Đại học Bách Khoa – ĐHQG TP Hồ Chí Minh
Khoa Khoa học và Kỹ thuật Máy tính

POINT OF SALE RESTAURANT 2.0

Task 3: Architecture Design

Lecturer:

Lê Đình Thuận

Member list:

Lê Nguyễn Minh Hiếu – 1913332

Lê Xuân Nghĩa – 1914310

Nguyễn Ngọc Nguyên – 1914384

Nguyễn Trọng Phúc – 1914704

Lê Thanh Sang – 1914914

Hồ Ngọc Thành – 1915133

Phạm Hoàng Trường – 1915745

Ho Chi Minh City, October, 2021

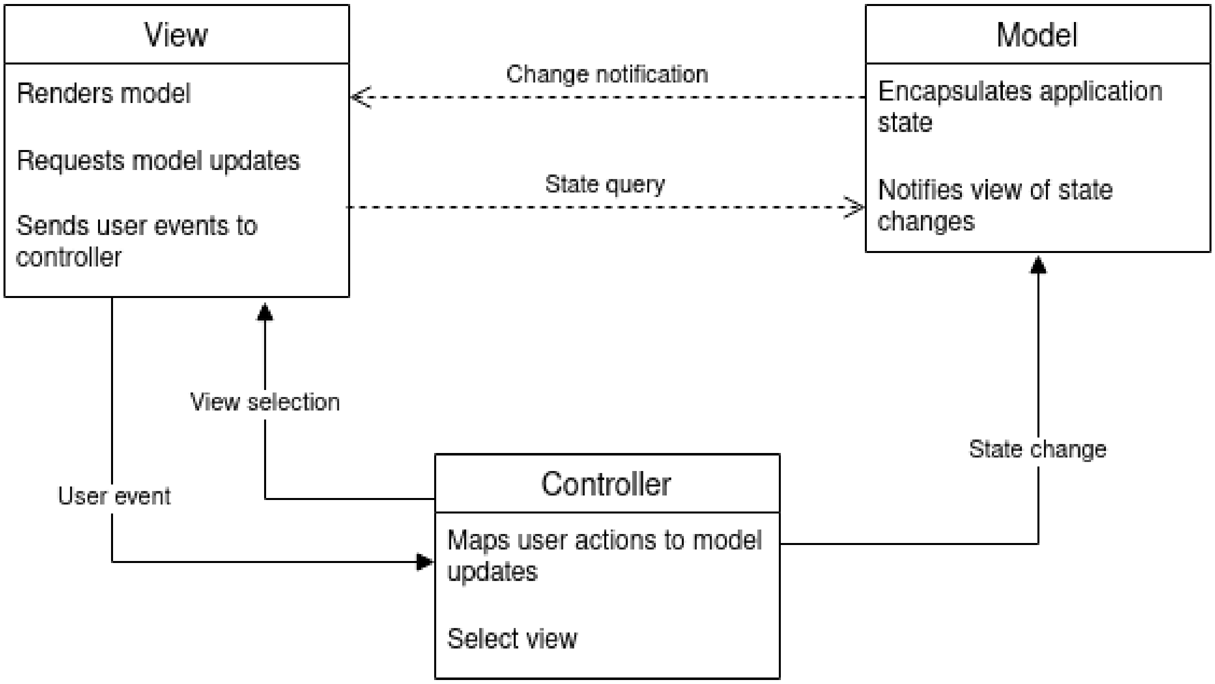
Contents

1. Describe an architectural approach:

We use Model - View - Controller (MVC) Pattern to implement the designed system.

The main reason for applying this architecture design is that MVC doesn't allow us to repeat ourselves and it helps to create a solid structure of our web application.

Description:
The system is structures into three logical components that interact with each other as image below:



1. *Model*: Known as the lowest level which is responsible for managing the system data and associating operations on that data. It handles data logically so it basically deals with data. The model is actually connected to the database so anything you do with data. Adding or retrieving data is done in the model component. Its response to the controller requests because the controller never talks to the database by itself. The model talks to the database back and forth and then it gives the needed data to the controller.

2. *View*: Data representation is defined and managed by the view component. It actually generates UI or user interface for the user. So in this web application when you think of the view component just think of the HTML/CSS part. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller, so the view only contacts the controller.

3. *Controller*: It enables the interconnection between the View and Model so it acts as an intermediary as well as manages user interaction such as: key press, mouse clicks,... The controller does not have to worry about handling data logic, it just tells the model what to do. After receiving data from the Model it processes them, then the Controller takes all that information and sends it to the View with an explanation of how to represent that data to the users.

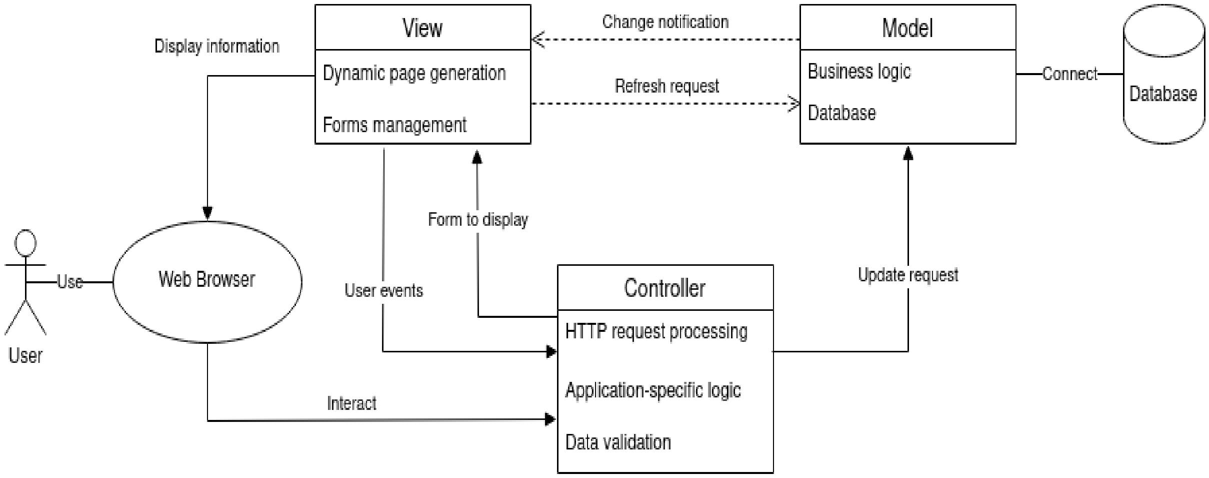
Advantages:

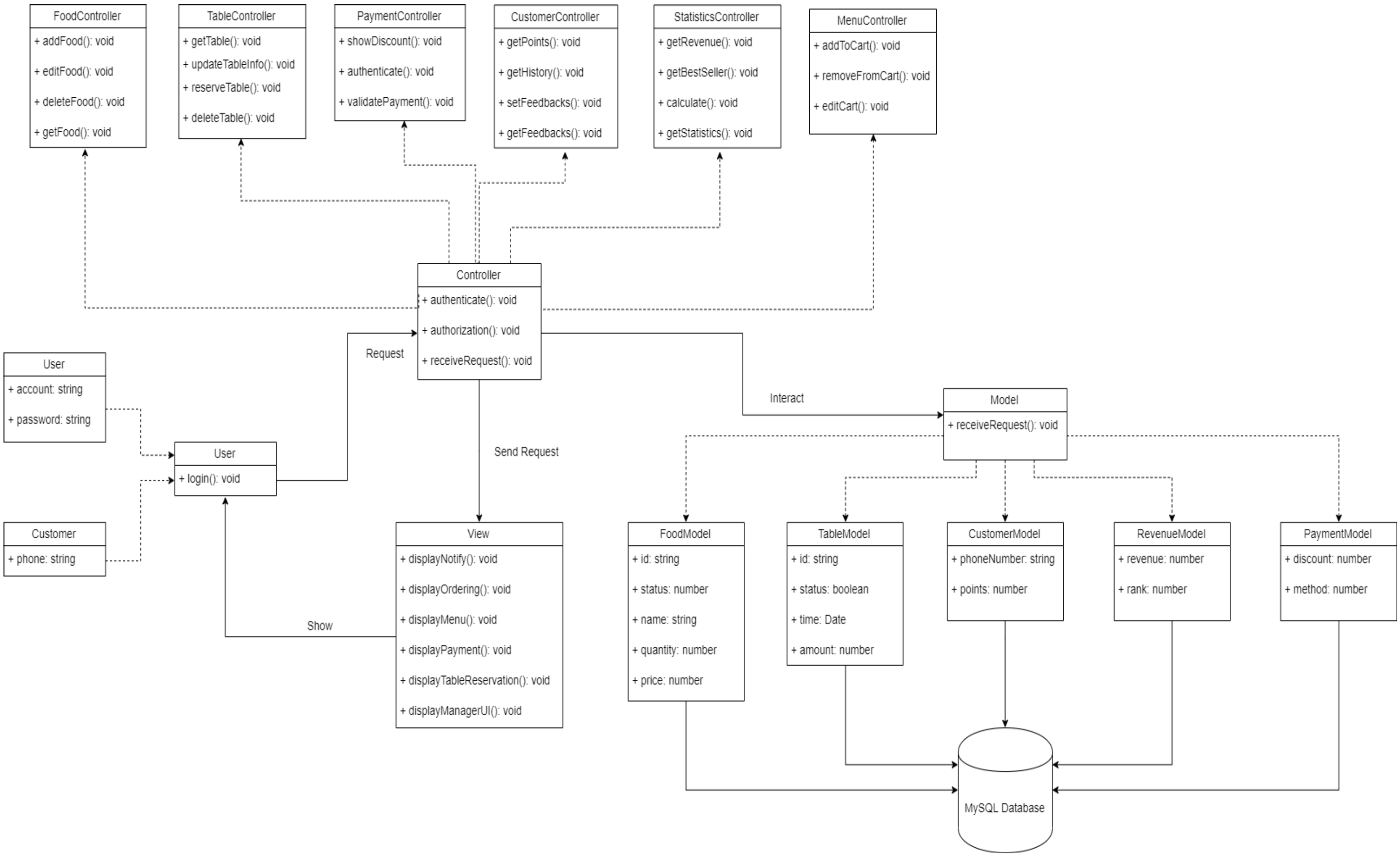
- Easy code maintenance which is easy to extend and grow
- MVC Model component can be tested separately from the user
- Easier support for new types of clients
- Development of the various components can be performed parallelly.
- It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
- It only uses a Front Controller pattern which processes web application requests through a single controller.
- Offers the best support for test-driven development
- It works well for Web apps which are supported by large teams of web designers and developers.
- Provides clean separation of concerns(SoC).
- Search Engine Optimization (SEO) Friendly.
- All classes and objects are independent of each other so that you can test them separately.
- MVC design pattern allows logical grouping of related actions on a controller together.

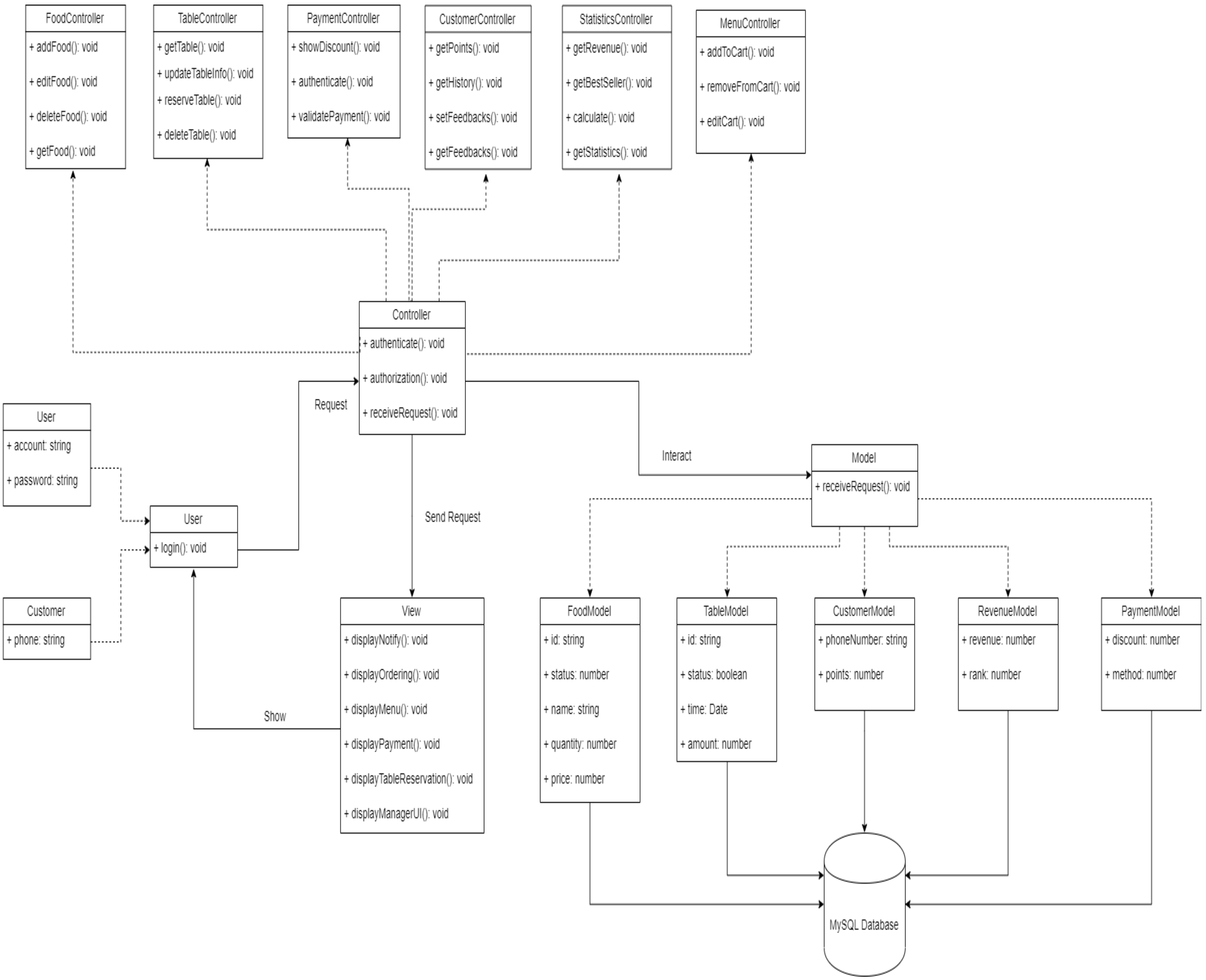
Disadvantages:

- Difficult to read, change, unit test, and reuse this model
- The framework navigation can sometimes be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
- No formal validation support
- Increased complexity and Inefficiency of data
- The difficulty of using MVC with the modern user interface
- There is a need for multiple programmers to conduct parallel programming.
- Knowledge of multiple technologies is required.
- Maintenance of lots of codes in Controller

Applied to our web application:



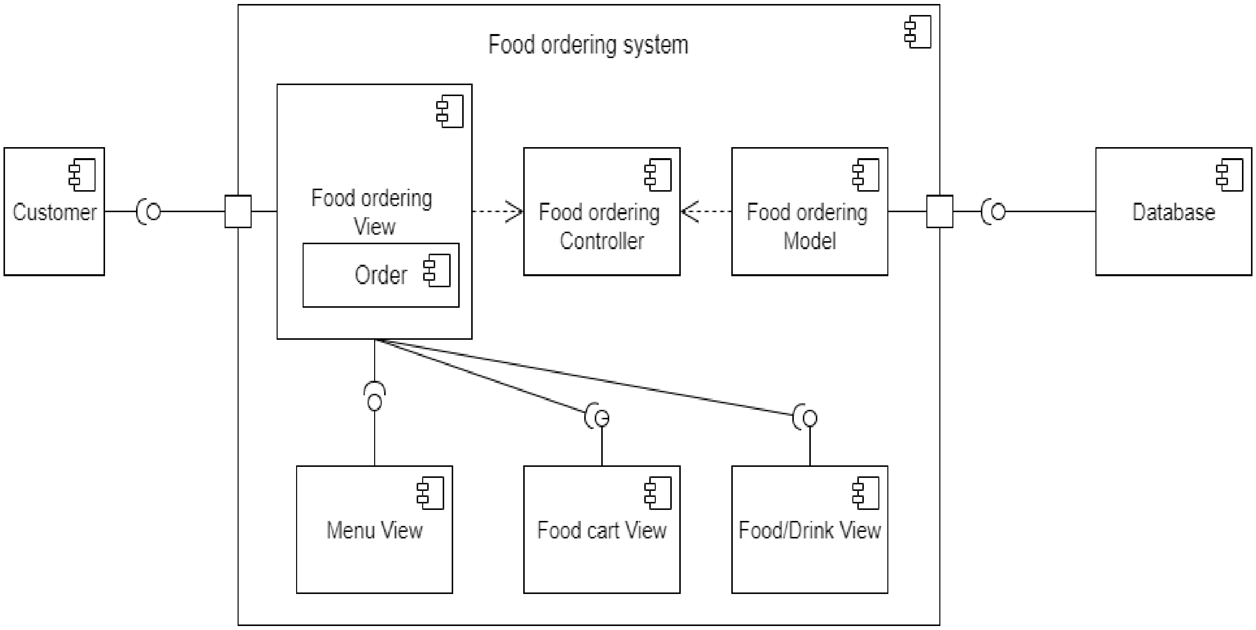



FoodController
+ addFood(): void
+ editFood(): void
+ deleteFood(): void
+ getFood(): void
TableController
+ getTable(): void
+ updateTableInfo(): void
+ reserveTable(): void
+ deleteTable(): void
PaymentController
+ showDiscount(): void
+ authenticate(): void
+ validatePayment(): void
CustomerController
+ getPoints(): void
+ getHistory(): void
+ setFeedbacks(): void
+ getFeedbacks(): void
StatisticsController
+ getRevenue(): void
+ getBestSeller(): void
+ calculate(): void
+ getStatistics(): void
MenuController
+ addToCart(): void
+ removeFromCart(): void
+ editCart(): void
Controller
+ authenticate(): void
+ authorization(): void
+ receiveRequest(): void
Request
User
+ account: string
+ password: string
User
+ login(): void
Customer
+ phone: string
Interact
Model
+ receiveRequest(): void
Send Request
View
+ displayNotify(): void
+ displayOrdering(): void
+ displayMenu(): void
+ displayPayment(): void
+ displayTableReservation(): void
+ displayManagerUI(): void
Show
FoodModel
+ id: string
+ status: number
+ name: string
+ quantity: number
+ price: number
TableModel
+ id: string
+ status: boolean
+ time: Date
+ amount: number
CustomerModel
+ phoneNumber: string
+ points: number
RevenueModel
+ revenue: number
+ rank: number
PaymentModel
+ discount: number
+ method: number
MySQL Database

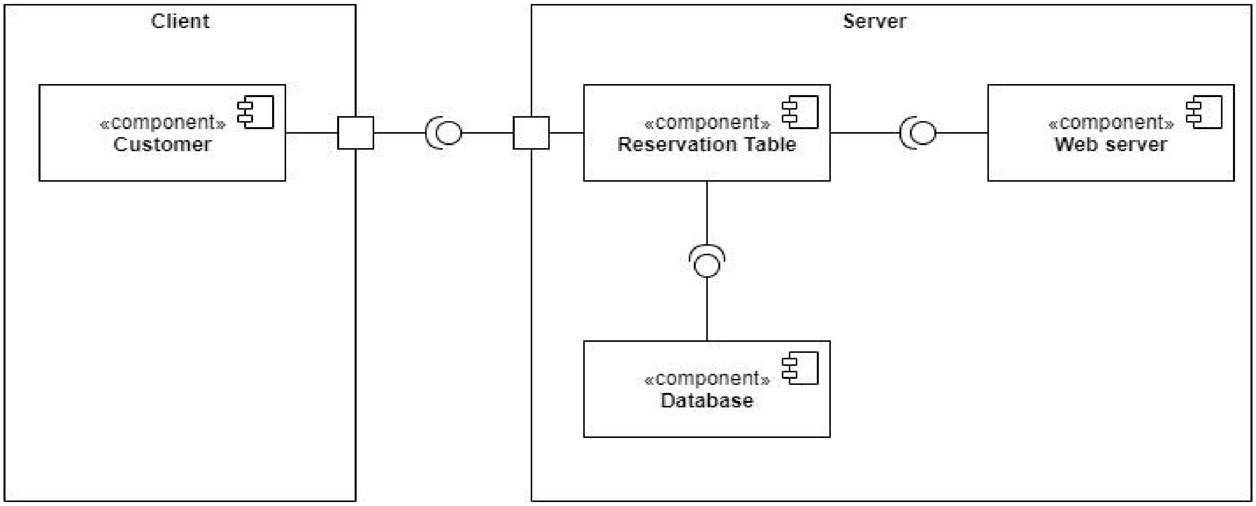
2. Draw implementation diagrams

2.1. Component diagrams

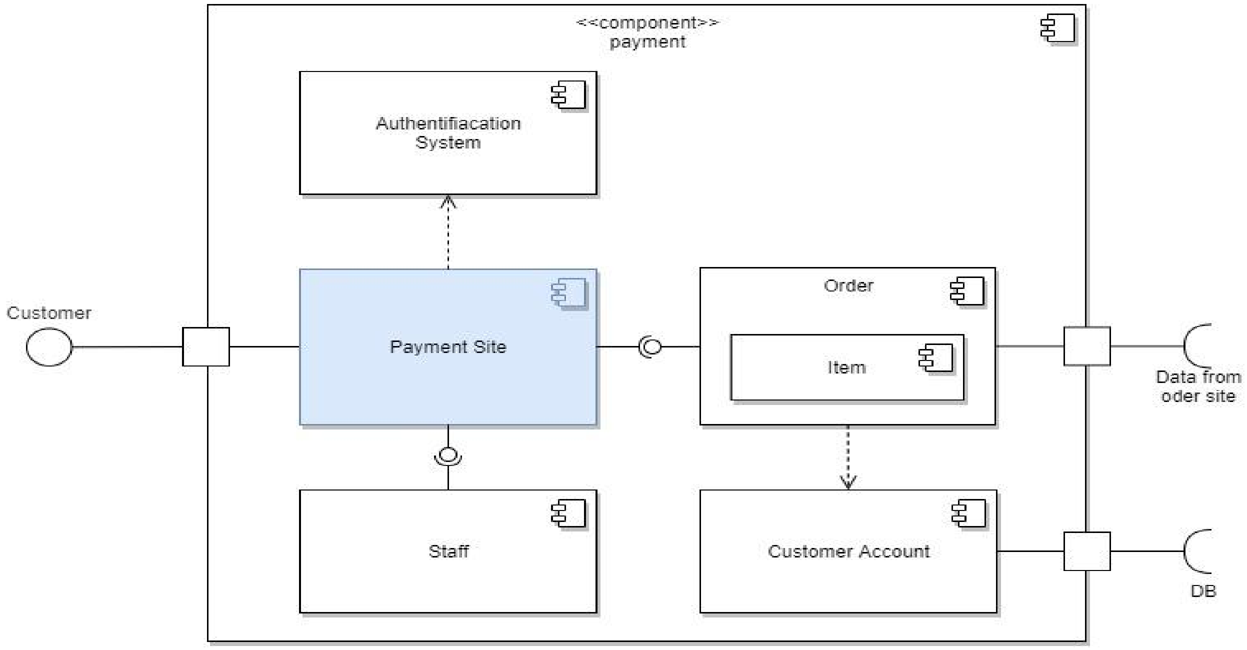
2.1.1. Food Ordering



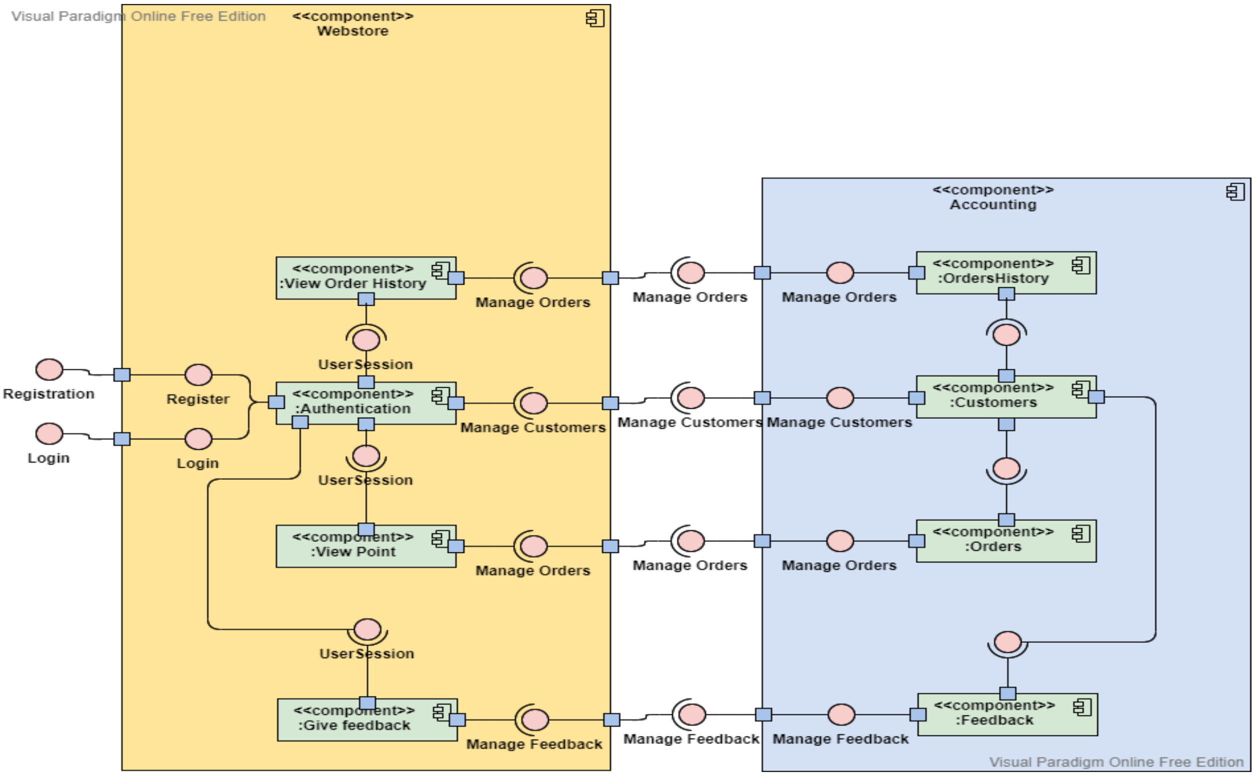
2.1.2. Table reservation



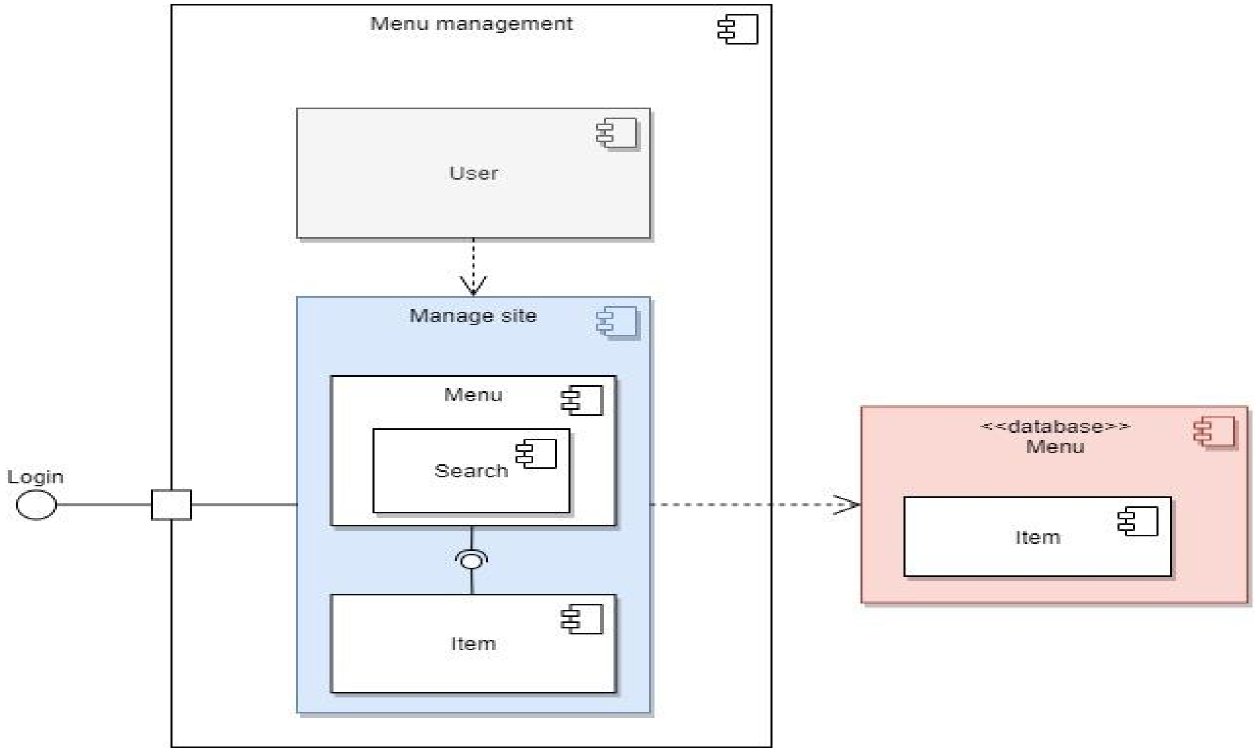
2.1.3. Payment System



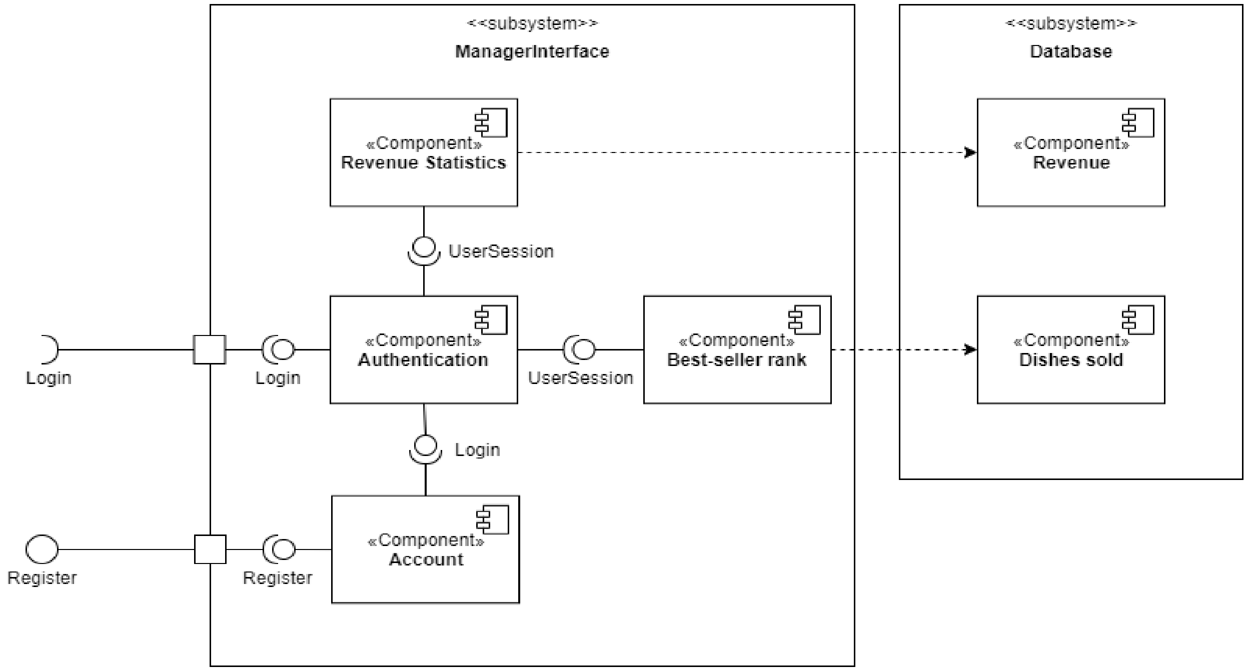
2.1.4. Customer management



2.1.5. Menu management

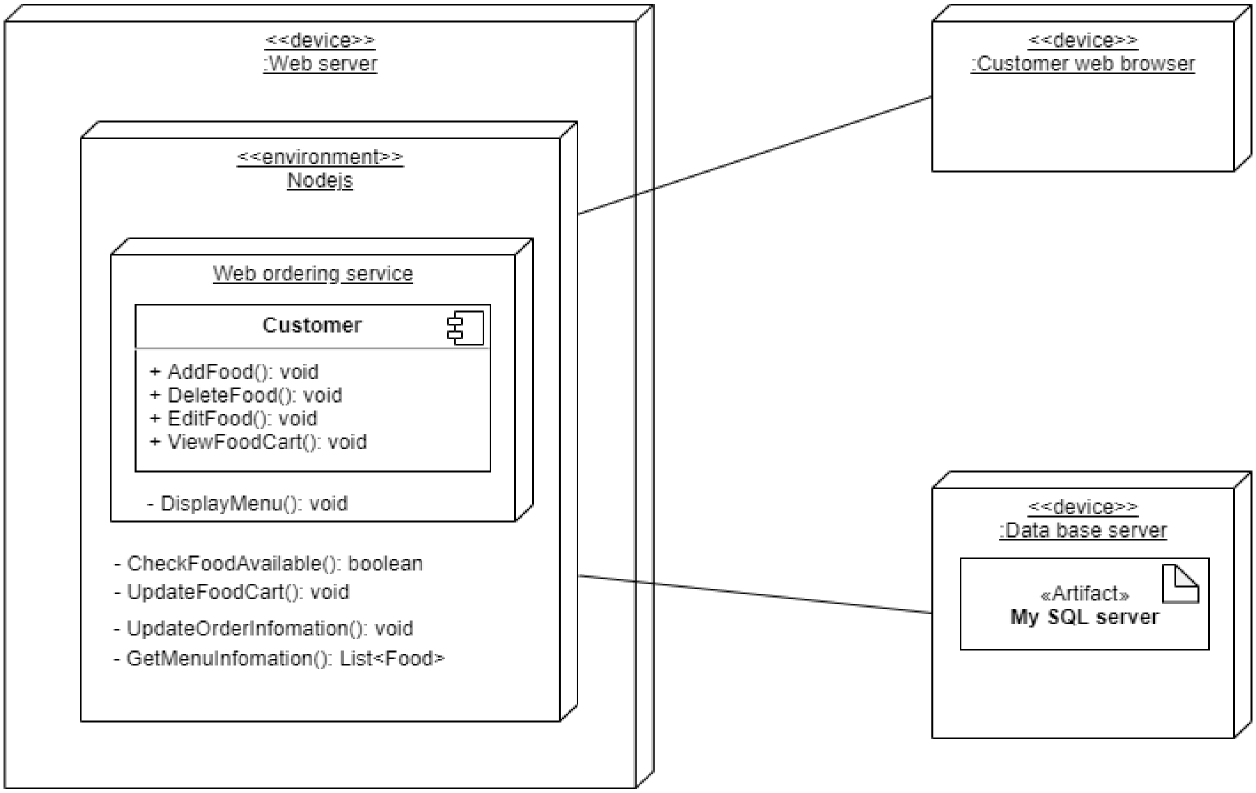


2.1.6. Statistic Revenue

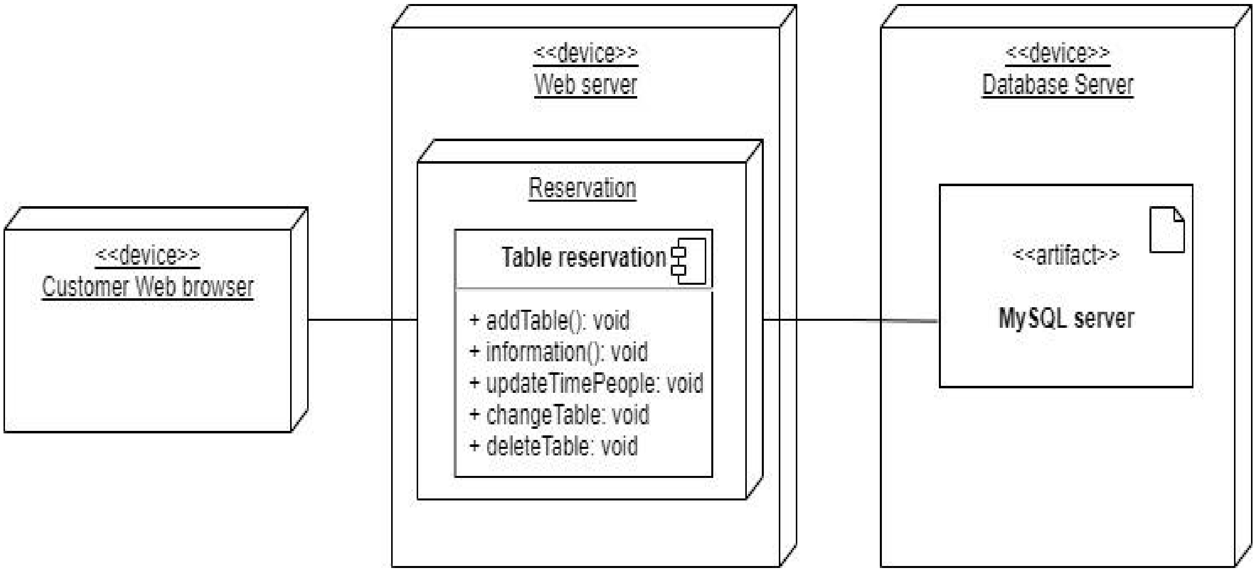


2.2. Draw deployment diagrams

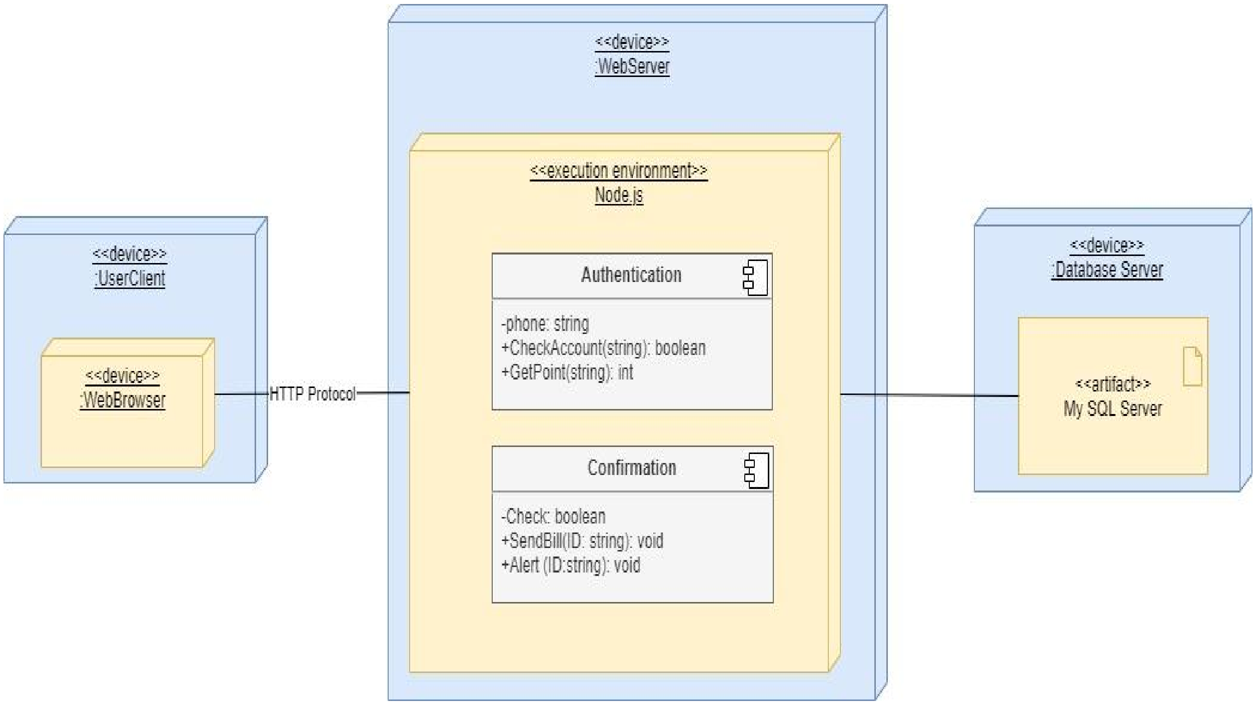
2.2.1. Food Ordering



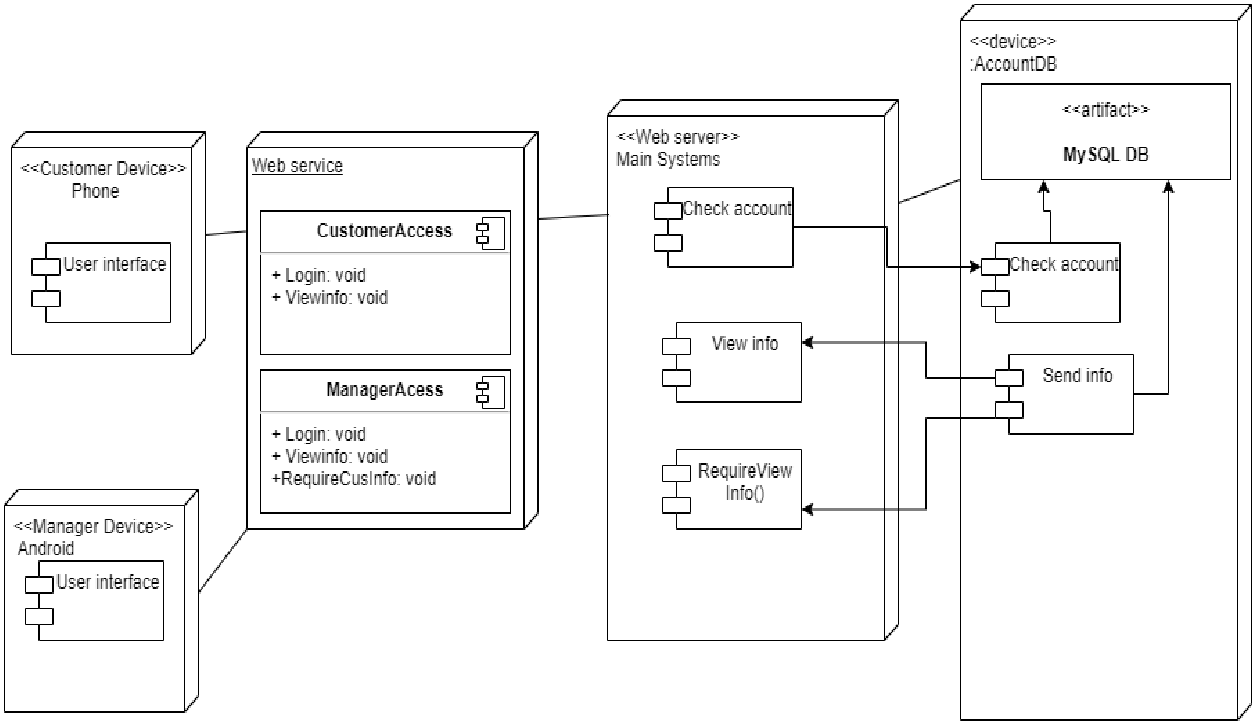
2.2.2. Table reservation



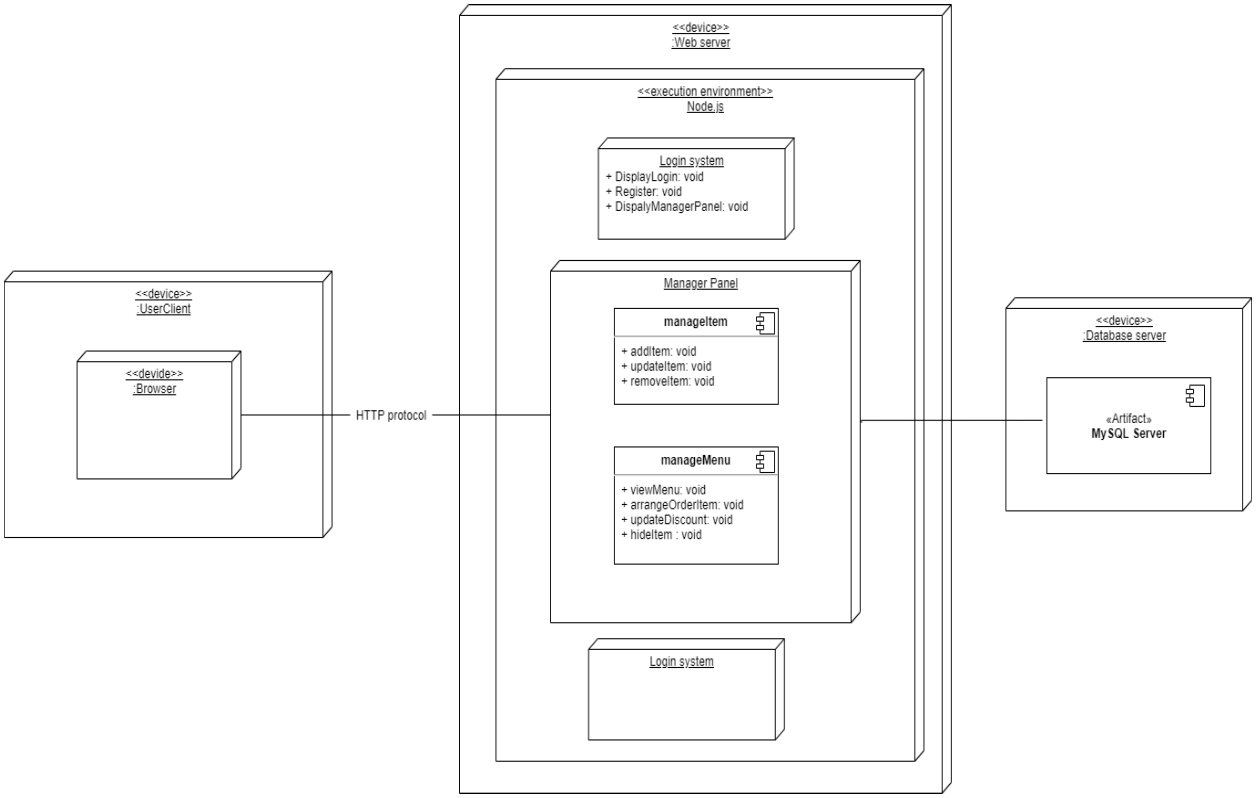
2.2.3. Payment System



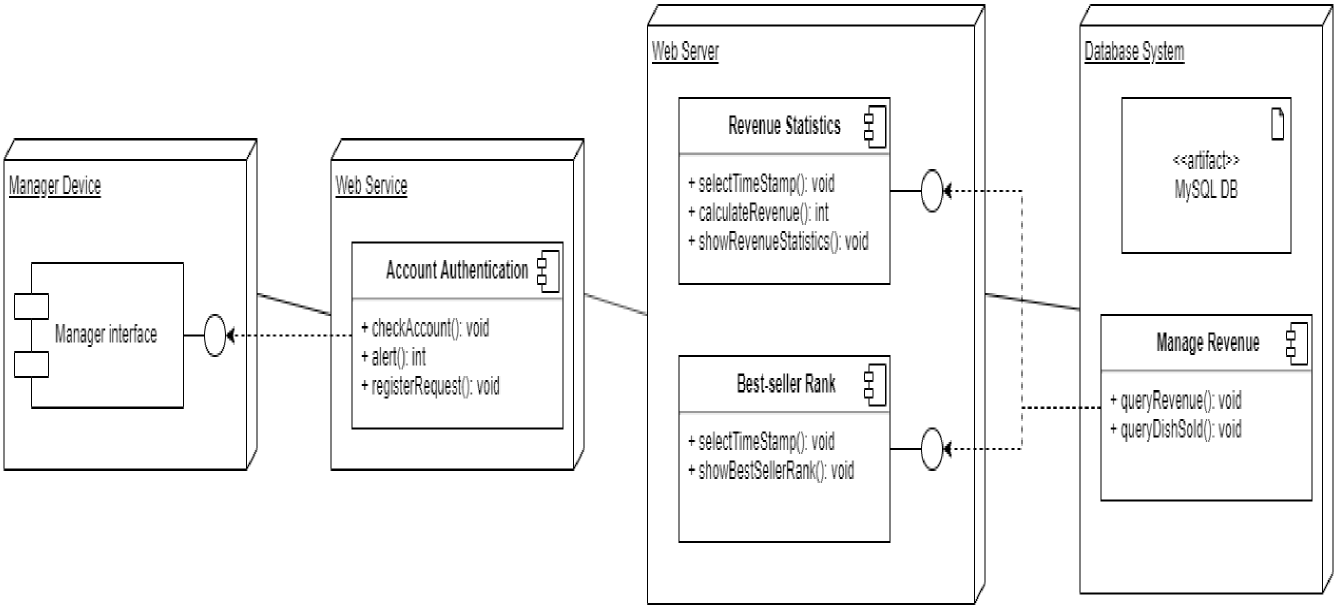
2.2.4. Customer management



2.2.5. Menu management



2.2.6. Statistic Revenue



3. Work assignment

Name	ID	Work
Lê Nguyễn Minh Hiếu	1913332	1 - 2.1.6 - 2.2.6
Lê Xuân Nghĩa	1914310	2.1.2 - 2.2.2 - meeting minutes
Nguyễn Ngọc Nguyên	1914384	2.1.4 - 2.2.4
Nguyễn Trọng Phúc	1914704	2.1.3 - 2.2.3
Lê Thanh Sang	1914914	2.1.1 - 2.2.1
Hồ Ngọc Thành	1915133	1 - report
Phạm Hoàng Trường	1915745	2.1.5 - 2.2.5