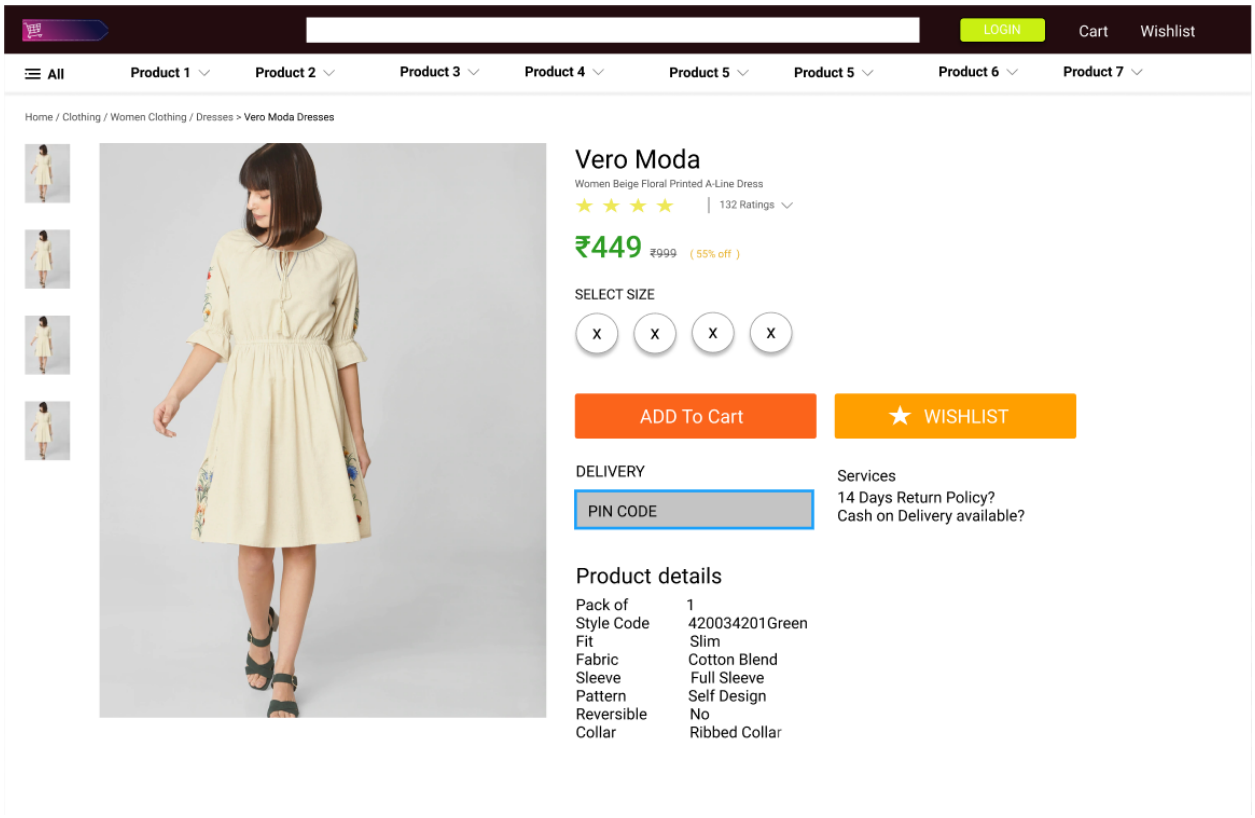
Designing a E-Commerce Service

E-Commerce platform



E commerce platform

Sumit Kapoor

Instructor

Munny Kumar

SDE @ LCX

storyofcoder1@gmail.com

Gurugram, India

What is E Commerce?

E Commerce, also known as electronic commerce or internet commerce, refers to the buying and selling of goods or services using the internet, and the transfer of money and data to execute these transactions, platforms, such as Amazon, Flipkart, walmart, and Myntra, etc . Global retail ecommerce sales are projected to reach $27 trillion by 2020.

Requirements and Goals of the System

We'll focus on the following set of requirements while designing Instagram:

Functional Requirements

1. Users can perform searches based on product titles.
2. Users should be able to add orders in cart / wishlist.
3. Users should be able to place orders.
4. Users should be able to pay online.
5. Users can see their orders history.
6. The system should generate and display a priority product of top from all other products.

Non-functional Requirements

1. Our service needs to be highly available.
2. The acceptable latency of the system is near 200ms for product search.
3. The system should be highly reliable during place order.

Not in scope: Adding tags to photos, searching photos on tags, commenting on photos, tagging users to photos, who to follow, etc.

Goals

1. Our goal is to create an ecommerce application which can be like myntra, snapdeal, shopclues, amazon or any other website or product that comes to your mind when you hear about ecommerce.
2. The end goal of this project is directly related to the placements calls which mentees would receive from partners of pesto.
3. We should aim for building a full stack E Com application in which the user can select a product, add it to his cart and then finally using available payment modes place orders through it.

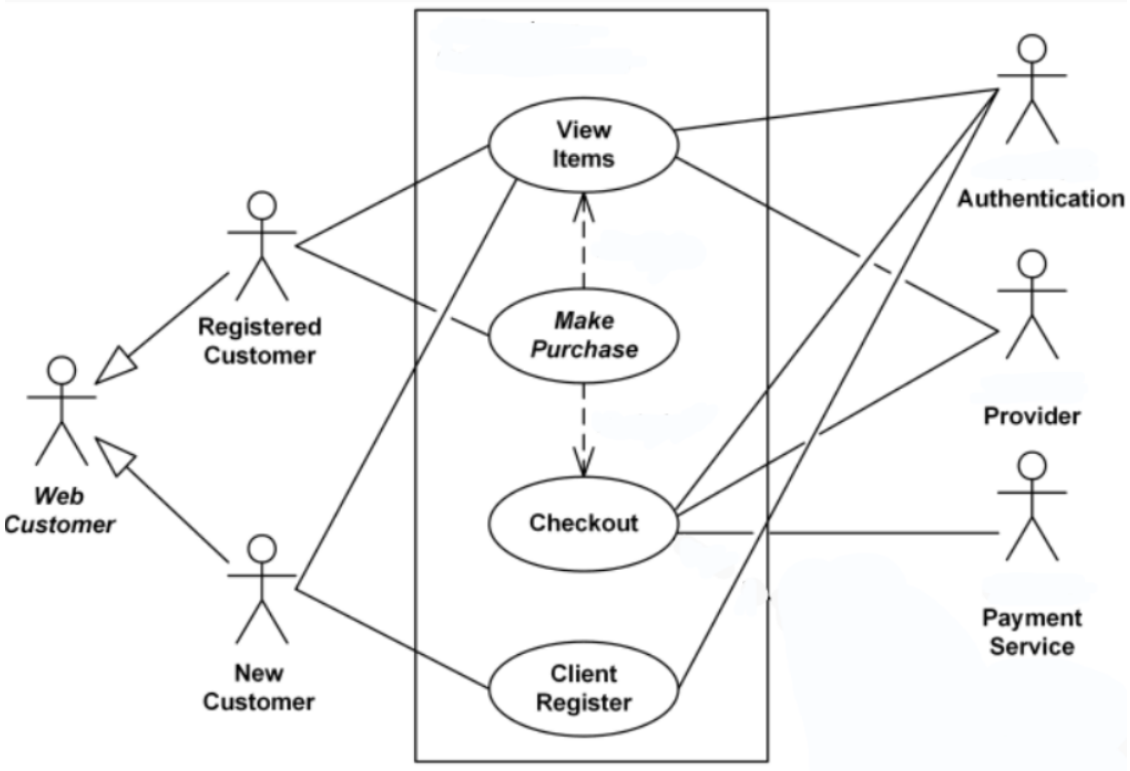
Some Design Considerations?

The system would be read-heavy, so we will focus on building a system that can retrieve products quickly.

1. Practically, users can place as many orders as they like; therefore, efficient management of transactions should be a crucial factor in designing this system.
2. Low latency is expected while viewing products.
3. Data should be 100% reliable. If an admin uploads a product, the system will guarantee that it will never be lost.

High Level System Design

At a high-level, we need to support one scenario, one to upload products and the other to view/search products. Our service would need some object storage servers to store photos and some database servers to store metadata information about the product.

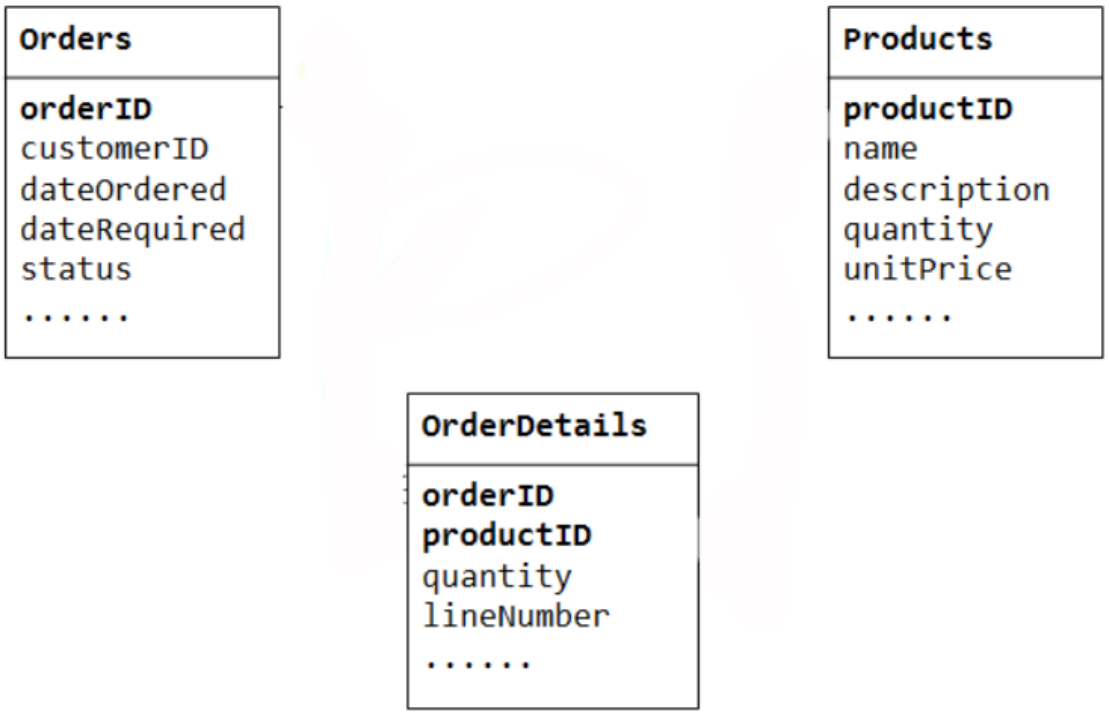
- 

Database Schema

1. We need to store data about the product, their uploaded photos, and metadata about the product. The Photo table will store all data related to a product; we need to have an index on (PhotoID)

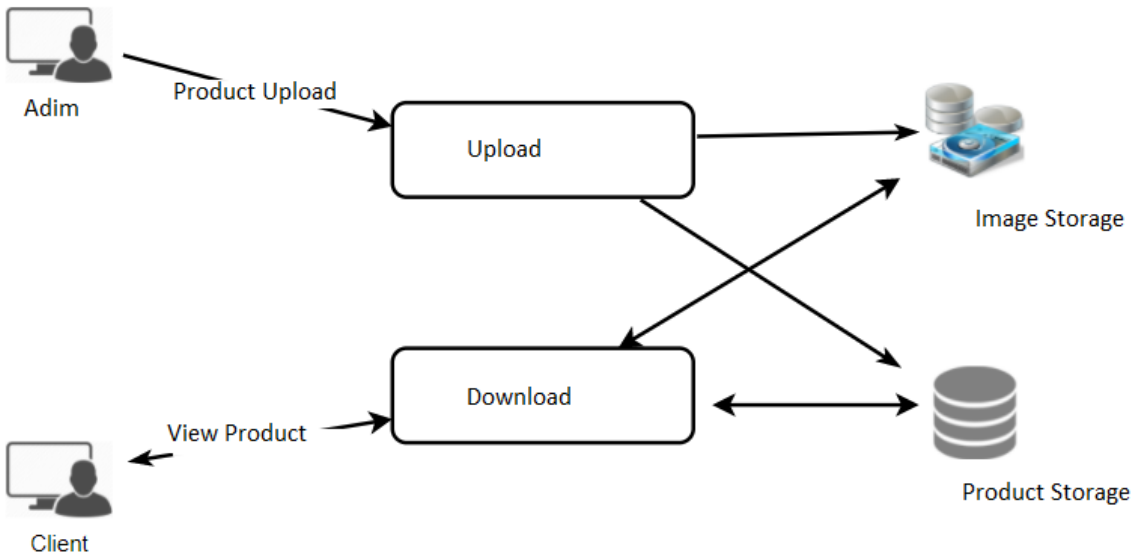
   We can store the schema in a distributed key-value store to enjoy the benefits offered by NoSQL. All the metadata related to product can go to a table where the 'key' would be the 'PhotoID' and the 'value' would be an object containing title, subtitle, price, mrp_price, quantity, etc

2. We need to store data about the user in one user table to store user details, like address details, number.
3. We need to store data about orders in a table and create a relation between order and user with a uuid.



High Level System Design

Product / photo uploads (or writes) can be slow as they have to go to the disk, whereas reads will be faster, especially if they are being served from cache.



Uploading users can consume all the available connections, as uploading is a slow process. This means that 'reads' cannot be served if the system gets busy with all the 'write' requests. We should keep in mind that web servers have a connection limit before designing our system. If we assume that a web server can have a maximum of 500 connections at any time, then it can't have more than 500 concurrent uploads or reads. To handle this bottleneck, we can split reads and writes into separate services. We will have dedicated servers for reads and different servers for writes to ensure that uploads don't hog the system.

Ranking and Products Feed Generation

To create the product list for any given user, we need to fetch the latest, most popular, and relevant product of the people the user follows.

For simplicity, let's assume we need to fetch the top 100 products for a user's products Feed. Our application server will first get a list of products according to user input and then fetch metadata info of each user's latest 100 photos. In the final step, the server will submit all these photos to our ranking algorithm, which will determine the top 100 products (based on recency, likeness, etc.) and return them to the user. A possible problem with this approach would be higher latency as we have to query multiple tables and perform sorting/merging/ranking on the results. To improve the efficiency, we can pre-generate the Product Listing.

Cache

Our service would need a massive-scale product delivery system to serve globally distributed users. Our service should push its content closer to the user using a large number of geographically distributed product cache servers and use CDNs, we can do that with the help of Redis or using server memory.

Thank You