Getir Case Study Documentation

1. Technologies
    a. Java 17
    b. Maven
    c. Spring-Boot
    d. MongoDB Atlas
    e. Docker
2. How to Start Project

```
$ docker-compose -f docker-compose-mongo.yml up -d
$ mvn clean package -DskipTests
$ docker-compose build
$ docker-compose up -d
```

    And create customer -> http://localhost:8080/api/customer/signup

3. Credentials
    a. Spring Security JWT Auhtentication

4. Design

    I used Domain Driven Design for packaging

    a. Controllers
        i. They receive request from user and send it to facade layer.
        ii. Getting result from facade layer and creates ResponseEntity based on that result.
        iii. If any exception happens, Controller advisor will catch the exceptions and throw ErrorResponse, which contains errorMessage, HttpStatus code and requested url.
    b. Facade
        i. Responsible with Dto to Entity, Entity to Dto conversions and simple validations.
        ii. EntityServices only used by EntityFacades.
        iii. Facades can call other Facades methods.
        iv. Main purpose of Facade layers is simplifying Service layers and make service layers only responsible with repository operations.
    c. Service
        i. Responsible with repository CRUD operations.
        ii. EntityRepositories only used by EntityServices.
        iii. Can't call other Service or other Repository methods.

Expandable features

- Spring security config can be used for configuration.
- Elastic search can be used for search services.
- Microservice architecture can be used.
- Kafka can be used for microservices to communicate with each other.