***E-Commerce Documentation***

***Project Objective***

The objective of this project is to develop a general-purpose e-commerce store that has 5 functionalities, where any kind of product can be bought from the comfort of home through the Internet. However, for implementation purposes, this will deal with an online shopping for Book products.

***Project Description***

* Any member can register and view available products
* Only registered users can cash out the products
* There are 2 roles available
  + User
  + Visitor
* Visitors can view, search, and compare product/s that are available
* User can view and purchase the product/s

***User Story***

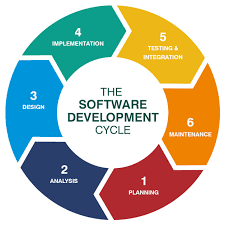
* As a Shopper I want to view a list of products so I can select some to purchase
* As a Shopper I want to review my cart so I can make adjustments prior to checkout
* As a Shopper I want to check out so I can get my products shipped to me
* As a Shopper I want to review my orders so I can see what I’ve purchased in the past

***Functionalities***

* Login
* Register
* View Products
* Cart History
* Checkout/Shipping Page

***The SDLC***

**Software Development Life Cycle** is the application of standard business practices to building software applications. It’s typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain.



### **1. Planning**

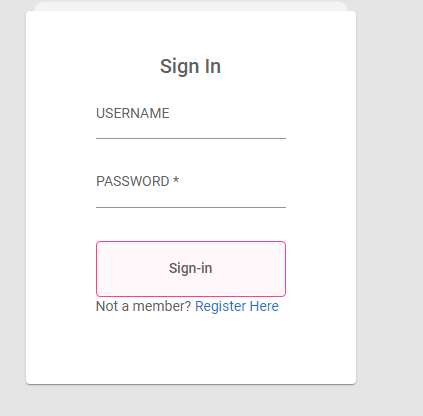
* Design my store and add products
* Create a Checkout experience
* Account Credentials for register/ login page.

### **2. Define Requirements**

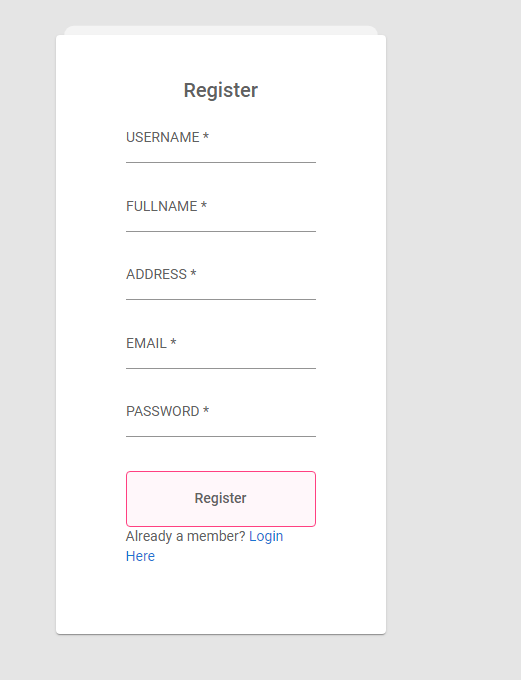
* Login/Register Page
* Home page that will display all the available products
* Adding items to cart, being able to check out only if the customer is a registered user.
* Header and the footer Page.

### **3. Design and Prototyping**

The Design phase models the way a software application will work.



Sign In will be for registered users who want to cash out their items from the cart.



Register will be for new users who want to continue using our site to purchase and explore more.

### **4. Software Development**

This is the actual writing of the program. These systems help developers track changes to the code. They also help ensure compatibility between different team projects and to make sure target goals are being met.

Finding and fixing errors and glitches is critical. Tasks often hold up the development process, such as waiting for test results or compiling code so an application can run.

**5. Testing**

It’s critical to test an application before making it available to users. Much of the testing can be [automated, like security testing.](https://phoenixnap.com/blog/automated-security-testing-best-practices) Other testing can only be done in a specific environment – consider creating a simulated production environment for complex deployments. Testing should ensure that each function works correctly. Different parts of the application should also be tested to work seamlessly together—performance test, to reduce any hangs or lags in processing. The testing phase helps reduce the number of bugs and glitches that users encounter. This leads to a higher user satisfaction and a better usage rate.

### **6. Deployment**

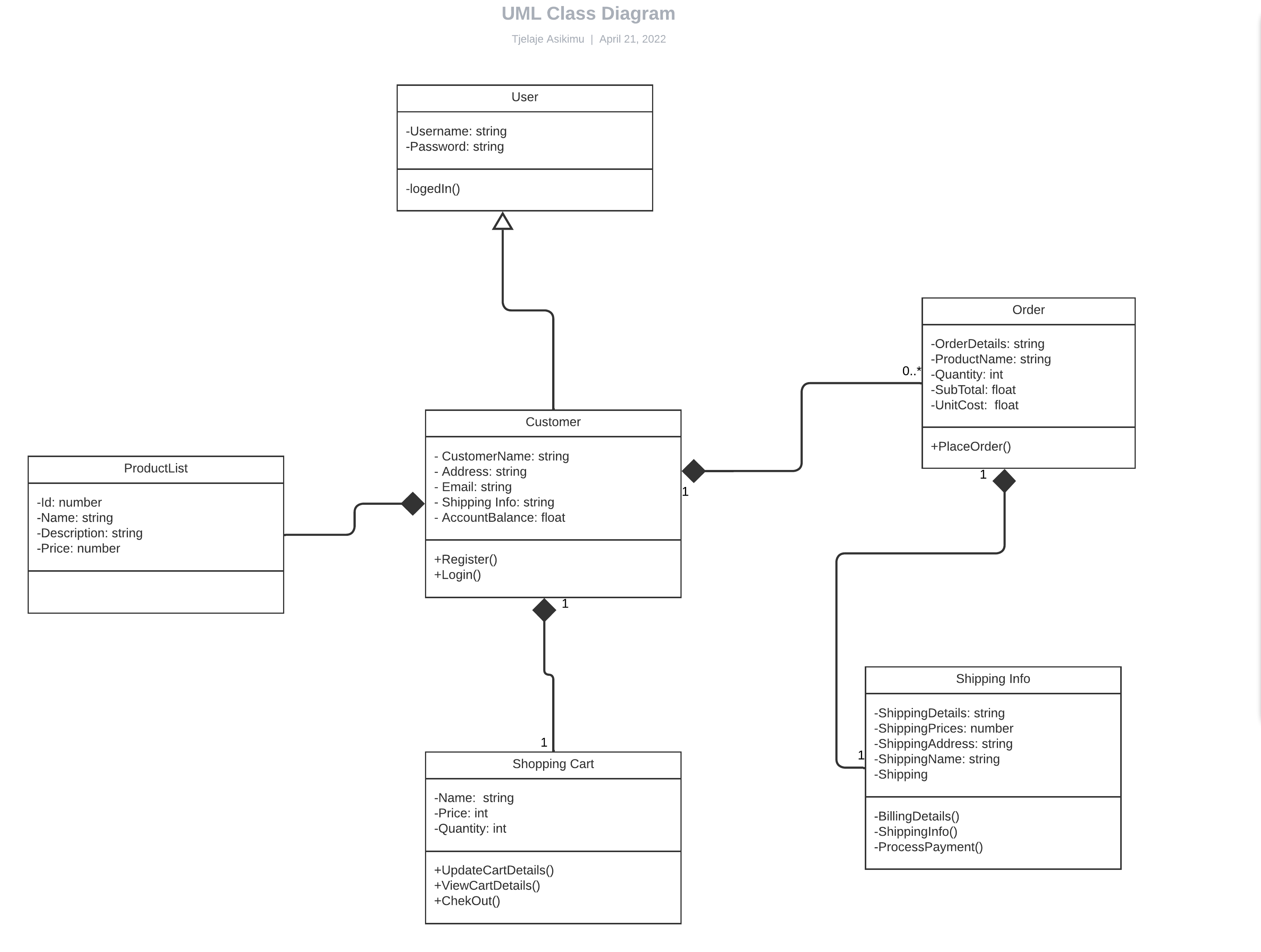
Deployment will take place when everything is done.

***OOP principles***

* Polymorphism refers to one name with many forms. It is the ability of one function to perform in different ways. In other words, it refers to an object’s ability to take on more than one single form.
* Inheritance It is the ability to acquire the properties of existing classes and create new ones. Inheritance allows you to reuse code without having to rewrite it in a program.
* Encapsulation It is a group of properties and members under a single class or Object. Programs can be really long and there can easily be a ton of moving parts in it.
* Abstraction is the process to hide the internal details and showing only the functionality

***Class Diagram***

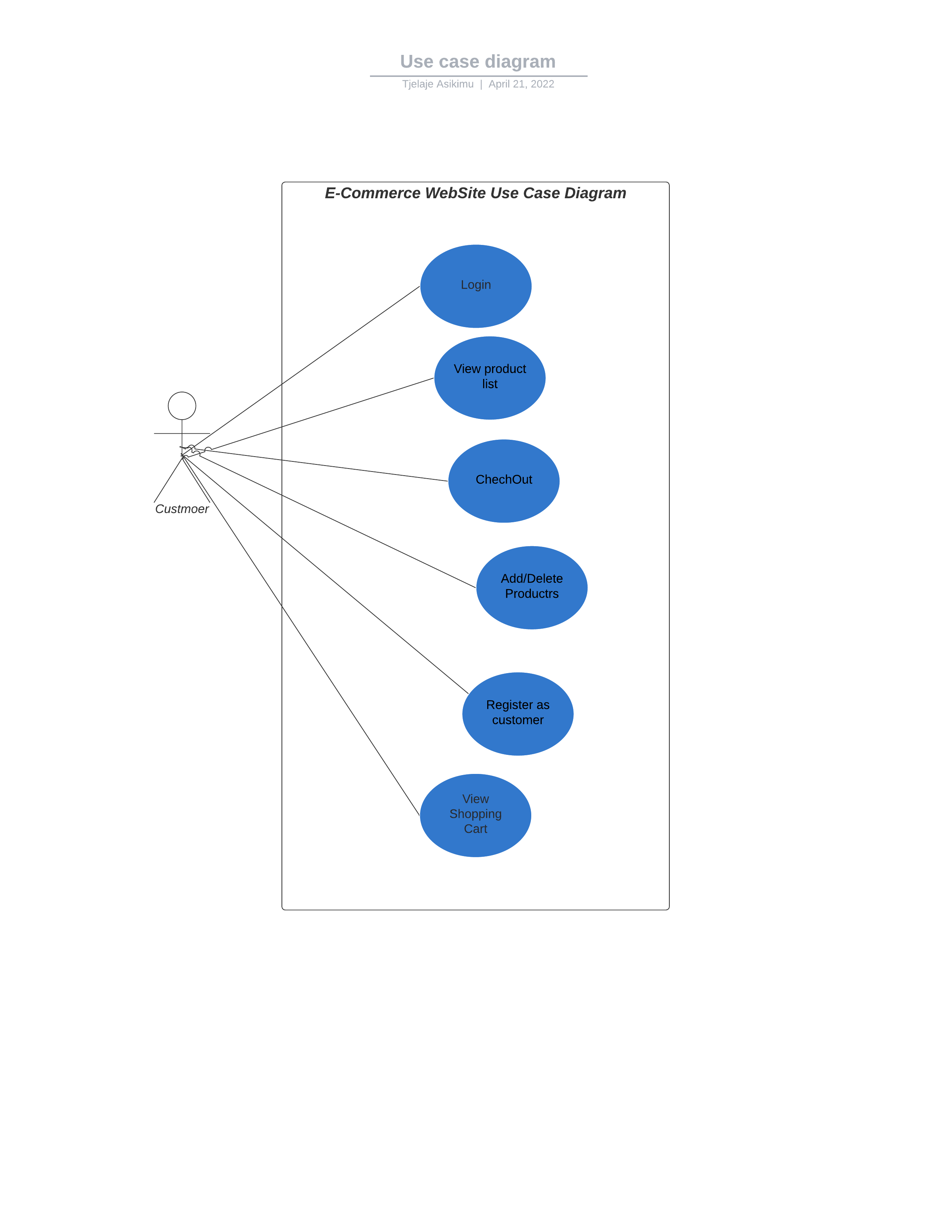
A class diagram is a UML diagram type that describes a system by visualizing the different types of objects within a system and the kinds of static relationships that exist among them. It also illustrates the operations and attributes of the classes



***Use Case Diagram***

These diagrams identify the interactions between the system and its actors.

It does not show the detail of the use cases: It only summarizes some of the relationships between use cases, actors, and systems



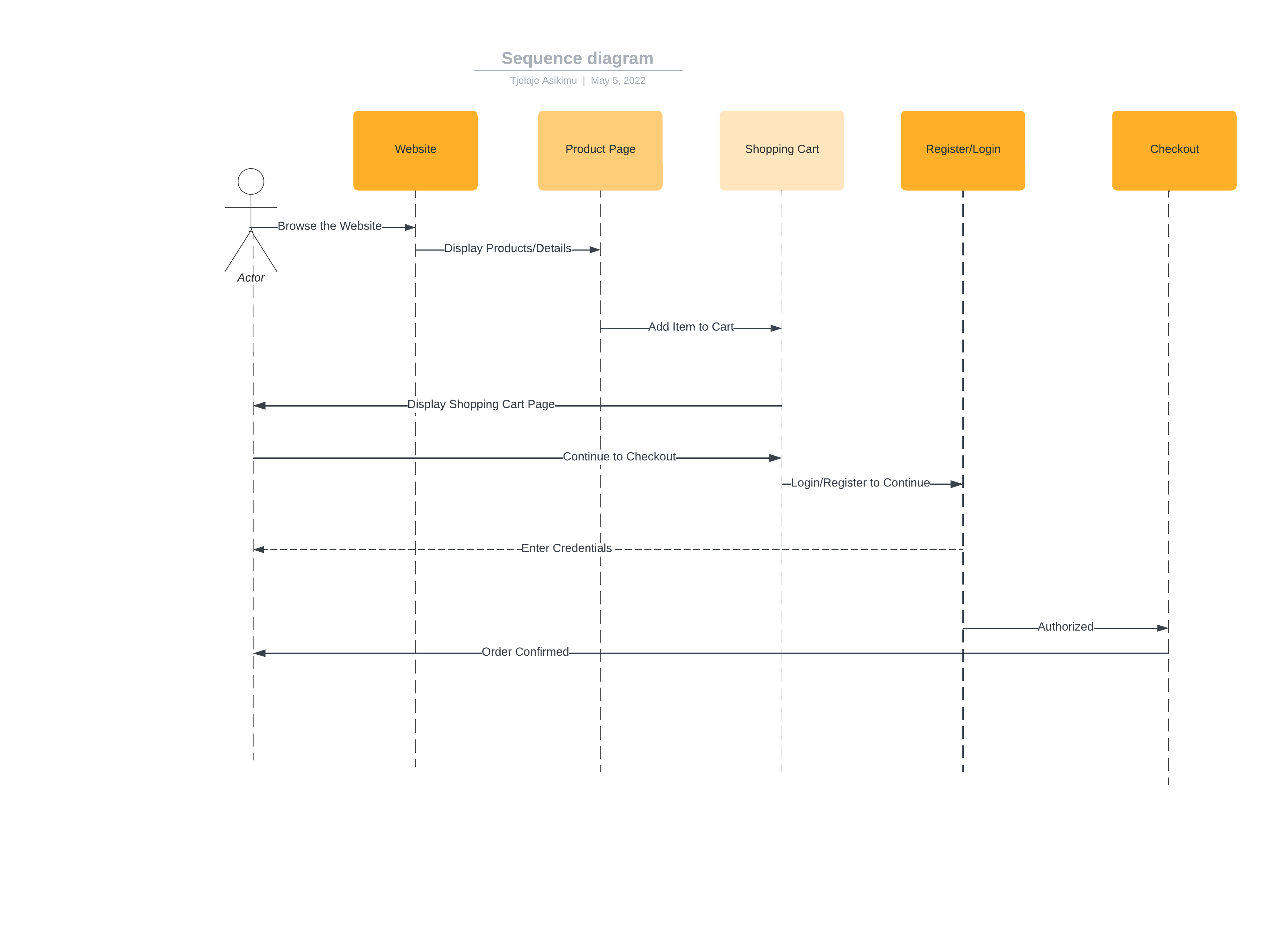
***Entity Relationship Diagram***

ER model is a design or blueprint of a database that can later be implemented as a database. It is also a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.



***Sequence Diagram***

Shows how the object/classes in the diagram interacts with each other.

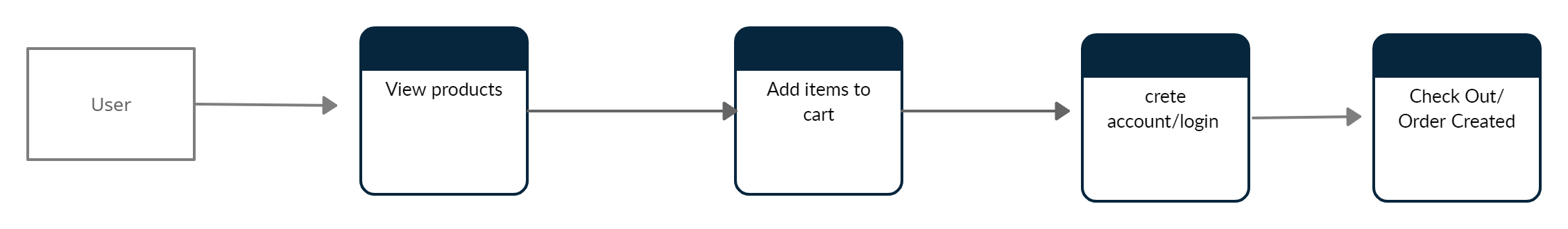


***Architecture Diagram***

A visual representation that maps out the physical implementation for components of a software system

***Data Flow Diagram***

A data flow diagram (DFD) maps out the flow of information for any process or system.



***UI Diagram***

The User Interface (UI) diagram is where you build out the visual structure of your application.

