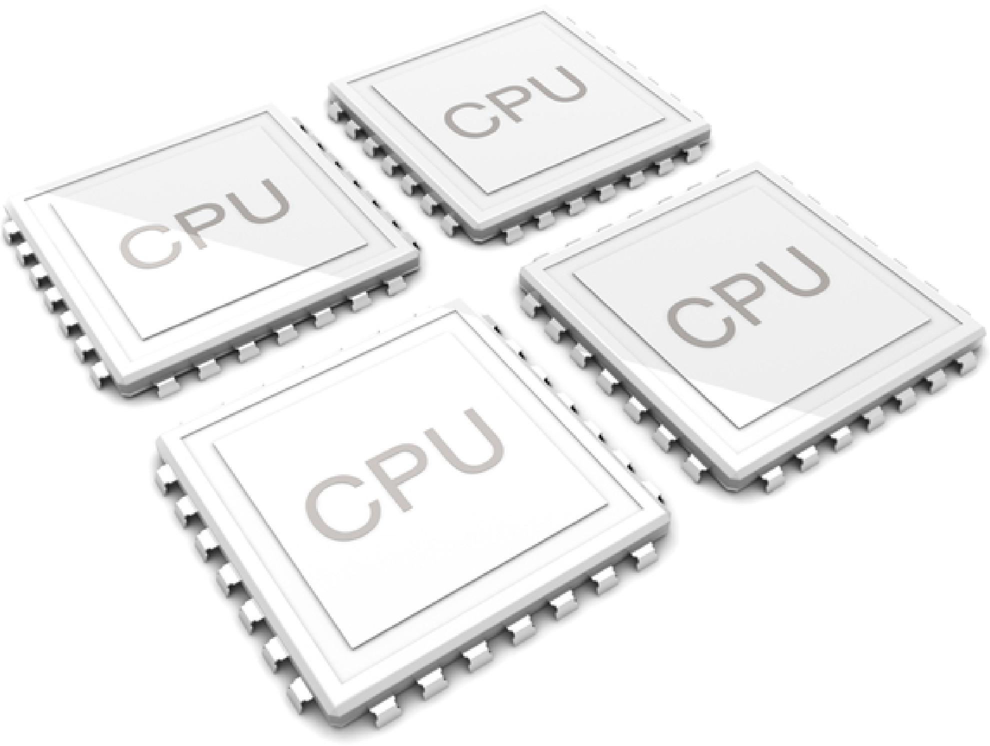
Application Design and Functional Requirements

Project: Traffic Camera Image Analysis

Client: DPSS, CSIR

Team: Quadcore Productions

Author(s):	**Student number(s):**
Mpho Baloyi	14133670
Hlengekile Jita	14077893
Mayimela Moses	14019702
Mbhele Themba	14007950



University of Pretoria, Department of Computer Science

22 October 2016

Contents

1 Introduction 3

2 Vision 3

3 Background 3

4 System Scope 4

5 Functional Requirements 4
- 5.1 Use Cases 4
- 5.2 Service Contracts 5
  - 5.2.1 Add Route 5
  - 5.2.2 Modify Route 6
  - 5.2.3 Delete Route 6
  - 5.2.4 Get Route Traffic 8
  - 5.2.5 Schedule Notification 9
  - 5.2.6 Extra Route Information 10
- 5.3 Required Functionality 11
  - 5.3.1 Add Route 11
  - 5.3.2 Modify Route 12
  - 5.3.3 Delete Route 13
  - 5.3.4 Get Route Traffic 13
  - 5.3.5 Extra Route Information 14
- 5.4 Process Specification 15
  - 5.4.1 Add Route 15
  - 5.4.2 Modify Route 15
  - 5.4.3 Get Route Traffic 15

6 Open Issues 15

Table 1: Version History

Version No.	Date	Changes	By
1	27 May 2016	Introduction, Background, Vision, Scope, Use cases	Mpho Baloyi, Hlengekile Jita, Moses Mayimela, Themba Mbhele
2	29 July 2016	Updated Scope, Updated Use Cases, Added Service Contracts, Added Required Functionality, Added Process Specification	Mpho Baloyi, Hlengekile Jita, Moses Mayimela, Themba Mbhele
3	22 October 2016	Added Extra Route Info Use Case, Added Captions and Figure References, Changed Open Issues	Mpho Baloyi, Hlengekile Jita, Moses Mayimela, Themba Mbhele

1 Introduction

This document describes the functional requirements and application design of the Traffic Camera Image Analysis System. The target system will run on a web server and be accessed by users on an Android Application which will provide the users with the necessary functionality to access real-time traffic information that assists them with things such as avoiding traffic and choosing the best alternative routes.

In this specification, we cover the use cases, their service contracts, required functionality and process specifications of the target system. The above, will be built on over time as the software is developed, as we are following an agile development methodology. In addition the domain model of the application as a whole will be provided.

2 Vision

For this project we aim to achieve a system that makes use of images obtained from highway cameras to provide users with up-to-date real-time traffic information. The system should simplify the user's travels by providing traffic information and notifying them before they depart of traffic conditions, calculating arrival times based on traffic conditions and help them select the most suitable route for their trips using the traffic information and additional metrics. Our vision for the target system is that it should be reliable and perform relatively quickly, both for user satisfaction and in order to be the an up-to-date traffic information system.

3 Background

As a commuter, traffic is something that is a part of everyday life, and it is not one of the more pleasurable aspects of life. Already there is software in place that assists us in dealing with this problem, such as Google Maps. This software uses the crowd-sourcing of GPS data in order to provide their up-to-date traffic information.

In our alternative system, we take an image analysis approach to solve the same problem. We will make use of the publicly available SANRAL highway cameras to get images. Processing these images we will perform image analysis and determine the traffic conditions in the area of a camera. Using this information we will be able to generate information pertaining to user specified routes in order to provide the information necessary to help them avoid traffic and choose the most suitable routes in order to do so.

4 System Scope

The purpose of this system is to provide the user with traffic information based on image analysis from images accessed from highway cameras. The system will need to provide the user with services that allow the achievement of this goal.

The scope of the system which is depicted below in Figure 1 includes the tasks the user can access such as add route, modify route, delete route and get route traffic. This document will further describe each of these tasks as system use cases in the functional requirements section.

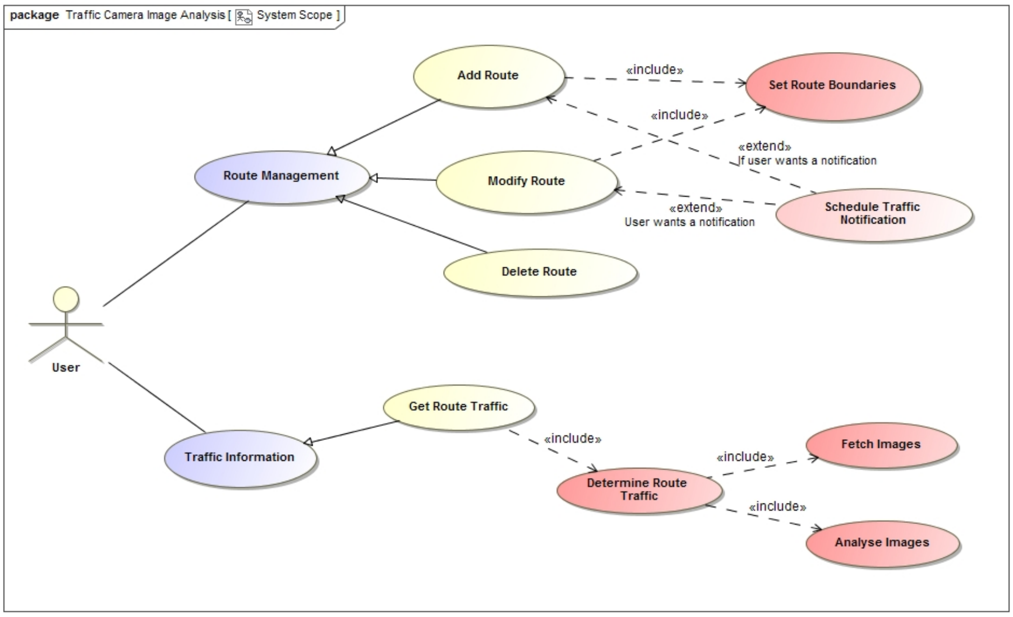


Figure 1: System Scope

5 Functional Requirements

5.1 Use Cases

The use cases for our system are described below. These use cases are the services that provide the user of the application some value. The following uses are listed in order of prioritization:

- Add Route (Critical)

- Get Route Traffic (Critical)
- Modify Route (Critical)
- Delete Route (Critical)
- Schedule Notification (Important)
- Extra Route Information (Nice To Have)

5.2 Service Contracts

For each of the above services described above, there are pre-conditions that must be met for the service to be carried out and post-conditions that are met to indicate the successful completion of a service. In addition the service contracts indicate what exceptions will be thrown in the case that a service fails.

5.2.1 Add Route

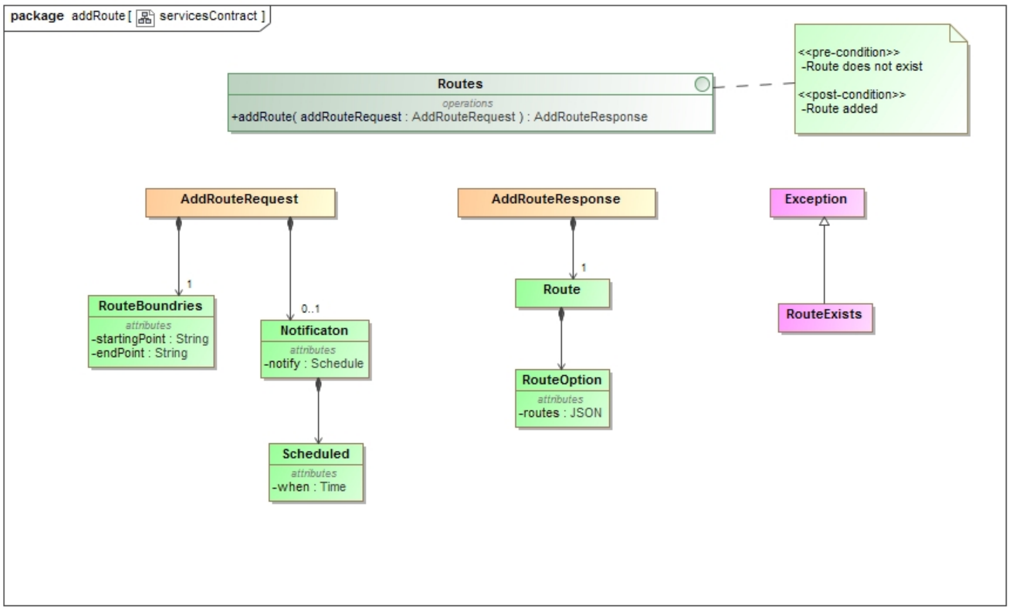


Figure 2: Service Contract for Add Route Service

Pre-Condition: Valid route boundaries.
Route does not already exist.

Post-Condition: New route is created and stored.

The Add Route service contract depicted above in Figure 2 shows the use case's pre and post conditions. This use case creates a new route and stores it so that it can be used in requests for traffic information. As a pre-condition it first checks that the route to be created is valid and does not already exist. Thereafter it creates and stores the route, meeting the post-condition. In the case that the route exists an exception is raised.

5.2.2 Modify Route

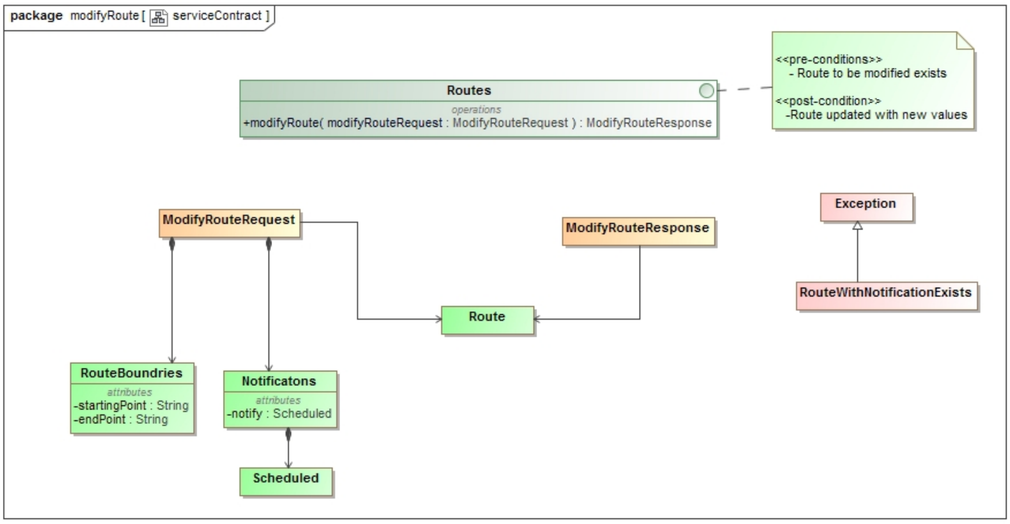


Figure 3: Service Contract for Modify Route Service

Pre-Condition: Route exists.
Post-Condition: Route is modified and the modified route is stored.

This use case makes changes to existing routes and stores the changes. The service contract depicted above in Figure 3 shows the use case's pre and post conditions. This use case first checks if the route exists, if so it then makes the necessary changes to the route and then stores the route. If the route does not exist an exception is raised.

5.2.3 Delete Route

Pre-Condition: Route exists.
Post-Condition: Route is removed

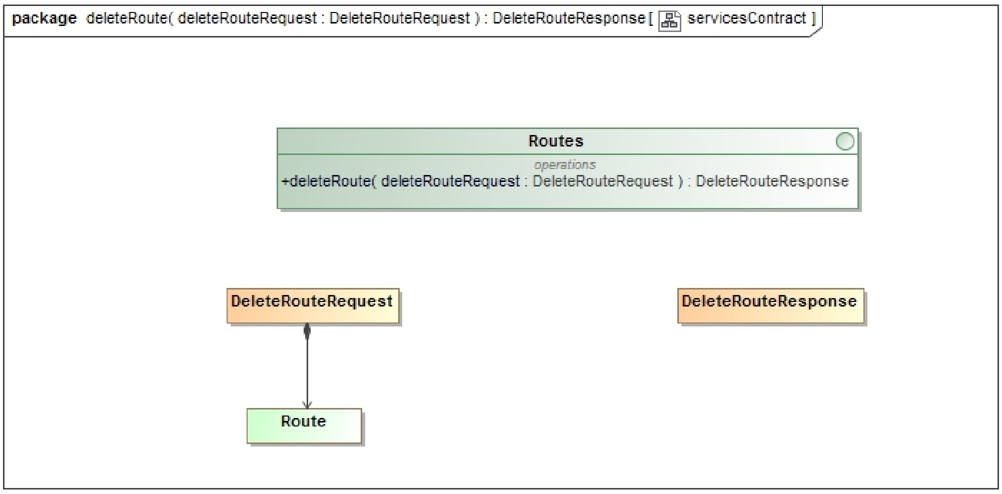

package deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse [servicesContract]
Routes
operations
+deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse
DeleteRouteRequest
DeleteRouteResponse
Route

Figure 4: Service Contract for Delete Route Service

This use case, whose service contract is depicted above in Figure 4, removes an existing route. Before it does this it checks if the route exists in storage then removes it. In the case that the route does not exist an exception is raised.

5.2.4 Get Route Traffic

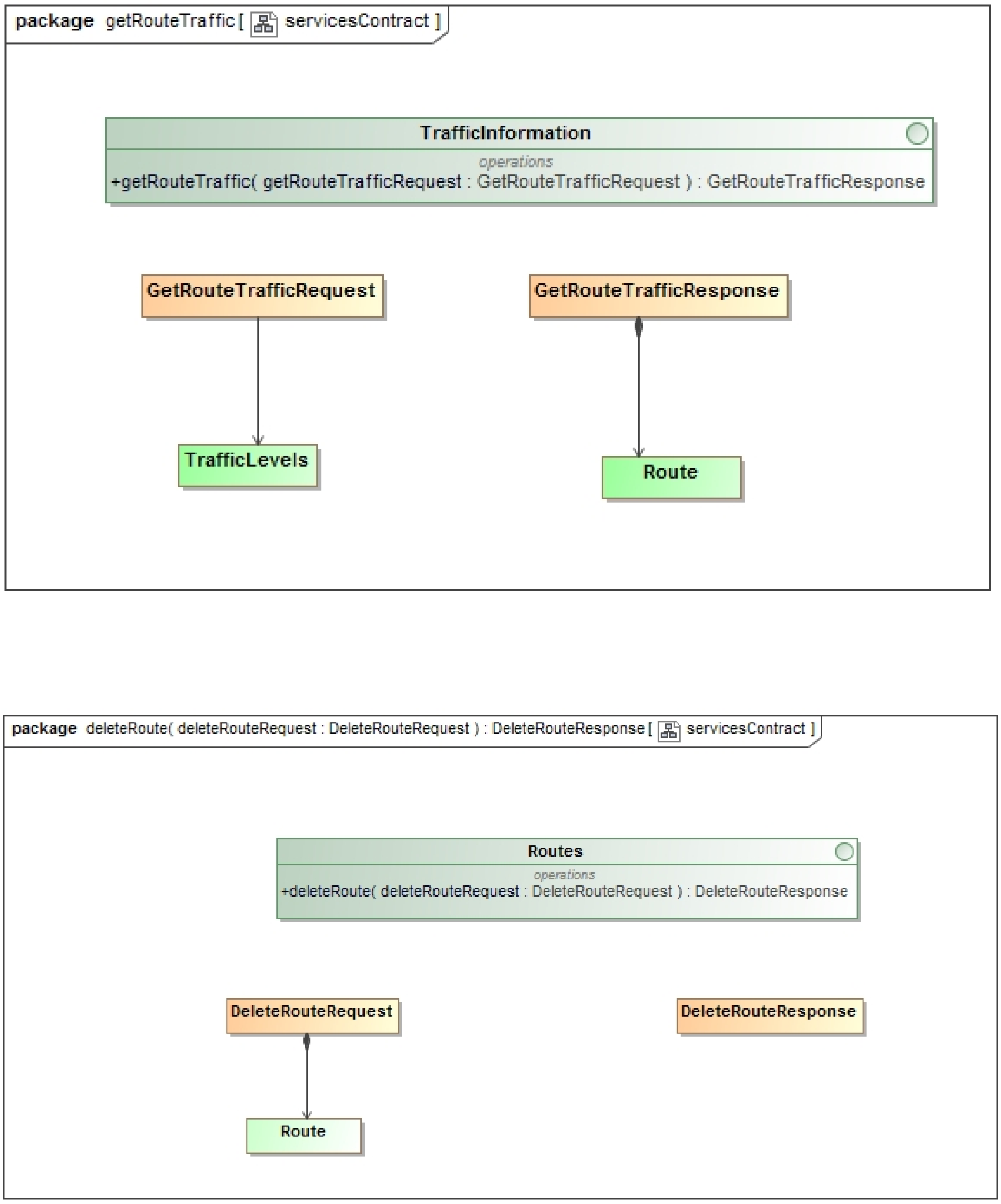
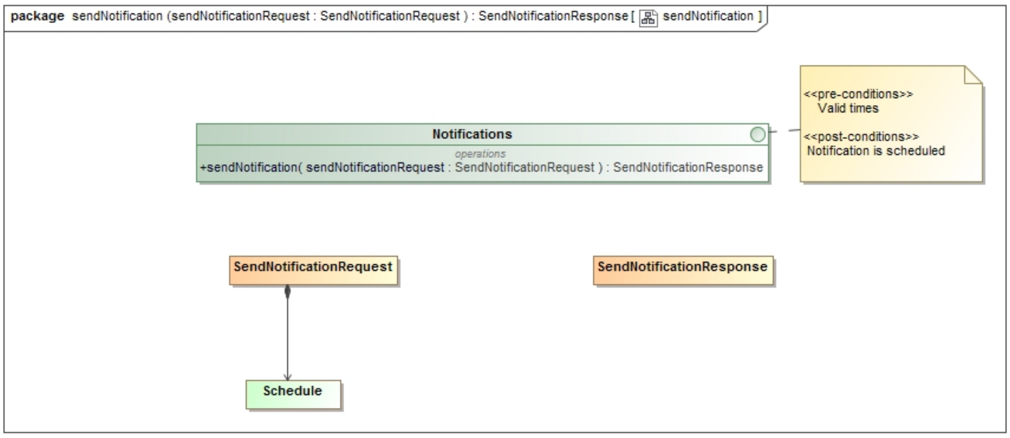

package getRouteTraffic [servicesContract]
TrafficInformation
operations
+getRouteTraffic(getRouteTrafficRequest : GetRouteTrafficRequest) : GetRouteTrafficResponse
GetRouteTrafficRequest
GetRouteTrafficResponse
TrafficLevels
Route
package deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse [servicesContract]
Routes
operations
+deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse
DeleteRouteRequest
DeleteRouteResponse
Route

Figure 5: Service Contract for Get Route Traffic Service

Pre-Condition: Route consists of camera locations.
Camera locations on route has valid images.
Post-Condition: Traffic information for route is generated.

The Get Route Traffic service contract depicted above in Figure 5 shows the use case's pre and post conditions. This use case generates traffic information based on a route provided with the request. To do this it first checks if the route covers any of the camera locations, if not no traffic information is generated and an exception is raised. Otherwise it continues and retrieves the images for the camera's within the route, thereafter it checks if these images are valid, if all are invalid, no traffic information is generated, it then performs image analysis on the valid images and determine the traffic levels and generates traffic information for the route.

5.2.5 Schedule Notification


package sendNotification (sendNotificationRequest : SendNotificationRequest) : SendNotificationResponse [sendNotification]
Notifications
operations
+sendNotification(sendNotificationRequest : SendNotificationRequest) : SendNotificationResponse
<<pre-conditions>>
Valid times
<<post-conditions>>
Notification is scheduled
SendNotificationRequest
SendNotificationResponse
Schedule

Pre-Condition: Valid times
Post-Condition: Notification is scheduled

This use case allows the user to schedule notifications of traffic information at specific times. It's service contract is depicted in Figure 6. This use case is completed by allowing the user to specify a time to be notified about a route when adding or modifying it. This time is then used to schedule automated traffic information requests.

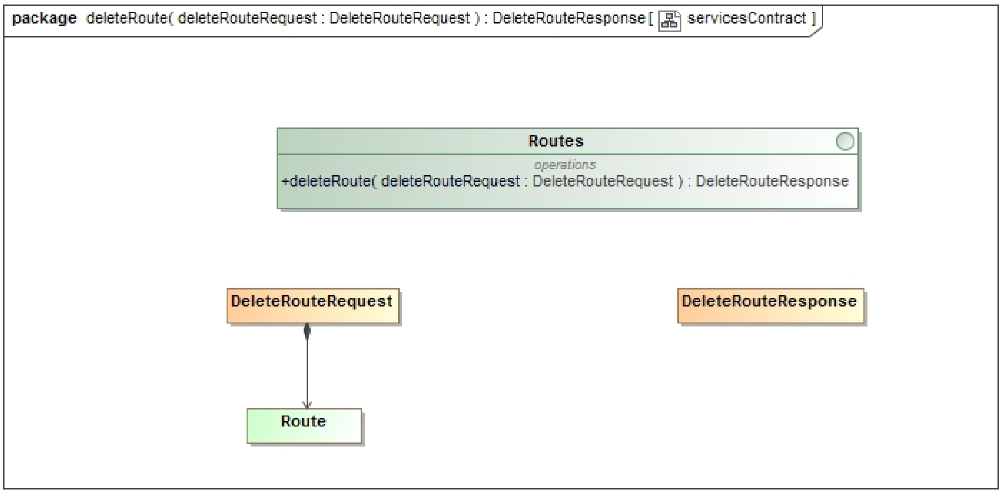

package deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse [servicesContract]
Routes
operations
+deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse
DeleteRouteRequest
DeleteRouteResponse
Route

Figure 6: Service Contract for Schedule Notification Service

5.2.6 Extra Route Information

images/scExtra_Route_Info.jpg

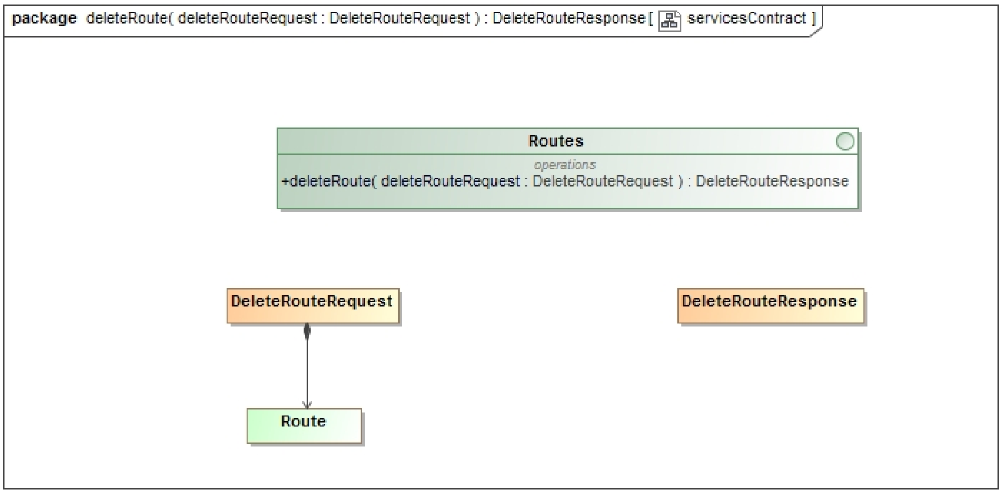

package deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse [servicesContract]
Routes
operations
+deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse
DeleteRouteRequest
DeleteRouteResponse
Route

Figure 7: Service Contract for Extra Route Information Service

Pre-Condition: Get Route Traffic use case succeeds.
Route has roadworks and incident information.
Post-Condition: Extra traffic information for route is generated.

The Extra Route Information service contract depicted in Figure 7 shows what pre- and post- conditions must be met to recieve extra route information, specifically information about roadworks and incidents. This service is completed if and only if the Get Route Traffic service is completed as an addittional add on. Before the service can run, the system also has to check if such information for that route is available. If the service is able to run successfuly, the post-condition is met.

5.3 Required Functionality

The use cases make use of other lower level functions in order to achieve their purpose. In the below diagrams for each high level use case, the lower level use cases that are required are described.

5.3.1 Add Route

The Add Route use case makes use of the following lower level functions depicted in Figure 8 above to add a route to the system:

- **getRoute -** This checks if the route exists already or not.

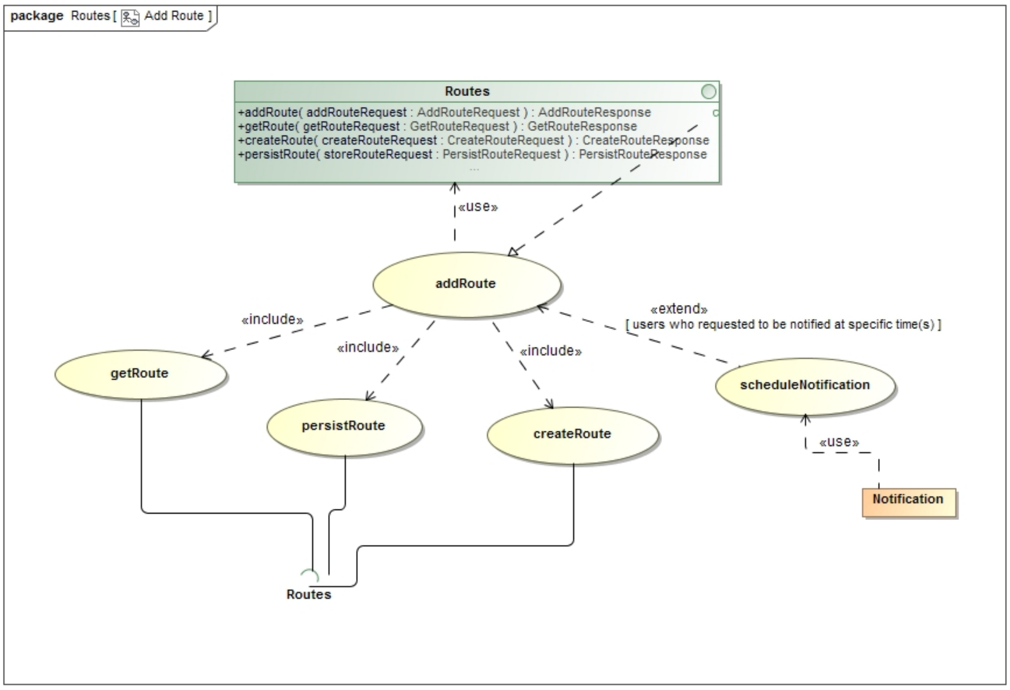

package Routes [Add Route]
Routes
+addRoute(addRouteRequest : AddRouteRequest) : AddRouteResponse
+getRoute(getRouteRequest : GetRouteRequest) : GetRouteResponse
+createRoute(createRouteRequest : CreateRouteRequest) : CreateRouteResponse
+persistRoute(storeRouteRequest : PersistRouteRequest) : PersistRouteResponse
...
«use»
addRoute
«include»
«include»
«include»
«extend»
[users who requested to be notified at specific time(s)]
getRoute
persistRoute
createRoute
scheduleNotification
«use»
Notification
Routes

Figure 8: Use Case Diagram for the Add Route Service

- **createRoute -** This creates a route, or a set of directions, based on the route boundaries (departure and destination points).
- **persistRoute -** This stores the route so that it is available for later use.

5.3.2 Modify Route

The Modify Route use case makes use of the following lower level functions depicted in Figure 9 above to make changes to an already existing route.

- **getRoute -** This checks if the route exists already.
- **createRoute -** This creates a route, or a set of directions, based on the route boundaries (departure and destination points). Only if the changes are in the route boundaries.
- **persistRoute -** This stores the modified route.

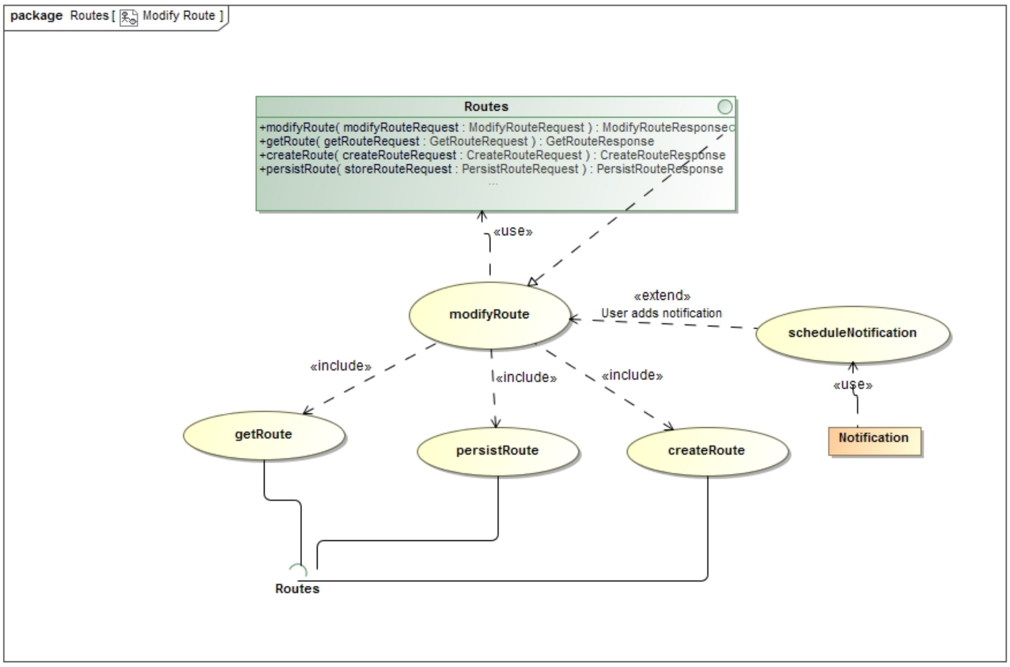

package Routes [Modify Route]
Routes
+modifyRoute(modifyRouteRequest : ModifyRouteRequest) : ModifyRouteResponse
+getRoute(getRouteRequest : GetRouteRequest) : GetRouteResponse
+createRoute(createRouteRequest : CreateRouteRequest) : CreateRouteResponse
+persistRoute(storeRouteRequest : PersistRouteRequest) : PersistRouteResponse
...
«use»
modifyRoute
«extend»
User adds notification
scheduleNotification
«include»
«include»
«include»
«use»
getRoute
persistRoute
createRoute
Notification
Routes

Figure 9: Use Case Diagram for the Modify Route Service

5.3.3 Delete Route

The Delete Route use case makes use of the following lower level functions depicted in Figure 10 above to assist in removing the route from storage:

- **getRoute -** This checks if the route exists already .

5.3.4 Get Route Traffic

The Get Route Traffic use case makes use of the following lower level functions, depicted in Figure 11 above, both within and outside the Traffic Information module to determine the traffic of a route and send it back to the user:

- **checkForCameras -** This function checks if the route passes b y any of the available cameras.
- **fetchImage -** This function fetches an image from the camera at the specified location.
- **analyseImage -** This function performs image analysis in order to determine the traffic level in that location

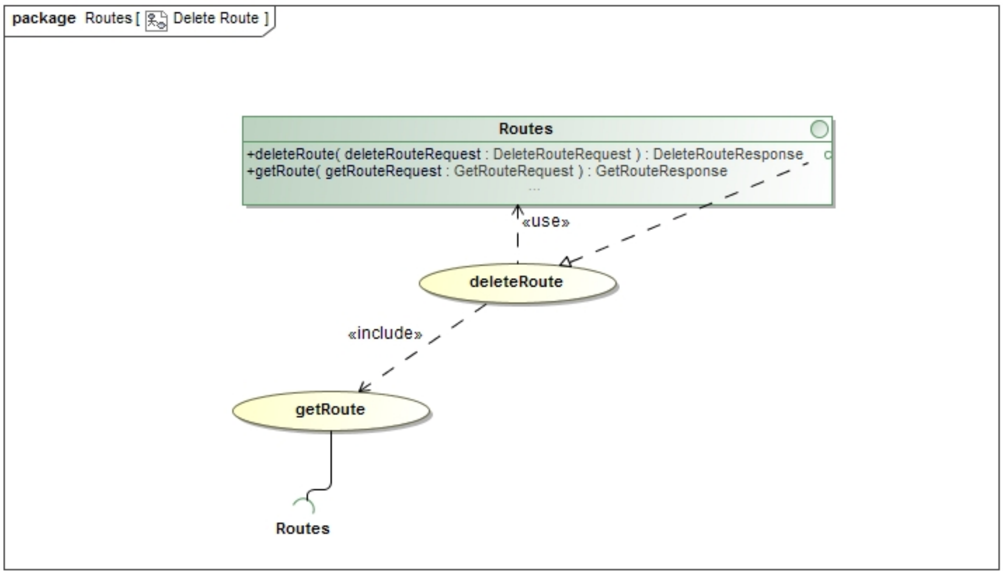

package Routes [Delete Route]
Routes
+deleteRoute(deleteRouteRequest : DeleteRouteRequest) : DeleteRouteResponse
+getRoute(getRouteRequest : GetRouteRequest) : GetRouteResponse
...
«use»
deleteRoute
«include»
getRoute
Routes

Figure 10: Use Case Diagram for the Delete Route Service

- **composeTrafficInfo -** This function composes all the traffic levels that make up the route and constructs a preliminary ”traffic info report”.
- **sendTrafficInformation -** This function creates a traffic information report and sends it to the user who requested it.
- **storeTrafficLevels -** The traffic levels are stored for any other user to be able to access it if their request requires similar information.

5.3.5 Extra Route Information

The Extra Route Information use case makes use of the following lower level functions, depicted in Figure 12 above, to get roadworks and incident information for the route:

- **fetchRoadworksInfo -** This function fetches the roadworks information, if any, of the specified route.
- **fetchIncidentInfo -** This function fetches he incident information, if any, of the specified route.
- **sendTrafficInformation -** This function sends the traffic information to the user.

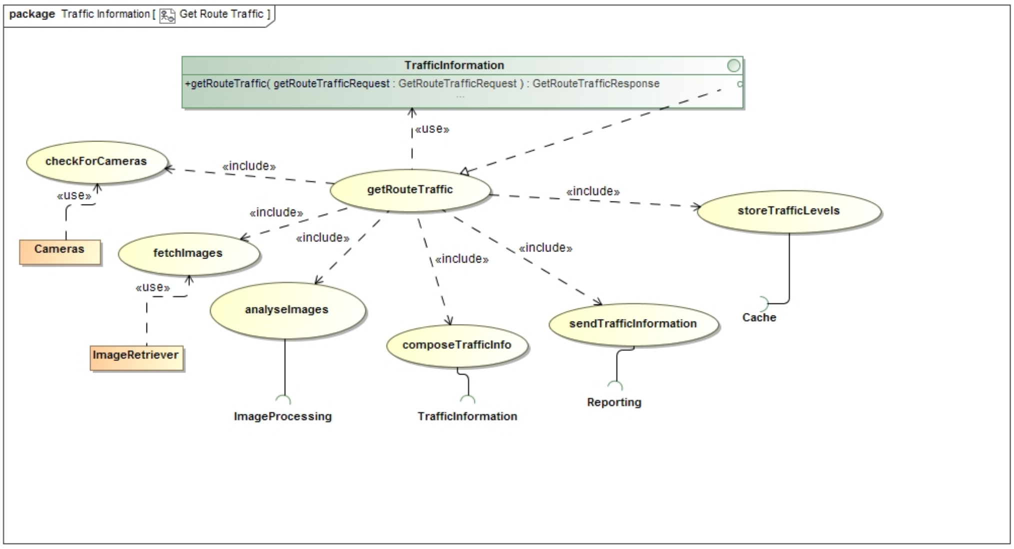


Figure 11: Use Case Diagram for the Get Route Traffic Service

- **storeTrafficInfo -** The traffic information is stored in cache for any other user who makes a similar request.

5.4 Process Specification

This section describes the process for completing the Add Route, Modify Route and Get Route Traffic Sections.

5.4.1 Add Route

The process for adding a route is shown below in Figure 13.

5.4.2 Modify Route

The process for modifying a route is shown below in Figure 14.

5.4.3 Get Route Traffic

The process for getting traffic information is shown below in Figure 15.

6 Open Issues

Possible future work to be done on the system:

- **Camera's from more sources** - The system only has access to SANRAL highway cameras. This limits the scope of the application. If other sources of camera's are found further work could include improving the analysis by having more images.

images/Extra_Route_Info.jpg

Figure 12: Use Case Diagram for the Extra Route Information Service

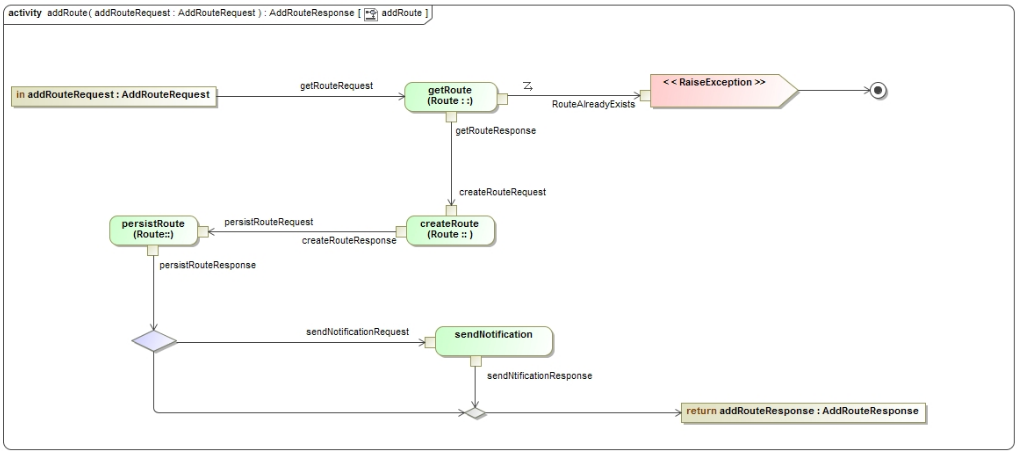


Figure 13: Process Specification Diagram for the Add Route Service

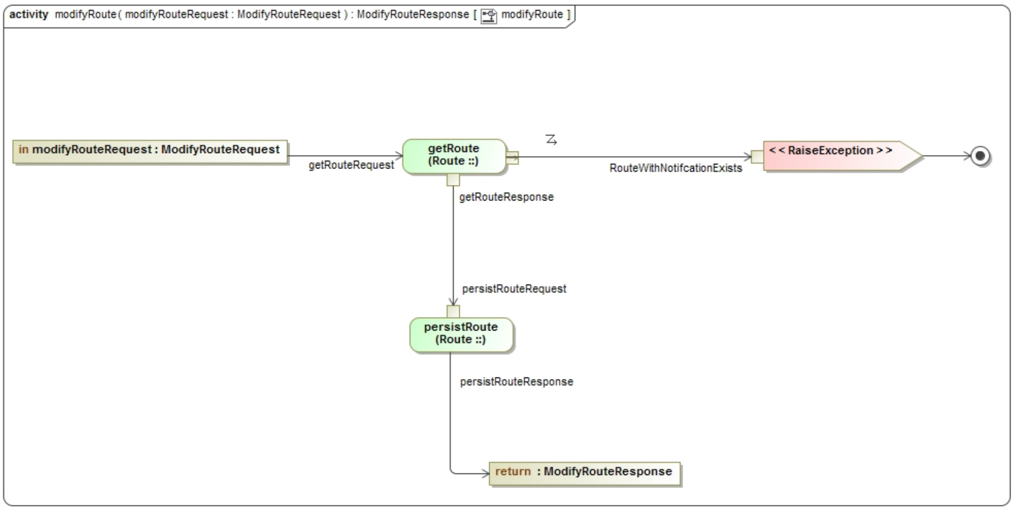

activity modifyRoute (modifyRouteRequest : ModifyRouteRequest) : ModifyRouteResponse [modifyRoute]
in modifyRouteRequest : ModifyRouteRequest
getRouteRequest
getRoute (Route ::)
RouteWithNotifcationExists
< < RaiseException > >
getRouteResponse
persistRouteRequest
persistRoute (Route ::)
persistRouteResponse
return : ModifyRouteResponse

Figure 14: Process Specification Diagram for the Modify Route Service

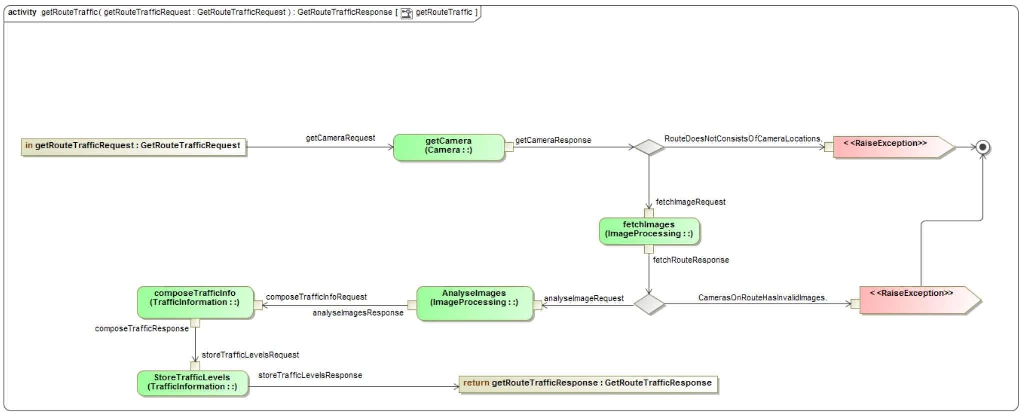

activity getRouteTraffic(getRouteTrafficRequest : GetRouteTrafficRequest) : GetRouteTrafficResponse [getRouteTraffic]
in getRouteTrafficRequest : GetRouteTrafficRequest
getCameraRequest
getCamera
(Camera : :)
getCameraResponse
RouteDoesNotConsistsOfCameraLocations.
< <RaiseException>>
fetchImageRequest
fetchImages
(ImageProcessing : :)
fetchRouteResponse
composeTrafficInfo
(TrafficInformation : :)
composeTrafficInfoRequest
analyseImagesResponse
AnalyseImages
(ImageProcessing : :)
analyseImageRequest
CamerasOnRouteHasInvalidImages.
< <RaiseException>>
composeTrafficResponse
storeTrafficLevelsRequest
StoreTrafficLevels
(TrafficInformation : :)
storeTrafficLevelsResponse
return getRouteTrafficResponse : GetRouteTrafficResponse

Figure 15: Process Specification Diagram for the Get Route Traffic Service