Online Shopping

UCS503 Software Engineering Project Report Mid-Semester Evaluation

Submitted By:

<102016060> Govind Singla

<102016044> Parth Vohra

<102196005> Aarushi Gupta

<102196004> Bipasha Gupta

**BE Second Year, CoSE**

**Group No:2CS10**

**Submitted to:**

Dr. Manisha Kaushal



**Computer Science and Engineering Department**

**TIET, Patiala May 2022**

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Assignment** | **Page No.** |
| **1.** | **Project Selection Phase** |  |
| 1.1 | Software Bid |  |
| 1.2 | Project Overview |  |
| **2.** | **Analysis Phase** |  |
| 2.1 | Use Cases |  |
| 2.1.1 | Use-Case Diagrams |  |
| 2.1.2 | Use Case Templates |  |
| 2.2 | Activity Diagram and Swimlane Diagrams |  |
| 2.3 | Data Flow Diagrams (DFDs) |  |
| 2.3.1 | DFD Level 0 |  |
| 2.3.2 | DFD Level 1 |  |
| 2.3.2 | DFD Level 2 |  |
| 2.4 | Software Requirement Specification in IEEE Format |  |
| **3.** | **Design Phase** |  |
| 3.1 | Class Diagram |  |
| 3.2 | Sequence Diagram |  |
| 3.3 | Collaboration Diagram |  |

* 1. Database Design : ER Diagram
  2. State Chart Diagrams

1. **Implementation**
   1. Component Diagrams
   2. Deployment Diagrams
   3. Screenshots
2. **Testing**
   1. Test Plan
   2. Test Cases
   3. Test Reports

|  |  |
| --- | --- |
| **1.** | **Project Selection Phase** |
| 1.1 | Software Bid |

Group:2CS10

**UCS 503- Software Engineering Lab**

Dated: 28-01-2021

**Team Name:** GAPB

**Team ID (will be assigned by Instructor):**

Please enter the names of your Preferred Team Members.

* You are required to form **a three to four person** teams
* Choose your team members wisely. You will not be allowed to change teams.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Roll No | Project Experience | Programming  Language used |
| Govind Singla | 102016060 | Coding Part/Documentation | Python |
| Aarushi Gupta | 102196005 | Coding Part/Documentation | SQL |
| Parth Vohra | 102016044 | Coding Part/Documentation | PLSQL |
| Bipasha Gupta | 102196004 | Coding Part/Documentation | PLSQL |

# Programming Language / Environment Experience

List the languages you are most comfortable developing in, **as a team**, in your order of preference. Many of the projects involve Java or C/C++ programming.

1. Django

2. Python

3. SQL

**Choices of Projects:**

Please select **4 projects** your team would like to work on, by order of preference: *[Write at-least one paragraph for each choice (motivation, reason for choice, feasibility analysis, etc.)]*

|  |  |
| --- | --- |
| First Choice | Online Shopping (E-commerce Website) |
| Second Choice | College Network () |
| Third Choice | Extension Hub with recommendation system |
| Fourth Choice | Hand gesture gaming using ML |

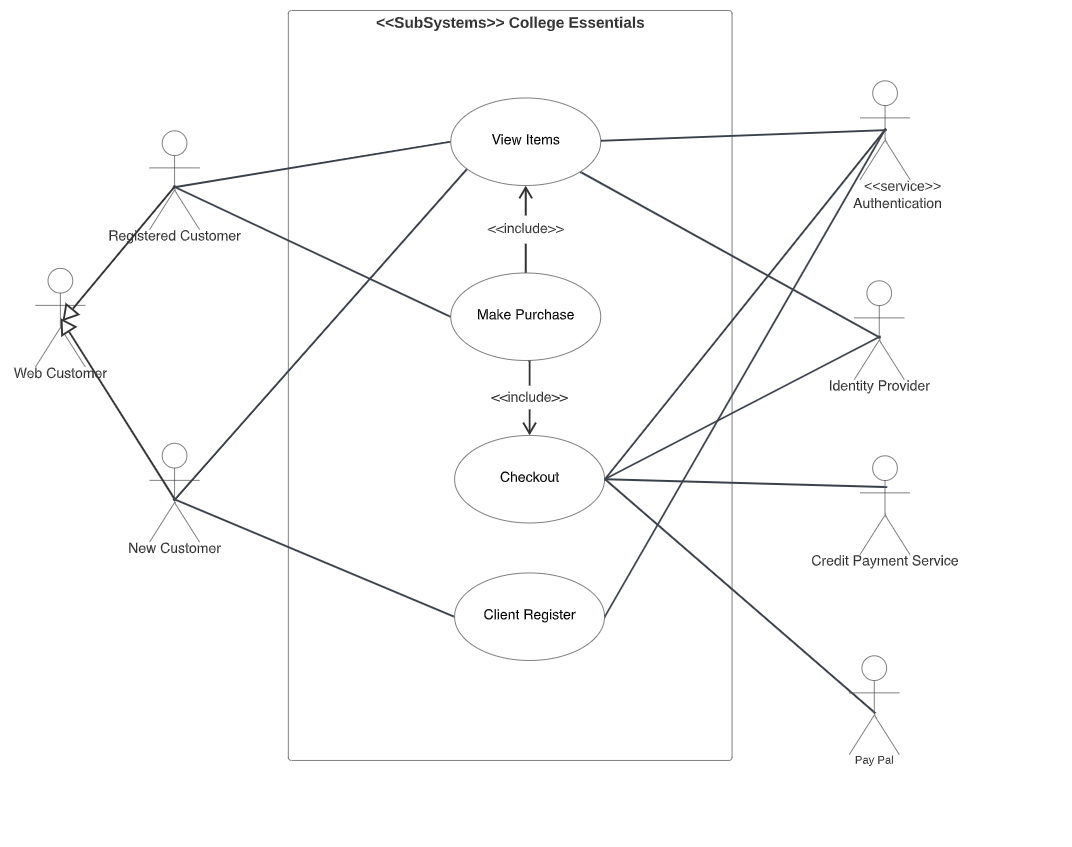
|  |  |
| --- | --- |
| 1.2 | Project Overview |

Our website sells vegetables, fruits, grocery products, stationary, and other essentials online. The website supports a number of features for customers living in the  residential areas and for shops available in that area. It makes users aware of their requirements, choices, and new products in the near future. Customers will be able to save time instead of standing in long queues. The product will be explained in an elaborated form and the user will have different choices like saving the product to buy later and adding it to the cart.

|  |  |
| --- | --- |
| **2.** | **Analysis Phase** |
| 2.1 | Use Cases |
| 2.1.1 | Use-Case Diagrams |

**Web Customer** **actors** use some websites to make purchases online. Top level use cases are **View Items**, **Make Purchase** and **Client Register**. View Items use case could be used by the customer as a top level use case if the customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customers to register on the web site, for example to get some coupons or be invited to private sales. Note, that **Checkout** use case includes a **use case** not available by itself - checkout is part of making a purchase.

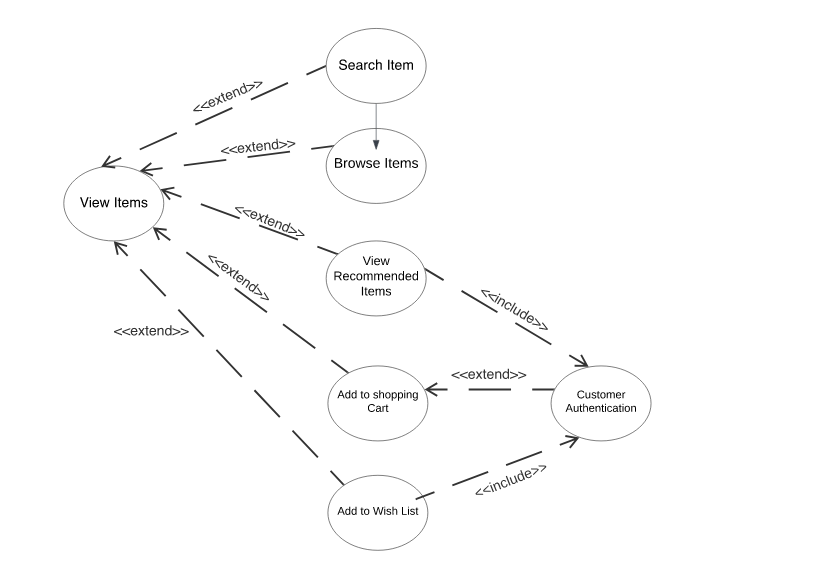
Except for the **Web Customer** actor there are several other actors which will be described below with detailed use cases.



*Online shopping UML use case diagram example - top level use cases.*

**View Items** use case is **extended** by several optional use cases - customers may search for items, browse catalog, view items recommended for him/her, add items to shopping cart or wish list. All these use cases are extending use cases because they provide some optional functions allowing customers to find items.

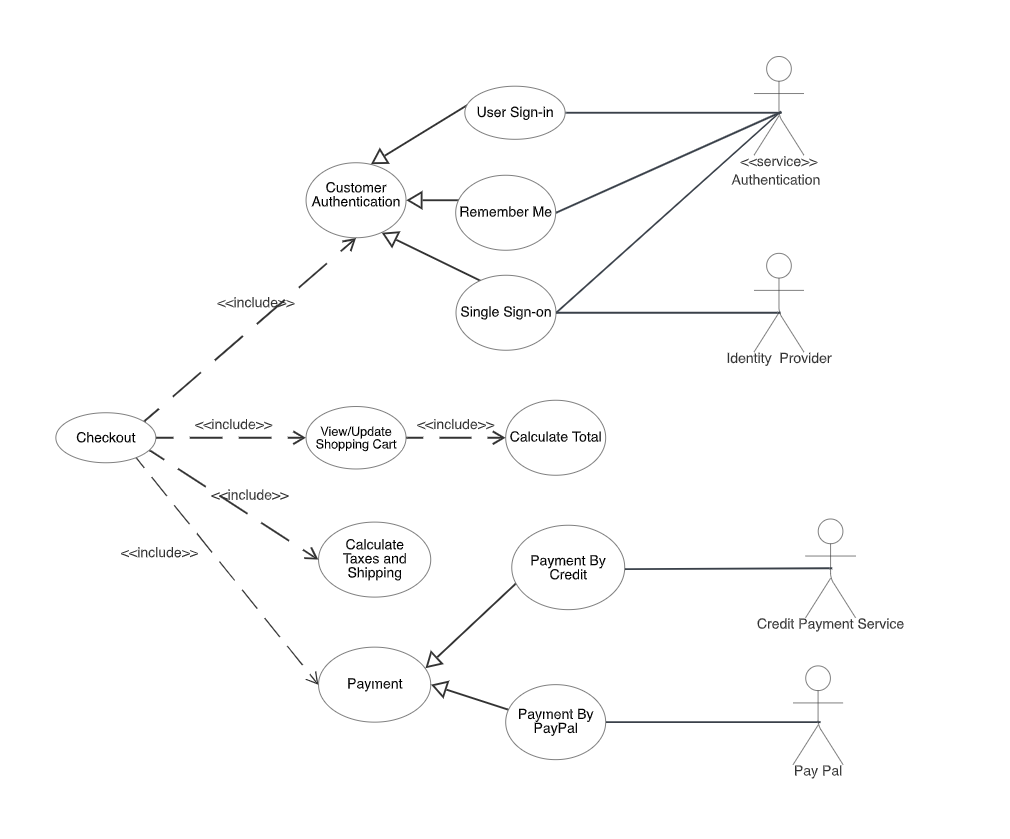
**Customer Authentication** use case is **included** in **View Recommended Items** and **Add to Wish List** because both require the customer to be authenticated. At the same time, items could be added to the shopping cart without user authentication.



*Online shopping UML Use Case diagram example - view items use case.*

**Checkout** use cases include several required use cases. Web customers should be authenticated. It could be done through a user login page, user authentication cookie ("Remember me") or Single Sign-On (SSO). Web site authentication service is used in all these use cases, while SSO also requires participation of external identity providers.

**Checkout** use case also includes **Payment** use case which could be done either by using credit card and external credit payment service or with PayPal.



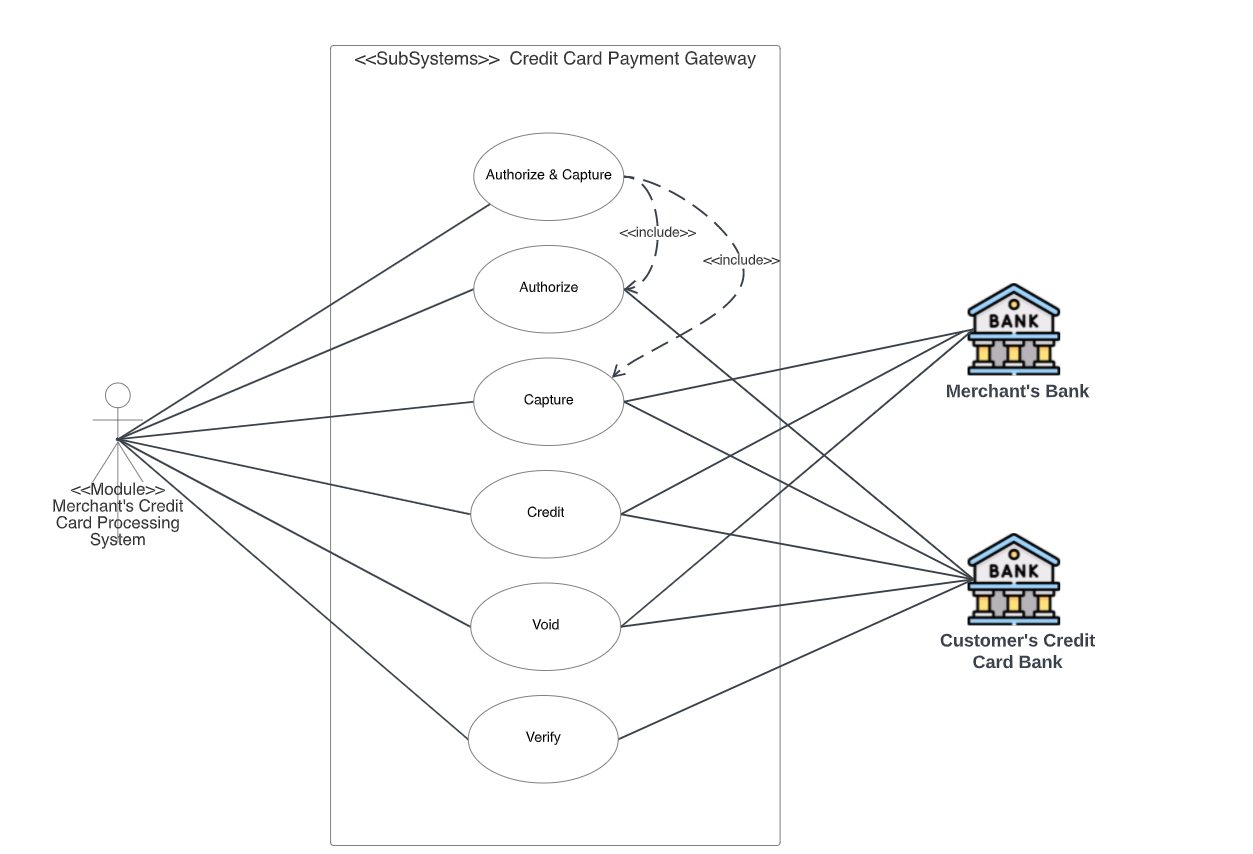
*Online shopping UML use case diagram example - checkout, authentication and payment use cases.*

**Credit Card Processing System** (aka Credit Card Payment Gateway) is a **subject**, i.e. system under design or consideration. Primary **actor** for the system is a **Merchant’s Credit Card Processing System**. The merchant submits some credit card transaction request to the credit card payment gateway on behalf of a customer. Bank which issued the customer's credit card is an actor which could approve or reject the transaction. If a transaction is approved, funds will be transferred to the merchant's bank account.

**Authorize and Capture** use case is the most common type of credit card transaction. The requested amount of money should be first authorized by **Customer's Credit Card Bank**, and if approved, is further submitted for settlement. During the settlement funds approved for the credit card transaction are deposited into the **Merchant's Bank** account.

In some cases, only **authorization** is requested and the transaction will not be sent for settlement. In this case, usually if no further action is taken within some number of days, the authorization expires. Merchants can submit this request if they want to verify the availability of funds on the customer’s credit card, if the item is not currently in stock, or if the merchant wants to review orders before shipping.

**Capture** (request to capture funds that were previously authorized) use case describes several scenarios when a merchant needs to complete some previously authorized transaction - either submitted through the payment gateway or requested without using the system, e.g. using voice authorization.



*UML use case diagram example for a credit cards processing system.*

**Credit** use case describes situations when a customer should receive a refund for a transaction that was either successfully processed and settled through the system or for some transaction that was not originally submitted through the payment gateway.

**Void** use case describes cases when it is needed to cancel one or several related transactions that were not yet settled. If possible, the transactions will not be sent for settlement. If the Void transaction fails, the original transaction is likely already settled.

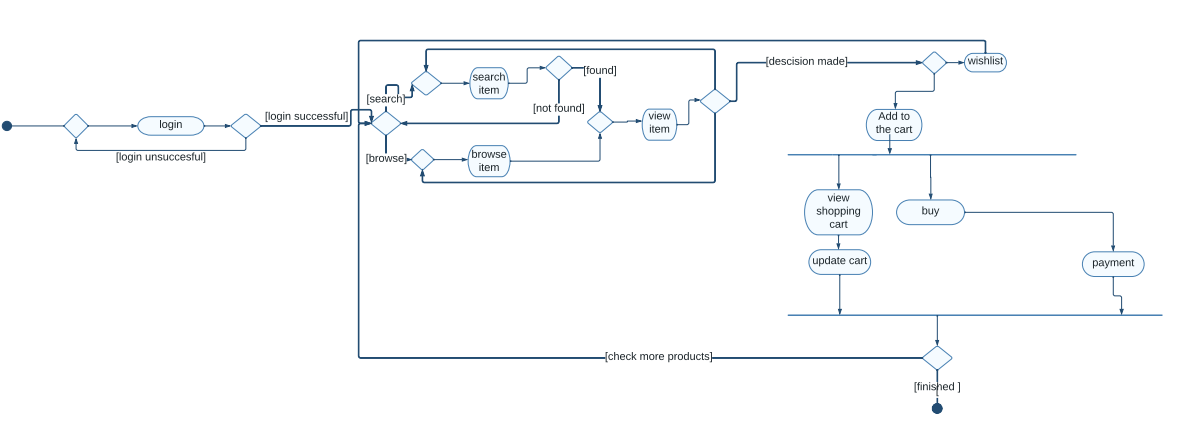
**Verify** use case describes zero or small amount verification transactions which could also include verification of some client's data such as address.

The Web Publishing System has two active actors and one cooperating system.

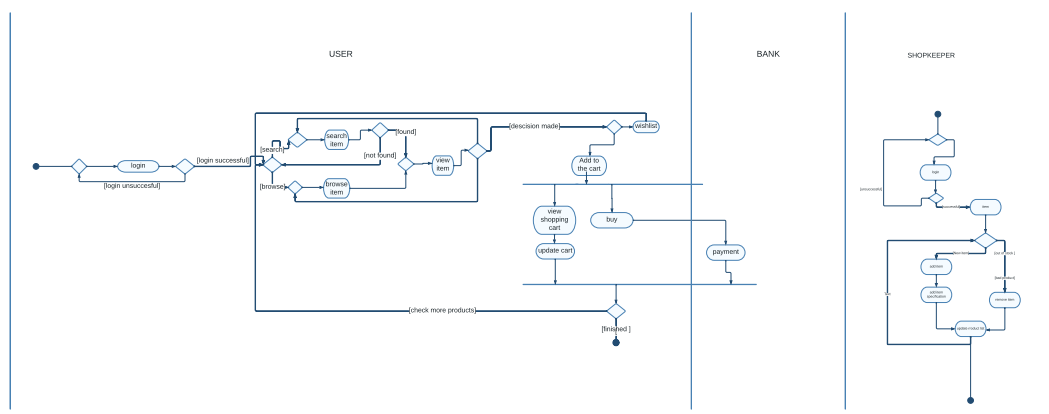
The Customer, Shopkeeper, or Reviewer accesses the Online products through the Internet. Any Customer or Reviewer communicates with the system through the internet. The Editor accesses the entire system directly. There is a link to the (existing) Database .

<< The division of the Web Publishing System into two component parts, the Online Website and the offline trade is an example of using domain classes to make an explanation clearer. >>

|  |  |
| --- | --- |
| 2.2 | Activity Diagram and Swimlane Diagrams |



*Fig. Activity Diagram*

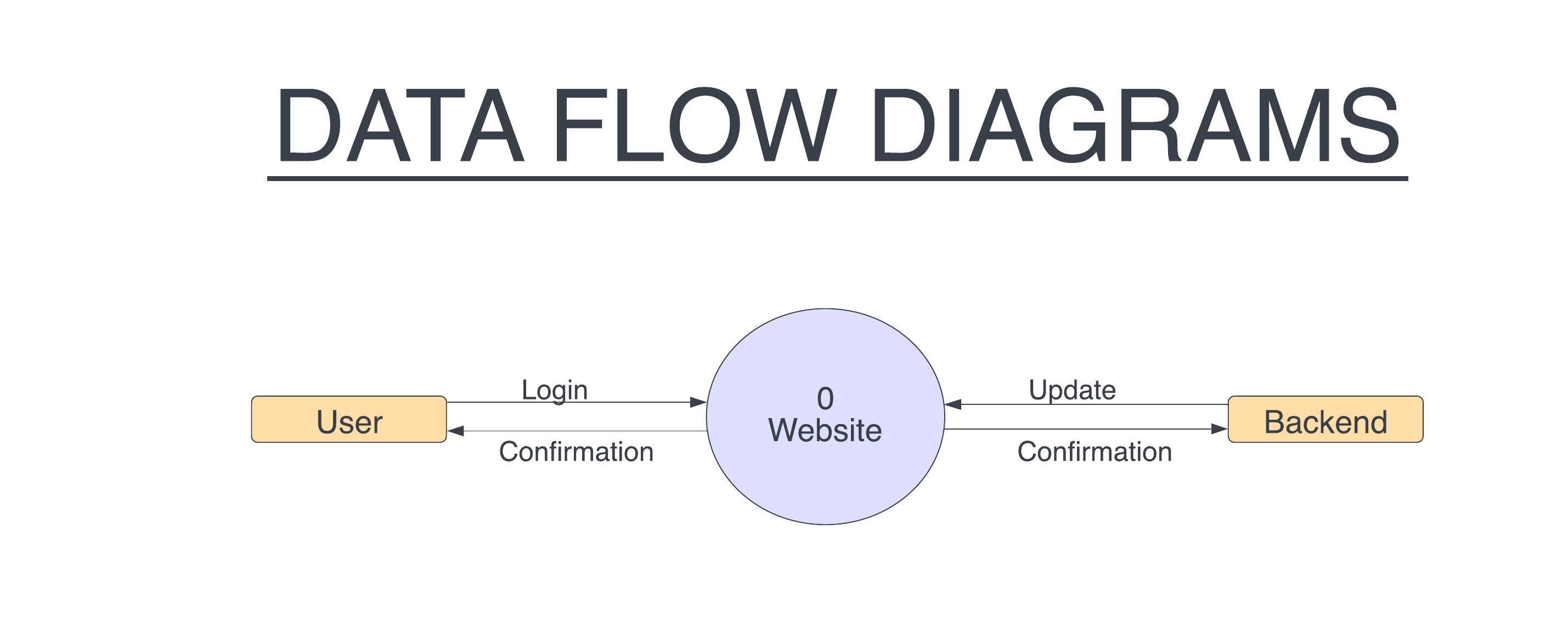


*Fig. Swimlane Diagram*

|  |  |
| --- | --- |
| 2.3 | Data Flow Diagrams (DFDs) |
| 2.3.1 | DFD Level 0 |

The **Level 0 DFD Diagram for E-commerce Website** contains the basic yet general process of the system. Its purpose is to give the system analyst and programmers the basis for further process. The reason why the DFD Level were done one-by-one is to see and avoid flaws while still designing the diagram.

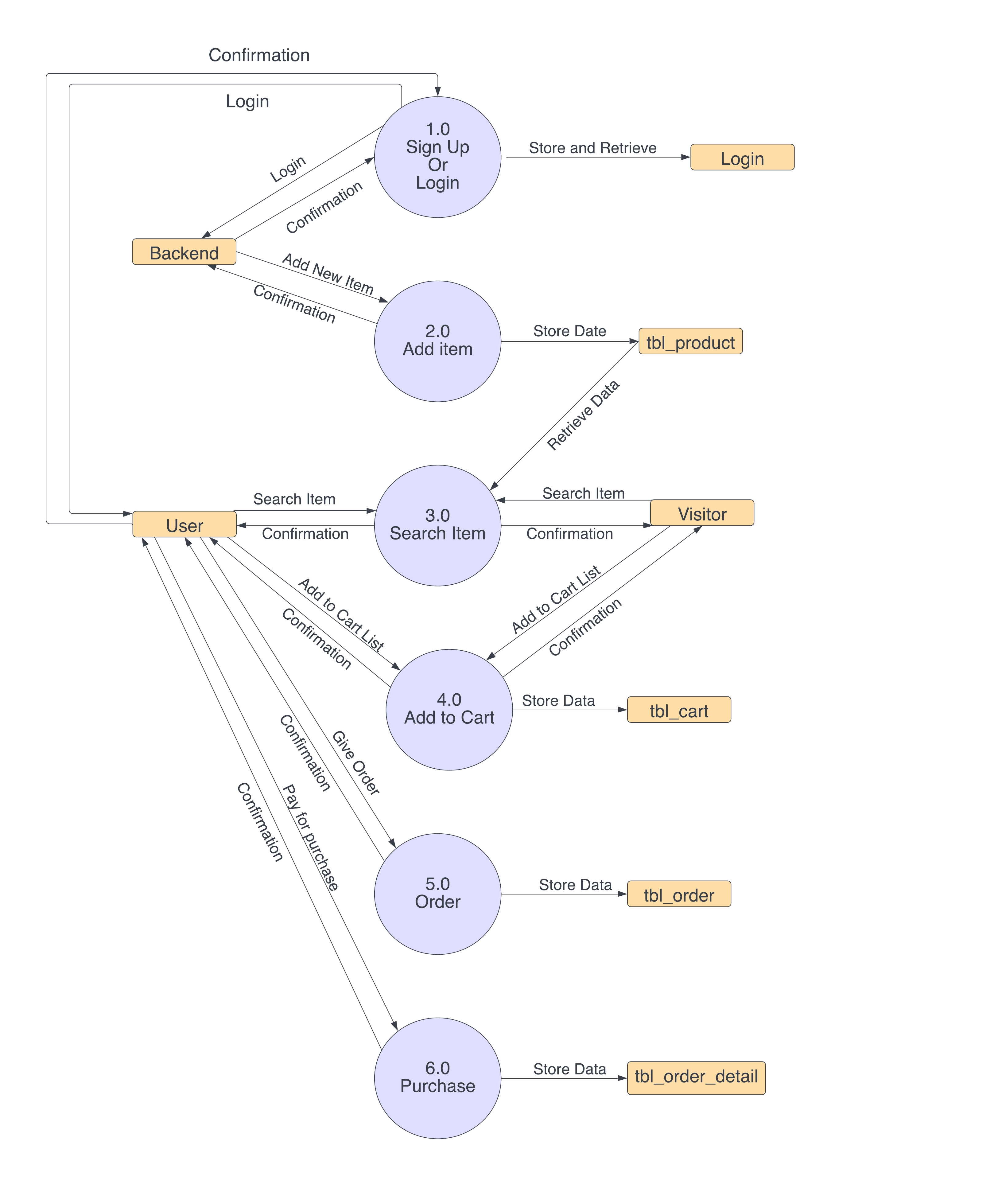
E-Commerce Website DFD Level 0 presents the main idea to be the basis for the proceeding levels. The basic idea is represented by a single process consisting the **main process**, **users**and **data**.



|  |  |
| --- | --- |
| 2.3.2 | DFD Level 1 |

The **DFD Level 1 Diagram for E-commerce Website**provides a broad overview and greater depth of DFD Level 0. The single process node from the context diagram is broken down into sub processes to see the included data that may enter and exits system.

**E-commerce Website** **DFD** **Level 1**lists all of the included processes that make up the entire system. It is the broadened context terms that consist of several processes derived from the main process. They were also numbers to see that were all part of the single process from E-Commerce Website DFD Level 0.



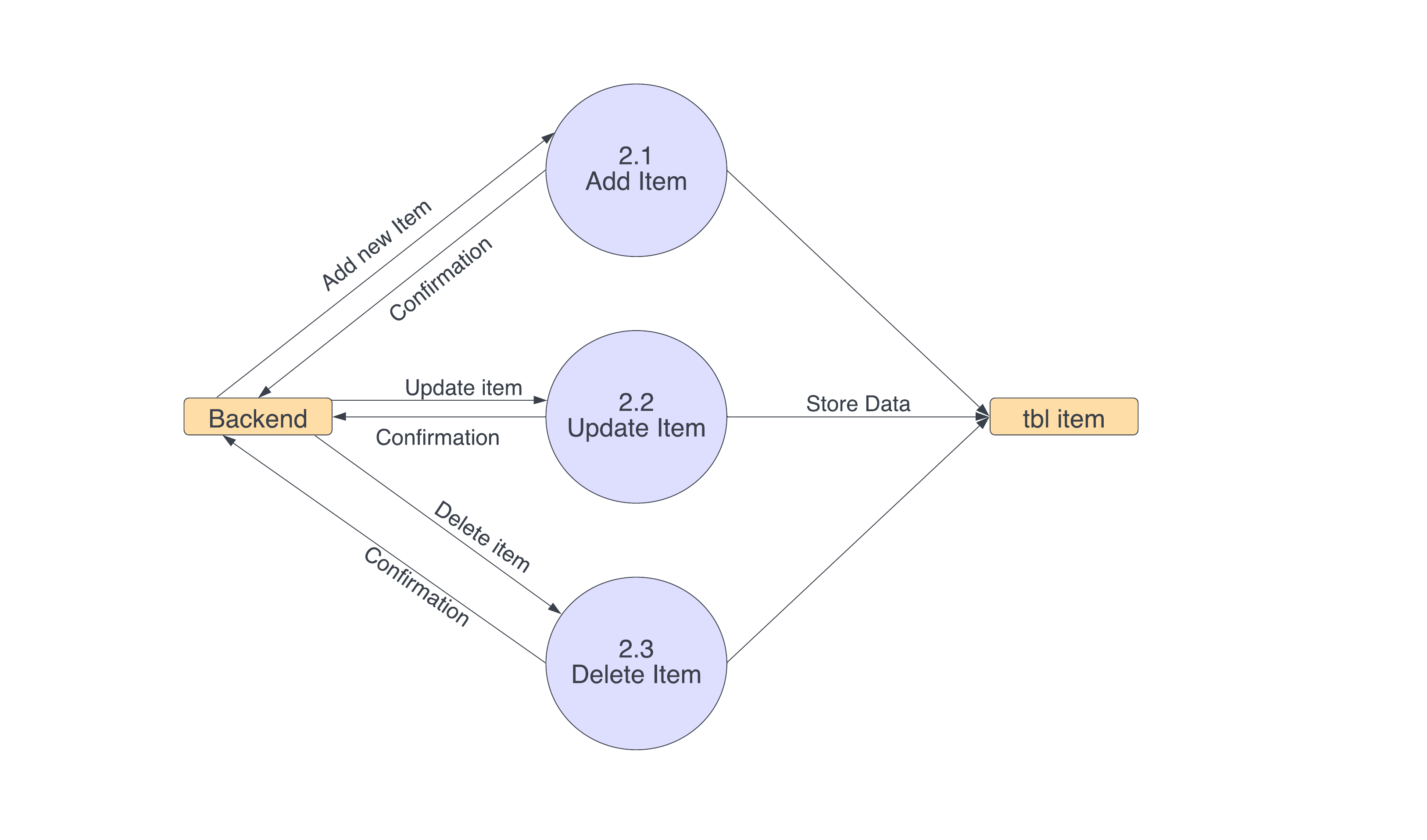
Basically the content of DFD level 1 is in accord to the context diagram. It illustrates the broadened concept of the former level and explains the included data and process. These data and processes were determined to complete the function of the project.

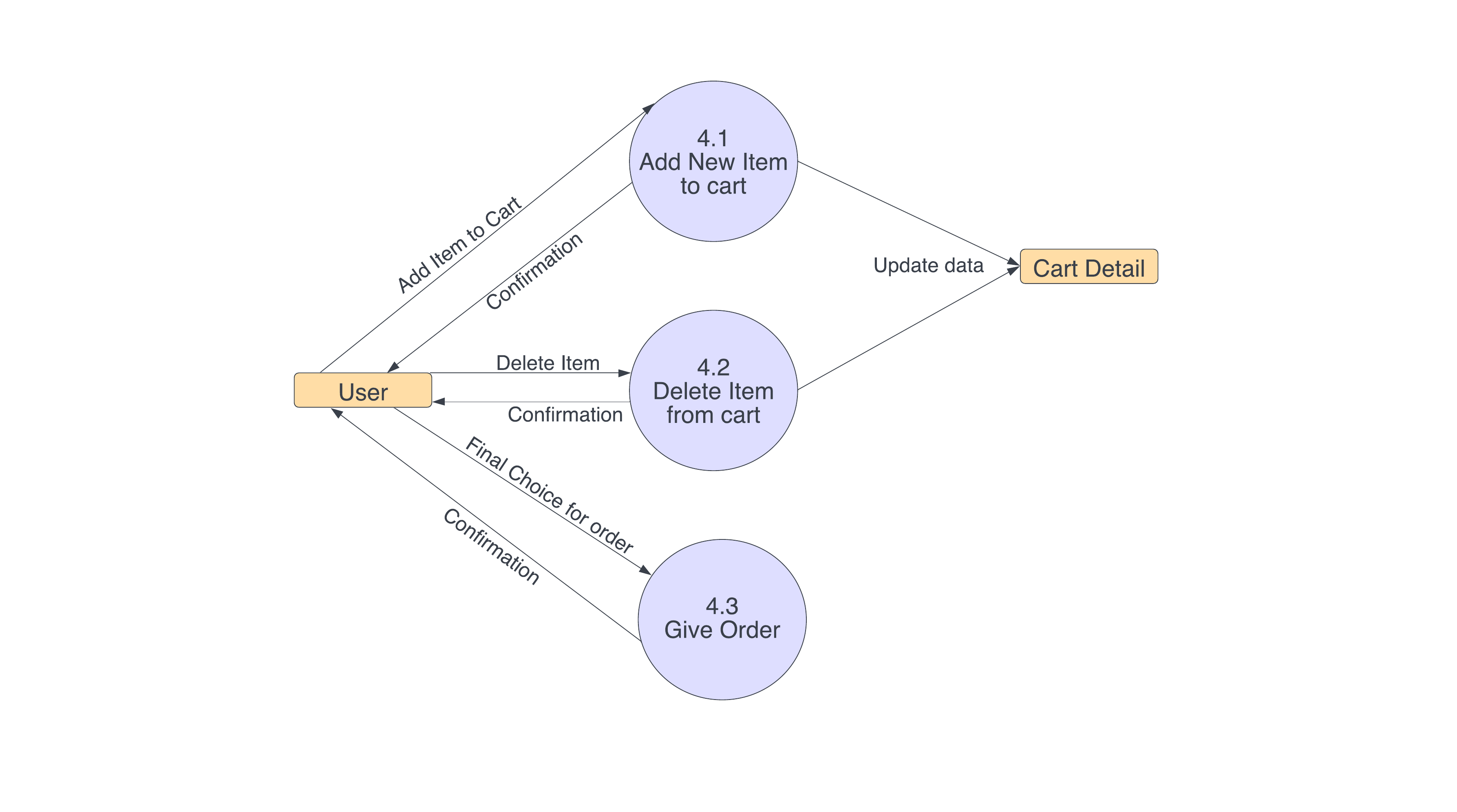
To design the DFD level 1 of E-Commerce Website, think of the included processes that makes up the main process. Then these processes will help you know the data that needs to be handled by the system.

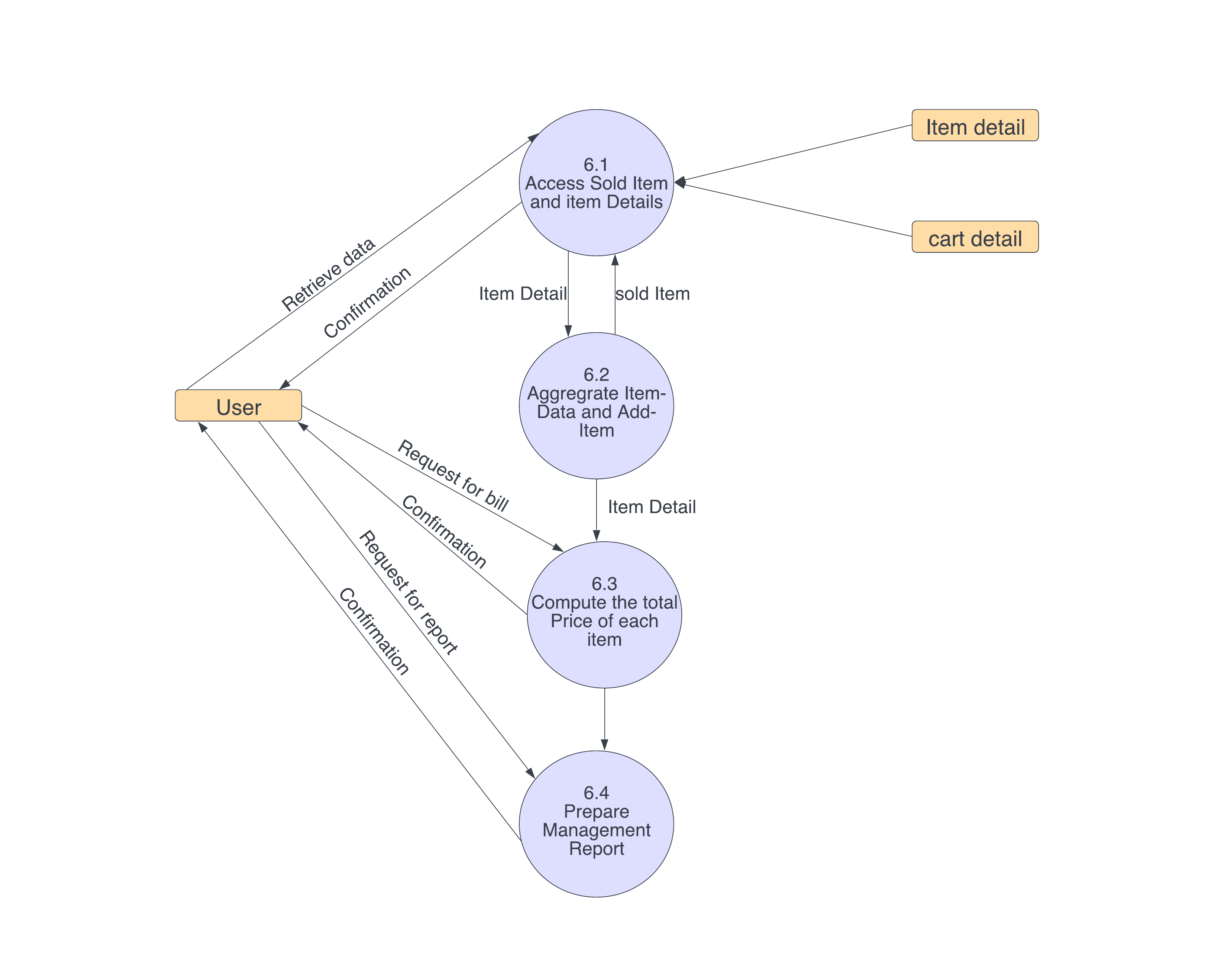
|  |  |
| --- | --- |
| 2.3.2 | DFD Level 2 |

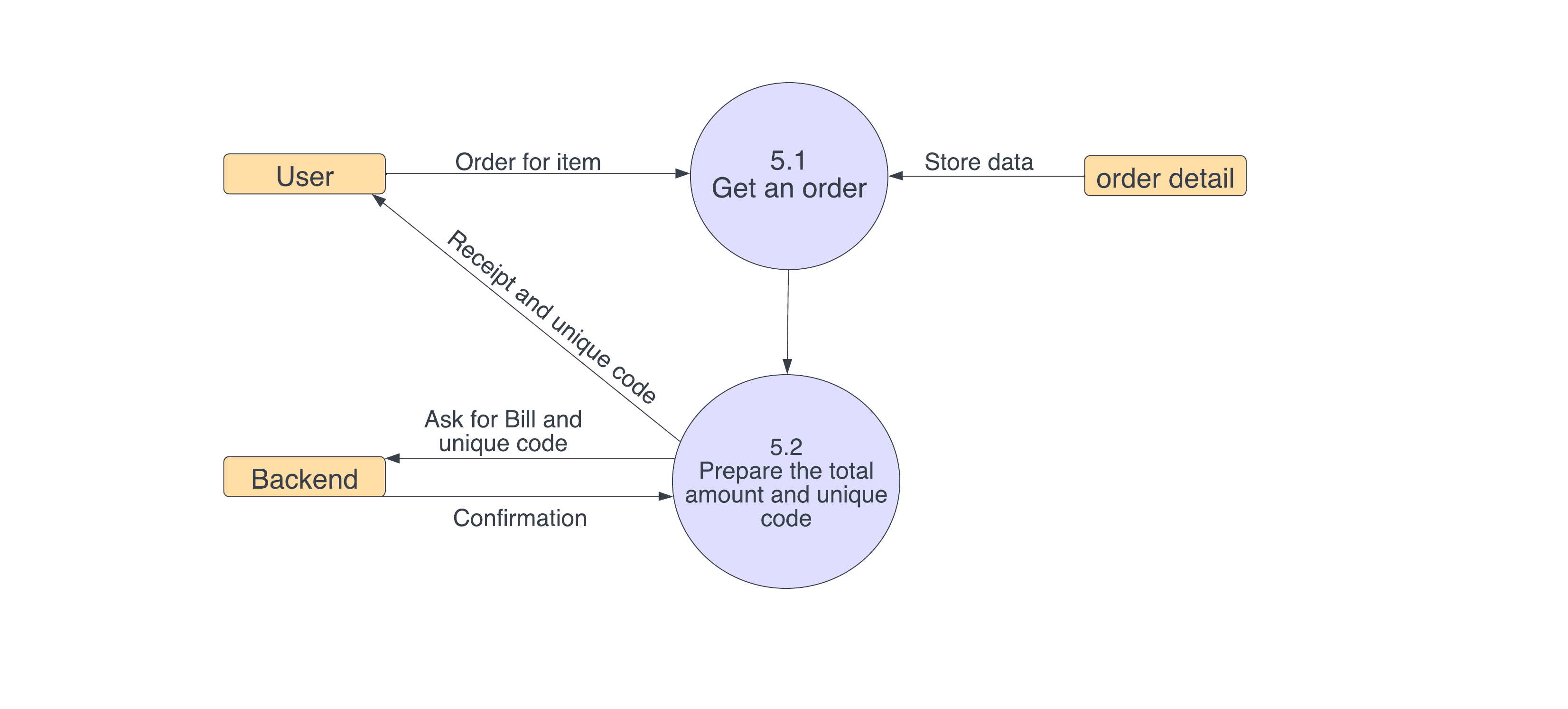
The **Level 2 DFD for E-commerce Website** portrays deeper concepts of DFD level 1. It can be used to plan or record all of the specific/necessary information about how the system works.

After presenting the E-Commerce Website DFD levels 0 and 1, next to that is level 2. The DFD Level 2 was considered as the highest abstraction of E-Commerce Website Data Flow Diagram. This level is expected to have the complete and detailed illustration of the project.









**E-Commerce Website DFD** **Level 2**represents the system’s specified modules as well as the data that flows between them. These modules includes the data flow, processes, external entities and the databases. Each of them serves as the guide on how to build the system.

As you see, when you build the levels of data flow diagrams, the connections of the transactions and data also broadens and gets more specific. Considering the dataflow levels mentioned above, you can determine well the importance of breaking the processes into more specific manner.

|  |  |
| --- | --- |
| 2.4 | Software Requirement Specification in IEEE Format |
| 2.4.1 | User and shopkeeper Side |

**Software** – chrome etc.

**Hardware** - Computer setup or mobile phone.

**Service** - Internet Connection.

2.4.2   Functional Requirements

Search Product

|  |  |
| --- | --- |
| **Use Case Name** | Search product |
| **Trigger** | The user assesses the Website. |
| **Precondition** | The Web is displayed with grids for searching |
| **Basic Path** | 1.     The User chooses how to search the Web site. The choices are by Shopkeeper, by Category, and by Keyword.  2.     If the search is by Shopkeeper, the system creates and presents an alphabetical list of all Shopkeepers in the database. In the case of a product with multiple Shopkeepers, each is contained in the list.  3.     The User selects a Shopkeeper.  4.      The system creates and presents a list of all products by that Shopkeeper in the database.  5.     The User selects a product.  6.     The system displays the Abstract for the product.  7.     The User selects to buy the product or to return to the product list or to the previous list. |
| **Alternative Paths** | In step 2, if the User selects to search by category, the system creates and presents a list of all categories in the database.  1.   The User selects a category.  2.   The system creates and presents a list of all products in that category in the database. Return to step 5.  In step 2, if the User selects to search by keyword, the system presents a dialog box to enter the keyword or phrase.            3.            The User enters a keyword or phrase.  4.   The system searches the Abstracts for all products with that keyword or phrase and creates and presents a list of all such products in the database. Return to step 5. |
| **Postcondition** | The selected product is added to the cart. |
| **Exception Paths** | The User may abandon the search at any time. |

Add Shopkeeper

|  |  |
| --- | --- |
| **Use Case Name** | Add Shopkeeper |
| **Trigger** | The Shop owner selects to add a new Shopkeeper to the database. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen. |
| **Basic Path** | 1.     The system presents a blank grid to enter the Shopkeeper information.  2.     The Shop owner enters the information and submits the form.  3.     The system checks that the name and email address fields are not blank and updates the database. |
| **Alternative Paths** | If in step 2, either field is blank, the Shop owner is instructed to add an entry. No validation for correctness is made. |
| **Postcondition** | The Shopkeeper has been added to the database. |
| **Exception Paths** | The Shop owner may abandon the operation at any time. |
| **Other** | The Shopkeeper information includes the name, mailing address and email address. |

Add Delivery person

|  |  |
| --- | --- |
| **Use Case Name** | Add the delivery agent. |
| **Trigger** | The Shop owner selects to add the delivery agent in the database. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen. |
| **Basic Path** | 1.     The Shop owner selects the Agent list.  2.     The system creates and presents an alphabetical list of people in the category.  3.     The Shop owner selects an Add agent.  4.     The system presents the database information in grid form for modification.  5.     The Shop owner updates the information and submits the form.  6.     The system checks that required fields are not blank. |
| **Postcondition** | The database has been updated. |
| **Exception Paths** | If the person is not added in the database, the use case is abandoned. In addition, the Shop owner may abandon the operation at any time. |

Update Product Status

|  |  |
| --- | --- |
| **Use Case Name** | Update Product Status |
| **Trigger** | The Shop owner selects to update the status of a product in the database. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen and the product is already in the database. |
| **Basic Path** | 1.     The system creates and presents an alphabetical list of all active products.  2.     The Shop owner selects the product to update.  3.     The system presents the information about the product in grid format.  4.     The Shop owner updates the information and resubmits the form. |
| **Postcondition** | The database has been updated. |
| **Exception Paths** | If the product is not already in the database, the use case is abandoned. In addition, the Shop owner may abandon the operation at any time. |

Send Communication

|  |  |
| --- | --- |
| **Use Case Name** | Send Communication |
| **Trigger** | The user selects to send a communication to a Shopkeeper. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen. |
| **Basic Path** | 1.     The system presents an alphabetical list of Shopkeepers.  2.     The user selects a Shopkeeper.  3.     The system invokes the Shop owner’s email and phone number.  4.     Users can send an email or make a call. |
| **Alternative Paths** | None. |
| **Postcondition** | The communication has been sent. |
| **Exception Paths** | The user may abandon the operation at any time. |

Add Product

|  |  |
| --- | --- |
| **Use Case Name** | Add Product. |
| **Trigger** | The Shop owner selects to transfer an approved product to the Website. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen. |
| **Basic Path** | 1.     The system creates and presents an alphabetical list of the active products.  2.     The Shopkeeper clicks on add product and fills the details.  3.     The system accesses the Online Database and transfers the product and its accompanying information to the Online database.  4.     The product is added to the active product database. |
| **Alternative Paths** | None. |
| **Postcondition** | The product is properly transferred. |
| **Exception Paths** | The Shop owner may abandon the operation at any time. |

Remove Product

|  |  |
| --- | --- |
| **Use Case Name** | Remove Product |
| **Trigger** | The Shop owner selects to remove a product from the active product database. |
| **Precondition** | The Shop owner has accessed the Product Manager main screen. |
| **Basic Path** | 1.     The system provides an alphabetized list of all active products.  2.     The shop owner selects a product.  3.     The system displays the information about the product and requires that the Shop owner confirm the deletion.  4.     The Shop owner confirms the deletion. |
| **Alternative Paths** | None. |
| **Postcondition** | The product is removed from the database. |
| **Exception Paths** | The Shop owner may abandon the operation at any time. |
| **Other** | Find out from the Shop owner to see if the product and its information should be archived somewhere. |

|  |  |
| --- | --- |
| 2.4.3 | Detailed Non-Functional Requirements |

The data descriptions of each of these data entities is as follows:

**Shopkeeper Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of Shopkeeper | Name |
| Email Address | Text | Internet address | Email Address |
| Phone no. | Integer | Phone number. | May be several. |

**Delivery person Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of delivery person | Name |
| ID | Integer | ID number of delivery person. | ID |
| Email Address | Text | Internet address | Email Address |
| Phone no. | Integer | Phone number. | May be several. |

**Product Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of Product |  |
| Shopkeeper | Pointer | Shopkeeper entity | Name of principle Shopkeeper |
| Other Shopkeepers | Text | Other Shopkeepers is any; else null | Not a pointer to an Shopkeeper entity |
| Contents | Text | Body of product | Contains Abstract as first paragraph. |
| Category | Text | Type of product | May be several |

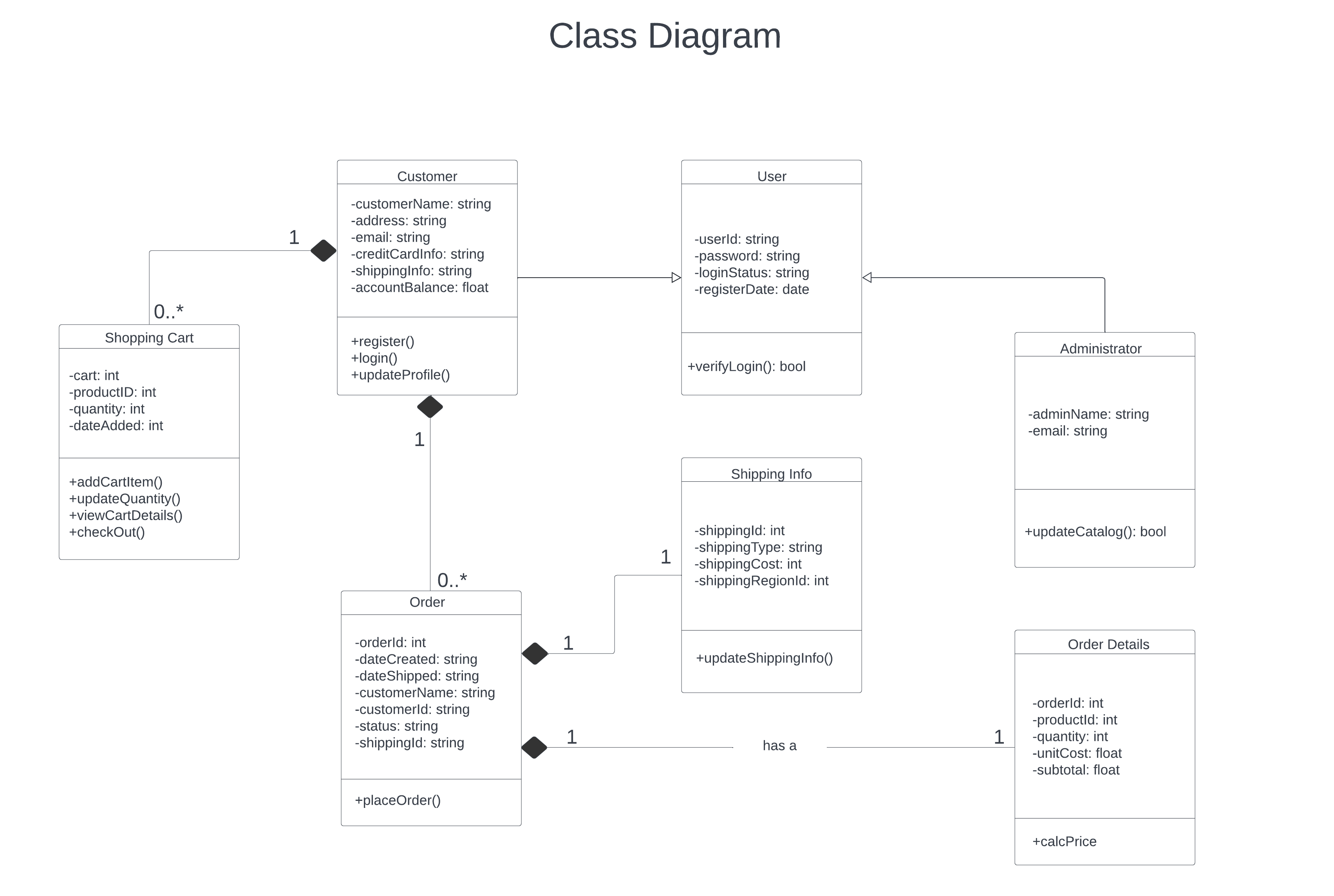
|  |  |
| --- | --- |
| **3.** | **Design Phase** |
| 3.1 | Class Diagram |

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

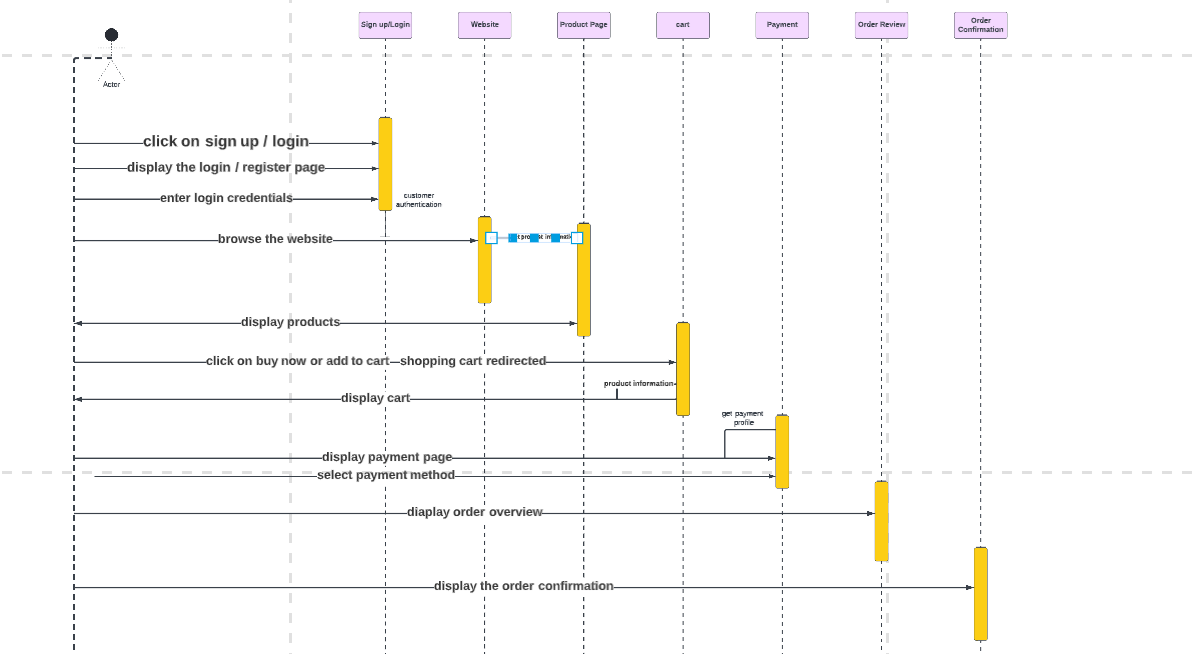
The purpose of the class diagram can be summarized as −

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

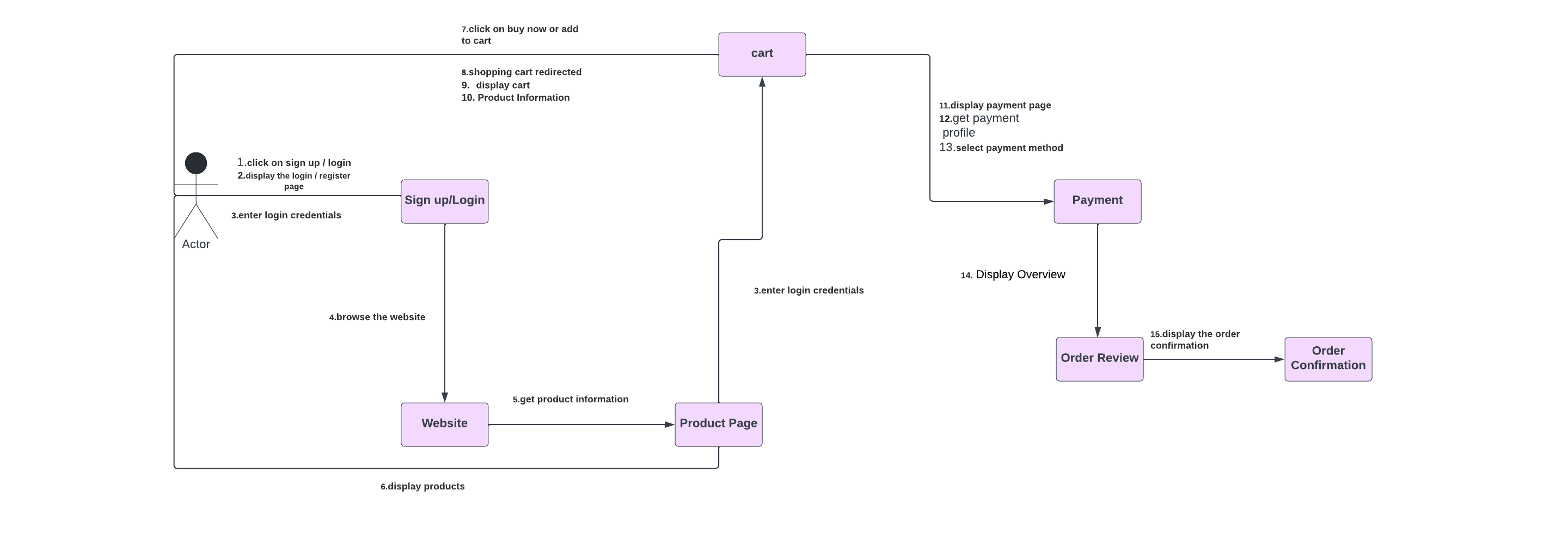


|  |  |
| --- | --- |
| 3.2 | Sequence Diagram |

 A **sequence diagram** is the most commonly used **interaction** diagram. **Interaction diagram –** An interaction diagram is used to show the **interactive behavior** of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system. **Sequence Diagrams –** A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place



|  |  |
| --- | --- |
| 3.3 | Collaboration Diagram |



3.5 State Chart Diagram

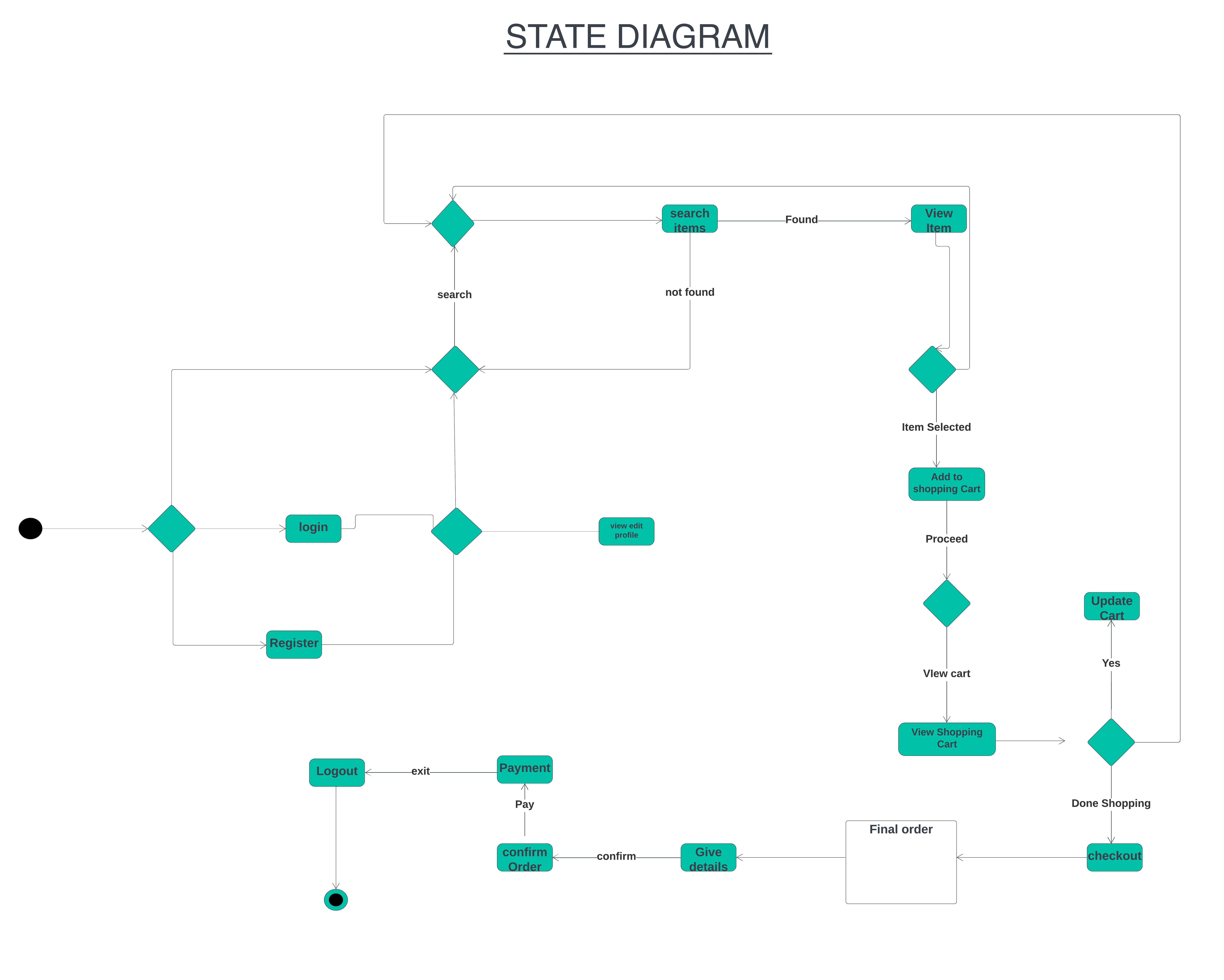
Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams −

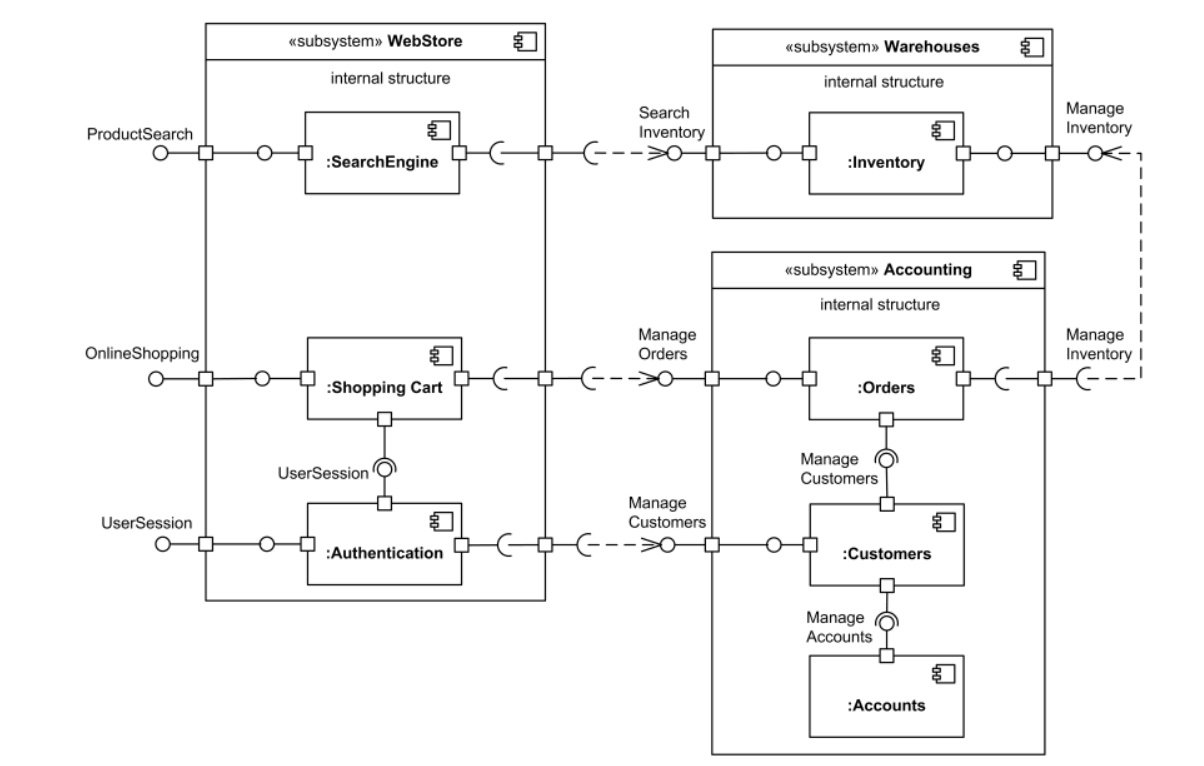
* To model the dynamic aspect of a system.
* To model the life time of a reactive system.
* To describe different states of an object during its life time.
* Define a state machine to model the states of an object.



**4.0 Implementation**

4.1 Component Diagram

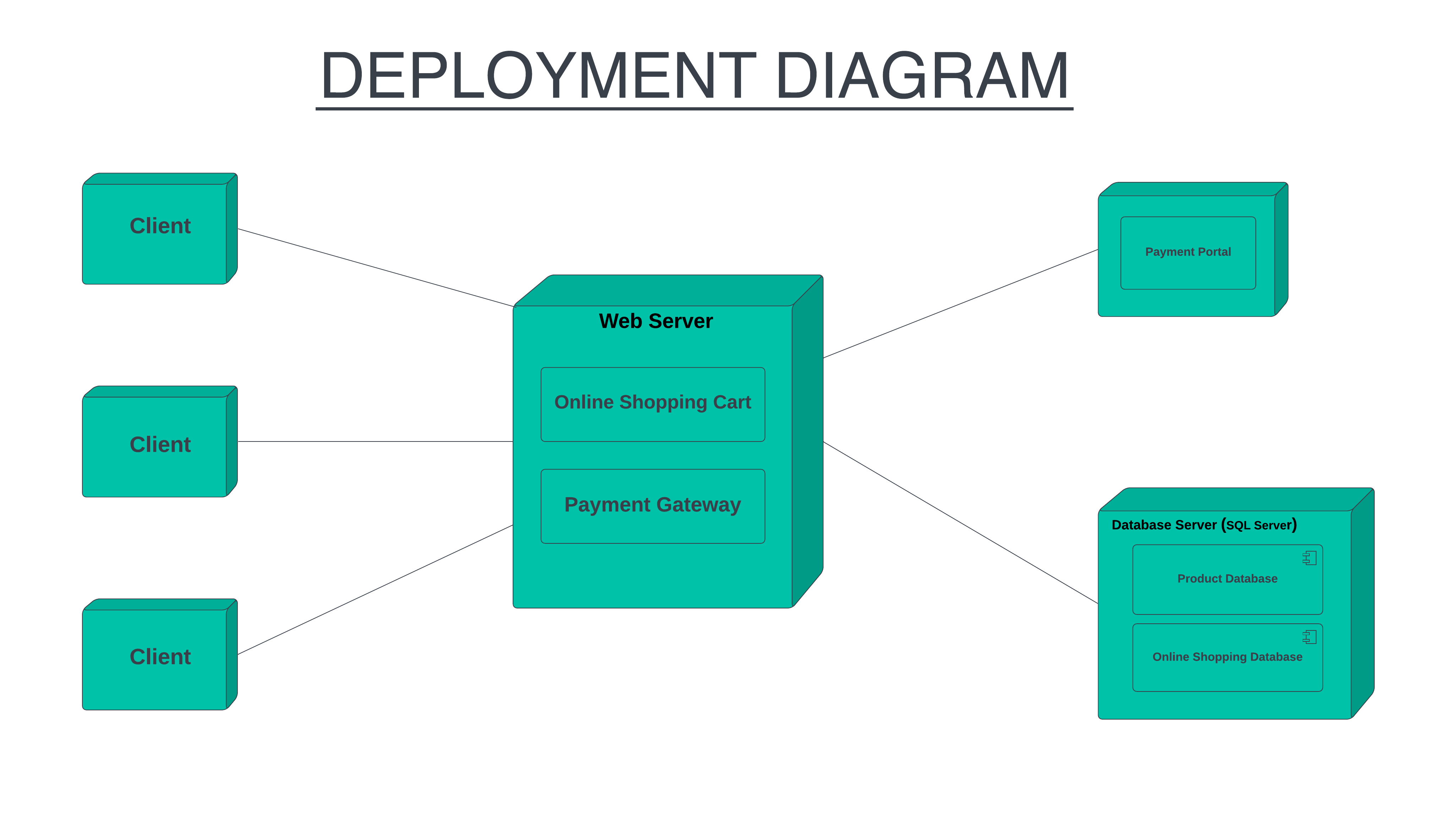
The purpose of a component diagram is to show the relationship between different components in a system. For the purpose of UML 2.0, the term "component" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.



4.2 Deployment Diagram

Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.

Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.



**5.0 Testing**

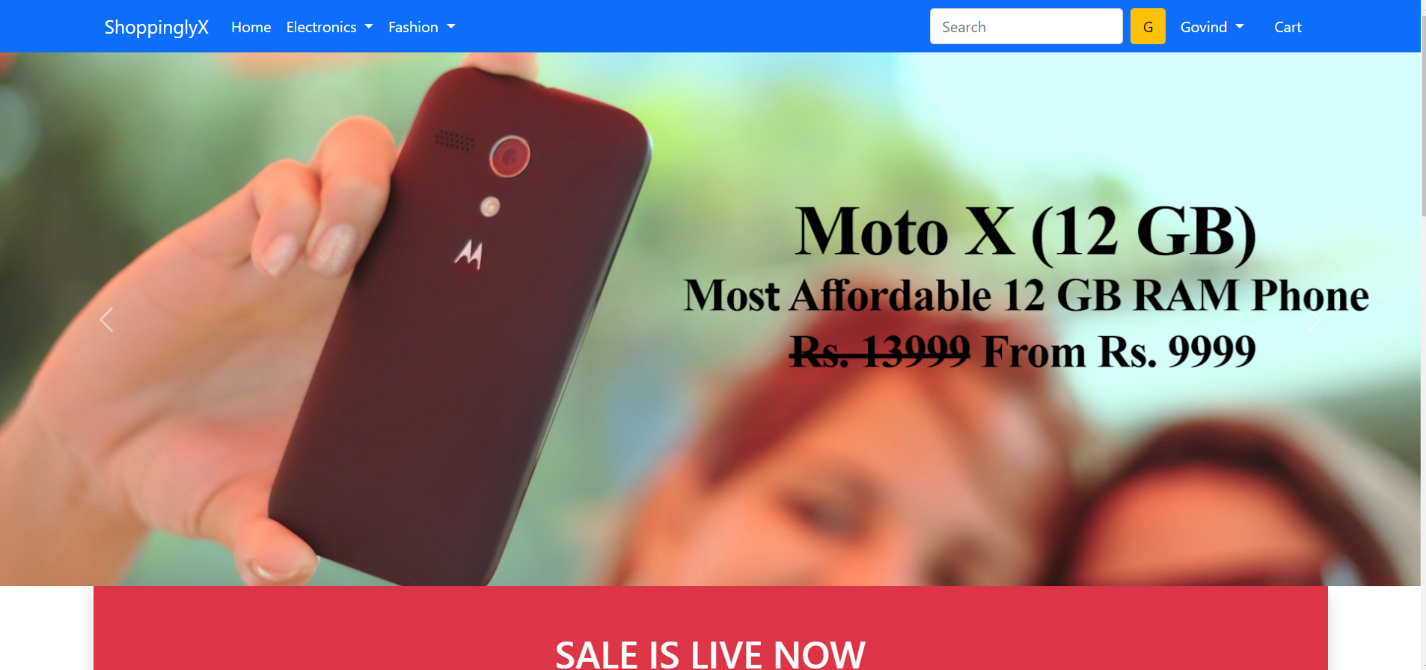
5.1 Test Plan

To test different working essential features of the website which are the most crucial and important parts of the website.

5.2 Test Cases

### #1) Homepage – Hero Image

Homepages of retail sites are busy. They have a lot going on. But almost all of them have a Hero Image:



This is the kind of the clickable image (a slideshow of sorts) that occupies the majority of the page

**The following are a few things to test:**

* Is it going to auto scroll?
* If yes, at what interval will the image be refreshed?
* When the user hovers over it, is it still going to scroll to the next one?
* Can it be hovered on?
* Can it be clicked on?
* If yes, is it taking you to the right page and right deal?
* Is it loading along with the rest of the page or loads last in comparison to the other elements on the page?
* Can the rest of the content be viewed?
* Does it render the same way in different browsers and different screen resolutions?



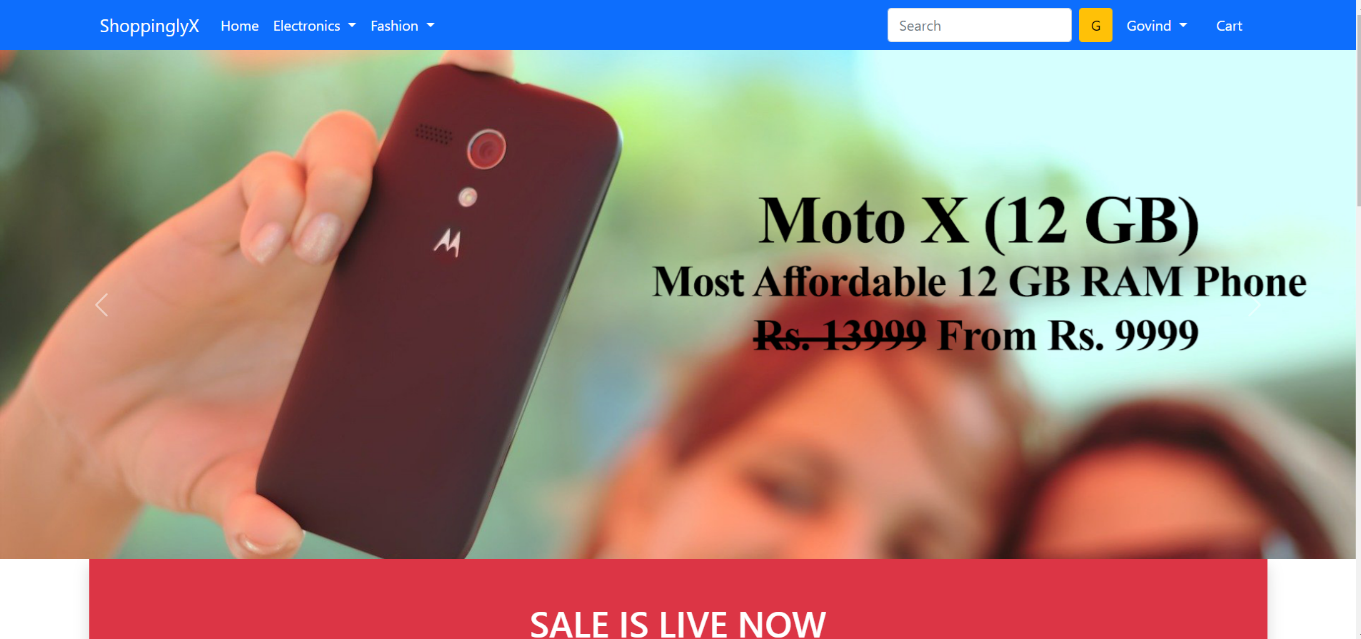
### #2) Filters

Search algorithms are very important for the success of a retail site because we can’t always place what the users want to see right in front of their eyes.

**Common tests are:**

* Search based on the Product name, brand name, or something more broadly, the category. **For example** Camera, Canon EOS 700D, electronics, etc.
* Search Results have to be relevant
* Different sort options have to be available- based on Brand, Price, and Reviews/ratings etc.
* How many results to display per page?
* For multi-page results, are there options to navigate to them
* Also, search happens in many places. Please take the search drilling down into multiple levels into consideration when validating this functionality. ***For example:*** When I search on the home page, I might see something like this:

When I navigate to categories and go to a sub-category, maybe movies, this is what I am going to see:

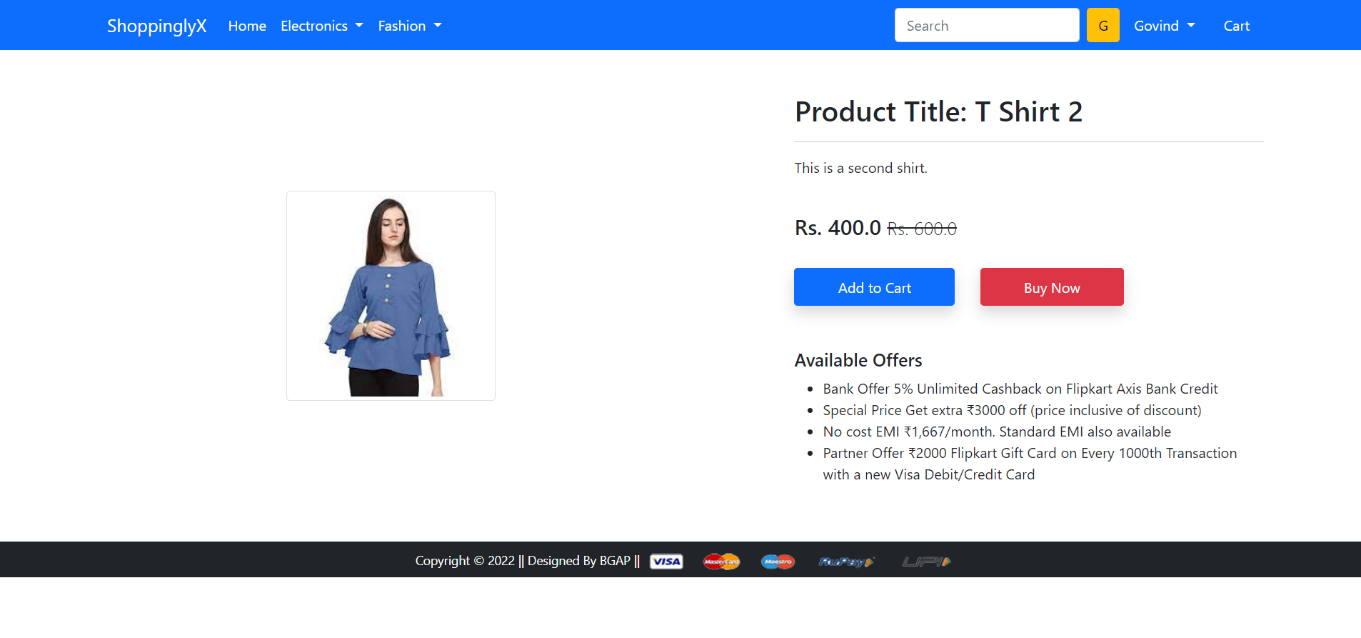


### #3) Product Details Page

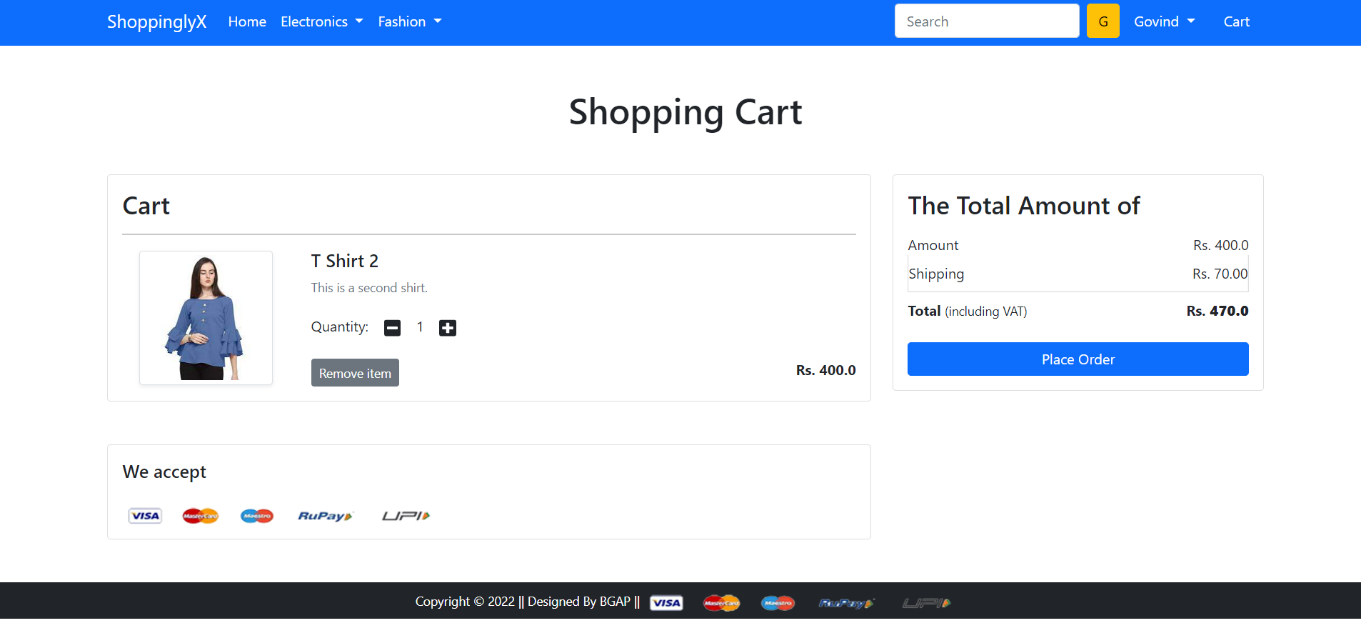
Once a user finds a product either through search or by browsing or by clicking on it from the homepage, the user will be taken to the product information page.

**Check:**

* Image or images of the product
* Price of the product
* Product specifications
* Reviews
* Check out options
* Delivery options
* Shipping information
* In-stock/Out of stock
* Multiple color or variations options
* Breadcrumb navigation for the categories (highlighted in Red below). If navigation such as that is displayed, make sure every element of it is functional.



### #4) Shopping Cart

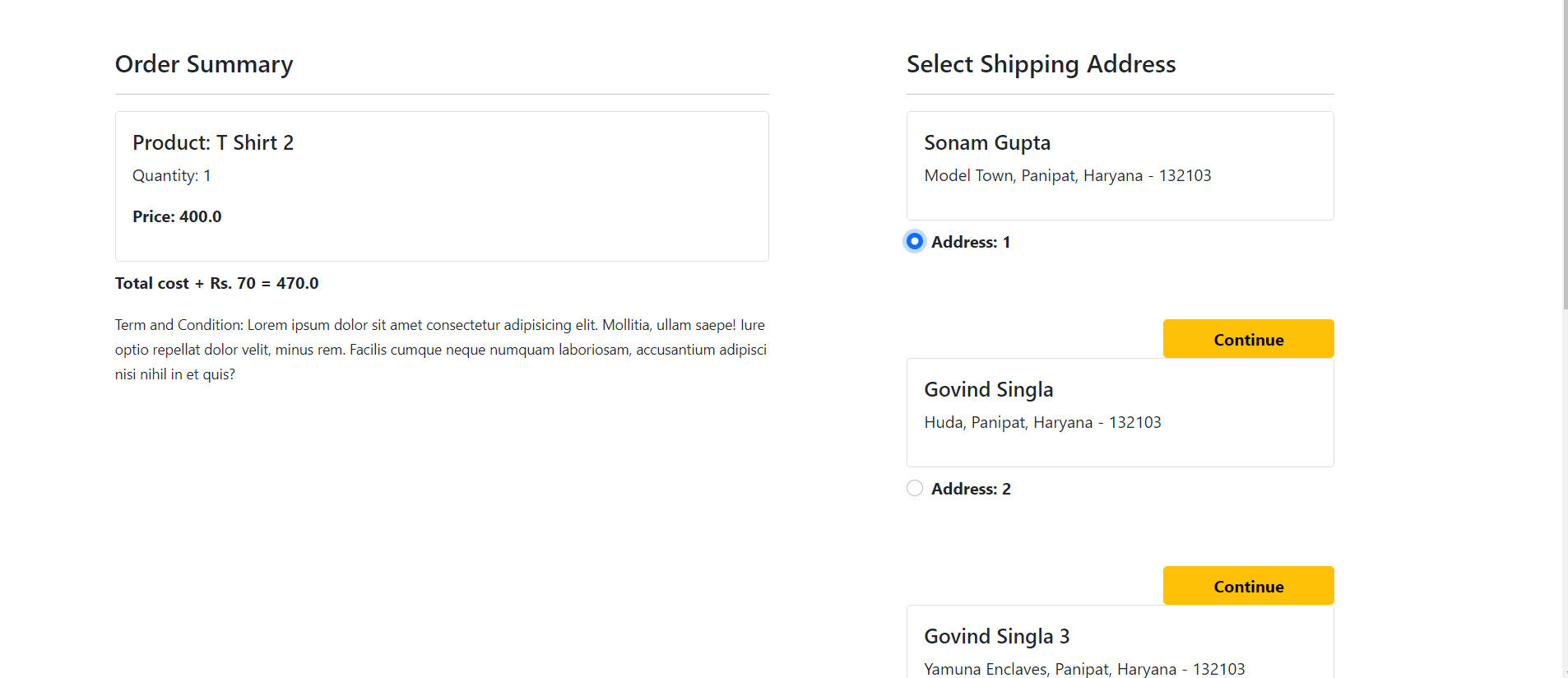


This is the penultimate stage before the user commits to the purchase.

**Test the following:**

* Add items to the cart and continue shopping
* If the user adds the same item to the cart while continuing to shop, the item count in the shopping cart should get incremented
* All items and their totals should be displayed in the cart
* Taxes as per location should be applied
* A user can add more items to the cart- total should reflect the same
* Update the contents added to the cart- total should reflect that too
* Remove items from the cart
* Proceed to checkout
* Calculate Shipping costs with different shipping options
* Apply coupons
* Don’t check out, close the site, and come back later. The site should retain the items in the car

### #5) After-Order Tests

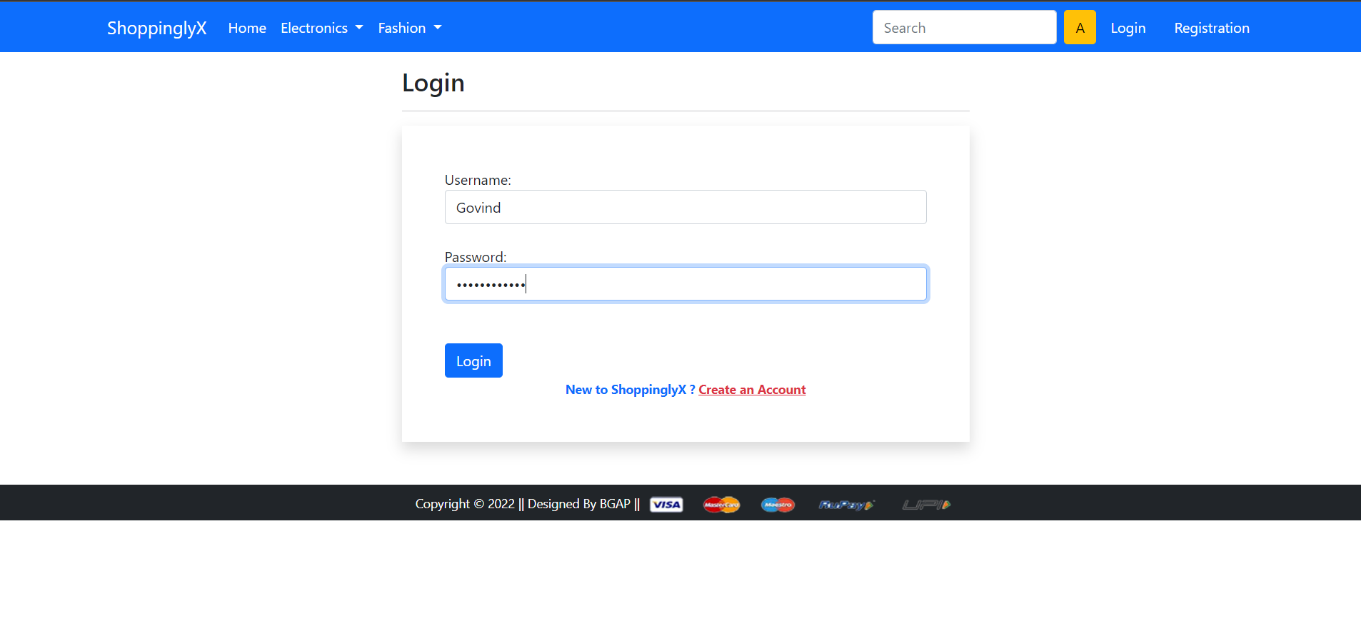


**Check:**

* Change the Order
* Cancel the Order
* Track the Order
* Returns

### #6) Other Tests

* Login
* FAQs
* Contact Us page
* Customer Service page etc.



* 1. Test Reports

Now, that we have a few tests listed out, let’s move on to a couple of **finishing thoughts on eCommerce Testing**.

A website should work – not just on computers but on mobile devices too. It needs to be responsive and secure. The Database should be optimized and the ETL processes should help maintain a Data Warehouse that aids for OLAP and BI. E-commerce testing should focus on all of that.

However, the most important part of E-Commerce Testing is whether the visitors are converting into paying customers or not. The number of visits that are becoming the customer is called “Conversion Rate”.

So does one feature promote better conversion as opposed to another, is important testing. That is why A/B testing and Usability Engineering for E-Commerce sites are gaining prominence.