Q1:

* Class name must be PascalCase
* Attributes must be private to ensure the encapsulation in OOP
* Add getter, setter for each attributes
* The private variables must follow camelCase or like this \_employeeName
* It should have a no agrs constructor
* The method must follow PascalCase

1. Fixed:

class Employee

{

private string \_employeeName;

private int \_age;

private double \_salary;

public Employee() {}

public Employee(string employeeName, int age, double salary)

{

\_employeeName = employeeName;

\_age = age;

\_salary = salary;

}

public string EmployeeName {

get { return this.\_employeeName; }

set { this.\_employeeName = value; }

}

public int Age

{

get { return this.\_age; }

set { this.\_age = value; }

}

public double Salary

{

get { return this.\_salary; }

set { this.\_salary = value; }

}

public void DisplayInfo()

{

Console.WriteLine("Employee: " + \_employeeName + ", Age: " + \_age + ", Salary: $" + \_salary);

}

}

Q3:

1. Frontend:

* Web app (ReactJS/Angular/VueJS): The website for customers can browse and booking hotel, contains all UX/UI of the application, and website for hotel administrators can manage their hotel.
* Mobile app (React Native/Flutter): It like the web app but it release as a mobile application, which is users can download on their smartphone

1. API Gateway (Nginx/Spring Cloud Gateway):

* When clients or managers request something in my app, it routes their requests to backend services

1. Backend Services (Microservice) (Spring Boot/NodeJS/Express.js/FireBase/JWT/…):

* Contains all services that the application can do for users such as booking, management, authentication & authorization,… They are main logics that the application is operating.

1. Load Balancer (HAProxy/Nginx/Kubernetes Ingress):

* This ensures system scalability and distributes request efficiently. It solve then problem that when there are too much users access to the server.

1. Firewall (DMZ):

* Secures the system from external threats. Everything want to access to the application server must pass this one first.

1. Database (PostgreQL/MongoDB/SQL Server/MySQL):

* It stores everything from entities or object of the application. The application can save and use those data from database.

1. Monitoring Server (Prometheus/Grafana):

* It tracks system performance and health, manage the whole system. Anything happens in the application, it can track it.

1. Registry Server (Consul/Eureka):

* Manages the whole services in backend, it remember all the service so it can route to it easily.

**System architecture diagram:**

