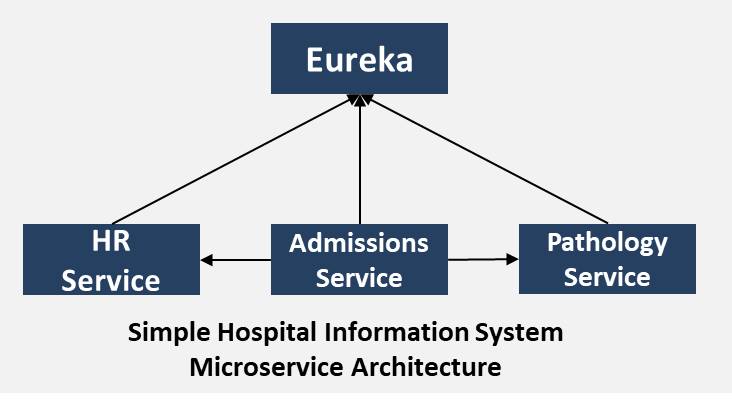
**Code Base Test**

1. Create microservice based on a simple hypothetical Hospital Information System with Eureka and apply following constrains



1. **Create 3 Spring Applications in start.spring.io (add the Web dependency)**
   * Admissions (list of patients)
   * HR (list of employees)
   * Pathology (list of diseases)
2. **Build the admissions-service**
   * Create the AdmissionsResource class(controller) in a resources package
   * Annotate the class with @RestController
   * Create a method getPatients() that return list of patients
   * Create a method getPatientById that takes an Id and returns a single Patient
   * Annotate the Id parameter with @PathVariable annotation
   * Create a class called Patient in the models package
   * Add some hardcoded patients to the getPatients method
   * Annotate the AdmissionsResource with RequestMapping of /admissions
   * Annotate the getPatients method with the RequestMapping of /patients
   * Annotate the getPatientById method with RequestMapping of /patients/{id}
   * Move the hardcoded patients outside the method
   * Use Thymeleaf where ever required
   * Test the addmissions-service
3. **Build the hr-service**
   * Create a HrResource class in the resources package
   * Annotate the class with the @RestController
   * Annotate the class with @RequestMapping of /hr
   * Add some hardcoded list of patients
   * Create the Employee class in the models folder
   * Create the getEmployees method to return list of employees
   * Annotate the getEmployees method with RequestMapping of /employees
   * Create the getEmployeeById method that take an Id to return a single employee
   * Annotate the Id parameter with @PathVariable annotation
   * Annotate the getEmployeeById method with @RequestMapping of /employees/{id}
   * Use Thymeleaf where ever required
   * Test the hr-service(remember the port!)
4. **Build the pathology service**
   * Create the PathologyResource class in the resources model
   * Annotate this class with the @RestController annotation
   * Annotate this class with the @RequestMapping annotation of /pathology
   * Create a hardcoded list of diseases
   * Create the Disease class in the models package
   * Create the getDiseases method to return list of diseases
   * Annotate the getDiseases method with RequestMapping of /diseases
   * Create the getDiseaseById method to return a single disease
   * Annotate the getDiseaseById method with RequestMapping of /diseases/{Id}
   * Annotate the Id parameter with the @PathVariable annotation
   * Use Thymeleaf where ever required
   * Test the pathology-service
5. **Call the hr-service from the admissions-service using RestTemplate**
6. **Call the pathology-service from the admissions-services using RestTemplate**
7. **Create the Discovery Server (Eureka) and make the service Eureka Client**
   * Create a spring boot application using spring initializr (add the Eureka Server dependency)
   * Add the @EnableEurekaServer annotation to the DiscoveryServerApplication
   * Ensure that the spring cloud dependency is in the pom.xml
   * Update the application.properties file with register-with-eureka and fetch-registry to false
   * Test the Eureka Server
8. **Pathology service should include** **DiseaseNotFoundException** exception class should extend the **RuntimeException**.
9. **Add annotation to existing REST APIs and Test REST APIs through Swagger URL**
10. **Testing different endpoints provided by Actuator**
11. **Configuration Circuit Breaker - Resilience4J to your service**
12. **For code quality - Integrate Sonar Lint with each service**

2. In AWS do following specifications

1. Launching an AWS EC2 instance for windows
2. Create S3