SOA Compositions

16/07 - Web Service Dev & SOA

Adenda

What’s On the menu? - Week 3

I: Common Pitfalls (from the Last Exercise)

II: SOA Introduction

III: SOA Compositions

I: Common Pitfalls in Exercise

16/07 - Web Service Dev & SOA

Common Pitfalls

Eliminate These During Group Project and Exams

Generic Answer
(Next Slides)

Incomplete Diagram
(Easy Fix)

- REST requests.
- Seq. no. for multiple requests.
- E.g. Type of arrow matters - (async & sync. requests).

Missing Answer
(Easy Fix)

- Just read questions carefully.

Generic Answer

- **Analyse. Don't report:** When there is a question **asking for a rationale for some decisions**, **analyse them** to your problem context. **More specific = More score**.

- Analysis creates more opportunity to **get even more score** from:
    - **Connect them to knowledge from this class (C1)** and **your experience (C2).**
    - **Make an example** to convey your design decision (C2).
    - Observe **limitations** from the current decision (C2).
    - Elaborate on what can be **future directions (C2)**.

- **Provisional Score for Generic Answer: C (At most)**
- **Provisional Score for Specific Answer: C+ (At least)**

Generic Answer

Tips to Make An Answer More Generic

Me = Customer

(During Exams & Presentation)

Convince me with your design decisions.

You = Software Architect

Design a system. Not a walking ChatGPT.

Outside of Reqs? Free to Decide.

Make sure its **sound and reasonable**.

Ask me

(During Group Exercises)

For your **practice** & **customised** feedbacks.

Generic Answer

Examples (From Last Week Question)

Q: Provide an example of a data structure that can gathered from **the system** in XML or JSON format. The example must include **at least one rationale** of why XML or JSON has been chosen.

(From ChatGPT 4o)

JSON is often preferred over XML for data interchange due to its simplicity, readability, and efficiency. JSON's compact structure, ease of parsing, and native support in JavaScript make it well-suited for web development. It directly supports common data types and integrates seamlessly with RESTful services.

Score: C (C1) + F (C2) = D

We chose JSON for its compact data format over XML **to improve communication between Front-End and other components, especially on wireless tablets, which is prone to connectivity issues.** **This applies to both sync. and async. requests.** **Although connectivity problems may still occur with JSON, we'll add a checksum to each request to identify the completeness of the request.**

Score: A (C1) + A (C2)

Generic Answer

Examples (With Last Week Question)

> *(From ChatGPT 4o)*
>
> JSON is often preferred over XML for data interchange due to its simplicity, readability, and efficiency. JSON's compact structure, ease of parsing, and native support in JavaScript make it well-suited for web development. It directly supports common data types and integrates seamlessly with RESTful services.

- **C1: C - Broad knowledge** (This can apply to apply any SW projects).
    - 52: Closer to D (Superficial understanding) than C+ (Systematic understanding)
- **C2: F - No attempt to analyse** (No analysis to the system).
    - 31: No analysis but sound generalisation.
- **Weight:** C1 (50/80), C2 (30/80)
- **Aggregate:** 52(*(50/80))+31(*(30/80)) = 32.5+11.63
  = **44.13 (D)**

Generic Answer

Examples (With Last Week Question)

> We chose JSON for its compact data format over XML to improve communication between Front-End and other components, especially on wireless tablets, which is prone to connectivity issues. This applies to both sync. and async. requests. Although connectivity problems may still occur with JSON, we'll add a checksum to each request to identify the completeness of the request.

- **C1: A -** Acknowledge **limitation of the course knowledge** (Pink Part) & Evidence of **knowledge and understanding** (Purple Part).
    - 92: Not many detailed added.
- **C2: A - Independent high-quality analysis** (Checksum (Pink Part) & connectivity issues (Green part) are not from this course; Sounds & Persuasive).
    - 95: No evidence of contradictions consolidation.
- **Weight:** C1 (50/80), C2 (30/80)
- **Aggregate:** 92(*(50/80))+95(*(30/80)) = 57.5+35.63
  = **93.13 (A)**

II: SOA Introduction

16/07 - Web Service Dev & SOA

Service-Oriented Architecture (SOA)

A SW architectural style, encompass of services. Key characteristics are:

Loose Coupling

(Week 2)

Interoperability

Platform or programing language of a service is independent to the other.

Scalable

(Week 5)

From last week example: Rush hour in Coffee Shop

Unified Data Format

(Week 4)

Based on XML & JSON (Week 2)

Observation: SOA is a distributed computing architecture. Can use the non-HTTP protocol (e.g. DCOM or ORBs), but we will not cover here.

Service-Oriented Architecture

Key roles are:

Service provider (e.g. A web service)

Service broker/registry Proving metadata or information of service provider to a requester.

Service requester (e.g. Client)

Service-Oriented Architecture

Real-life example: AirBnB

Service provider: Tenants: Provide a place to stay & facilities.

Service broker/registry AirBnB: Provide Tenant info, price & location to customers

Service requester: Customers : Get info from AirBnb, Stay with a tenant.

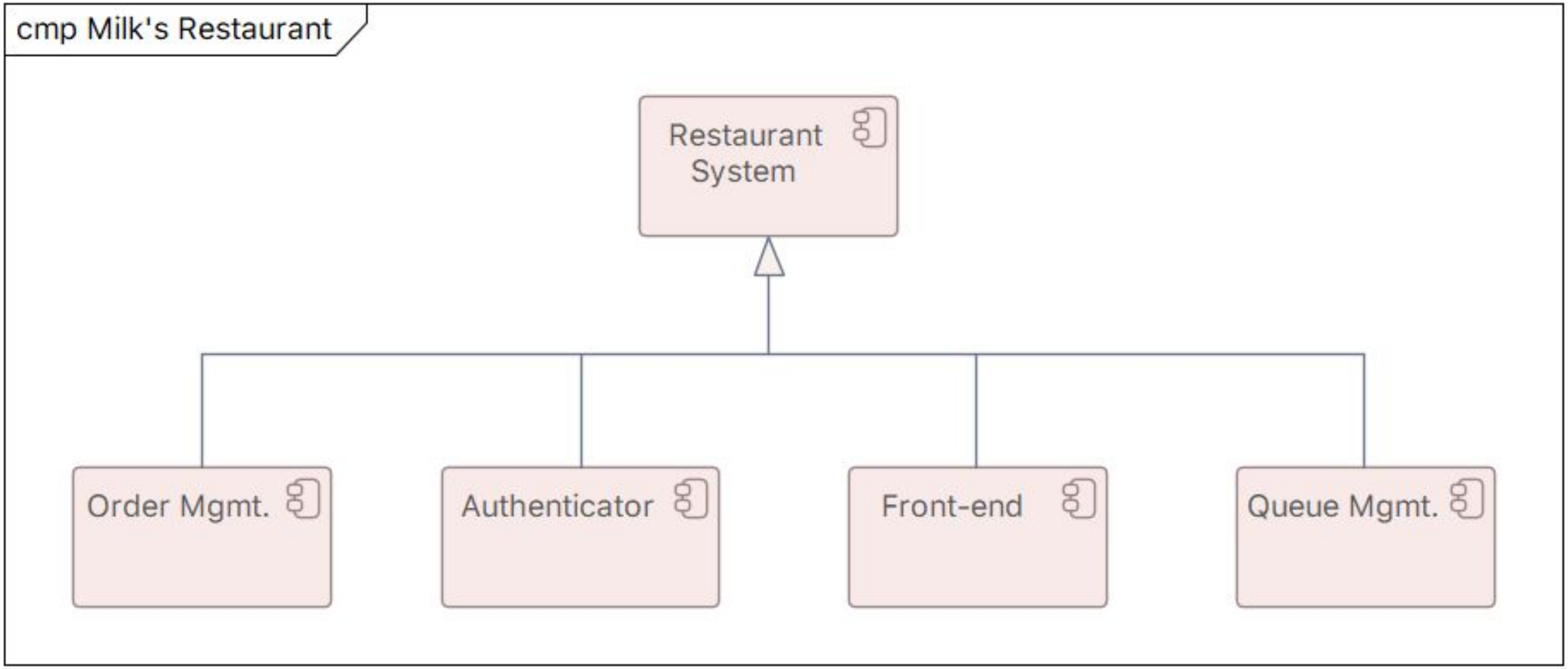
Observation: Service broker/registry decouples Service Provider & Requester. They do not need to they each other info from the beginning. **Will look into how it does next week.**

III: SOA Compositions

09/07 - Web Service Dev & SOA

SOA Compositions

From Last Week: We know what services are, but How to Compose them.



SOA Composition

Two Topologies:

Mediator-based

Peer-to-Peer

Note: A system can have both composition topologies (see running example).

SOA Composition

: Mediator-based

A mediator service is responsible for:

- **Receive** a request from a **service consumer.**
- **Control** the execution of **the other services.**
- (Optionally) Check a well-being/status of the other services.

Real-life example: A conductor in an orchestra.

SOA Composition

: Mediator-based - Pros

A mediator service can be:

- **Easy to adapt from three tier architecture:** A mediator is like a “control” layer for other services.
- **Centred execution:** Facilitate in verify **the correctness/accuracy** of service outputs.
- **Fast:** One-two hops max. from a mediator to other component services.

SOA Composition

: Mediator-based - Cons

A mediator service can be:

- **Complex to maintain:** Can be **time-consuming to identify** which parts is for which services later on.
- **Single point of failure:** A mediator **down** = the system **down**.
- **High impact after an attack:** One **overtaken** service to rule them all.
- **Non-reusable:** A mediator is **specifically implemented** for a specific set of services**.**
- **Unreliable** as performance overhead depends on the number of service consumers.

SOA Composition

: Peer-to-Peer

A peer-to-peer service is responsible for:

- **Receive** a request.
- **Sent** a response.

Real-life example: A train/a subway; One carriage connect to the other.

SOA Composition

: Peer-to-Peer - Pros

A peer-to-peer service can be:

- **Easy to maintain:** Always one client & server in itself.
- **Isolated Failure:** A service down = one function down, not the whole system.
- **Low impact from the attack**: Only one chain of the system is overtaken. The system is still unexposed.
- **Stable Performance**: Always one client & server.

SOA Composition

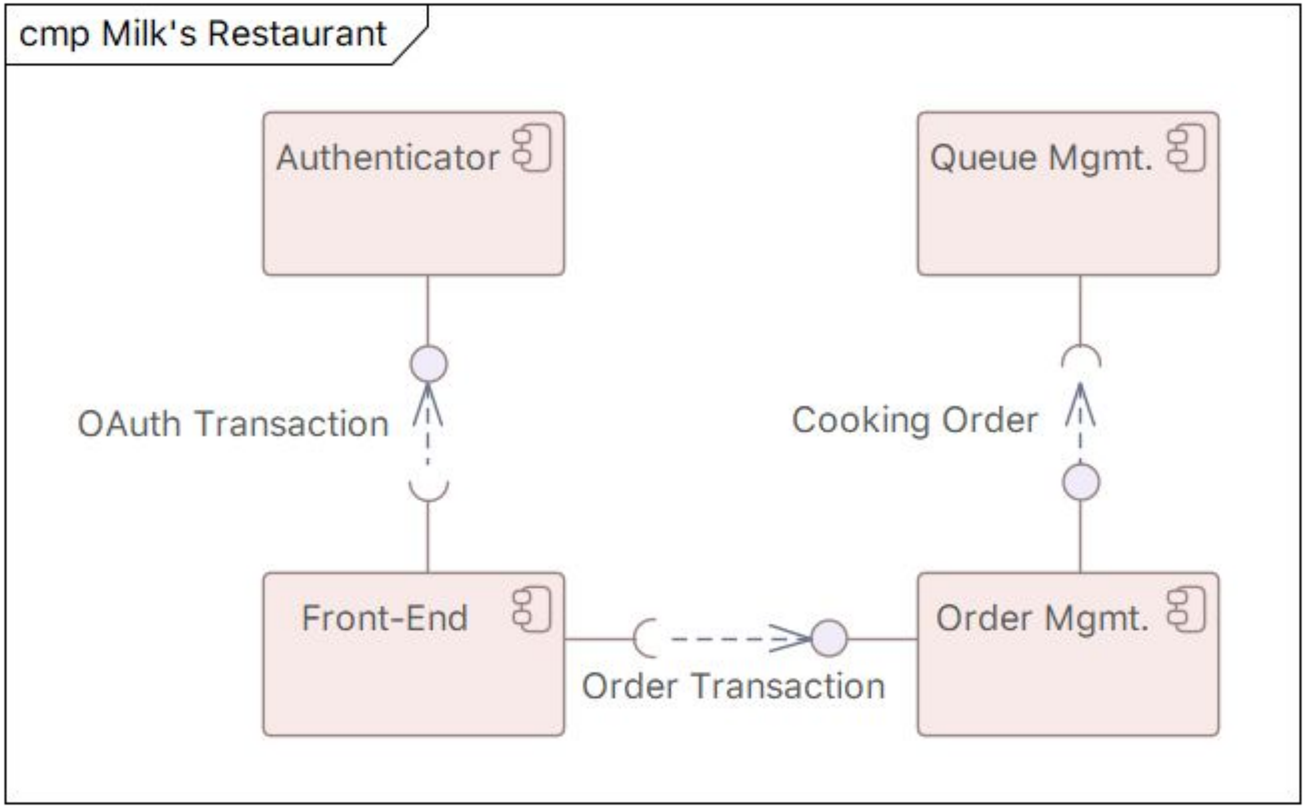
: Peer-to-Peer - Cons

A peer-to-peer service can be:

- **Costly to adapt,** it’s can be labour-intensive to form a service to serve only one service consumer/provider.
- **Difficult to backtrack/check** for the transaction correctness.
- **Slow:** Several hops may be required for one use case.

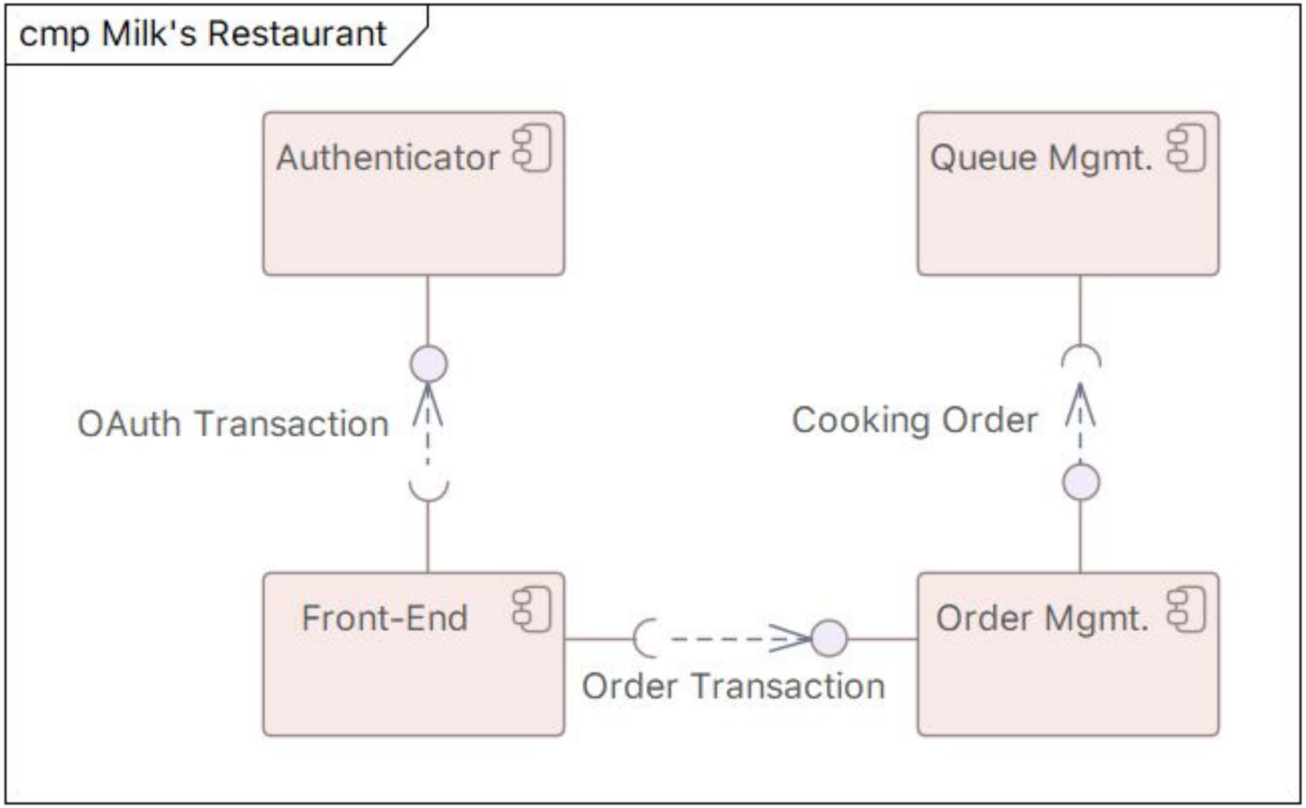
Running Example

Mediator: Front-End -> Authenticator
-> Order Mgmt.



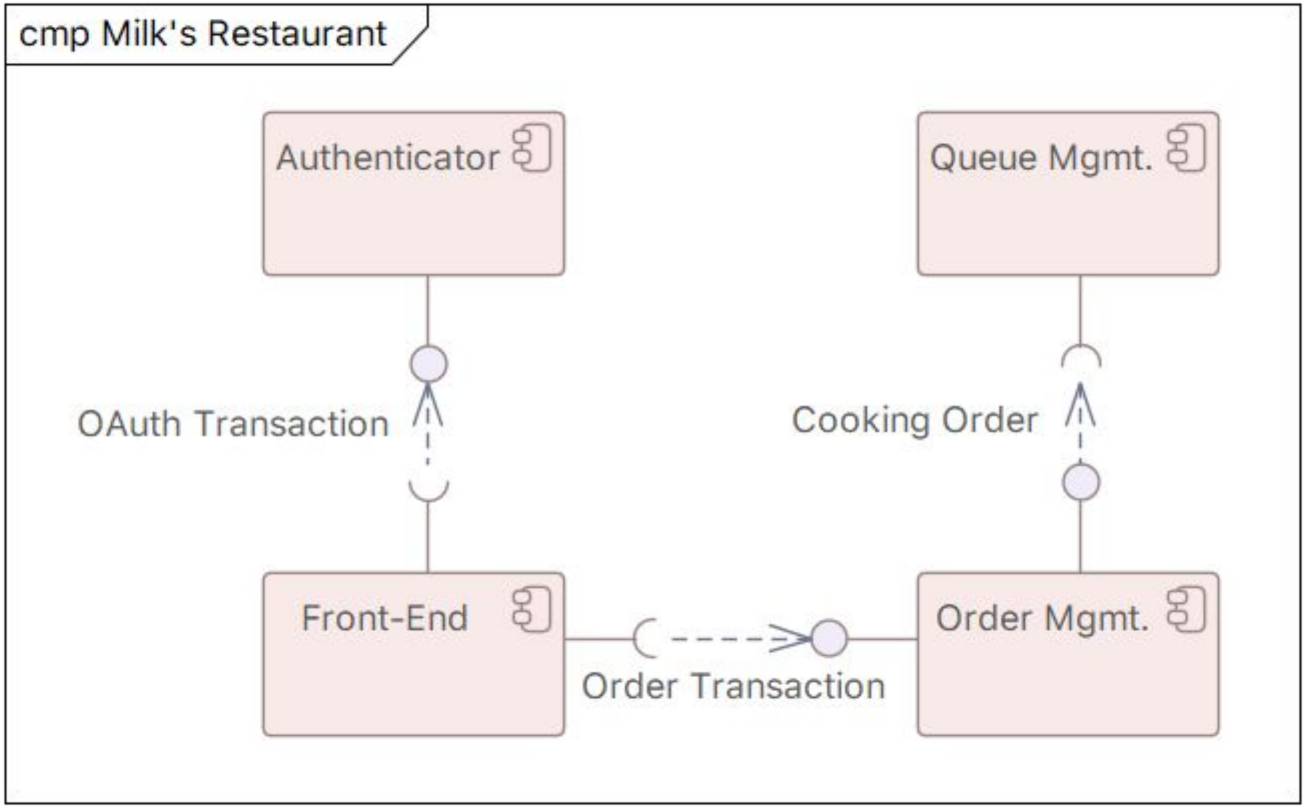
Running Example /w Analysis

Mediator - Front-End: Can flood Authenticator & Order Mgmt. to requests.



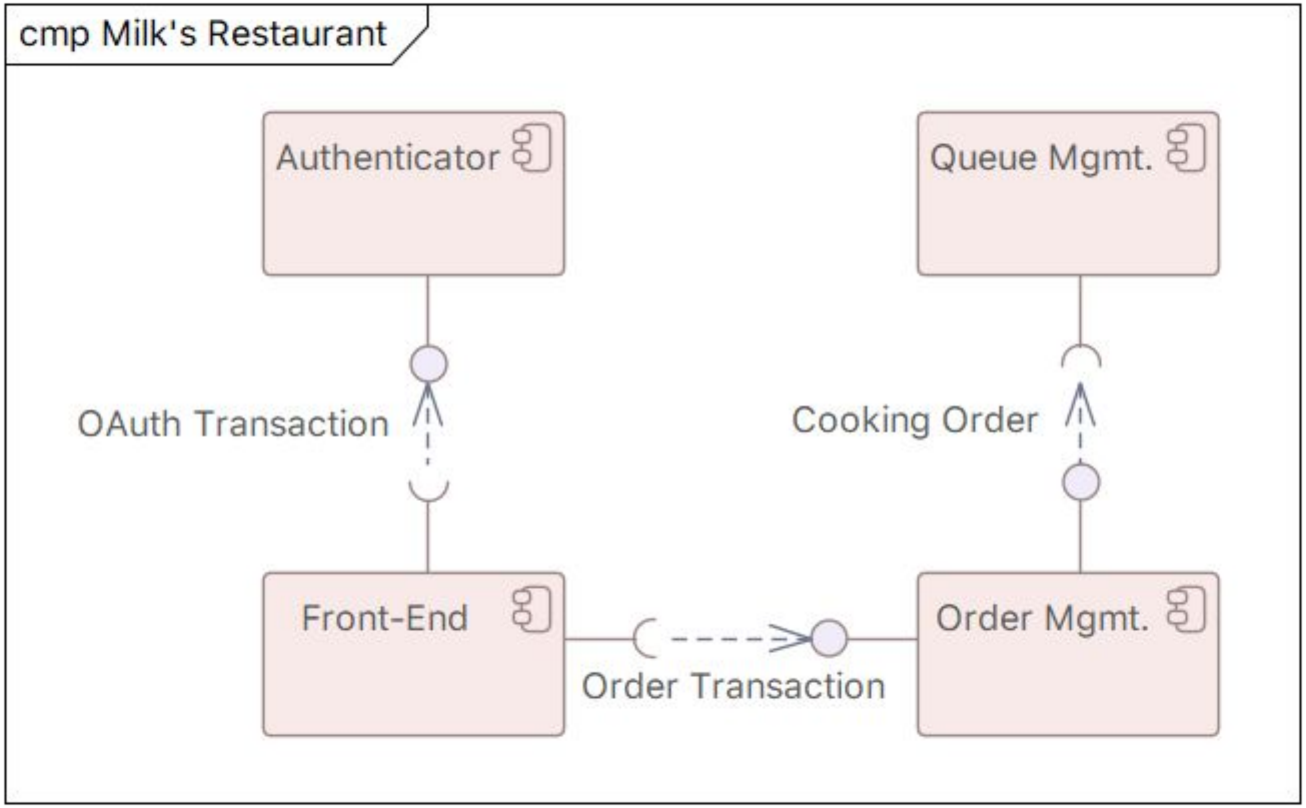
Running Example /w Design Decision

Mediator - Front-End: Stateless. No user info stored (except SessID in RAM). If down during operation, reset & login again.



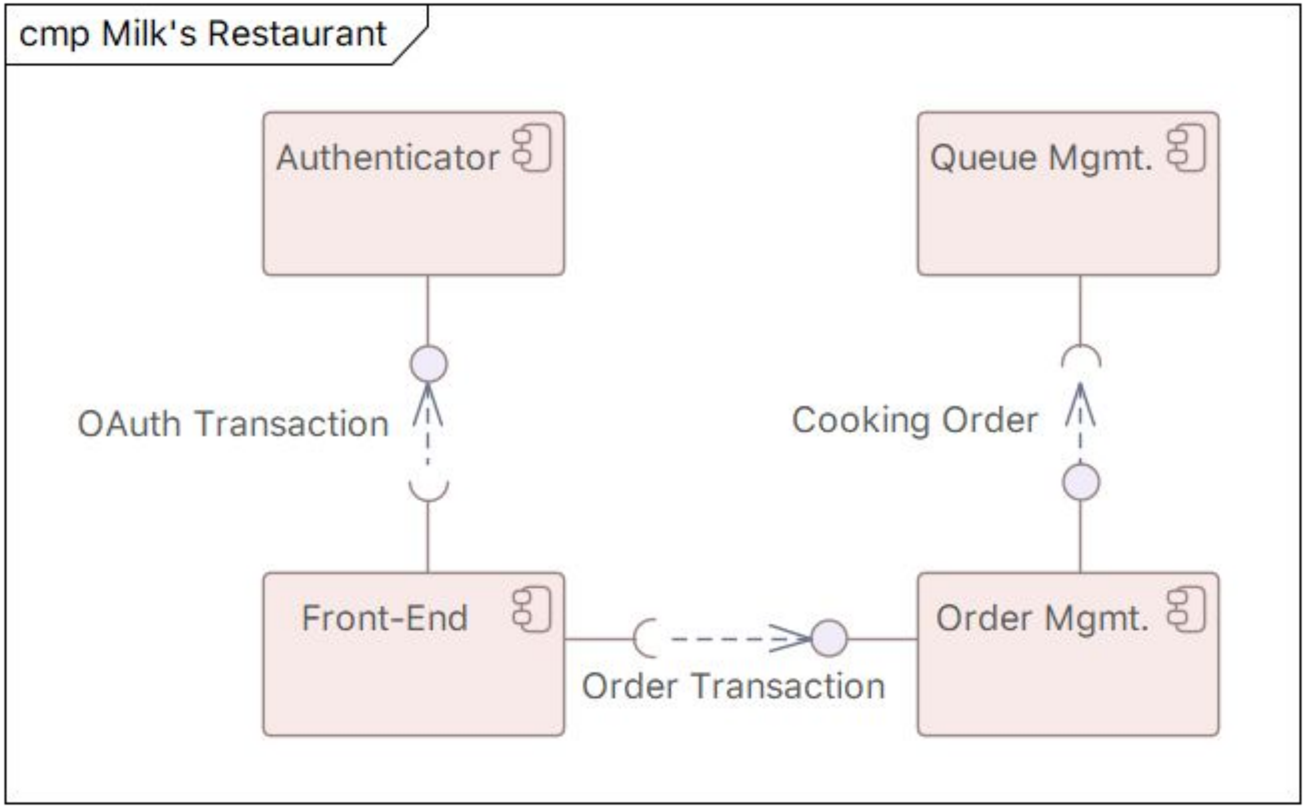
RUNNING EXAMPLE /W DESIGN DECISION

Mediator - Flood protection. Only one order can be active by an user (via SessID). Need to log out before create a new Order.



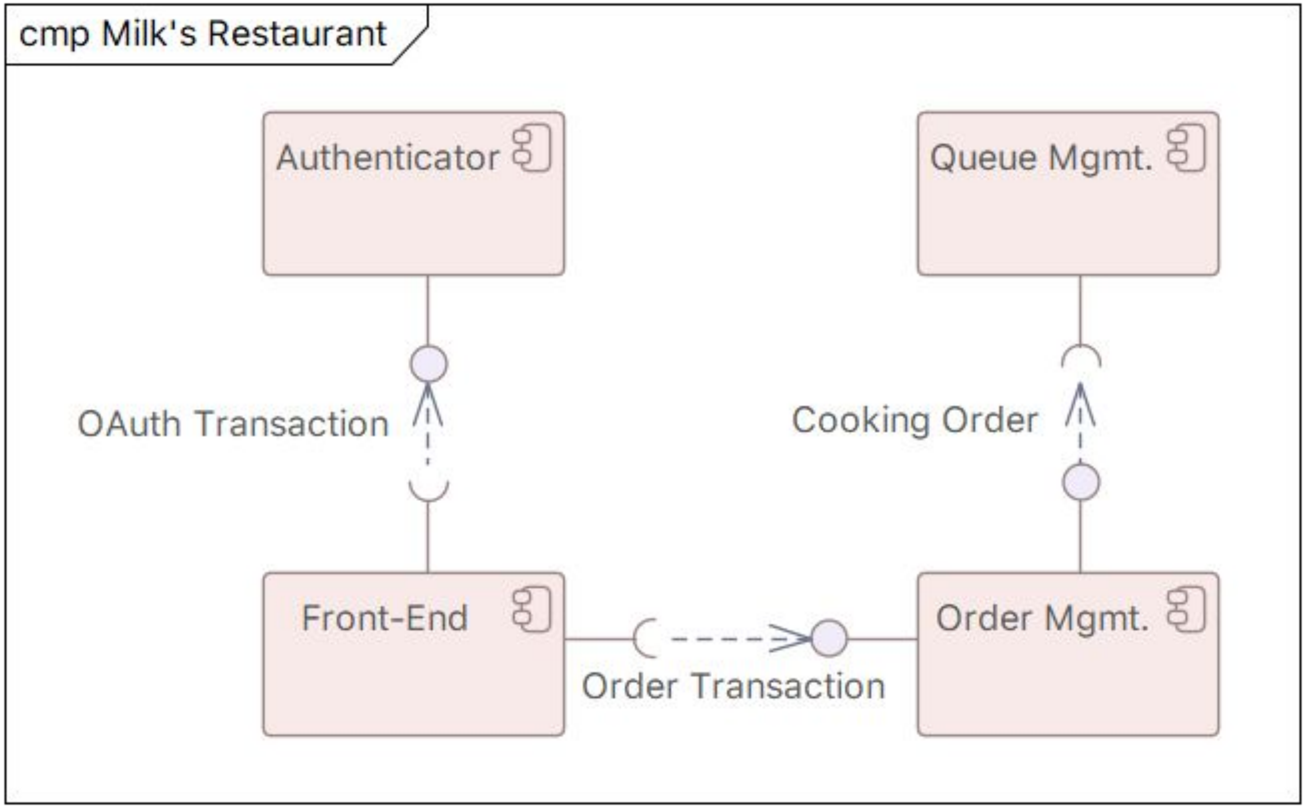
Running Example

Peer-to-Peer: Queue Mgmt. -> Order Mgmt.



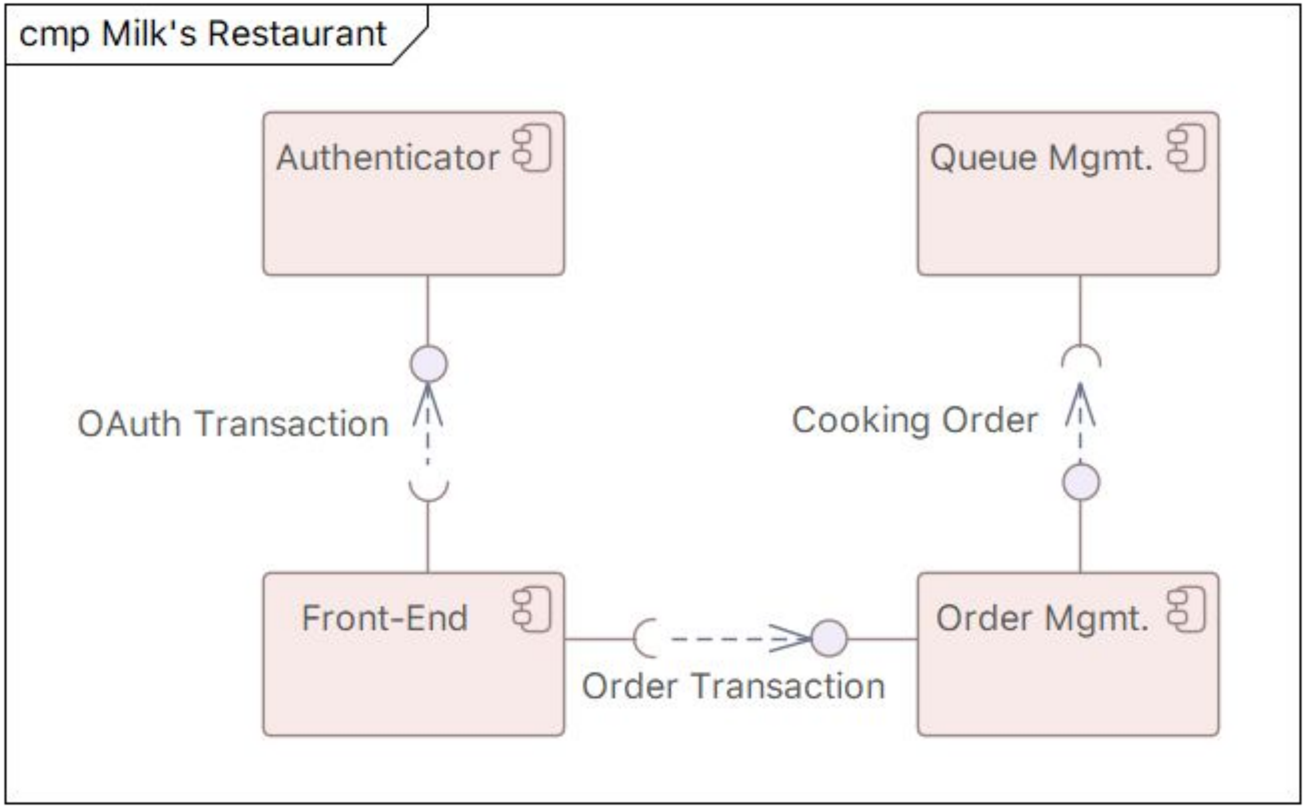
Running Example /w Analysis

Peer-to-Peer: Queue Mgmt. -> Order Mgmt.
Queue Mgmt. can flood Order Mgmt. but unlikely as only Chefs are using.



Running Example /w Design Decision

Peer-to-Peer: Queue Mgmt. -> Order Mgmt.
Each service has its own database to store Order or Queue. If they are down, data stays.



Group Exercise

16/07 - Web Service Dev & SOA