System Architecture & Component Design

Software Engineering BCS 6D

Restaurant Management System

Prepared By:

Muhammad Ahmad 18L-0965
Zulfiqar Chaudhry 18L-1037
Muhammad Daud Mazhar 18L-0919
NUCES FAST, Lahore Campus
June 4th, 2021

Table of Contents

1.0 System Architecture:

1.1 Style Choices:

For our application, we have come up with the ***commercial and nonprofit organization*** genre for software-based systems. This choice has been made considering that our application depicts real, commercial business uses, and comprises systems that are fundamental to the operations of a business enterprise.

For our architectural style, the most dominant one is the ***Layered Architecture****.* Since our application is based on the proper synchronization between the user interface, application layer and the storage layer, this layered architecture fits the best. The application provides a strong, interactive user interface layer, with the application layer beneath, providing all the necessary operations and methods. Lastly, it is also connected to a database which falls under the storage layer for retrieving and storing employee data etc. This is an appropriate model because all these layers are abstracted from each other and can run parallel with the code.

Since our application is not very data-intensive, and only employee profiles are kept in the database, we choose not to consider our application a part of data-centric architecture. For this purpose, the application does consist of features of the ***Server-Client Architecture*** since a central server at the restaurant handles all the connected clients, mostly for database access.

Also, we make use of the ***Object-Oriented Architecture*** as all of our entities, classes and procedures depict abstraction and classification of real world objects/people. The communication between these components is achieved through message passing, which is the backbone of our entire system since it relies on communications between different entities regarding orders placed, table status, food preparation status, etc. This data is encapsulated by these components and different operations are performed for data manipulation.

1.2 Architecture Context Diagram:

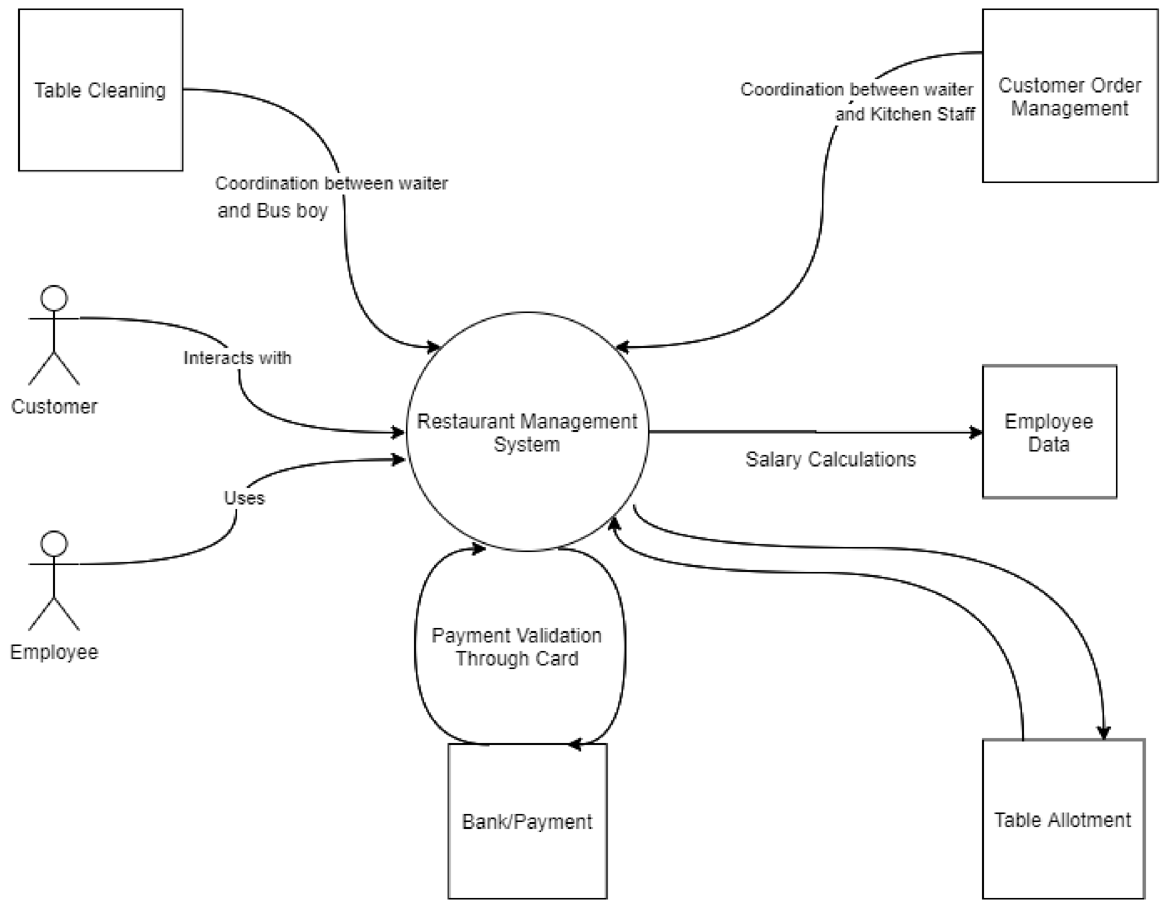
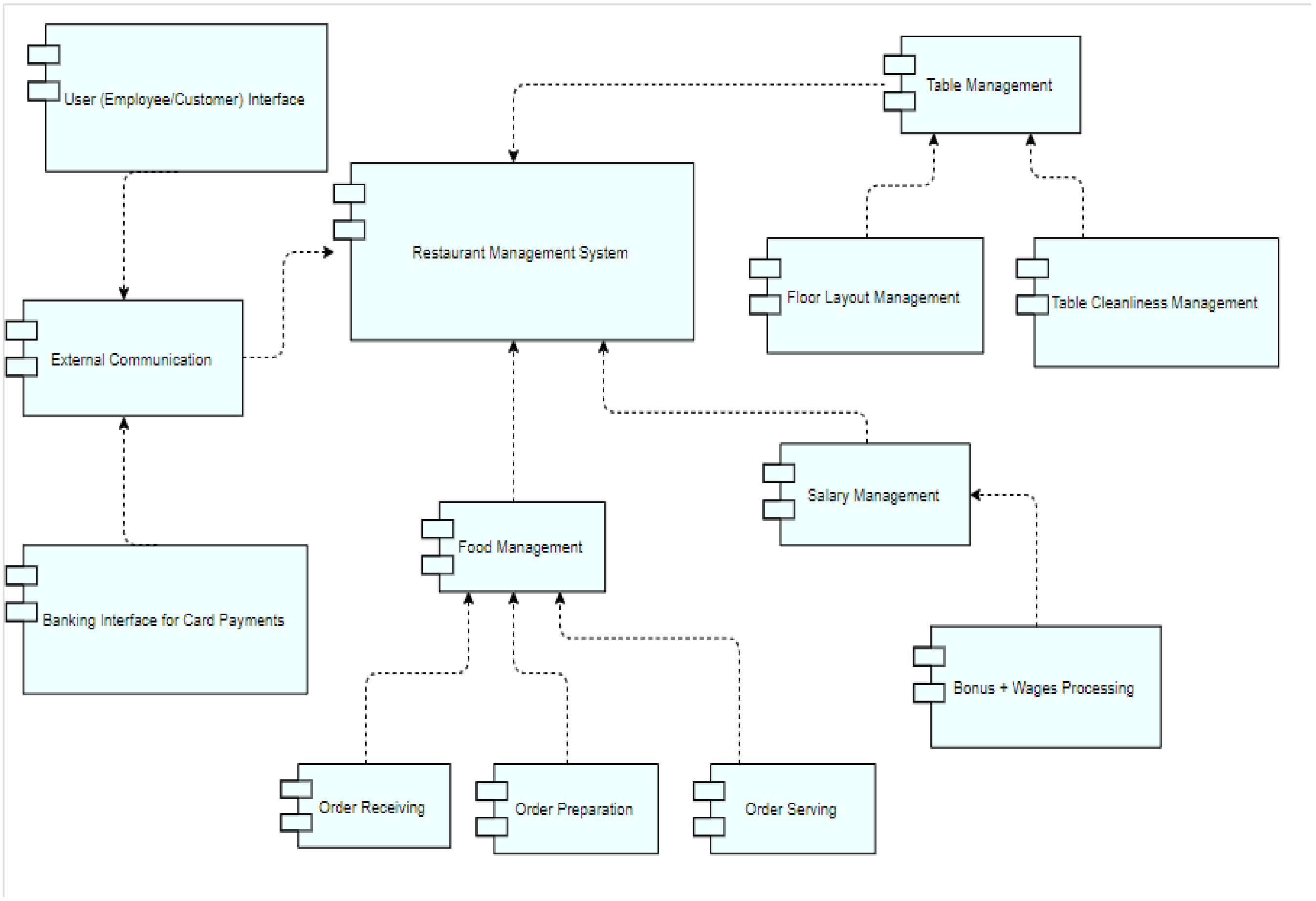
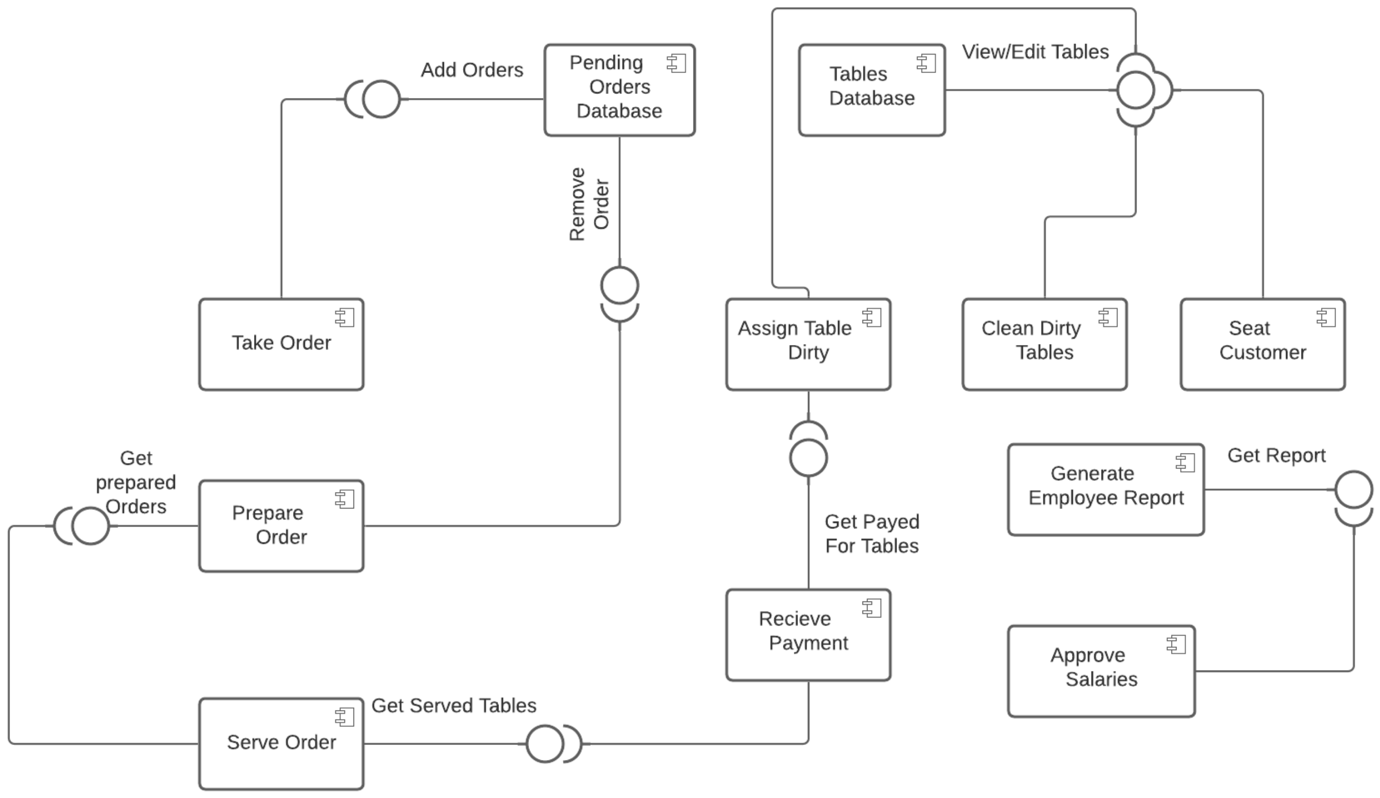


Table Cleaning
Coordination between waiter and Bus boy
Customer Order Management
Coordination between waiter and Kitchen Staff
Customer
Interacts with
Restaurant Management System
Employee Data
Salary Calculations
Uses
Employee
Payment Validation Through Card
Bank/Payment
Table Allotment

1.3 Overall Architecture Structure:



2.0 Component Level Diagram:



Design Patterns Used:

Single Responsibility Principle:

Every component has just a single responsibility and only a single reason to change. **Take Order** just takes the order, **Clean Dirty Tables** only cleans the table etc.

Interface Segregation Principle:

A class only has access to the functions it will use. Furthermore, a class only needs to implement the functions it will use. For example, take a look at **Pending Orders Database**, instead of providing just one interface to add and remove order it provides two separate interfaces. As a result **Take Order** is only aware of and only has access to the **Add Order** functionality because it is the only functionality it needs. Similarly, **Prepare Order** only has access to the **Remove Order** functions of the database. Lastly, no subclass will need to implement extra functions that it will not use due to the division of interfaces.

Liskow’s Substitution Principle:

No subclass is stricter than the parent class and hence Liskow’s Substitution Principle is not violated.

Dependency Inversion/Injection principle:

Every single dependency is on interfaces rather than on implementation. As a result, there are no bad dependencies and any component can be easily switched out without affecting the whole system.