MicroServices:

- Add dependency first .
- Create RestTemplateConfig class, MicroService 1 will have a DTO of MicroService 2 and vice versa
- For Eureka Server add: Eureka Server dependency (take dependency management and version tags if necessary). Annotate main class with @EnableEurekaServer
- For Eureka Client add: Eureka Discovery Client and Cloud Bootstrap. Remember to add dependency management and version tags, annotate all microservices with @EnableDiscoveryClient
- Change Spring application name based on services used e.g., POST-SERVICE & COMMENT-SERVICE. Update service layers wherever calls are made, remove hardcoded port values, and make use of the application name
- Run Eureka Server first followed by each of the microservices.
- For API Gateways add the following dependency: Cloud Bootstrap, Lombok, Spring Reactive Web, Gateway, and Eureka Client. Again remember to add @EnableDiscoveryClient to make API gateway as a client to the discovery service.
- Add CircuitBreaker, Spring AOP, and Spring Actuator dependencies in each microservice.
- Update application.yml file to get microservices health metrics, endpoint health metrics, and add instance name in Circuit Breaker.
- Handler methods responsible for interacting with other microservices must be updated for fault tolerance.
- Add @CircuitBreaker annotation and create fallback method with the same method argument and same return type.

MICROSERVICES application.yml config

```
datasource:
  url: jdbc:mysql://localhost:3306/commentms
  username: root
  password: test
  driver-class-name: com.mysql.cj.jdbc.Driver

jpa:
  database-platform: org.hibernate.dialect.MySQLDialect
  hibernate:
    ddl-auto: update
```

```
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        use_sql_comments: true

# Other application configurations
server:
  port: 8082

# Eureka Client configurations
eureka:
  instance:
    prefer-ip-address: true
  client:
    register-with-eureka: true
    fetch-registry: true
    serviceUrl:
      defaultZone: http://localhost:8761/eureka

# Actuator Properties
management:
  health:
    circuitBreakers:
      enabled: true
  endpoints:
    web:
      exposure:
        include: health

  endpoint:
```

```
  health:
    show-details: always


# Circuit Breaker Properties
resilience4j:
 circuitbreaker:
  instances:
    postBreaker:
      registerHealthIndicator: true
      eventConsumerBufferSize: 10
      failureRateThreshold: 50
      minimumNumberOfCalls: 5
      automaticTransitionFromOpenToHalfOpenEnabled: true
      waitDurationInOpenState: 6s
      permittedNumberOfCallsInHalfOpenState: 3
      slidingWindowSize: 10
      slidingWindowType: COUNT_BASED
```

Application.yml file for API GATEWAYS

```
#  API GATEWAY
================================================================
==Server settings
server:
  port: 8083


# Eureka client settings
eureka:
  client:
    fetch-registry: true
```

```
    register-with-eureka: true
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true


# Spring application settings
spring:
  application:
    name: API-GATEWAY
 # Spring Cloud Gateway settings
  cloud:
   gateway:
    routes:
     - id: post-service-route
       uri: lb://POST-SERIVCE # Load balanced URI of the Post service
       predicates:
        - Path=api/posts/**
     - id: comment-service-route
       uri: lb://COMMENT-SERVICE # Load balanced URI of the Comment service
       predicates:
        - Path=api/comments/**
```

Example of a fallback method:

Controller

```
@GetMapping("/{postId}/comments")
@CircuitBreaker(name="commentBreaker", fallbackMethod = "commentFallback")
public ResponseEntity<PostDtoMS> getPostWithListOfComments(@PathVariable
String postId){
   PostDtoMS dtoms= postService.getPostWithListOfComments(postId);
    return new ResponseEntity<>(dtoms,HttpStatus.OK);
}

public ResponseEntity<PostDto> commentFallback(String postId,Exception
exception){
    System.out.println("Fallback is executed because service is down :
"+exception.getMessage());
    exception.printStackTrace();

    PostDto dto=new PostDto();
    dto.setId("1234");
    dto.setTitle("Service Down");
    dto.setContent("Service Down");
    dto.setDescription("service Down");
    return  new ResponseEntity<>(dto,HttpStatus.BAD_REQUEST);
}
```

IMPL:

```
@Override
public PostDtoMS getPostWithListOfComments(String postId) {
    Post post = postRepository.findById(postId).get();

    ArrayList comments =
restTemplate.getRestTemplate().getForObject("http://COMMENT-
SERVICE/api/comments/" + postId, ArrayList.class);

    PostDtoMS postDtoMS=new PostDtoMS();
    postDtoMS.setPostId(post.getId());
    postDtoMS.setTitle(post.getTitle());
    postDtoMS.setDescription(post.getDescription());
    postDtoMS.setComments(comments);

    return postDtoMS;
}
```

PROJECT DESC: