**Assignment 2**

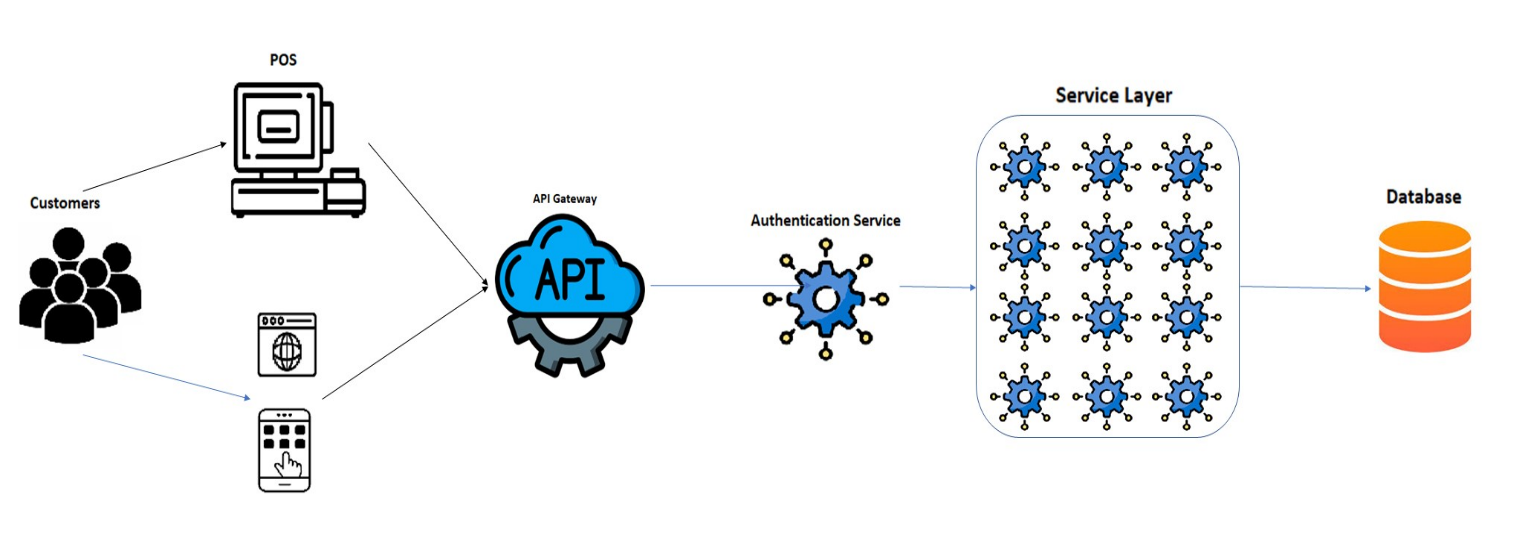
**Implementing a Globally-Accessible Distributed Service  
Distributed Systems - CS7NS6  
Group 9**  
  
**Submitted by:**Anastasiia Khomenko (22304231)  
Charbel Nebo (18324228)  
Karan Dua (21331391)  
Shauna Gurhy (18324848)

**Introduction**

The purpose of this report is to outline the technical architecture of our global café chain service and the technologies that are planned to be used in the implementation of the service.

**Architecture**

The figure below illustrates the architecture diagram for our application

****

**Breakdown of the components in the architecture and justification:**

**Point of Sale (POS) Systems:**

The POS systems are used to process transactions at the café locations and record data about the items purchased, the time and location of the purchase, and the amount spent. This data is sent to the Loyalty Card System for processing.

**Customer App/Website**

The loyalty card app enables customers to earn rewards and redeem them in-store. It also provides personalized recommendations and promotions based on the customer's transaction history and profile.

**Service Component**

We plan to use the microservice architecture for developing the service component of the architecture. Microservice architecture provides some major benefits over traditional monolithic architecture like:

**Scalability:** Microservice architecture allows us to scale up and scale down specific services as needed without affecting the rest of the system.

**Flexibility:** Since each microservice is independent, it is easier to make changes to individual services without affecting the rest of the system.

**Resilience:** Microservices are designed to be fault-tolerant, which means that if one service fails, the rest of the system can continue to function. This is achieved through techniques such as load balancing, redundancy, and failover.

**Simplicity:** Each microservice has a well-defined scope and a simple interface, making it easier to understand and maintain.

We plan to use **Springboot framework and Django Framework** to develop these services. Rest service use HTTP protocol which is TCP as it underlying transport Protocol.

**Rest APIs in the Microservice Architecture:**

1. **Authentication Service**: This service is used to authenticate the customer on the basis of user login credentials.
2. **Account Creation Service:** This service will be used to add new customer to the loyalty program.
3. **Customer Profile Service:** This service will be used to fetch details of the customer like current loyalty points, voucher details etc. based on customer’s username.
4. **Update Transaction Service:** This service will update the transaction details for every transaction that customer does in the cafe
5. **Update Loyalty Points Service:** This service will be used to update loyalty points on the basis of transactions of the customer.
6. **Get Voucher Service:** This service will be used to get the list of vouchers that customer currently have.
7. **Redeem Voucher Service:** This service will be used to validate and redeem the vouchers that customer has.
8. **Generate Voucher Service:** This service will be used to generate the vouchers for customer on the basis of loyalty points.

**Database Component**

We plan to use **MongoDB** to store all the information related to customers and their transactions. The primary reason for using MongoDB is that it is NoSQL database that offers a range of benefits for our Global Loyalty Program application. One of the key benefits of MongoDB is horizontally scalable which makes it perfect choice for our application as it require high availability and scalability. Furthermore, MongoDB also provide dynamic schema which will enable us to handle large number of high-speed, low-latency read and write operations required for our application.

**System Requirements**

**Performance**

We plan to implement Caching mechanisms in our system on the basis of customer transaction history so that customer who visit the store frequently. The primary reason for implementing caches for frequent customer is that when a customer makes a purchase, the system needs to check their transaction/purchase history to determine if they are eligible for any discounts or rewards. Without a cache mechanism, this process can be slow and resource-intensive, particularly when dealing with a large number of customers and transactions. Therefore, caching data for these customers can help us to improve the overall performance of the system. Furthermore, we also plan to set expiration to caches so data is not stored indefinitely.

**Scalability**

We are using microservice based architecture which is high scalable both horizontally and vertically

**Availability**

We plan to use orchestration service like Kubernetes for our application as Kubernetes can automatically detect and replace failed or unhealthy containers, ensuring that the application remains available and responsive even in the event of a failure. Also, Kubernetes can automatically distribute incoming traffic across multiple containers or pods, ensuring that the workload is evenly distributed and the application remains responsive.

**Reliability**

We plan to use database replication strategy like Master-Slave Replication for our application. This will provide better reliability for our system as in the event of a database failure we can involve promoting a slave database instance to become the new master, or restoring from a backup. Furthermore, we will take regular backups of the database to ensure smooth restoration of the system from point of failure.

**Usage Patterns**

We expect users to be globally distributed and they can access the application from browser or mobile application. Users will utilise this feature to apply for loyalty program, check their loyalty point bonus and applicable vouchers. Since the loyalty program is based on the transaction history of the user, we don’t expect any data to be shared among users. We also expect more write operations on the database for every purchase the customer makes, therefore, we are using MongoDB which highly efficient in handling high volumes of write requests. We are also implementing cache mechanism for storing more frequently used items in the caches to improve overall performance of the application. We don’t expect system to fail frequently, however to handle failure scenarios we are using Kubernetes to ensure High Availability and Database Replication strategy for database failures.

Overall, this architecture provides a scalable and flexible solution for managing a loyalty card scheme for a large global café chain with a highly mobile set of customers.