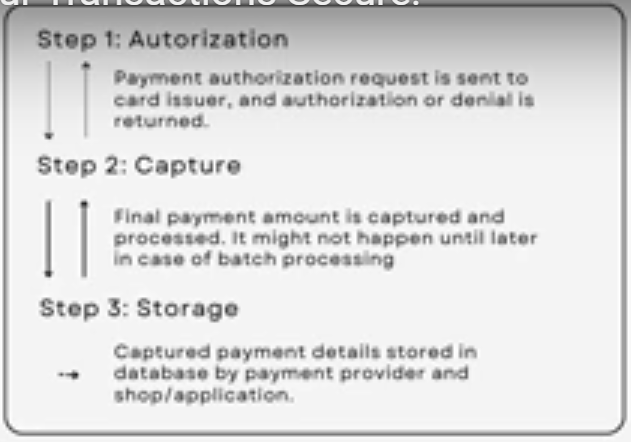
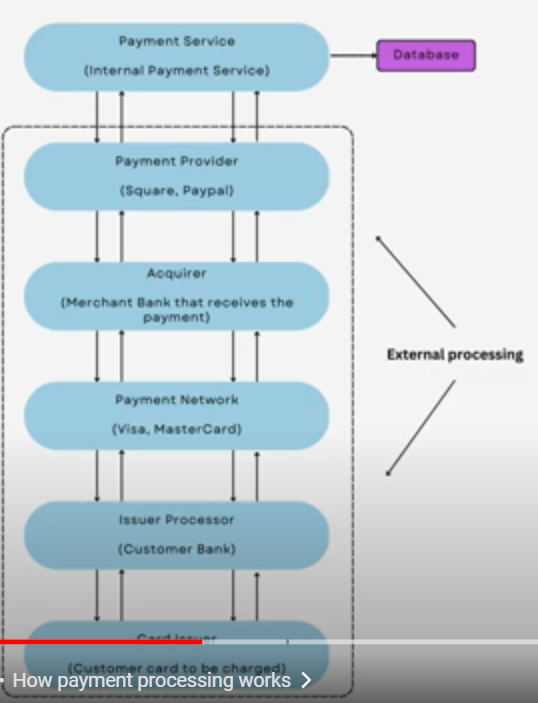
**Stages of Payment System :**

****

****

**Functional requirement**

* Consistency (ACID)
* User - Digital good
* Archiving Payments(GDPR , PCI)
* Notification of payment (Invoice)

**Non-functional requirement**

* Consistency is important
* Minimal latency
* High scalability
* Availability

**Estimates :**

Daly Active users: 10Millinon (may be they buy on an average 2 books per day)

Request per Second would be 10Million \* 2 / 86400 = 232

For the storage capacity

For payment details lets say : 200 bytes for each transaction (20M\*200 = 4Billion bytes = 4GB per day )

4GB \* 365 = 1,460 ( 200TB approximate)per year for the transaction data

For user details : 250 byts for each users ( 10 M \* 250 = 250 12Billion = 10GB a day + 1TB ) = 100 TB per year

In total it becomes around 300 TB data a year

**Storage for ingress :**

Ingress : 200GB per day so it should be 200GB / (3600\*24) = 4MB per second

**API :**

1) payment

POST: /api/v1/payment

Parameters : paymentId : UUID

method : ENUM (Debit card, Credit card, Online Payment)

amount : float

currency : ENUM (USD, INR, CAD, ERU)

userId : UUID

bookId: UUID

1. getAllPayments

GET : /api/v1/payments

Parameter : userID : UUID

Return : payment[] (array)

**Datamodel**

* Id : UUID
* user\_id : UUID
* Status : enum(pending, success, faild)
* Method : ENUM (Debit card, Credit card, Online Payment)
* Currency : enum (USD, INR, CAD, ERU) ISO\_4217
* Amount : float
* Trsaction\_Id : UUID (from Payment Service Provider)
* Created\_at : timestamp

