4YourEars

Background

• 4YourEars is a chain that sells music through various retail partners. They realized that their

spending on commission is high, and so they want to directly sell music to customers through an

online channel.

• In addition, they also realize that there is potential to cross-sell other items like antique LPs,

audio books, etc., to their customers.

• Their intent is hence to build an ecommerce site where they can directly sell to their retail

customers.

Requirements Overview

• The initial target is to have a simple commerce site that can sell music CDs and audio streams to

retail customers.

– CDs are physical merchandize, whereas audio streams are digital and are downloaded to

customers’ computers. The store will hence need to support both physical and digital

merchandise.

– For the audio streaming, 4YourEars has already tied up with a technology partner to be

able to do copyright protection, etc. This partner’s system will expose an API that will

return a media stream. The partner’s system will be co-deployed with the rest of the

solution, but capacity needs to be planned for.

– Customers will access the site initially from India, but will later expand into Europe and

US

• 4YourEars has a tie-up with two types of suppliers:

– Medium-size suppliers will access the site to be able to view orders that they need to

fulfill

– Large suppliers require 4YourEars to provide a data dump of orders that they need to

fulfill

– Large suppliers manage their own inventory, so for some items, inventory and shipping

information will come from these companies directly

• The company will also look to having a call center that will be able to service customer queries,

etc.

want to be able to target them as well.

Key Quantitative Information

• 4YourEars expects that they will be able to have a target audience of 1,000,000 visitors daily,

and that they will grow at 30% year on year.

• Conversion rate is expected to be 5% initially, but the store is looking to build a strong targeting

based on behavior analytics to increase this to 10%.

• Once they expand into other geographies, volumes are expected to triple.

• Visitors can access the site any time of the day

Concerns and Thoughts

• 4YourEars knows that they are a late entrant into this market, so they need to be able to

differentiate themselves quickly.

• They however, have strong brand recall, and the management feels that they can use this to

advantage

• Some other sites have mooted the idea that they could cross-sell 4YourEars’ products through

their site, and so they have asked 4YourEars if they can supply their catalog information to

them. While 4YourEars feels this increases presence, they are thinking they only ought to share

targeted information instead of a full catalog dump.

Your Deliverables

• You have been asked to give a solution to help 4YourEars reach their objectives

- 4YourEars needs to self-copyright advertise via online channels and social media.

- Surveys and customer feedback and analyse and work on the problem getting customers via online channel

- Set a small target audience and try to increase on daily basis.

• Use DDD Approach to come up with Logical Design explaining each domain

* Use of ServiceMesh,Sidecar, CQRS ,Saga, Event Souring and Circuit breaker patterns

**ServiceMesh:**

* Apply security policies to all traffic at ingress, and encrypt traffic using mTLS traveling between services.
* Add Zero-Trust networking.
* Govern certificate management for authenticating identity.
* Enforce level of least privilege with role-based access control (RBAC)

**Sidecar:**

* Sidecars handle interservice communications, monitoring, and security‑related concerns – indeed, anything that can be abstracted away from the individual services.

**CQRS:**

* CQRS may simplify understanding of domain by dividing problem into the command and query parts. In situations, when your UI is based on workflows and utilizes the Interface pattern. It is easier to identify user's intents and translate them into domain events.

**Saga:**

* Each service in a saga performs its own transaction and publishes an event. The other services listen to that event and perform the next local transaction. If one transaction fails for some reason, the saga also executes compensating transactions to undo the impact of the preceding transactions

**Event Souring:**

* One obvious form of return is that it's easy to serialize the events to make an Audit Log. ...
* Another use for this kind of complete Audit Log is to help with debugging. ...
* Event Sourcing is the foundation for Parallel Models or Retroactive Events.

**Circuit breaker patterns:**

* The Circuit Breaker pattern prevents an application from performing an operation that is likely to fail. An application can combine these two patterns by using the Retry pattern to invoke an operation through a circuit breaker
* Ensure Performance, Reliability and Scalability using right tools and frameworks
* Response Time: Response Time is the time between the user request and the application response.
* Throughput: Throughput is the measure of a number of requests processed over a unit time by the application
* Network usage
* Provide API First Approach
* Follow SOLID principle
* **S** (Single Responsibility Principle)- Each class should be responsible for a single part or functionality of the system.
* **O** (Open-Closed Principle) - Software components should be open for extension, but not for modification.
* **L** (Liskov Substitution Principle) - Objects of a superclass should be replaceable with objects of its subclasses without breaking the system.
* **I** (Interface Segregation Principle) - No client should be forced to depend on methods that it does not use.
* **D** (Dependency Inversion Principle) - High-level modules should not depend on low-level modules, both should depend on abstractions.
* Suggestion on TDD and End to End Testing approach
* Use TDD for each API
* Cover each service before deliver
* Sure 100% success result
* Agile model, Also provide MVP approach
* Follow CI/CD model
* Plan a monthly task
* Assign task on weekly basis
* Take daily progress update
* Update task status daily

• You are supposed to work on the following:

– Solution Approach

* Solution selling refers to the philosophy or practice of uncovering a customer's pain points and then providing products and services that address the underlying business problem.

– Logical Architecture

* Minimize the complexity and time performance.

– Technical Architecture

* Latest version of technology
* Follow latest design patters

– High level risks that you see in such a program

* Security checks in every points
* Monitoring and raise alarm

– Any other architecture views that you perceive as being relevant

* Always follow up updated versions
* Follow to develop Rest api Richardson maturity model .