ID2207
Modern Methods in Software Engineering

Homework 1 - Group 19

September 2021
Royal Institute of Technology

Aksel Uhr
Abdullah Abdullah

Use case identification

Step one was to identify roles in the business process (the salary raise request system). These were developer, manager and the financial manager. Each role has different responsibilities seen below. The responsibilities were defined based on the problem description.

Role	Responsibilities
Developer	1. Fill salary raise request with required information. 2. Send salary raise request to the manager.
Manager	1. Check the performance report. 2. Check potential conflicts in priorities. 3. Approve or reject salary raise request and send it forward.
Financial manager	1. Determine if today's budget allows the approved salary raise.

Afterwards, given the details of the case description, the use cases below were defined. After the login is finished, the user's credentials are validated and the user is authorized. This means for example that the developer only has the authority to create a salary raise request, and the financial manager only has permission to approve the request with regards to the budget. The authorization is not part of the case description, but is worth considering since there are certain responsibilities for a certain actor. A few use cases below (e.g. SendSalaryRaiseRequest) are not distinct with regards to the given case details, but are still necessary for the base use cases (e.g. CreateSalaryRaiseRequest) and the abstraction of the system. Those use cases will be noted as <<include(s)>> in the diagram and are always mandatory. Ultimately, the identified exceptional use cases below will be noted as <<extend(s)>> in the diagram. They are initiated on different occasions seen in the diagram and dependent on its base use case.

Use cases that were identified:	Exceptional use cases:
a. Login (+ Register) b. ValidateUser c. AuthorizeUser d. CreateSalaryRaiseRequest e. SendSalaryRaiseRequest f. ReviewApplication g. CheckPerformance h. CheckPriority i. SendApprovedRequset j. CheckBudget k. FinalizeApplication	a. InvalidCredentials b. InvalidAccess/Permission c. PerformanceConflict d. PriorityConflict e. BudgetConflict f. DelayRaise

Q1: Use Case Description

The use case identification above facilitated the definition of the use case description below. The entry conditions consist of the required conditions for the rest of the use case to continue. The exit condition is the output of the use case, namely an agreement of whether to approve or reject the salary raise request. The event flow consists of the to-be use case diagram's flow.

Name: ManageSalaryRaiseRequests

Participating actor(s):

1. Developer
2. Manager
3. Financial manager

Entry conditions:

1. Actors login to the system.
2. System validates the actor's credentials and authorization is executed.
3. The authorization follows:
    a. The developer accesses the salary raise request functionality.
    b. The manager accesses the request(s).
    c. The financial manager accesses approved request(s).

Exit conditions:

1. Developer, manager and financial manager are in an agreement considering whether to approve or reject the salary raise request.

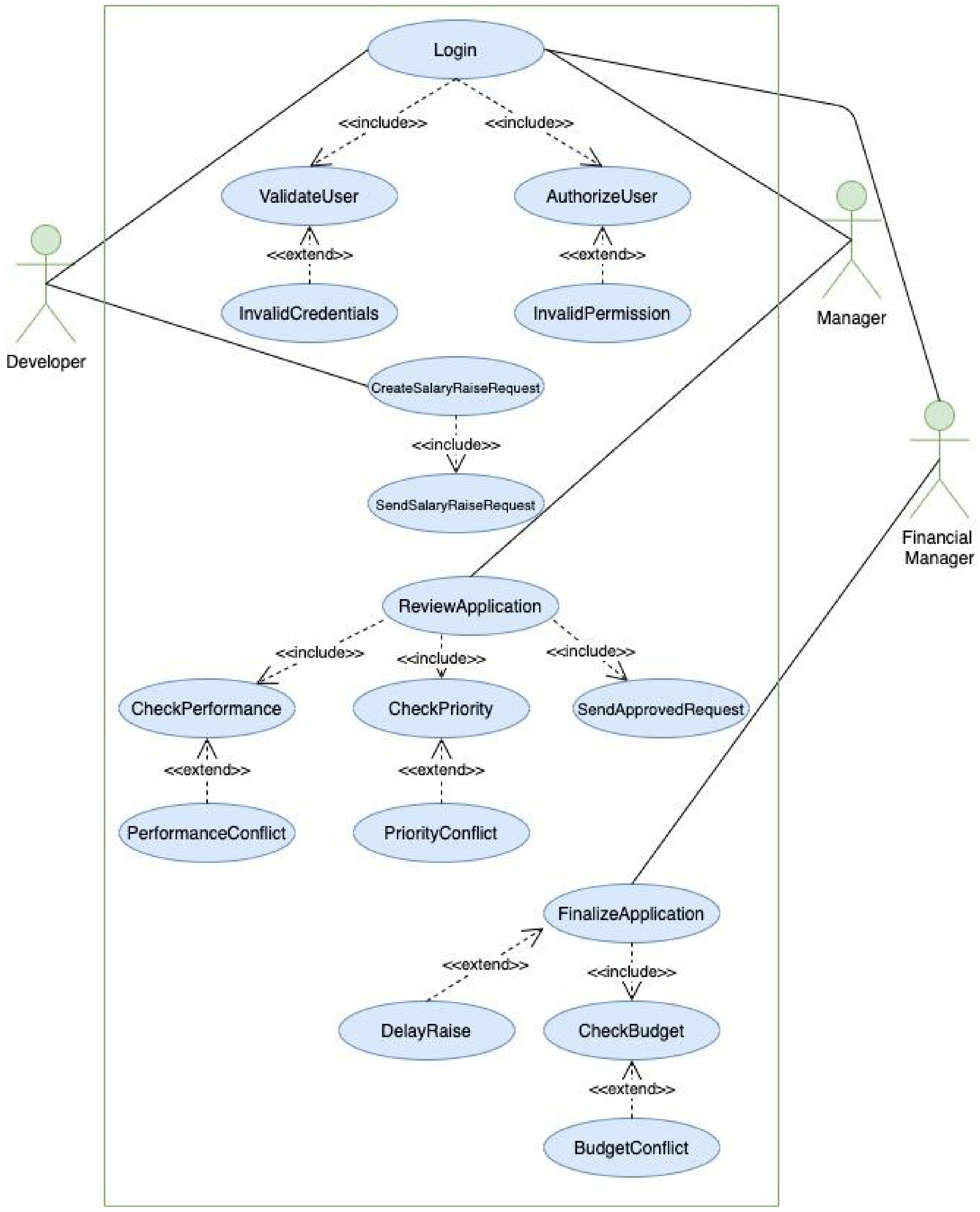
Quality conditions:

1. The system should be available.
2. The system should be easy to use.
3. The system should be functioning without unexpected interruptions.
4. The system should be easy to maintain.
5. The system should be secure.

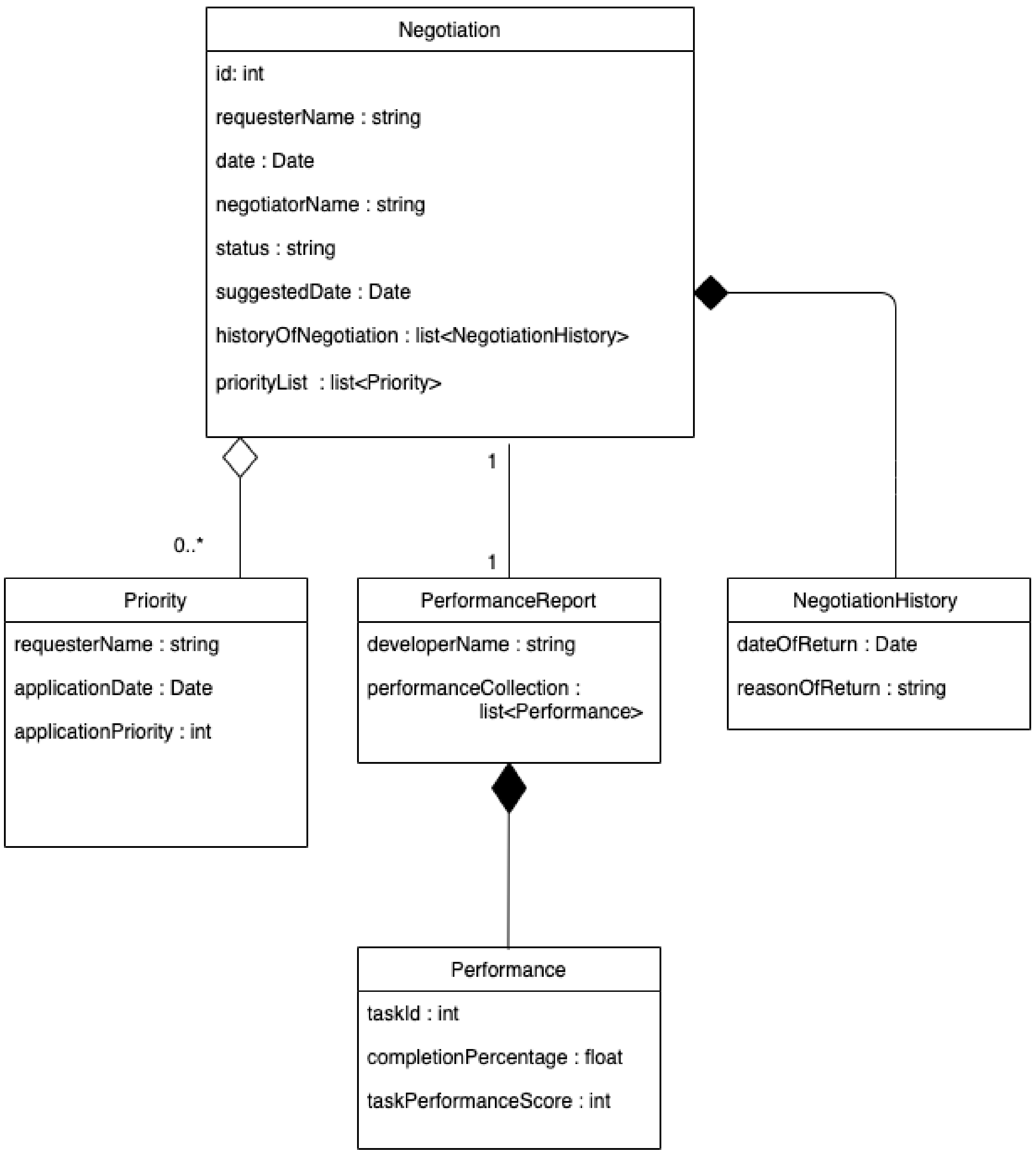
Event Flow:

1. Developer creates a new salary raise request.
2. The system displays the form.
3. Developer fills the application with necessary information about the salary raise.
4. Developer sends the form.
5. The application is now available and displayed for the manager.
6. The system displays charts of performance and priorities.
7. The manager reviews the application.
8. The manager reviews performance and priorities with regards to the current application.
9. The manager approves or rejects the request.
10. If rejected, the system displays the rejected application for the developer. The application process is now terminated.
11. If approved, the application is now available and displayed for the financial manager.
12. The system displays the budget.
13. The financial manager approves or delays the request, based on budgeting.
14. The system displays the final state of the application to the developer.

Q1: Use Case Diagram:



Q2: Class diagram



Performance Report Summary - For Performance report summary a separate class is created with attributes developerName and performanceCollection which is a list of Performance. The Performance class is created in composition with PerformanceReport. Performance class is created with attributes taskId, completionPercentage, taskPerformanceScore. The PerformanceReport class itself will be in one to one association with the Negotiation template Class because for one negotiation we can have only one performance report for the developer.

Priority - For Priority a separate class is created with attributes requesterName, applicationDate and applicationPriority. As the priority can be added optionally by the manager so it means that priority can range from 0 to infinity. Furthermore the Priority class will be in aggregation with the Negotiation template class which means that if the Negotiation class gets destroyed the Priority class can still exist.

Negotiation - For Negotiation template class is created with attributes of id, date, requesterName, negotiatorName, status, suggestedRescheduleDate, historyOfNegotiation as list of NegotiationHistory and priorityList as a list of Priority.

NegotiationHistory - For NegotiationHistory class is created with attributes of dateOfReturn and reasonOfReturn. The NegotiationHistory class is in composition with the Negotiation template class because if the Negotiation template is destroyed no NegotiationHistory will exist.

Q3: Sequence Diagrams

Solution

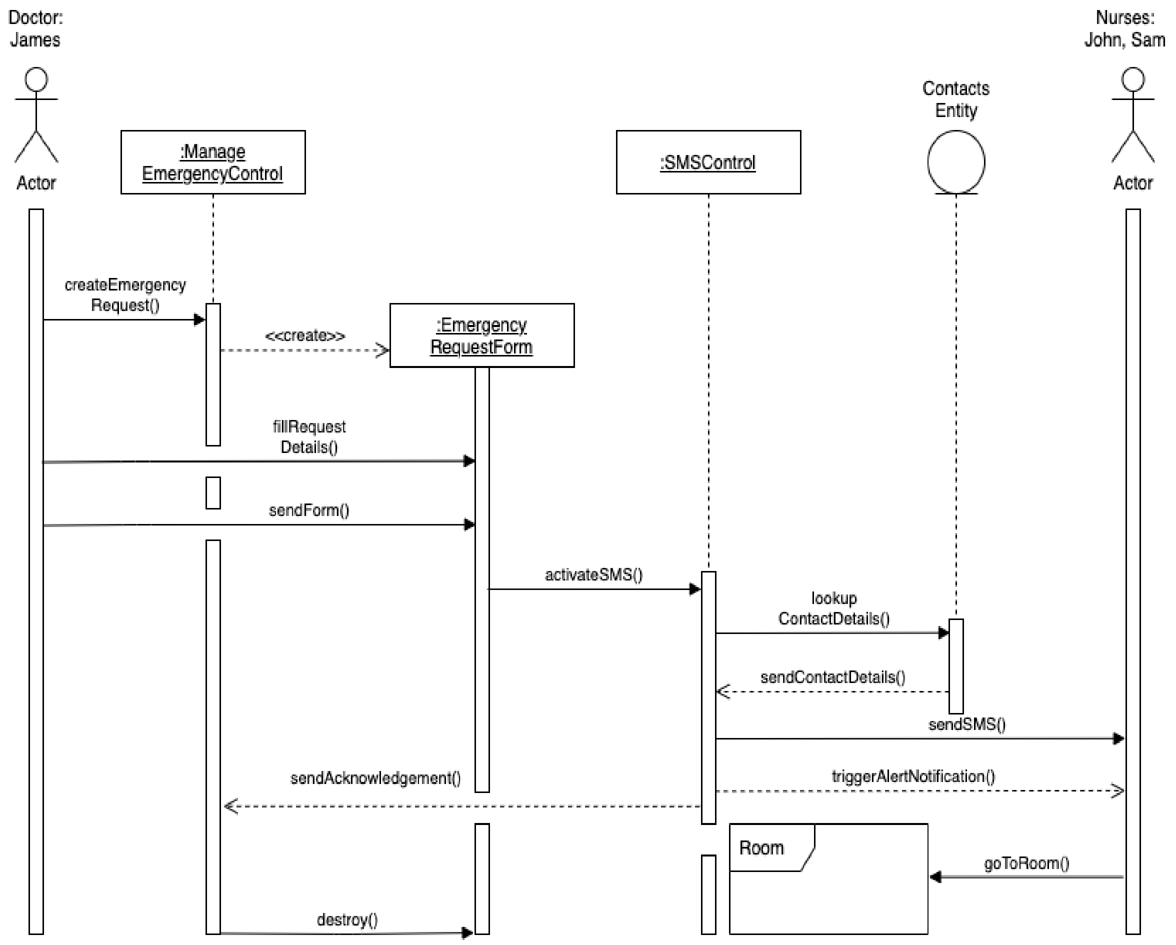
From the description above we get the idea of the actors for the sequence diagram. It involves an instance of Doctor as James and two instances of Nurses as John and Sam. For the flow of events after discussion we came up with instances of ManagerEmergencyControl, EmergencyRequestForm, SMSControl and a ContactsEntity to show it as a db instance where contacts of all nurses are stored.

The flow of events is illustrated in the following way in our sequence diagram.

- Actor James after login chooses to create a new emergency request and sends a synchronous message “createEmergencyRequest” to ManageEmergencyControl.
- The ManageEmergencyControl instance asynchronously creates a new instance of EmergencyRequestForm and displays it to actor James.
- James then sends a “fillRequestDetails” synchronous message and after that sends a “sendForm” synchronous message to submit the form in the system.
- The system activates the SMS function by sending a synchronous message “activateSMS” to SMSControl.
- The SMSControl sends a synchronous message “lookupContactDetails” to the db instance/entity of Contacts to get the contact information of Nurses.
- The db instance gets the data from db and responds with an asynchronous message “sendContactDetails” to SMSControl.
- The SMSControl then receives the data and sends SMS to the concerned nurses with an asynchronous message of “sendSMS”.
- The “sendSMS” message triggers an asynchronous message “triggerAlertNotification” which sends notifications to the nurses on their phones.
- After successful trigger of notification another asynchronous message “sendAcknowledgement" is triggered to ManageEmergencyControl to inform about the completion of the process.
- The ManageEmergencyControl after getting acknowledgement sends a synchronous message to destroy the EmergencyRequestForm instance.

- On the other hand the Nurses John and Sam receive the notification and sms on their phones with room number details and they immediately visit the requested room numbers.

The Sequence Diagram is:



Q4: Activity Diagrams

Draw an activity diagram that visualizes the process of **online ordering of the materials** in case they are not available locally. Consider using an **online system, payment gateway, delivery department** and an **employee** in the main branch. Consider the following special cases:

- The requested materials are not available in online systems.
- The wrong materials were ordered (wrong package delivered)
- The delivery man didn't reach the address and he/she returned the package.
- Use parallel flows where possible.

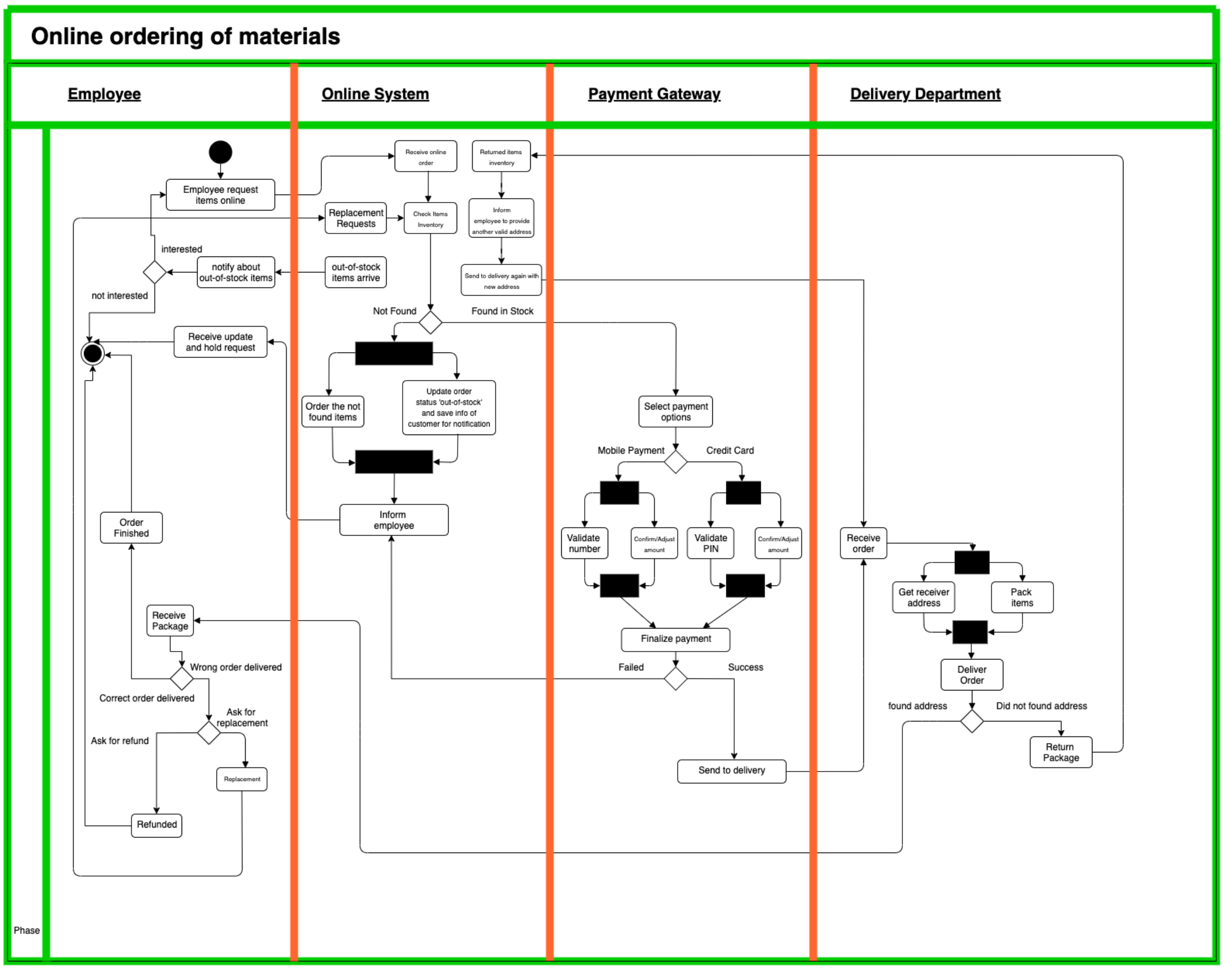
Solution

For the activity diagram of online ordering of the material system we have 4 resources in the main branch. An **Employee** who is sending a request to order the items online. The **Online System** which receives the order and processes it accordingly. The **Payment Gateway** where after processing the order the customer is asked for payment of the items. The **Delivery Department** which receives the order after payment, packs it and delivers it to the customer or an employee of hospital in this case.

We have covered the special cases mentioned in the question with the following approach

- If the requested materials are not available in the online systems inventory the online system will order those items for their inventory in order to be available later and at the same time will update the order status to out-of-stock for customer and save customer info so that they can send notification to the customer later when the requested items are available in the online system. Later on when the items are available in the system the customer can be informed using notifications on app or through email about the availability of items and if the user is interested they can then order again the requested items.
- If the wrong package is delivered to the customer, the customer can have two options. Either ask for a refund of the order or ask for a replacement of the order items. If the customer asks for a refund the online system after confirming can complete the request of refund for the customer. If the customer asks for a replacement of the order items they will be redirected to replacement requests form in the system where they can enter details and follow the whole system to complete the order.
- If the delivery man didn’t reach the address and returned the package. The returned package will be placed in returned items inventory in the system and the system can then request the customer for a new valid address and again send the order towards delivery department with new valid address.

The Activity diagram is given below.



September 2021
Royal Institute of Technology

Aksel Uhr
Abdullah Abdullah