Table of Contents

Version Information

Version	Date	Author	Description
1.0	Nov 21, 2020	Dhaval Mehta	First Draft
1.1	Nov 25, 2020	Dhaval Mehta	Fixed Customer Registration sequence Diagram

Abstract

This document outlines the design and architecture of Coffee Shop Services.

Requirements

Background

A global coffee shop chain / franchise intends to launch an app to allow their regular customers to pre-order coffee to pick up (say, on their way to work).
They have identified the following needs:

1. The coffee shop chain is a global network. So, they need to service shops locations across multiple geographies.
2. The space is quite limited. So, they want everything to work easily on an app.
3. They need two apps (a) one for the shop owner and (b) one for the customer
4. They have decided to build (a) on Android and (b) on iOS
5. Not all coffee shops have the same menu. So, they need to be able to handle a menu based on the shop
6. Most of their shops have only one queue, but some shops are able to support up to 3 queues
7. They would like their service to be API enabled, so that others (3rd parties) can build apps using their APIs

The coffee shop app

This section outlines the requirements for the coffee shop app.

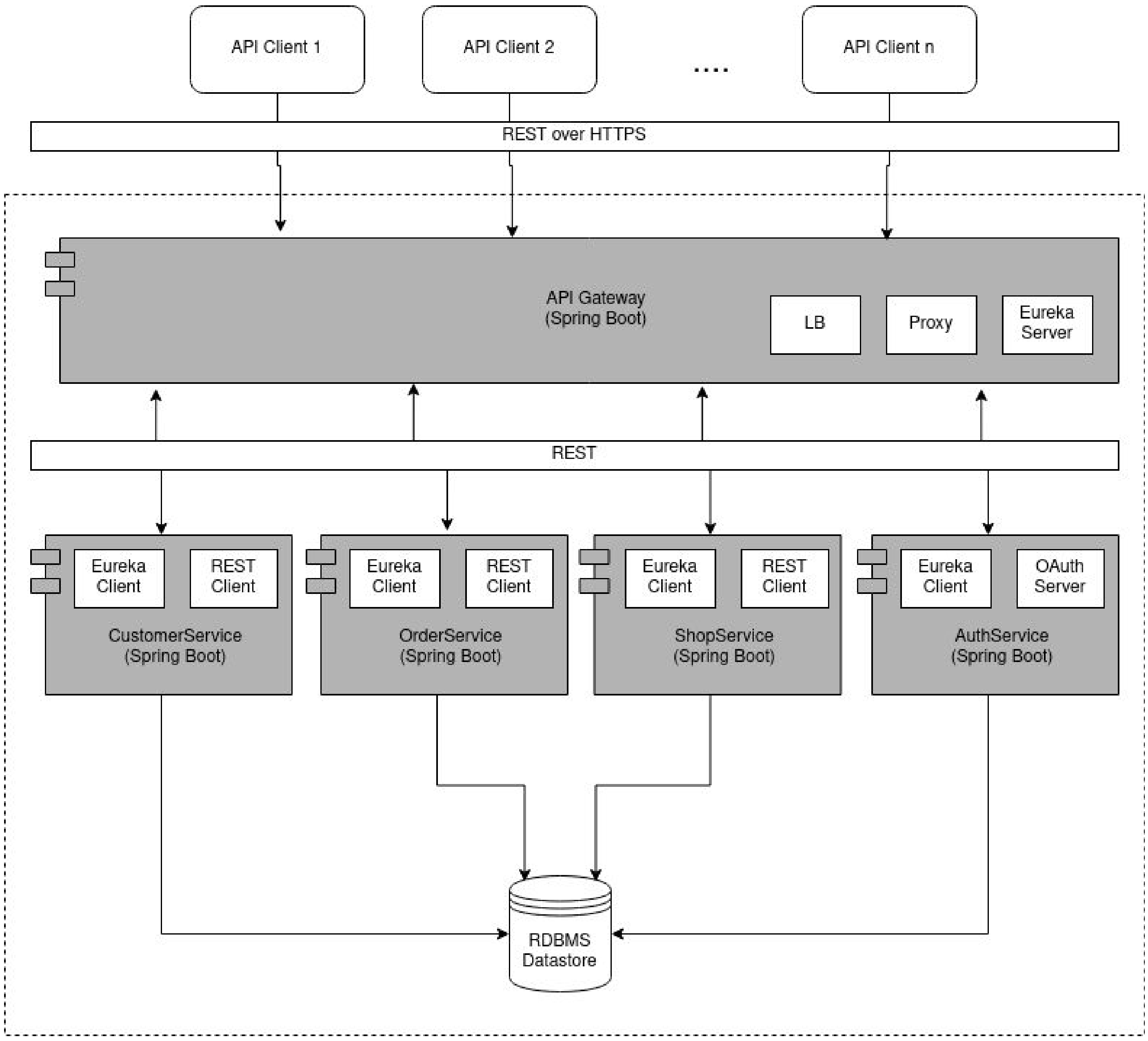
1. Allows the shop owner to login as an admin user
2. Allows the shop owner to setup / configure the app to support their shop
3. Allows the shop owner to configure the shop / app as follows:
    a. Location and Contact details
    b. The coffee menu & pricing
    c. Number of queues and the maximum size of the queue
    d. Opening / closing times
4. Allows the shop operator to login and manage the queue
5. To view / see the size of the queue and the number of waiting customers
6. To easily view the orders placed by the customers in the queue
7. The name of the persons in the queue
8. A score indicating the number of times that customers has been served by the coffee shop chain
9. Take a customer off the queue and service them

The customer app

This section outlines the requirements for the customer app.

1. Allows the customer to register with their mobile number, name and regular address (home or work)
2. Allows the customer to view and find the coffee shops closest to them
3. Place an online order for a coffee from the menu
4. See their position in the queue (and expected waiting time before collecting the coffee)
5. Exit the queue at any time (and notify the shop to cancel the order)

Architecture



- Microservices based architecture is chosen due to following advantages:
  - Each microservice can be developed / deployed / scaled independently
  - Provides for better isolation between services
- Each microservice will be implemented in Spring Boot as it has excellent support for developing microservices - Service Discovery (Eureka), API Gateway (Zuul Proxy), REST Client for Interservice Communication
- As indicated in the diagram, the APIs will be exposed as REST APIs
- All the external communication will pass through an API Gateway. It will also act as Eureka server to enable service discovery. API Gateway will be responsible for forwarding requests to individual services
- Interservice communication will also happen through API Gateway. API Gateway will host a Eureka server instance to enable service discovery
- All services will share a common datastore. Postgres will be used as primary datastore

Security Considerations

- All the APIs will be exposed over HTTPS (which will terminate in API Gateway). This will prevent from MITM attacks
- AuthService will provide Username / Password based authentication. Passwords will be stored after Hashing and Salting

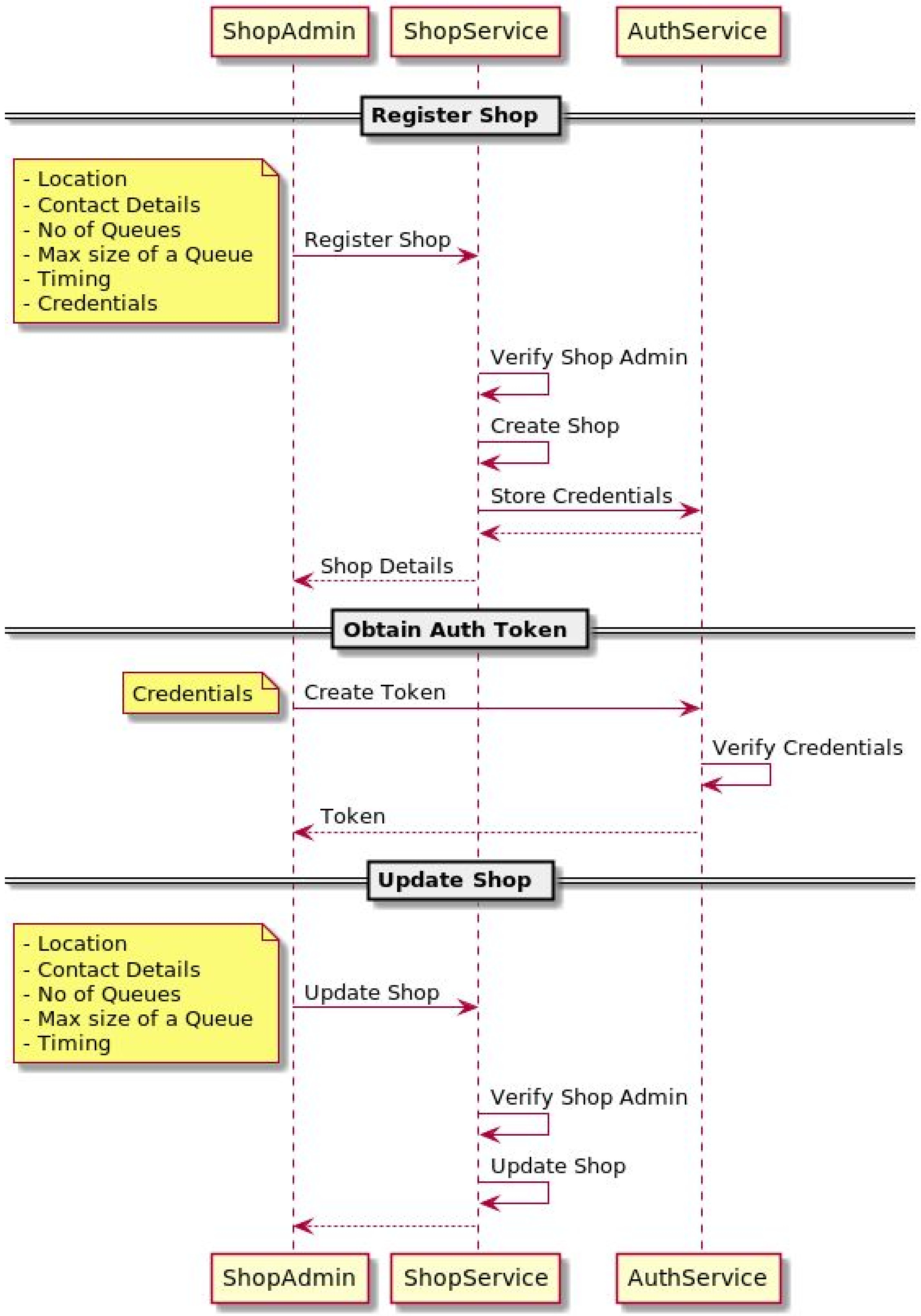
Authentication and Authorization

- AuthService will act as an OAuth server. Having an OAuth server will allow us to provide different auth mechanisms for different clients. Interservice communication will also be authenticated through OAuth, with each RESTClient obtaining JWT Tokens through ClientCredentials grant type
- Each service will implement Role Based Access Control (RBAC) for each resource that it owns. E.g. ShopService will implement RBAC for Shops related services
- Keycloak should be evaluated for federated authentication. Using it will enable to have easy integration to LDAP (for in-store staff) / Social Auth / OTP based Auth (for customers) mechanisms

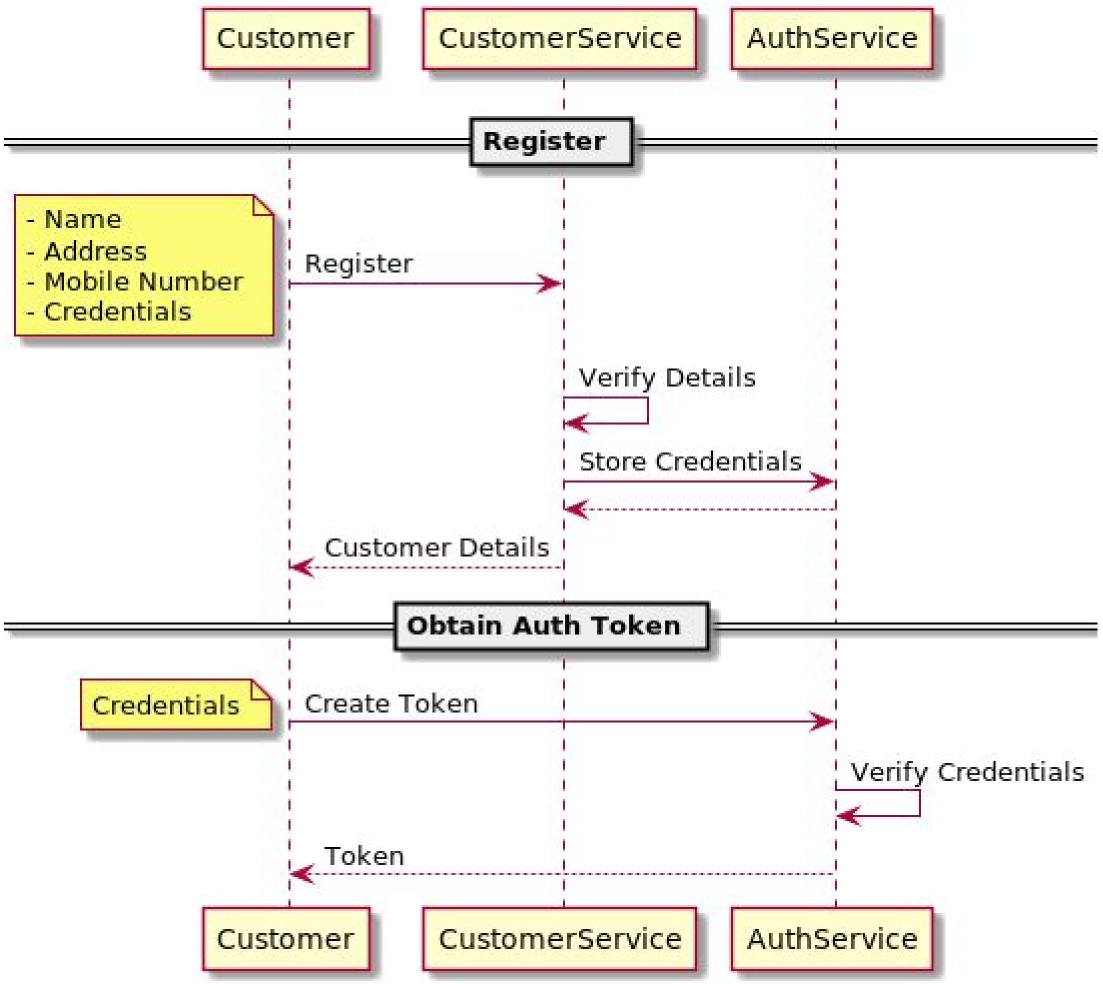
Data Flow / Sequence Diagrams

Following sequence diagrams indicate flow of data across multiple services for various use cases

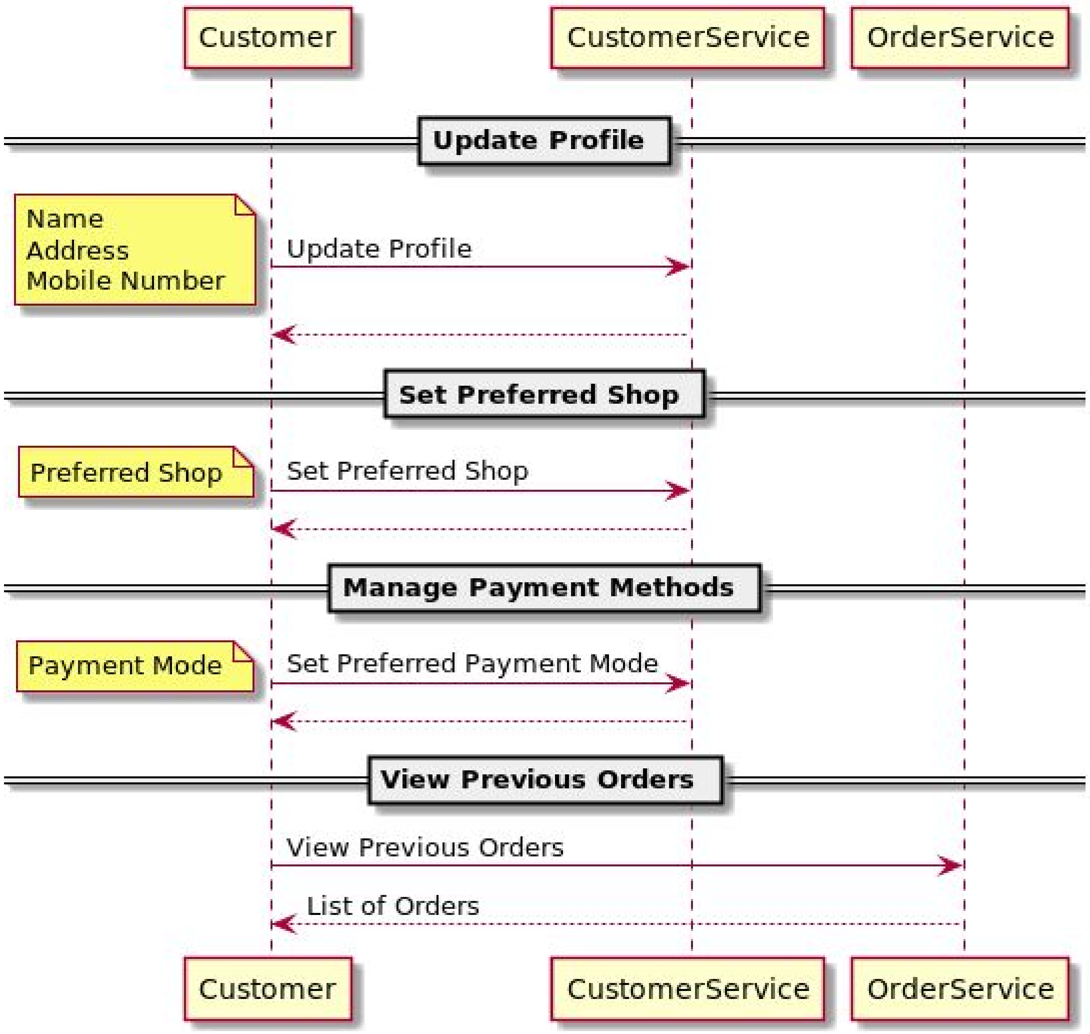
Manage Shop


ShopAdmin
ShopService
AuthService
Register Shop
- Location
- Contact Details
- No of Queues
- Max size of a Queue
- Timing
- Credentials
Register Shop
Verify Shop Admin
Create Shop
Store Credentials
Shop Details
Obtain Auth Token
Credentials
Create Token
Verify Credentials
Token
Update Shop
- Location
- Contact Details
- No of Queues
- Max size of a Queue
- Timing
Update Shop
Verify Shop Admin
Update Shop
ShopAdmin
ShopService
AuthService

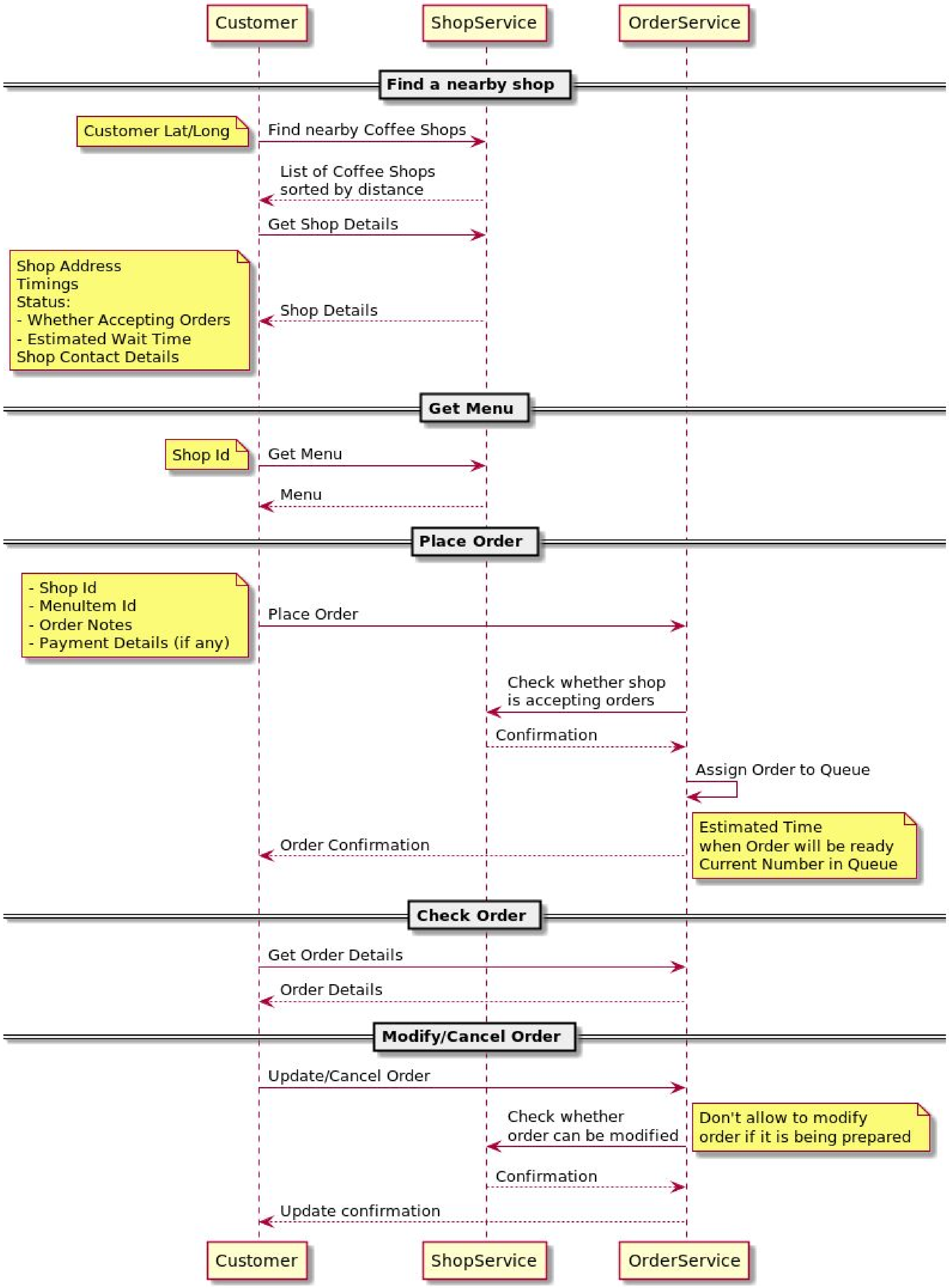
Customer Registration



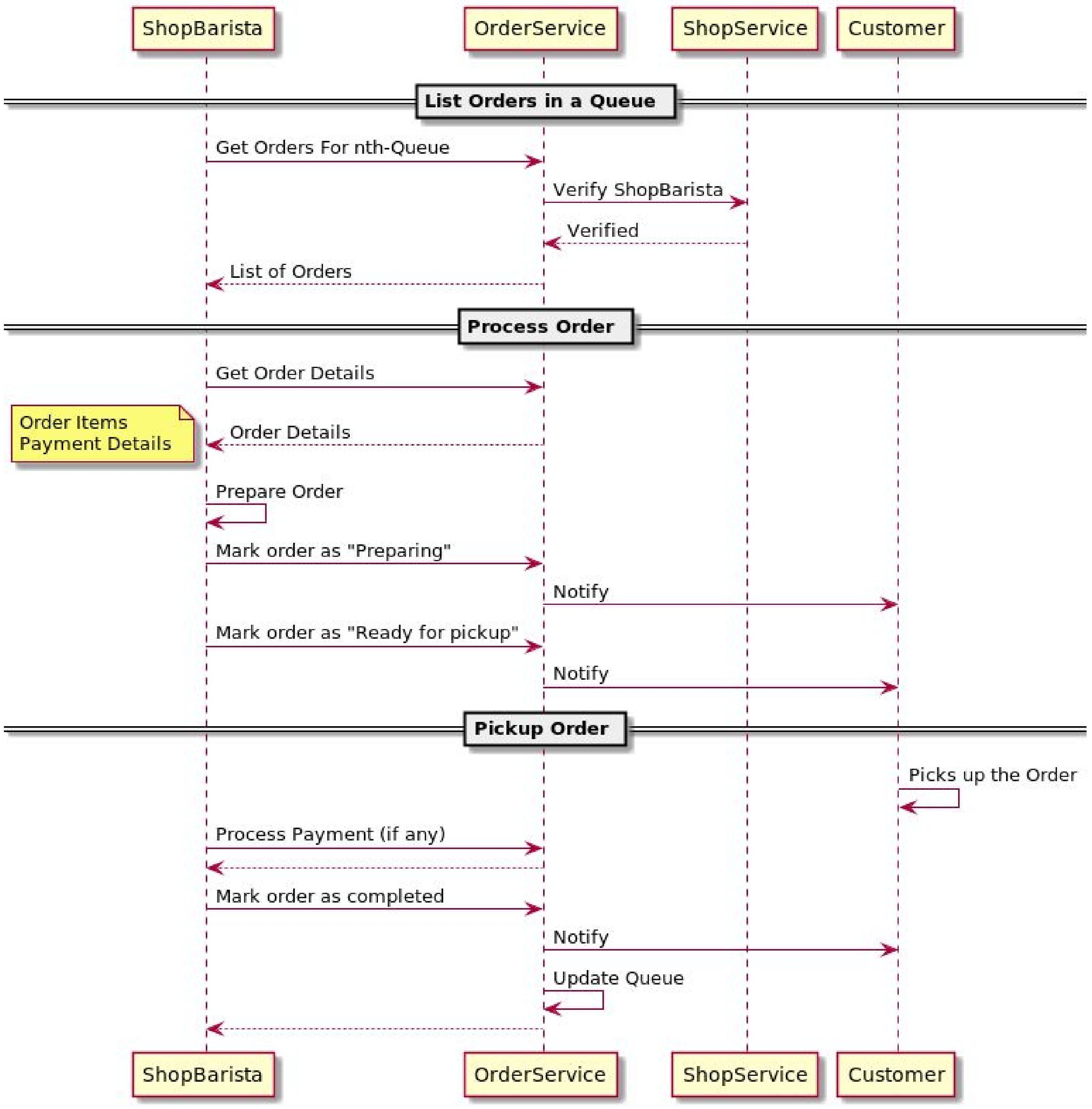
Customer Profile Management



Order Placement



Processing an Order



Estimates

Description	**Estimated Efforts (hours)**
DB Design	6
Bootstrapping Services (Setting up Spring Boot microservices, Docker-compose setup for local development, Liquibase scripts, etc)	10
Build and Deployment Pipeline	8

API Gateway (Setting up Eureka Server, Proxying etc)	8
AuthService (OAuth Server, JWT Token APIs)	10
Shop Registration / Management	8
Customer Registration / Profile	8
Order Placement	14
Order Processing	12
Total	**84**

Development Guide

- The whole project will be a git monorepo containing all the services and ops related code
- maven will be used for dependency management and building each individual microservices
- Each deployable component will produce a docker image (spring-boot-maven-plugin will be used to build optimized docker images)
- Local dev environment will be setup using docker-compose
- Coding Standards : Google Java Style Guide will be following for coding and naming conventions
- Unit tests for services will be written using JUnit / Mockito
- End-to-end tests will be run using Postman Runner
- API endpoint documentation will be created via Swagger

API Endpoints

APIs are documented at https://app.swaggerhub.com/apis-docs/mehtadhaval/CoffeeShopServices/1.0.0