Problem Statement

A nationwide fast-food chain, AnytimeFood with over 7,000 restaurants stores all consumer related information such as food preferences, loyalty status etc in a centralized IT core in their datacenter.

Last week, it experienced major disruptions at its centralized IT core resulting in a 48-hour downtime. The headquarters of AnytimeFood had no way of getting a pulse of the situation at the different restaurants as all this is only available by accessing the centralized IT core in the datacenter. The outage prevented real time engagement with customers, resulted in long delays for customers and left customers very unhappy. AnytimeFood made a huge loss during this outage and it anticipates loss of many customers to their competitors as well.

What Anytime Food has experienced is quite common and can be overcome by using modern technologies and architectures such as containers, microservices and hybrid cloud. Container technology abstracts underlying infrastructure and ensures that the applications can be deployed in a consistent manner across any environment, from the data center to the cloud. A microservices architecture ensures scalability and fault tolerance. Finally, adopting a hybrid cloud architecture improves agility, scalability and business continuity.

Here are some links where you can learn more about container and cloud native technologies:

1. https://www.suse.com/products/rancher-kubernetes-engine/
2. https://www.suse.com/products/suse-rancher/
3. https://community.suse.com/

In this challenge, you will need to design and build a modernized hybrid cloud architecture for AnytimeFood using new container and cloud native technologies to ensure that their critical operations such as food ordering and delivery can still function normally to provide an acceptable consumer experience even during a downtime at its centralized IT core.

Objectives:

You need to design and build a hybrid cloud solution using open source technology. The solution should leverage at least one public cloud provider, together with the existing datacenter and be platform independent. The choice of open source technology and public cloud provider is at your discretion but the solution should be scalable, highly available and provides business continuity in the event of outages.

1. Applications and data that are important for critical restaurant operations such as orders and delivery will be available across the hybrid cloud environments. To save cloud-consumption costs, non-essential applications such as customer loyalty will only be available in the datacenter, however the design should cater on future move to the public cloud as well.
2. All the existing applications should be modernized using container technology.

3. The existing monolithic orders and delivery applications that AnytimeFood currently has should be refactored using a microservices architecture before containerizing them.
4. To scale the microservices effectively, you will need to incorporate Kubernetes.
5. To ease the management and deployment of this new containerized and microservices architecture, a container management platform will also need to be incorporated.
6. There should be a dashboard to indicate the uptime/downtime status of the datacenter and restaurant chains
7. If there is any outage, customers should be able to get notified through the mobile app/sms of the same with alternate options for an acceptable experience
8. During normal solution, all incoming requests can be routed to either the datacenter or public cloud. However during outages, only orders and delivery requests will be entertained and these will be routed to the respective applications on the public cloud. For other requests e.g. checking of loyalty account, the user will be shown a downtime page.

Submission Instructions:

You need to submit following files.

- Properly documented source code.
- A software design document mentioning the followings:
    - Software Requirement Specifications,
    - High level design with data flow and control flow diagrams,
    - Detailed design with class diagrams and relations
    - Test cases and test results with sample screenshots

- A README file mentioning the following,
    - Required packages to run the software
    - Instructions to build the software
    - Instructions to install the software
    - Instructions to run the software