**Tools and Technologies used**

1. **Eureka Server**

This is used for service discovery. Microservices will get registered with eureka server in order to be available to each other.

Configuration references:

<https://dzone.com/articles/spring-cloud-amp-spring-bootimplementing-eureka-se>

<https://www.baeldung.com/spring-cloud-netflix-eureka>

<https://spring.io/guides/gs/service-registration-and-discovery/>

<https://cloud.spring.io/spring-cloud-netflix/multi/multi__service_discovery_eureka_clients.html>

1. **Feign Client**

This is used for inter-communication of services.

Configuration references:

<https://www.devglan.com/spring-cloud/spring-cloud-feign-example>

<https://dzone.com/articles/microservices-communication-feign-as-rest-client>

<https://cloud.spring.io/spring-cloud-openfeign/reference/html/>

1. **Hystrix**

This is used for circuit breaker i.e. if a service is unavailable then the fallback approach will be taken.

Configuration references:

<https://www.baeldung.com/spring-cloud-netflix-hystrix>

<https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-feign.html>

<https://howtodoinjava.com/spring-cloud/microservices-monitoring/>

1. **Ribbon**

This is used for service load balancing. E.g. if we have multiple instance registered of a service on eureka then using ribbon we can define the rule to distribute the request among them.

Configuration references:

<https://howtodoinjava.com/spring-cloud/spring-boot-ribbon-eureka/>

<https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-ribbon.html>

1. **Zipkin**

This is used to track the request flow. It is using elastic search as a storage.

Configuration references:

<https://dzone.com/articles/spring-cloud-amp-spring-bootimplementing-zipkin-se>

<https://spring.io/projects/spring-cloud-sleuth>

1. **ELK (Elasticsearch, Logstash, Kibana)**

This is used to centralize the logging of microservices.

Configuration references:

<https://dzone.com/articles/monitoring-microservices-with-spring-cloud-sleuth>

1. **Keycloak**

This is used as a OAuth server for SSO.

Configuration references:

<https://medium.com/@bcarunmail/securing-rest-api-using-keycloak-and-spring-oauth2-6ddf3a1efcc2>

1. **Spring Actuator**

This is used to get the microservice info, healthcheck etc.

1. **Zuul**

This is used to create a centralized gateway service and route the traffic to desired microservice.

Configuration references:

<https://cloud.spring.io/spring-cloud-netflix/multi/multi__router_and_filter_zuul.html>

<https://howtodoinjava.com/spring-cloud/spring-cloud-api-gateway-zuul/>

<https://www.baeldung.com/spring-rest-with-zuul-proxy>

1. **spring-cloud-zuul-ratelimit**

This is used to apply throttling concept i.e. limiting the number of request serve for a given time.

Configuration references:

<https://www.baeldung.com/spring-cloud-zuul-rate-limit>

<https://github.com/marcosbarbero/spring-cloud-zuul-ratelimit>

1. **Swagger**

This is used to create the documentation of the different API available in microservices.

Configuration references:

<https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

<https://springfox.github.io/springfox/docs/current/>

1. **Spring configuration server**

This is used to centralizing and profiling the configuration of different microservices.

Configuration references:

<https://www.baeldung.com/spring-cloud-configuration>

<https://cloud.spring.io/spring-cloud-config/reference/html/>

1. **Docker**

This is used to containerize the microservices and different component used.

Configuration references:

<https://docs.docker.com/compose/compose-file/>

1. **Kubernetes**

This is used to container-orchestration for microservices and different component used to scaling and manage the deployment.

Configuration references:

<https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>