1. **CATfish: Requirement from the PM**

Build next capabilities to pizza online shop:

    1. Customer can order a pizza in the online shop;

    2. Customer can select additional ingredients/toppings in the order;

    3. Customer can select different online & offline payment methods for each order;

    4. Delivery address can be specified with every order;

    5. Customer is allowed to cancel the order if order is not yet sent for delivery;

    6. Customer can verify the status/details of the order.

ReqPojo – dto – response -- adapters

Implementation hints:

   - Provide REST API for the online shop selling pizza;

   - Assume that customer information is already in the system, payment card is already registered in his profile;

   - No need to care about Authentication and Authorization of REST API;

   - No need to care about API documentation / Swagger UI;

   - Light-weight in-memory implementation of persistence layer is sufficient;

   - Implement solution as Spring Boot project in Java / Kotlin, whatever is more comfortable for you;

   - Use any build system you want (Gradle, Maven, or not at all);

   - Cover with tests as you think is necessary for a normal production code, considering different type tests (unit/integration);

   - Push results of your working code + short description explaining your implementation to a git repository;

   - Commit changes to the local git repository with every step (commit granularity is up to you);

   - Describe any compromise you make to implement business requirements in the README.MD file.

   - Implementation must not be 100% complete.

1. **Development road map for the project pizza online shop**

* Scope
* Not in the scope
* prerequisites
* Design: HLD, LLD, DB estimates and design, Performance characterisation
* Architecture diagram
* Design APIs
* TDD BDD
* Tech stack
* Implementation, logging and metrics
* Cache, load balancing, replication
* Testing: Junit, functional testing, Integration testing
* deployment
* monitoring and alerting
  1. **Scope**

1. Customer can order a pizza in the online shop;
2. Customer can select additional ingredients/toppings in the order;
3. Customer can select different online & offline payment methods for each order;
4. Delivery address can be specified with every order;
5. Customer is allowed to cancel the order if order is not yet sent for delivery;
6. Customer can verify the status/details of the order.
   1. **Not in scope**
7. Authentication and Authorization of REST API
8. API documentation / Swagger UI
   1. **Prerequisites**
9. customer information is already in the system
10. payment card is already registered in his profile.
    1. **Design**

Components

* **Order Service**
* User Service
* Payment Service
* **Notification Service**

DB Design

Orders: order\_id, customer\_id, total\_price, date, delivery\_time, status, employee\_id,vehicle\_id

Employees: id, name, salary, shift, phone

Vehicles: id, reg\_no, type

Items: item\_id, price, added\_on, discount

Order\_item: order\_id, item\_id, quantity, per, phone number

customer: customer\_id, name, Address, card, phone number, email

Address: user\_id, addressId, zip\_code, phone, city, country, address

DeliverySattus: PENDING, DISPATCHED, DELIVERED, FAILED

Payment: id, type, price

User\_payment: option, card

Delivery\_Mode: urgent, normal

**ER Diagram**

Diagram

Description automatically generated

DB Estimates: Capacity estimates and constraints

* Bandwidth
* Storage
* Cache

TODO : choose the DB

Performance characterisation:

* 1. Architecture

Diagram

Description automatically generated

* 1. Design APIs

POST pizza/v1/order HTTP1.1

Host catfish.com

Authorization: Basic yamuna:password

content-type: application/json  
Accept: application/json  
{

“customerid”: 123,

“totalprice”: 690,

"items": [

{

“id”: 3

“name”: “thinscrust”

"quantity": 1,

“unitprice”: 250

“toppings”: [

{

“id”: 1

“name”: “cheese”

},

{

“id”: 1

“name”: “onion”

},

]

},

{

“id”: 2

“name”: “panpizza”

"quantity": 1,

“unitPrice”: 200

},

{

“id”: 5,

“name”: lemonfresh,

"quantity": 2,

“unitPrice”: 120,

“toppings”: [

{

“id”: 1

“name”: “sugar”

}

]

}

]

}

HTTP/1.1 201 Created

Date: Mon, 20 Jun 2011 21:15:00 GMT

Content-Type: application/hal+json

Location: