# System design overview

Cloud Sales System – Crayon

Table of Contents

[System design overview 1](#_Toc195404464)

[1. Introduction 2](#_Toc195404465)

[1.1 Purpose 2](#_Toc195404466)

[1.2 Scope 2](#_Toc195404467)

[1.2 Definitions, Acronyms and Abbreviations 2](#_Toc195404468)

[1.3 Context 2](#_Toc195404469)

[2. System design 3](#_Toc195404470)

[2.1 High Level System Overview 3](#_Toc195404471)

## Introduction

**Project name**: Cloud Sales System

### 1.1 Purpose

This System Design Document (SDD) has been prepared as part of an assignment to demonstrate the system overview of **Cloud Sales System**.

### 1.2 Scope

The intended audience: Crayon

### 1.2 Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Definition |
| CSS | Cloud Sales System |
| CCP | Cloud Computing Provider |
| API | Refers to a set of component/system Endpoints |
| HTTP | Hypertext Transfer Protocol |
| MS | Microservice |
| Gateway/BFF | Main entry point of external systems (web apps, other systems) |
| SLA | Service-Level Agreement |
| AKS | Azure Kubernetes Service |
| AZ | Availability Zone |

### 1.3 Context

Crayon wants to implement a solution for cloud sales, which will serve customers in Europe and APAC. Crayon has a business partner, a Cloud Computing Provider (CCP). The CCP offers an API that Crayon can use to automate the business.

Crayon wants to create a way to sell these services to its customers in two ways:

1. A web portal where Crayons customers can log in and perform the actions.

2. A web API where Crayons customers can integrate their systems, and perform the same operations without any user interaction. System to System.

Crayon wants to provide exactly the same set of services in both of the above cases. A customers should be able to choose to use alt 1 or alt 2, or use a combination of 1 and 2.

Primary challenges:

* Entry point for different types of clients (Web applications and services)
* Provide exactly the same set of services for any client
* Data consistency in multi-region deployment

## System design

### 2.1 High Level System Overview

Key challenges considered while designing this system:

* Data consistency and replication
  + Ensuring that all data is available and up-to-date in every region. This is done by continuous geo-replication.
  + Having Primary (Read-Write) database and Secondary (Read-only) is requirement multi-regional systems. When Primary region fails (or primary database fails), Secondary database is promoted to Primary, and all write requests are re-routed to the new Primary.
* Service latency and performance
  + Multi-regional services need to ensure low latency for users from different geographical locations. The goal is to route traffic to the closest region (or the least-loaded) to minimize delays. Azure Front Door is one of the possible tools for geo-aware traffic routing, load balancing.
* Availability and fault tolerance
  + To ensure high availability, proposed solution is to deploy application services into multiple AZs
  + Azure Storage blob is replicating storage into three AZs by default
* Disaster recovery/Backup
* Security and access control
  + Each service must use modern authentication/authorization mechanisms. Proposed solution is Microsoft Entra ID for Identity and access management, and it’s part Microsoft Entra External ID (B2C) for management of end clients (customers)
* Monitoring and logging
  + Azure monitor can aggregate data from multiple regions

Solution of system design overview can be seen in the following diagram.

A screenshot of a computer

AI-generated content may be incorrect.

Figure High Level Overview - Cloud Sales System

### 2.2 Application Components

One of the business requirements is that the application services should be consumed by Web Frontend and by external systems (services).

Proposed solution is implemented as a set of microservices with the main service (Gateway).

Gateway is a entry point of the system. It should be accessed by any type of client via HTTP/S protocol. Gateway also can be easily configured to limit backend services access and acts as a Façade.

With microservice architecture, implementation can be separated between multiple domains:

* Product management
* Customer Management,
* Account management
* Licence Management
* Document management
* Business configuration
* …..

…..more to come….

In the following Figure you can find proposed solution components.