SEPP CW2

SE part

By Binbin Zhan s1945599
Louis(Chang) Yu s1802871

3.1.1

In my view, the client-server architecture would be a good choice for web-based and mobile software products. To be more specifically, I will select service-oriented architecture (a type of client-server) for the food box delivery system. Here are my reasons.

Firstly, in client-server architecture, clients send requests to servers, which process these requests and provide a response, client responsible for user interaction, based on the data moving between it and the server. For food box delivery system, shielding individuals send their request to the system and wait for a response, this perfectly match the performance of client-server architecture.

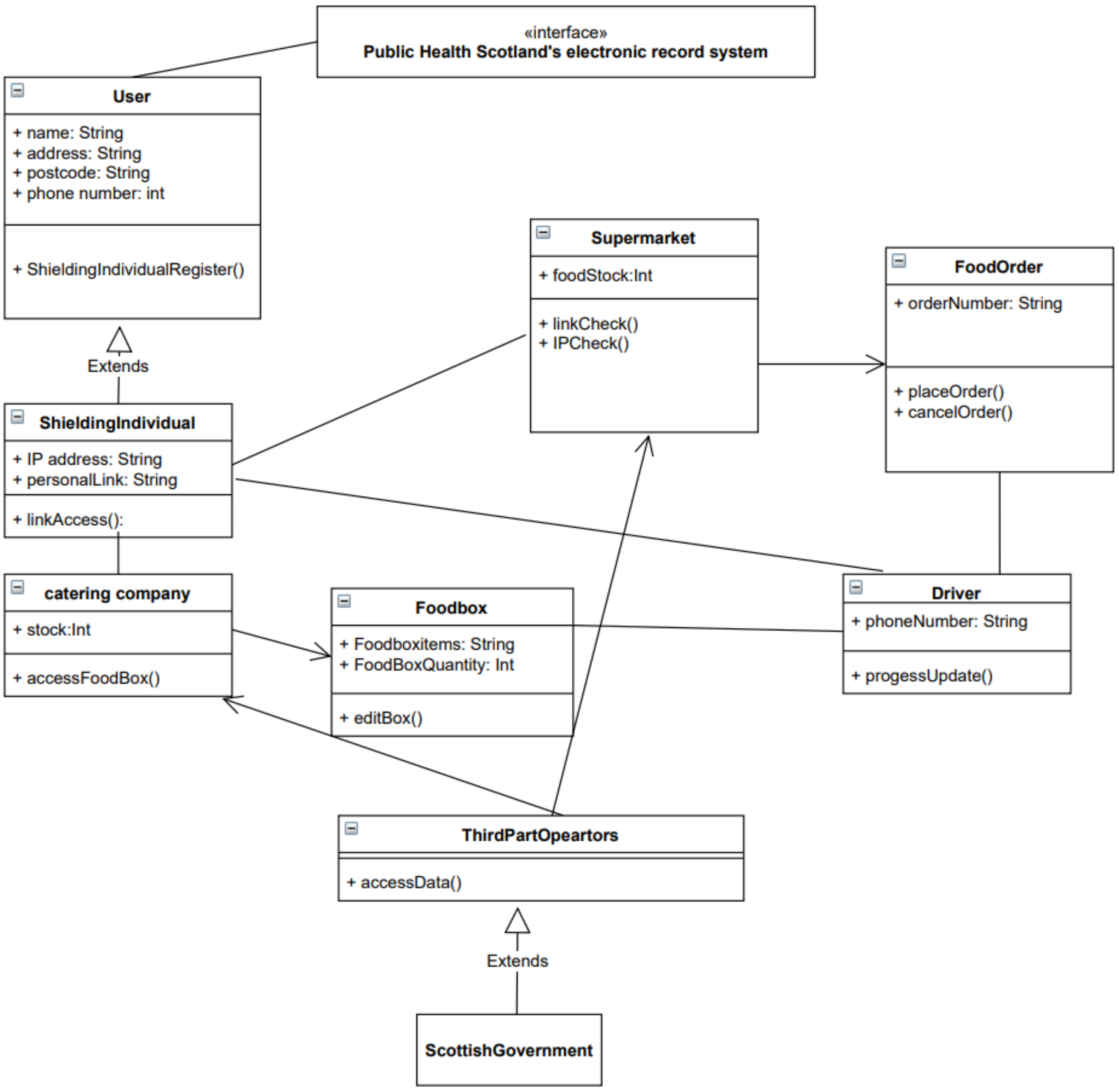
Secondly, in client-server architecture, now clients are computers or mobile devices with large processing power so significant processing on clients. There are several servers like web and database. These leads the speed of the system will be much quicker and more efficiency in resource allocation.

Thirdly, load balancer distributes requests to servers, this ensures no single server bears too much demand, by spreading the work evenly, load balancing improves application responsiveness, it also increases availability of applications and websites for users.

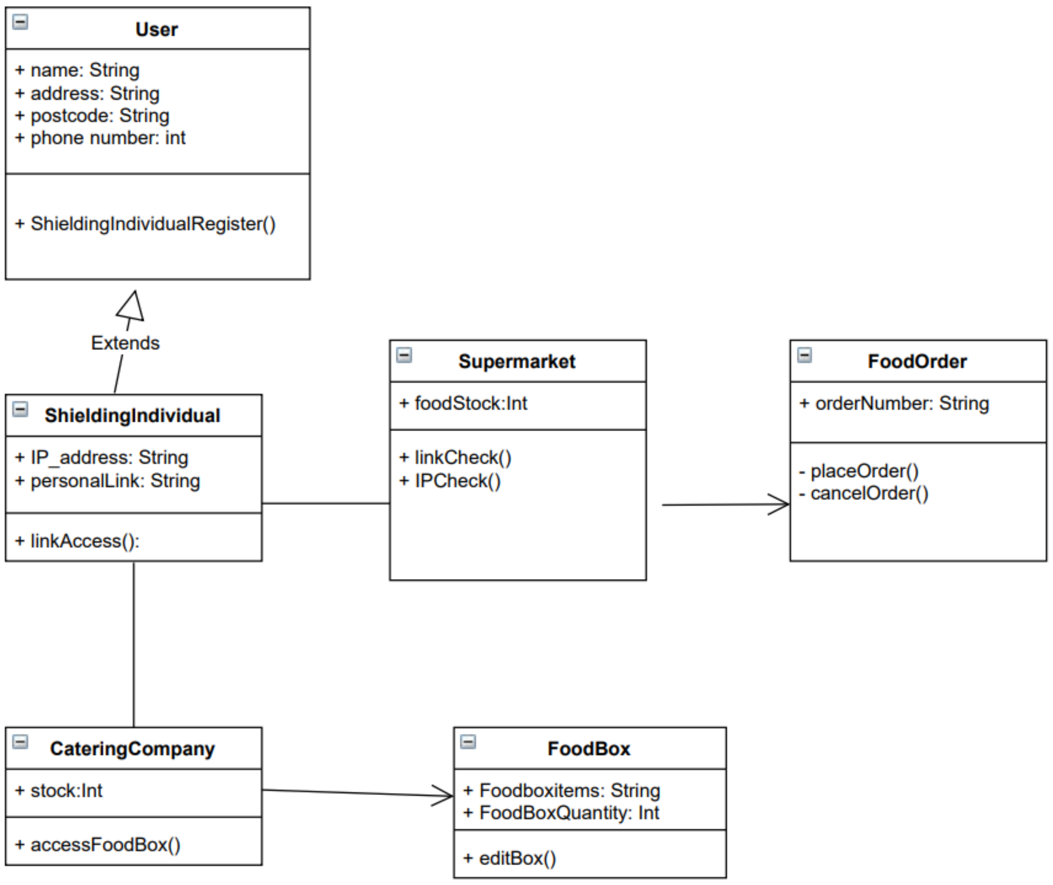
As for why I prefer service-oriented architecture, this is because the description mentioned that the system might extend in the future in order to operate more other uses and interact with many other systems. These requirements can be satisfied under service-oriented architecture because this type of system is with many servers providing services, and services are stateless, so can be replicated, distributed, migrated between servers. It is also good if system components need to change regularly, need for scalability (matched extend requirement) and resilience to failure.

3.1.2 UML class model

UML class model for whole system



UML class model for 5 use case in the description

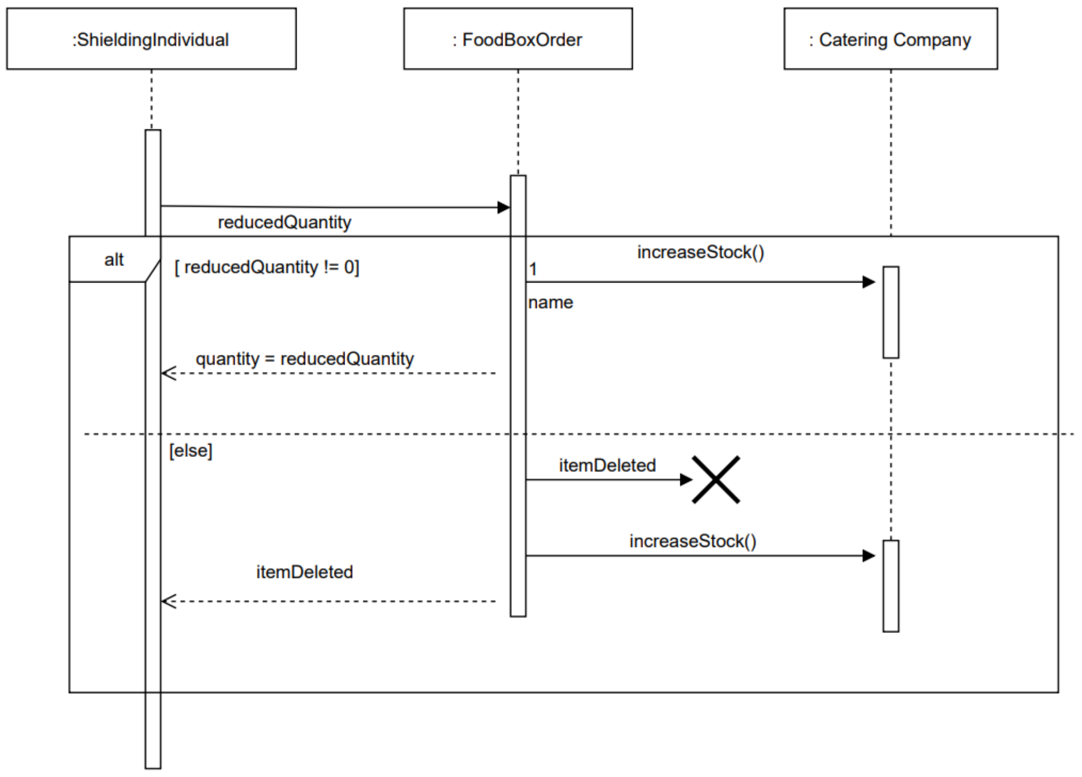


3.1.3 High-level description of the UML class model

the chosen design was Template Method Pattern, if Singleton Pattern has been used, method can be overwritten, it is unsafe for the system which has users personal information and IP address and CHI number, for the safety reason, Template Method pattern was chosen, there might lead to duplicated functionality, or maintainability cost, but system is more safe.

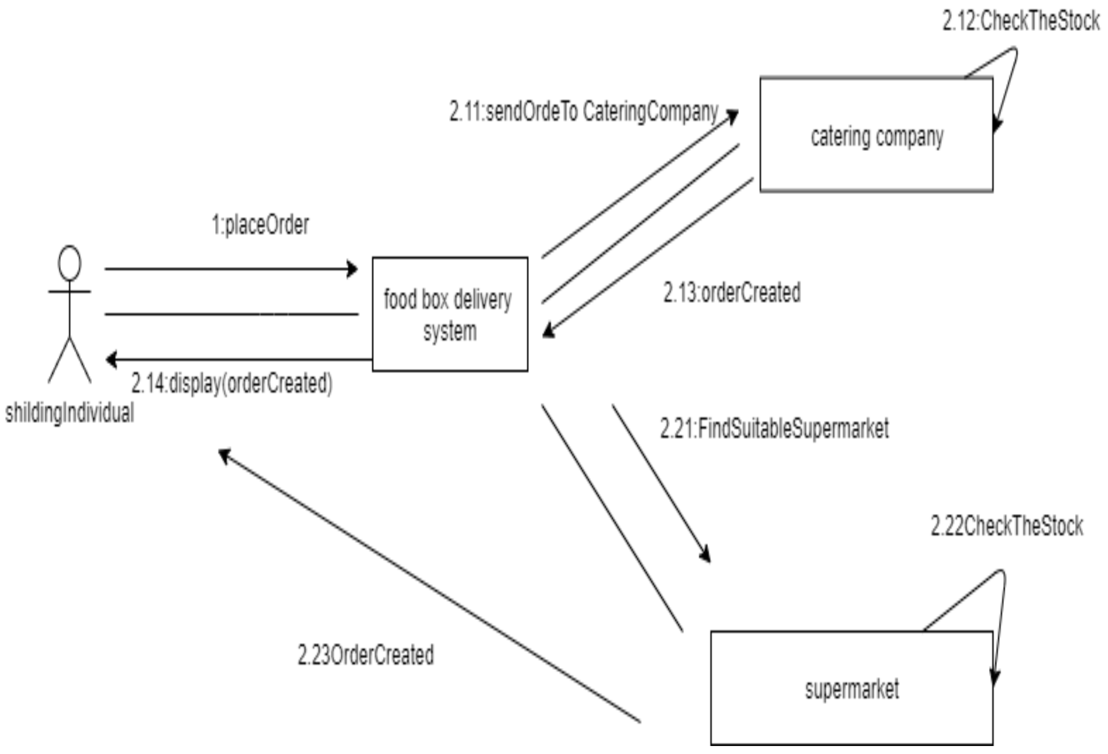
3.1.4 UML sequence diagram

UML sequence diagram for the Edit Food Box use case

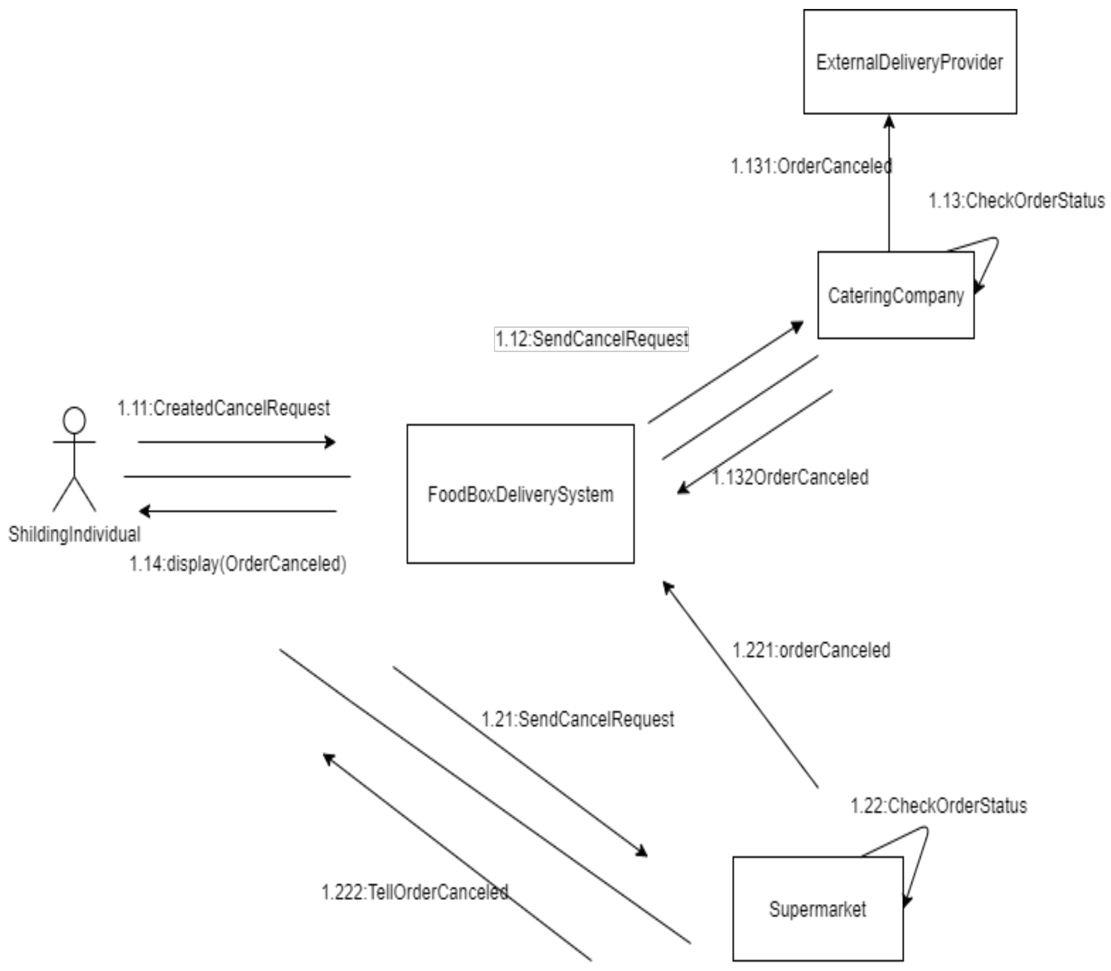


3.1.5

Place order:



Cancel order:



3.1.6

The concept of use case diagram in cw1 is not consistent to the UML class model in cw2, we missed several of the use cases and not linked them to the correct primary actors.

Also updated non-functional requirements descriptions in cw1

Add one more use case ‘update order status’.

3.2.1

Overall, we are doing a plan-driven process in our project. The design is a separate stage in the software development. Approach in which our food box delivery system's design is completed and perfected before starting the implementation.

Thorough documentation is kept on the design. We also formally using modelling and notation. Architectural and detailed design carried out and other features all matched this type.

3.2.2
In agile software development process, the design interleaved with requirements and implementation in each iteration and focusing on most important unfinished features. There is no formal, detailed, design documents in agile processes. Outcomes of agile design may not be specification documents but reflected later in the code. Models may be informally used to facilitate team communication.

3.3.1: First of all, I made sure the general type of food box delivery system is client-server architecture and specialized the type into service-oriented architecture according to the coursework description about the further use plan of the system.

3.3.2: I used rectangles to represent class, inside each class has method and attributes. Arrow and lines were used to represent constraint of access

3.3.3: The static design in class model is built for object-oriented design, and it is easy for the further implementation in Java language.

3.3.4: I used a subclass shielding individual under the user class, to show the user need to register to become valid shielding individual and then choosing the food order from catering company or food box from supermarket.

3.3.5: There is one option before actually reduced quantity. If the quantity after reduced is not zero, the information will send to catering company, and reduce the quantity, otherwise the item will be deleted.

3.3.6 Since the sequence diagram can easy be complimented in object-oriented language java, we may draw the conclusion that it has reasonable interactions quality.

3.3.7 I created two communication diagrams for the place order and cancel order, they contained basic stakeholders (shielding individuals, system, supermarket, catering companies, external delivery service providers) and explained the successful condition when these two functions operated.

3.3.8 I noticed the order of the operation and the name of each stage, generally I followed the basic requirements about good object-oriented design.

3.3.9 After finishing the work of UML model in cw2, we realized that the model is conflicted to the use case diagram in cw1. So we drew a new use case diagram which follow the concept of UML model.

3.3.10 We deleted some ideas about the system issued in cw1 since they seem to be useless after we reconsider the function of the system need to reach and for the consistency of cw1 and cw2.

3.3.11 The design would be significantly different when we consider using different software development processes which generally divided into plan-driven process or agile process.

3.3.12 Take every question really serious and do not just simply repeat previous answer. Explained the shortages and things we need to improve.