UNIVERSITY OF STIRLING

COMPUTING SCIENCE AND MATHEMATICS

CSCU9P5 GROUP PROJECT

Autumn 2020

Project Group 4

Students in the Group:

- **2712141**
- **2718715**
- **2721301**
- **2933668**

Contents:

Requirements Specification: Understanding the problem

Assumptions

Assumptions about situations and ways of working had to be made as we were only able to use the information in the design brief and had no ability to gain domain knowledge from stakeholders directly.

One assumption we made is that the courier should be able to have the ability to look up packages in the system without a QR Code. This gives the courier better flexibility for unexpected situations, for example when collecting a package if the customer has lost the QR code they were sent, or if their scanning device stops working. The alternative would be to only allow the courier to look up packages via a QR code, this would limit their ability respond to issues with QR codes.

As the app requires several details to be entered by the customer sending a package, we have assumed that it should be possible for a customer to create an account. This will allow the information to be stored, and so reducing the time it takes to request a subsequent collection/delivery.

The QR code Printer is a hardware element which we assume would be linked with the courier app. As we are looking at the hardware side of the project it has been assumed that this is outside our remit, and so expect that it will always be working and always have ink available.

Use Case Diagram

The Use Case Diagram (Figure 1) is designed to give a broad overview over goals the actors might want to achieve using the app, rather than specifying every possible action (for example scanning a QR-Code and providing confirmation). Therefore, the use cases depicted implicitly bundle multiple actions that couriers, and customers can perform.

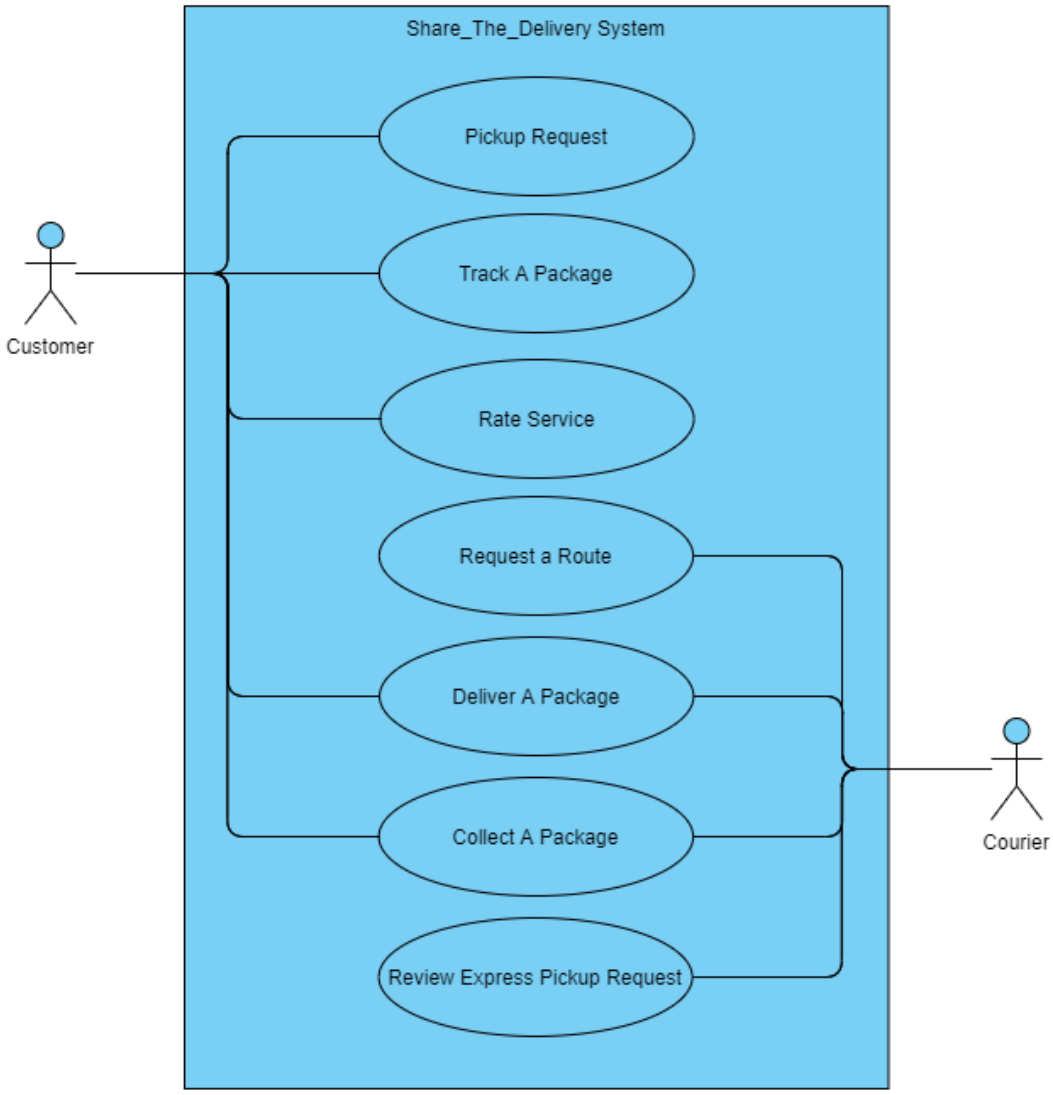

Figure 1: Use Case Diagram

Use Case Descriptions

Pickup Request

Goal	Customer Places Pickup Request
Actors	Customer
Preconditions	Customer has an account
Postconditions	A new package is created, and the status is “With Sender”
Trigger	Customer requests a pickup
Main Flow	1. System requests package details (delivery mode, package size, sender, and recipient addresses) 2. Customer enters package details 3. System requests confirmation of package details 4. Customer confirms 5. System quotes a price 6. Customer enters payment details 7. System takes payment, submits the package to the system and produces QR code

Extension Condition	3a. Customer refuses confirmation
Extension Flow	i. Error message is displayed ii. Continue at step 1

Extension Condition	6a. Customer enters incorrect payment details
Extension Flow	i. Error message is displayed ii. Continue at step 5

Extension Condition	6b. Customer clicks the “Change Information” button
Extension Flow	i. Continue at Step 1

Extension Condition	2a. Customer selects ‘XL’ for package size.
Extension Flow	ii. System requests dimensions and weight of package iii. Customer inputs the dimensions and weight of package iv. Continue to Step 3

Track A Package

Goal	Customer receives information about the package
Actors	Customer
Preconditions	The package is registered on the system
Postconditions	The system displays all required information
Trigger	Customer requests display of information about package
Main Flow	1. System displays options to present the QR-code 2. Customer selects an option to present the QR-Code 3. System prompts the customer for a QR-Code 4. The Customer provides the appropriate QR-Code 5. The System displays appropriate information

Extension Condition	4a. Customer fails to provide the QR code
Extension Flow	i. An error message is displayed ii. continue at step 1

Request a new Route

Goal	Courier receives a route
Actors	Courier
Preconditions	There are packages with status “with Sender”; Courier is logged into their account
Postconditions	Courier has received a route; System has created all relevant notifications
Trigger	Courier requests route
Main Flow	1. System proposes route to the courier and asks for confirmation 2. Courier confirms 3. System creates notifications for collection and delivery time slots

Extension Condition	2a. Courier rejects the proposed route
Extension Flow	i. Courier amends the route as required ii. System notifies customers of collection time slots

Deliver A Package

Goal	Receiver receives package
Actors	Courier
Preconditions	The status of the package is “out for delivery"
Postconditions	The status of the package is “with receiver”
Trigger	Courier selects delivery
Main Flow	1. The Courier scans the QR-Code on the package. 2. The System marks the package as delivered. 3. The System records the arrival time. 4. System sends a rating notification to the customer and provides confirmation to the courier

Extension Condition	1a. The delivery attempt fails
Extension Flow	i. Courier manually marks the pickup as failed and enters reason ii. ii. Status of the package remains as out for delivery

Extension Condition	1a. QR has failed to scan
Extension Flow	i. System provides error message and asks the user to try again ii. ii. Continue step 1

Collect A Package

Goal	Courier picks up the package from the Customer at their location
Actors	Courier, Customer
Preconditions	Customer has successfully requested a pickup and the status of the package is “with sender”
Postconditions	The status of the package is “In Transit”
Trigger	Courier selects pick up
Main Flow	1. Courier scans QR-code 2. System changes status of package to “In Transit” 3. Courier prints off QR code 4. System sends a rating notification to the customer and provides confirmation to the courier

Extension Condition	1a. Customer fails to provide QR-code
Extension Flow	i. Courier uses customer’s name and address to manually find package in system ii. Continue from Step 2

Extension Condition	1b. QR Scan failed to scan
Extension Flow	i. System provides error message asking to try again. ii. Continue from Step 1

Extension Condition	3a. Error printing the QR Code
Extension Flow	i. System displays error ii. ii. Continue from Step 3

System Design Description: Modelling Design of Solution
Static Design Model

Class Diagram

The Class Diagram (Figure 2) shows the class diagram for the Share_The_Delivery System. Boundary Classes are the CourierApp and the CustomerApp. Controller classes are the PickUpManager, RouteManager, PersonManager, PerformanceEvaluationManager and NotificationPusher. Finally, Customer, Courier, Package, Route, OpenPickups and Person are entity classes. External systems are perceived as actors in this model and are therefore not part of the diagram.

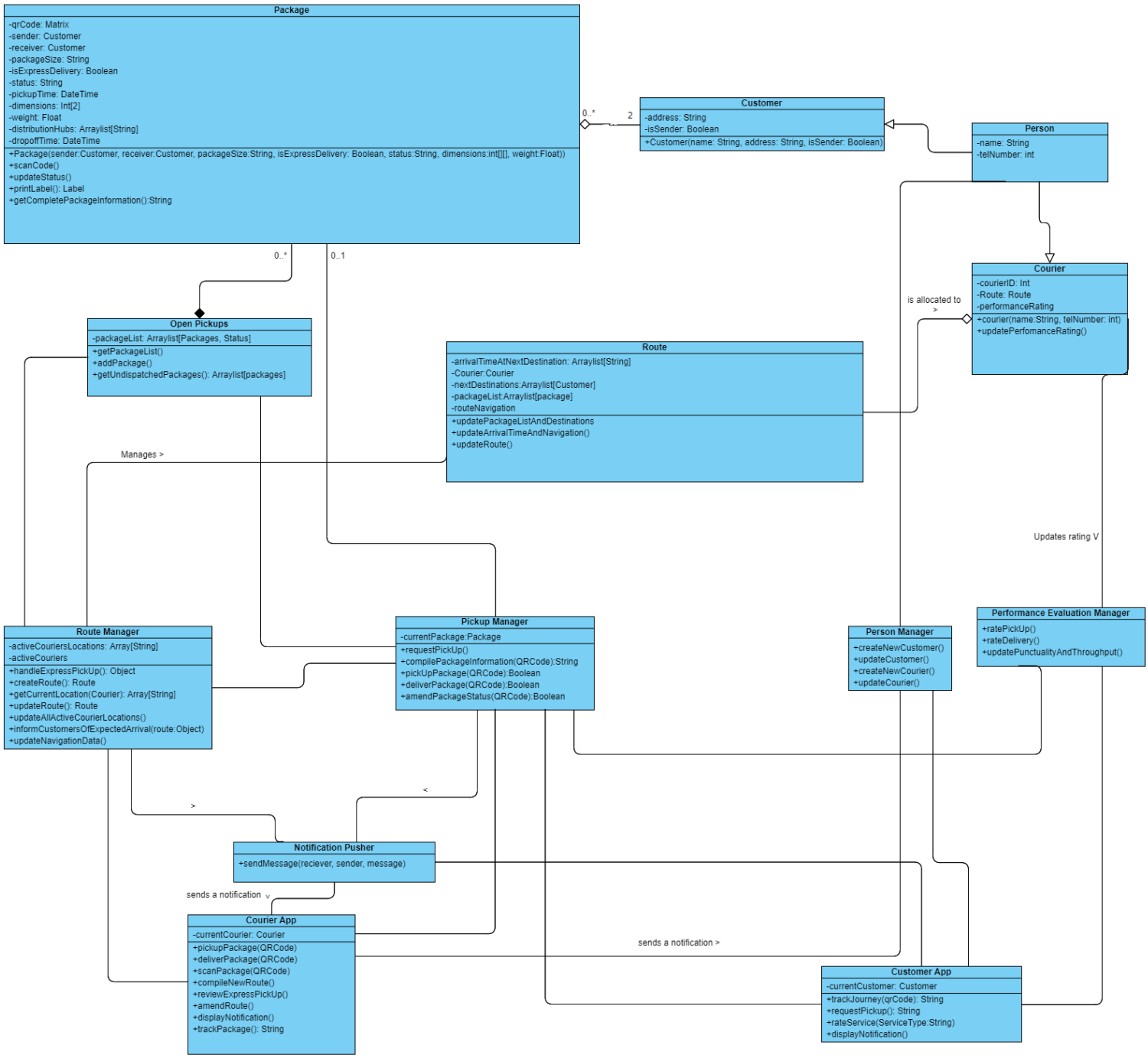

Figure 2: Class Diagram

Class Descriptions

The system consists of boundary (customerApp and courierApp), controller (PickUpManager, RouteManager, PerformanceEvaluationManager, PersonManager and NotificationPusher) and entity (Package, Courier, Customer, Route, OpenPickUps and Person) classes. Below are detailed descriptions of some vital classes.

OpenPickups

This is an entity class responsible for storing all packages that are currently registered in the system. This class allows controller classes to get a list of packages with a certain status, such as a list of all packages that haven't been allocated to a courier yet, as well as to add new packages to the system.

Attributes

- PackageList: Arraylist[packages,status] - a list containing all packages currently recorded in the system together with a status unique to OpenPickups (Open, allocatedToCourier, Closed)

Operations

- getPackageList() - returns all packages regardless of their status. This will be called from multiple controller classes

- addPackage() - when a new pickup request is placed, this function will be called by the route manager to add the package to the PackageList

- getUndispatchedPackages(): Arraylist[packages] - this will return all packages with the status "Open". This is for the routeManager to plan a new route for a courier.

RouteManager

RouteManager is a control class storing all active couriers and their current locations as well as managing all operations concerned with routing of couriers, such as updating and creating routes for couriers. This includes the regular and routinely communication with the external routing system (not pictured in the class diagram) and the resulting updating of routes.

Attributes

- ActiveCouriersLocations: [String] - coordinates on all the current active couriers
- activeCouriers: [Courier] - current active couriers

Operations

- The handleExpressPickUp(): –
  called by the PickUpManager upon receiving an express pickup request. Will compute the nearest couriers using the ActiveCouriersLocations and trigger the addition of the package after a courier has requested the pickup request
- CreateRoute(): Route -
  starts the process of creating a route, by calling the “X” external system and uses it to create a route object, to then be returned to the courier app
- GetCurrentLocation(Courier): [String] -
  Returns the current location of the courier that is passed to this operation
- UpdateRoute(): Route -
  returns an updated version of the route, this is needed if an express pickup has been added to the list of packages
- UpdateAllActiveCourierLocations(): -
  updates all the current locations of the active couriers
- InformCustomersOfExpectedArrival(route:Object) -
  pushes the notification manager to send a notification of arrival time to all customers on the current route
- UpdateNavigationData () -
  calls the UpdateArrivalTimeAndNavigation() - operation of all routes on a regular basis

Pickup Manager

Pickup Manager is a control class that is responsible for handling all requests primarily concerning packages as well as triggering all operations that result from operations on packages.

Attributes

- CurrentPackage:Package –
  used to store the package the PickUpManager is currently working on in order to access package information. It is set by the PickUpManager calling OpenPickUps.getPackage(QRCode).

Operations

- RequestPickUp() -
  Called only by the CustomerApp. Responsible for triggering all operations relevant to the creation of a new pickup, including the creation of a pickup request.

- CompilePackageInformation(QRCode):String –
  Can be called by both the CustomerApp and the CourierApp using a QRCode to display relevant package data. The PickUpManager will take care of the communication with the external systems as well as extracting information from the package itself and return all information as bundle

- PickUpPackage(QRCode):boolean / deliverPackage(QRCode):boolean –
  Only called by the CourierApp to confirm the pickup/delivery of a package. They will take care of all necessary changes to the system.

- AmendPackageStatus(QRCode):boolean –
  Only called by the CourierApp during scanning at checkpoints (distribution hubs). Updates the status of a package as well as the data of the courier transporting the package.

PerformanceEvaluationManager

PerformanceEvaluationManager is a control class handling all updates to the performance rating of a courier. It receives a coustomer's rating from the customerApp and faciliates communication with other controller classes as well as with the external performance tracking system.

Operations

- RatePickUp() / rateDelivery() -
  Only called by the CourierApp in order to rate a pickup / delivery experience. Can be called independently using a QRCode or upon receiving a notification with the proposal of rating the courier. After checking that the customer hasn't already rated the service, it will take care of computing and updating the performance rating of the respective courier object.

- UpdatePunctualityAndThroughput() –
  Only called by other controller classes. Responsible for computing and updating the performance rating of the respective courier object based on data received during a pickup or delivery process.

Route

Route is an entity class that holds the information regarding the route each courier must take. This entity class is updated by the RouteManager every 5 minutes and provides an estimated arrival time for the courier and customer.

Attributes

- ArrivalTimeAtNextDestination:Arraylist [String] -
  All arrival times for all destinations along a route. Routinely updated by the Route Manager calling the updateArrivalTimeAndNavigation() operation.

- Courier:Courier - the courier currently working on the route

- NextDestinations:Arraylist[Customer] -
  a list of the next pickup-/dropoff- locations along a route

- PackageList:Arraylist[package] - a list of all packages that are part of this route

- RouteNavigation –
  Geo Information Data used by the Courier for Navigation. Routinely Updated by the RouteManager calling updateArrivalTimeAndNavigation ().

Operations

- UpdatePackageListAndDestinations() -
  Responsible for removing a package from the route after delivery

- UpdateArrivalTimeAndNavigation() -
  Called by the RouteManager on a regular basis. Responsible for updating all information concerning navigation and arrival times

- UpdateRoute() -
  Used by the Route Manager after being prompted to do so by the pickupManager after an Express Delivery PickUp was confirmed by a courier.

Dynamic Design Model
Sequence Diagrams

Figure 3 shows the process of a customer requesting a pickup. The System first gathers all relevant information, before quoting a price and taking payment. It then creates a package and returns the appropriate QR-Code to the customer.

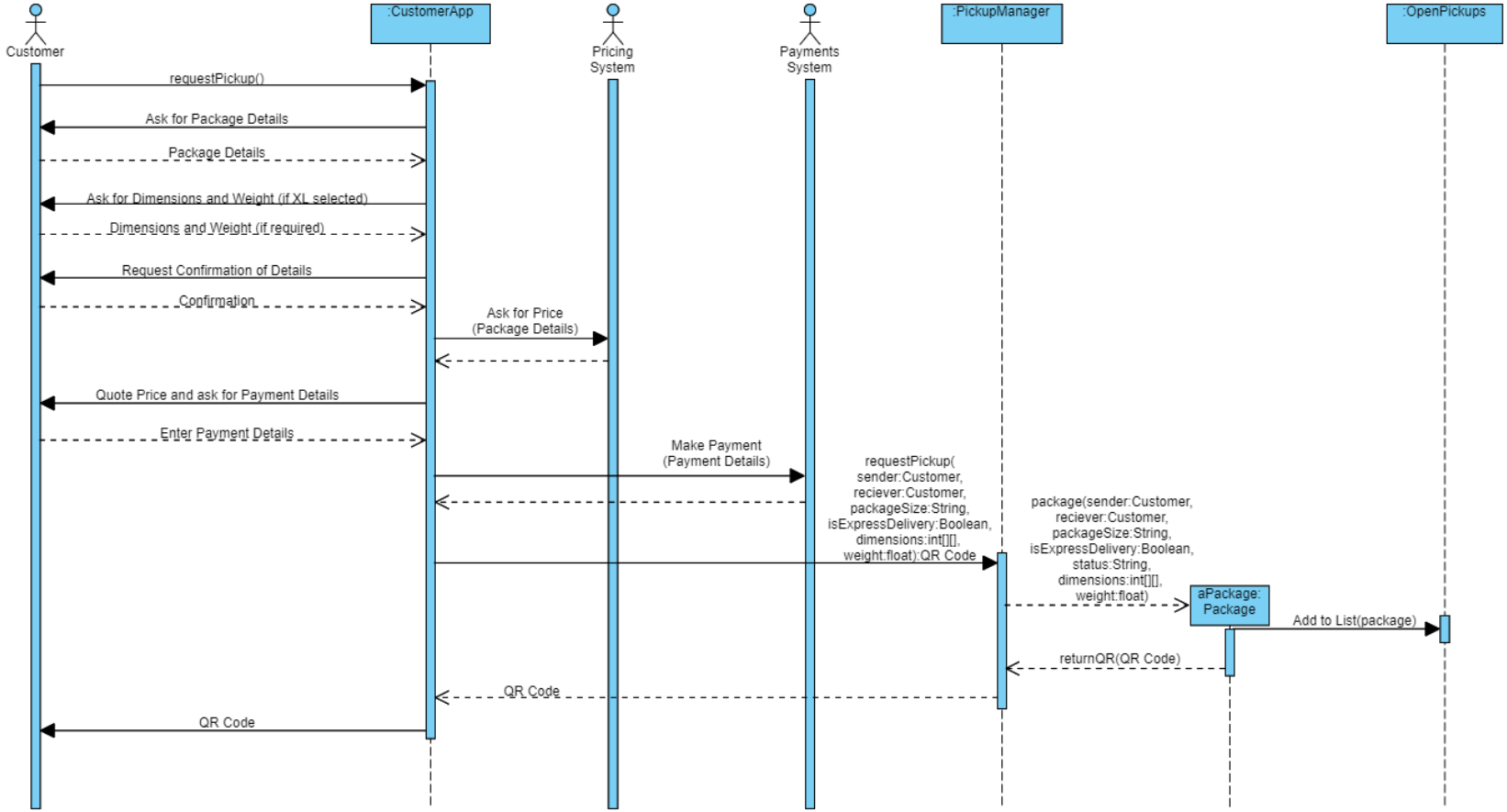

Figure 3: Sequence Diagram (Customer places Pickup Request)

Figure 4 shows the working of the system when a customer requests information of a package that is currently “Out for Delivery”. In addition to the standard information about a package, the estimated arrival time is returned to the customer.

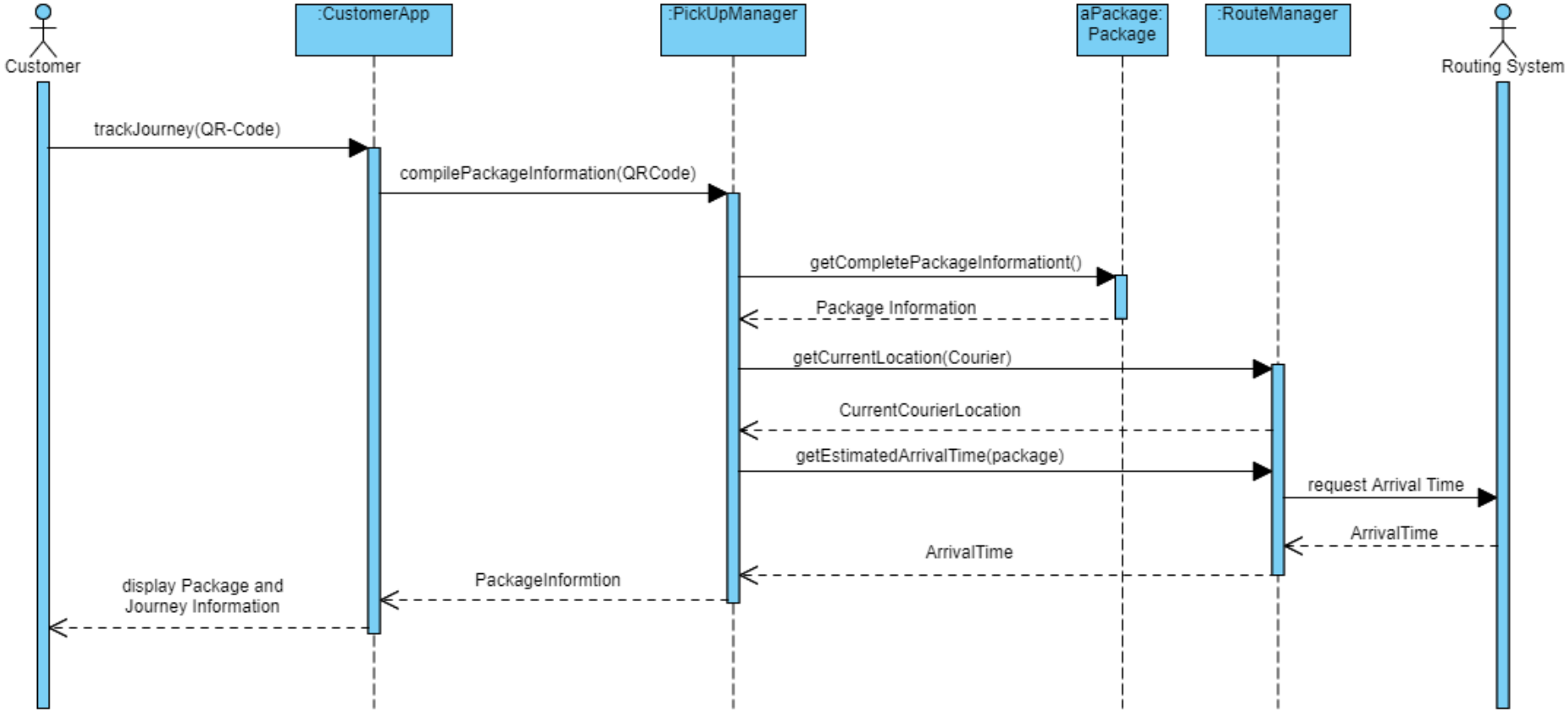

Figure 4: Sequence Diagram (Customer tracks the journey of a package)

Figure 4a shows a detail of the scenario above: each time the PickUpManager is accessed, the package is obtained from the list of open Pickups

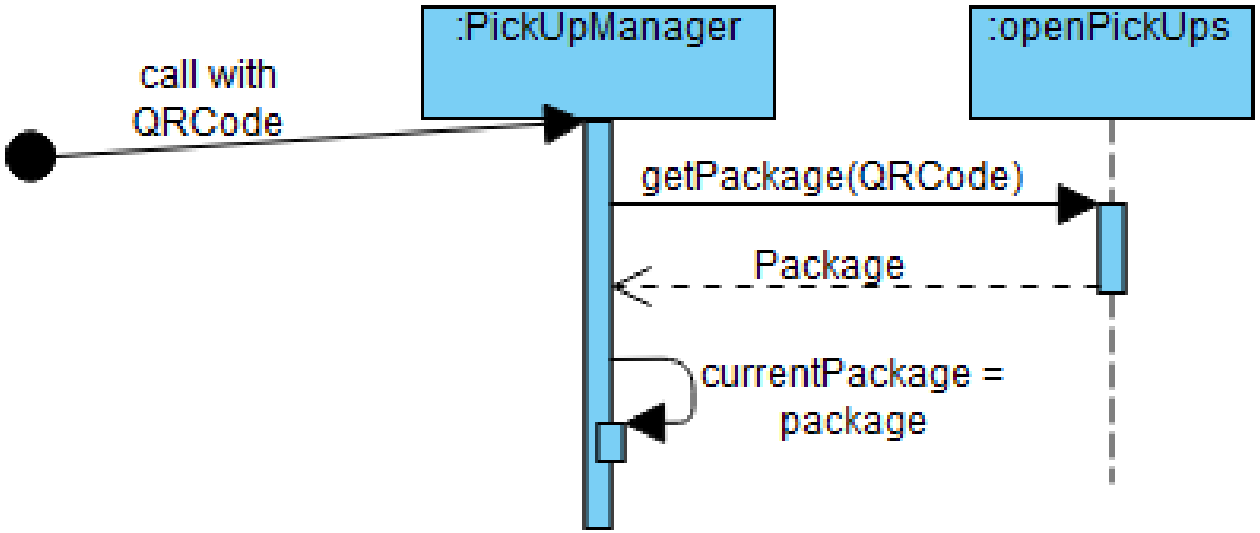

Figure 4a: Sequence Diagram (Detail)

Figure 5 shows what happens when a courier requests a new route at the beginning of his shift. After getting packages not yet allocated to a courier, the system requests a route from the external routing System and updates the courier's route before informing customers of expected arrival times.

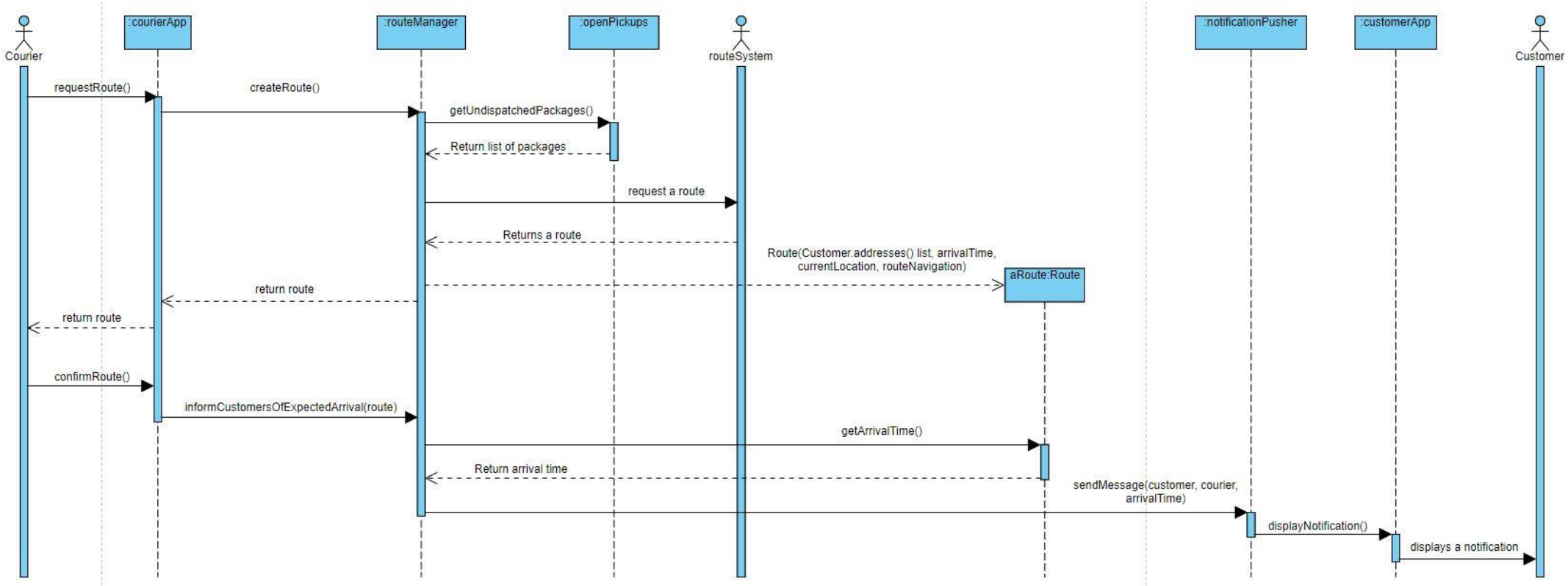

Figure 5: Sequence Diagram (courier requests a new route)

Figure 6 shows a courier dropping of a package. The route of the courier and the status of the package are updated before the arrival time is used to update the performance rating of the courier. Finally, the system sends a notification to the customer to prompt them to rate the service.

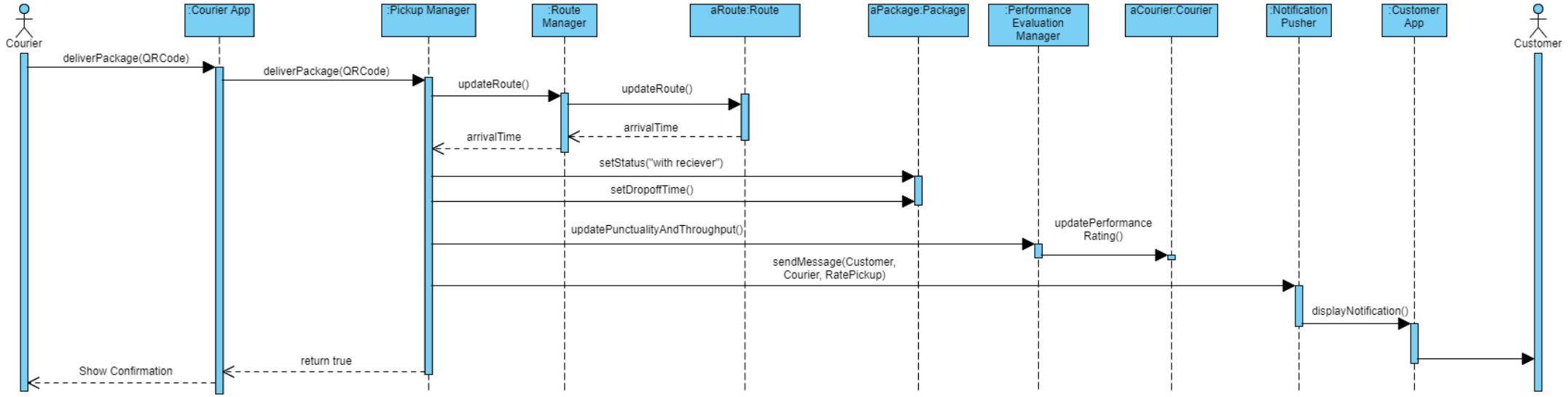

Figure 6: Sequence Diagram (A courier delivers a package)

Figure 7 shows the effects of a customer rating the courier picking up a package. In this case the customer has not rated the service yet and accepts the notification to rate the pickup**.** The system requests the new performance rating based on the customer's rating and the current performance of the courier from the performance tracking system. It updates the courier performance and thanks the customer for participation.

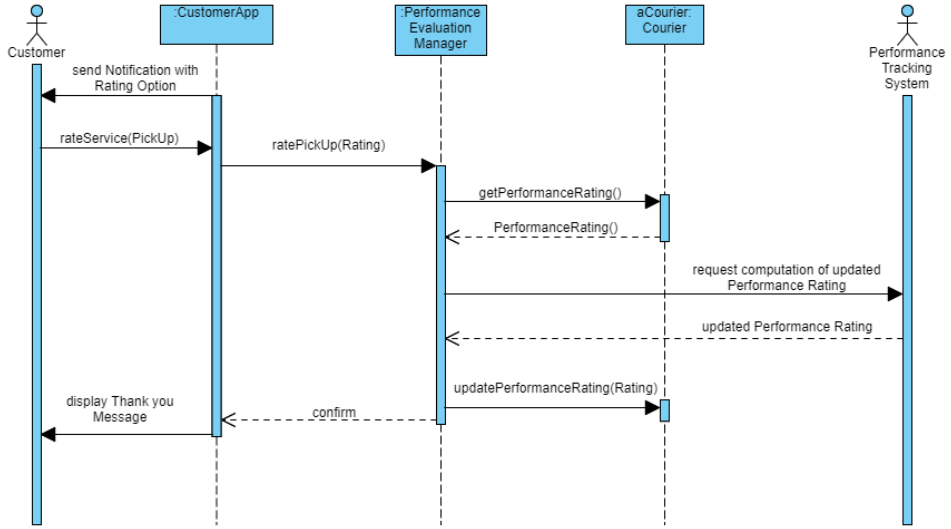

Figure 7: Sequence Diagram (A customer rates a courier picking up a package).