# Summary of Documentation Beyond Views

## Document Control Information

Issuing Organization: Team 01 - CS9549

Version Number: 2.0

Date of Issuing: 04/04/2019

Status: Under development

Change History:

* 04/04/2019: Updated with new diagrams for existing services
* 13/03/2019: Created

To submit a change request, please email the request to mdudycz@uwo.ca.

## Section 1: Documentation Roadmap

### Scope and summary

This document is intended to provide insight into the architectural design and reasoning of the PurplePages project. It outlines the details regarding the structure and pattern implementation of each microservice, along with the delivery method, and the intended user interactions.

Within this document are three view templates, corresponding to the Module view, Component-and-Connector view, and Allocation view. There are also two quality views and an outline of the service-oriented architecture pattern.

### Organization

Section 2: This section outlines the documentation standards employed throughout this document.

Section 3: This section outlines the system’s intended functions and interactions with users. It also contains an outline of the service-oriented architecture view.

Section 4: This section provides an understanding of the various interactions between the views outlined in this document.

Section 5: This section elaborates on the design decisions of the development team, outlined within the document.

Section 6: This section offers reference material to aid in understanding the document.

### View Overview

The module view demonstrates all the layers involved in a microservice system. It also includes the interrelationship between each layer. It serves as a guideline for the organization of modules and their responsibilities. Additional module view are included to apply to each microservice.

The component-and-connector view is an overall view of a system with the component connection. It’s a relationship between all components and with each other. It consists of client, gateway and microservices.

The allocation view has utilized the deployment view of a system.It gives a brief view of how to deploy three parts of the whole project on bluemix and how to deploy them with specific parameters. All the microservices and database are included to make a clear identification on the whole project.

The first type of quality view is a Sequence Diagram. It depicts a sequence diagram for the submission and reception of a request to the microservices. It is modelled as a RESTful API to facilitate the client-server structure. The elements featured are the user and the respective components that decode the request to access information from the database. The diagram is modelled over instances and time, reflecting the interactions between elements as the sequence progresses.

The second type of quality view is a Use Case Diagram. The diagram features the specific use cases, and their interrelations in the process to accomplish a goal by an actor.

### Documentation Relevance

|  |  |  |
| --- | --- | --- |
| View | Stakeholders and Concerns | Example |
| Module View | * Users * Acquirer * Developers * Maintainers | * A user wishes to retrieve information about a database item * An acquirer wishes to integrate their own modules in data retrieval * A developer wants to integrate a new service using the existing module structure * A maintainer wishes to know which modules will be affected by a change |
| Component-and-Connector view | * Users * Acquirer * Developers * Maintainers | * A user wishes to understand the dataflow within the system * An acquirer wishes re-direct the dataflow to their own applications * A developer wishes to integrate new validation information into the dataflow * A maintainer wishes to identify the source of a resulting error |
| Allocation View | * Users * Acquirer * Developers * Maintainers | * A user wishes to know which environment the front ends can be executed in * An acquirer wishes to know if services can be transferred to other environments * A developer wishes to add a new service to an existing environment * A maintainer must migrate a service to an new environment |
| Quality View: Sequence Diagram | * Users * Acquirer * Developers * Maintainers | * A user wishes to retrieve information about a database item * An acquirer wishes to communicate a database detail to their clients * A developer wishes to add additional search functions * A maintainer wishes to update the format in which resort information is sent |
| Quality View: Use Case Diagram | * Users * Acquirer * Developers * Maintainers | * A user wishes to know the processes to accomplish a goal * An acquirer wishes to integrate their users into the system * A developer wishes to add additional information to a process * A maintainer wishes to manually check a system action |

## Section 2: How a View is Documented

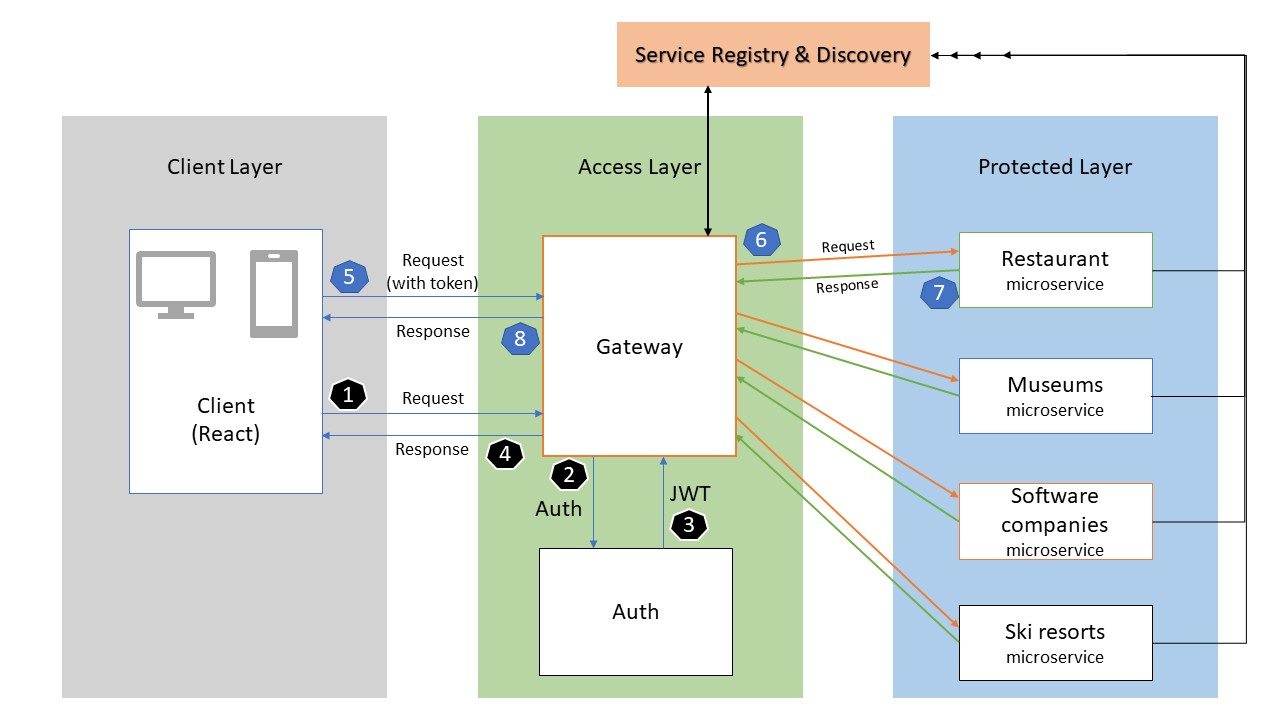
The views contained in this document match that of the textbook “Software Architecture in Practice: Third Edition”, reflecting the documentation philosophy of ISO/IEC/IEEE standard 42010:2011.

## Section 3: System Overview

The primary function of this system is to provide directory relevant microservices, imitating that of a directory book such as the Yellow Pages. These services are accessible to multiple independent users seeking information from a web portal. The microservice structure allows for on-the-fly editing, adding, and removal of services by administrators. As the system stands, it offers directory information on fortune 500 companies, ski resorts, museums, and restaurants. Presently, the system relies on information sourced from pre-existing databases, which is hosted and managed through the IBM Cloud.

### SOA view:

Software-oriented architecture presents in layers. An end-user can access the client layer. An end-user can access a website from their browser or phone device. Client services send a valid token to a Gateway, and gateway verifies that token. A Gateway sends a request to an Auth service and middleware verifies authentication. If a request is authenticated only then can a user access the protected layer. The protected layer consists of all microservices offered by the system. Each microservice is connected with a database. Service Registry and discovery identifies live/dead microservices.



## Section 4: Mapping Between Views

|  |  |  |
| --- | --- | --- |
| View1 | View2 | Correlation |
| Module View | C&C View | C&C View shows the architectural details of the system, while the Module Views unveils the the implementation level details. That is to say, the Module view can be used to flesh out the details of each component in the C&C View. In turn, C&C View connects each module with others to shed lights on how each module can fit in the bigger picture which is the overall system. |
| Module View | Quality View | Quality View talks about the actual implementation methods, strategies and procedures of Module View. |
| C&C View | Allocation View | Allocation View supports the C&C View when stakeholders planning to deploy the system on the hardware level. At the same time, C&C View requires Allocation View to meet exact conditions in order to run the system smoothly and user-friendly. |

## Section 5: Rationale

The primary focus of this system is to implement an architectural structure to facilitate the use of multiple microservices. As such, the views reflect a client-server structure, where the client can request information from the server, ranging from which services are available, to the information each service provides.

The core microservices reflecting the functionality of a YellowPages directory all utilize a RESTful implementation to retrieve information from independent databases. This design was specifically chosen to allow individual services to go down, for editing or deletion, without impacting the end user.

All microservices are tested from a postman environment. Each microservice returns a common response with code and message. A 200 code is for success, 500 for a server-side error, 401 for a client error. Moreover, the message contains success, error message, or JSON response of the microservices.

## Section 6: Directory -- index, glossary, acronym list

Please see the document-wide directory.