Task 1: Requirement elicitation

1.1Identify the context of this project. Who are relevant stakeholders? What are expected to be done? What are the scope of the project?

**Type of program:** Web-based POS

**Stakeholders:**

* Users
* Restaurant employees
* Owners
* Clerks
* Kitchen
* Bank
* Owners
* Web admins

**Requirements:**

* Non-direct contact between clerks and customers
* Use Web technologies and QR code
* Must work on mobile, tablet and desktop.
* Must be extendable to multiple restaurants in the future (for now only 1 restaurant)
* Holds for 300 transactions per day

**Scope/Goal:**

* Increase business intelligence
* Reduce wasted efforts
* Opportunity to scale to a large business

1.2 Describe all functional and non-functional requirements of the desired system. Draw a use-case diagram for the whole system

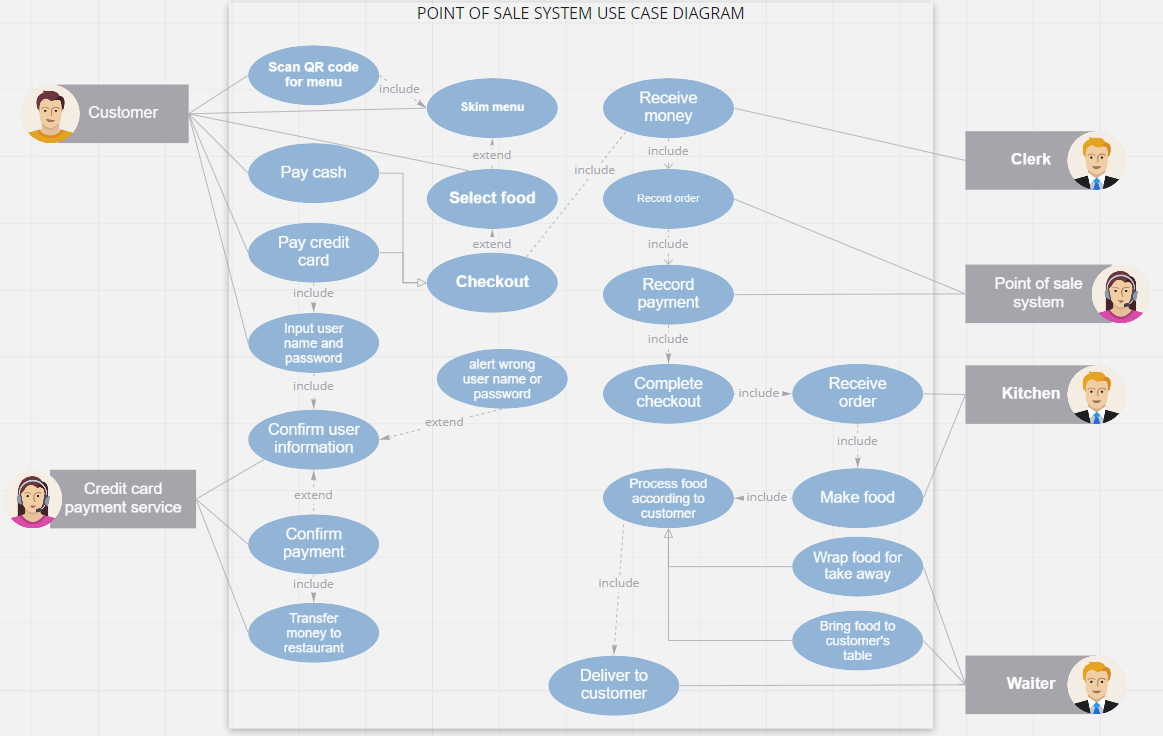
**Functional requirements:**

* Help customers place orders: select, change or delete meals (in case of mistakes).
* Retrieve and confirm the meal and price ordered by customers.
* Handle payment by credit card: provide customers with the payment page after order.
* Print receipt.
* Provide customers with feedback options.
* Record transaction after receiving payment from customers.
* Manage the users: handle login and logout.
* Customers can scan QR code for the website/menu of the restaurant. -Customers can search, order things of the restaurant.
* Customer can choose the method they want to pay.
* The system can help the customer to check their order.
* The system can calculate the money the customers have to pay, confirm payment, and transfer money to the restaurant.
* Support take-away options.

**Non-functional requirements:**

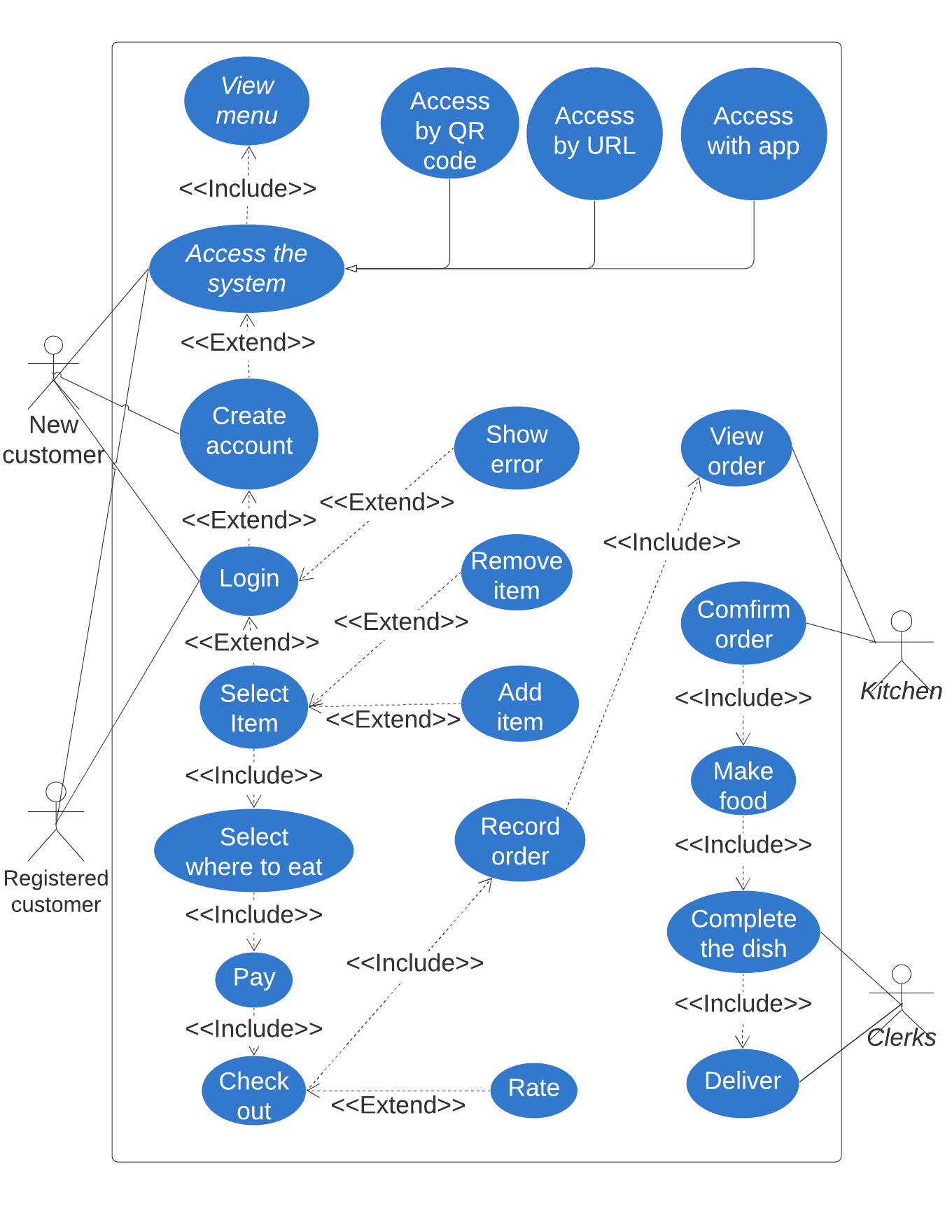
* Allow non-direct contact between Clerks and Customers.
* Not required to install apps.
* Usable from a mobile device, a tablet device or a normal computer/ laptop.
* Extendable to use in multiple restaurants in the future.
* Current transactions: about 300 orders per day.
* Being able to thrive and work during the coronavirus pandemic.
* Each function must respond within 1.0 second.
* Secure the payment.
* The system should be maintainable when there is any error occur.
* The system should be extendable to use in multiple restaurants in the future.

Use-case diagram for the whole system:



1.3: Choose one specific feature, i.e. food ordering, table reservation, customer management. Draw its use-case diagram and describe the use-case using a table format

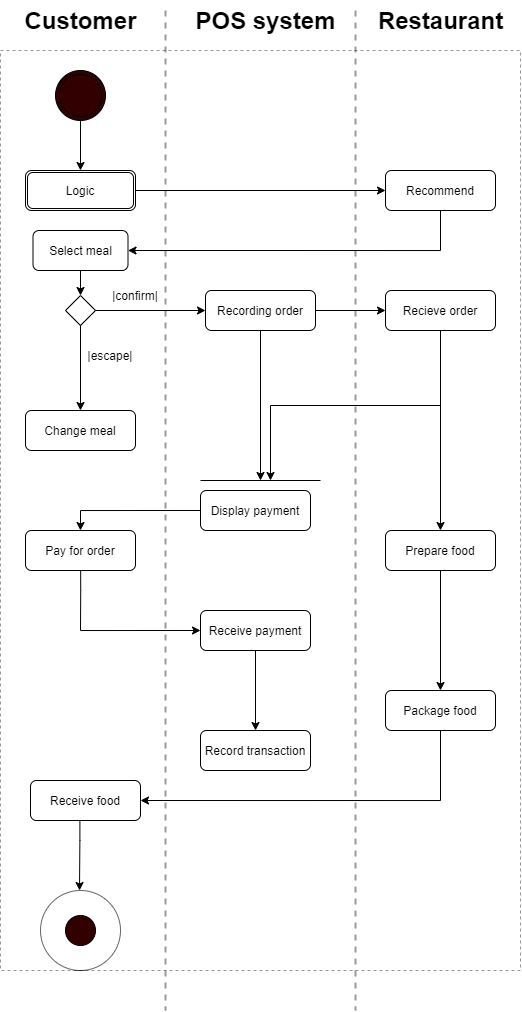
* Use-case diagram for the food ordering feature:

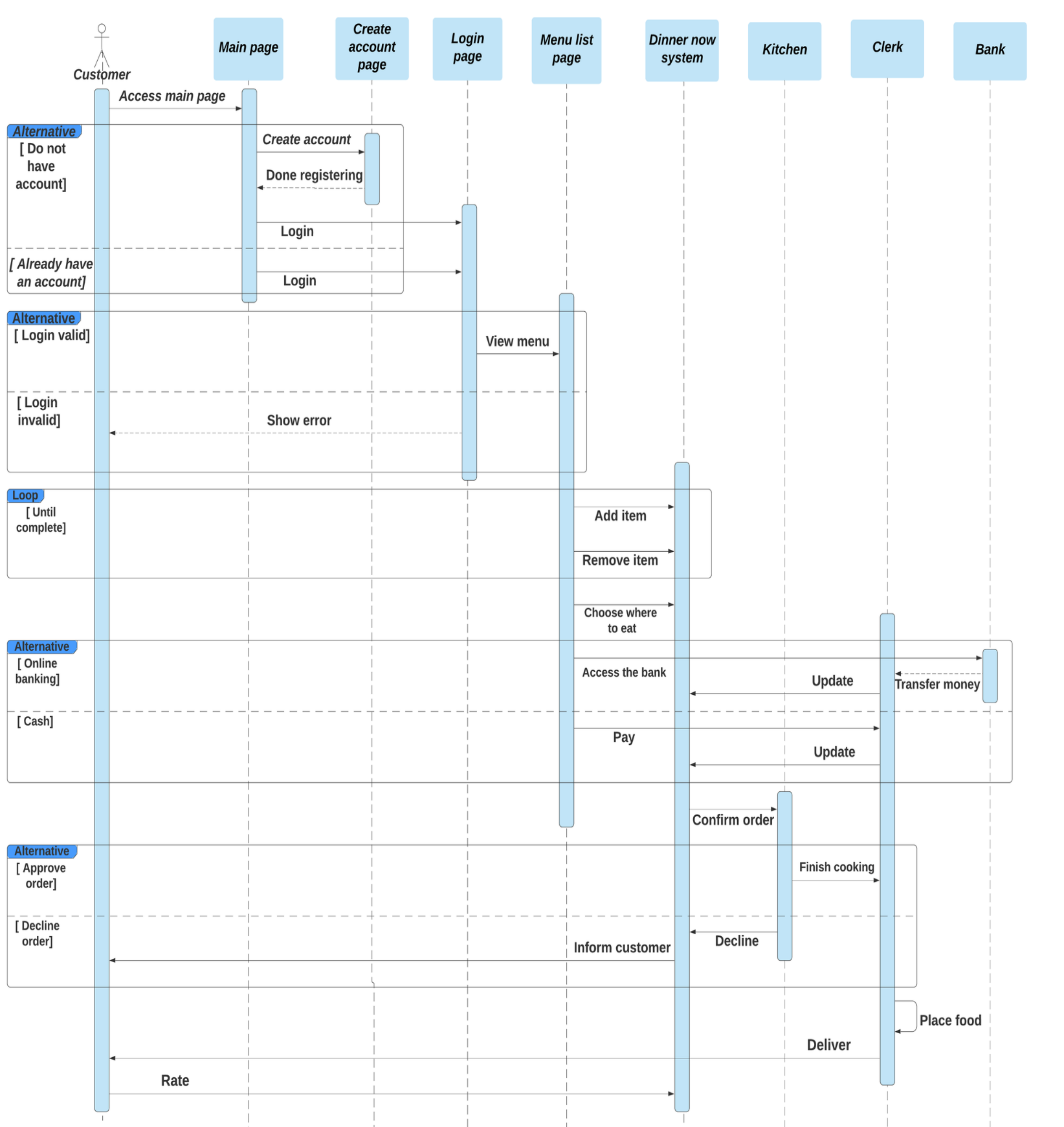


|  |  |
| --- | --- |
| Name | Ordering food |
| Actor | Registered customers, the kitchen, clerks, |
| Description | Registered users can order food, choose where to eat and pay in many ways. |
| Preconditions | Users need to access to home page. |
| Normal flow | 1. User access the home page of the website by QR code, URL or app.  2. User must login to their account, if user doesn’t have an account yet, user must create an account, and then login.  3. User views and selects items.  4. User selects where to eat.  5. User pays.  6. User checks out the system.  7. The kitchen receives and confirms the order.  8. The kitchen process the food.  9. Clerk delivers the dishes to the customers. |
| Exceptions | Exception 1: at step 4: If user did not select where to eat, can’t go to next steps.  Exception 2: at step 5: If user hasn’t logged into their bank account but attempt to pay by online banking method, the system will show error.  Exception 3: at step 7: The kitchen can decline the orders. |
| Alternative flow | Alternative 1: at exception 1: Website will notify that user hasn’t select where to have their dishes.  Alternative 2: at exception 2: Website will ask and forward you to the page where you can log in your bank account.  Alternative 3: at step 3: Users can search for specific items.  Alternative 4: at step 5: Users can apply coupon codes. |

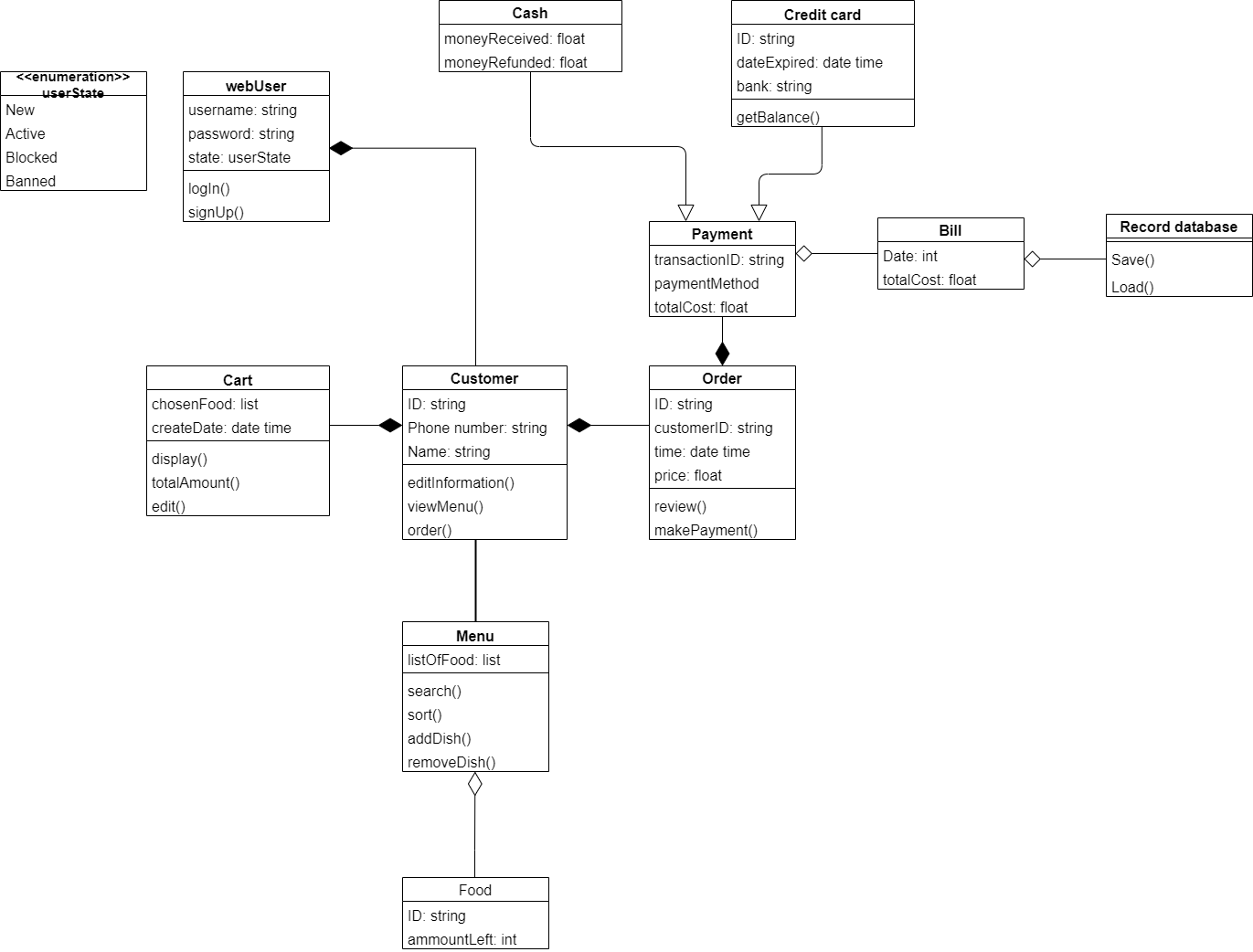
Task 2: System modeling

Task 2.1. Draw an activity diagram to capture **Major (not all)** functional requirements of the desired system

  
Task 2.2. Draw a sequence diagram for use-case in Task 1.3.



Task 2.3. Draw a class diagram



Task 3.1. Describe an architectural approach you will use to implement the desired system

For this task, we decided to choose the layered architecture (also known as onion architecture or n-tier architecture) because this pattern is the standard for most Java EE applications and therefore is widely known by most architects, designers, and developers; moreover the layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts and also matches the “single responsibility principle”.

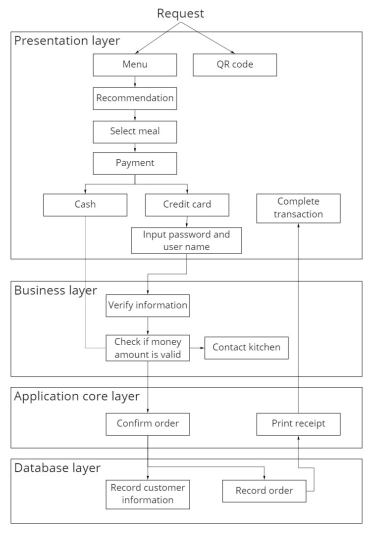
Pros and cons of the layered architecture:

Pros:

* We only need to consider a smaller scope in each layer, which makes the problem much more straightforward.
* Each layer has less case to test and therefore more testable.
* Changes in one layer will not affect downstream layers.
* You are able to replace the layer with another layer, as long as they are implemented in the same interface.
* Flexibility, maintainability, and scalability
* It is possible to configure different levels of security to different components deployed on different boxes

Cons:

* Too many layers can be costly.
* Too many layers can make the performance slow.
* Leaky abstraction can disturb layered intent.

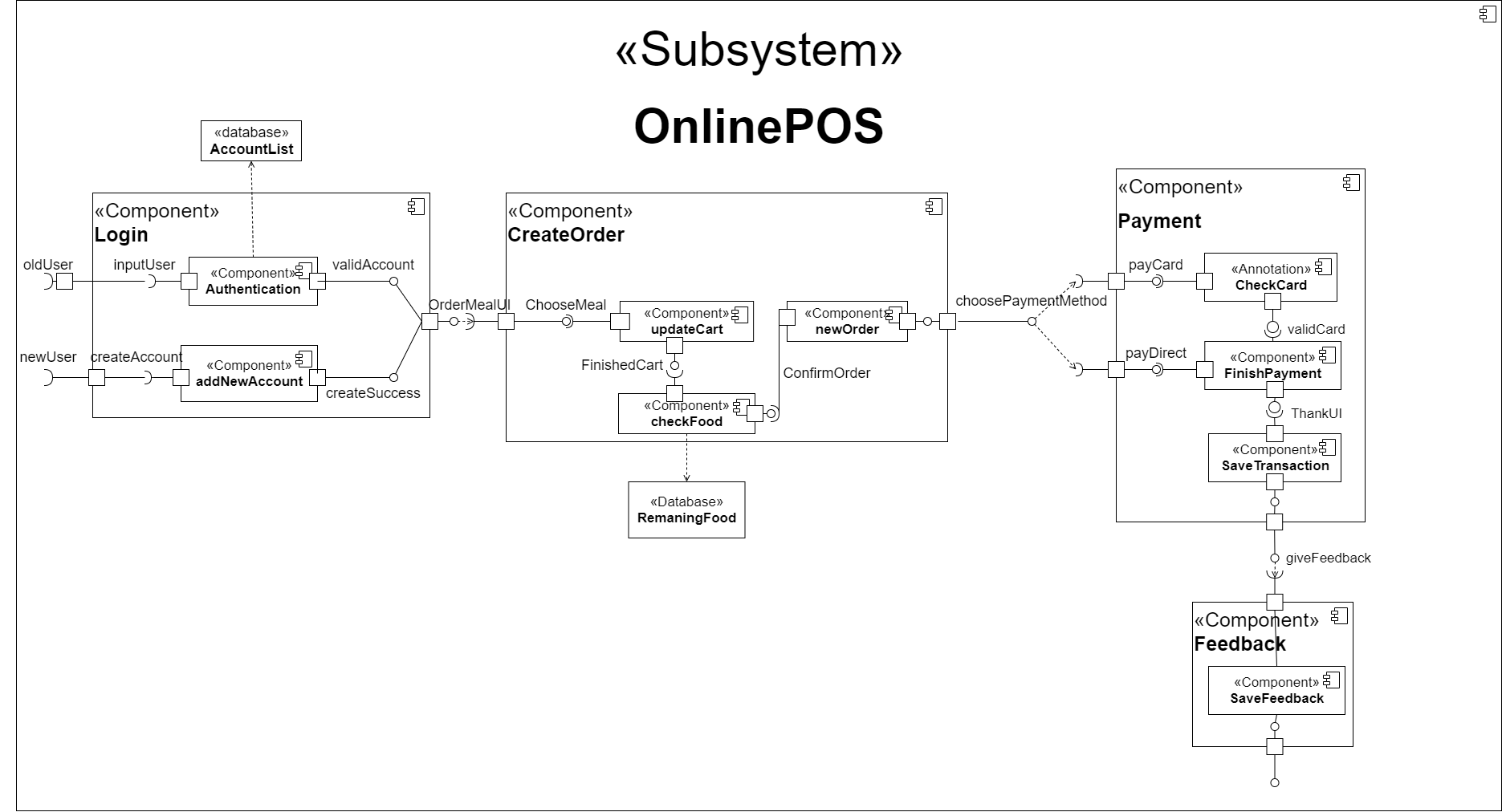


The above image is our architecture for the POS system. We will divide it into 4 layers:

* The presentation contains the UI for interaction between the user and the system.
* The business layer contain the logic need for
* The application core layer will contain useful function for the system
* The database layer contain the database to store information

Of all the layers, only the business layer will be open since we want the application core and the presentation core to interact with each other. There will be some cases like the print receipt function does not need to go through the business layer.

Task 3.2. Draw an implementation diagram for Major (not all) functional requirements

Component diagram:

Deployment diagram:

