Ain-Shams University

Faculty of Engineering

Computer Engineering and Software Systems Program

CSE222: Software Engineering (2)

Project name: E-Commerce

Ayman Hesham Mohamed	16P3037
Engy Samy Salah	16P3004
Gina Emil Attia	16P3022
Mayar Wessam Hassan	16P3008
Rowan Hazem Wagieh	16P3023
Yara Hossam Mohamed	16P3002

1. Functional and Non-functional requirements:

Functional requirements:

- Authenticate the admin and the user.
- Collect the data of the user.
- Display the categories and its items.
- Display the cost of each item.
- Sort the products according to the price.
- Calculate the total price of the items' ordered.
- Display the favorite page of each user.
- Send mails to the users containing promo codes.

Non-Functional requirements:

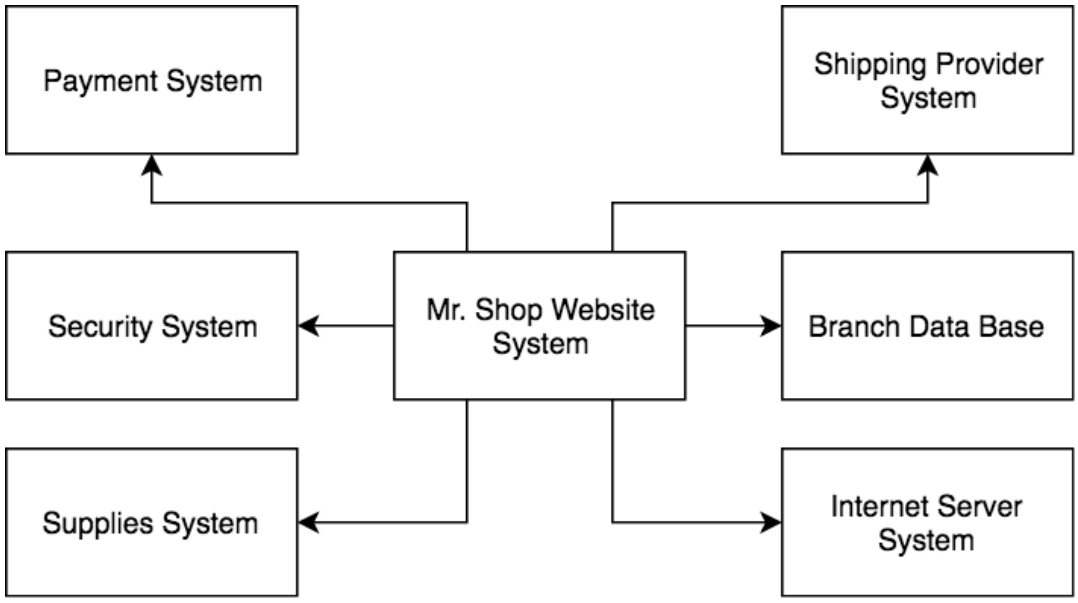
- Must be reliable.
- Must have high security
- Have limited cost
- Implemented using PHP
- High speed.
- Easy for the users to use.

2. System Models

1. Context Models:

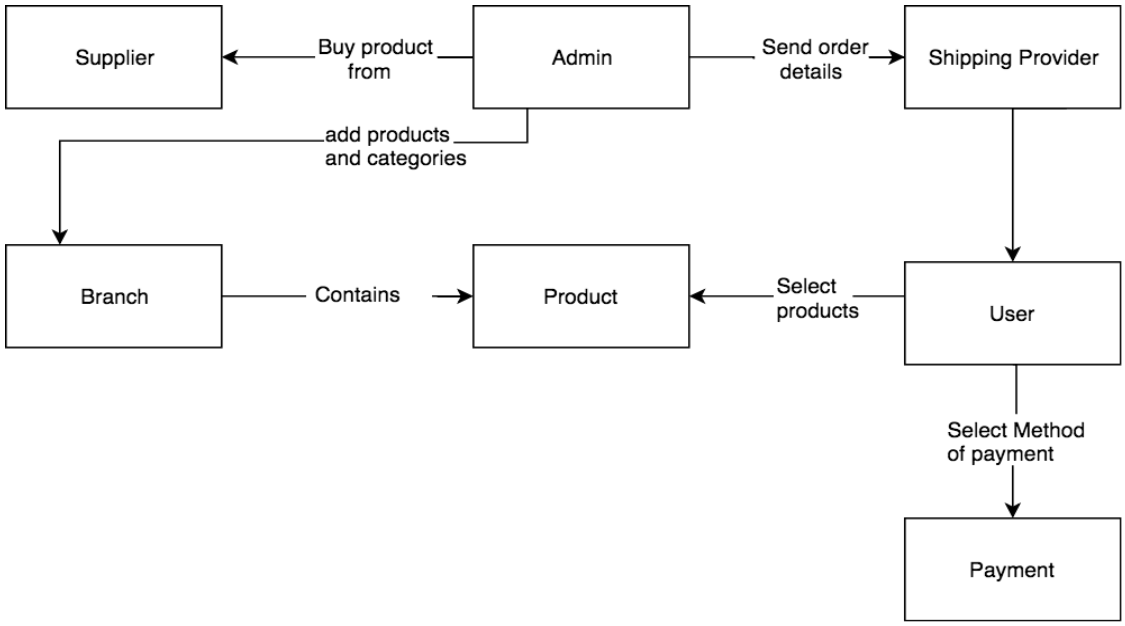
Context models are used to illustrate the operational context of the website they show what lies outside the system boundaries.

a. Context model:



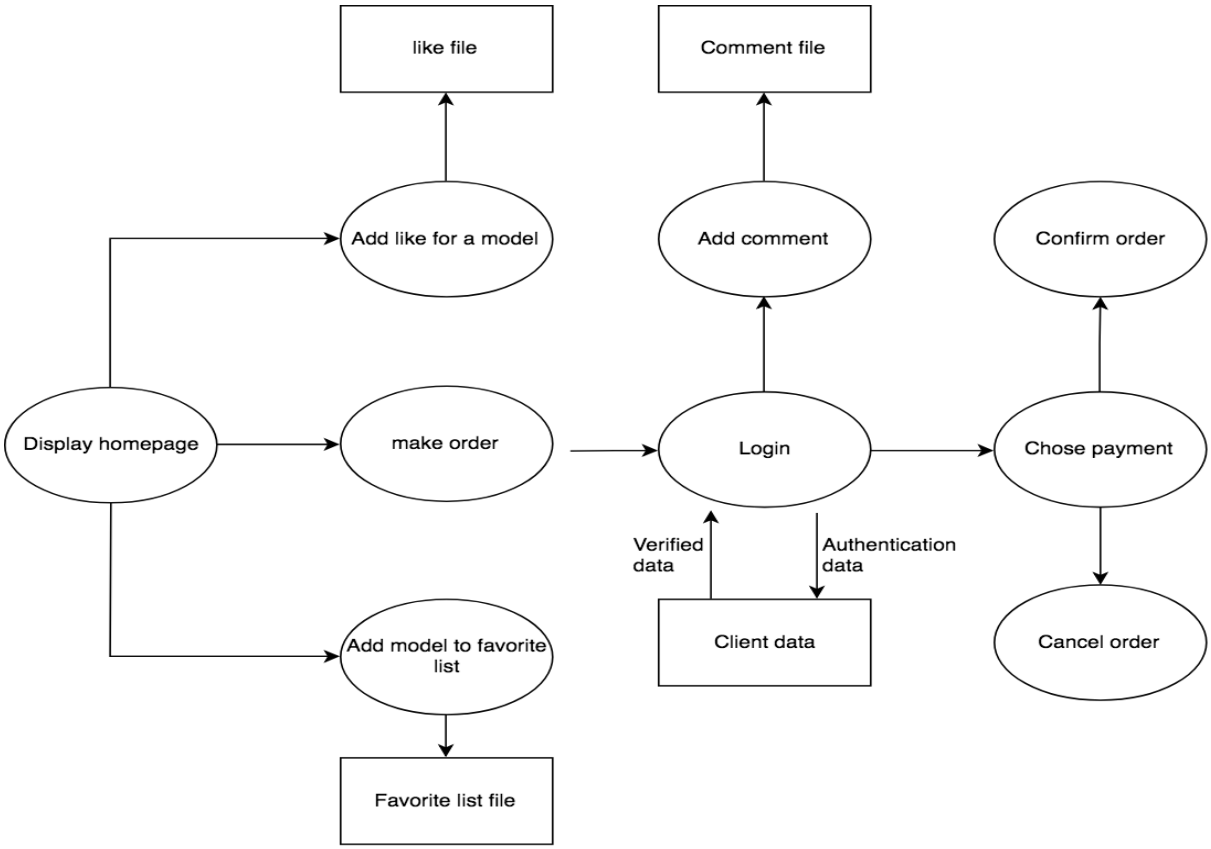
Mr. Shop website works in an environment composed of many systems as: security system responsible for ensuring safety of the user's account information and payment , supplement system responsible of the process of importing new products from the supplier to the admin so that he could add it on the system in the corresponding category with its price , internet server system to maintain the connection of the internet for the website and have no bugs , data base system for saving information of the users, products , orders .. , payment system responsible for collecting the order fees from users and shipping provider system that has contract with the website to deliver orders to customers.

b. Architecture model:



Mr. shop website is divided into different components and subsystems as supplier responsible for selling product to the admin, admin responsible for adding, upgrading , removing products and categories into the website and also communicate with the shipping provider to give it the orders details ,branch is the inventory which contains the number of items and all the details concerning it , user that make orders and choose its preferable method of payment to complete his checkout.

2. Process Model

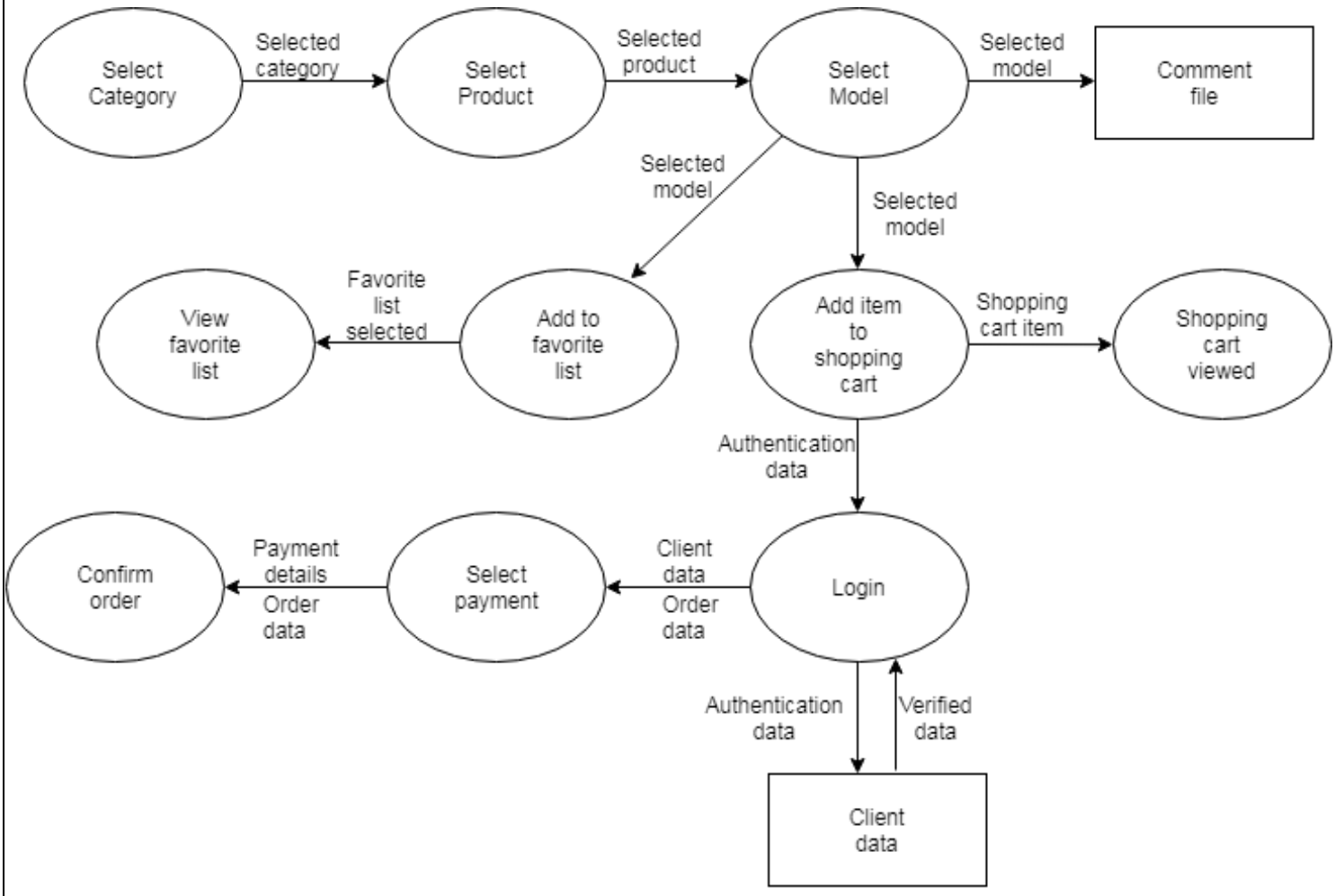


The process model describes:

- The system is able to display the homepage when the user opens the website
- Once it is opened he is able to add a model to his favorite list even if he hasn't logged in, and this list is saved in a file.
- Also, he could add like to a certain model as a review for others for this model.
- When he makes an order, the system asks him to log in to complete checkout. So, when he login he has to enter more information about the shipping.
- Once logged in the user is able to write comments and his username is displayed with his account to refer the comment to this user.
- The system asks the user to choose his method of payment for the order
- So, he could confirm the order or cancel it if he changes his mind before sending the shipping details to the shipping provider

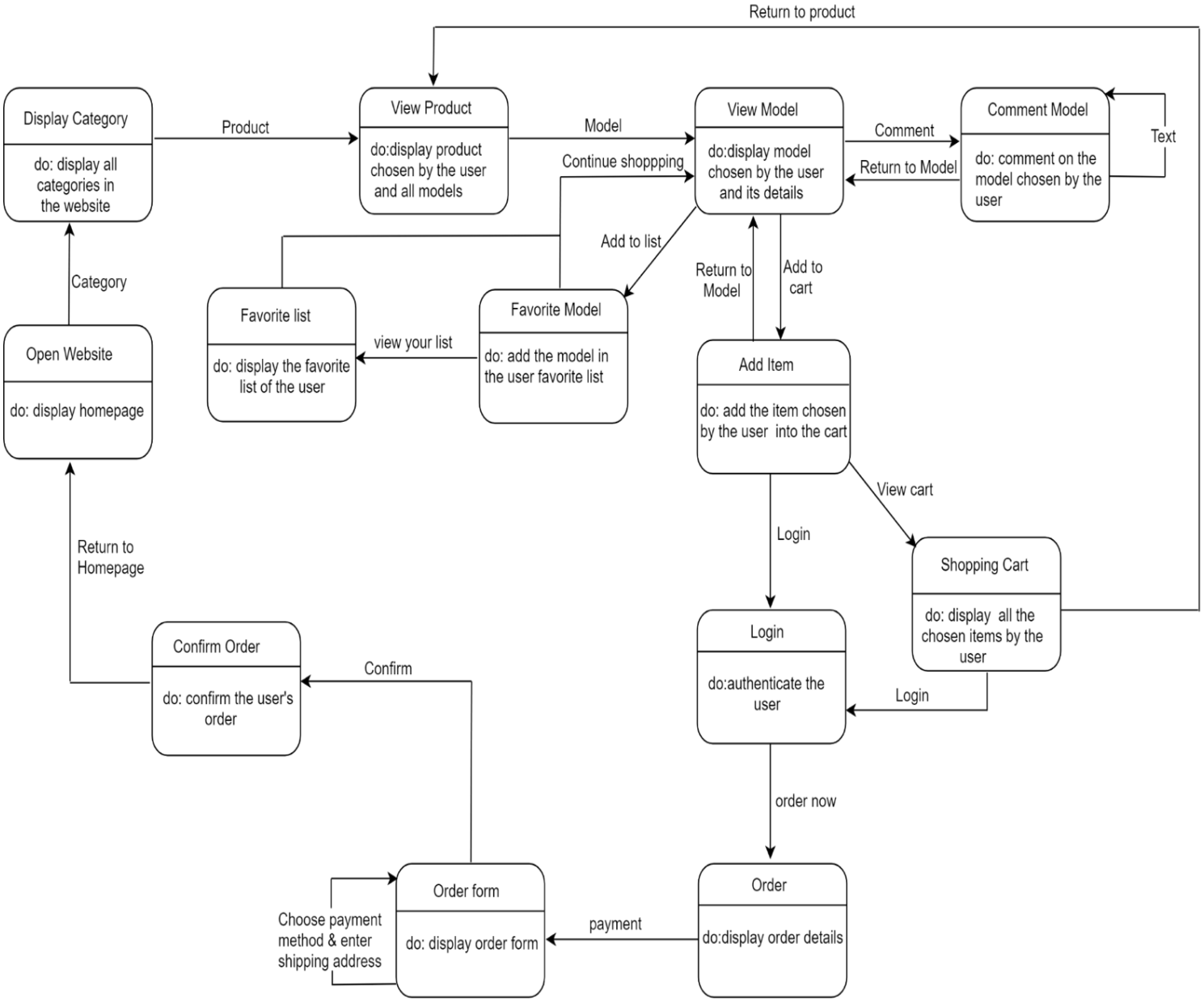
3. Behavioral Models:

a. Data Flow Diagram (DFD):



The data flow model described the data between each process of the program, for example the select category process has an output selected category which will be the input of the process select product and so on. Also, the output of a process could be saved in an entity as for example the process select model has an output selected model which will be saved in comment file entity. Also, an entity could have an output which would be an input for a process as the client data entity and the login process that have an input data verified data.

b. State Machine:



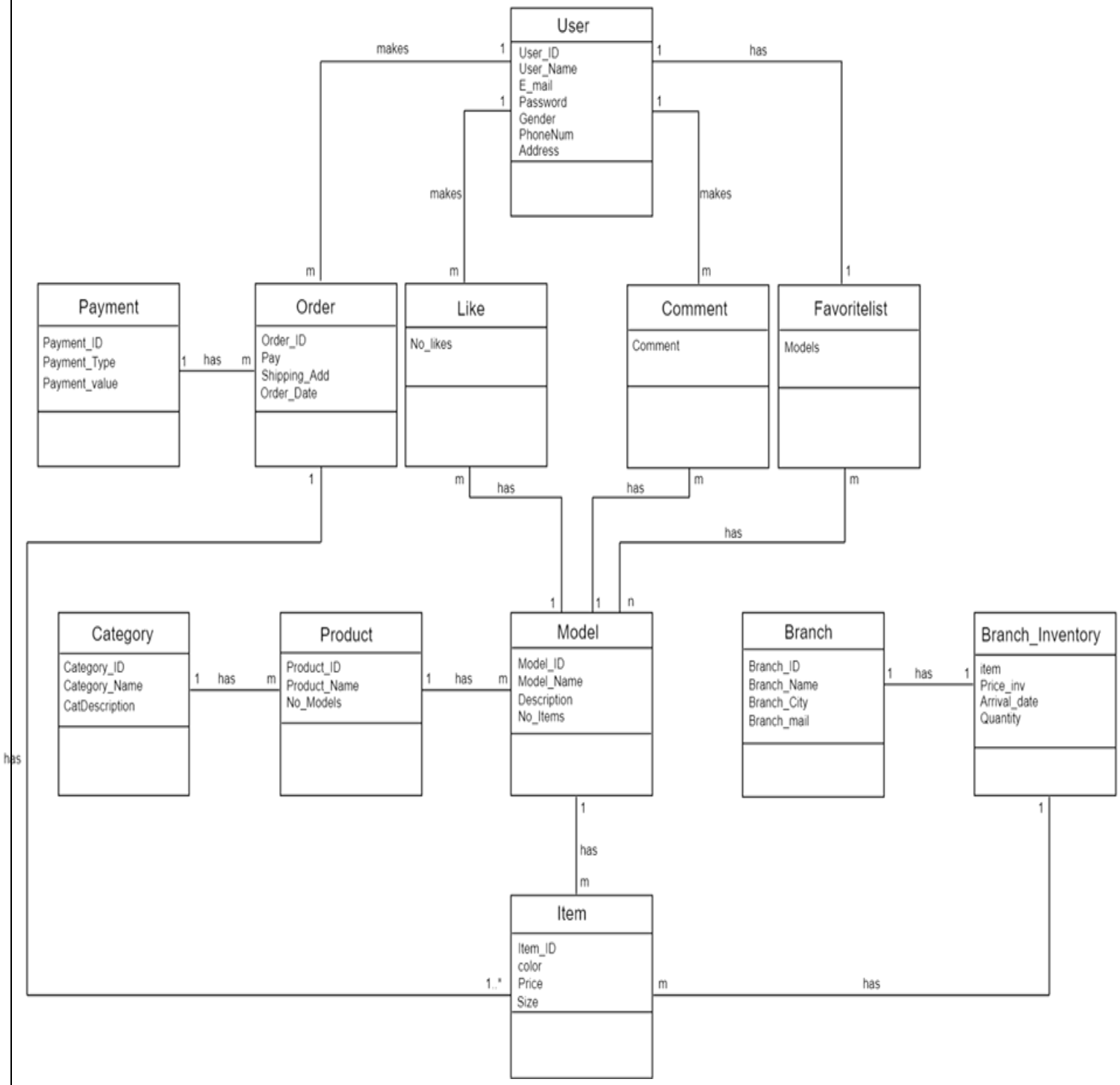
State Description:

State	Description
Open Website	The user opens the website and the website displays the homepage
Display Category	The website displays all the categories
View Product	The website displays the product chosen by the user and its models
View Model	The website displays the Model chosen by the user and its details
Comment Model	The model's comment is set to the user's input text and the website update the comments and displays it
Like Model	The user likes the model and the website saves this like on the model
Favorite Model	The website adds the item chosen by the user into his/her favorite list
Favorite List	The website displays the favorite list of the user
Add item	The website adds the items chosen by the user into his/her shopping cart
Shopping cart	The website displays the shopping cart of the user
Login	The website authenticates the user
Order	The website displays the order details of the user
Order form	The website displays the order form and the user enter the payment method and the shipping address and the website saves this data
Confirm Order	The website informs the user that the order is confirmed
Return to homepage	The order has been confirmed

Stimulus Description:

Stimulus	Description
Category	The user has pressed the category button
Product	The user chooses a specific product
Model	The user chooses a specific model
Comment	The user has pressed on the comment text
Return to Model	The comment has been saved on the model
Text	The user has written a comment
Like	The user has pressed the like button
Add to list	The user has pressed the Add to list button
Continue shopping	The user has pressed the continue shopping button
View your list	The user has pressed the View your list button
Add to cart	The user has pressed the Add to cart button
Return to model	The item has been added to the cart
View cart	The user has pressed the View cart button
Return to product	The user has pressed the Return product button
Login	The user has pressed the login button
Order now	The user has pressed the Order now button
payment	The user has pressed the payment button
Choose payment and enter shipping address	The user chooses the payment method and enter the shipping address
Confirm	The user has pressed the confirm button

c. Semantic Data Model:



d. Data Dictionary:

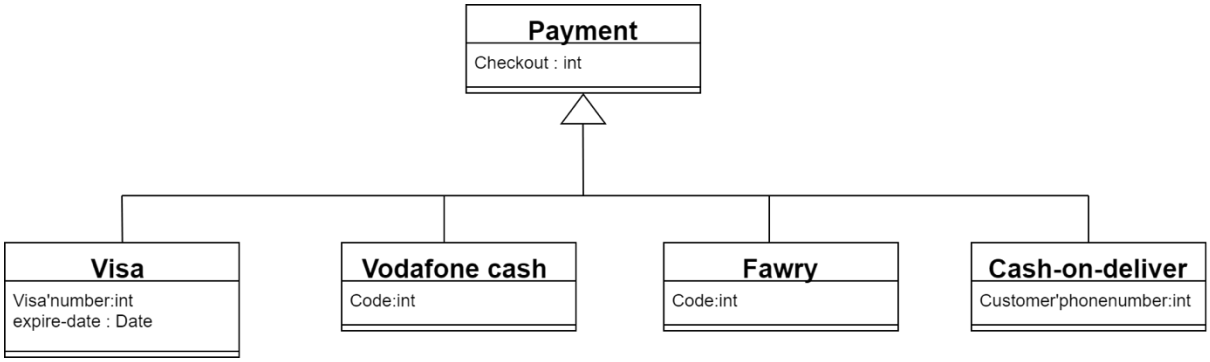
Name	Description	Type	Date
Product	Contains the details of each product added by the admin to any branch.	Entity	1.12.2018
User	Contains the account information of each user; ID, mail, password so that he could login and make his functionality on the website.	Entity	1.12.2018
Payment	Contains the type of payment and get and set the value that have to be paid.	Entity	1.12.2018
Order	Contains order details done by any user when he selects certain items.	Entity	1.12.2018
Like	Contains the number of likes for all models as a review for all models.	Entity	1.12.2018
Comment	Contains all comments for written by users for models describing them.	Entity	1.12.2018
Favoritelist	Contains the models favorited by the user.	Entity	1.12.2018
Model	Contains the name and description of each model and how many items available for it.	Entity	1.12.2018
Branch	Contains the information of each branch; its name, address, city ... so, the admin could add new branch or remove it.	Entity	1.12.2018
Branch_Inventory	Contains an ArrayList of items available in the branch with their price and quantity and arrival date.	Entity	1.12.2018
Item	Contains the ID, color, price, size of items. and the admin could add, remove or set a price of a certain item.	Entity	1.12.2018
Address	The address of the user that creates an account in the website to start shopping online and make orders and checkout.	Attribute	4.12.2018

Order_ID	The order of the ID made by user when he adds items to his shopping cart and proceed to order.	Attribute	4.12.2018
Branch_ID	The ID of the branch that contains the item storage.	Attribute	4.12.2018
Order_Date	The date of the order made by the user.	Attribute	4.12.2018
Arrival_Date	The arrival time of an item in the branch_inventory.	Attribute	4.12.2018
No_likes	The number of likes for a certain model. It is used as a review for models.	Attribute	4.12.2018
Category_Name	The name of the category containing certain products.	Attribute	4.12.2018
Category	The category of the website can contain many products related to a certain category for example (electronics category contains mobiles, laptops, headphones …	Entity	6.12.2018
Phone	The phone number of the user is saved in the data base to ensure the safety of the order and deliver it.	Attribute	9.12.2018
Payment	The payment is selected by each user to complete the checkout. It could be by visa, cash on delivery, Vodafone cash …	Attribute	3.12.2018
Shipping provider	The shipping provider is responsible for delivering orders to customers and could collect the money if the user chooses to pay cash on delivery.	Entity	4.12.2018
User-makes-order	A 1:m relation between the user and the order to add an order to a certain user.	Relation	1.12.2018
User-has-favoritelist	A 1:1 relation between the user and the favoritelist when the user add new items to his favoritelist.	Relation	1.12.2018
User-makes-like	A 1:m relation between the user and the like when the user likes certain product and be saved in the like.	Relation	1.12.2018

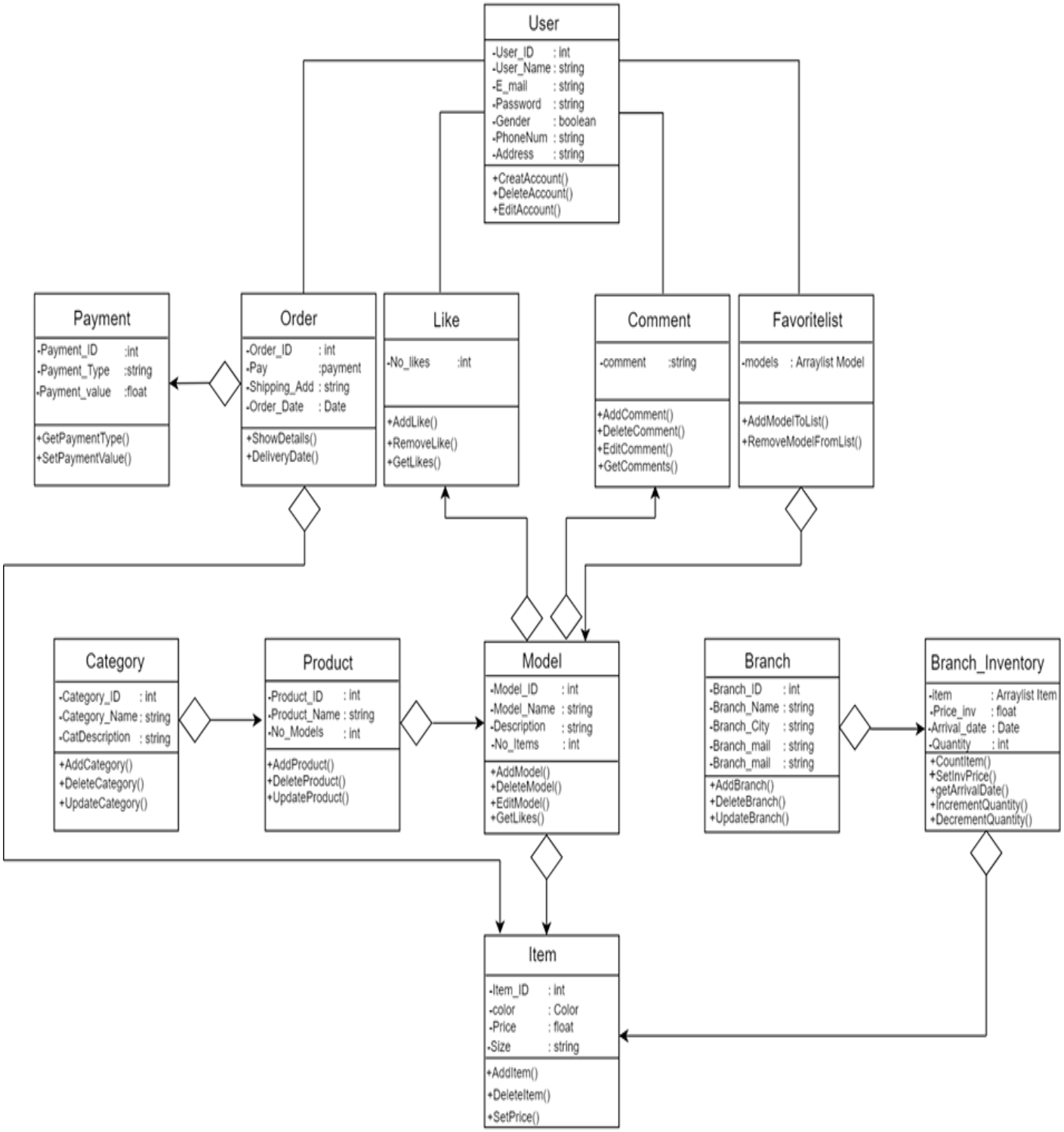
User-makes-comment	A 1:m relation between the user and the comment when the user writes a comment on a model.	Relation	1.12.2018
Order-has-item	A 1:m relation between the order and item as one order can contain one or more items, but one item should be in one and only one order.	Relation	1.12.2018
Like-has-model	A m:1 relation between like and model as we could have one model with many likes.	Relation	1.12.2018
Comment-has-model	A m:1 relation between the comment and the model as one model could have many comments.	Relation	1.12.2018
Branch-has-branch_Inventory	A 1:1 relation between the branch and the branch_Inventory to save all storage in each store in an inventory. Each branch should have only one branch_inventory.	Relation	1.12.2018
Branch_Inventory-has-Item	A 1:m relation between the branch_Inventory and the item to save items in this inventory. Each branch_Inventory has many items.	Relation	1.12.2018
Favoritelist-has-Model	A m:m relation between the Favoritelist and the model. A favorite list could have many models and also many models could be in many favorite lists.	Relation	1.12.2018
Payment-has-Order	A 1:m relation between the payment and the order. One payment method could be assigned for many orders. For example, paying by visa could be chosen by many to complete their order checkout.	Relation	1.12.2018
Product-has-Model	A 1:m relation between Product and Model. One product could be many models. For example, mobile could be iPhone, Samsung note 10...	Relation	1.12.2018
Category-has-Product	A 1:m relation between Category and Product. One category can have many products. For example, electronics category has mobile phones, laptops …	Relation	1.12.2018

e. Object Models:

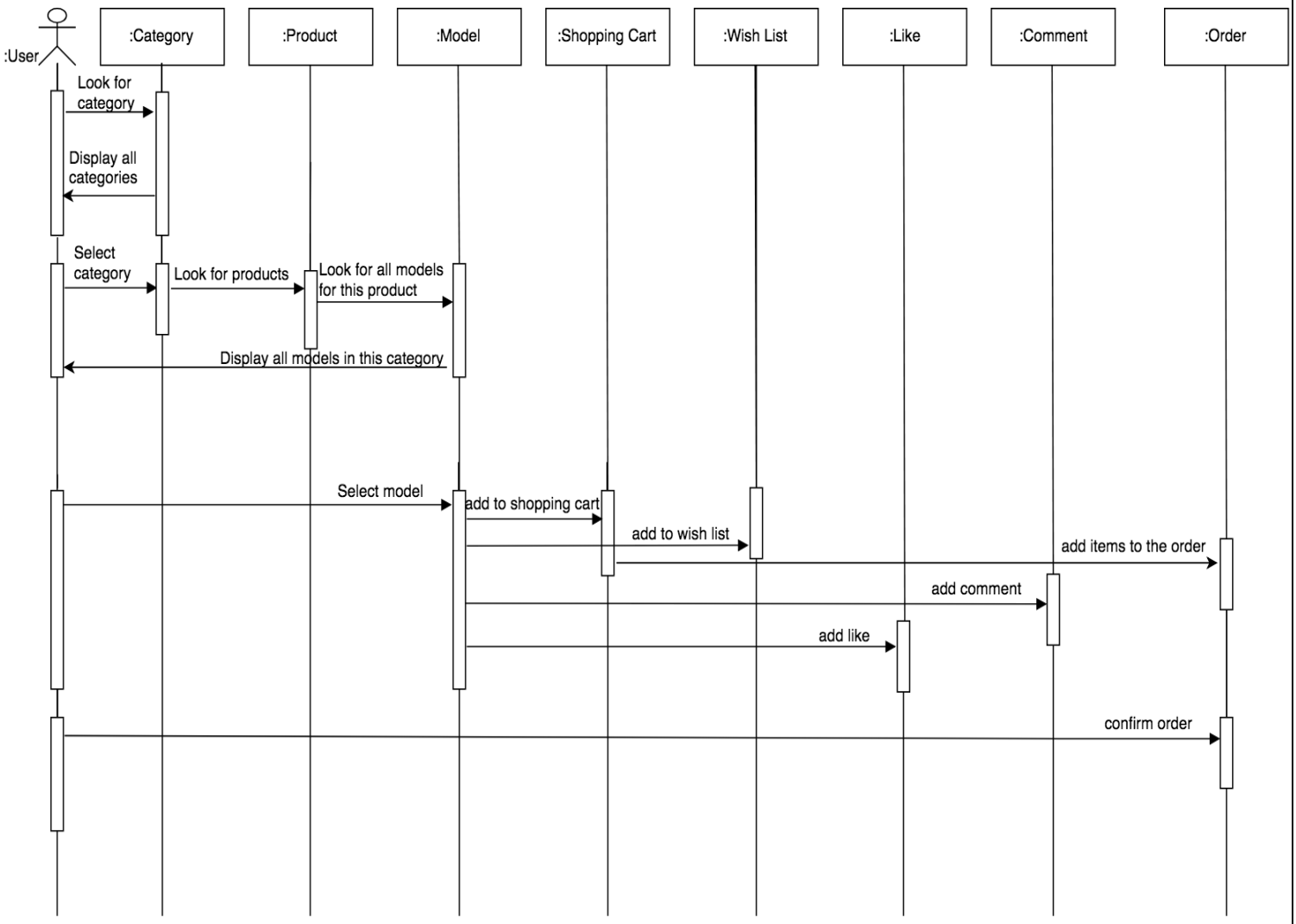
- **Class Hierarchy**



- **Aggregation Model:**



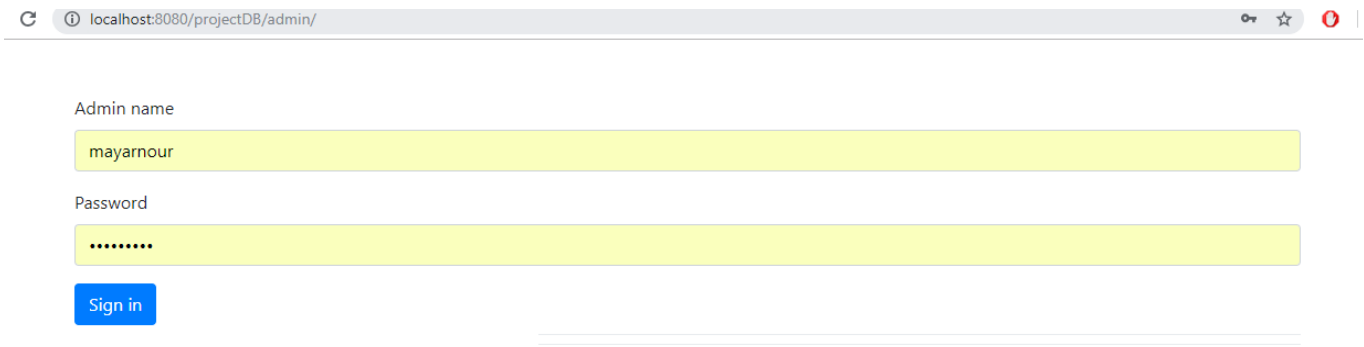
- **Sequence Diagram:**



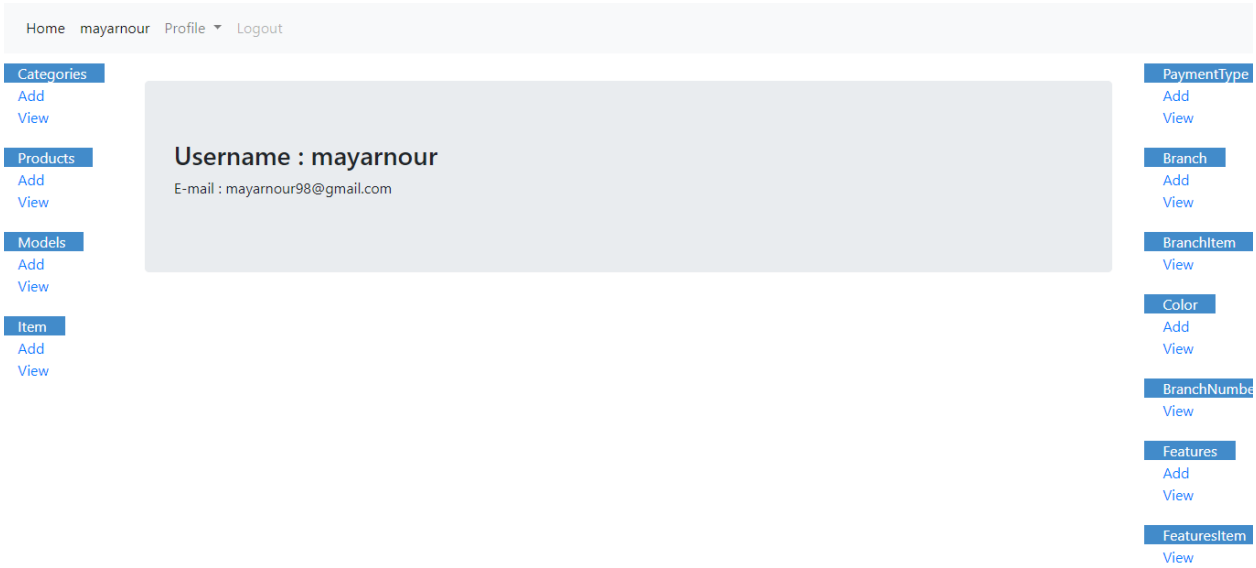
This sequence diagram is for a main use case in our program which is “add new item”. When the user selects a certain category all products with their models are displayed so he could select a model and add it to his shopping cart and then can confirm his order, or add a like or comment or add to his wish list

3. Prototype of functional requirements that are not clear:

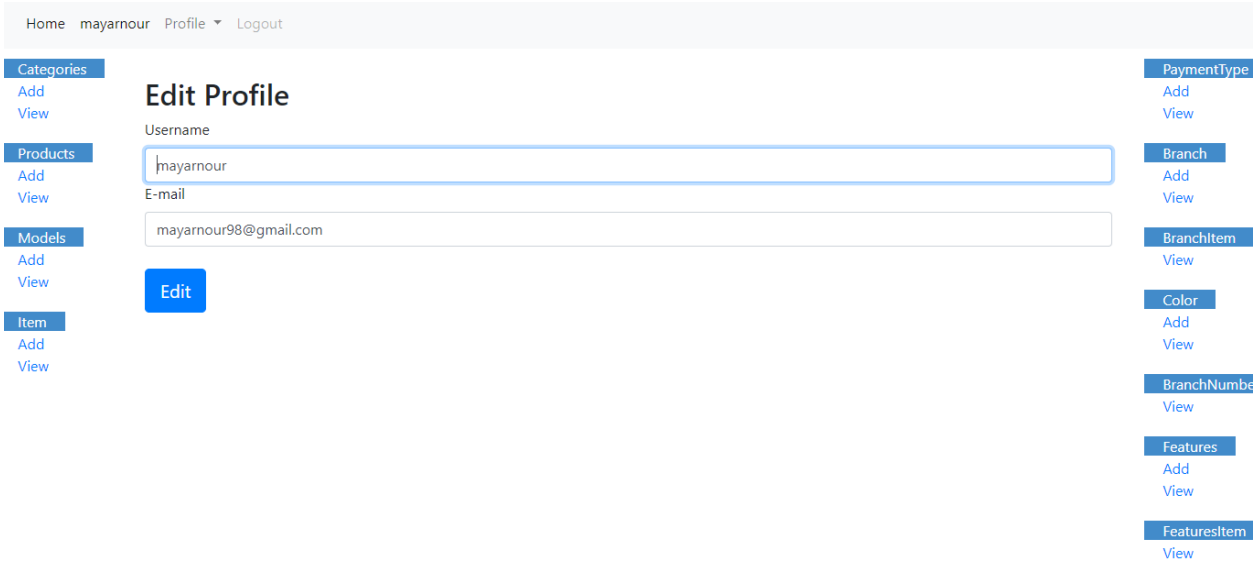
The admin part:



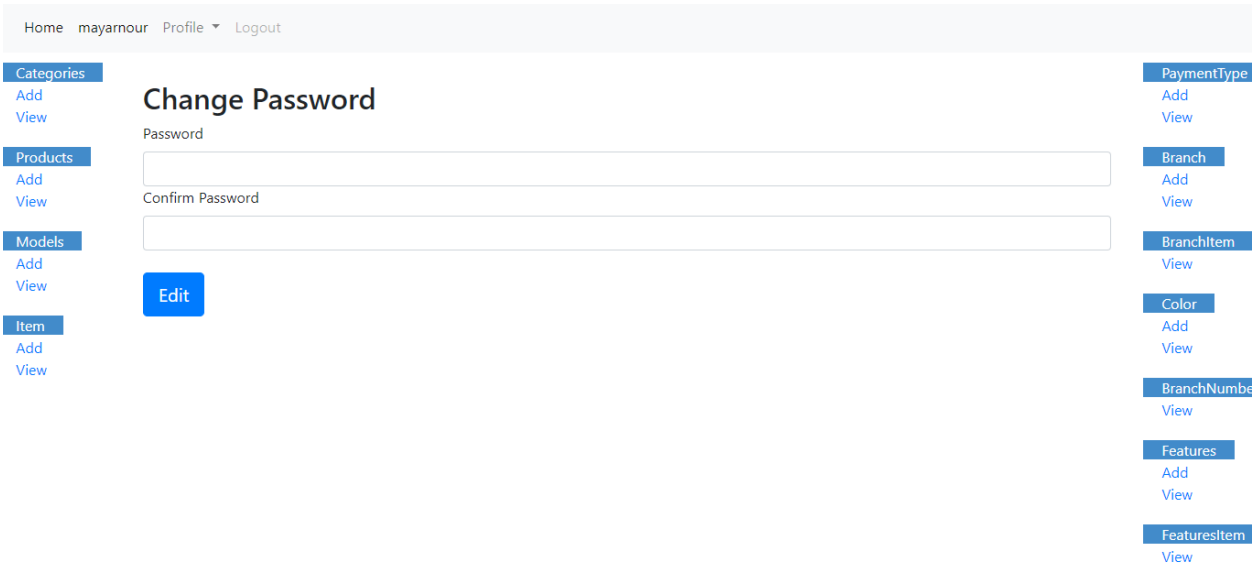
The admin enters the name and password to login. The admin logs into to add, delete, edit or view the database.



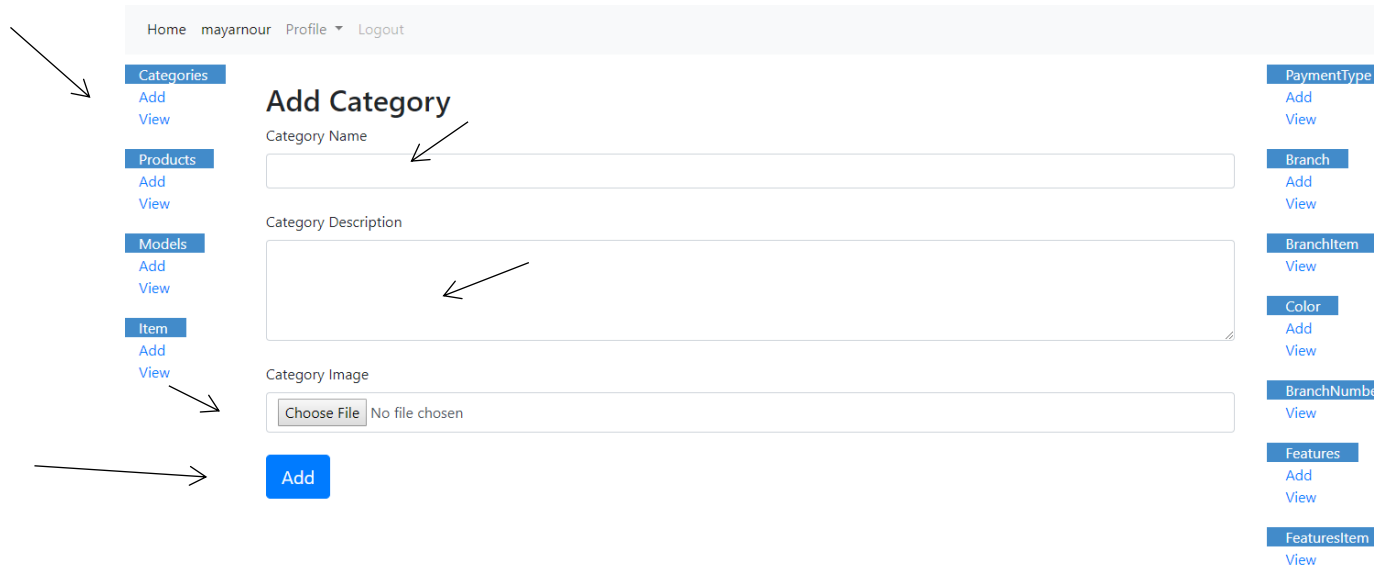
The login process isn't done successfully as long as the admin name and password isn't correct as saved in the database. When the admin logs in successfully, he can go to his profile to view his username and email.



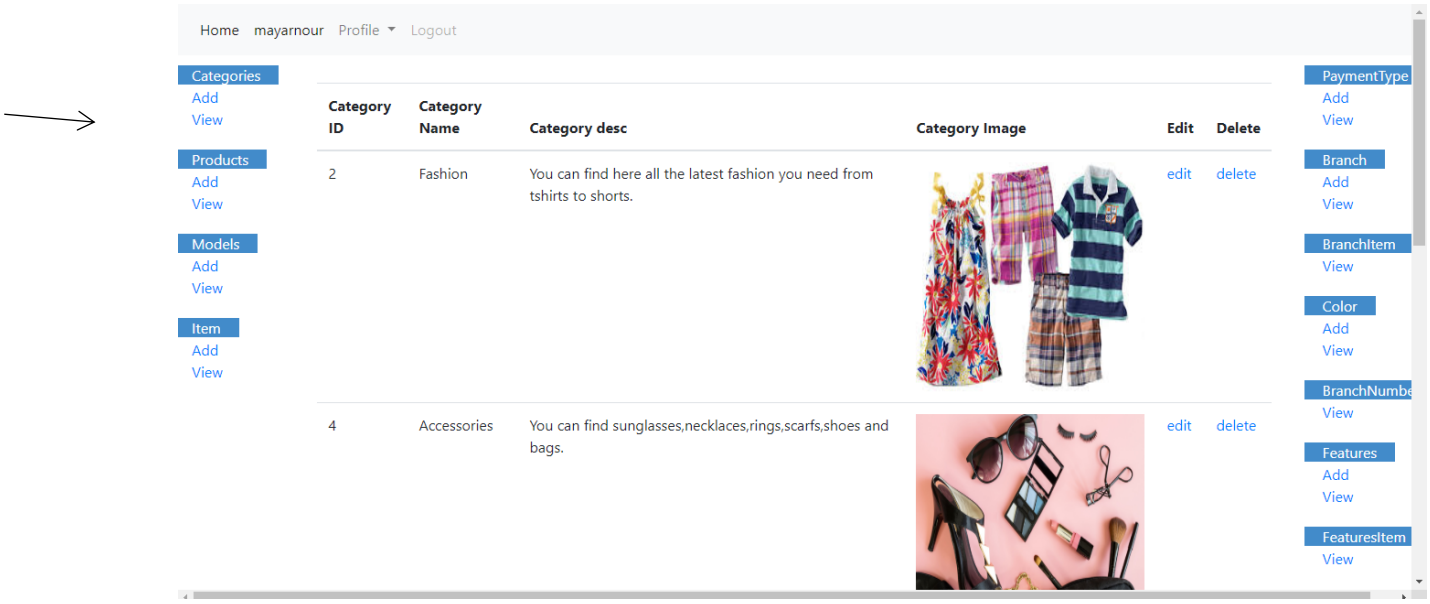
The admin can edit his profiel by changing the username or email or both.



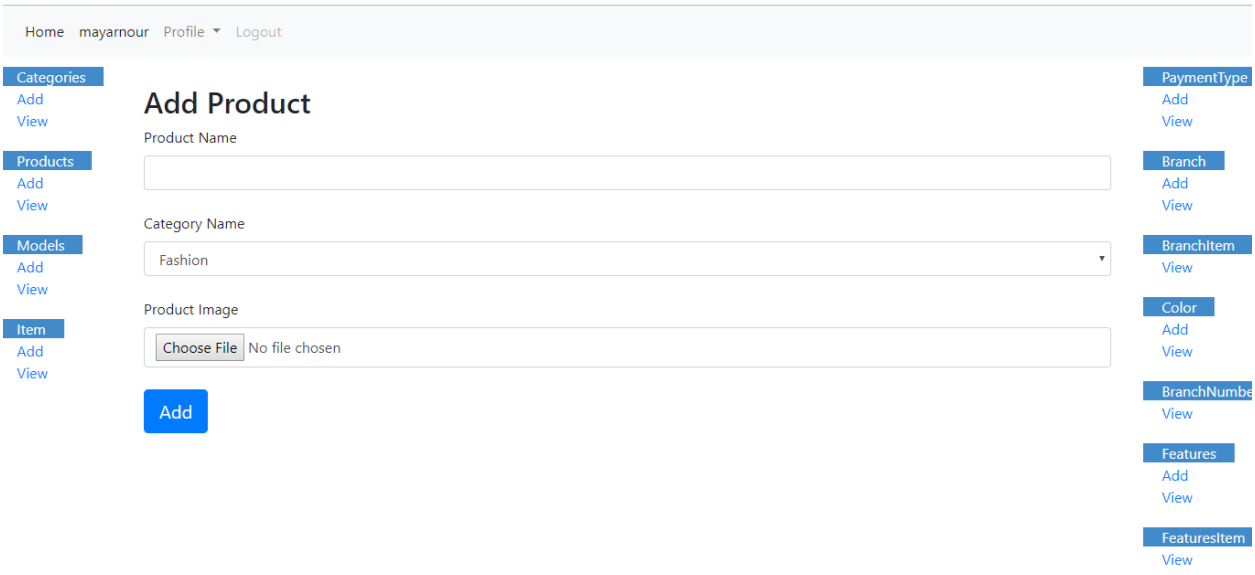
Also the admin can change his password by entering the new one and confirming it.



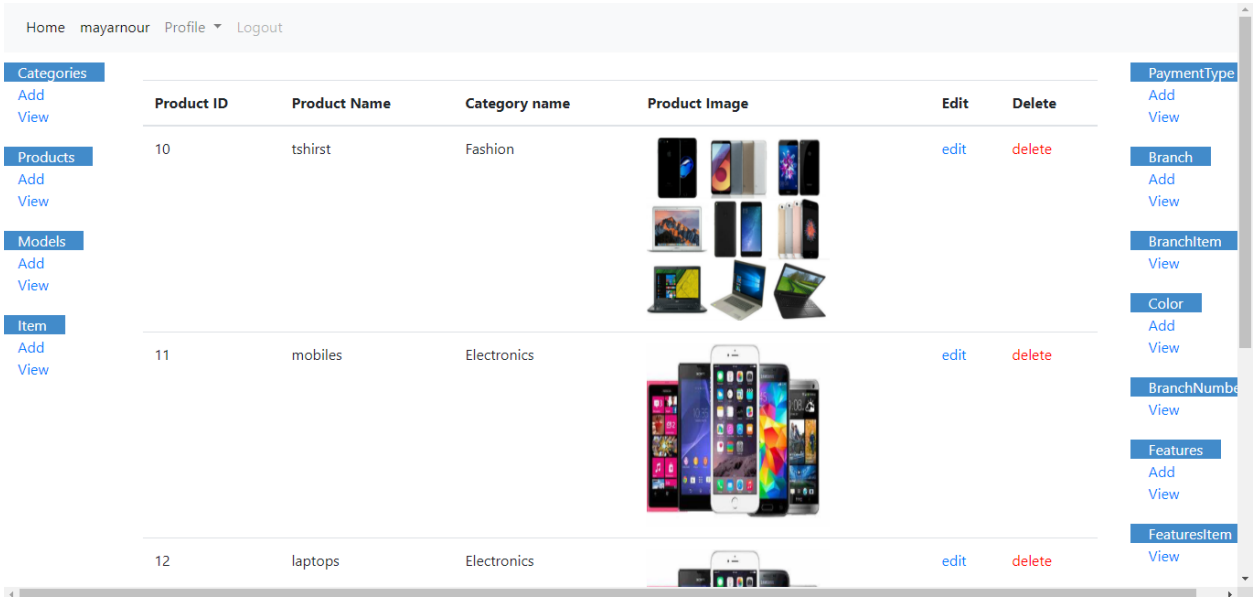
When the admin press the add button under the title categrory, He can add a new category to the database (electronics,fashion,cloths,…etc). Add to the categry it's name,description and a picure that describes it. When the Blue rectangular add buttin is pressed the category is added to the table category in the database.



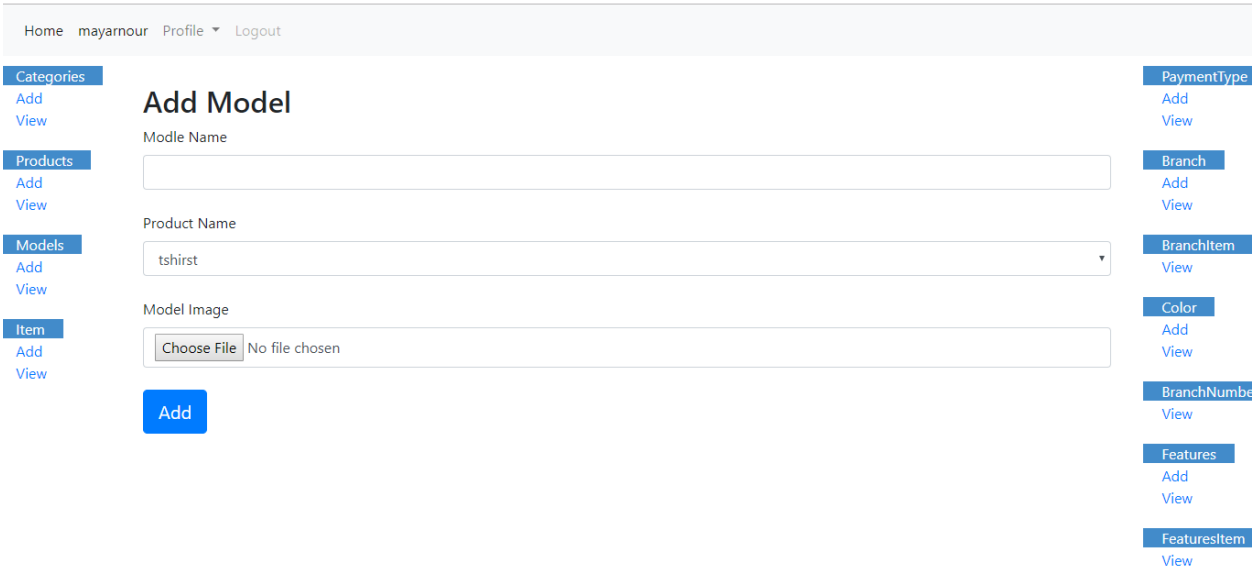
When the admin presses the view button under the title category,he can view the categories that are already in the database and can also edit or delete them.

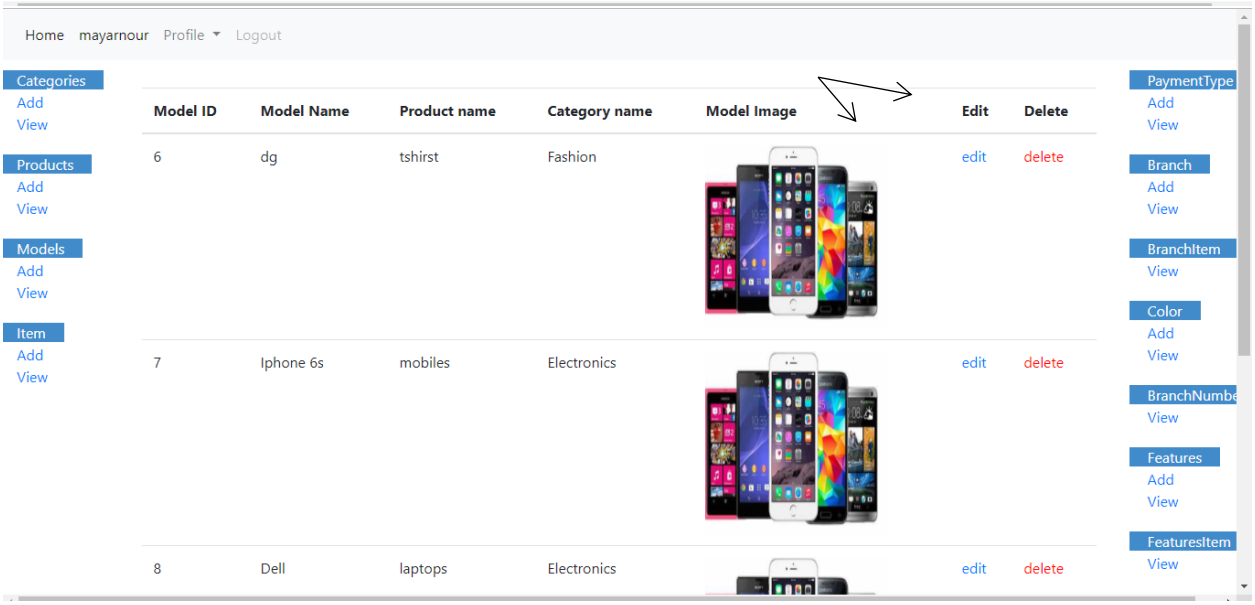


By clicking on the add button under products, the admin can add new product and choose which category it's in and also, he can add to that product an image and a name.

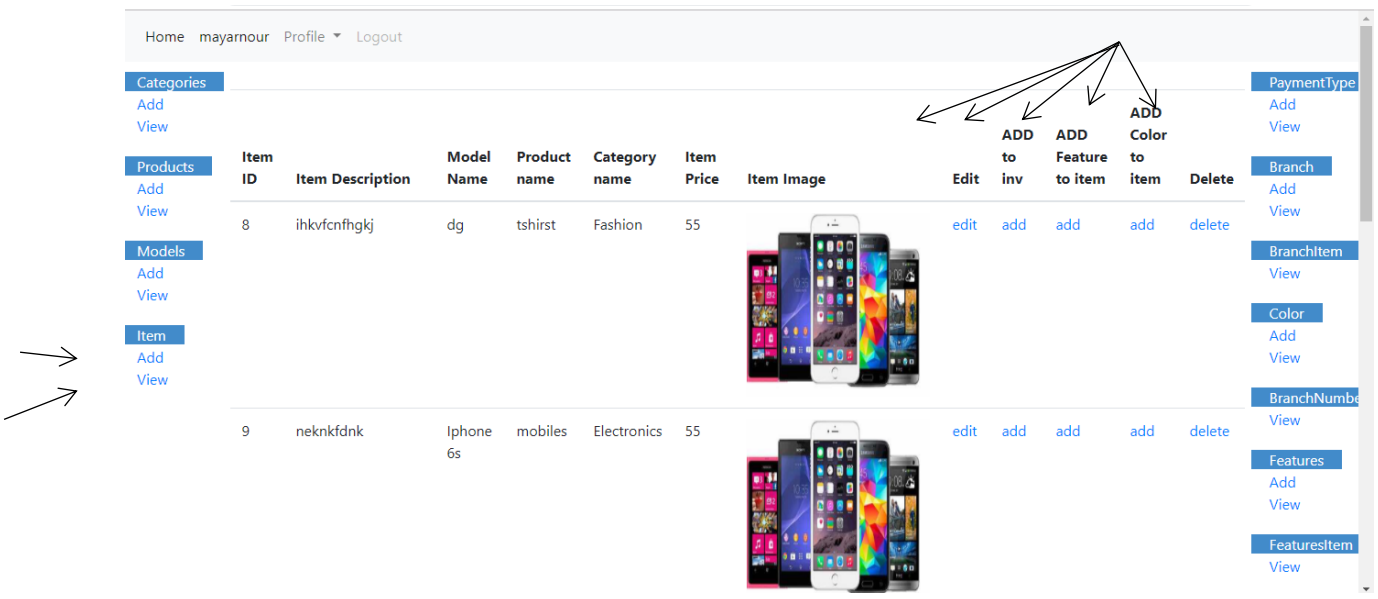


By clicking the view button, the admin can view all the products he has added to the table product in the database with its id, name, image and the category it's in, he can also edit or delete them.

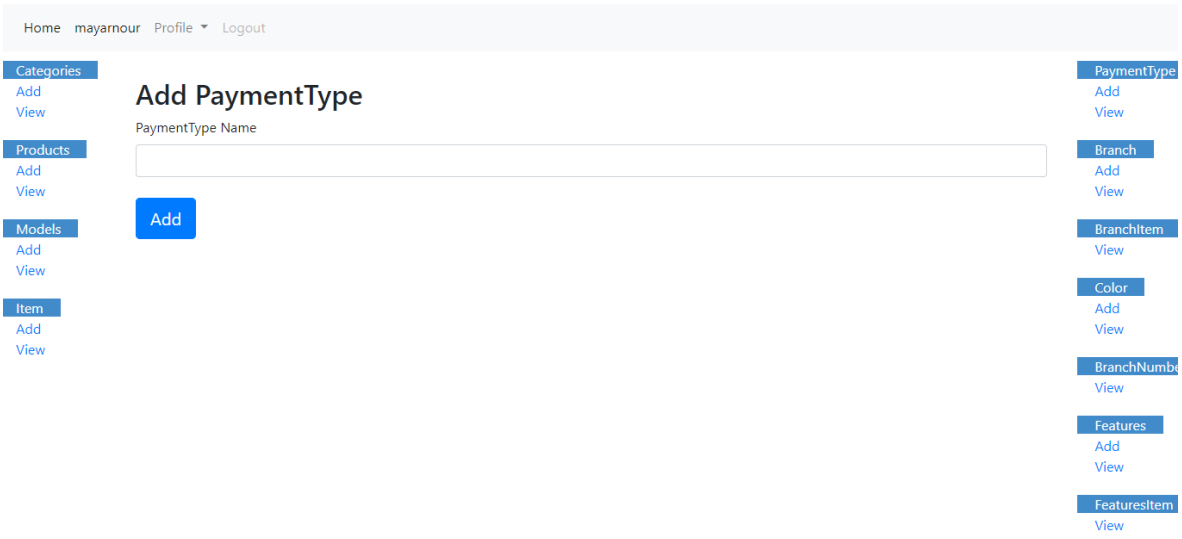




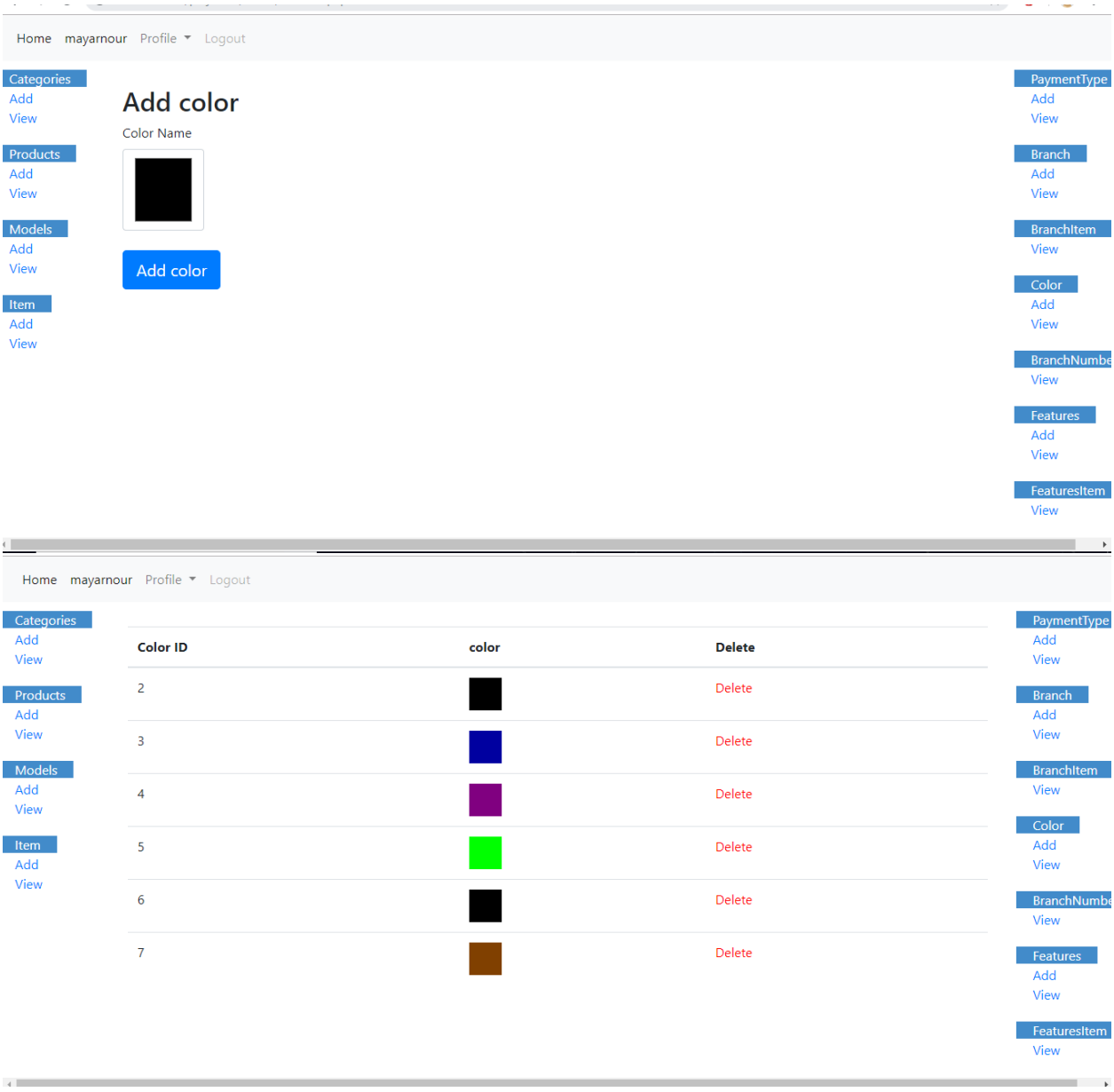
The admin can also add a model (iphone 6s) to a specific product and add an image to it. He can also then view what models he has added and edit them or delete them.



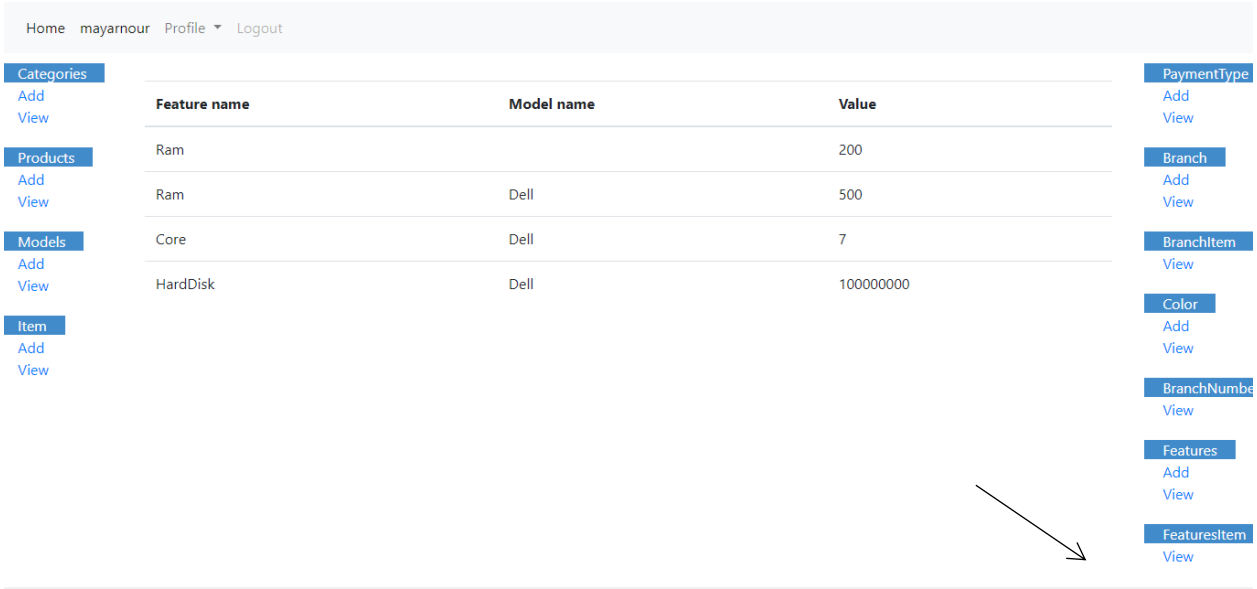
The admin can add an item with specific features and color. The admin decide which branch include this item and add it to the inventory. The admin can also edit or delete this item.



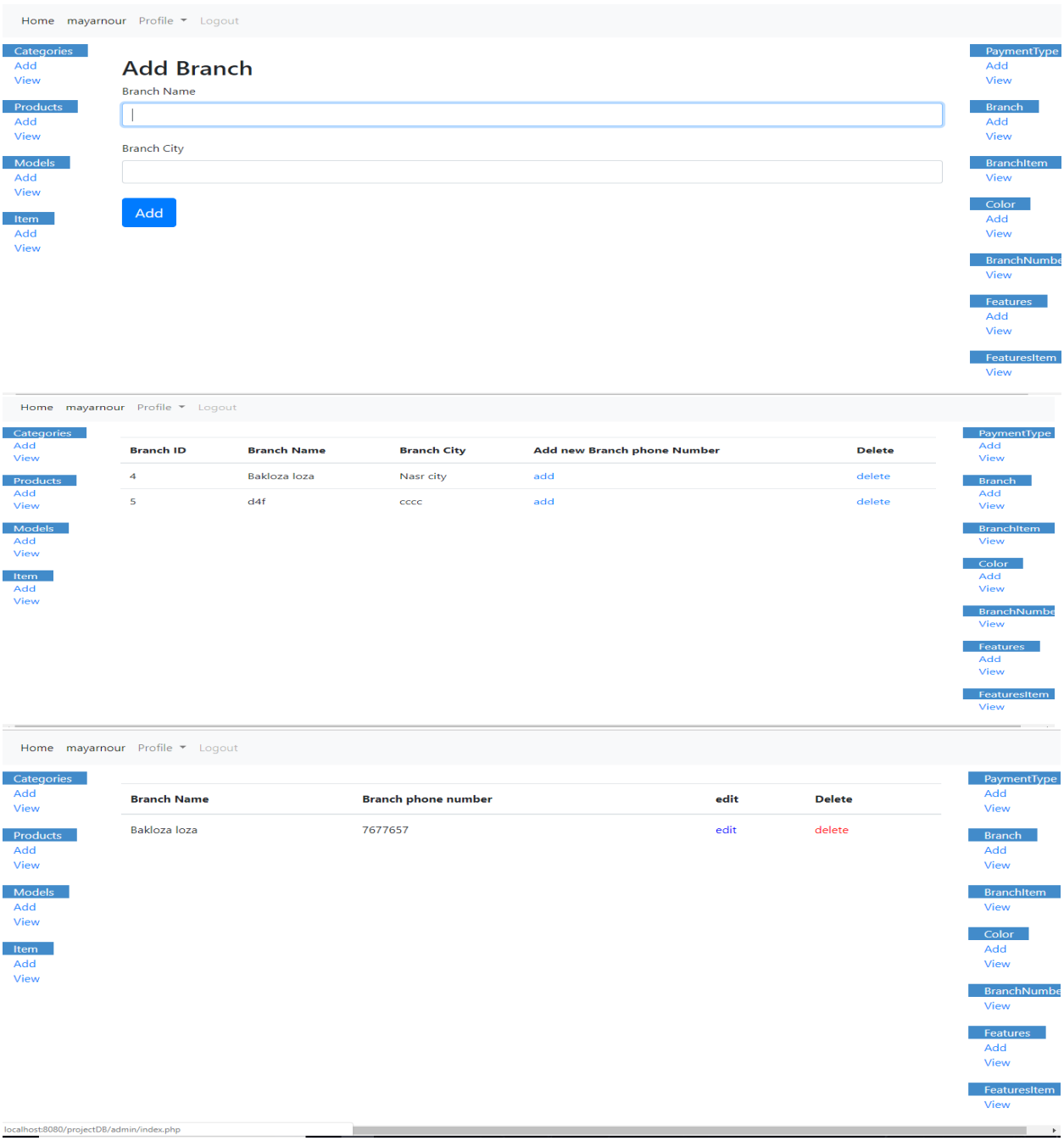
Admin can add different kinds and ways to pay. Customer can pay by credit card, cash or…etc. as long as the admin has added this type of payment to the database.



Admin can add new colors that is then added to items. The admin can also view each color with its id.



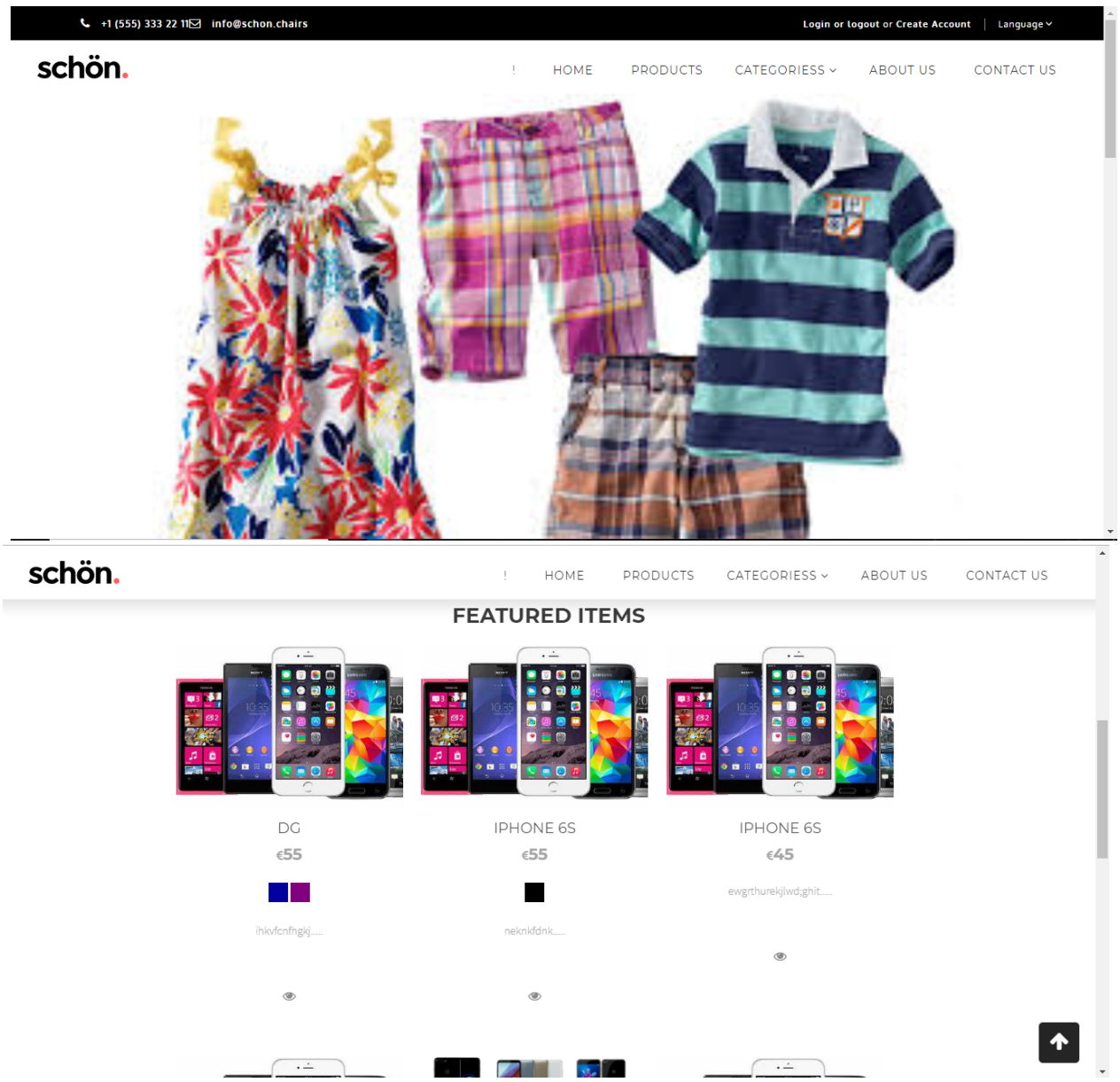
Admin can add features and then assign these features to items. Then the admin can view the features of each item by clicking in view under the title FeaturesItem.



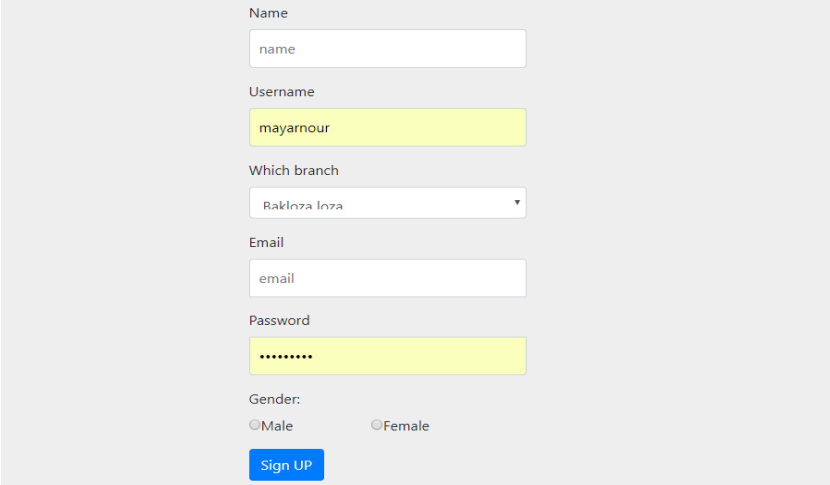
The admin can add new branches, add their names, cities and add multiple phone numbers to each branch. The admin can edit these details or delete a whole branch. The admin can also view each branch with the phone numbers it has.

The user part:

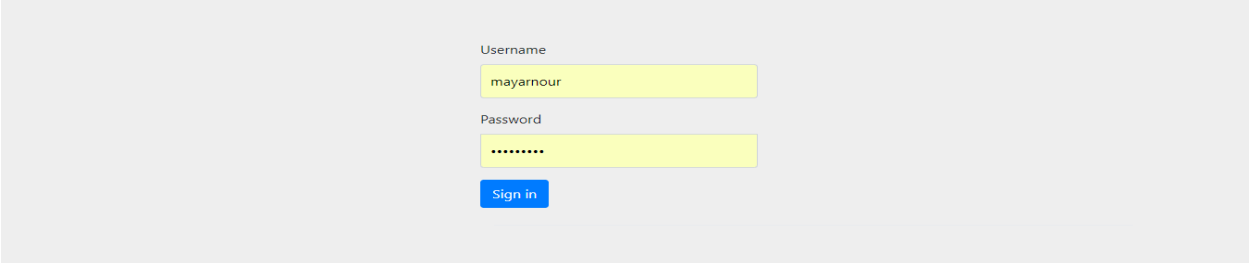
4.



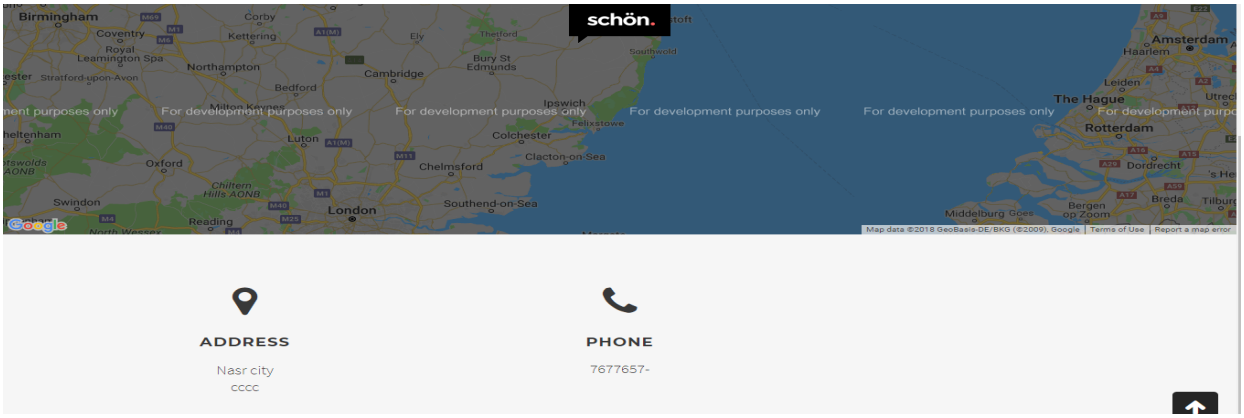
This is the home page for the website. The user can only search for different categories, products, models and view all the features items on the website without creating an account. By clicking the login button at the top the user can login to his account and if he hasn't created one yet he can create a new one by clicking on the create account button on the top.



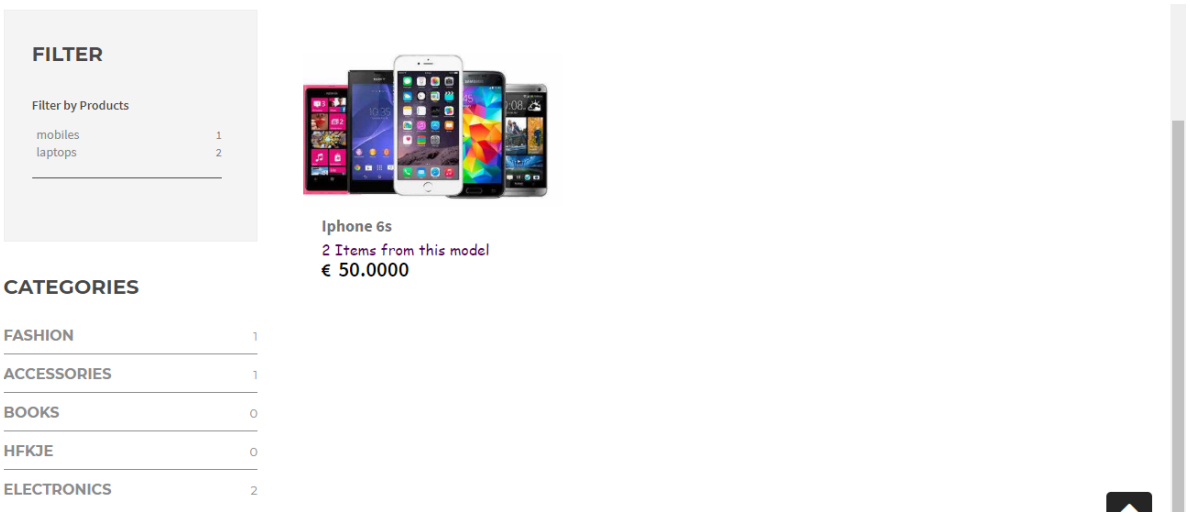
On clicking the create new account. The user can sign up for a new account with a new username,email and branch. The user should also choose which branch he will order his things from.



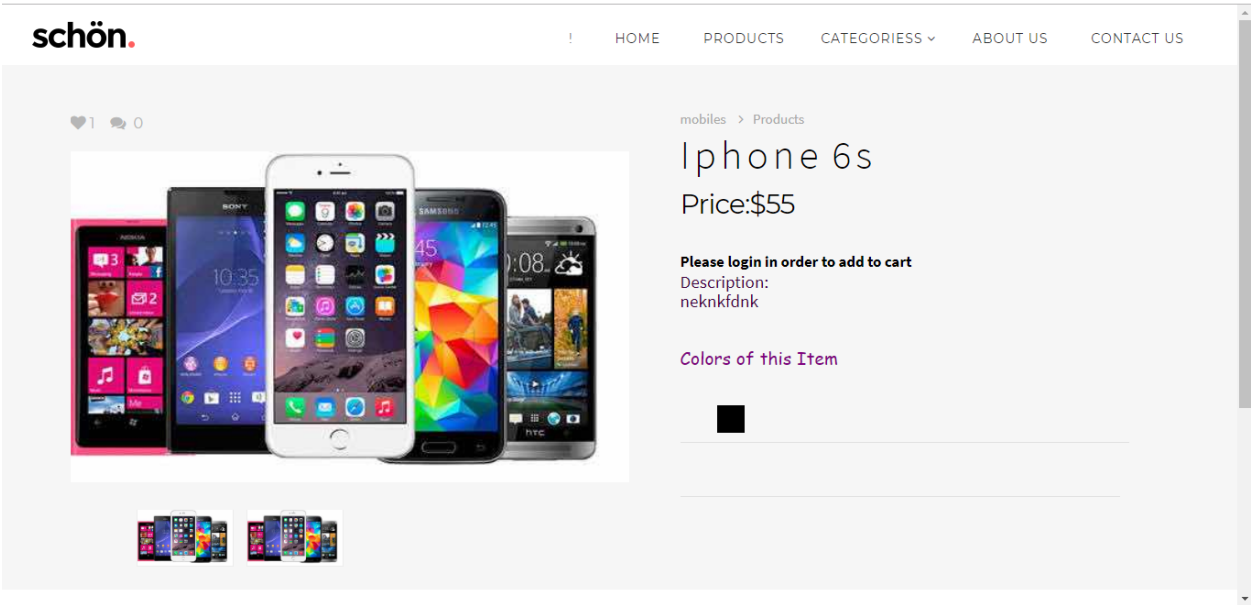
After creating an account the user can sign in to the website and start ordering, liking items, commenting on items and adding things to his wishlist.

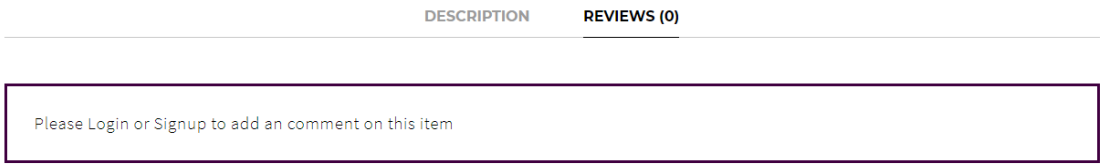


On clicking the contact us on the top, this page appears with the address of the branch the user dealing with and its phone number or numbers if available.

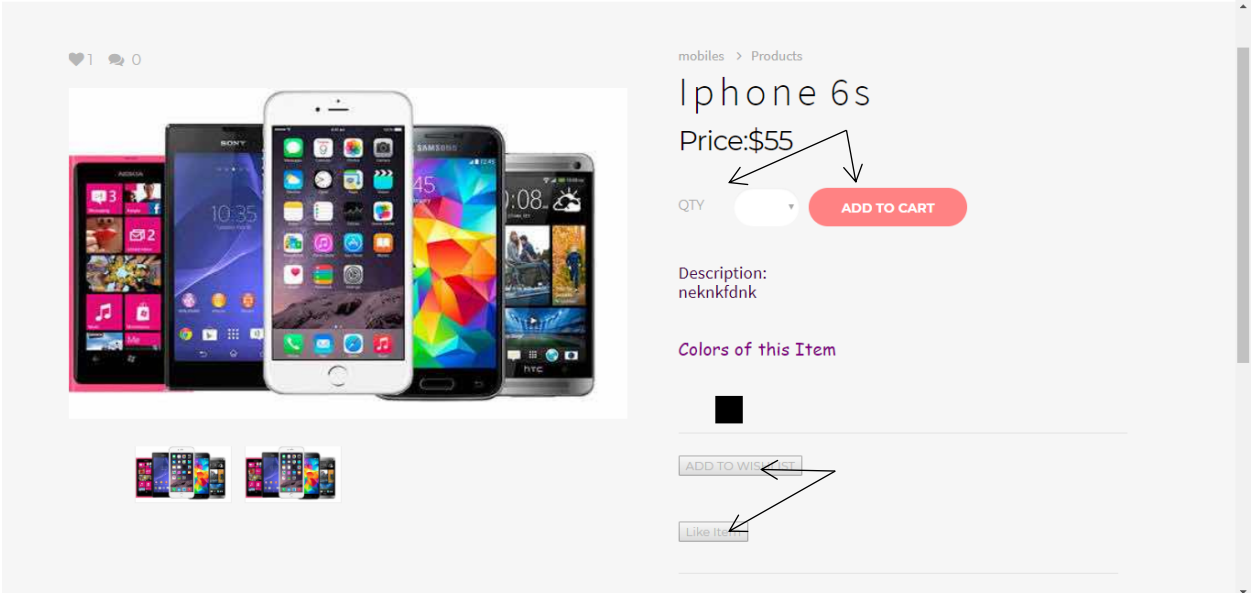


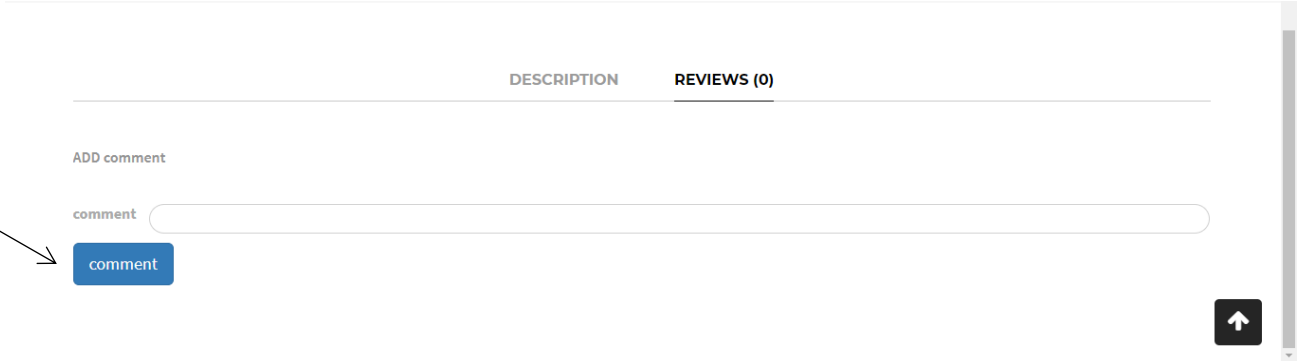
On choosing one of the categories (electronics for an example), a page with all the products and models of this category is shown. On the top left is a filter page that shows how many models is there from each product (one model in the mobiles which is the iphone 6s). On the bottom left there is all the categories and how many product in each category (electronics has 2 products, mobiles and laptops).





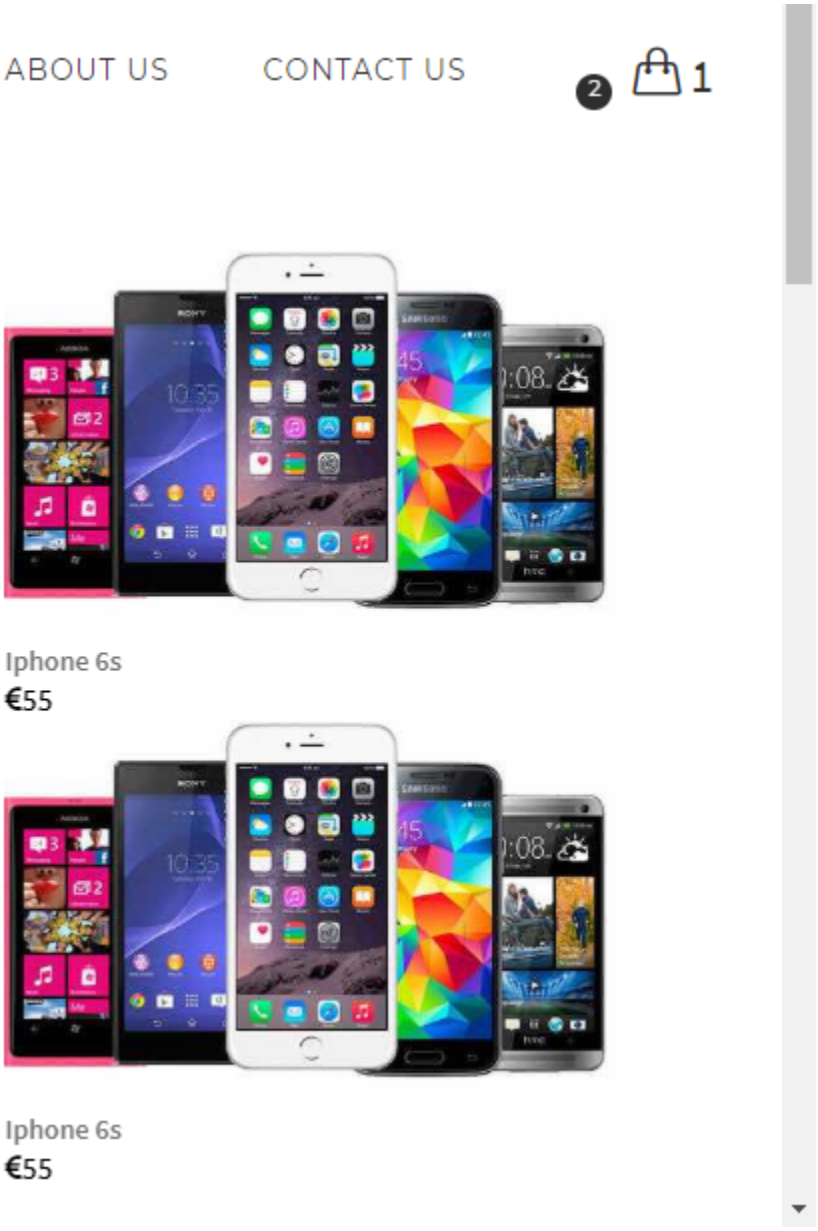
On clicking on the model shown above, a page appears showing the details of this model and different items from it with different features and colors (iphone 6s “model” has an item with 64GB memory with black color and another item with a 16GB memory with silver color. The user can only add items to the cart and comment on them when he is logged in.



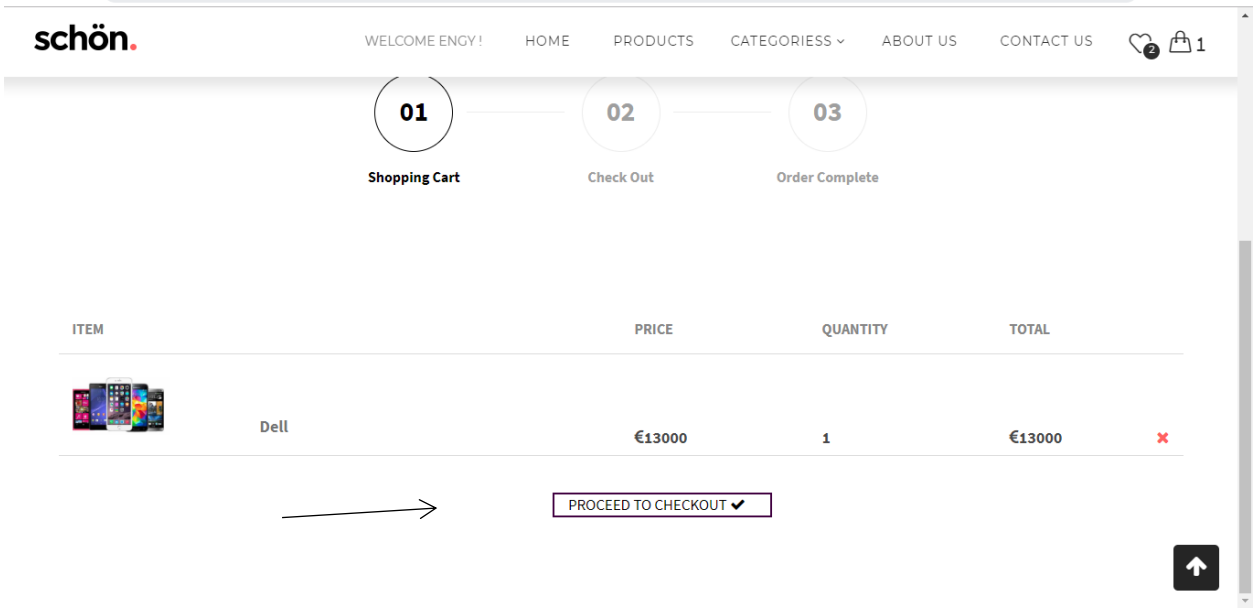


When the user is logged in he can item to the cart by determining how many quantity he wants and then click on the Add To Cart button. He can also add this item to his wish list by clicking on the ADD TO WISHLIST button. Moreover, the

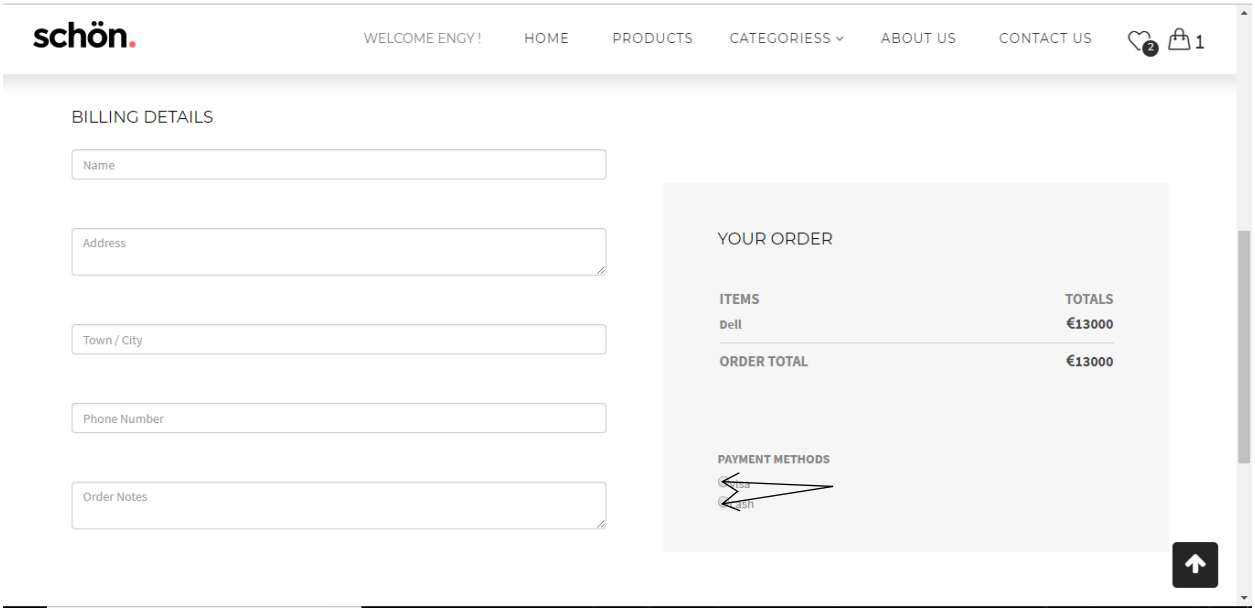
user once he is logged in he can add comments on any item he wants. Also, he can like this item, each item has a number of likes shown on the top right.



On clicking on the **heart icon** on the top right of the page, the wish list of the user appears with all the items he has added to his wish list. The wish list is a list with items the user likes and wishes to buy later.



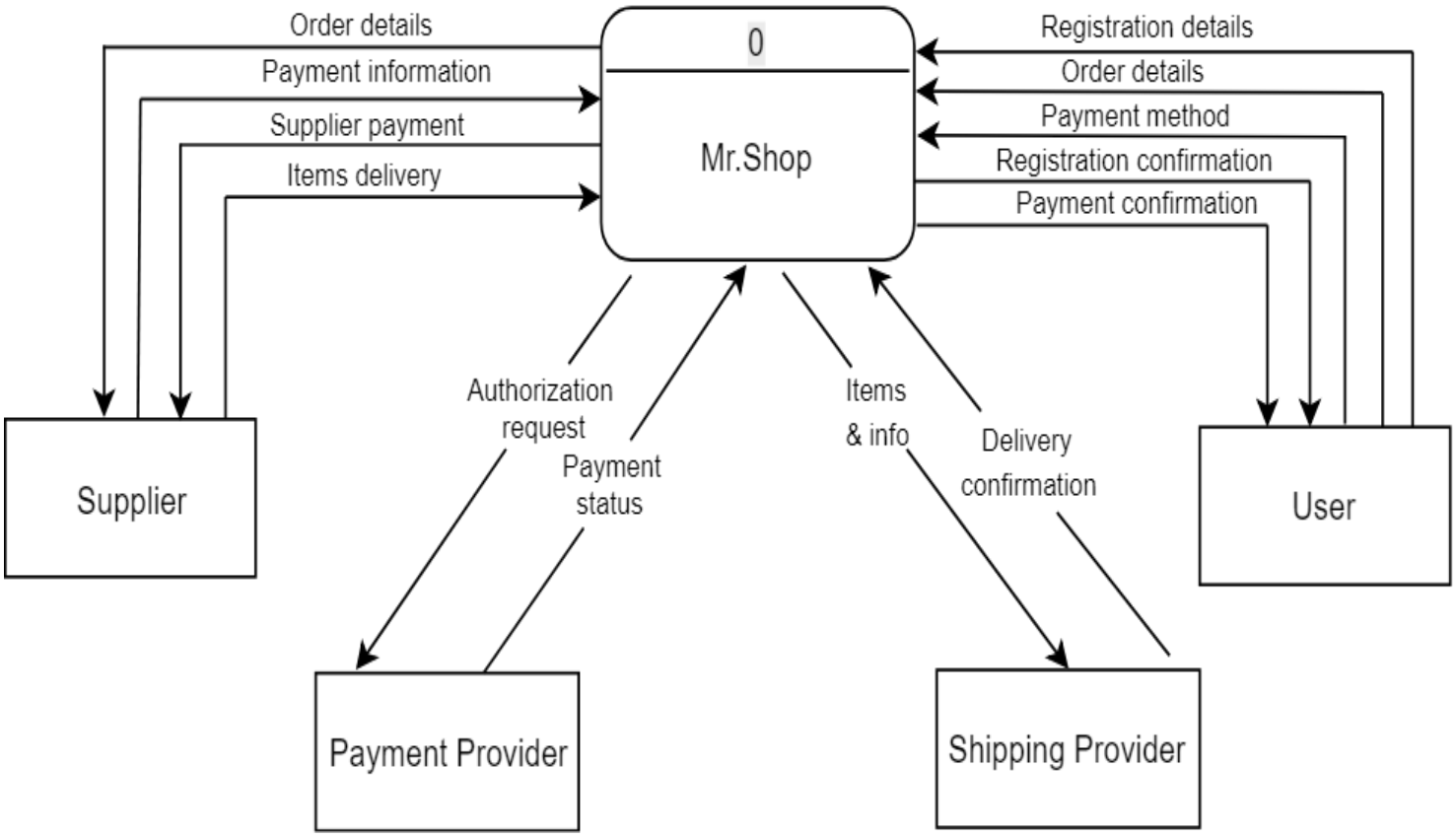
When the user clicks on the **bag icon** on the top right of the screen, a page with a list of all the items the user had added to his cart, each item has its price and if more than one item is added the quantity is viewed and the total price for this item. The user can remove items from the cart if he doesn't want to buy it, when he is ready he presses the proceed to checkout button



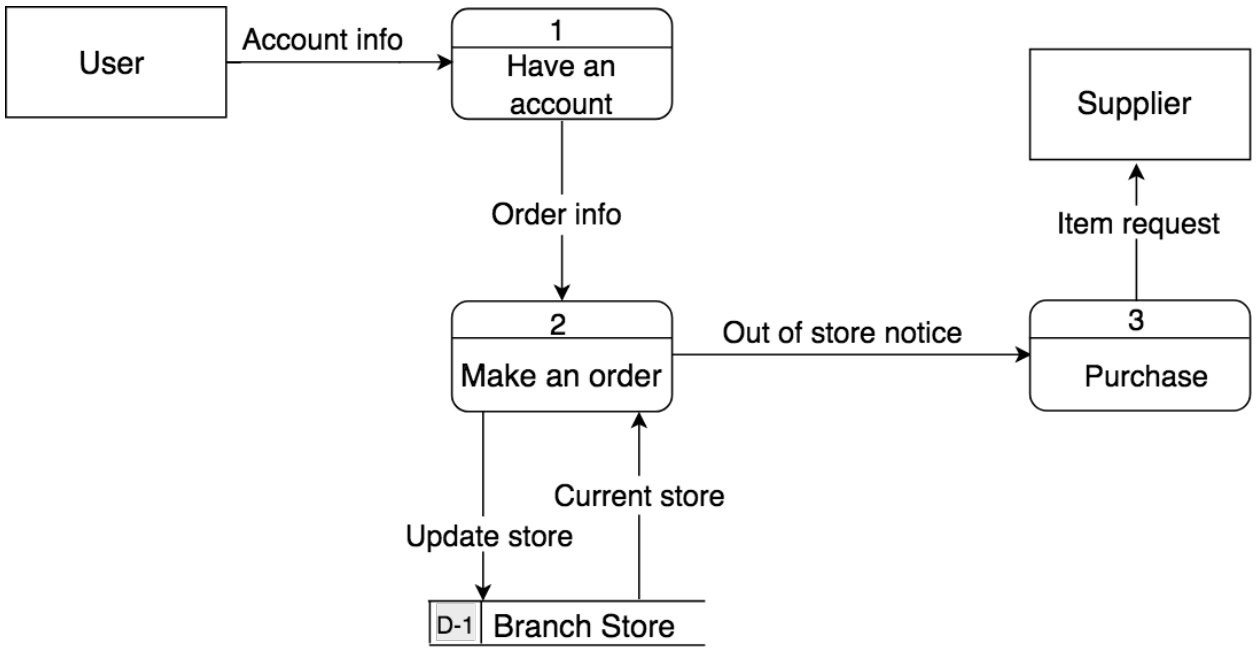
When the proceed to checkout button is pressed, this page appears with the order bill details. The user should fill his name, address to be shipped to, city, phone number and any notes needed. Then he chooses whether he will pay by visa or cash.

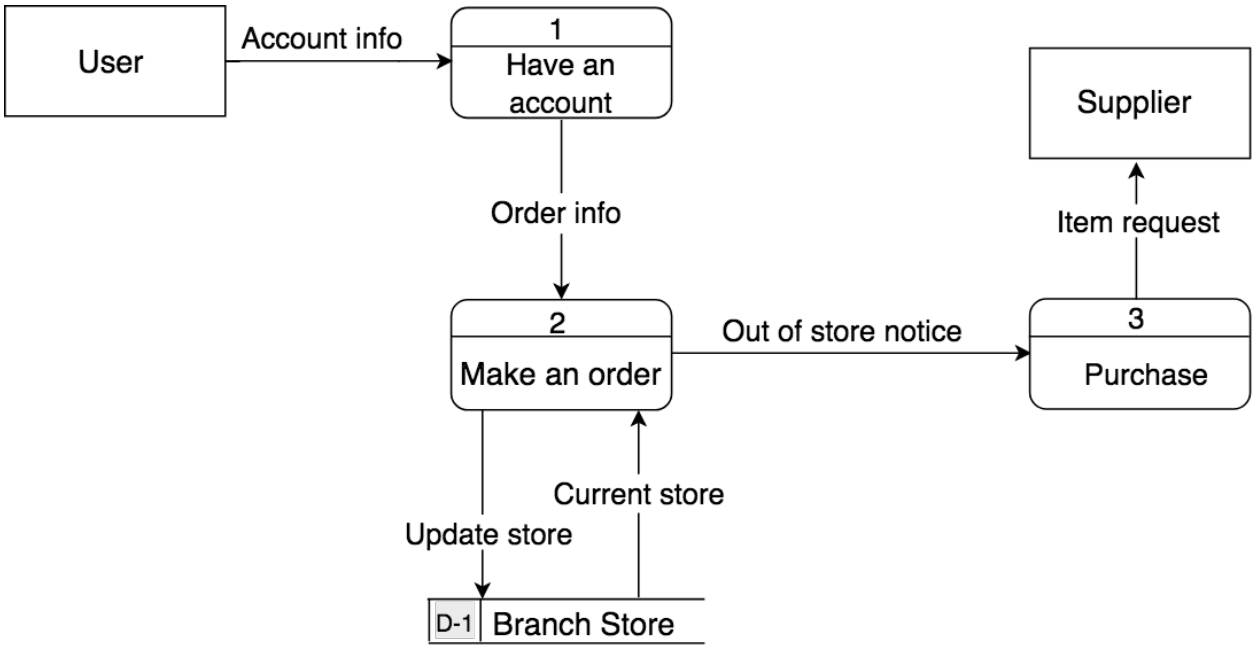
4. 3 DFD Diagram

a. Context level

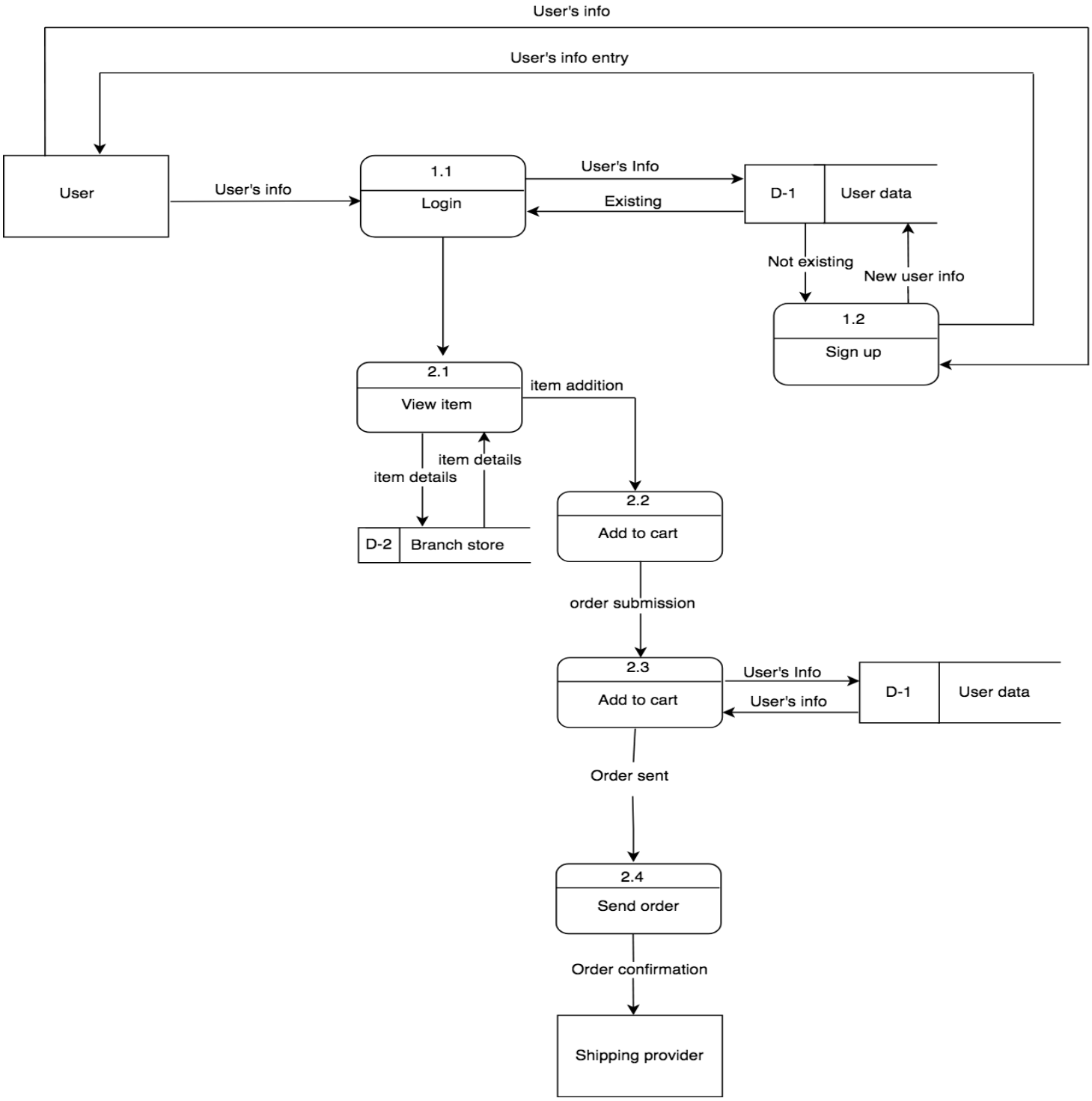


b. Level 0

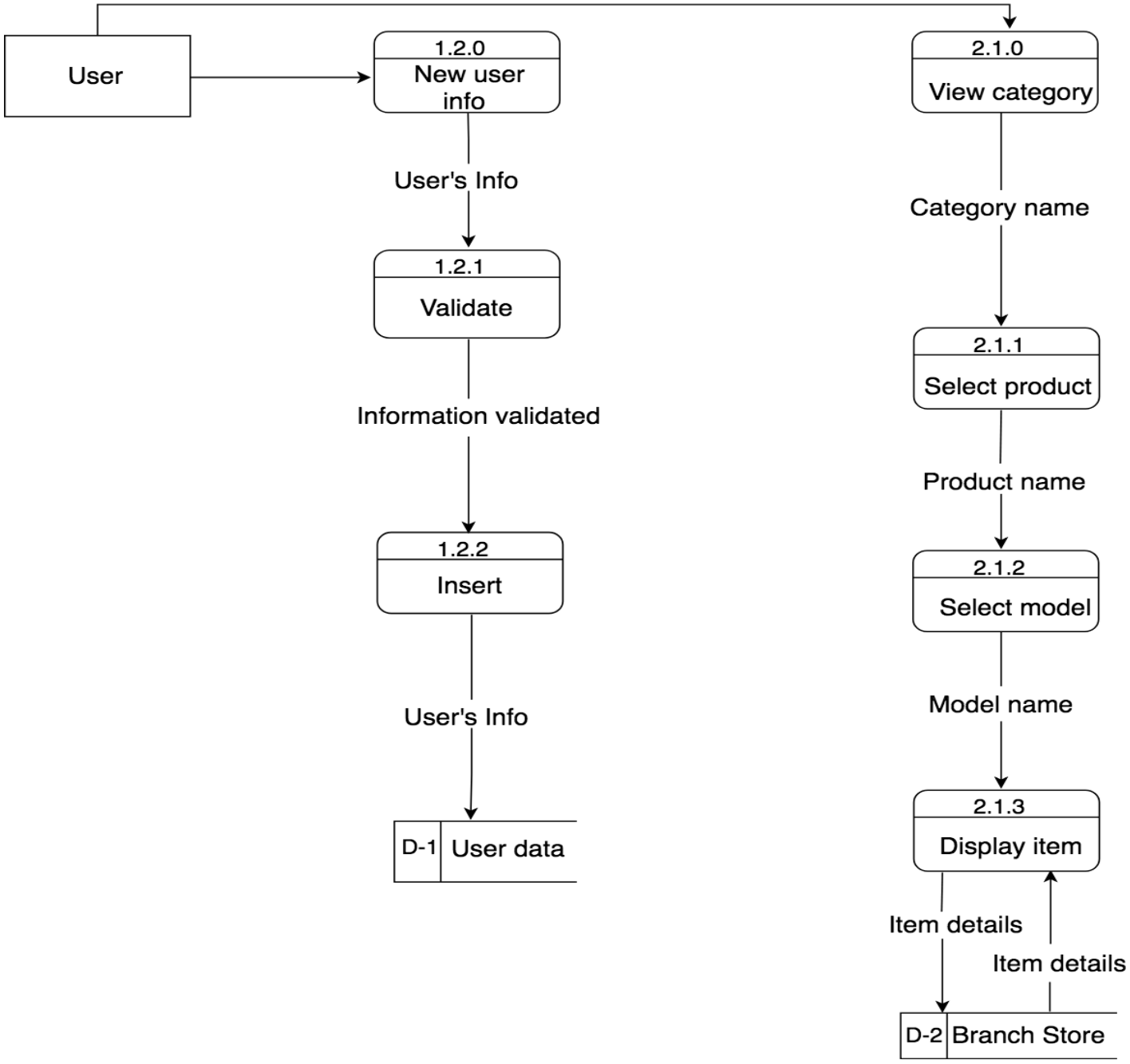

User
Account info
1
Have an account
Supplier
Order info
Item request
2
Make an order
Out of store notice
3
Purchase
Current store
Update store
D-1
Branch Store


User
Account info
1
Have an account
Supplier
Order info
Item request
2
Make an order
Out of store notice
3
Purchase
Current store
Update store
D-1
Branch Store

c. Level 1


User's info
User's info entry
User
User's info
1.1
Login
User's Info
Existing
D-1
User data
Not existing
New user info
1.2
Sign up
2.1
View item
item addition
item details
item details
D-2
Branch store
2.2
Add to cart
order submission
2.3
Add to cart
User's Info
User's info
D-1
User data
Order sent
2.4
Send order
Order confirmation
Shipping provider

d. Level 2


User
1.2.0
New user info
User's Info
1.2.1
Validate
Information validated
1.2.2
Insert
User's Info
D-1
User data
2.1.0
View category
Category name
2.1.1
Select product
Product name
2.1.2
Select model
Model name
2.1.3
Display item
Item details
Item details
D-2
Branch Store

5. Components:

1) User:

This component is responsible for saving the account information as his name, address, email; and for handling the user's operations including his favorite list, his orders, adding items to his shopping cart.

This component is considered as *White Box* component because it is open to inspection and updating the user's information.

component because it is open to inspection and modification of the product.

2) Management:

This component is responsible for adding new products into different categories and prices for each one to be shown in the website.

This component is considered as *White Box* component because it is open to inspection and modification of the product.

3) Data:

This component is considered as the inventory of the website as it saves how many items for each product and detail of it as (color, size, features …); And also saving the information of each branch as (address, phone number, email).

This component is considered as *White Box* component because it is open to inspection and modification of the product items and updating the store's information.

4) Payment:

This component controls the method of payment of each user to complete his orders; There are many ways of payment, so the user can choose from paying by visa or Fawery or Vodafone cash or cash on delivery.

This component is considered as *Grey Box* because part of the internals may be inspected & limited modification is allowed the payment method is known by the website system to ensure the safety of the order, but the internal details may be not known as the bank transactions

5) Shipping provider:

This component handles the final step of the order; By saving two important data which are the shipping address and the order details (expected arrival time) and finally deliver the order to the user.

This component is considered as *Black Box* because we didn't know how the order was delivered.

6) Supplier information service:

This component is responsible for selling the product to the admin.

This component is considered as *Black Box* because we didn't know how the supplier supply the products.

6. Detailed analysis of the components:

- **User**:
  - **Component Characteristic:**
    - **Independent:**

      The user component is independent; it doesn't depend on any other component to be created or to make any modifications on data. In case it needs it should use the required interfaces to have it.
    - **Deployed:**

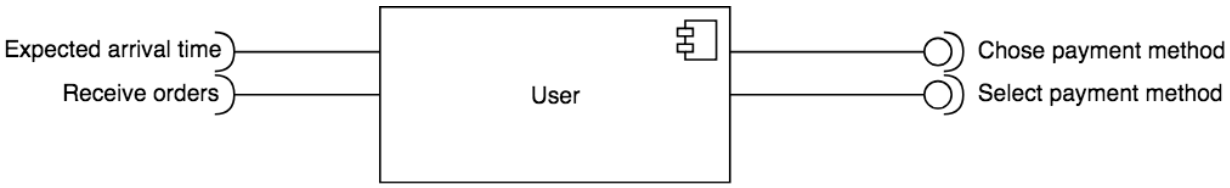
      The user component can do its functionality anytime when deployed and operate as stand-alone entity on some platform that implement the user component model.
    - **Documented:**

      The user component is fully documented with the required and provides interfaces to simplify the transactions with it. The semantics of all component interfaces must be specified
    - **standardized:**

      The user component has to be standard for example it has to fill in a form with his first and last name, address, email, phone number. otherwise the account wouldn't be created (to confirm some standardized component model that define the interfaces, metadata, documentation, composition and deployment of the user component).
    - **Composable:**

      All external interactions of the user component must take place through the interfaces (required and provided) that will be mentioned in the next point. In addition; the user component must provide access about itself such as account information (name, address, age...) using methods and attributes.

- **Component Interfaces**



  - Required interface:
    The user component requires from the shipping provider component the expected arrival time and receive his order.
  - Provided interface:
    The user component provides his account information and specify his own way of payment.

- **Reusability:**

  The user component is reusable because this component could be used in any program which asks user to have an account for any reason for example Facebook, Twitter.

- **Component Adaptation**
  - Wrapper. A wrapper is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the user could be able to make and receives orders.

- **Component Identification Issues**
  - Trust: is trusted because it is a component internally created
  - Requirements. Different groups of components will satisfy different requirements.

- **Component specification**

  This component is responsible for saving the account information as his name, address, email; and for handling the user's operations including his favorite list, his orders, adding items to his shopping cart.

- **Management:**
  - **Component Characteristic:**
    - **Independent:**

      The management component is independent; it doesn't depend on any other component to be created or to make any modifications on data.
    - **Documented:**

      The management component is fully documented with the required and provides interfaces to simplify the transactions with it.
    - **Deployed:**

      The management component can do its functionality anytime when deployed.
  - **Component Interfaces**

      - Required interface:

        The management component requires from supplier information service component the items to be able to add it to his website with the prices.
      - Provided interface:

        The management component provides the data component the products and all its details.

- **Reusability**:

  The management component isn't reusable because every admin has to do different tasks in each website, so it is not common to use. So, it hasn't to be reusable.

- **Component Adaptation**
  - Wrapper. A wrapper is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the admin could be able always to add new items and upgrade his website.
- **Component Identification Issues**
  - Trust: is trusted because it is a component internally created
  - Requirements. Different groups of components will satisfy different requirements.
- **Component specification**
  - This component is responsible for adding new products into different categories and prices for each one to be shown in the website.

- **Data**:
  - **Component Characteristic:**
    - **Independent**:

      The data component is independent; it doesn't depend on any other component to be created or to make any modifications on data. In case it needs it should use the required interfaces to have it.

    - **Deployed**:

      The data component can do its functionality anytime when deployed. And operate as stand-alone entity on some platform that implement the branch component model.

- **Documented:**

  The data component is fully documented with the required and provides interfaces to simplify the transactions with it. And contains a detailed documentation of all branches with its storage to maintain safety and prevents missing items. The semantics of all component interfaces must be specified

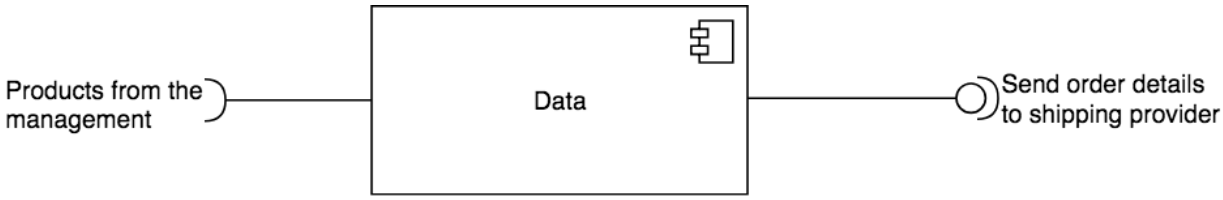
- **Composable:**

  All external interactions of the data component must take place through the interfaces (required and provided) that will be mentioned in the next point. In addition; the data component must provide access about itself such as branch information concerning the name and ID and address and the storage (number of items and description of each one) using methods such and attributes.

- **Standardized:**

  The data component has to be standard for example it must have a unique user name and ID to save the storage of this branch and information about all items it contains … so that it could confirm some standardized component model that define the interfaces, metadata, documentation, composition and deployment of the branch component).

- **Component Interfaces**



  - **Required interface:**

    The data component requires from admin component the products to be added to each branch storage.

- **Provided interface:**

  The data component provides the shipping provider component the order details to be delivered to the user(customer)

- **Reusability:**

  The data component is reusable because this component could be used in any program which want to develop an inventory to store anything it needs.

- **Component Adaptation**
  - Wrapper. is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the branch keep track of all the items in the branches and the information related to each branch.
- **Component Identification Issues**
  - Trust: is trusted because it is a component internally created
  - Requirements. Different groups of components will satisfy different requirements
- **Component specification**
  - This component is considered as the inventory of the website as it saves how many items for each product and detail of it as (color, size, features …); And also saving the information of each branch as (address, phone number, email).

➢ **Payment**:

- **Component Characteristic:**
  - **Standardized:**

    The payment component has to be standard for example it has to follow global standards to ensure security for each the user and the rights of the website. … so that it could confirm some standardized component model that define the interfaces, metadata,

documentation, composition and deployment of the payment component.

- **Documented**:

The payment component is fully documented with the required and provides interfaces. The semantics of all component interfaces must be specified

- **Independent**:

The payment component is independent; it doesn't depend on any other component to be created or to make any modifications on data. In case it needs it should use the required interfaces to have it.

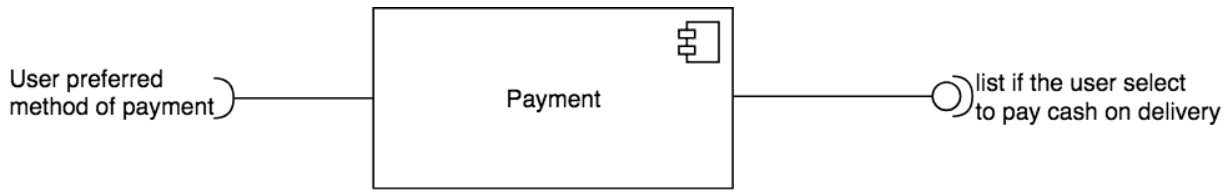
- **Deployed**:

The payment component can do its functionality anytime when deployed. And operate as stand-alone entity on some platform that implement the payment component model.

- **Composable:**

All external interactions of the payment component must take place through the interfaces (required and provided) that will be mentioned in the next point. In addition; the payment component must provide access about itself such as payment ID and method of payment chosen by the user using methods and attributes.

- **Component Interfaces**



- **Required interface:**

  The payment component requires from the user component information about his preferable payment method.

- **Provided interface:**

  The payment component provides the shipping provider information about if the user wants to pay cash on delivery or not.

- **Reusability:**

  The payment component is reusable because this component could be used in any program which sell products to users online.

- **Component Adaptation**
  - Wrapper. A wrapper is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the payment method is assigned for all orders.
- **Component Identification Issues**
  - Trust: is not completely trusted because it is a component bought through an API for each payment method as Vodafone cash, visa, fawry ..
  - Requirements. Different groups of components will satisfy different requirements.
- **Component specification**
  - This component controls the method of payment of each user to complete his orders; There are many ways of payment, so the user can choose from paying by visa or Fawery or Vodafone cash or cash on delivery.

- **Shipping Provider:**
  - **Component Characteristic:**
    - **Independent:**

      The shipping provider component is independent; it doesn't depend on any other component to be created or to make any modifications on data. In case it needs it should use the required interfaces to have it.
    - **Deployed:**

      The shipping provider component can do its functionality anytime when deployed. And operate as stand-alone entity on some platform that implement the shipping provider component model.
    - **Standardized**:

      The shipping is following global standards for delivering products to users in an estimated arrival time. so that it could confirm some standardized component model that define the interfaces, metadata, documentation, composition and deployment of the shipping provider component.
    - **Documented:**

      The shipping provider component is fully documented with the required and provides interfaces to ensure the contract between the website (admin) and the shipping provider to ensure that the order will be delivered on time and for the right users and having good feedbacks at the end. The semantics of all component interfaces must be specified
    - **Composable:**

      All external interactions of the payment component must take place through the interfaces (required and provided) that will be mentioned in the next point. In addition; the shipping provider component must provide access about itself such as ID, name and

arrival time date and shipping address using methods and attributes.

- **Component Interfaces**

  - **Required interface:**

    The shipping provider component requires from the management component the order information (shipping information)

  - **Provided interface:**

    The shipping provider component provides the user component his order to complete the shipping process.

- **Reusability:**

  The shipping provider component is reusable because this component could be used in any program which must deliver a user a certain product

- **Component Adaptation**

  - Wrapper. A wrapper is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the user could be able to make and receives orders.

- **Component Identification Issues**

  - Trust: is not completely trusted because it is a component is bought. It represent different shipping provider company as Aramex ..
  - Requirements. Different groups of components will satisfy different requirements.
  -

- **Component specification**
  - This component handles the final step of the order; By saving two important data which are the shipping address and the order details (expected arrival time) and finally deliver the order to the user.

- **Supplier information service:**
  - **Component Characteristic:**
    - **Independent:**

      The supplier information service component is independent; it doesn't depend on any other component to be created or to make any modifications on data. In case it needs it should use the required interfaces to have it.
    - **Deployed:**

      The supplier information service component can do its functionality anytime when deployed. And operate as stand-alone entity on some platform that implement the supplier component model.
    - **Documented:**

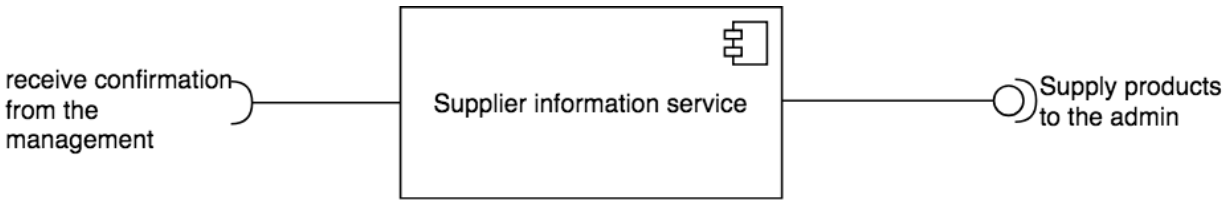
      The supplier information service component is fully documented with the required and provides interfaces to save all products taken from a certain supplier to be added by the admin in the website. The semantics of all component interfaces must be specified
    - **Standardized**:

      The supplier information service is following global standards for selling products to admins. so that it could confirm some standardized component model that define the interfaces, metadata, documentation, composition and deployment of the supplier component.

- **Composable:**

  All external interactions of the supplier information service component must take place through the interfaces (required and provided) that will be mentioned in the next point. In addition; the supplier information service component must provide access about itself such as ID, name, price, list of products he sells using methods and attributes.

- **Component Interfaces**



  - **Required interface:**

    The supplier information service component requires from the management component the confirmation to give him his products

  - **Provided interface:**

    The supplier information service component provides the management component the products

- **Reusability:**

  The supplier information service component can be reusable because this component could be used in any program because the suppliers can do his functionalities in any other programs or websites.

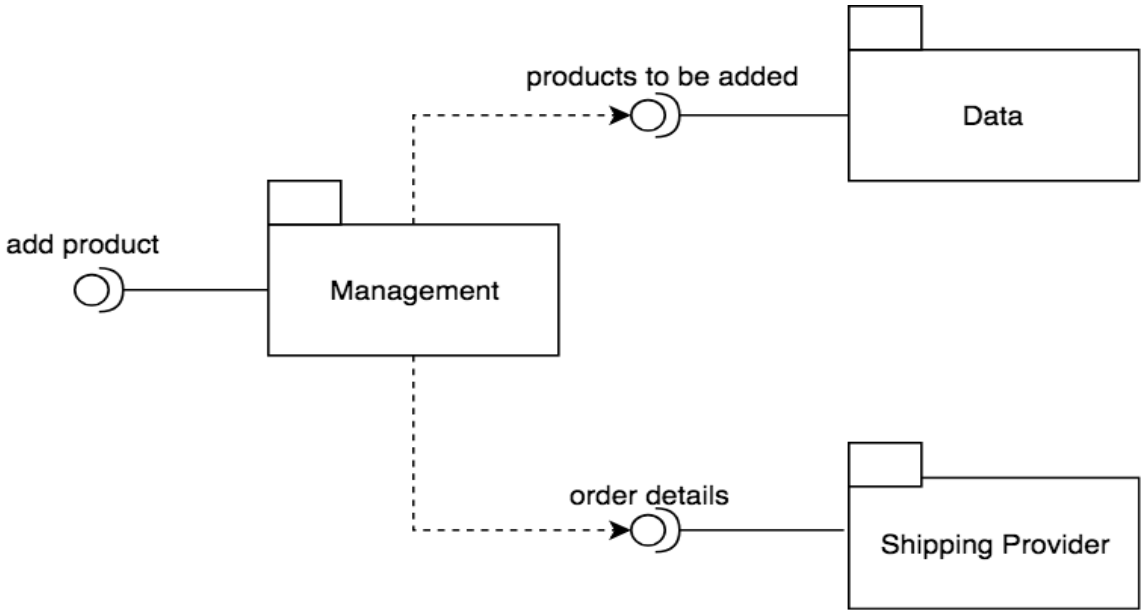
- **Component Adaptation**

  - Wrapper. A wrapper is a special type of a glue-code that encapsulates a component and provides a new interface that extend the original interface, or to add or ensure that the supplier information service provides the admin the wanted products.

- **Component Identification Issues**
    - Trust: is not completely trusted because it is a component bought to be able to be supplied with different products from different companies.
    - Requirements. Different groups of components will satisfy different requirements.
- **Component specification**
    - This component is responsible for selling the product to the management.

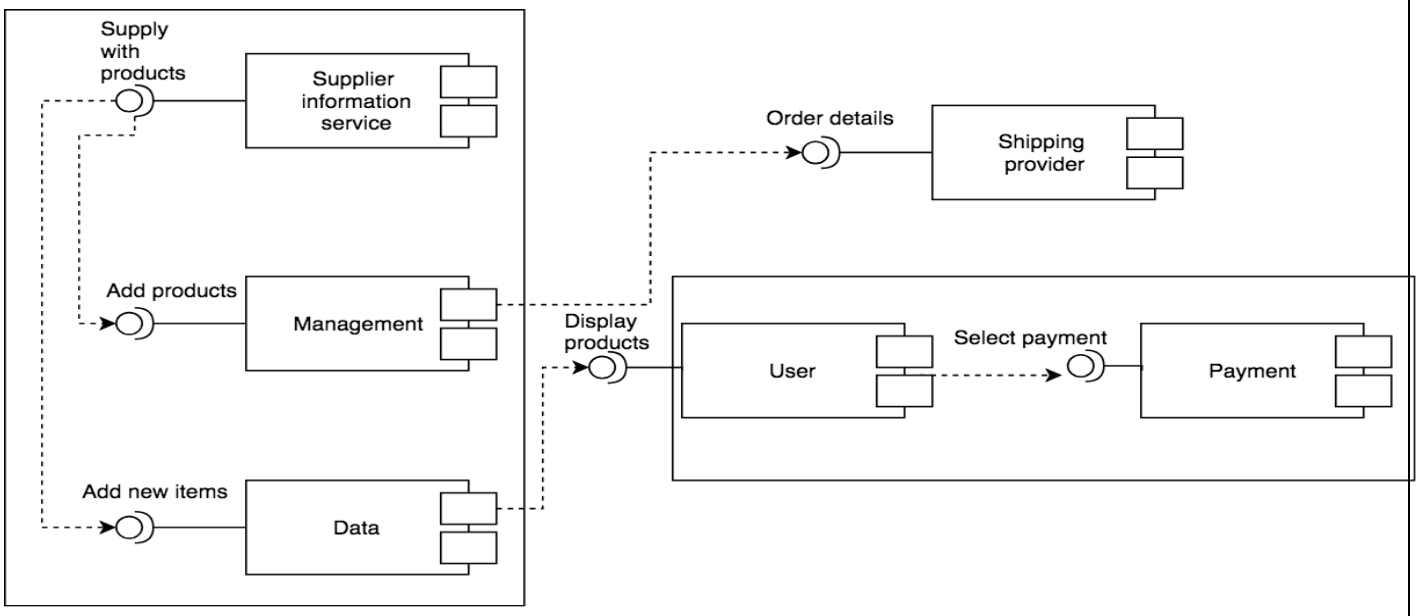
- **Different Architecture view:**
  - **Phase 1:**

    Conceptual Architecture Decomposition of the system is done into 3 main components: management, data, Shipping Provider



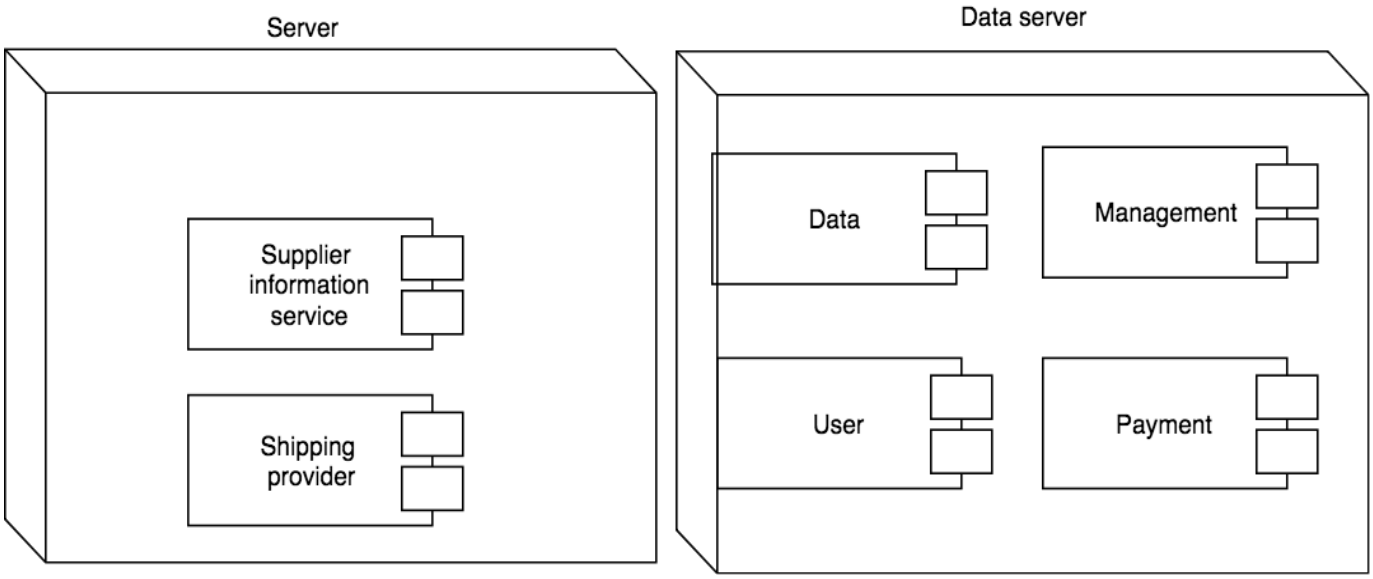
- **Phase 2**

  Component Identification is done and relationship between components.



- **Phase 3:**

  Deployment architecture



- **System Maintenance Phase**

  The approach of CBSE is to provide maintenance by replacing old components by new components or by adding new components into the systems. For example, the shipping provider component can be replacing by another in case that any contractual problem occurs between the admin and the shipping provider.

  Component development process

  A generic assessment process activity

  - **Requirements:** the required functionality is to have components that implements the user functionality as having an account and could make orders and other component for the management to control the products and categories of the website also for the data stored in branches and shipping provider management responsible for delivering orders and component responsible for the payment methods.
  - **Find:** the components that must provide the required functionality are admin, model, order, product, user, supplier information service, payment, shipping provider, data,
  - **Select:** We found three external components with reasonable price to be brought to this project which are: supplier information service (responsible for selling products to the admin to be able to add each item to the website with its price), shipping provider (which handles the delivery of orders to users within an expected time and could collect money from them if they chose to pay cash on delivery), and payment (to control the preferred method for each user and associate it with his orders to be able to deliver it within the expected time)and for the other components we have implemented and design our own components.
  - **Verify**

    Test functional and certain extra-functional properties of a component in isolation

We have made a test for each one of the components implemented in combination with other components integrated in an assembly. For example, we have tested the shipping provider and supplier information service and payment component with the management and the user and the result was that the integration was safe and effective.

- **Integrate**
  - **Payment:**

    Once integrating the payment component is could have the ability to perform all its functionality as providing different methods for the user to pay for his orders.
  - **Supplier service information:**

    Once integrating the supplier service information component is would have the ability to perform all its functionality as providing the management system all products to be added to the different categories with its price.
  - **Shipping provider:**

    Once integrating the shipping provider component is would have the ability to perform all its functionality as providing orders to users and could collect money from them if the user has chosen to pay cash on delivery as his method of payment.
- **Test**
  - **Payment:**

    we have made many test cases to ensure the integration of the payment with others. So, we have made test case to try to make a user choose his method of payment for the alternatives. And also test cases for failure conditions as entering an incorrect method of payment (invalid credit card number) to ensure that the system would be able to not complete the process and display an error for the user.

- **Supplier service information:**

  we have made many test cases to ensure the integration of the supplier service information with others. So, we have made test case to try to buy products from this component and add them to the website by the management. And also test cases for failure conditions as selling invalid products (with no name for example) to ensure that the system would be able to not complete the process

- **Shipping provider:**

  we have made many test cases to ensure the integration of the shipping provider with others. So, we have made test case to deliver order to a user. And also test cases for failure conditions as having an order with no shipping information to ensure that the shipping provider would be able to not complete the process

- **Store**

  There are three components selected to be stored the in a component repository with their metadata.

For the other components which we didn't find:

- **Requirements:**

  it is required to have three other components responsible for the user to be able to make orders and a component to store the data in all branch, and other component for the management to be responsible to add categories, products, and confirm orders and communicate with the suppliers.

- **Design:**

  - A component for the user contains all the transaction concerning him like have an account which save his information (name, address, email, phone number …) and can be able to make orders and adding items into his shopping cart or liking a model and add

comment to it as a review for others. Add also he can choose his payment method and confirm or cancel his order.

- A component for the management responsible for adding new categories and products with its models in the website and could confirm orders and sending information to the shipping provider
- A component for the data which keep track of the available storage and the branch which contain it and the description of each item (size, color …)

- **Implementation:**
  - **User:** to implement this component we have need to create methods and attributes for making or cancel orders, selecting models and method of payment
  - **Management:** to implement this component we have to create methods to be able to add categories, products and models in the website and sending details of the order to the shipping provider.
  - **Data:** to implement this component we need to have a data base to store each item and display it by order of their price or by the best seller
- **Integration:**
  - **User:** Once integrating the user component with others it could have the ability of doing all its functionality. So, it is integrable with other components.
  - **Management:** Once integrating the management component with others it could have the ability of doing all its functionality. So, it is integrable with other components.
  - **Data:** Once integrating the data component with others it could have the ability of doing all its functionality. So, it is integrable with other components.

- **Teat:**
  - **User:** we have made many test cases to ensure the integration of the user with others. So, we have made test case to try to create account and other to try to make order. And also test cases for failure conditions as entering an incorrect method of payment (invalid credit card number) to ensure that the system would be able to not complete the process and display an error for the user.
  - **Management:** we have made many test cases to ensure the integration of the management with others. So, we have made test case to try to create categories and other to add products and models. And also test cases for failure conditions as entering an incorrect category (with no name for example) to ensure that the system would be able to not complete the process
  - **Data:** we have made many test cases to ensure the integration of the data component with others. So, we have made test case to try to store many items into the storage. And also test cases for failure conditions as entering an invalid item (with no name for example or with no price) to ensure that the system would be able to not complete the process
- **Release:**
  - **User:** This is the first release of this component (self-implemented for the first time for this project) v.1.0
  - **Management:** This is the first release of this component (self-implemented for the first time for this project) v.1.0
  - **Data:** This is the first release of this component (self-implemented for the first time for this project) v.1.0
- **Maintenance:**
  - **User:**

This component has to maintain all the data concerning to each user to ensure the protection of orders. No one could login with an incorrect password or username

- **Management:**

  This component has to maintain the ability of the admin to add categories and products and manage all functionality of the website. This management have to be hidden from any normal user except the admin.

- **Data**

  this component has to maintain information about all data stored in all branches and make weekly check to maintain the protection of all products

- **Component composition**

The process of assembling components to create a system.
Composition involves integrating components we have to write *'glue code'* to integrate components. We have to form the system by integrating the admin component which communicate with the supplier to order the products and add it immediately with reasonably prices, also the admin creates the inventory of all branches, so the component of the user is ready to make orders and chose payments and the shipping provider finish the process finally.

- **Types of composition:**

1. **Sequential composition**

   where the composed components are executed in sequence. This involves composing the provides interfaces of each component which are:

   - the user:

     The user component provides his account information and specify his own way of payment.

   - The management:

     The admin component provides the information about items to the data component

- the supplier information service:
  The supplier information service component provides the management component the products
- the payment:
  The payment component provides the shipping provider information about if the user wants to pay cash on delivery or not.
- the shipping provider:
  The shipping provider component provides the user component his order to complete the shipping process.
- the data:
  The data component provides the shipping provider component the order details to be delivered to the user(customer)

2. Hierarchical composition

where one component calls on the services of another. The provides interface of one component is composed with the requires interface of another.

Examples:

- The payment provides the shipping provider if the user will pay cash on delivery or not which is required by the shipping provider to know if he will collect money from customer or not.
- The user provides the payment his method of payment which is required by the payment component to complete checkout.

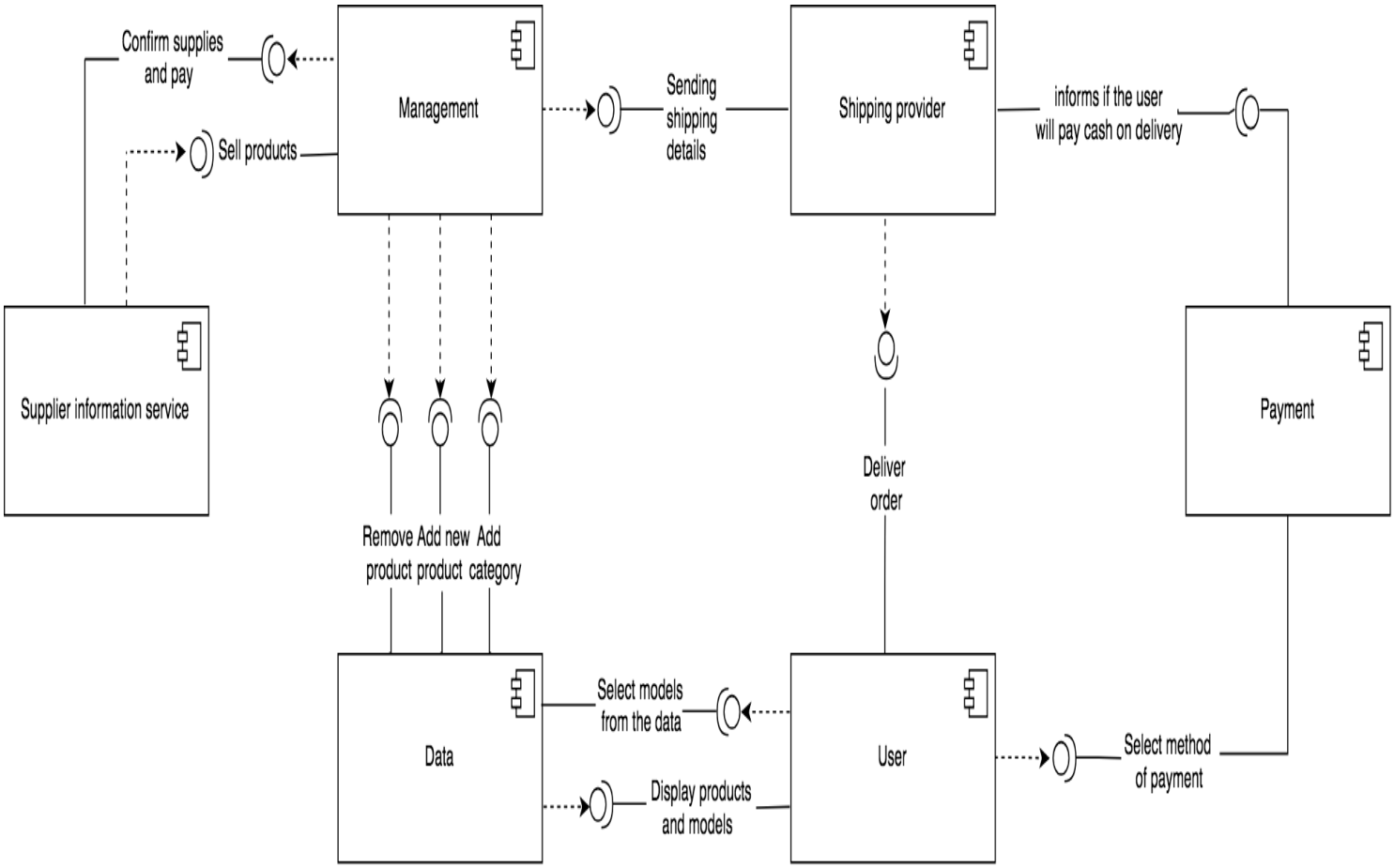
3. Additive composition

Where the interfaces of two components are put together to create a new component.

Examples:

- the shipping provider component is created from the two interfaces of the management and the payment; when the management component provides sending shipping details of the order and the payment method is provided by the payment this component is created.

7. Detailed Component Diagram



Detailed Interfaces of the components:

- **User:**
  - **Required interface:**

    The user component requires from the shipping provider component the expected arrival time and receive his order.
  - **Provided interface:**

    The user component provides his account information and specify his own way of payment.

- **Management:**
  - **Required interface:**

    The management component requires from supplier information service component the items to be able to add it to his website with the prices.
  - **Provided interface:**

    The management component provides the data component the products and all its details.
- **Supplier information service:**
  - **Required interface:**

    The supplier information service component requires from the admin component the confirmation to give him his products
  - **Provided interface:**

    The supplier information service component provides the admin component the products
- **Data:**
  - **Required interface:**

    The data component requires from management component the products to be added to each branch storage.
  - **Provided interface:**

    The data component provides the shipping provider component the order details to be delivered to the user(customer)
- **Payment:**
  - **Required interface:**

    The payment component requires from the user component information about his preferable payment method.
  - **Provided interface:**

The payment component provides the shipping provider information about if the user wants to pay cash on delivery or not.

- **Shopping provider:**
  - **Required interface:**

    The shipping provider component requires from the management component the order information (shipping information)

  - **Provided interface:**

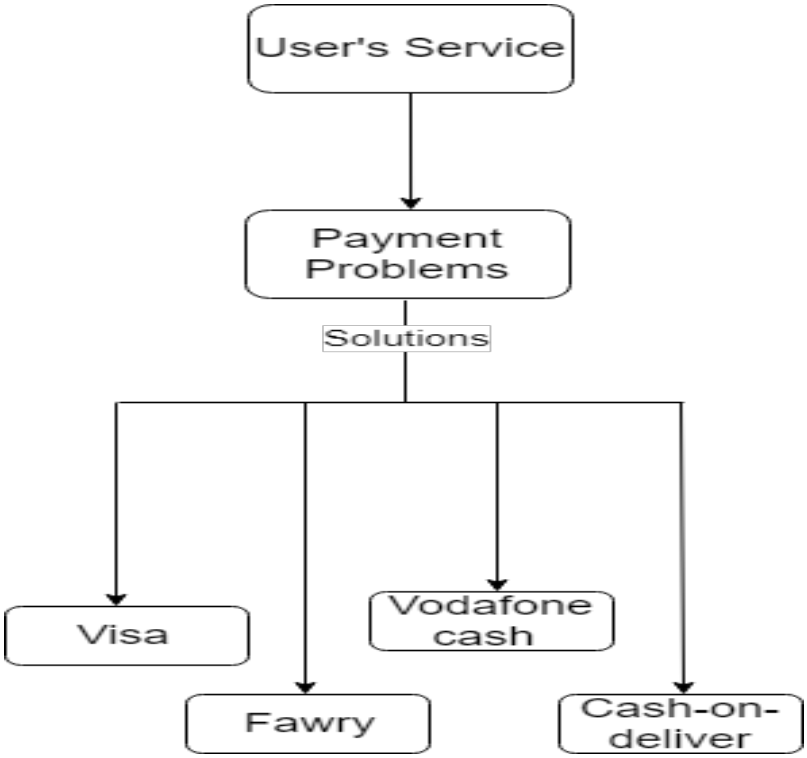
    The shipping provider component provides the user component his order to complete the shipping process.

8. The detailed architecture of the project:

Stakeholders

- Software companies
- Suppliers
- Implementor
- Designer
- Tester, integrator
- Maintainer
- Manager
- Quality assurance people

Decisions:



Problem:

We have a main problem concerning the user's service which is the payment problems and the solution was to implement methods of payment: visa, fawry , cash on delivery.

For example: Our website has to be able to process payments from Users. And our Users will want to have options in terms of how they make a purchase.

Alternatives:

1. PayPal

   PayPal is an e-commerce payment processing company owned by eBay. Users set up a PayPal account and pay a fee for each cash transaction. Rules and fees vary for currencies used and cash amounts transacted. The PayPal website has an easy, step-by-step sign on process. Rules are spelled out in detail. Users can pay from their PayPal account. (For related reading, see 8 Secrets For Selling On The New eBay.)

   We don't choose this solution because the user must have an account on PayPal and also pay a fee for each transaction

2. Amazon Payments

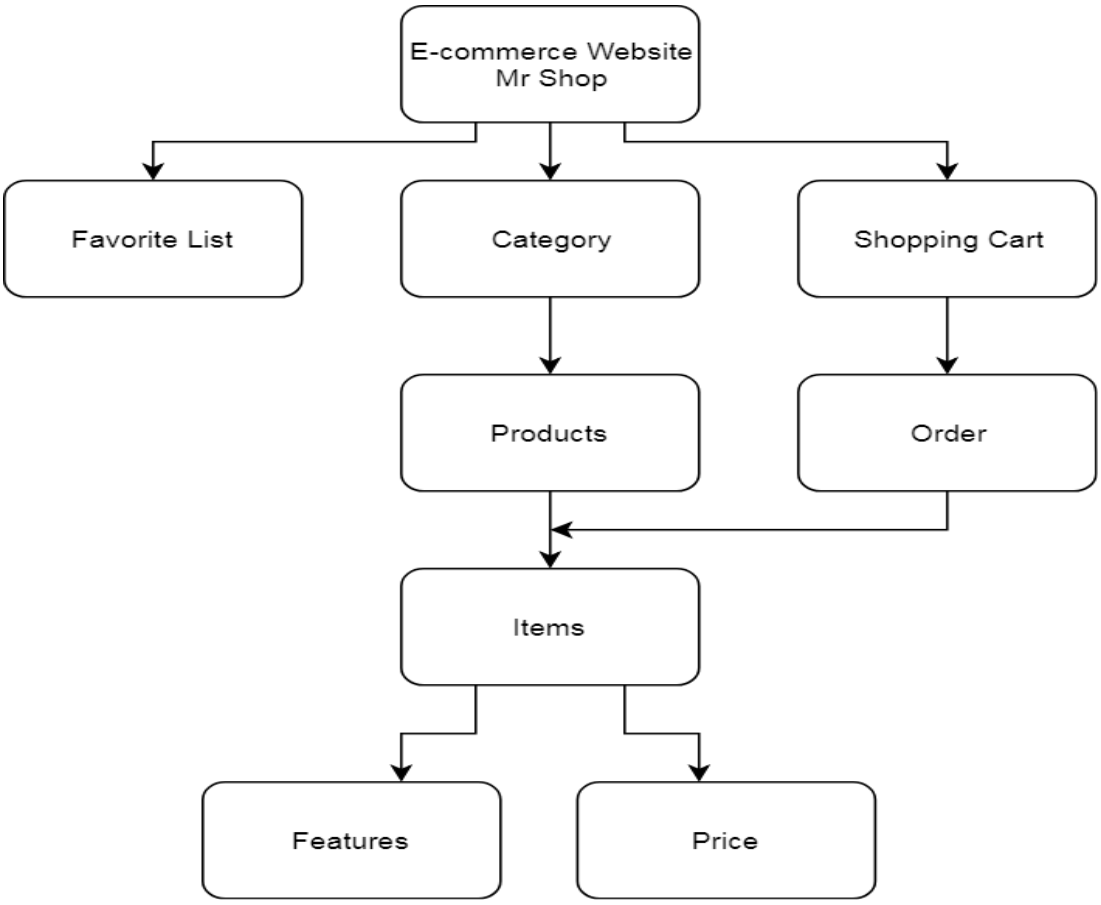
   We don't choose this solution because the user must have an Amazon account

Solution:

So, we implemented many methods such as visa, Fawry, cash on delivery, Vodafone cash; And the User can select his suitable way of payment.

Modular View:

1. Decomposition:



- **Category:** is a sub module of Mr. Shop Website.
  - **Products:** is a sub module of the Category.
    - **Items:** is a sub module of the products.
      - **Price:** is a sub module of Items.
      - **Features:** is a sub module of items.
- **Shopping Cart:** is a sub module of Mr. Shop Website
  - **Order:** is a sub module of shopping cart.
- **Favorite List:** is a sub module of Mr. Shop Website.

The e-commerce website (Mr. Shop) is decomposed into main parts; Wish List, Category, Shopping Cart. The Category part is composed of products and each product is composed of items each one is related to different features and price and other description. Also, the shopping cart part is related to a certain order with different product items and total price for checkout. The wish list part is optional for users who want to have a set of products they like but don't have to buy them.

2. Uses

Admin uses the supplier to choose new products to be added into Mr. Shop website and give him confirmation and pay.

Admin uses the Branch to keep track of the available items in the different categories and can add or delete or update a category with new items.

The product uses the branch store to be displayed if the quantity is sufficient in the stores.

The user uses the products to select one of them to be added into his shopping cart and be ready to checkout.

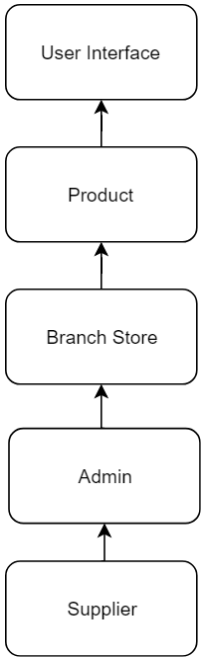
3. **Layers:**

Supplier can use the admin to sell him different products to be added on the website.

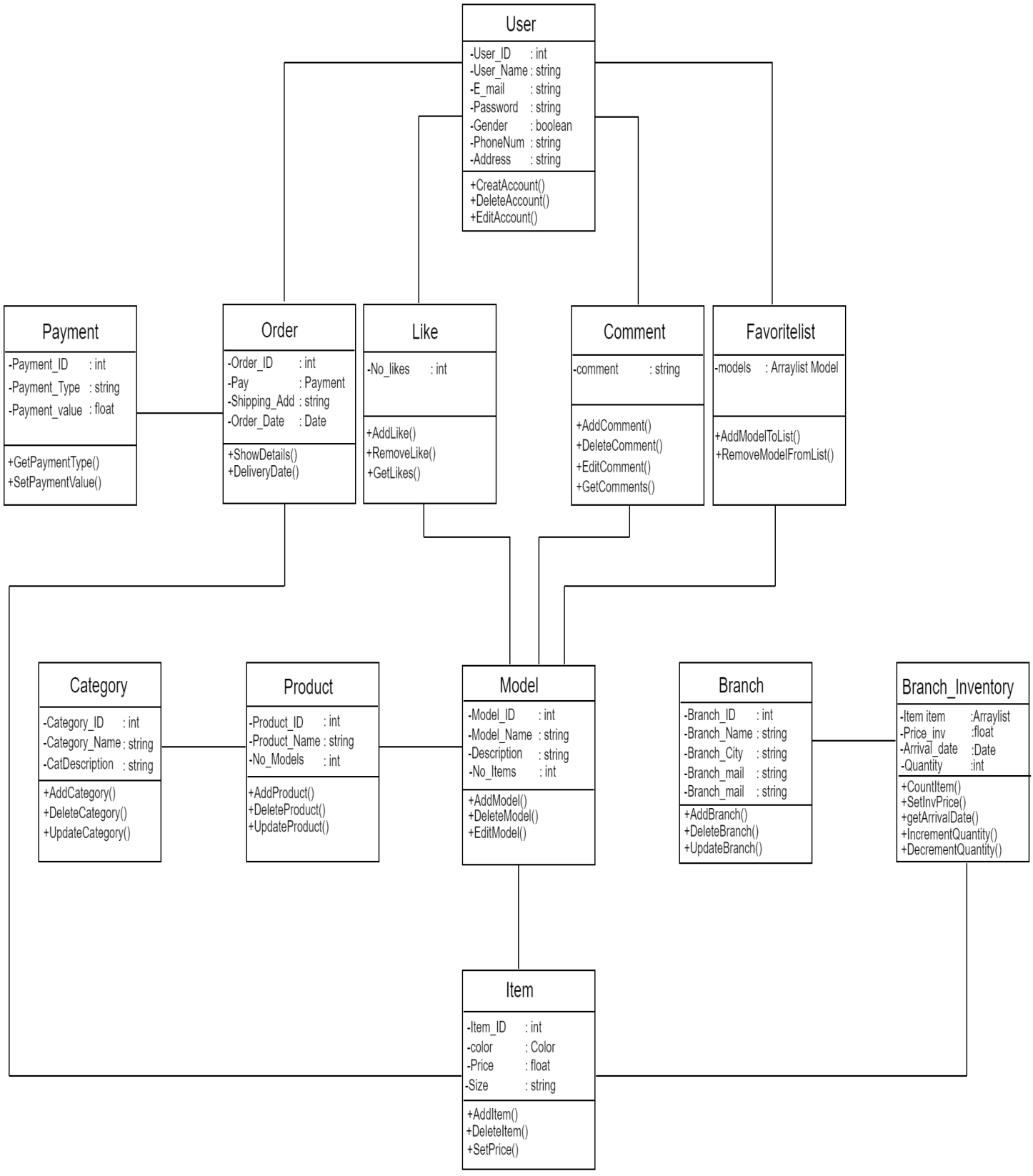
Admin can use the branch store to add the new items in each branch with reasonable price and can also check the quantity of each item.

Branch store use the product to be displayed.

Product can used by user interface to be added in the shopping cart or in the favorite list.

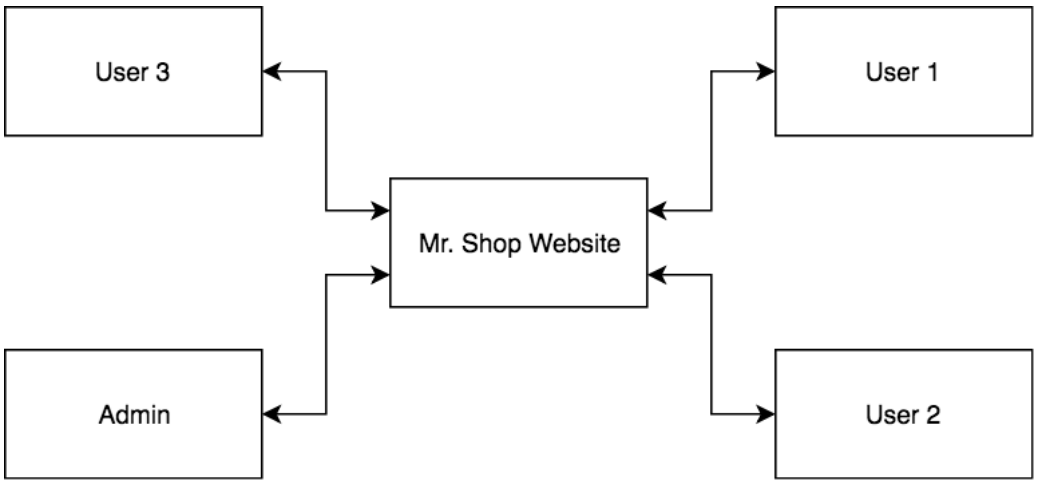


4. Class Diagram



Architecture Style:

- **Repository style:**



Problem:

Our website is richly structured information containing data about all categories with their products and items and details about their price, description, quantity in addition to the account information of users and their orders. So, to the data is long-lived and have to be manipulated in many different ways.

Context:

Shared data (categories, products, models, likes, comments) to be acted upon by multiple users.

Solution:

- **system model:**

  Centralized body of information for the website which is independent computational elements.

- **components:**

  One memory for saving all information, many computational.

- **connectors:**

  Direct access or procedure call. The user is able to access any category and its products and items easily and could add a comment, like or add to his shopping cart or his favorite list.

- **control structure:**

Varies, may depend on input or state of computation. For example, the control structure used for adding an item to the shopping cart is different from removing an item from the shopping cart, adding item to favorite list.

variants:

traditional data base systems, blackboard systems

9. Viewpoints:

1- Logical Viewpoint:

Supports the functional requirements (the services the system should provide to its end users)

The system can display the categories and its items, can display the price of each item, also it can display the page that contains all the favorite products of each user and can calculate the total price of the items' ordered. The system provides payment methods and send mails to the users to confirm their orders.

2- Process Viewpoint:

Each user must run in different thread in order to work concurrently and optimize the whole system performance.
Each branch should run in its own thread in order to ensure the system availability.
The Admin is allowed to access the database of the system, the system must auto backup user & product data and configure before any system updates to make sure that in case of system failure the data can be recovered (fault tolerance).

3- Implementation Viewpoint:

- User module
- Admin module
- Branch module
- Payment module
- Shipping provider module
- Supplier module

4- Deployment Viewpoint:

Supplier module provides the Admin module with the products, so the Admin module add these products to the Branch module.
The User module inputs its data to be able to make an order, also he chooses the

payment method from the payment module and inputs the order details so that the shipping provider module could deliver the user's order.

5- Scenario Viewpoint:

The scenario viewpoint consists of use cases for example:

- The user can make an order.

  By adding the item to his shopping cart and choose the method of payment.

- The admin can update the branch store.

  By checking the quantity of items in each branch and demand from the Supplier the required items.

10. Functional Points of The Project:

- **Based on a combination of program characteristics which are:**
  - The website should authenticate the admin to be able to add categories and products with its models and also the user
  - Create account for each user to be able to make orders
  - Collect data from users
  - Save accounts in a data base
  - Display the categories and the products with its models.
  - Display the cost of each model
  - Display the number of likes for each model
  - Display the comment concerning each model
  - Sort the products according to price, best seller
  - Create shopping cart for each user containing the items he would buy
  - Save the inventory of each branch in a data base with the quantity of items and their specifications
  - Calculate the total price of the items ordered
  - Display the favorite list of the user logged in
  - Send mails to users containing promocodes and offers
- **Function Points – Calculation**
- **Function Points – Taking Complexity into Account -14 Factors Fi**

  Each factor is rated on a scale of: Zero: not important or not applicable & Five: absolutely essential

Complexity factor: Fi	Value= 0	Value= 1	Value= 2	Value= 3	Value= 4	Value= 5	Fi
Backup and recovery	0	0	0	0	0	1	5
Data communication	0	0	0	0	1	0	4
Distributed processing functions	0	0	0	1	0	0	3

Is performance critical?	0	0	0	0	0	1	5
Existing operating environment	1	0	0	0	0	0	0
Online data entry	0	0	0	0	0	1	5
Input transaction	0	0	0	1	0	0	3
Master files updated on-line	0	0	0	0	0	1	5
Complexity of inputs, outputs, files, inquiries	0	0	0	0	1	0	4
Complexity of processing	0	0	1	0	0	0	2
Code design for re-use	0	0	0	0	0	1	5
Are conversion/installation included in design?	1	0	0	0	0	0	0
Multiple installations	1	0	0	0	0	0	0
Application designed to facilitate change by the user	0	0	0	1	0	0	3
							Sigma = 44
Complexity adjustment factor	0.65 + 0.01 * Sigma(F) = 1.09						

Measurement parameter		Weighting Factor			
	count	Simple	average	complex	
Number of user inputs	5	3	4	6	20
Number of user outputs	5	4	5	7	25
Number of user inquiries	3	3	4	6	12
Number of files	14	7	10	15	140
Number of external interfaces	3	5	7	10	21
Count-total (UFC)	→				218
Complexity adjustment factor					1.09
Function points	→				240

Assuming F = 45

$$FP = UFC * [0.65 + 0.01 * \sum_{I=0}^{I=14} F_i]$$

FP = 218*[0.65 + 0.01 * 45]

= 218*1.09 = 240

Complexity adjustment factor = 1.09

Estimated FP = 240

Assuming:

Team average productivity (similar project type) =7 FP/p-m (person-month)

Burdened labor rate = 8000 $/p-m

Then:

Estimated effort = FP/ Organization average productivity =33/5.5 = 6 p-m

Cost per FP = Burdened labor rate / Organization average productivity = 8000/7

= 1142.85 $/FP

Project cost = 8000 * 6

= 48000 $

11. Detailed analysis of the project cost:

Cost estimation is needed early for s/w pricing

S/W price = cost + profit

1. Software Cost:

- **Software Cost Components**

1. Effort costs:

This project is developed by a group of 6 students so there is no salary for each one.

2. Tools costs:

Hardware and software used for the development of this project is Microsoft office free version for the students and Microsoft Visio and Xamp.

3. Travel and training costs:

No need for it.

4. Overheads:

a. costs of co-working space, heating, lighting
b. costs of networking and communications (Internet, telephone …)
c. costs of shared facilities (staff restaurant, etc.)

- **Software pricing policy:**

Factor	Description
Market opportunity	The project is for developing an e-commerce website which wishes to move into the huge markets which include similar websites as amazon, souq, Jumia … So, we have to decrease the profit to gain a reputation as a startup e-commerce website.
Contractual terms	The submission requires submitting the source code of the project so high prices may be charged.

- **Programmer productivity measures:**

  S/W productivity measures are based on:

  1. Size related measures:
     - Based on some output from the software process
     - Number lines of delivered source code (LOC)

Functions	Estimated LOC	LOC/pm	$/LOC	Cost	Effort
AddCategory	900	315	20	18.000	2.8
AddProducts	500	200	15	7500	2.5
AddItem	550	150	14	7700	3.6
CreateAccount	700	300	20	14.000	2.3
AddItemToShoppingCart	500	320	17	8500	1.56
AddModelToList	690	210	22	15.180	3.2
AddBranch	900	220	19	17.000	4.09
UpdateCategory	800	140	18	14.000	5.7
CountItem	990	300	21	20.000	3.3
Totals	6530			1033880	29.05

Assuming

Estimated project LOC = 6530

Organisational productivity (similar project type) = 545 LOC/p-m

Burdened labour rate = 8000 $/p-m

Then

Effort = 6530/545 = (11.98) = 12 p-m

Cost per LOC = 8000/545 = (14.6) = 15 $/LOC

Project total Cost = 8000 * 12 = 96000$

2. Function-related measures
   - based on an estimate of the functionality of the delivered software:
     - Function-points (detailed in the previous section)
     - Object-point

       The website will have 6 screens and will produce 1 report:

       A Homepage screen: shows the name of the website and login or create account options.

       A product screen: shows the products in each category

       A model screen: shows the model description and details

       An item details screen: shows the item details (quantity, color, size …)

       An add to cart screen: shows the content of the shipping cart to proceed checkout

       An Order details screen: shows the order details; items ordered and their price and total price for the order.

       – A documentation report: shows a brief description about the website and all the contents.

Name	objects	complexity	weight
product	screen	simple	1
model	screen	simple	1
Item details	screen	simple	1
Add to cart	screen	medium	2
Order details	screen	medium	2
documentation	report	medium	5
		Total	12

Developer's experience and capability / ICASE maturity and capability	Very low	low	nominal	high	Very high
PROD: Productivity object point per person month	4	7	13	25	50

The developers' experience is very low (4)

The CASE tool is low (7).

So, we have a productivity rate of 5.5.

Effort in p-m = NOP / PROD

NOP = number of OP of the system

Then, Effort = 12/5.5 = 2.8 p-m

If we consider that 20% of the project can be reused:

Adjusted NOP = NOP * (1 - % reuse / 100)

Adjusted NOP = 12 * (1 – 20/100) = 15 OP

Adjusted effort = 15/5.5 = (2.7) = 3 p-m

- **Expected software size:**

  Compute Expected Software Size (S) as weighted average of:

  - Optimistic estimate: S(opt) = 5000 LOC
  - Most likely estimate: S(ml) = 4600 LOC
  - Pessimistic estimate: S(pess) = 7000 LOC

  S = {S(opt) **+** 4 S(ml) **+** S(pess) } / 6

  = {5000 +4*4600 + 7000} / 6

  = 5066 LOC

- **Estimation techniques**
  - **Expert judgement**

    One or more experts in both software development and the application domain use their experience to predict software costs. These experts are the doctor teaching this course and the TA. Process iterates until some consensus is reached; most of the implementation of this project was revised by them and many iterations occurs until it is finally finished so they estimate a total cost for it based on their direct experience of similar systems

  - **Estimation by analogy**

    The cost of a project is computed by comparing the project to a similar project in the same application domain as amazon, souq.com, jumia and others e-commerce websites. It may be Accurate because that those project data were available. But it may be not accurate because they are not similar in the market so not 100% comparable.