# Component-and-Connector View

### Service-Oriented Architecture (SOA) View

## Section 1: The Primary Presentation

# 

## Section 2: The Element Catalog

### 2.1 Client layer

A client layer is access from mobile or browser. Only authenticated user can access micro-service. Client sends a token in a header form to a Gateway. It can be access from tablet, phone or browser. Client layer handles a return response from access layer for authenticated user or not-authenticated users.

### 2.2 Access layer

Access layer is combination of Gateway and authentication service. Gateway handles all request from a client and communicate with authentication service. If Gateway get a request without token it will return not-authenticated user response. When a client sends a request with a token, Gateway verify a request with authentication service and forward to protected layer to access each microservice. Protected layer returns microservice response.

### 2.3 Protected layer

The elements within the protected layer can be accessed only via the api gateway. The requests that originated from the UI will firstly send to Access layer where the API gateway locates, then these requests would be routed by the API gateway to the corresponding services based on the entries in the registry. After processing, the responses from microservices in the protected layer would be transmitted back to the api gateway. In this sense, protected layer is hidden behind the access layer. If any elements need offline for any reasons, the function of that element would be automatically decommissioned in the registry and subsequently api gateway. So without restarting the whole system, we can add/remove elements in the protected layer. This is the flexibility that is provided by the design.

### 2.4 Service registry & discovery

Service registry and discovery come from eureka organization and are implemented by Java Spring. Discovery is responsible to receive heartbeats from registries in order to maintain an list of alive microservices. At the same time, in terms of making sure synchronization, registry will check the status of microservices positively. Finally, API Gateway can access to all microservices registered in the discovery server and return retrieved data to frontend.

## Section 3: Context Diagram

## 

## Section 4: Variability Guide

If a gateway configurations changes, client connection will break-down. All the services end-points are defined in a Gateway. So, if any microservice api-endpoints are updated it is necessary to update in Gateway configuration file. Moreover, if database connection as login credentials, url or database name updates, it should be update in each microservice. All microservice has logging system to check logs of each service.

## Section 5: Rationale

This identify components of a system. We used various framework and language as Java Spring, Node.js and React. Gateway is implemented using node.js, Client is in React.js and other microservices are in Java Spring. Moreover, microservice, in terms of a kind of SOA architecture, need a registry to manage all the microservices. Eureka has set the connections between the server and clients by allowing the client to send a heartbeat to the server and requiring the server to check the status of clients lifecycle in an exact time interval. All components are deployed on Bluemix and able to accessed by users anytime and any platform with responsive layout of a website.

## 