Software Architecture Document

Coffee shops system – 101 Digtal

Tran Dong Hai

Version 1.0

October 2023

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 10/01/2023 | 1.0 | Initial version of SAD for comments by team | Tran Dong Hai |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc377952936)

[1.1. Purpose 4](#_Toc386690300)

[1.2. Scope 4](#_Toc1407080541)

[1.3. Definitions, Acronyms, and Abbreviations 4](#_Toc1134723265)

[2. Architectural Representation 4](#_Toc1071970347)

[3. Architectural Goals and Constraints 4](#_Toc1355445763)

[4. Use-Case View 5](#_Toc794018881)

[4.1. Actors 6](#_Toc1631826054)

[4.2. Architecturally Significant Use Cases: 6](#_Toc1837744416)

[5. Implementation view: 7](#_Toc67967224)

[6. Logical View 7](#_Toc95617949)

[6.1. Order states diagram: 7](#_Toc783477728)

[7. Process View 8](#_Toc471429970)

[8. Data View 9](#_Toc258668597)

[9. Deployment View 9](#_Toc1240138193)

[10. Size and Performance 9](#_Toc1432331566)

[11. Issues and concerns 10](#_Toc1496389954)

Software Architecture Document

# Introduction

## Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Coffee Shops system. It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made on the system.



## Scope

The scope of this SAD is to depict the architecture of the Coffee shop system that is being developed by 101 Digital to support selling coffee online.

This document describes the aspects of the Coffee shop system design that are architecturally significant; that is, those elements and behaviors that are most fundamental for guiding the construction system and for understanding this project. Stakeholders who require a technical understanding of the system are encouraged to start by reading this document, then reviewing the UML model, and then by reviewing the source code.

## Definitions, Acronyms, and Abbreviations

* **PostgreSQL** – relational database management system (RDBMS)
* **SAD -** Software Architecture Document
* **Shop owner -** This is a user who can create/modify a Shop and its settings such as menu, price, product list ...
* **Shop operator -** This user can manage the order queue

# Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view, implementation view, and deployment view. These are views on an underlying Unified Modeling Language (UML) model developed using Rational Rose.

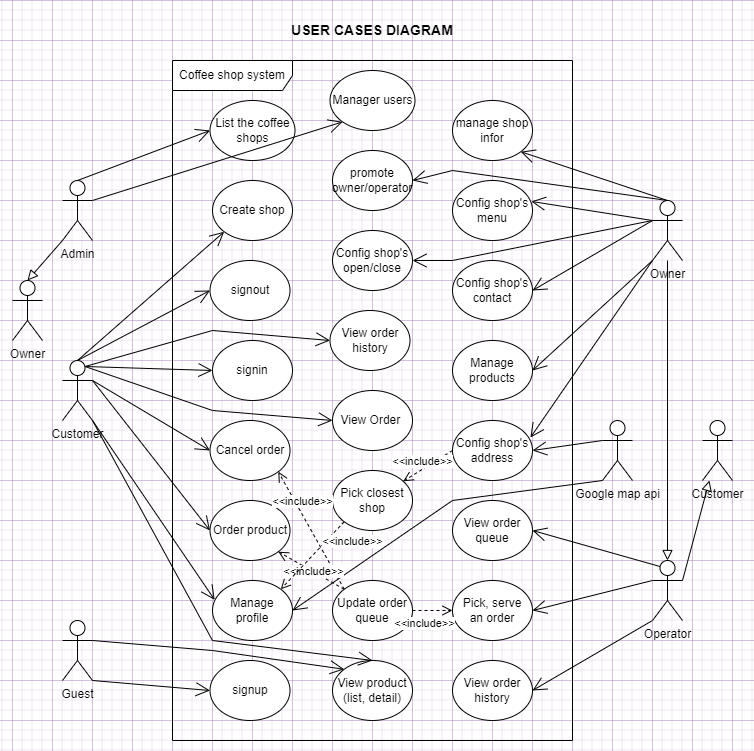
# Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

* Client side: there are two mobile apps for the shop owner and customer. They are both iOS and Android mobile apps.
* Server side:
  + Deploy on AWS EC2 containers.
  + It has microservice architecture
  + Developing by Java and Spring Boot framework
  + Backend services will be hosted on one of Apache Tomcat web servers running on a Linux platform.
* MQ server: RabbitMQ or AWS SQS in case there is a flow can process asynchronously
* Database: PostgreSQL
* Performance: There is no particular constraints related to system performance. The actual performance can be determined only after system deployment and testing.
* Security:
  + With microservice architecture, we will use build authentication and authorization feature with OAuth2/OpenId protocol for SSO. For this mini project we can use the open-source keycloak is an IDP server. For the real project, we can use AWS Cognito with API Gateway and AWS Lambda.
  + About the role: The guest rights will be granted to access the mobile app for the customer to view the product list of the default shop. The customer rights will be granted to access the mobile app for the customer to view the product list of the closest shop, order coffee and config profile, and address. The shop owner rights will be granted to access the mobile app for the shop to configure shop’ settings such as menu, max order queue, product list, price, shop address, shop open/close time … The shop operator rights will be granted to access the mobile app for the shop to manage the order queue, pick a customer in the order queue to serve, and view the list of shops.

# Use-Case View

This is a list of use-cases that represent major functionality of the final system [SRS]:



## Actors

As described in the actors’ correspondence diagram below, web user could be one of five types:

1. **Shop owner:** Create, update, delete own shop, config shop’s settings such as menu, address, contact, open/close time, max order queue ….
2. **Shop operator:** view list order queue, pick an order to serve, delete an order in queue
3. **Customer:** view the list of products by category or search, view product details, make/omit an order and view order queue
4. **Guest:** view the list of products by category or search, view product details.
5. **Admin:** Manage the shop list, user list
6. **Google Map API:** It is a third-party API to identify the longitude/latitude of the address, and support to find the closest shop of a customer.

The admin extends features from the owner, the owner extends features from the operator, and the operator extends features from the customer, they are login users and have account credentials. The guest has no account credentials.

## Architecturally Significant Use Cases:

Sign Up, Sign In, and Sign Out: These use cases are for authentication and authorization.

Manage shop information, Config shop's address, Config shop's menu, Config shop's open/close time, Config shop's contact …: These use cases allow the shop owner to change the shop’s settings: menu, contact, address, maximum queue, and open/close time.

Manage product: Allow the shop owner to manage category, product, and price.

View order queue, pick and serve an order, View order history: Allow the shop operator to manage queue and serve user.

View order, order product, cancel order, …: These user cases allow customers to make an order online.

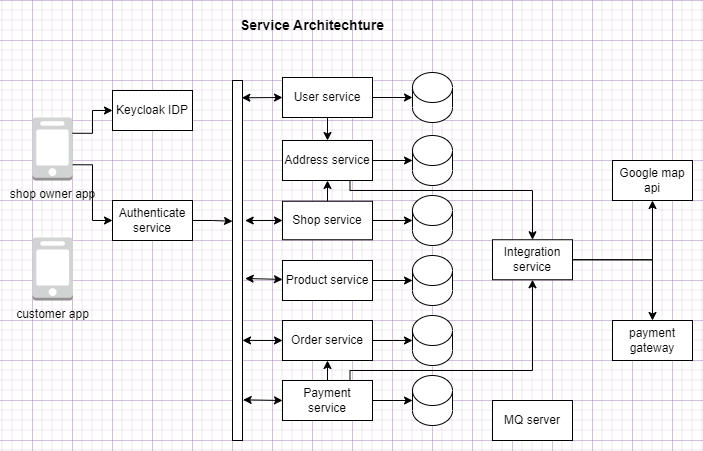
# Implementation view:

The system has two mobile apps for shop owners/operators and customers.

The backend has a microservice architecture with the following services:

* Address service: manage address information, interact with google map api via integration service
* Shop service: manage shop information, settings such as menu, contact, address, open/close time, max queue.
* Product service: manage category, product, price
* Order service: manger product order, order queue
* Payment service: interact with payment gateway (Fo example: stripe) via integration service.
* Integration service: communicate with google map api
* User service: manager user account
* Authentication service: SSO authentication and authorization with Keycloak IDP, KeyCloak IDP is also embedded in this service.

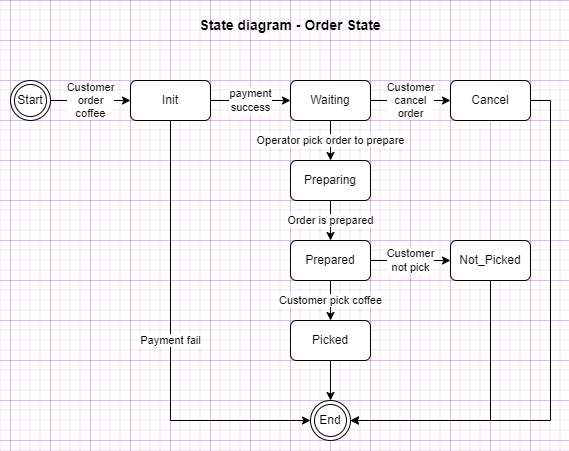
Each backend service has a separate database schema and other services do not manipulate with database schema of other services. These services will communicate with each other synchronously via HTTP protocol. Each service exposes a swagger document for its rest API.



# Logical View

## Order states diagram:

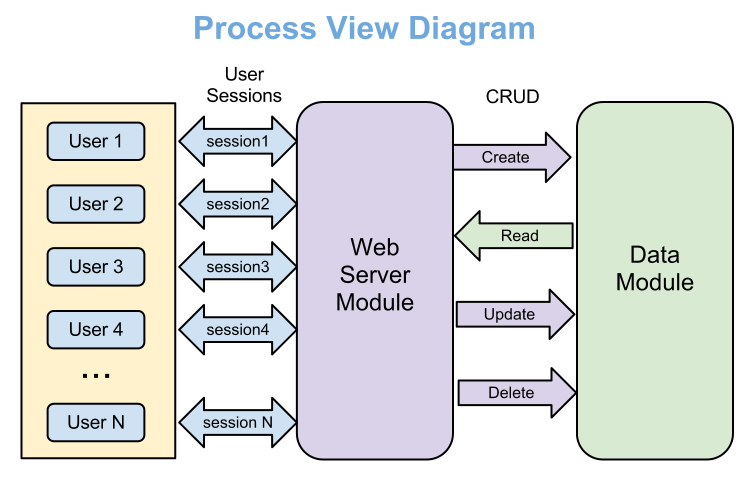
From an order is created to it is closed, there are following stated: Init, waiting, preparing, prepared, picked, not\_picked, cancel. Here is order state diagram:



When the shop is busy, the order queue is full, the order is put into another queue, if all order queues are full, the order is rejected, its state is still Init, but payment process is done. System must refund money to customer when payment is done but state is still Init or Cancel.

# Process View

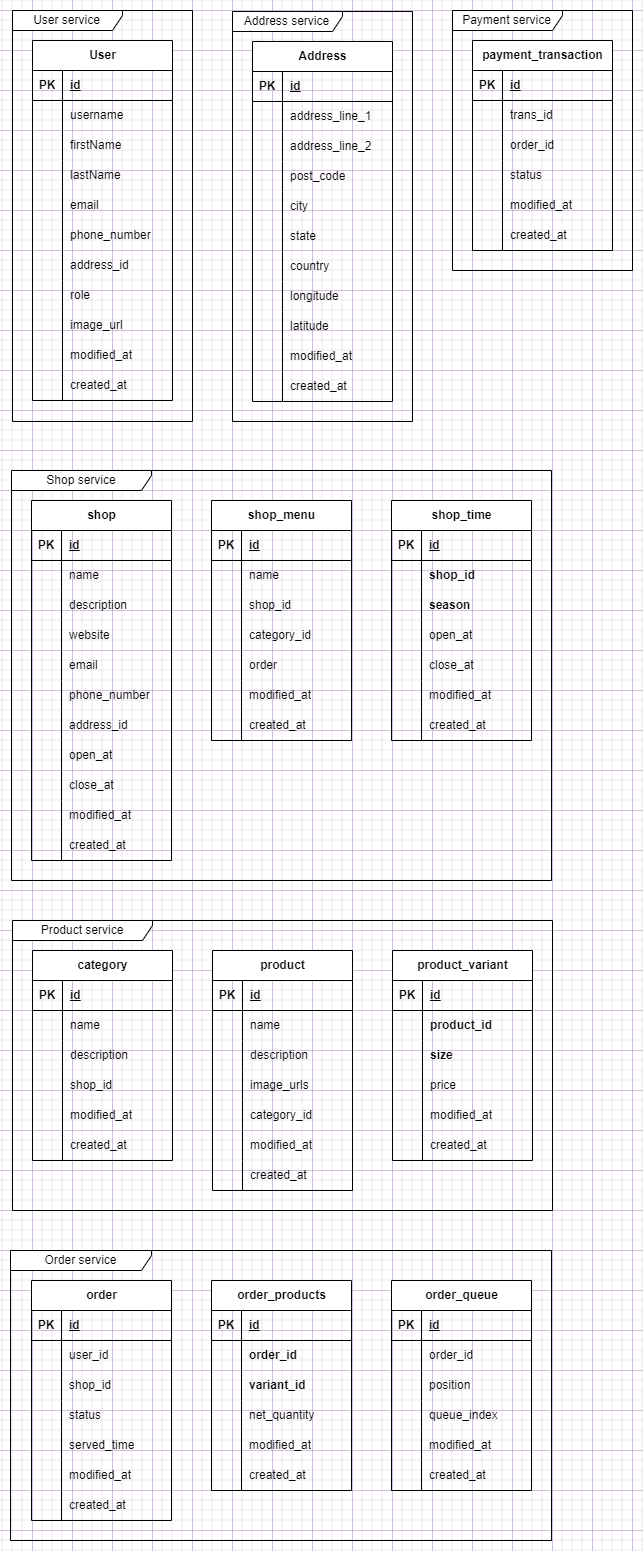
Due to disconnected nature of HTTP request / response and ability of relational database management system (RDMS), DHCPII tool will handle multiple users simultaneously. Therefore, concurrency issues such as synchronous versus asynchronous mechanisms will be not considered in this document.



* User – Creator, Reader or Administrator
* Session – HTTP session assigned by web server automatically
* CRUD – Create-Read-Update-Delete

# Data View

Each service has separate database schemas. Here are databases:



# Deployment View

DTCPII tool deployment has not been considered yet. All future implementation details will be included in this section.

# Size and Performance

Volumes

* Simultaneous users 30 max (OMSE students)
* Data storage under 10MB per user (including uploaded charts)

Performance

* With maximum load all transactions well under standard server script / database connection timeout – 20 seconds.

# Issues and concerns

* User authentication / integration with PSU’s systems
* Charts (image) uploading – is it feasible to save images as binary field into database and what the upload file size limits
* Do we need to create user interface for Process Specification Templates
* The data structure will allow creating only two-level process hierarchy (Process Components and Sub-components). Additional level of hierarchy will require changing data structure.