

**Computer Science Department**

**CCP6427: Cloud Engineering**

**Spring 2022**

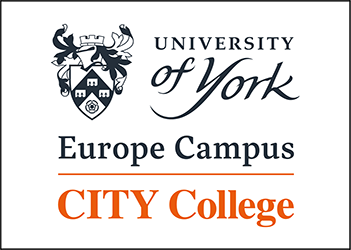
**Design and development of a microservices-based application**

**Submission Deadline: 22/5/2022**

**Actual Submission Date: 22/5/2022**

|  |
| --- |
| **Student Registration Number/s** |
| WMY21040 |

**Declaration**: *I have completed and submitted this work by myself without assistance from or communication with another person either external or fellow student. I understand that not working on my own will be considered grounds for unfair means and will result in fail mark for this work and might invoke disciplinary actions. It is at the instructor’s discretion to conduct an oral examination, if necessary, which will result in the award of the final grade for that particular piece of work.*



Report

**File Conversion Microservices with Eureka**

**Abstract (250 – 300 words)**

My abstract here.

**Keywords:** Microservices, Springboot, Eureka Netflix, Model View Controller (MVC), Representational State Transfer (REST), Service Discovery Protocol (SDP).

Table of Contents

[1. Introduction 1](#_Toc89161934)

[2. First Section 2](#_Toc89161935)

[2.1 First Sub-section 2](#_Toc89161936)

[3. Conclusion 3](#_Toc89161937)

# 

**Table of Figures**

[Figure 8‑1. A figure A-1](#_Toc89161938)

**Table of Tables**

[Table 1‑1. Hello 1](#_Toc89161939)

# Introduction

In this project the microservices software engineering architecture will be put in practice through a minimal system. The system in question will be responsible for receiving plain text data, converting it to a pdf file and sending it back to the client application. The goal is to create a realistic and flexible microservice application which addresses traditional issues from service-oriented design.

# System Specification

The system does not require extensive business logic, but several architectural constraints that require consideration. System requirements will be presented organised by type below.

**Architectural system requirements:**

* Microservices are designed independently of each other’s implementation, communicating strictly through HTTP requests.
* Use Model View Controller (MVC) software design pattern.
* Register microservices to Netflix Eureka service registry.

**Functional system requirements:**

* Users expect a pdf file to be generated containing user provided plain text.

# Design

Harbouring understanding of the system specification allows for the creation of the software design. An appropriate representation to demonstrate the system’s design is through the use of a software architecture diagram with service-to-service communication.

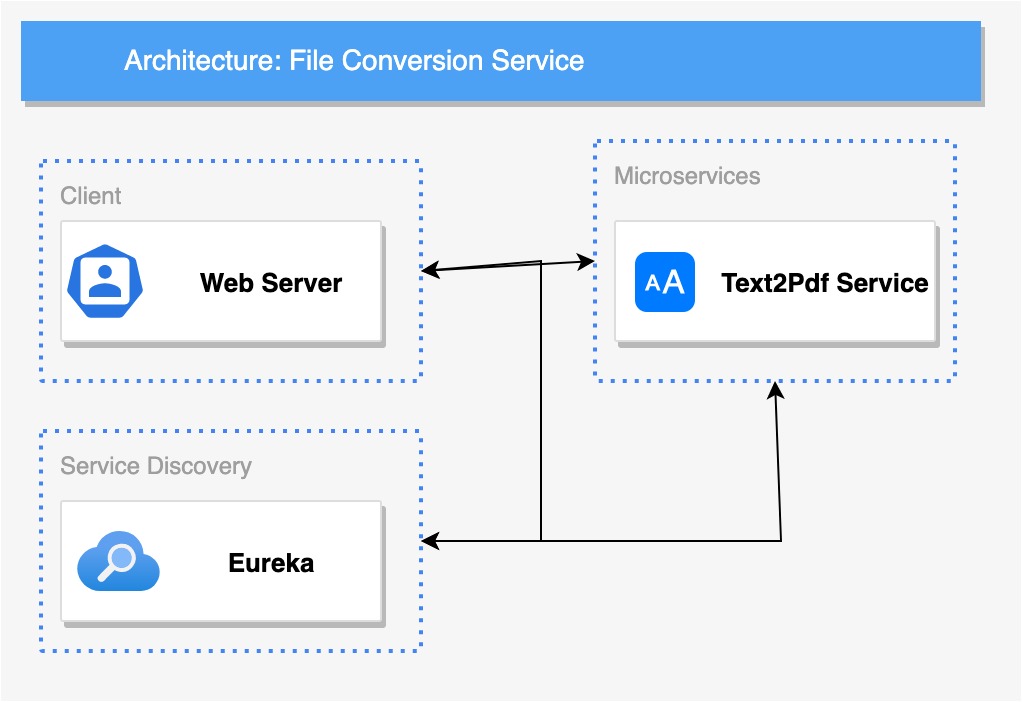


Figure 3.1. File Conversion App Architectural Diagram

Elaborating on the figure above, clients and microservices have to register to the Eureka service discovery. Eureka will map services using the service discovery protocol to allow for communication between registered systems through the lookup operation. Lookup only requires the provision of the service name from the requestor. Eureka also monitors the state of each registered service in order to react to system failures during runtime.

The web server is responsible for providing to users a UI through which they can initiate the conversion process. The user sends a request containing the text that needs to be transformed and the relevant microservice (as routed by Eureka) is expected to serve the request. The text to pdf conversion service then receives a request for a conversion and performs the operation. Independent of a successful or erroneous outcome, a request is sent back to the web server with the file or exceptions that led to operation failure. Lastly, the web server reacts to the microservice response by serving the file or informing the user of errors that may have occurred.

By addressing the communication between the services and their purpose, an effort will be placed in explaining each service’s design individually. This will be performed with a diagram depicting each service’s package. Eureka will be excluded from the process due to its minimal setup and package footprint.

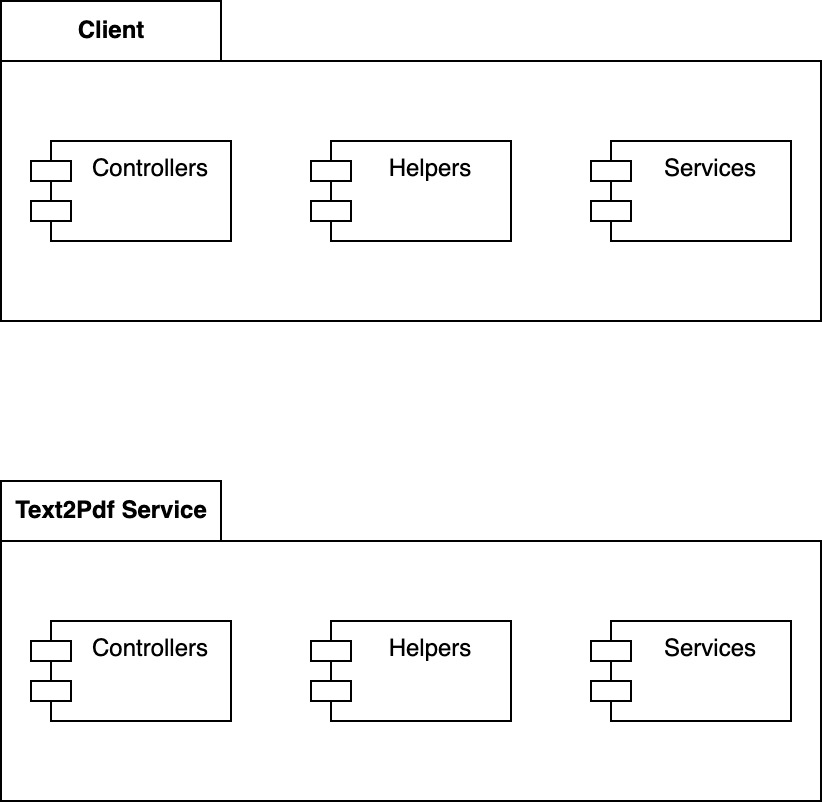


Figure 3.2. Software Package Diagram

There are three common elements amongst the client and the

# Implementation

# Testing

# Evaluation

# Conclusion