Software Design Description  
*<9Meal>*

1. Apply UML
2. Each component and connector in diagram must have name/description
3. Each diagram must have an ID
4. Ensure no spelling mistake!

# Summary

This document describes architectural design for *<9Meal>* via 4+1 View approach

The intended audiences are software development team and deployment team.

## Reference

* *<9Meal Description document>*

# Architecture Style Selection

Provide justification for choosing architecture style, relevant to FRs, NFRs known in product description

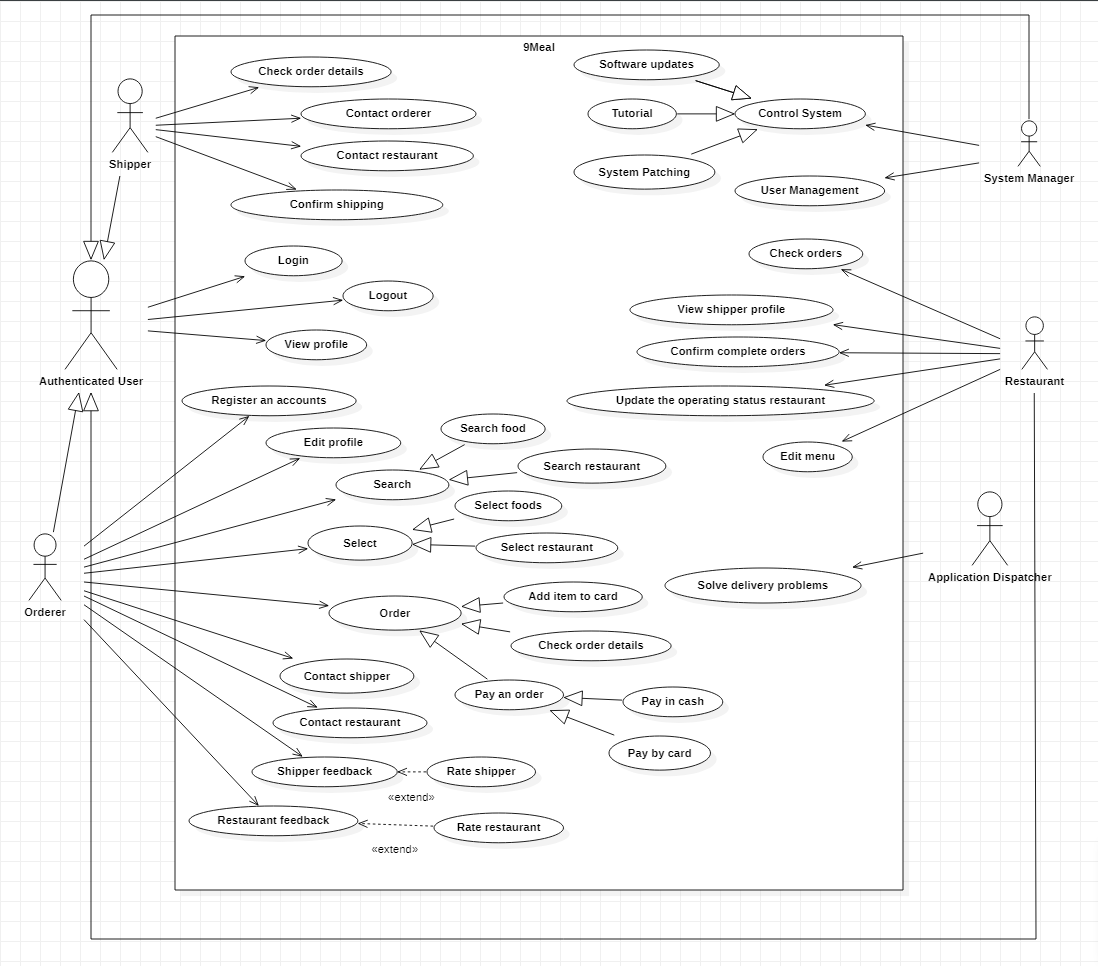
| **Selected Architecture Style** | **Justification of Selection** | **Affected FRs** | **Affected NFRs** |
| --- | --- | --- | --- |
| **Pipe-Filter** | During processing, the first component must be done before the next component. After a successful output, must is store data on DataSink (Database). | - Choose restaurant or food  - CRUD food in cart  - Check out  - Accept order |  |
| **Repository** | Because the nature of a large system is database, it will manage and retrieve data stored on the database. | - Manage accounts  - Manage user details  - Manage shipper details  - Manage restaurant details  - Manage restaurant rates and reviews |  |
|  |  |  |  |
|  |  |  |  |

# Architectural Design

## Scenario View

*[Usecase diagram and use case description of key (not all) use cases*

| **UC ID** | **Use case** | **Actors** | **Pre-condition** | **Post-condition** | **Logical Processing** |
| --- | --- | --- | --- | --- | --- |
| **UC1** | Solve delivery problems | Application Dispatcher | - Resolve defective orders (orders not delivered on time, shipper not picking up and delivering on time, ..) | Problem solved | 1.ddfs  2.ddsd |
| **UC2** | Edit Menu | Restaurant | - CRUD food list | Update Successful |  |
| **UC3** | Order | Orderer | - Select food  - Add item to card  - Pay an order | Order Successful |  |
| **UC4** | User Management | System Manager | - CRUD user list | Update Successful |  |



## Logical View

Show static diagrams: class, object

## Process View

* 1. Dynamic diagrams: state, sequence, activity, communication,…  
     Each diagram must have a note to tell which use cases are related
  2. If using implicit asynchronous communication styles, must include descriptions for each type of message/event

## Development View (optional)

Component diagrams

## Physical View

deployment diagrams: server nodes, server connections

# Design Traceability

| **Use case ID** | **Logical View** | **Process View** | **Development View** | **Physical View** |
| --- | --- | --- | --- | --- |
| **UC1** | LV2 | PV2 |  |  |
| **Uc2** | LV3, 4 | PV1 |  |  |
| **UC3** | LV1 | PV3 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |