ID2207
Modern Methods in Software Engineering

Homework 1
Group 6

FALL 2021

Yuchen Gao
Weikai Zhou

September 12, 2021

KTH EECS Embedded Systems

1 Question 1

First, according to the directions of the question, we generate the use case description (Table 1) on Page 2. We have listed the Name, Participating Actor(s), Entry Conditions, Exit Conditions, Quality Conditions, and Event Flow.

According to the use case description, we draw the use case diagram (Figure 1), which is shown on Page 3. For the "Login", it is a generalization of "SoftwareDeveloperLogin", "ITManagerLogin", and "FinancialManagerLogin". It also include "ValidateUse", which extend "InvalidCredentials". After the developer logs in, he/she can "CreateSalaryRaiseRequest". Then, after the IT Manager logs in, he/she can review the application, which includes two parts, "CheckPerformance" and "CheckPriority". They also extend "PerformanceUnsatisfied" and "PriorityConflict" respectively. After that, the IT manager can decide on the application, either reject or approve. If the IT manager approves, the application should sent to financial manager. After the financial manager logs in, he/she can further review the application, where the budget should be check. And it extends to "InsufficentBudget". After that, the application can be finalized, which means the developer can get salary raised immediately, but it extends to "RaiseSalaryLater" when the budget is insufficient.

Name: Manage Salary Raise Requests
Participating actor(s): Software Developer (Initiator) IT Manager Financial Manager
Entry Conditions: 1. Developer logins to the system. 2. System validates the developer credentials. 3. Developer accesses the create salary raise request functionality. This use case extends InvalidCredentials, PerformanceUnsatisfied, PriorityConflict, InsufficientBudget, and RaiseSalaryLater. They are initiated when the login credentials are invalid, or when the performance of the initiator developer is unsatisfied, or when there are other developers who requested salary raise in the same period and the developer has no priority, or when the budget is not enough to support the salary raise, or when the salary raise needs to be delayed due to the insufficient budget.
Exit Conditions: 1. The IT manager has decided to approve or reject the salary raise request. 2. If the request is approved by the IT manager, the financial manager has checked if the budget meets the raise request, and the raise may be effective at a later date when the budget is not allowed currently.
Quality Conditions: The system should be available and functioning without unexpected interruptions.
Event Flow: 1. Developer chooses to create a new salary raise request application. 2. System displays the requested form. 3. Developer fills the application with reasons and expected salary. 4. Developer chooses to send the application to the IT manager. 5. System displays the new request to the IT manager. 6. IT manager reviews the salary raise request and checks the performance reports of the developers. 7. System displays the performance report to the IT manager. 8. The IT manager chooses to review the raise priorities for the other developers. 9. Systems displays the list of salary raise requests to the IT manager. 10. IT manager chooses to approve or reject the request. 11. If the request is rejected, system displays the result of the request to the initiator developer. 12. If the request is approved, the IT manager sends the request to the financial manager. 13. System displays the request to the financial manager. 14. Financial manager reviews the request and chooses to check the budget. 15. System displays the budget. 16. Financial manager approves the request at the current time if the budget is sufficient or makes the raise effective at a later date when the budget is insufficient. 17. System displays the result of the request to the initiator developer.

Table 1: Use case description

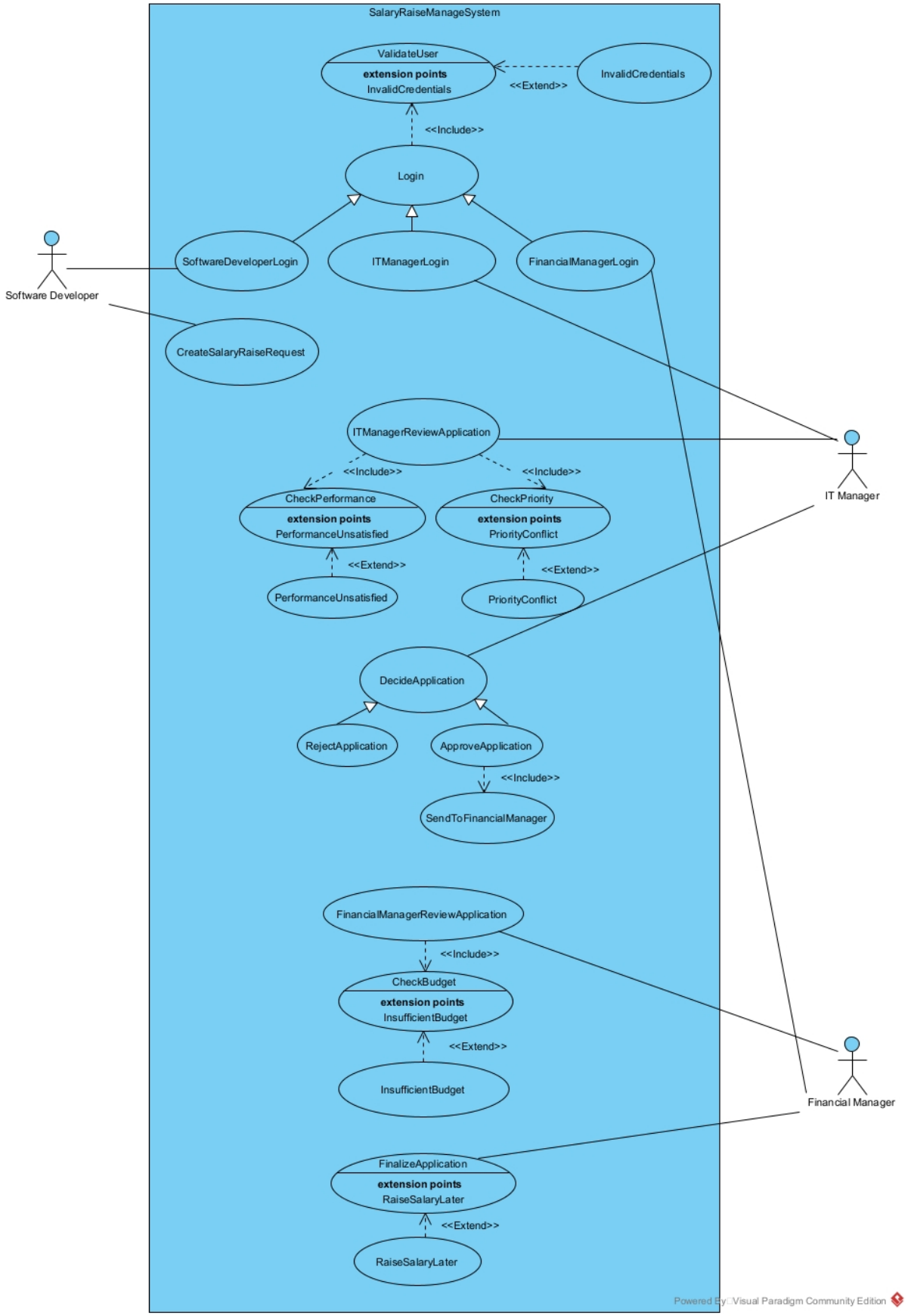

Figure 1: Use case diagram

2 Question 2

In question 2, we are using class diagrams to reflect the problems when the manager finds there is a conflict for the salary raise requests, which is what we are going to resolve. Firstly, we have the uppermost class: the negotiation template, which owns several properties and one operation. The properties are listed below and the operation for manager is to add the priority parts through the template, which is optional. Then, the negotiation template is composed of the conflict reason and history of negotiation, it is indicated the history of negotiation could be shown, and also could be none if there was no request. The conflict reason must be displayed, we use the composition relationship to show the relationship between template and conflicts, aggregation is the connection between template and historical negotiation. Next, the Conflicts is a set of report, priority report and insufficient budget report. These relationship can be denoted as aggregation.

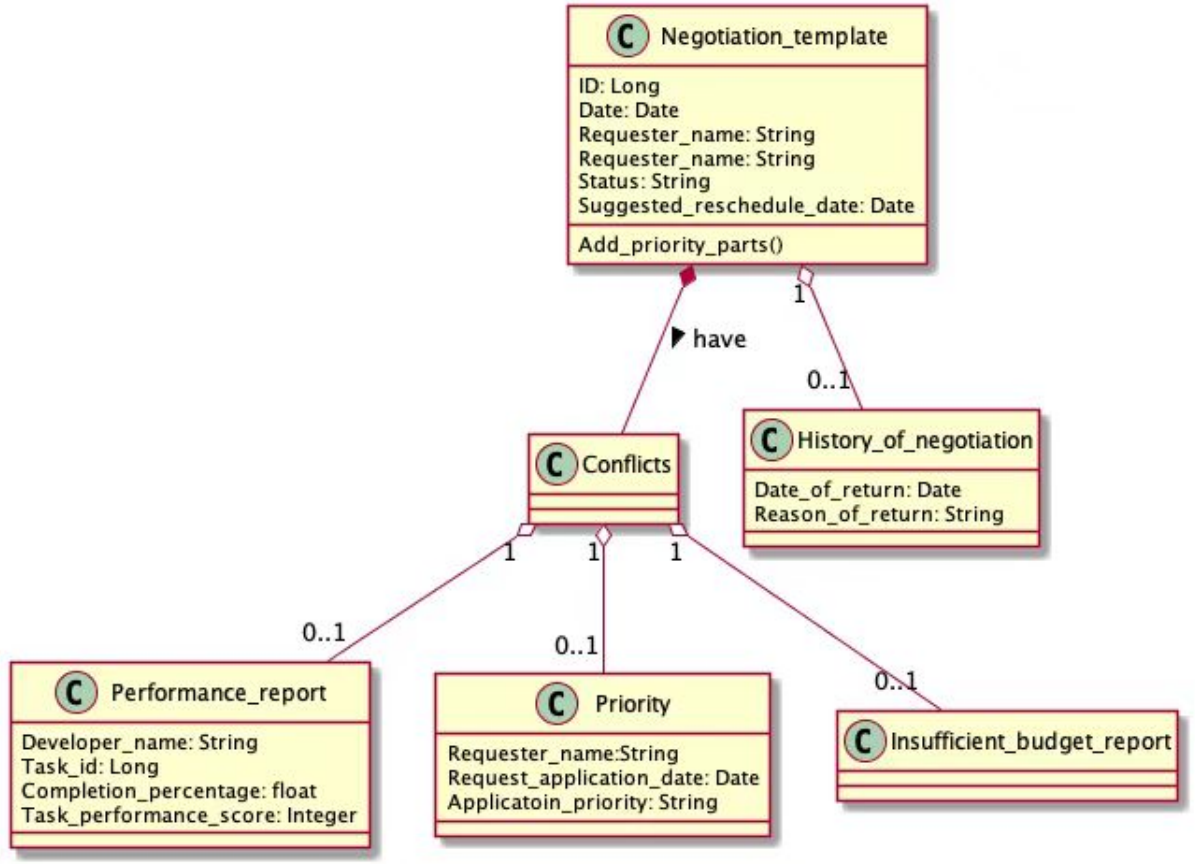

Figure 2: Class diagram

3 Question 3

We create the following sequence diagram (Figure 3). First, James create a request on "EmergencyRequestControl", which creates the "EmergencyRequestForm". Then, James selects the nurses, enters the room number and chooses to send the application. After that, the "EmergencyRequestForm" activates the "SMSControl" and sends corresponding information to it. The "SMSControl" look up the contact details in the "NurseList" and sends to messages to correspond-

ing nurses with the information it gets. After John and Sam receive the message, they respond to it. And the statuses are updated to inform the doctor.

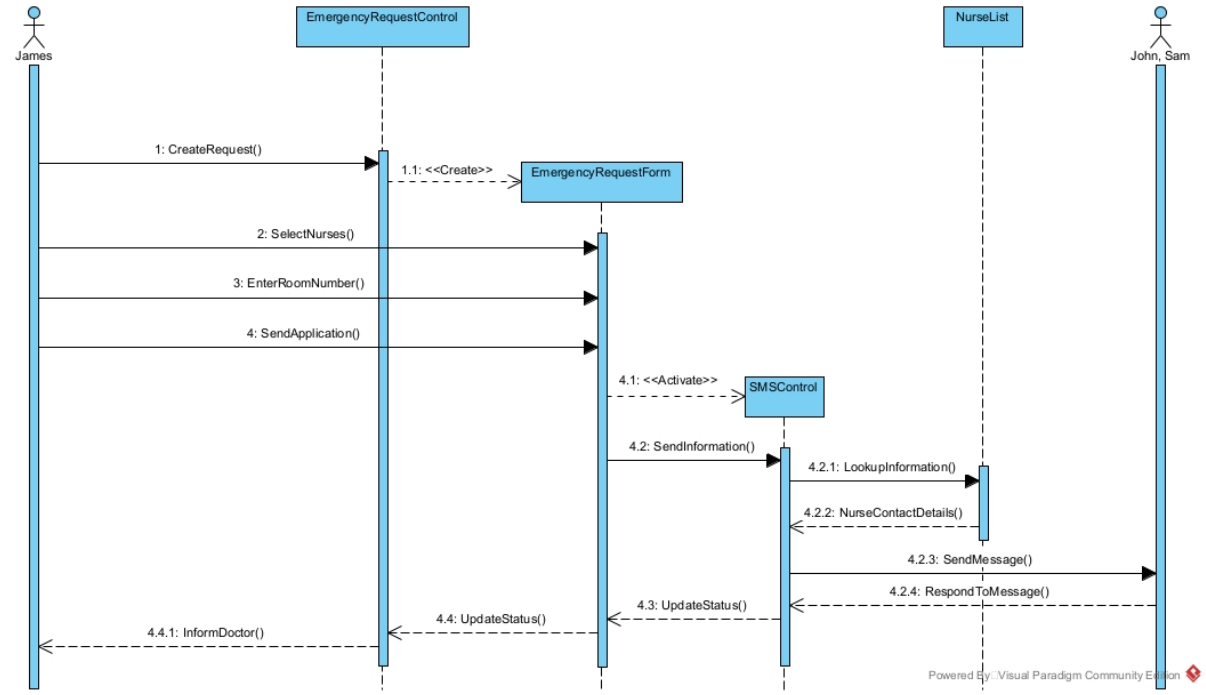

Figure 3: Sequence diagram

4 Question 4

4.1 Version 1

It contains 5 parts, which are “Employee”, “Main branch resources department”, “Online system”, “Payment gateway”, and “Delivery department” respectively. The initial activities are similar to the tutorial. First, the employee logs in to make an order, where the validity is checked. Then, the availability is check in main branch resources department. If it is available, the order will be sent to delivery and the status is update with “Waiting delivery”. If it is unavailable, it will be requested online. They online system will check the availability. If it is also unavailable, the status will be updated with “Unavailable” and the employee will be informed so that he/she can exit. If it is available, the order information will be sent to the payment gateway to create an invoice and update status with “Pending”. The invoice will be sent to employee to pay. After the payment gateway receives money, a confirmation will be sent to online system so that it can send order to delivery and update the status with “Waiting delivery”. When delivery department receives the order, it will retrieve the address and update the status with “Under delivery” and start to deliver. If the delivery man cannot find the address, he/she will return the package and inform the employee so that the employee can check the address for anther delivery. If the delivery finds the address successfully, the employee will get the package and check it. If it is wrong items, the employee can refuse to receive and make another order. If it is correct items, the employee can close the application so that the status is updated with “Received”.

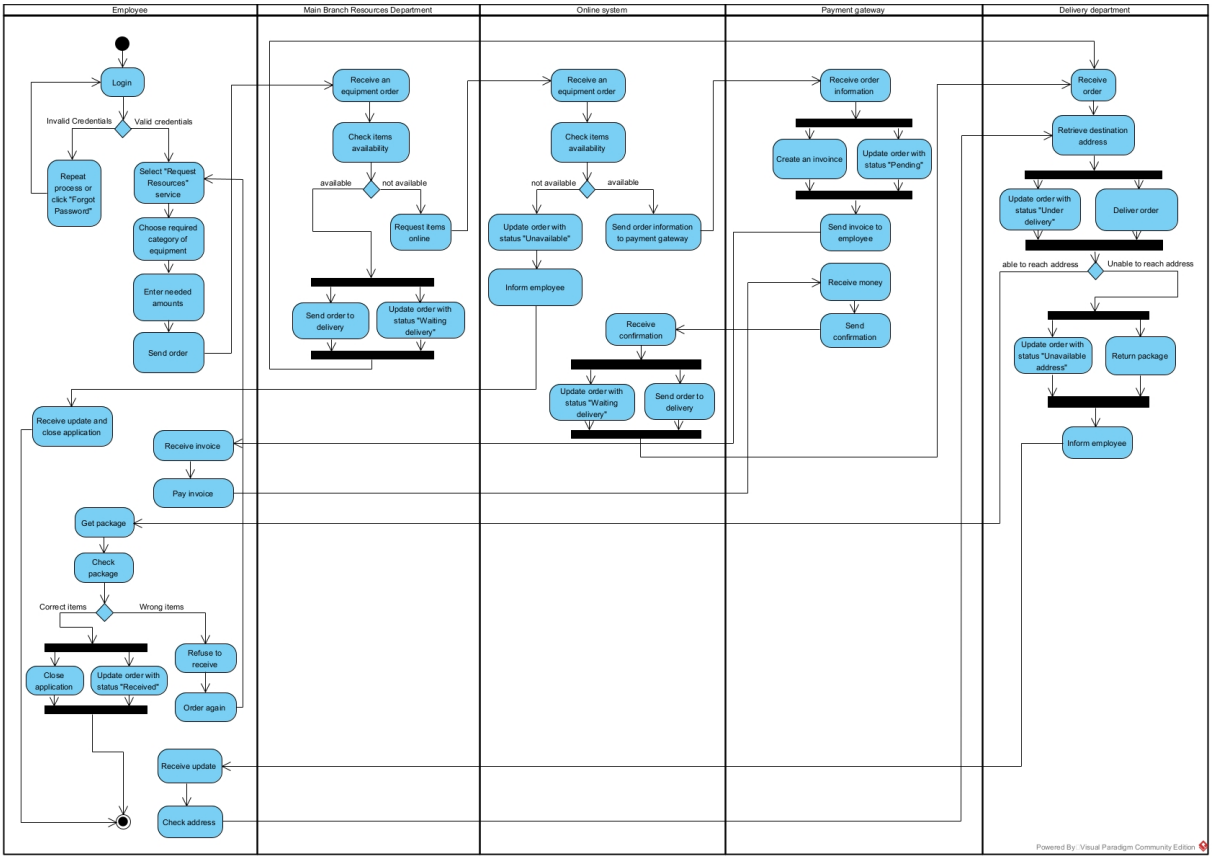

Figure 4: Activity diagram (version 1)

4.2 Version 2

We visualize this process by defining different working domains, the Main Branch is the core of the model. Firstly, the validation process is denoted, and we enter the Online system when orders are sent, and we have our items status printed online so that the one who orders know what is happening right now. Once the availability of order is confirmed, the bill is sent to customer. Next, the invoice is paid by the customer, and the confirmation will be send by Payment Gateway, which is the associated part that controls money flow. Then, the order can be prompted to next stage, which is going to be sent by Delivery Department. The Delivery Department checks the address and arranges the delivery man. There are two possible problems have to be postulated, first one, the order was sent to wrong person and returned, we have to check the order again and send again. The second one is that when the Main Branch receives items, finding out the wrong things are sent, refusing to accept, then we also have to go back to the check loop, so that the correct things can be sent again.

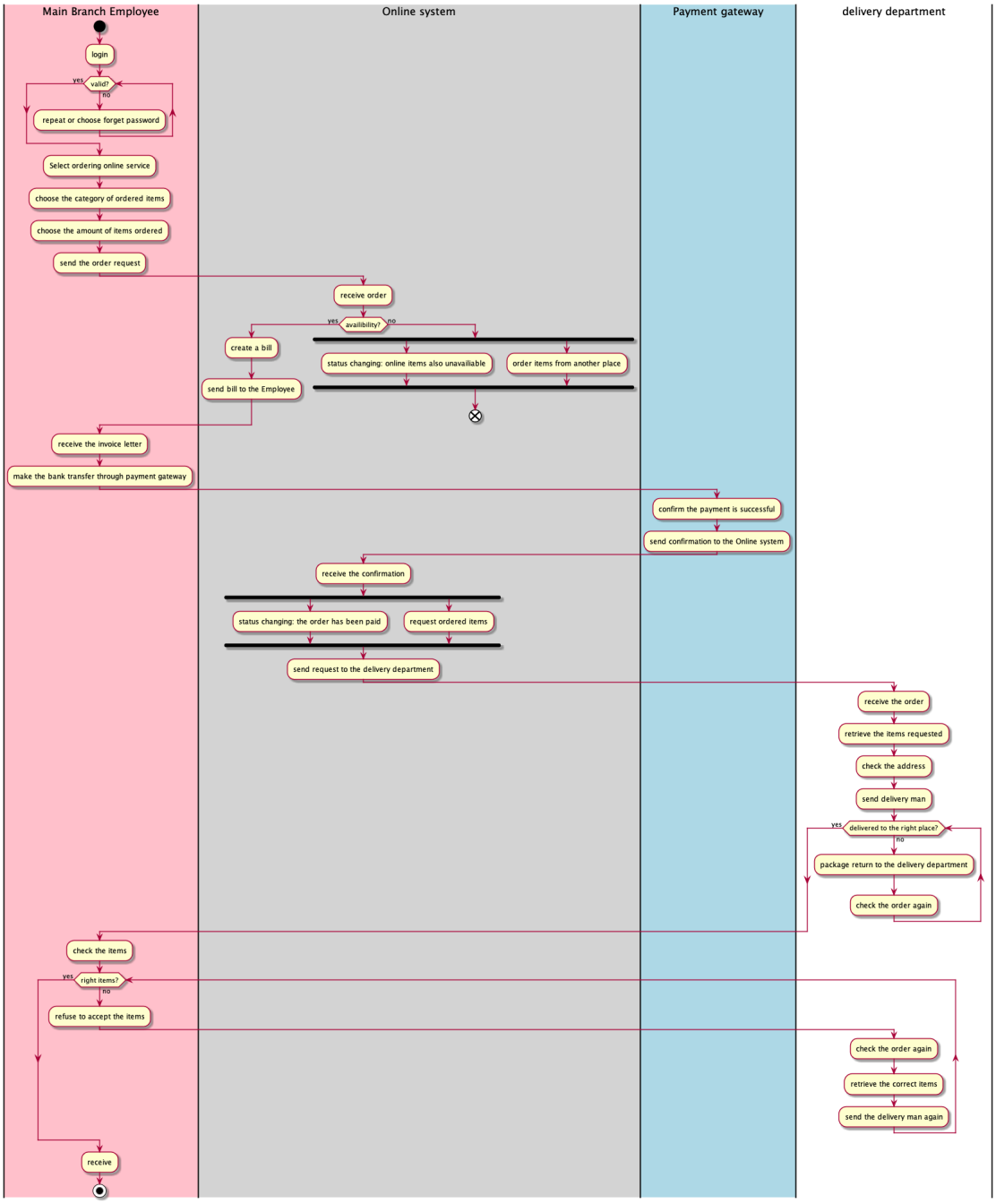

Figure 5: Activity diagram (version 2)