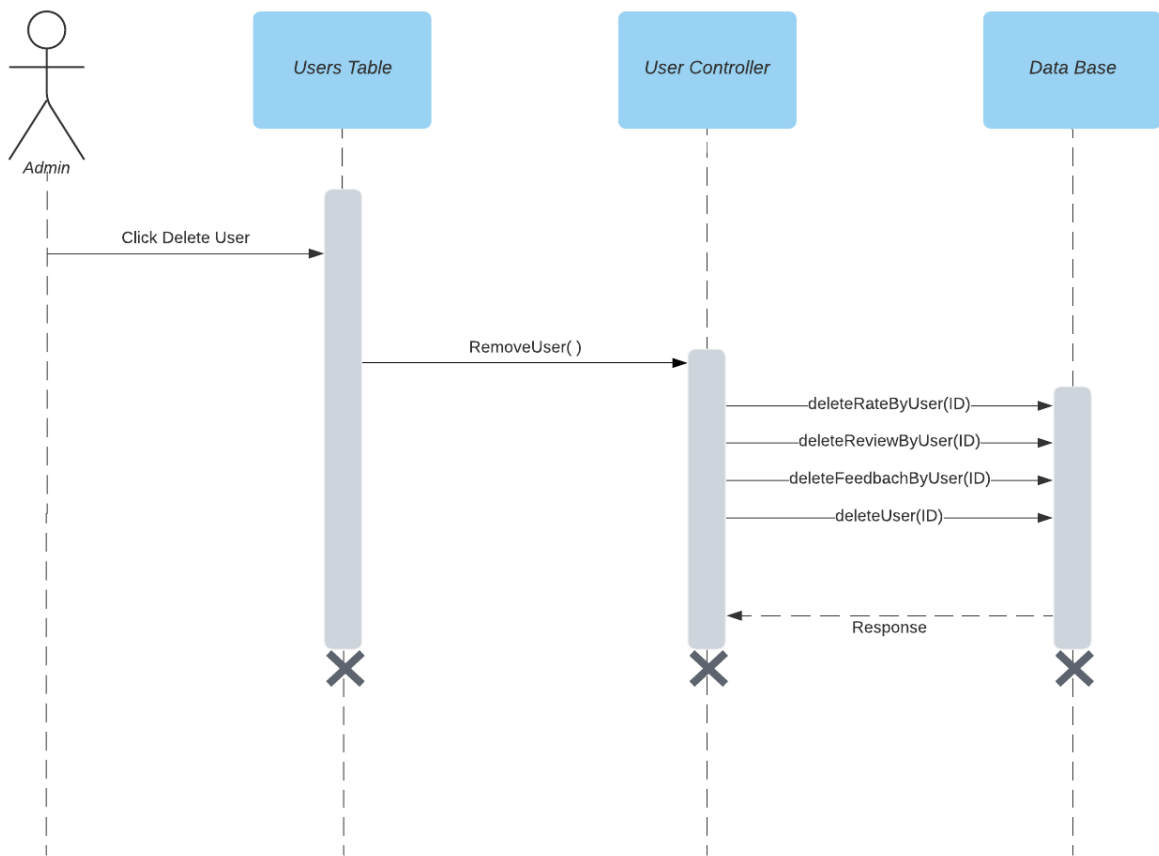
Remove Notification



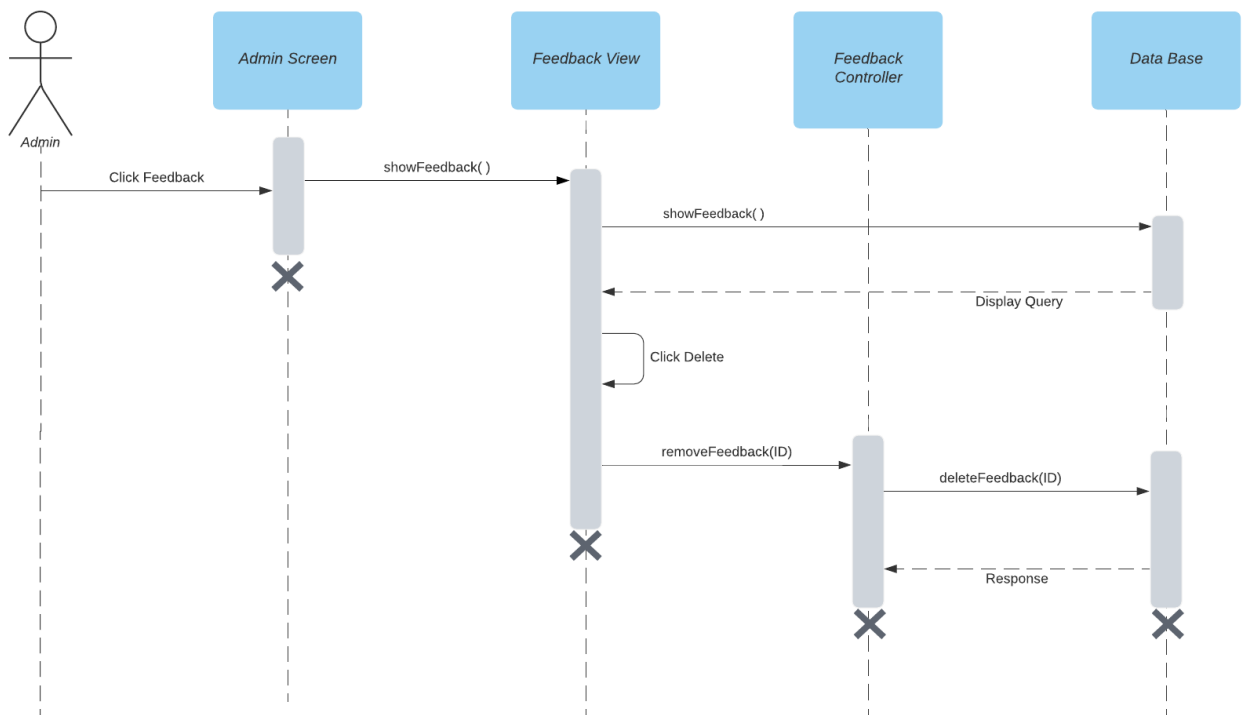
Remove Product



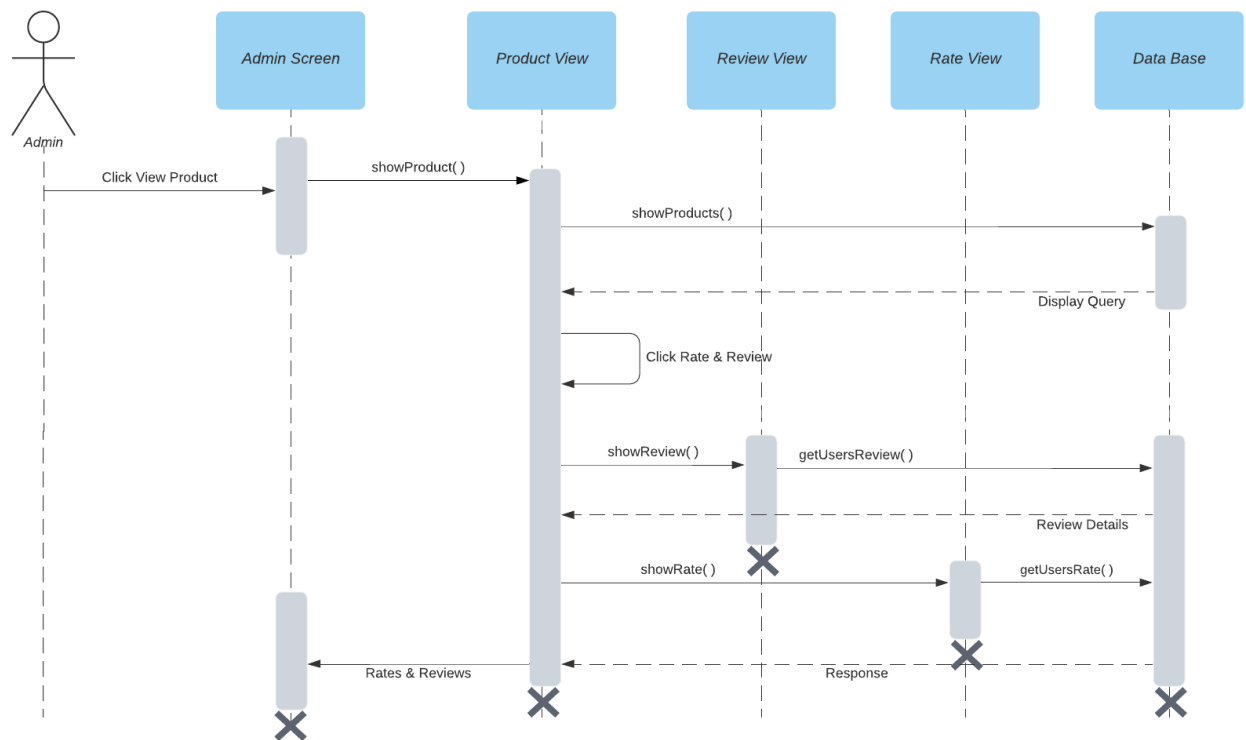
Delete a User



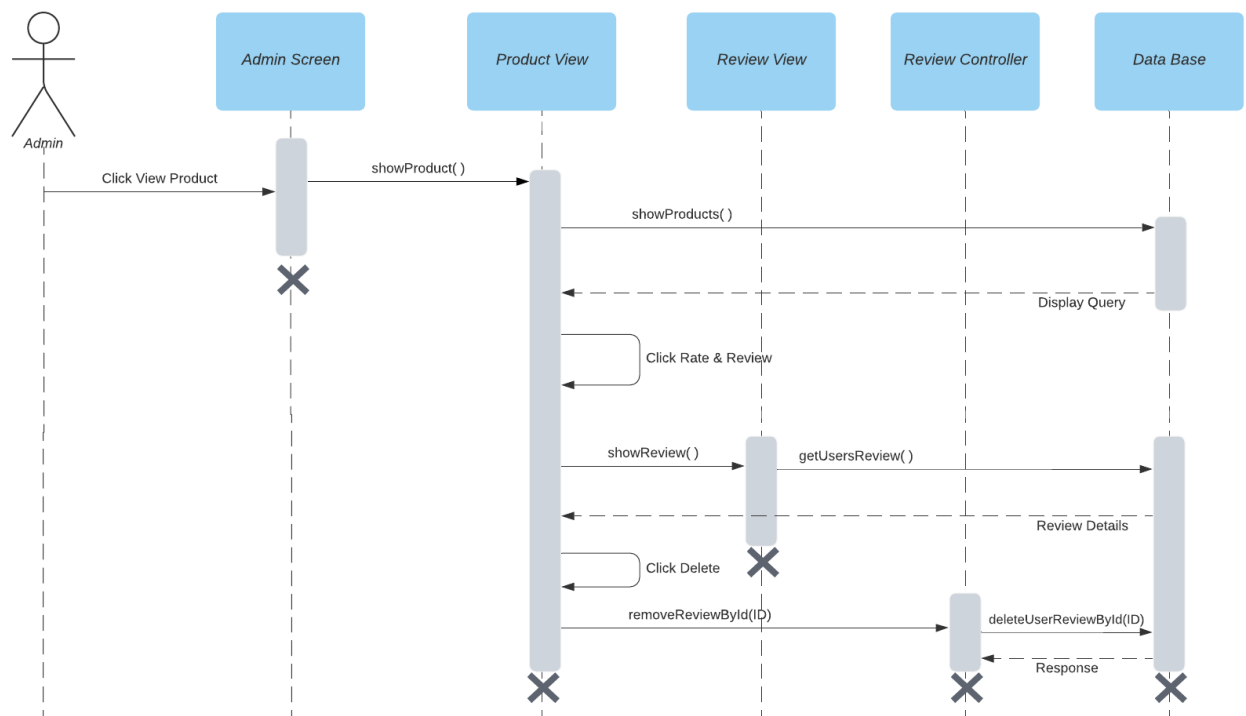
Remove Feedback



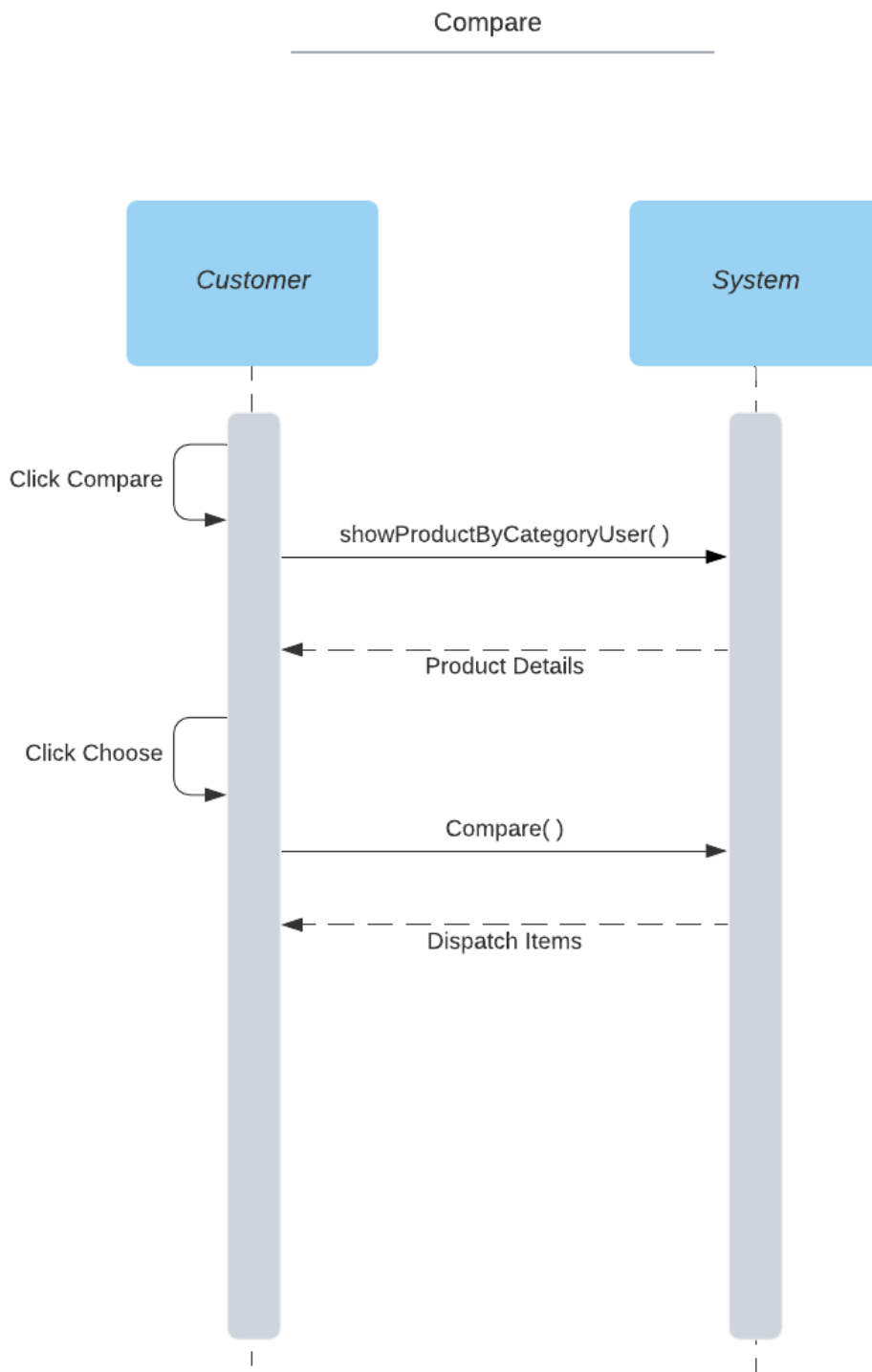
Show Rates & Reviews



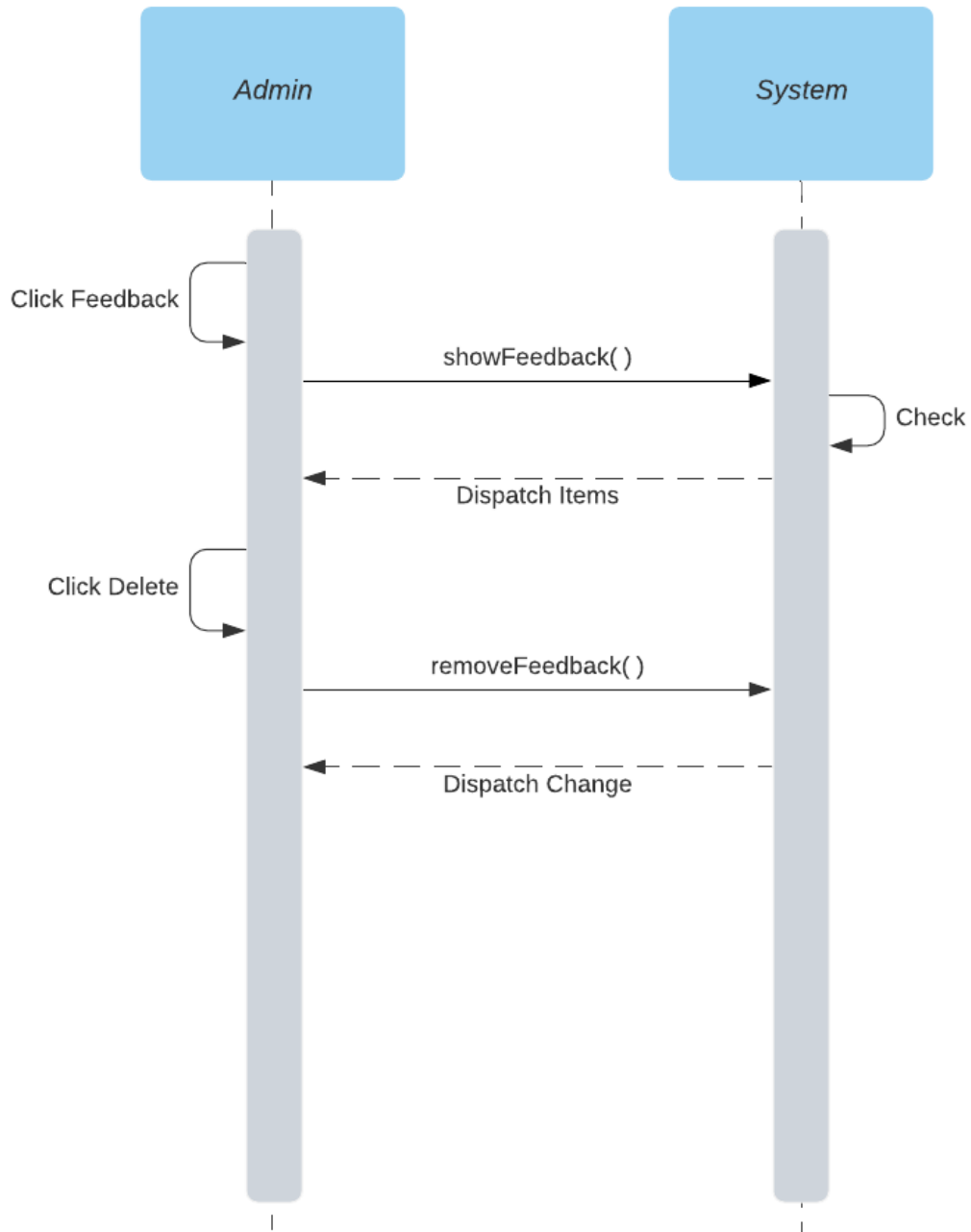
Delete a Review



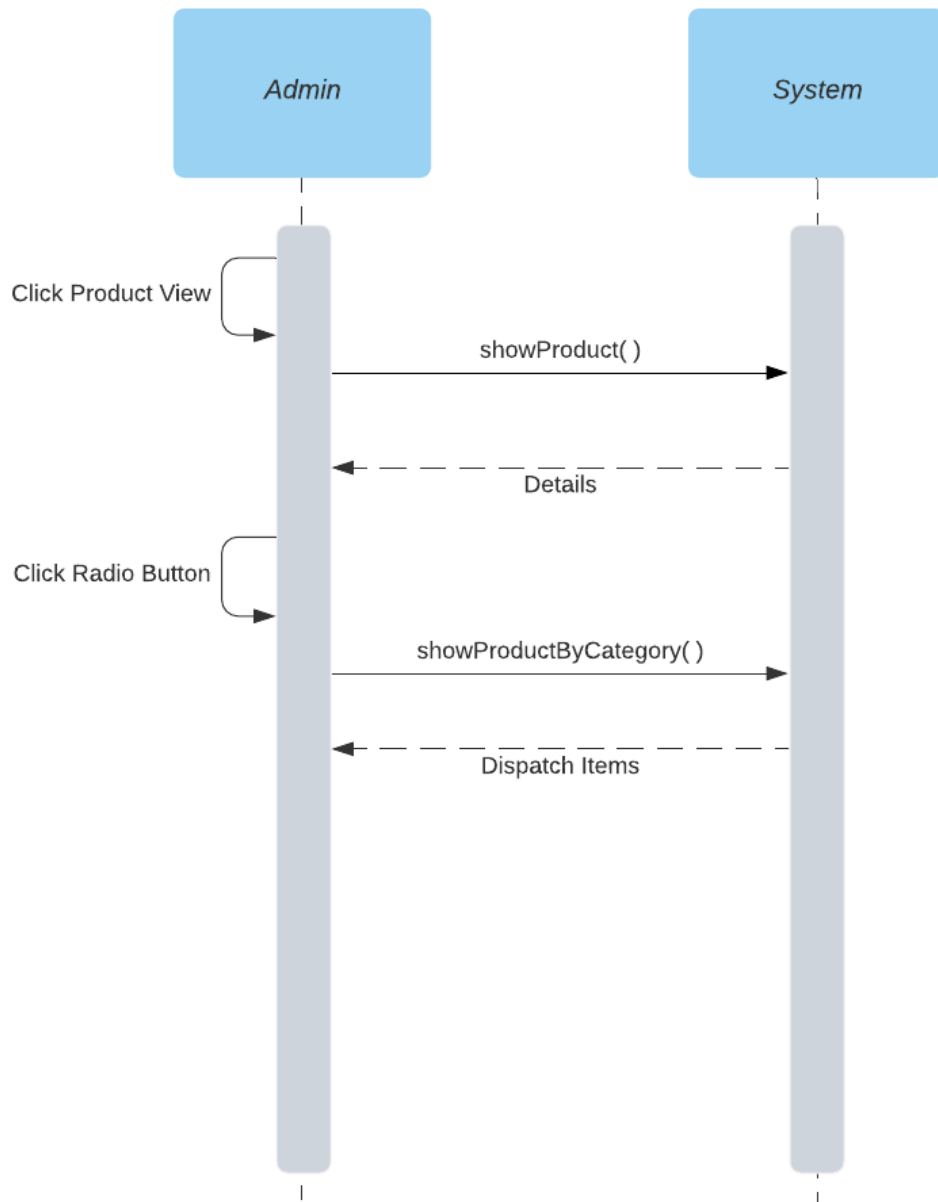
g) System Sequence Diagrams (SSDs)



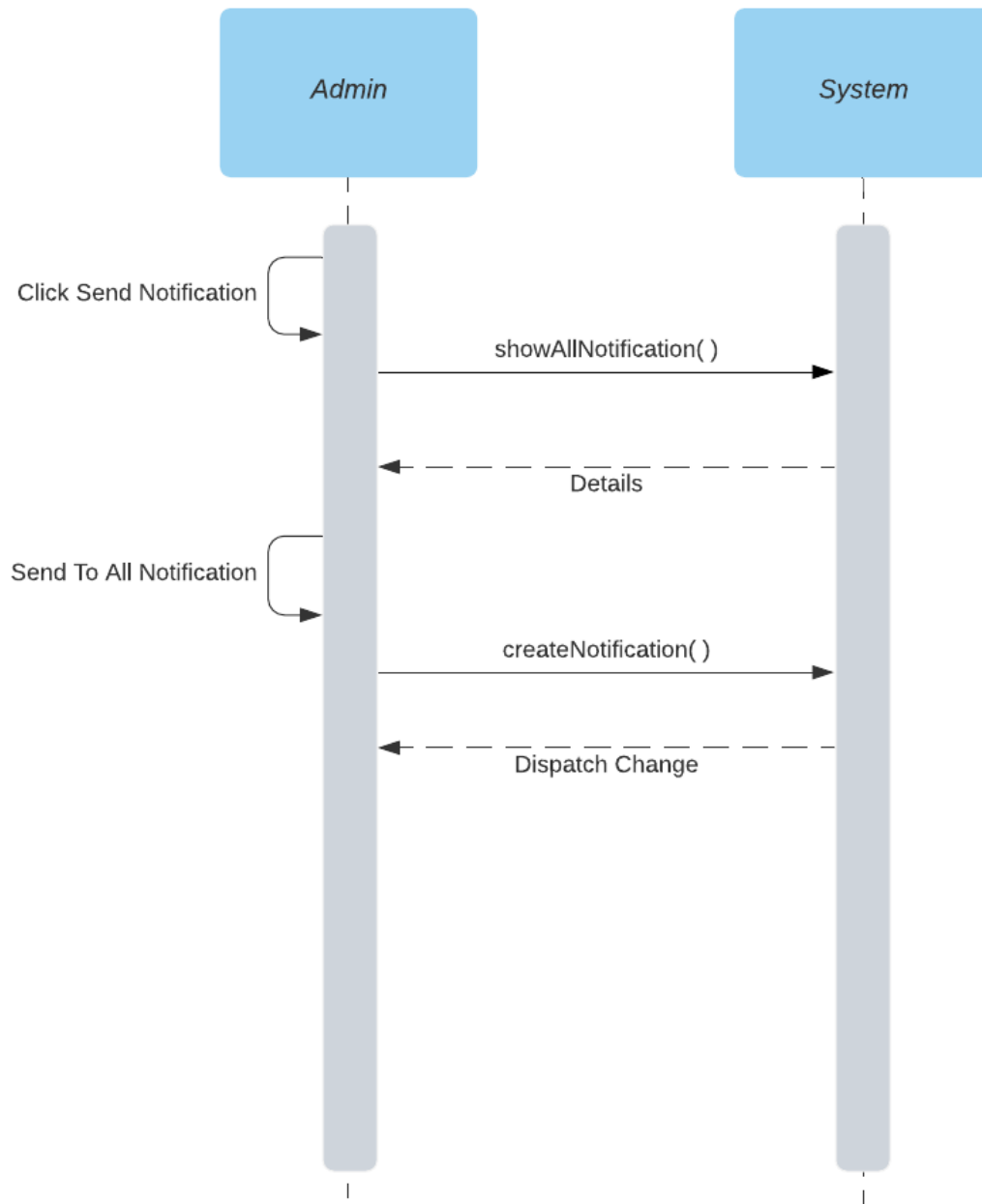
Remove Feedback



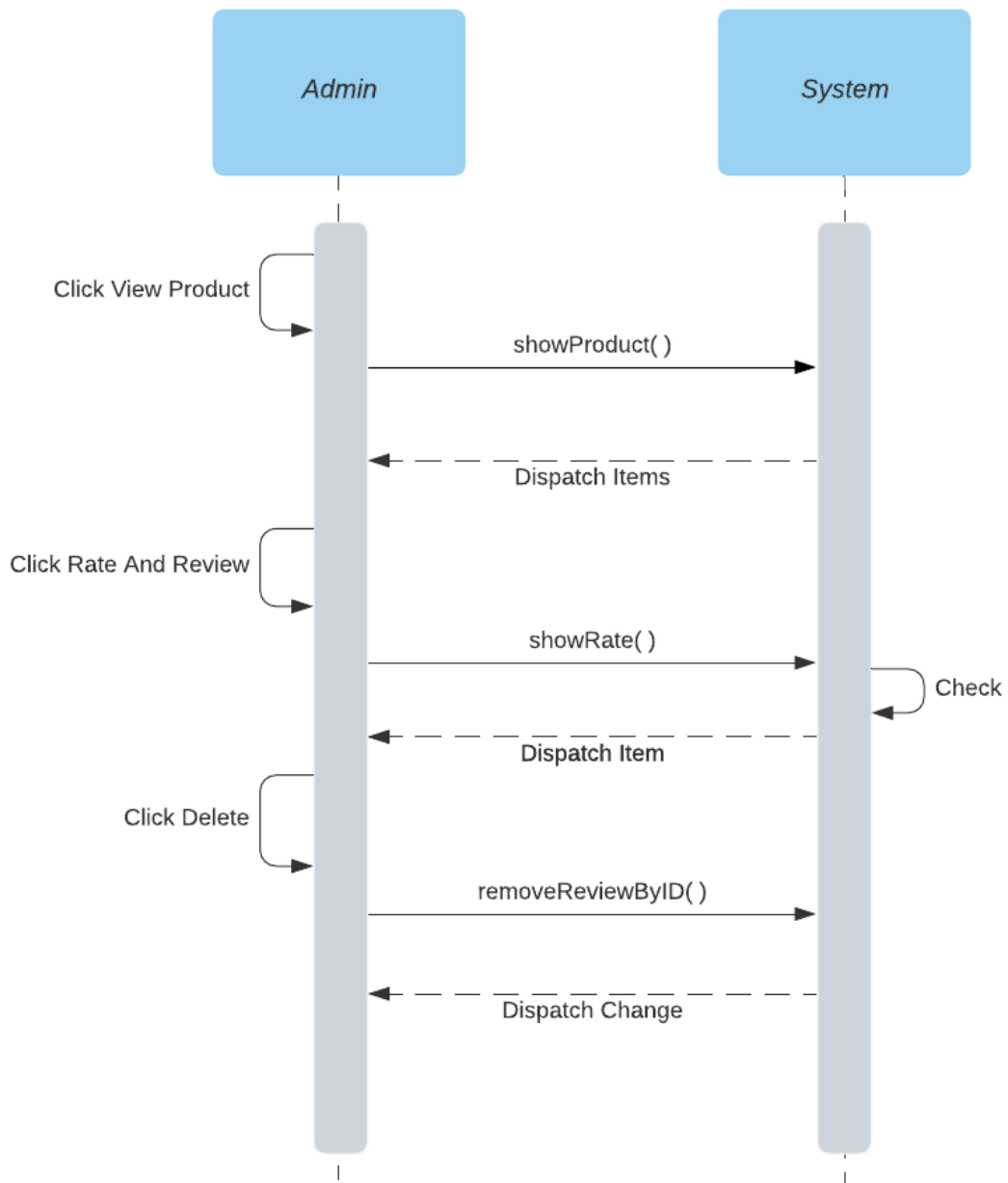
Show Product In One Category



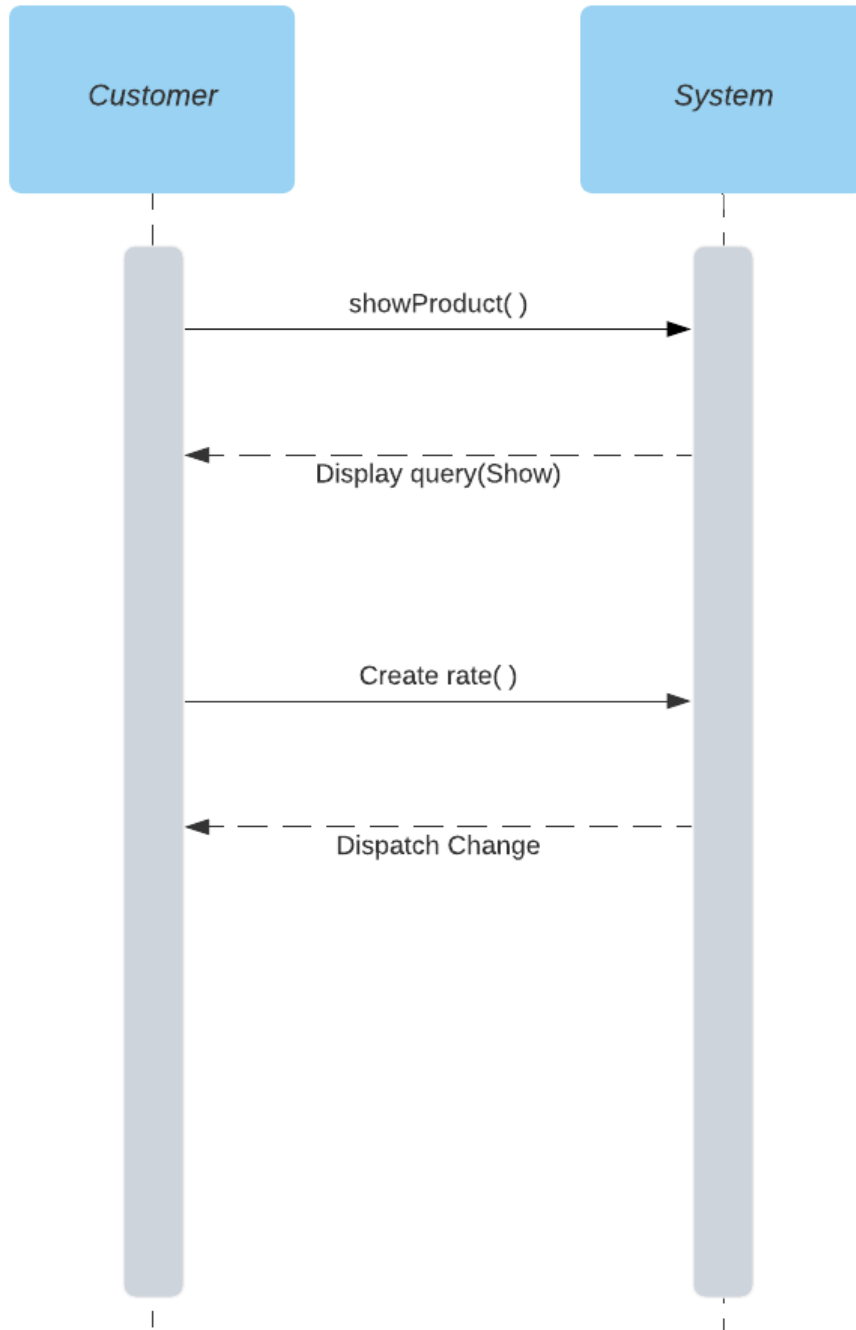
Send Notification



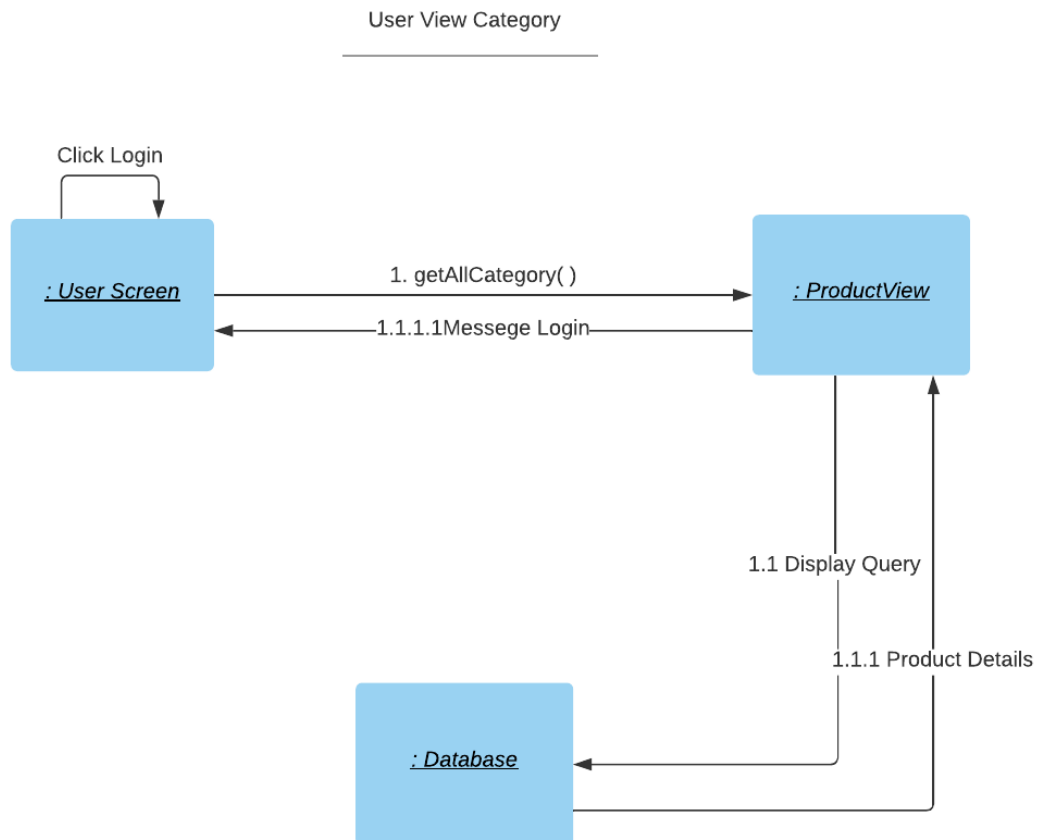
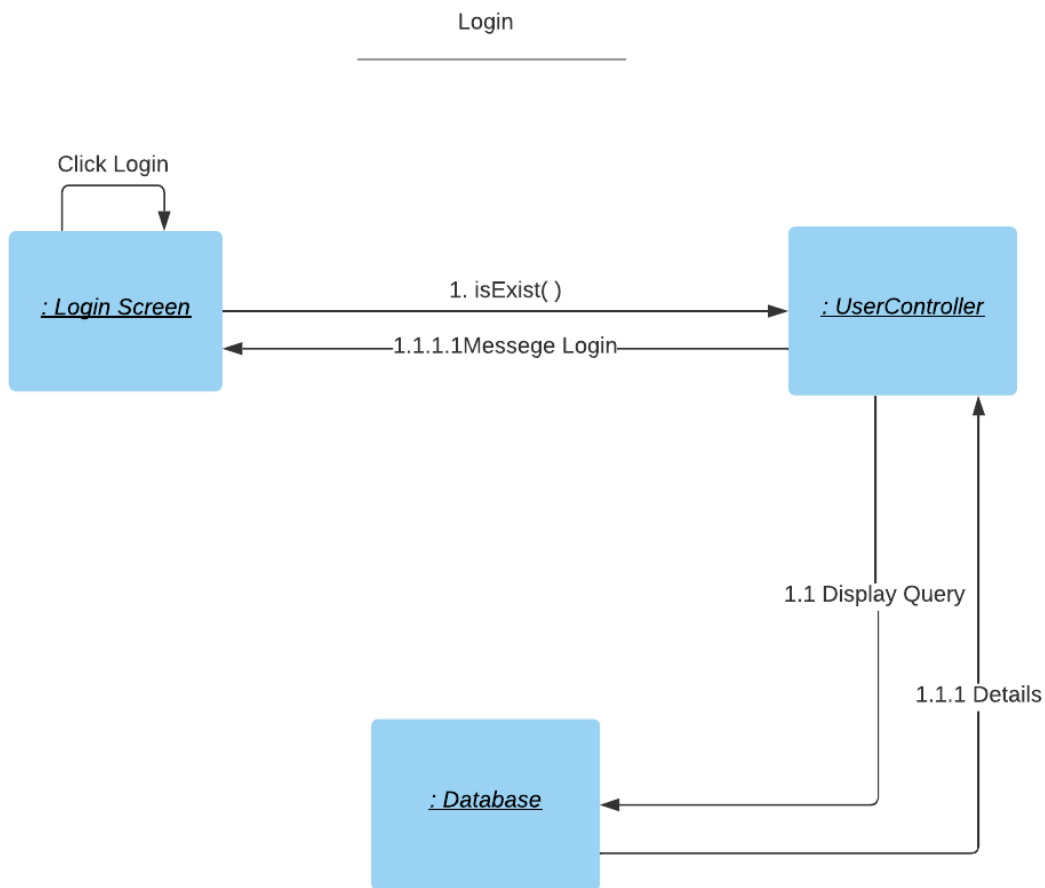
Remove Review



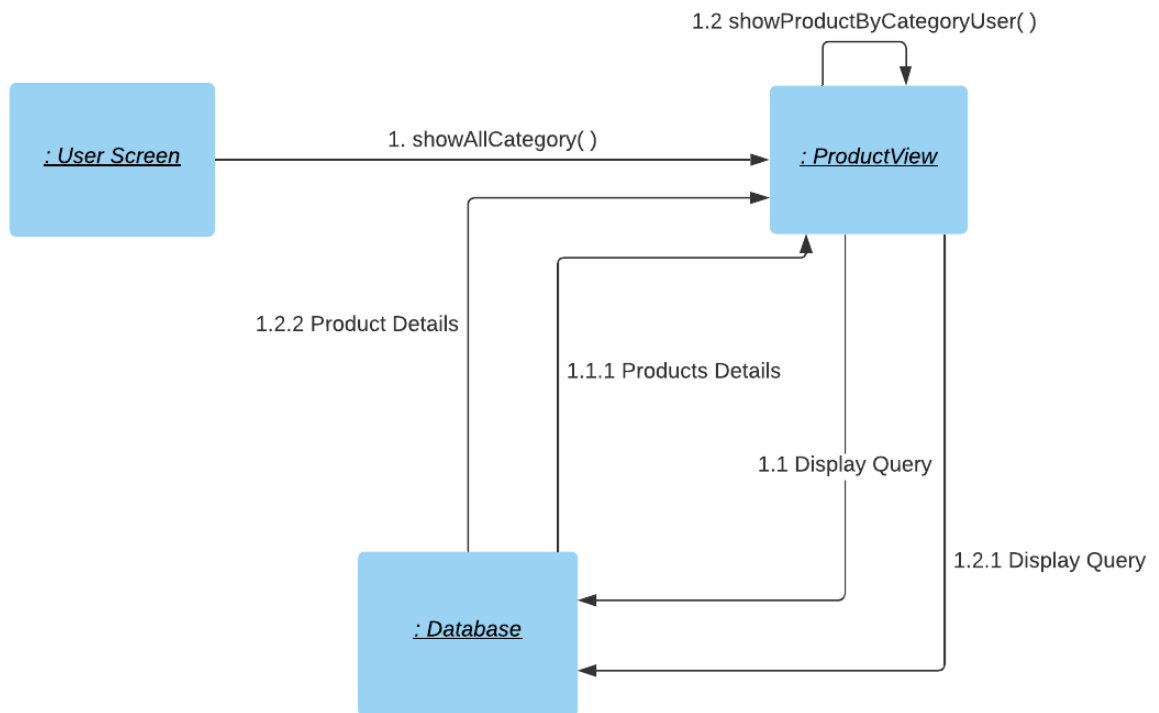
Create Rate



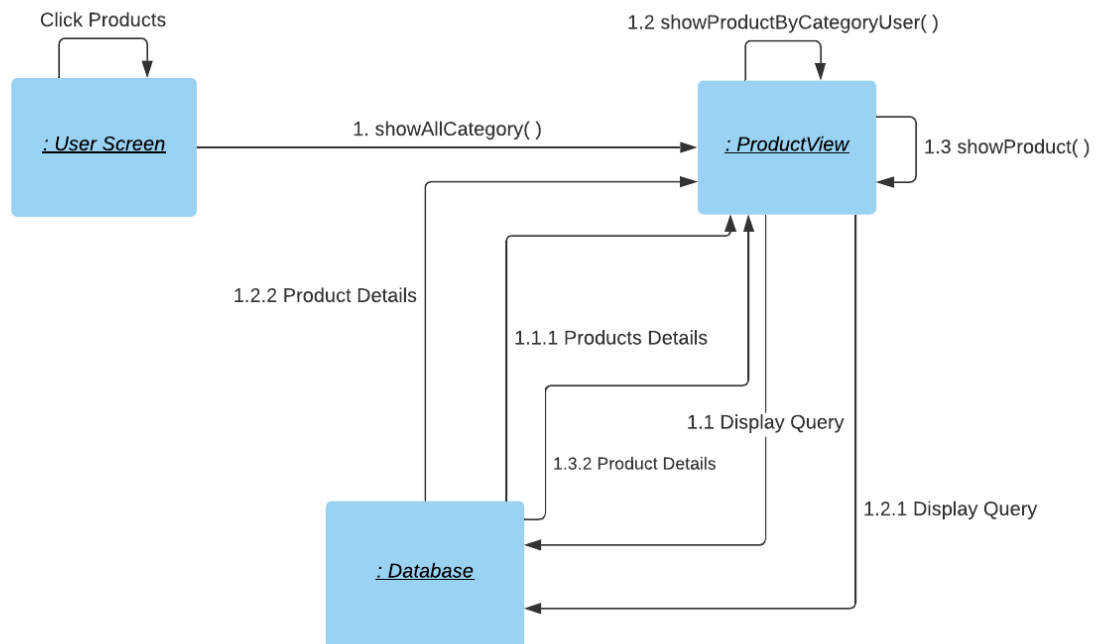
h) Collaboration/Communication Diagram(s)



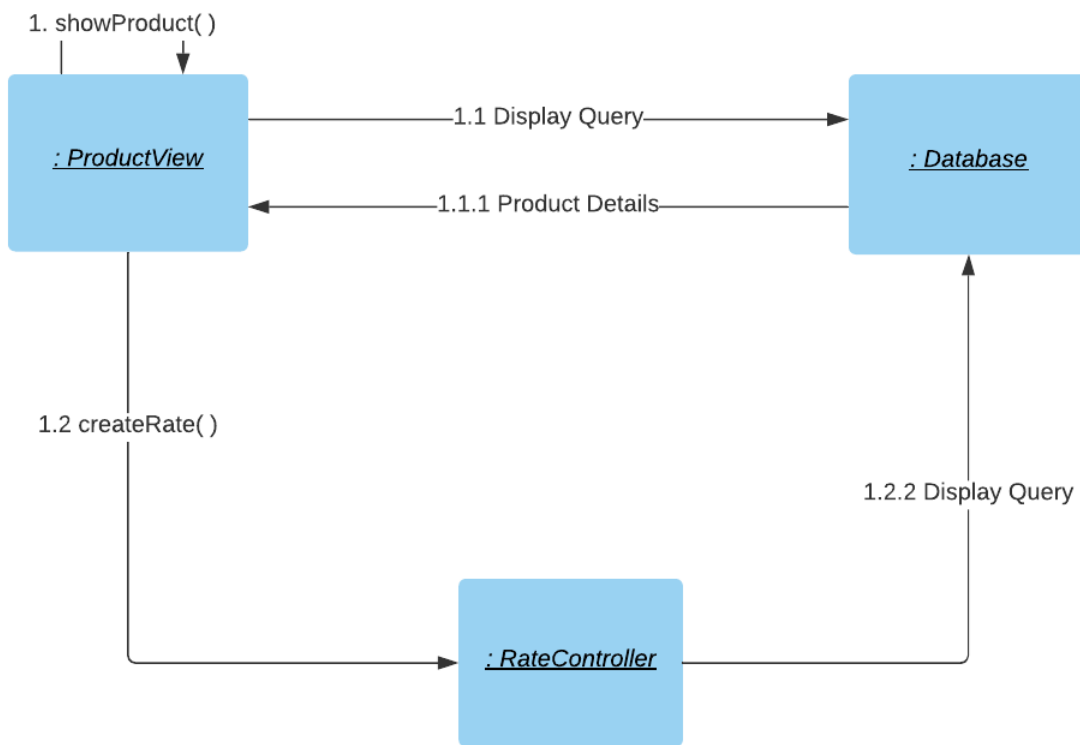
Show one Category From All Categories



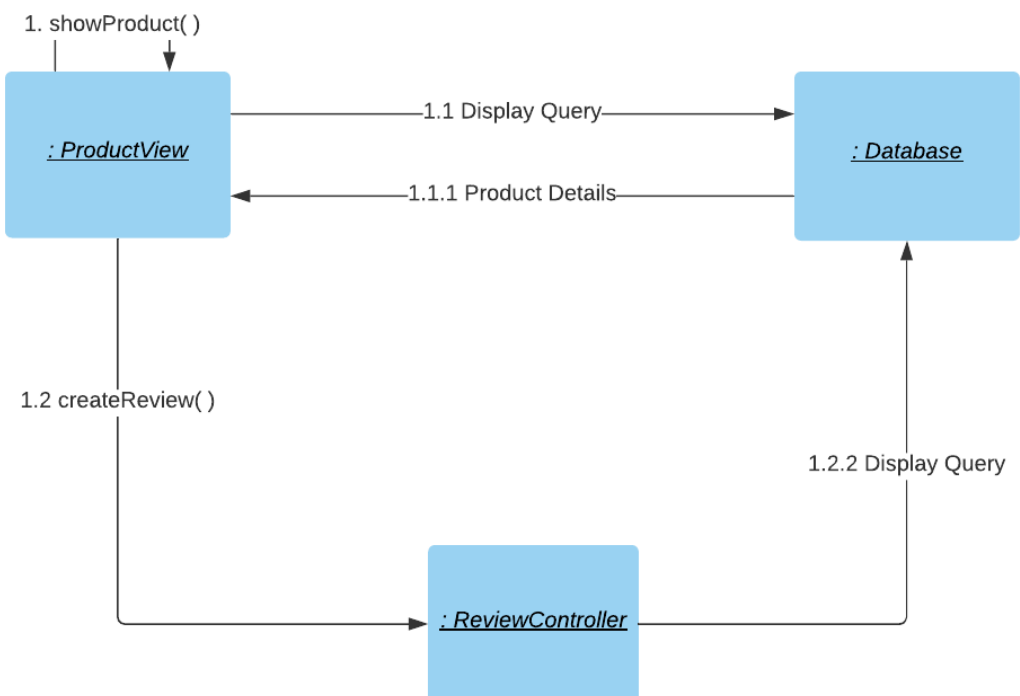
Show Product By Category User



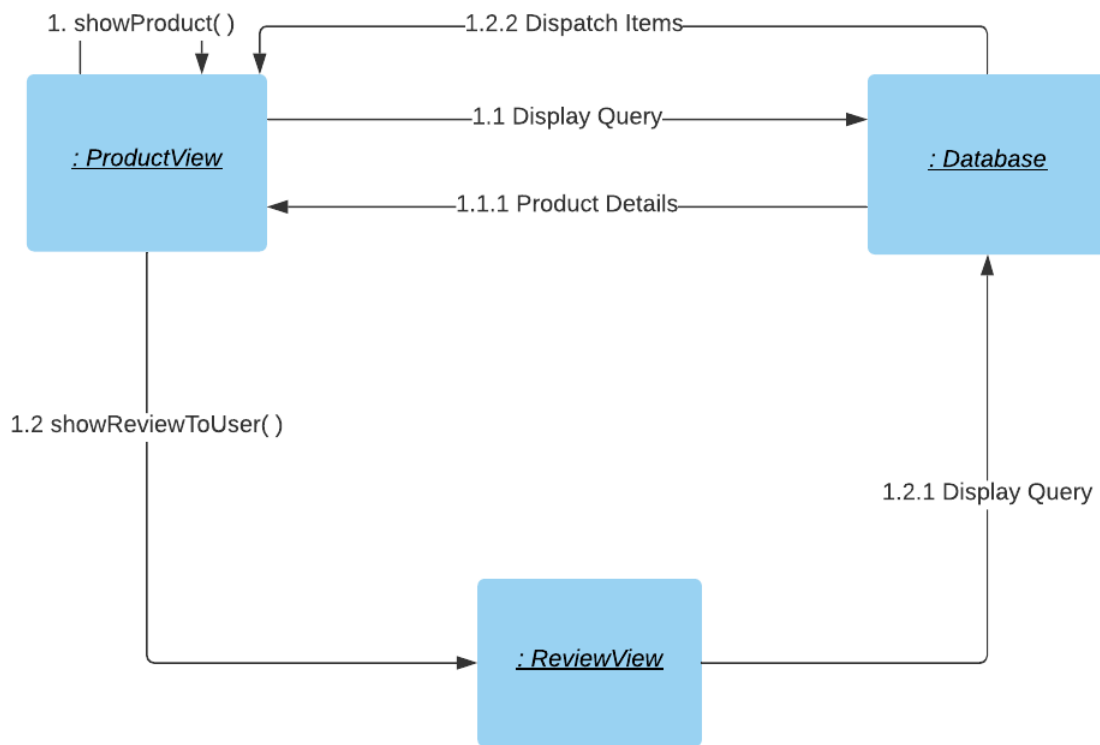
Create Rate



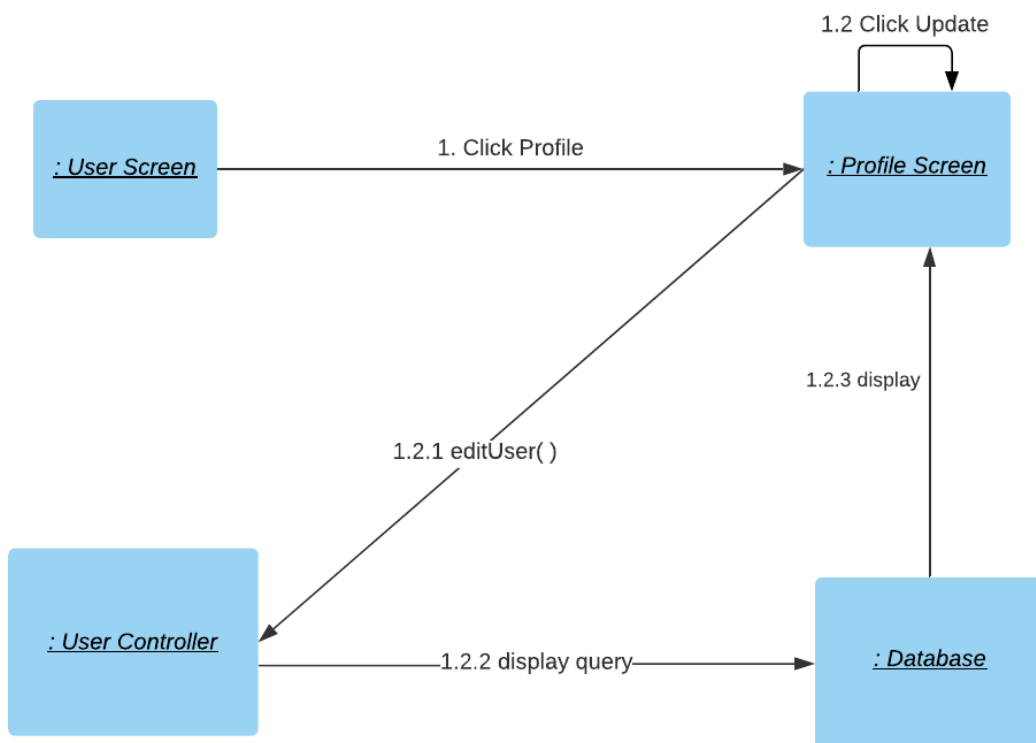
Create Review



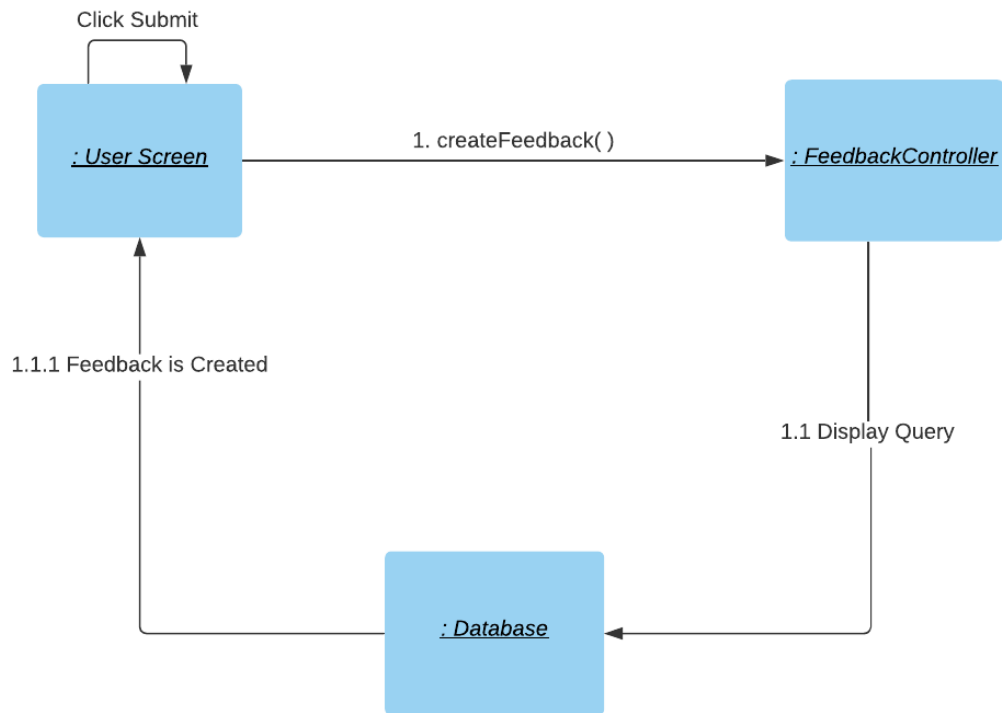
Show Review



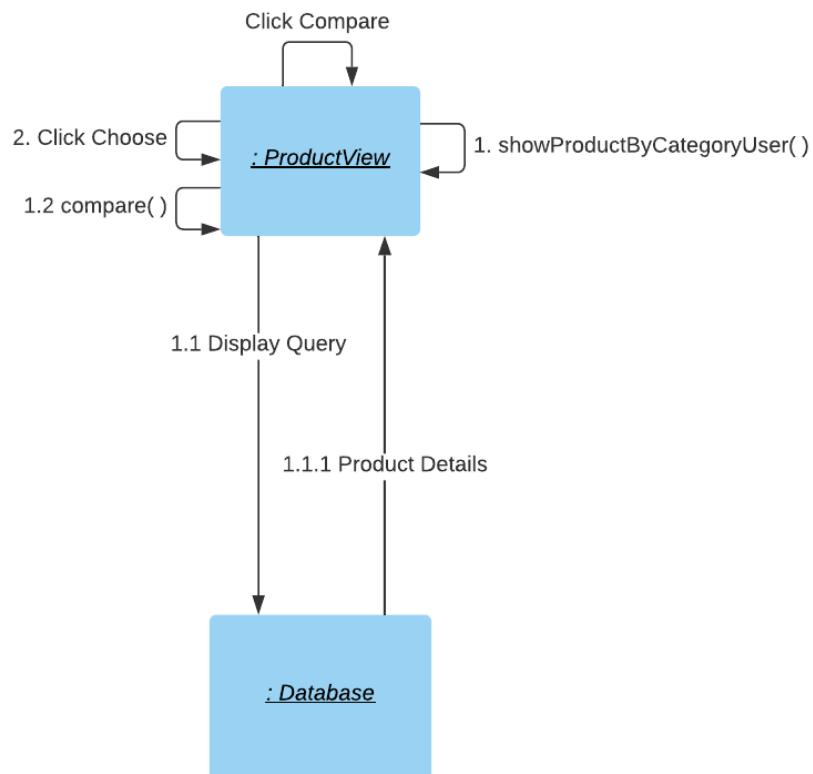
Update User



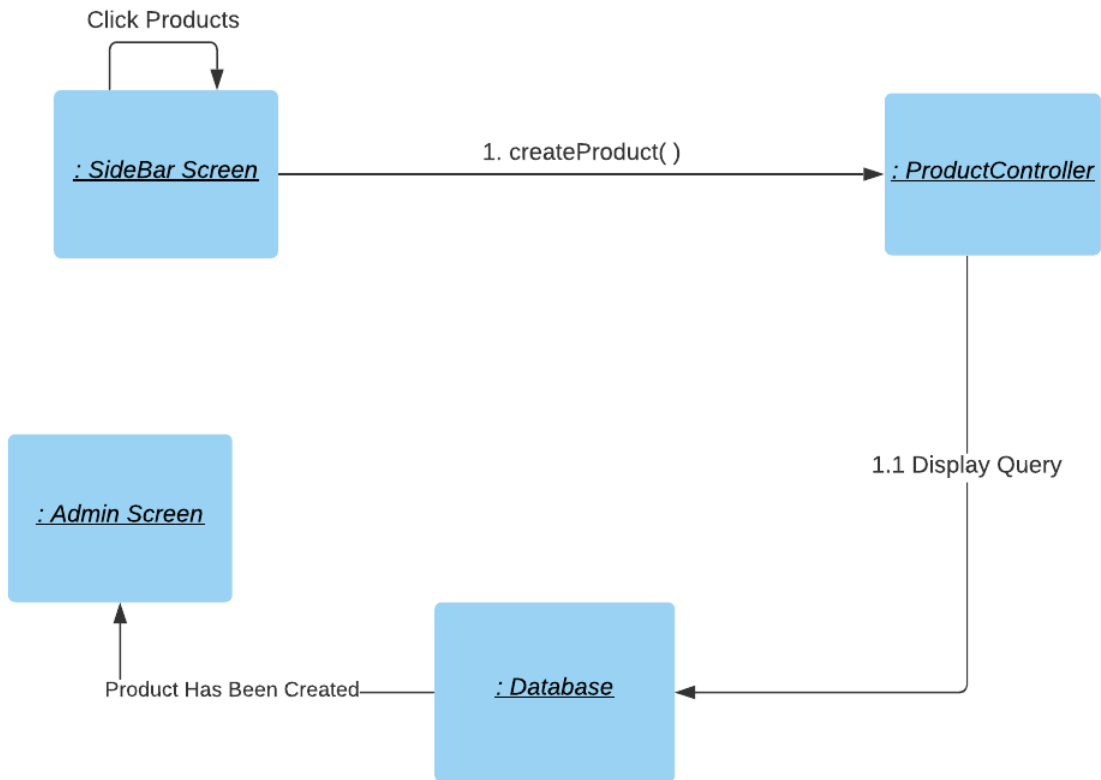
Create Feedback



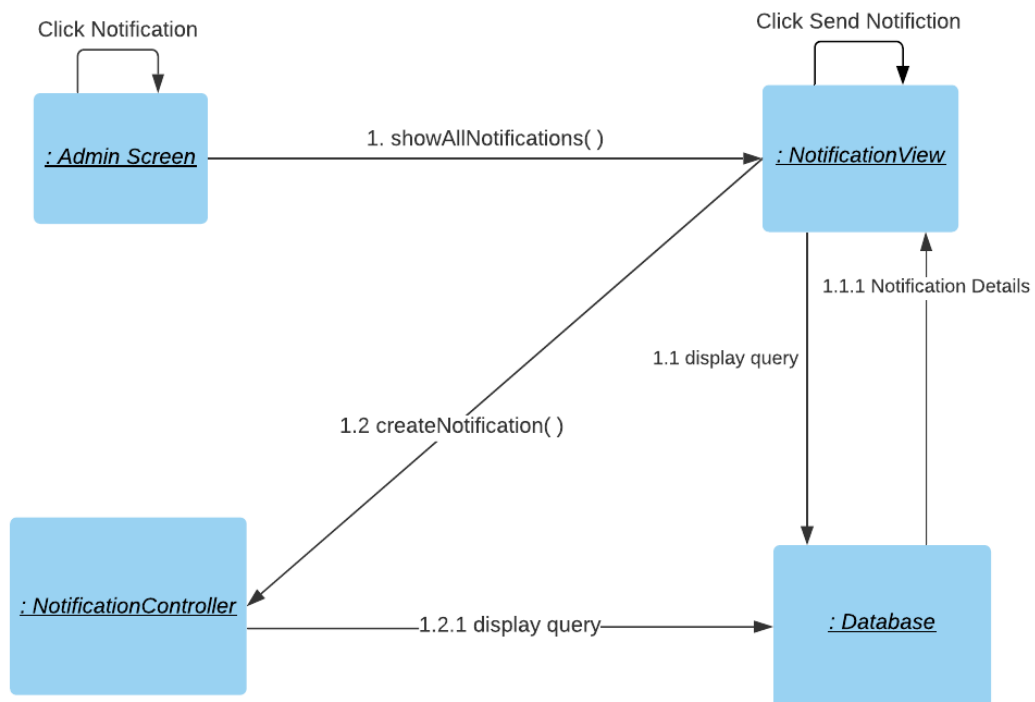
Compare



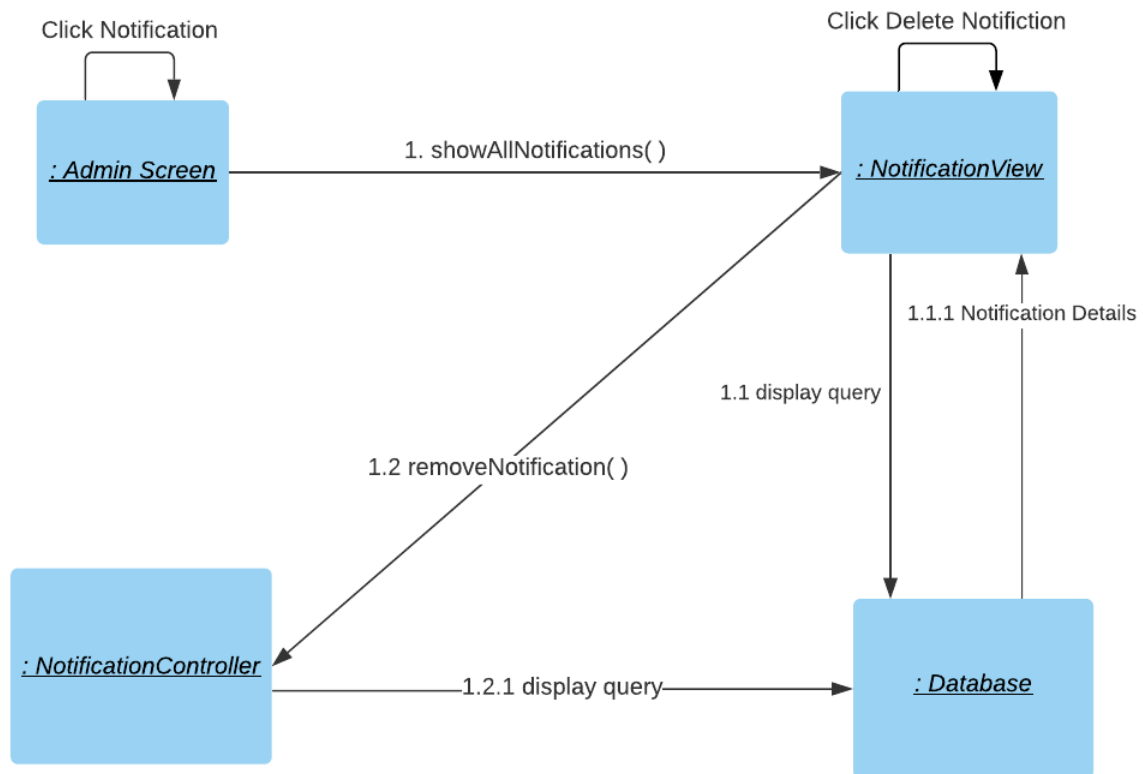
Create Product



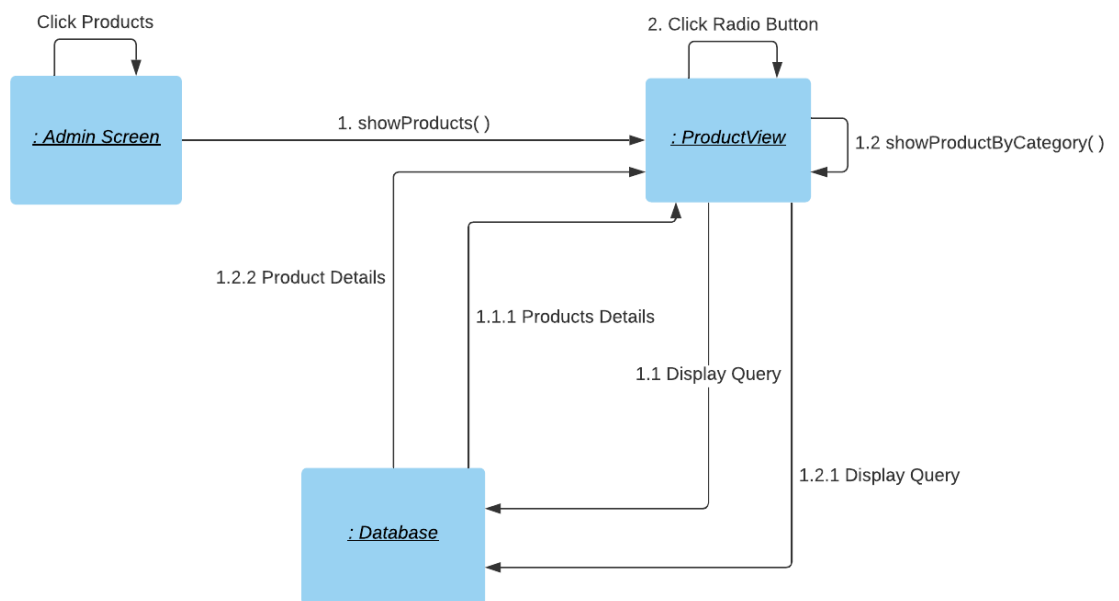
Create Notification



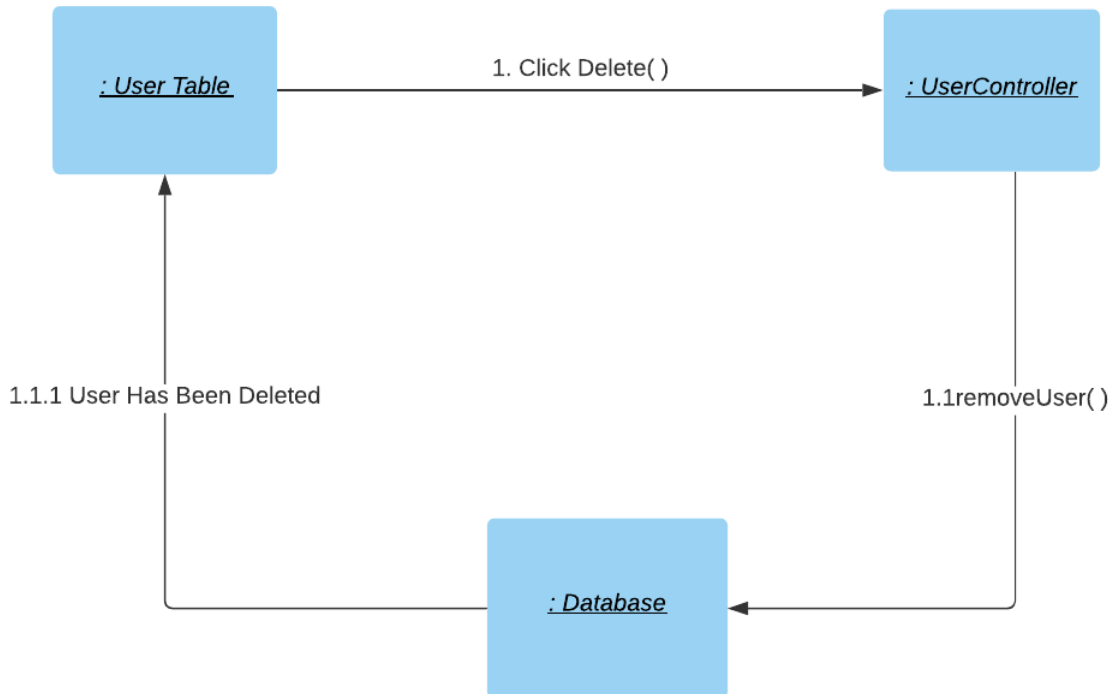
Delete Notification



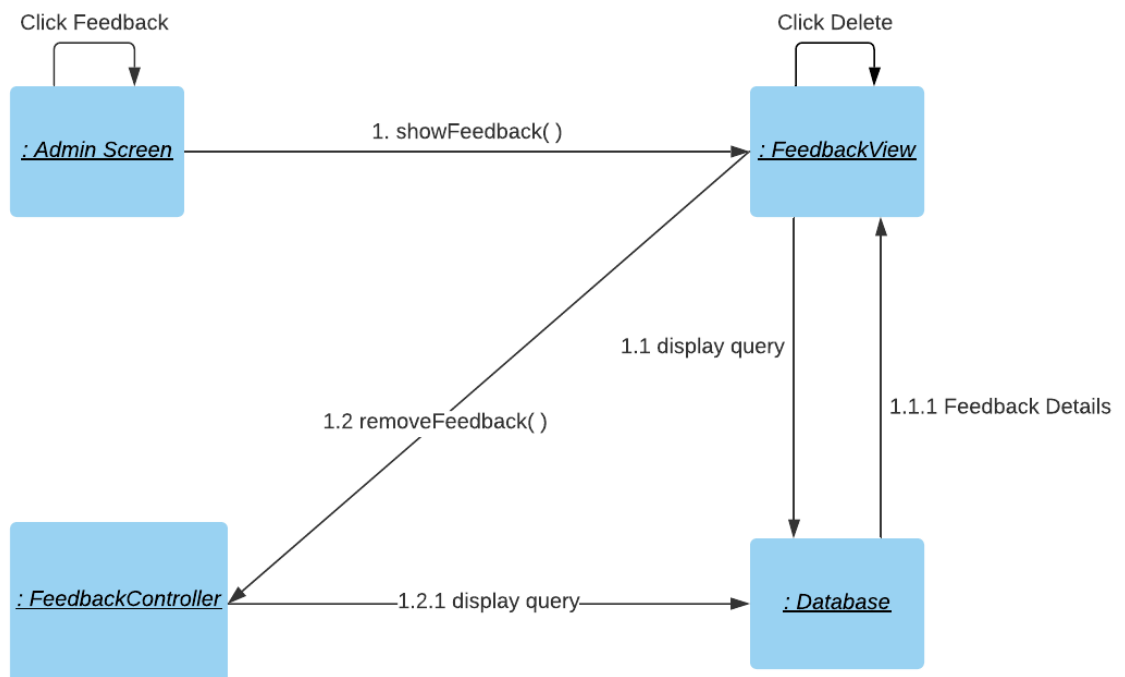
Show Product By Category



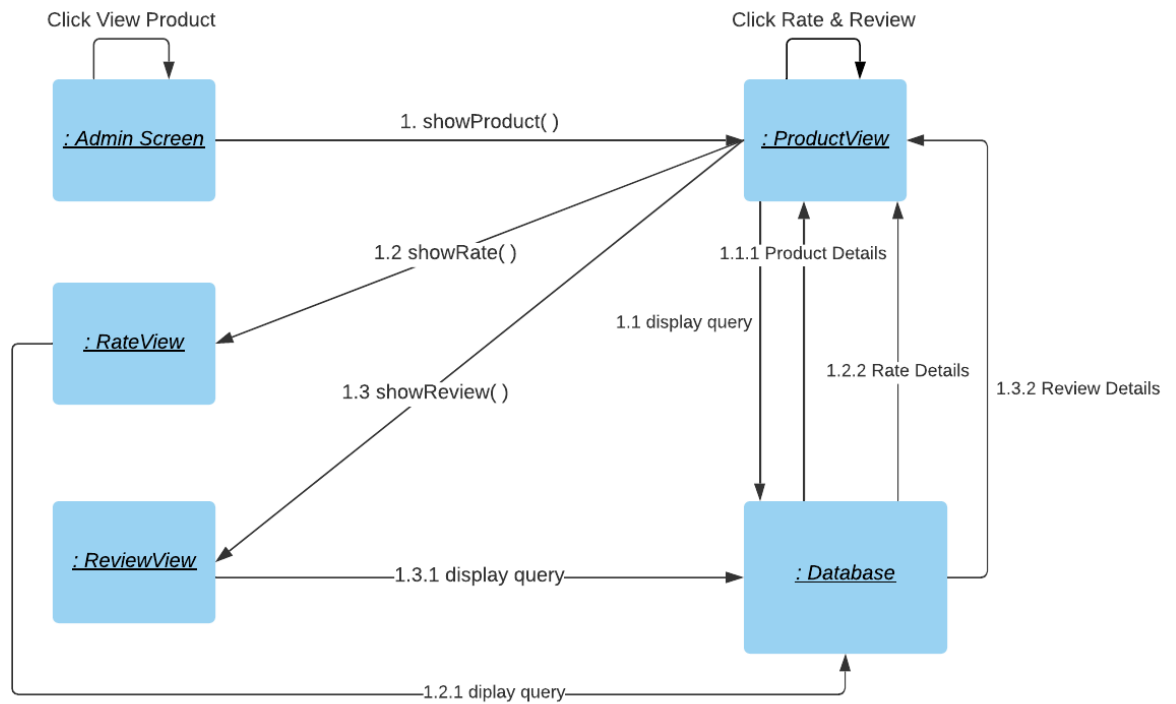
Delete User



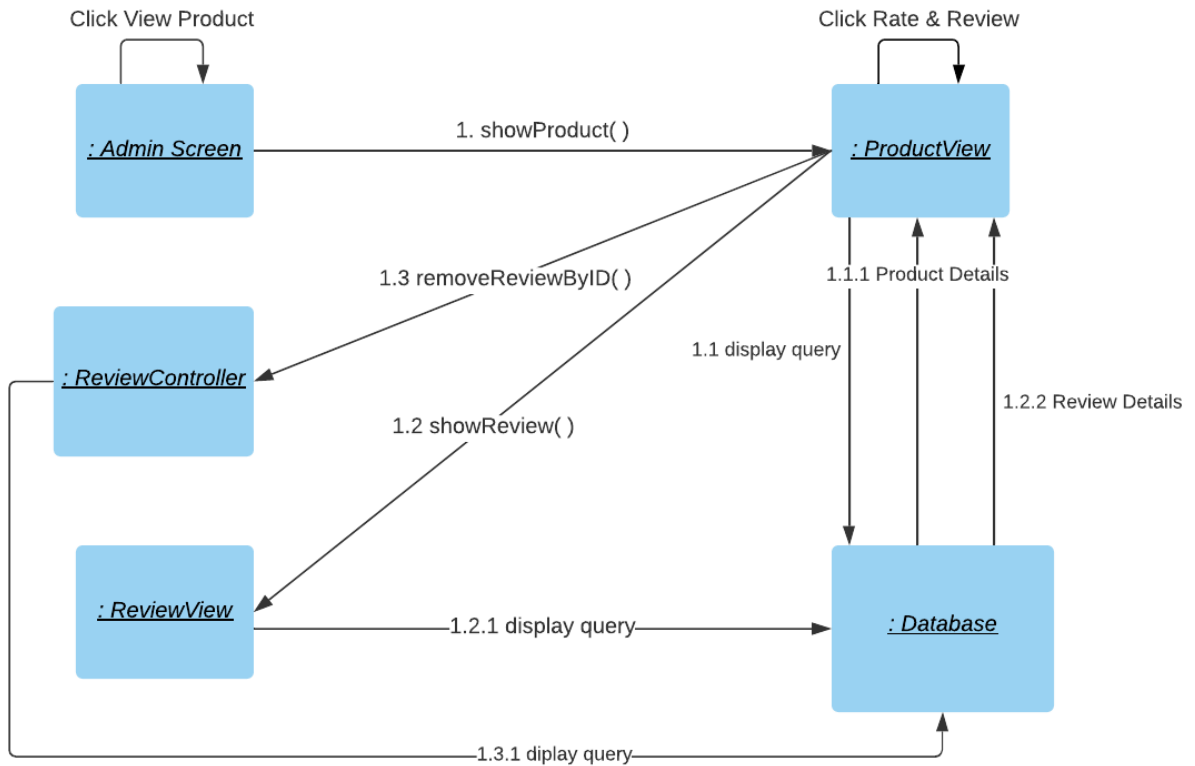
Remove Feedback



View Rate & Review



Delete Review



i) Which strategy (or strategies) did you use to implement the use-cases:

Actor Class: the message should be sent to the software object corresponding to the real-world actor object who initiated the operation. (use actors as classes)

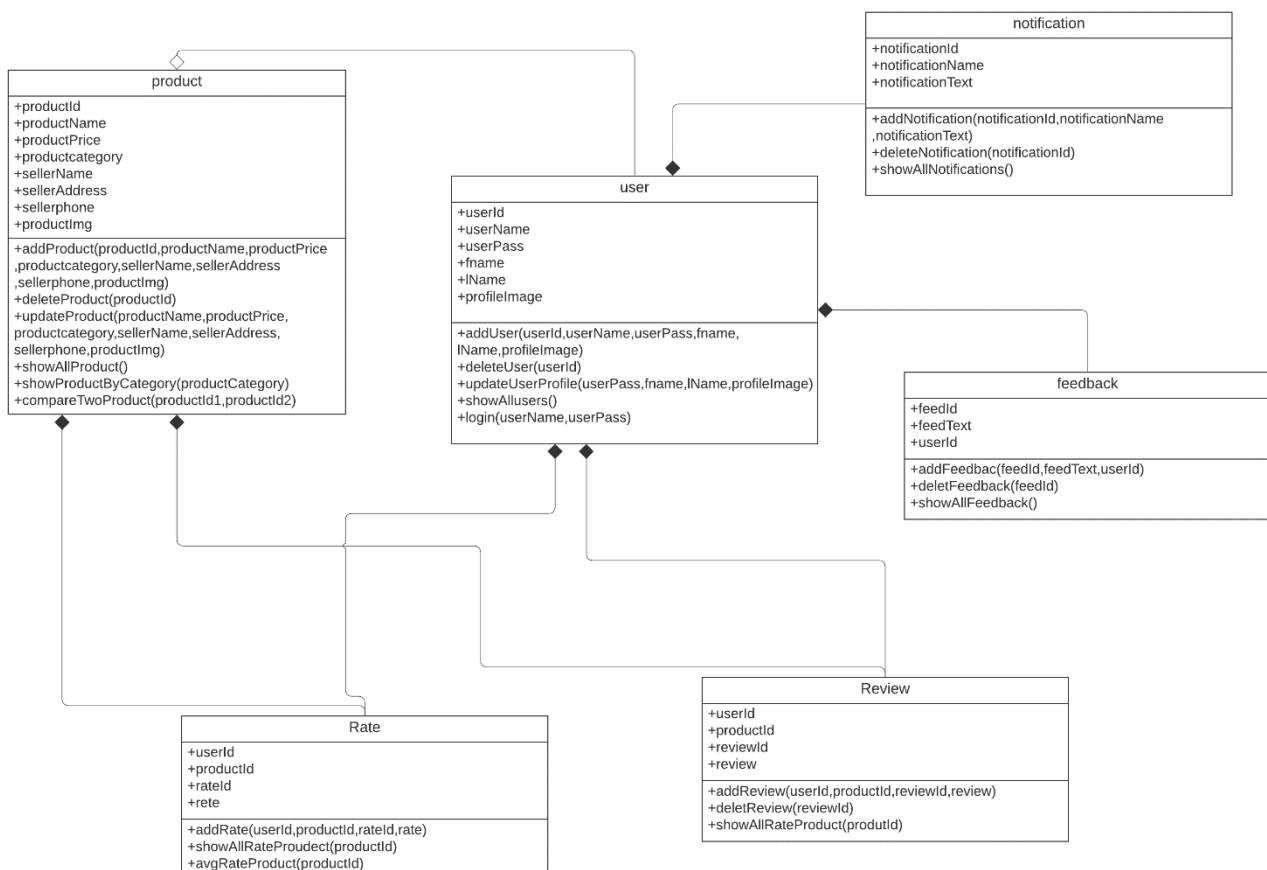
Advantages:

- clearer structure

Disadvantages:

- traceability is more difficult than one central class
- Limitation of reusability
- there is a further complication when there are two actors that can initiate an interaction

j) Class Diagram 2:



k) Four Design Patterns Applied

1–Singleton pattern:

Description

Using when we need to ensure that only one object of a particular class need to be created

Problem it solves

this is very useful when exactly one object is needed to coordinate actions across the system, to make sure that class has just one object

how it affects my system's design

it helps us in *DataBaseConnection* class to be sure that one Instance has been created form database.

2–Delegation pattern:

Description

the delegation pattern is an object-oriented design pattern that allows object composition to achieve the same code reuse as inheritance.

Problem it solves

We use this design pattern when we want to use function from class in another class, it will minimize development cost.

how it affects my system's design

it helps us in function compare which in class *ProductView* we use inside it function showAVg which in class *RateView*.

3–immutable pattern:

Description

this pattern uses to saving unique information, if you make instance can't be changed

Problem it solves

Protected important data from any changing.

how it affects my system's design

it helps us to protect user data like user id, user name, user password and admin from any changes, And all of them without duplication and any problem that will happen such as more than member has the same user id.

4-MVC pattern:

Description

Model, view and controller. Model is interact with database, controller is interact with model in changing data, view is show and list data from model.

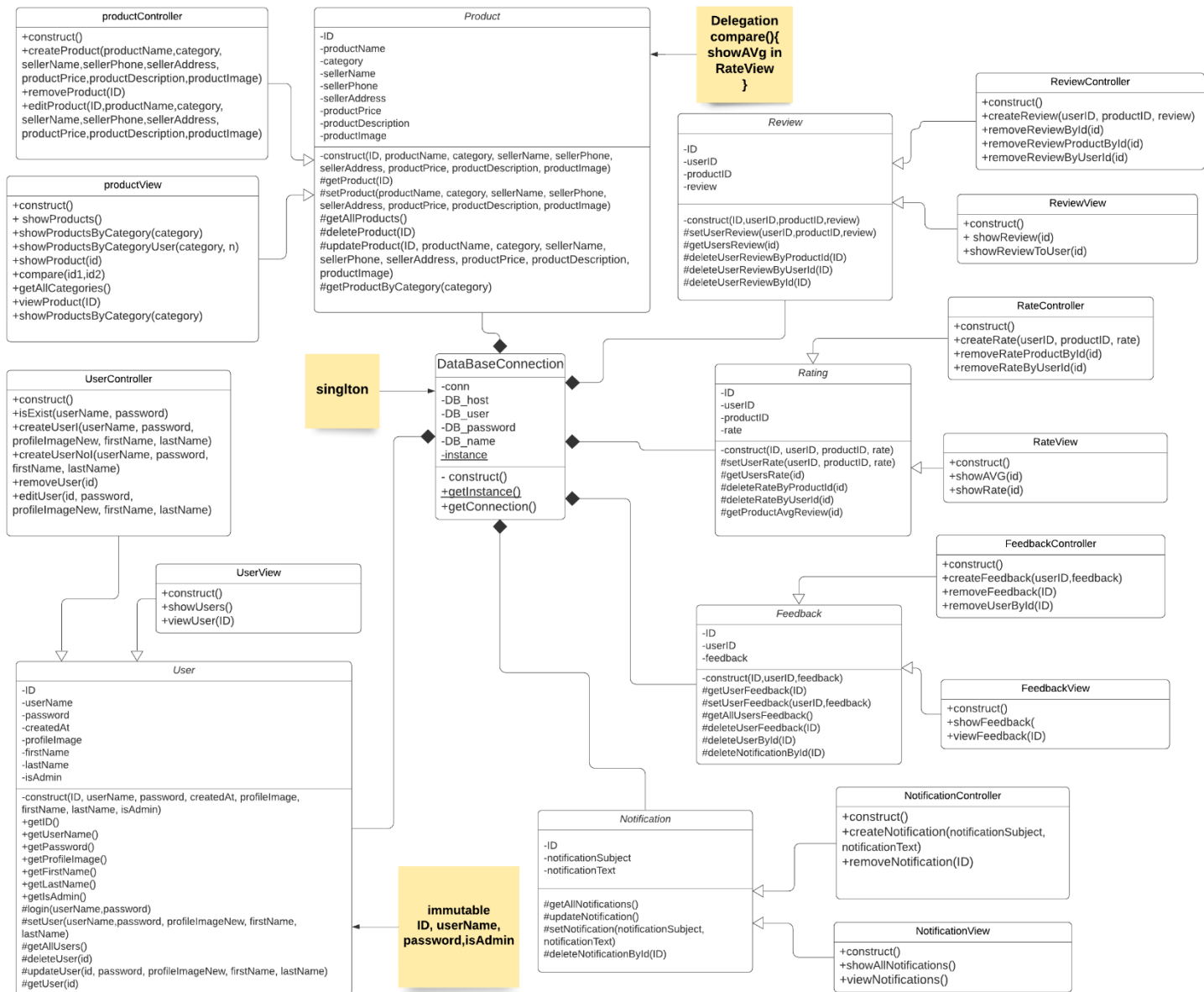
Problem it solves

Make refactor to code so easy, make code very organized.

how it affects my system's design

it helps me in control my code very simple way to change many functions in the system without effect in another function and make system very secure because no one interact or use function of database in model directly, just interact with controller or view.

I) Class Diagram 3:



PART 3: Development Phase (Implementation Details)

12) Create and document a Front-End Design for all Functions (HTML, Bootstrap).

```
<?php
require_once 'Model/Product.php';

class ProductView extends Product
{
    public function __construct()
    {
    }

    public function showProducts()
    {
        $Products = $this->getAllProducts();

        foreach ($Products as $product):?>
            <tr>
                <td><?= $product['ID'] ?></td>
                <td></td>
                <td><?= $product['product_name'] ?></td>
                <td><?= $product['category'] ?></td>
                <td><?= $product['product_desc'] ?></td>
                <td><?= $product['product_price'] ?></td>
                <td><?= $product['seller_name'] ?></td>
                <td><?= $product['seller_phone'] ?></td>
                <td><?= $product['seller_address'] ?></td>

                <td><a href="Delete.php?productID=<?= $product['ID'] ?>" class="btn btn-danger col-12"
                    style="margin-top: 5%">Delete</a>
                <a href="UpdateProduct.php?productID=<?= $product['ID'] ?>" class="btn btn-success col-12"
                    style="margin-top: 5%">Edit</a>
                <a href="RateAndReview.php?productID=<?= $product['ID'] ?>" class="btn btn-info col-12"
                    style="margin-top: 5%">Rate And Review</a></td>
            </tr>
        <?php
        endforeach;
    }

    /*
    this function must show when user choose type of category you must call this function
    */
    public function showProductsByCategory($category)
    {
        $Products = $this->getProductByCategory($category);
        foreach ($Products as $product):?>
            <tr>
                <td><?= $product['ID'] ?></td>
                <td></td>
                <td><?= $product['product_name'] ?></td>
                <td><?= $product['category'] ?></td>
                <td><?= $product['product_desc'] ?></td>
                <td><?= $product['product_price'] ?></td>
                <td><?= $product['seller_name'] ?></td>
                <td><?= $product['seller_phone'] ?></td>
            </tr>
        <?php
        endforeach;
    }
}
```



```

<td><? = $product['seller_address'] ?></td>

<td><a href="Delete.php?productID=<? = $product['ID'] ?>" class="btn btn-danger col-12"
    style="margin-top: 5%">Delete</a>
<a href="UpdateProduct.php?productID=<? = $product['ID'] ?>" class="btn btn-success col-12"
    style="margin-top: 5%">Edit</a>
<a href="RateAndReview.php?productID=<? = $product['ID'] ?>" class="btn btn-info col-12"
    style="margin-top: 5%">Rate
    And Review</a></td>
</tr>
<?php
endforeach;

}

public function showProductsByCategoryUser($category, $n)
{
    $Products = $this->getProductByCategory($category);
    foreach ($Products as $product):?>
        <div class="card">
            
            <div class="card-body">
                <h5 class="card-title"><? = $product['product_name'] ?></h5>
                <h2 class="card-title"><span style="color:#000000 "><? = $product['product_price'] . " EGP" ?></span>
                </h2>
                <span class="fa fa-star checked"></span>
                <span class="fa fa-star checked"></span>
                <span class="fa fa-star checked"></span>
                <span class="fa fa-star checked"></span>
                <span class="fa fa-star checked"></span>
                <br><br>
                <?php if ($n == 0):?>
                    <a href="Details.php?productID=<? = $product['ID'] ?>" class="btn btn-primary"
                        style="background: steelblue; !important;border: none;!important;">More Details</a>
                <?php else:?>
                    <a href="compare.php?productID1=<? = $_GET['prodID1'] . "&productID2=" . $product['ID'] ?>"
                        class="btn btn-primary"
                        style="background: steelblue; !important;border: none;!important;">choose</a>
                <?php
            endif;?>
        </div>
    </div>
<?php
endforeach;

}

public function showProduct($id)
{
    $Products = $this->getProduct($id);
    $rateView = new RateView();
    ?>
    <div class="col-sm-5">
        <a href="chooseProdTocompare.php?category=<? = $Products['category'] . "&prodID1=" . $_GET['productID'] ?>"
            class="btn btn-info">Compare to another product</a>
        <hr>
        <h3 class="card-title"><? = $Products['product_name'] ?></h3>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
    </div>

```



```

        <h2 class="card-title"><span style="color:#000000;font-size: 25px "><?= $Product1['product_price'] ?> <span
style="color:#75846f;font-size: 15px ">EGP</span></span></h2>        <br>
        <h5>Description :</h5>
        <p>
            <?= $Product1['product_desc'] ?>
        </p>
        <br>
        <h5>Seller Name :</h5>
        <p>
            <?= $Product1['seller_name'] ?>
        </p>
        <br>
        <h5>Seller Address :</h5>
        <p>
            <?= $Product1['seller_address'] ?>
        </p>
        <br>
        <h5>Seller Phone :</h5>
        <p>
            <?= $Product1['seller_phone'] ?>
        </p>

    </div>

    <div class="col-sm-4">

        <h3 class="card-title"><?= $Product2['product_name'] ?></h3>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        <span class="fa fa-star checked"></span>
        (
            <?php $rateView->showAVg($id2);?>
            from 5)
        <br><hr>
        
    </div>
    <div class="col-sm-2">
        <h2 class="card-title"><span style="color:#000000;font-size: 25px "><?= $Product2['product_price'] ?> <span
style="color:#75846f;font-size: 15px ">EGP</span></span></h2>
        <br>
        <h5>Description :</h5>
        <p>
            <?= $Product2['product_desc'] ?>
        </p>
        <br>
        <h5>Seller Name :</h5>
        <p>
            <?= $Product2['seller_name'] ?>
        </p>
        <br>
        <h5>Seller Address :</h5>
        <p>
            <?= $Product2['seller_address'] ?>
        </p>
        <br>
        <h5>Seller Phone :</h5>
        <p>
            <?= $Product2['seller_phone'] ?>
        </p>
    </div>

```

```
</div>
</div>
<?php
}
```

```
<?php

require_once 'Model/User.php';

class UserView extends User
{
    public function __construct(){}

    public function showUsers()
    {
        $users = $this->getAllUsers();
        foreach ($users as $user):?>
            <tr>
                <td><?= $user['ID'] ?></td>
                <td></td>
                <td><?= $user['first_name'] ?></td>
                <td><?= $user['last_name'] ?></td>
                <td><?= $user['user_name'] ?></td>
                <td><?= $user['password'] ?></td>
                <td><?= $user['created_at'] ?></td>
                <td><a href="Delete.php?userID=<?= $user['ID'] ?>" class="btn btn-danger col-12">Delete</a></td>
            </tr>
        <?php
        endforeach;
    }

    public function updateProfile($id, $password, $profileImageNew, $firstName, $lastName){

    }
}
?>
```

```
<?php

require_once 'Model/Review.php';

class ReviewView extends Review
{
    public function __construct()
    {
    }

    public function showReview($id)
    {
        $reviews = $this->getUsersReview($id);

        foreach ($reviews as $review):?>
```

```

        <tr>
            <td><?= "@" . $review['user_name'] ?></td>
            <td><?= $review['review'] ?></td>
            <td><a href="Delete.php?reviewID=<?= $review['ID'] ?>" class="btn btn-danger col-12"
                style="margin-top: 5%">Delete</a></td>
        </tr>

    <?php
        endforeach;
    }
    public function showReviewToUser($id)
    {
        $reviews = $this->getUsersReview($id);

        foreach ($reviews as $review):?>

            <tr style="text-align: center">
                <td><?= "@" . $review['user_name'] ?></td>
                <td class="text-center"><?= $review['review'] ?></td>

            </tr>

        <?php
            endforeach;
        }
    }
}

```

```

<?php
session_start();
require_once 'Controller/UserController.php';
if (isset($_SESSION['userSession'])) {
    header("Location: Index.php");
    exit(0);
} elseif (isset($_SESSION['adminSession'])) {
    header("Location: Admin.php");
    exit(0);
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <title>Sign In</title>

    <!-- Bootstrap CSS CDN -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
        integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddlhOegiu1FwO5qRGvFXOdJZ4"
        crossorigin="anonymous">
    <!-- Our Custom CSS -->
    <link rel="stylesheet" href="sidebar.css">
    <!-- Scrollbar Custom CSS -->

```

```

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/malihu-custom-scrollbar-
plugin/3.1.5/jquery.mCustomScrollbar.min.css">

<!-- Font Awesome JS -->
<script defer src="https://use.fontawesome.com/releases/v5.0.13/js/solid.js"
integrity="sha384-tzzSw1/Vo+0N5UhStP3bvwWPq+uvzCMfrN1fEFe+xBmv1C/AtVX5K0uZtmcHitFZ"
crossorigin="anonymous"></script>
<script defer src="https://use.fontawesome.com/releases/v5.0.13/js/fontawesome.js"
integrity="sha384-6OIrr52G08NpOFSZdxxz1xdNSndlD4vdcf/q2myIUV00VsqaGHJsB0RaBE01VTOY"
crossorigin="anonymous"></script>
<style type="text/css">
#container {
display: flex;
justify-content: center;
align-items: center;
width: 100%;
height: 100%;
margin-top: 10%;
}
</style>

</head>
<body>

<div id="container">
<form method="post">
<h1>Login</h1>
<div class="form-group">
<label>User Name</label>
<input class="form-control" name="username" type="text" required>
</div>

<div class="form-group">
<label>Password</label>
<input class="form-control" name="password" type="password" required>
</div>
<div class="form-group">
<input name="btn_login" class="btn btn-primary" type="submit" value="Sign In">
</div>
<p>Do not have an account? <a href="Register.php">Register here</a>.</p>
<?php
if (isset($_POST['btn_login'])) {
$username = $_POST['username'];
$password = $_POST['password'];

$userController = new UserController();
$user = $userController->isExist($username, $password);

if ($user) {
if ($user->getIsAdmin() == 0) {
$_SESSION['userSession'] = [$user->getID(),$user->getUserName(),$user->getPassword(),$user-
>getProfileImage(),$user->getFirstName(),$user->getLastName(),$user->getIsAdmin()];
header("Location: Index.php");

} elseif ($user->getIsAdmin() == 1) {
$_SESSION['adminSession'] = [$user->getID(),$user->getUserName(),$user->getPassword(),$user-
>getProfileImage(),$user->getFirstName(),$user->getLastName(),$user->getIsAdmin()];
header("Location: Admin.php");
}
} else {
echo '<div class="alert alert-danger" role="alert">User Name or Password Is not Correct!</div>';
}
}

```

```
}  
?>  
</form>  
  
</div>  
</body>  
</html>
```

13) Document an Implementation based on the abovementioned Requirements & Design

a) User Role Management Module.

To define access for some users we need to use Roles. Administrator can manage roles from the database panel. Each role defines access to certain which is applied to users who own that role.

One user can only have one role. This role can be selected for a specific user so it defines the restrictions assigned to the user.

By default, users have minimal access. You need to assign roles to grant users a specific access level, "0" represents a customer and "1" represents an admin.

Example in code

```
if ($user) {  
    if ($user->getIsAdmin() == 0) {  
        $_SESSION['userSession'] = [$user->getID(), $user->getUserName(), $user->getPassword(), $user->  
>getProfileImage(), $user->getFirstName(), $user->getLastName(), $user->getIsAdmin()];  
        header("Location: Index.php");  
    } elseif ($user->getIsAdmin() == 1) {  
        $_SESSION['adminSession'] = [$user->getID(), $user->getUserName(), $user->getPassword(), $user->  
>getProfileImage(), $user->getFirstName(), $user->getLastName(), $user->getIsAdmin()];  
        header("Location: Admin.php");  
    }  
}
```

b) User manipulation Module

```
<?php
class DataBaseConnection
{
    private $conn;
    private $DB_host = "localhost";
    private $DB_user = "root";
    private $DB_password = "";
    private $DB_name = "lasttest";
    private static $instance = null;

    private function __construct()
    {
        $this->conn = new mysqli($this->DB_host, $this->DB_user, $this->DB_password, $this->DB_name);

        // Error handling
        if (mysqli_connect_error()) {
            trigger_error("Failed to connect to MySQL: " . mysqli_connect_error(), E_USER_ERROR);
        }
    }

    // Create Instance If It There Is no Instance
    public static function getInstance()
    {
        if (!self::$instance) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    // Return connection
    public function getConnection(){
        return $this->conn;
    }
}
```

In model

```
<?php

require_once 'DataBaseConnection.php';

class Product
{
    private $ID;
    private $productName;
    private $category;
    private $sellerName;
    private $sellerPhone;
    private $sellerAddress;
    private $productPrice;
    private $productDescription;
    private $productImage;

    private function __construct($ID, $productName, $category, $sellerName, $sellerPhone, $sellerAddress,
$productPrice, $productDescription, $productImage)
    {
        $this->ID = $ID;
        $this->productName = $productName;
        $this->category = $category;
    }
}
```



```

        $this->sellerName = $sellerName;
        $this->sellerPhone = $sellerPhone;
        $this->sellerAddress = $sellerAddress;
        $this->productPrice = $productPrice;
        $this->productDescription = $productDescription;
        $this->productImage = $productImage;
    }

    protected function getProduct($ID)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("SELECT * FROM products WHERE ID = '$ID'")->fetch_assoc();
    }

    protected function setProduct($productName, $category, $sellerName, $sellerPhone, $sellerAddress, $productPrice, $productDescription, $productImage)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO products(product_name,category,seller_name,seller_phone,seller_address,product_price,product_desc,product_image) VALUES('$productName','$category','$sellerName','$sellerPhone','$sellerAddress','$productPrice','$productDescription','$productImage')");
    }

    protected function getAllProducts()
    {
        return DataBaseConnection::getInstance()->getConnection()->query("SELECT * FROM products");
    }

    protected function deleteProduct($ID)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM products WHERE ID = $ID;");
    }

    protected function updateProduct($ID, $productName, $category, $sellerName, $sellerPhone, $sellerAddress, $productPrice, $productDescription, $productImage)
    {
        if ($productImage != null)
            return DataBaseConnection::getInstance()->getConnection()->query("UPDATE products SET product_name = '$productName', category = '$category', seller_name = '$sellerName', seller_phone = '$sellerPhone', seller_address = '$sellerAddress', product_price = '$productPrice', product_desc = '$productDescription', product_image = '$productImage' WHERE ID = $ID ");
        else
            return DataBaseConnection::getInstance()->getConnection()->query("UPDATE products SET product_name = '$productName', category = '$category', seller_name = '$sellerName', seller_phone = '$sellerPhone', seller_address = '$sellerAddress', product_price = '$productPrice', product_desc = '$productDescription' WHERE ID = $ID ");
    }

    protected function getProductByCategory($category)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("SELECT * FROM products WHERE category = '$category'");
    }
}

```

```
<?php
```

```
require_once 'DataBaseConnection.php';
```

```
class User
{
    private $ID;
    private $userName;
    private $password;
    private $createdAt;
    private $profileImage;
    private $firstName;
    private $lastName;
    private $isAdmin;

    private function __construct($ID, $userName, $password, $createdAt, $profileImage, $firstName, $lastName, $isAdmin)
    {
        $this->ID = $ID;
        $this->userName = $userName;
        $this->password = $password;
        $this->createdAt = $createdAt;
        $this->profileImage = $profileImage;
        $this->firstName = $firstName;

        $this->lastName = $lastName;
        $this->isAdmin = $isAdmin;
    }

    public function getID()
    {
        return $this->ID;
    }

    public function getUserName()
    {
        return $this->userName;
    }

    public function getPassword()
    {
        return $this->password;
    }

    public function getProfileImage()
    {
        return $this->profileImage;
    }

    public function getFirstName()
    {
        return $this->firstName;
    }

    public function getLastName()
    {
        return $this->lastName;
    }
}
```

```

public function getIsAdmin()
{
    return $this->isAdmin;
}

protected function login($userName,$password){
    $res = DataBaseConnection::getInstance()->getConnection()->prepare("SELECT * FROM users WHERE user_name
= ? AND password = ? ");
    $res->bind_param("ss", $userName, $password);
    $res->execute();
    $res->store_result();
    $res->bind_result($id, $name, $pass, $at,$proImage,$firstName,$lastName,$isAdmin);
    if($res->fetch())
        return new User($id, $name, $pass, $at,$proImage,$firstName,$lastName,$isAdmin);
    return null;
}

protected function setUser($userName, $password, $profileImageNew, $firstName, $lastName){
    return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO
users(user_name,password,profile_image,first_name,last_name)VALUES('$userName','$password','$profileImageNew',
'$firstName','$lastName')");
}

protected function getAllUsers(){
    return DataBaseConnection::getInstance()->getConnection()->query("SELECT * FROM users WHERE is_admin =
false");
}

protected function deleteUser($id){
    return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM users WHERE ID = $id;");
}

protected function updateUser($id, $password, $profileImageNew, $firstName, $lastName)
{
    if ($profileImageNew != null)
        return DataBaseConnection::getInstance()->getConnection()->query("UPDATE `users` SET `password` =
'$password', `profile_image` = '$profileImageNew', `first_name` = '$firstName', `last_name` = '$lastName' WHERE
`users`.`ID` = '$id';");
    else
        return DataBaseConnection::getInstance()->getConnection()->query("UPDATE `users` SET `password` =
'$password', `first_name` = '$firstName', `last_name` = '$lastName' WHERE `users`.`ID` = '$id';");
}

protected function getUser($id){
    return DataBaseConnection::getInstance()->getConnection()->query("SELECT * FROM users WHERE ID = '$id' ")
->fetch_assoc();
}
}

```

```
<?php
```

```

require_once 'DataBaseConnection.php';

class Rating
{
    private $ID;
    private $userID;
    private $productID;
    private $rate;

    private function __construct($ID, $userID, $productID, $rate)
    {
        $this->ID = $ID;
        $this->userID = $userID;
        $this->productID = $productID;
        $this->rate = $rate;
    }

    protected function setUserRate($userID, $productID, $rate)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO
rating(user_id,prod_id,rate)VALUES('$userID','$productID','$rate')");
    }

    protected function getUsersRate($id)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("SELECT
users.first_name,users.last_name,users.user_name,products.product_name, rating.rate
FROM rating
INNER JOIN users ON rating.user_id=users.ID
INNER JOIN products ON rating.prod_id=products.ID
WHERE prod_id = $id AND rate IS NOT Null");
    }

    protected function deleteRateByProductId($id)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM rating WHERE prod_id =
$id;");
    }

    protected function deleteRateByUserId($id)
    {
        return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM rating WHERE user_id =
$id;");
    }

    protected function getProductAvgReview($id)
    {
        $sum = DataBaseConnection::getInstance()->getConnection()->query("SELECT SUM(rate) as sum FROM rating
WHERE prod_id = $id");
        $count = DataBaseConnection::getInstance()->getConnection()->query("SELECT COUNT(rate) as coun FROM
rating WHERE prod_id = $id");
        $data = $count->fetch_assoc();
        $cou = $sum->fetch_assoc();
        if ($data['coun'] == 0) {
            echo "0";
        } else {
            echo ($cou['sum'] / $data['coun']);
        }
    }
}

```

```

<?php

require_once 'DataBaseConnection.php';

class Feedback
{
    private $ID;
    private $userID;
    private $feedback;
    private function __construct($ID,$userID,$feedback){
        $this->ID = $ID;
        $this->userID = $userID;
        $this->feedback = $feedback;
    }

    protected function getUserFeedback($ID){
        $res = DataBaseConnection::getInstance()->getConnection()->prepare("SELECT * FROM feedback WHERE ID = ? ");
        $res->bind_param("i", $ID);
        $res->execute();
        $res->store_result();
        $res->bind_result($ID,$userID,$feedback);
        if($res->fetch())
            return new Feedback($ID,$userID,$feedback);
        return null;
    }

    protected function setUserFeedback($userID,$feedback){
        return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO
feedback(user_id,feedback)VALUES('$userID','$feedback')");
    }

    protected function getAllUsersFeedback(){
        return DataBaseConnection::getInstance()->getConnection()->query("SELECT feedback.ID,
users.first_name,users.last_name ,users.user_name, feedback.feedback
FROM feedback
INNER JOIN users ON feedback.user_id=users.ID;");
    }

    protected function deleteUserFeedback($ID){
        return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM feedback WHERE ID = $ID;");
    }

    protected function deleteUserById($ID){
        return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM feedback WHERE user_id =
$ID;");
    }
}

```

```

<?php

require_once 'DataBaseConnection.php';

class notification
{
    private $ID;

```

```

private $notificationSubject;
private $notificationText;

private function __construct($ID,$notificationSubject,$notificationText){
    $this->ID = $ID;
    $this->notificationSubject = $notificationSubject;
    $this->notificationText = $notificationText;
}

public function getID()
{
    return $this->ID;
}
public function getNotificationSubject()
{
    return $this->notificationSubject;
}
public function getNotificationText()
{
    return $this->notificationText;
}
protected function getAllNotifications(){
    return aBaseConnection::getInstance()->getConnection()->prepare("SELECT * FROM notification ");
}
protected function updateNotification(){
    return aBaseConnection::getInstance()->getConnection()->prepare("UPDATE notification SET status=0 WHERE status=1");
}

}

protected function setNotification($notificationSubject,$notificationText){
    return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO notification(notification_subject,notification_text)VALUES('$notificationSubject','$notificationText')");
}

protected function getUserNotifications(){
    return DataBaseConnection::getInstance()->getConnection()->query("SELECT notification_subject,notification_text
                                                                    FROM notification WHERE status = 1");
}

protected function deleteNotificationById($ID){
    return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM notification WHERE notification_id = $ID;");
}
}

```

```

<?php

require_once 'DataBaseConnection.php';

class Review
{
    private $ID;
    private $userID;
    private $productID;
    private $review;
    private function __construct($ID,$userID,$productID,$review){
        $this->ID = $ID;

```

```

$this->userID = $userID;
$this->productID = $productID;
$this->review = $review;
}

protected function setUserReview($userID,$productID,$review){
    return DataBaseConnection::getInstance()->getConnection()->query("INSERT INTO reviewing(user_id,prod_id,review)VALUES('$userID','$productID','$review')");
}

protected function getUsersReview($id){
    return DataBaseConnection::getInstance()->getConnection()->query("SELECT users.user_name, reviewing.review,reviewing.ID
                                FROM reviewing
                                INNER JOIN users ON reviewing.user_id=users.ID
                                WHERE prod_id = $id ;");
}

protected function deleteUserReviewByProductId($ID){
    return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM reviewing WHERE prod_id = $ID;");
}

protected function deleteUserReviewByUserId($ID){
    return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM reviewing WHERE user_id = $ID;");
}

protected function deleteUserReviewById($ID){
    return DataBaseConnection::getInstance()->getConnection()->query("DELETE FROM reviewing WHERE ID = $ID;");
}
}

```

Controller

```

<?php
require_once 'Model/Product.php';

class ProductController extends Product
{
    public function __construct()
    {
    }

    public function
createProduct($productName,$category,$sellerName,$sellerPhone,$sellerAddress,$productPrice,$productDescription,$productImage){
    return $this->
>setProduct($productName,$category,$sellerName,$sellerPhone,$sellerAddress,$productPrice,$productDescription,$productImage);
}

    public function removeProduct($ID){
    return $this->deleteProduct($ID);
}

    public function
editProduct($ID,$productName,$category,$sellerName,$sellerPhone,$sellerAddress,$productPrice,$productDescription,$productImage){

```

```

        return $this->updateProduct($ID,$productName,$category,$sellerName,$sellerPhone,$sellerAddress,$productPrice,$productDescription,$productImage);
    }

    public function viewProduct($ID){
        return $this->getProduct($ID);
    }
    public function showProductsByCategory($category){
        return $this->getProductByCategory($category);
    }
}

```

```

<?php

require_once 'Model/User.php';

class UserController extends User
{
    public function __construct()
    {
    }

    public function isExist($userName, $password)
    {
        return $this->login($userName, $password);
    }

    public function createUserI($userName, $password, $profileImageNew, $firstName, $lastName)
    {
        return $this->setUser($userName, $password, $profileImageNew, $firstName, $lastName);
    }

    public function createUserNoI($userName, $password, $firstName, $lastName)
    {
        return $this->setUser($userName, $password, "default3.png", $firstName, $lastName);
    }

    public function removeUser($id)
    {
        return $this->deleteUser($id);
    }

    public function editUser($id, $password, $profileImageNew, $firstName, $lastName)
    {
        return $this->updateUser($id, $password, $profileImageNew, $firstName, $lastName);
    }

    public function viewUser($ID)
    {
        return $this->getUser($ID);
    }
}

```

```

<?php

require_once 'Model/Rating.php';

class RateController extends Rating
{

```



```

public function __construct()
{
}

public function createRate($userID, $productID, $rate)
{
    return $this->setUserRate($userID, $productID, $rate);
}

public function removeRateProductById($id)
{
    return $this->deleteRateByProductId($id);
}

    public function removeRateByUserId($id)
    {
        return $this->deleteRateByUserId($id);
    }

}

```

```

<?php

require_once 'Model/Review.php';

class ReviewController extends Review
{
    public function __construct()
    {
    }

    public function createReview($userID, $productID, $review)
    {
        return $this->setUserReview($userID, $productID, $review);
    }

    public function removeReviewById($id)
    {
        return $this->deleteUserReviewById($id);
    }

    public function removeReviewProductById($id)
    {
        return $this->deleteUserReviewByProductId($id);
    }

    public function removeReviewByUserId($id)
    {
        return $this->deleteUserReviewByUserId($id);
    }

}

```

```

<?php

require_once 'Model/Review.php';

class notificationController extends notification

```

```

{
    public function __construct()
    {
    }

    public function createNotification($notificationSubject, $notificationText)
    {
        return $this->setNotification($notificationSubject,$notificationText)
    }

    public function removeNotification($id)
    {
        return $this->deleteNotificationById($id);
    }

    public function updateNotificationStat()
    {
        return $this->updateNotification();
    }
}

```

```

<?php

require_once 'Model/Feedback.php';

class FeedbackController extends Feedback
{
    public function __construct()
    {
    }
    public function createFeedback($userID,$feedback){
        return $this->setUserFeedback($userID,$feedback);
    }
    public function removeFeedback($ID){
        $this->deleteUserFeedback($ID);
    }

    public function viewFeedback($ID){
        return $this->getUserFeedback($ID);
    }
    public function removeUserById($ID){
        return $this->deleteUserById($ID);
    }
}

```

c) Controlling Resources Module

```

public function showProductsByCategory($category)
{
    $Products = $this->getProductByCategory($category);
}

```

```

foreach ($Products as $product):?>
    <tr>
        <td><?= $product['ID'] ?></td>
        <td></td>
        <td><?= $product['product_name'] ?></td>
        <td><?= $product['category'] ?></td>
        <td><?= $product['product_desc'] ?></td>
        <td><?= $product['product_price'] ?></td>
        <td><?= $product['seller_name'] ?></td>
        <td><?= $product['seller_phone'] ?></td>
        <td><?= $product['seller_address'] ?></td>

        <td><a href="Delete.php?productID=<?= $product['ID'] ?>" class="btn btn-danger col-12"
            style="margin-top: 5%">Delete</a>
        <a href="UpdateProduct.php?productID=<?= $product['ID'] ?>" class="btn btn-success col-12"
            style="margin-top: 5%">Edit</a>
        <a href="RateAndReview.php?productID=<?= $product['ID'] ?>" class="btn btn-info col-12"
            style="margin-top: 5%">Rate
            And Review</a></td>
    </tr>
<?php
endforeach;
}

```

f) Sending Emails or Notifications Module.

```

<?php
if (isset($_POST["view"])) {

    require_once 'DataBaseConnection.php';
    if ($_POST["view"] != '') {
        $notiObj = new notificationController();
        $update_query = $notiObj->updateNotificationStat();
    }
    $query = "SELECT * FROM notification ORDER BY ID DESC LIMIT 3";
    $results = DataBaseConnection::getInstance()->getConnection()->query($query);
    $output = '';

    if ($results->num_rows>0) {
        foreach ($results
as $result):
            $output .= '
<li style="padding: 0.5rem">
                <strong>' . $result["notification_subject"] . '</strong><br>
                <small><em>' . $result["notification_text"] . '</em></small>
                <hr>
            </li>
';
        endforeach;
    } else {
        $output .= '
<li><a href="#" class="text-bold text-italic">No Notification Found</a></li>';
    }
}

```

```
}  
  
$query_1 = "SELECT * FROM notification WHERE status=1";  
$result_1 = DataBaseConnection::getInstance()->getConnection()->query($query_1);  
$count = $result_1->num_rows;  
$data = array(  
    'notification' => $output,  
    'unseen_notification' => $count  
);  
echo json_encode($data);  
}
```

g) File Uploaders.

```
$productImage = $_FILES['product_image']['name'];  
$tempProductImage = $_FILES['product_image']['tmp_name'];
```

```
$profileImage = $_FILES['profile_image']['name'];  
$tempProfileImage = $_FILES['profile_image']['tmp_name'];
```

14) Design some User Interfaces

Human Computer Interaction

The user interface is one of the most important things that make users use system services and are attracted to it. So, we can say the interaction design is the relationship between users and the services they use. So in Our System We designed the UI based on Several principle for designing interaction Such as

- 1.UX: That the user's expectations will be matched with his experience, because it's simple and clear UI.
- 2.Consistent design: All website's pages have the same colors, fonts
- 3.Learnability: The design is simple and easy to learn, with simple terms that make it easier with using.

Some user interface

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Toggle Sidebar





Welcome hameed00!

Logout

Feedback

User Name	Feedback	Action
@hameed00	good	Delete
@jjjj	I love this site to much <3	Delete
@jjjj	asdf sdfasdfsaf sfsdfas	Delete
@jjjj	kahs skadhsd sdahlsf sdf asdfi	Delete

Users Table

ID	Profile Picture	First Name	Last Name	User Name	Password	Created At	Action
24		shady	ali	shadii2l	274150dc5ba7126aab0bb94ccaf72f37	2020-04-22 22:03:09	Delete
25		sdgs	sdfgds	dfgdfgs	809af994628b97989f27db26c90dlca4	2020-04-30 07:04:30	Delete
27		hjjjjjj	gfjgfhj	khaled57	809af994628b97989f27db26c90dlca4	2020-05-01 05:57:42	Delete
28		sgf	sdfg	ffff	3028879ab8d5c87dc023049fa5bb5c1a	2020-05-01 05:58:39	Delete

Toggle Sidebar

Welcome hameed00!

Logout

Products Table

Category


TV

Laptop

Watch

Mobile

Show

ID	Product image	Product name	Product category	Description	Product Price	Seller Name	Seller Phone	Seller Address
24		any watch	Watch	sdfgdfs dfsgdfsgdsf dffsgsdfgdsfgdfs dsfgdsfgdfsgsdfgds dsfg	554	dfsg	5505100551	510 dfsgd

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Notification

Enter Subject

Enter Comment

Send notifications to all users

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Login

User Name

Password

Sign In

Do not have an account? Register here.

Control Panel

Home

Products

View Products

Add Products

Users

View Users

Add Users

Feedback

Notifications

Add Product

Upload Product picture

Browse...

No file selected.

☐ TV

☐ Laptop

☐ Watch

☐ Mobile

Enter Product Name

Enter Product Price

Enter Seller Name

Enter Seller Phone

Enter Seller Address

Product Review Analysis for Genuine Rating

Home

Profile


Compare

Team

Log Out

dell

★★★★★ (0 from 5)



4325 EGP

Description :
reg s dfgsfdgsfd gf
sdg sdfgsdf gf
sdgfsfgfs dgs s
fdgsfdgthy jy j dyfjfj
dfgj


Seller Name :
fdgh

Seller Address :
510 dfsgd

Seller Phone :
5505100551

laptop

★★★★★ (0 from 5)



432 EGP

Description :
werter tertewrt
erwtwert
rwtwertwr
erwtewtet
ewtwertet

Seller Name :
fdgh

Seller Address :
50 sdfas sadf dddd

Seller Phone :
5505100551

Product Review Analysis for Genuine Rating

Home

Profile

Compare


Team

Log Out

Compare to another product

dell

★★★★★ (0 from 5)



4325 EGP

Description :
reg s dfgsfdgsfd gf sdg sdfgsdf
gf sdgfsfgfs dgs s fdgsfdgthy jy
j dyfjfj dfgj

Seller Name :
fdgh

Seller Address :
510 dfsgd

Seller Phone :
5505100551

Enter Your Rate from 1 to 5

☐ ★★★★★

☐ ★★★★☆

☐ ★★★☆☆

☐ ★★☆☆☆

☐ ★☆☆☆☆

☐ ☆☆☆☆☆

Submit

Enter Your Review

Feedback

Minimum 20 Character ...

Feel Free To Write What You Want .

Submit

Our Hours

Sunday:9am - 12am

wednesday:9am - 12am

Friday:Closed


Service Area

Sunday:9am - 12am

wednesday:9am - 12am

Friday:Closed

Product Review Analysis for Genuine Rating
Home
Profile
Compare
Team
Log Out




dell

235 EGP

★★★★★

More Details




laptop

432 EGP

★★★★★

More Details




laptop

4500 EGP

★★★★★

More Details




dell 500

5243 EGP

★★★★★


More Details

Our Categories




Laptop

This Category shows you the




Mobile

This Category shows you



TV

This Category shows you the



Watch

This Category shows you the

Hameed hassan	Part3 (point 12) Part3 (point 13 point b (login ,add,delete) Part 3 (point f,g) Part 3 (point 14(HCI) Part two(use case description ,point 11(point a, e, j,l) Part 3 point (12)	(
Khaled riyad	Part 3 (point a) Part 3 (point b (update ,search,list) Part 3(point 13point c) Part 3 (point 13 point c) Part 3 (point 14 (hci) Part 2 (point 11 points g,h)	
Hussien wagdy	Part 2 (point 10 points a,c) Part 2 (point 11 points b,c,f,g,h,I,d) Part 4 (point 17 point a)s	

