

# Application Requirements and Design

---

## Real-time Geospatial Data Processor and Visualiser

*Client: Werner Raath*

---

C O E U S



### *Members:*

Nsovo Baloyi 12163262  
Maluleki Nyuswa 13040686  
Keletso Molefe 14222583  
Kamogelo Tswene 12163555

22 October 2016

v0.4

# Contents

<b>Document History</b>	<b>2</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Vision</b>	<b>3</b>
<b>3 Background</b>	<b>3</b>
<b>4 Functional Requirements and Application Design</b>	<b>4</b>
4.1 System Overview . . . . .	4
4.2 Use Case Prioritization . . . . .	4
<b>5 Modular System</b>	<b>6</b>
5.1 Disaster Management Module . . . . .	6
5.1.1 Scope . . . . .	6
5.1.2 viewDisaster . . . . .	7
5.1.3 queryDisaster . . . . .	9
5.1.4 notifyNewDisaster . . . . .	11
5.1.5 trackDisaster . . . . .	13
5.1.6 addDisasterType . . . . .	15
5.1.7 modifyDisasterType . . . . .	17
5.2 Climate Management Module . . . . .	19
5.2.1 Scope . . . . .	19
5.2.2 viewWeather . . . . .	20
5.2.3 queryForecast . . . . .	22
5.3 User Module . . . . .	25
5.3.1 Scope . . . . .	25
5.3.2 registerUser . . . . .	25
5.3.3 login . . . . .	27
5.3.4 logout . . . . .	30
5.3.5 addPreferences . . . . .	32
5.3.6 editPreferences . . . . .	34
5.3.7 viewPreferences . . . . .	36
5.4 Domain Model . . . . .	39
<b>6 Open Issues</b>	<b>39</b>

## Document History

Version	Date	Changed By	Summary
v0.1	27 May 2016	Nsovo Baloyi Maluleki Nyuswa Keletso Molefe Kamogelo Tswene	First Draft
v0.2	29 July 2016	Maluleki Nyuswa Kamogelo Tswene	Second Draft
v0.3	9 September 2016	Nsovo Baloyi Maluleki Nyuswa Keletso Molefe Kamogelo Tswene	Third Draft
v0.4	22 October 2016	Nsovo Baloyi Maluleki Nyuswa Keletso Molefe Kamogelo Tswene	Final Draft

# **1 Introduction**

The purpose of this document is to outline the core functionality of the Real-time Geospatial Data Processor and Visualiser system in a technology neutral design specification. The design is not necessarily a depiction of what the final system will look like as during the development of the system it is possible that new functionalities are added, or old ones removed.

In this document we will thoroughly discuss and layout the project's functional requirements to provide a clear overview of the system as a whole. An agile approach is being followed which involves an interactive and incremental method of managing and designing the system as described by the scrum methodology.

# **2 Vision**

The aim of this project is to provide a fully functional system to facilitate the processing and visualisation of geospatial data in real-time<sup>1</sup>, including but not limited to climate and fire disasters. The system should be maintainable, with detailed supporting documentation for the Real-time Geospatial Data Processor and Visualiser system. When implemented, the system should be able to store, process and visualise large amounts of data in real-time.

# **3 Background**

In the business of disaster management, being able to monitor a disaster's spread and change in near-real time is essential for assessing damages e.g. wildfire burned area, flooded area, etc. and risk to human lives and assets. There are numerous mature open-source technology for processing the voluminous real-time data, but systems to visualise this data is woefully incomplete and its development remains at an infancy stage. Thus we need to contribute to this quest for knowledge. The need for systems such as the one suggested is very high. The system can save lives and serve as a tool to do research amongst other things.

---

<sup>1</sup>real-time

## 4 Functional Requirements and Application Design

The purpose of this section is to outline the functional requirements and application design. The application requirements and design specify the core functional requirements, i.e. the user's requirements around use cases and not secondary functional requirements arising from realizing non-functional requirements.

### 4.1 System Overview

Figure 1 shows a high-level overview of the system.

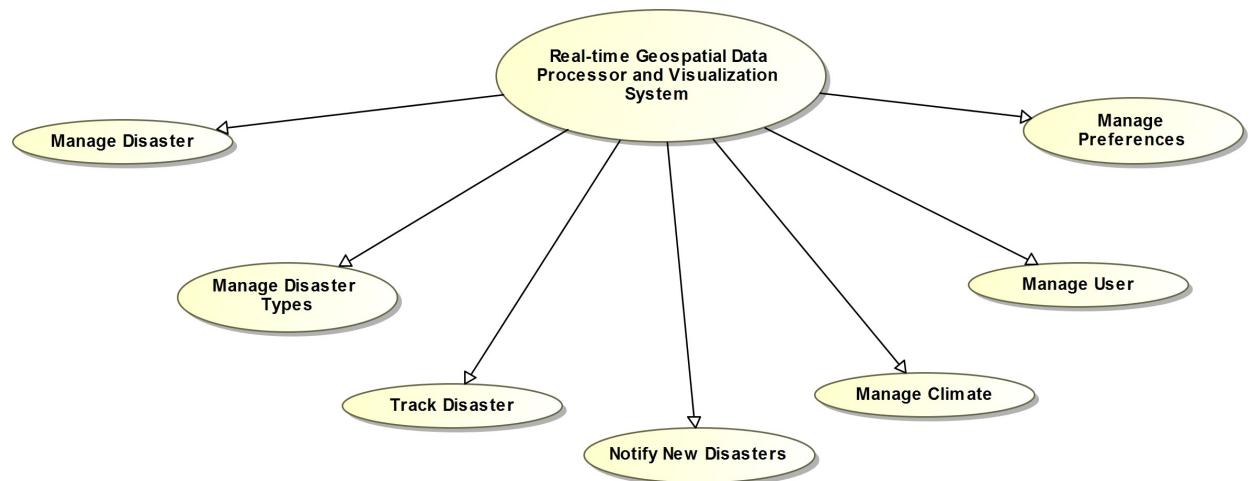


Figure 1: Overview of the System

### 4.2 Use Case Prioritization

The system use cases are characterised as follows:

- **Manage Disaster - Critical**

This use case is critical since the main functionality of the system requires disaster management.

- **Manage Disaster Types - Important**

An authorized user should be able to add a new disaster type or remove a redundant disaster type.

- **Track Disaster - Important**  
Users need to be able to track an active disaster that has occurred for feedback purposes.
- **Notify New Disaster - Important**  
A user that is currently online should be able to get instantaneous notification of a newly occurring disaster.
- **Manage Climate - Important**  
A user should be able to view normal weather climate at any time they wish to.
- **Manage User - Important**  
A user should be able to register with the system because as a registered user, once logged into the system they would be able to manage their weather and disaster preferences as well as log out of the system once done. An authorized user (admin) once logged in is able to access and/or modify relevant system data.
- **Manage Preferences - Important**  
A registered user should be able to add, edit or view their weather and disaster preferences.

## 5 Modular System

### 5.1 Disaster Management Module

#### 5.1.1 Scope

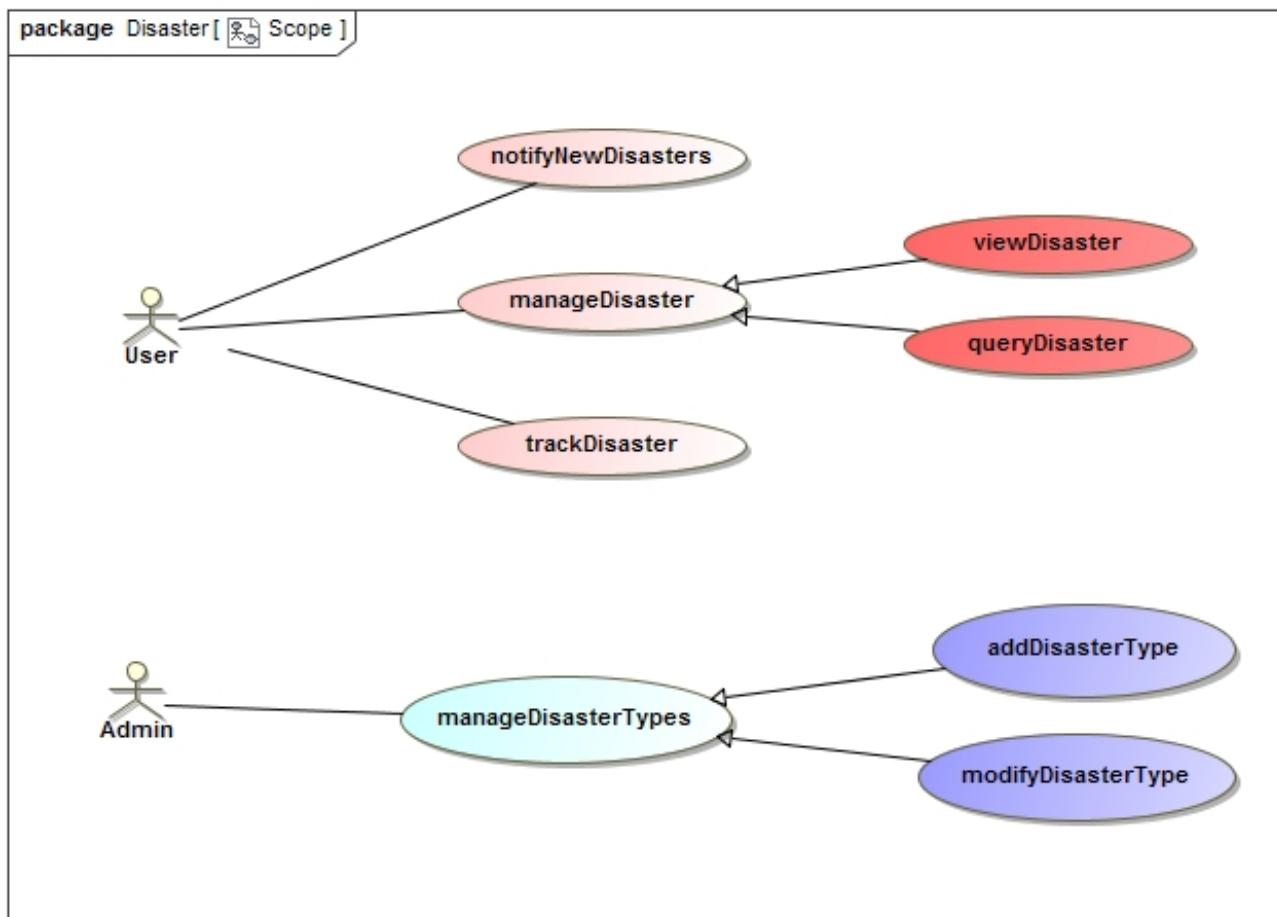


Figure 2: The scope of functionality required from the disaster module

Admin can add and/or modify the different kinds of disaster types that are required whilst users are able to track and manage disasters as well as get notifications for new disasters.

### 5.1.2 viewDisaster

A user can view disasters by going on to the disasters tab on the system and viewing all the disasters, active or not, that are occurring throughout the world. Below are the service contract, activity diagram and functional requirements diagram for viewDisaster.

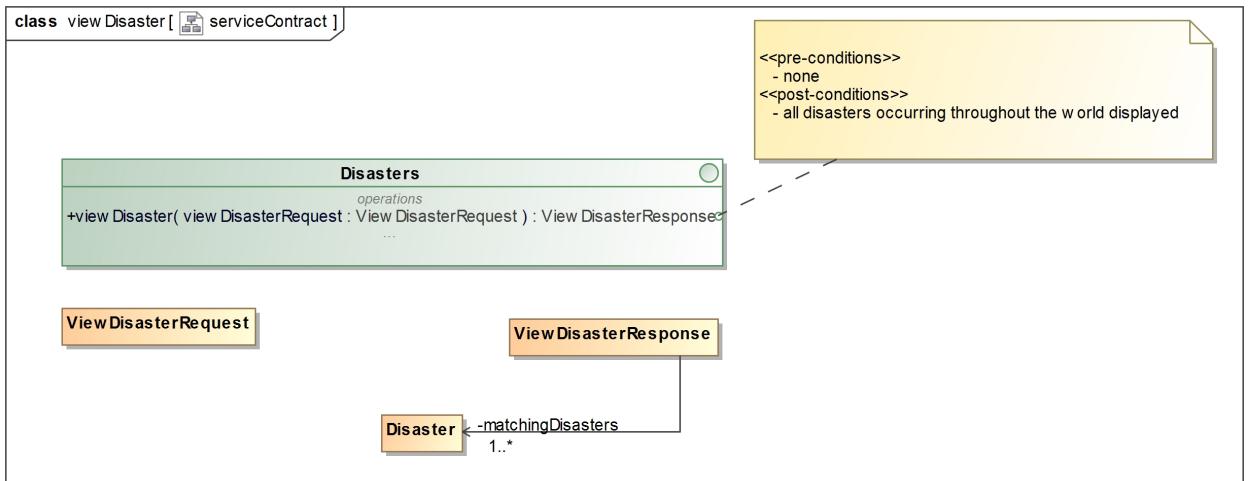


Figure 3: The service contract for `viewDisaster`

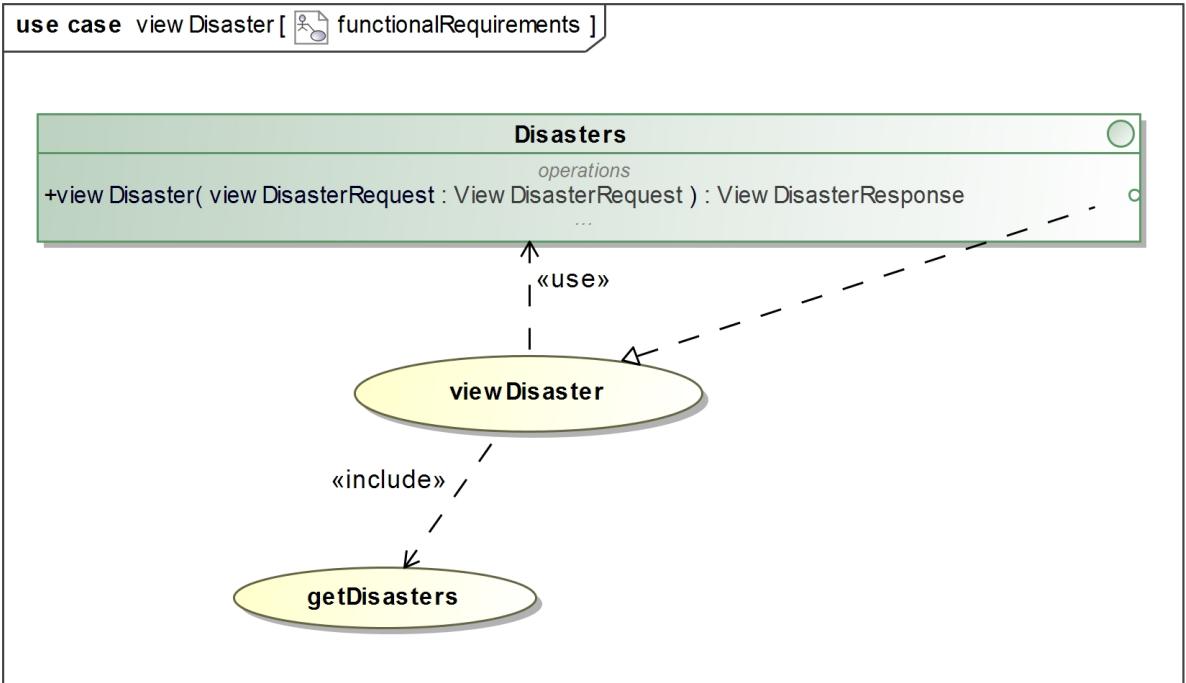


Figure 4: The functional requirements diagram for viewDisaster

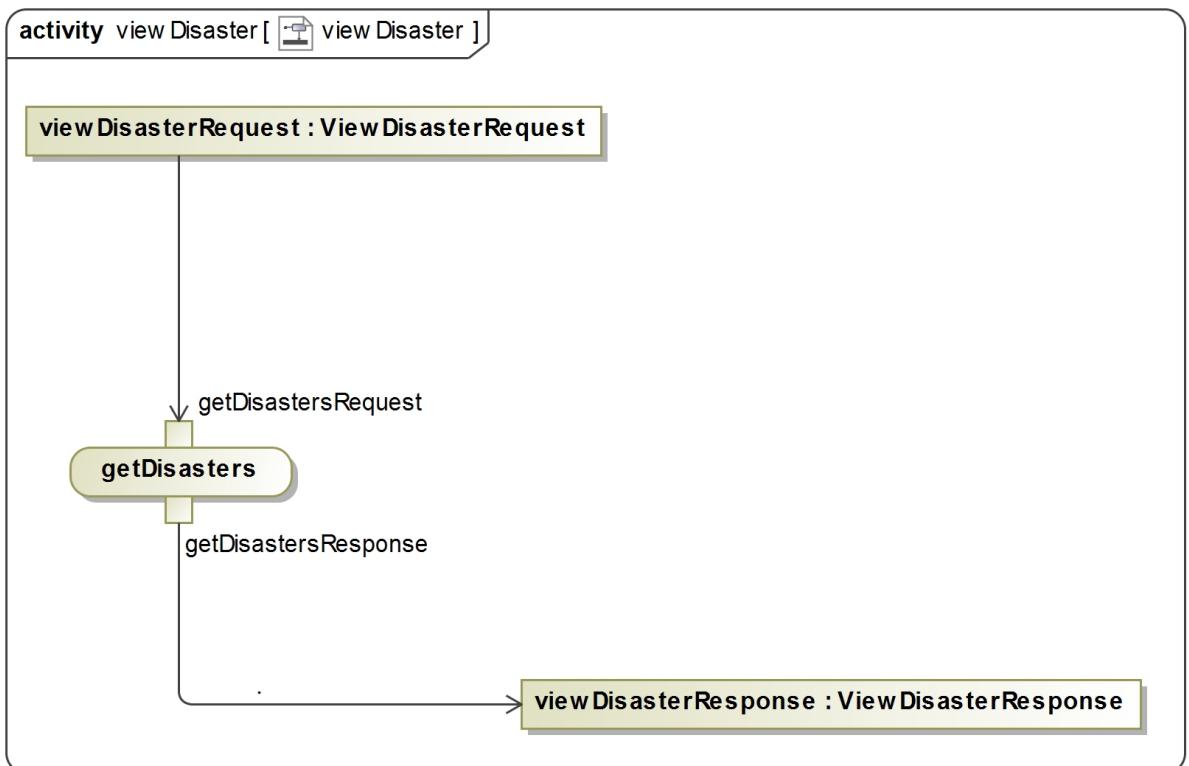


Figure 5: The activity diagram for viewDisaster

### 5.1.3 queryDisaster

The user will be able to query the system for specific disasters, by entering search criteria, such as a start date and/or an end date, so that only disasters matching the specified criteria are returned to the user. Below are the service contract, activity diagram and functional requirements diagram for queryDisaster.

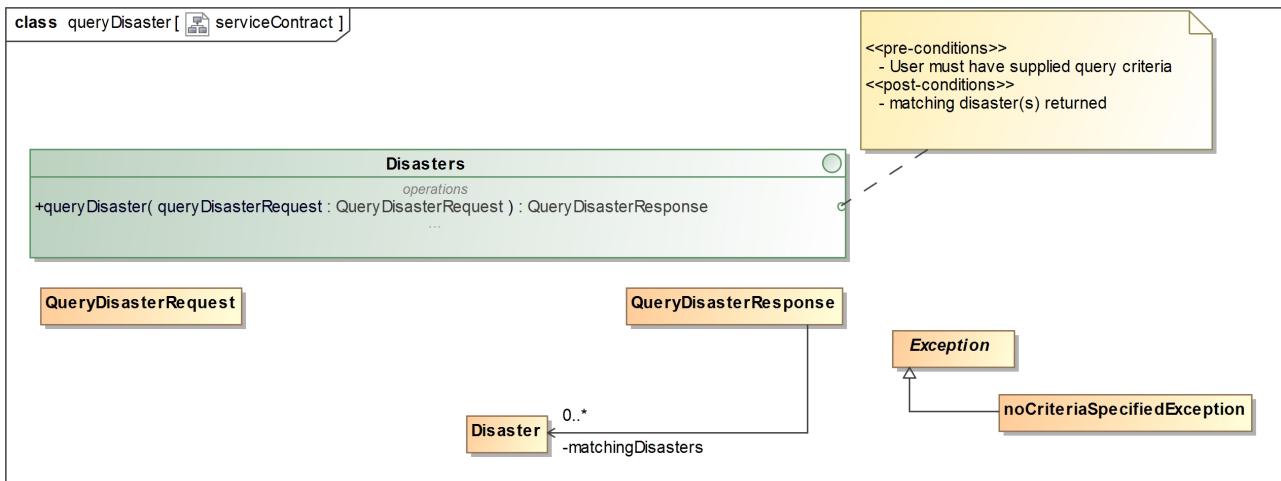


Figure 6: The service contract for queryDisaster

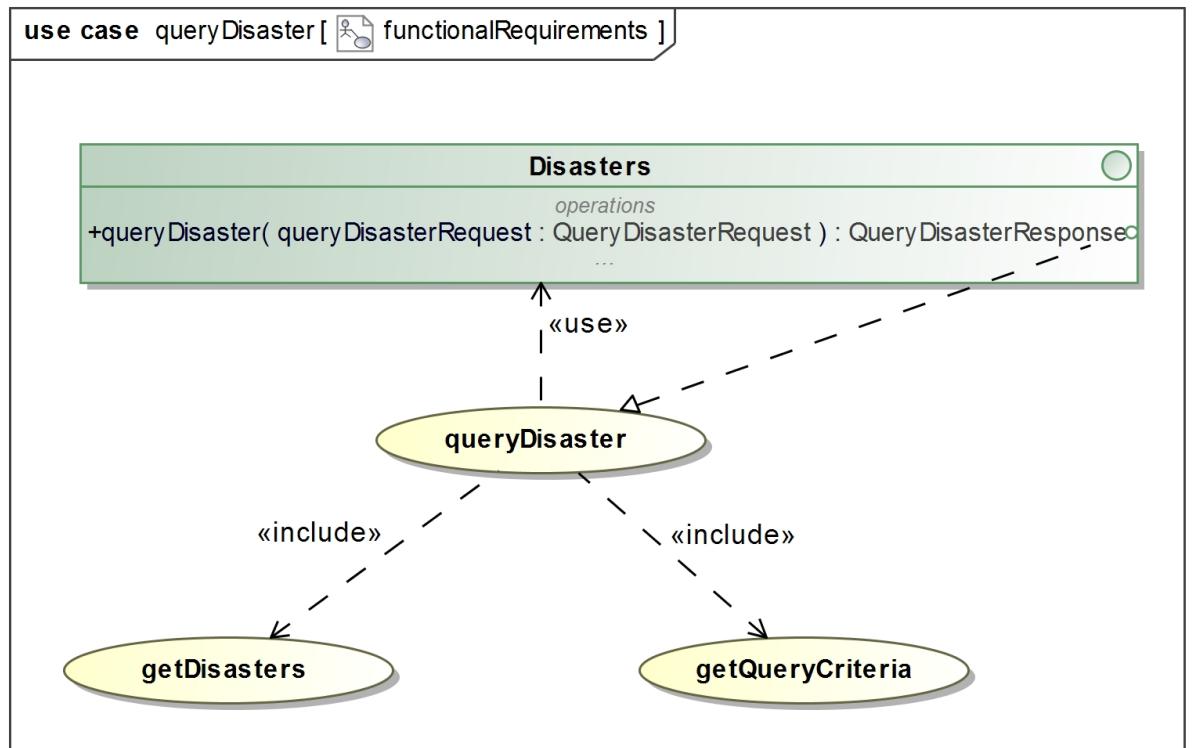


Figure 7: The functional requirements diagram for queryDisaster

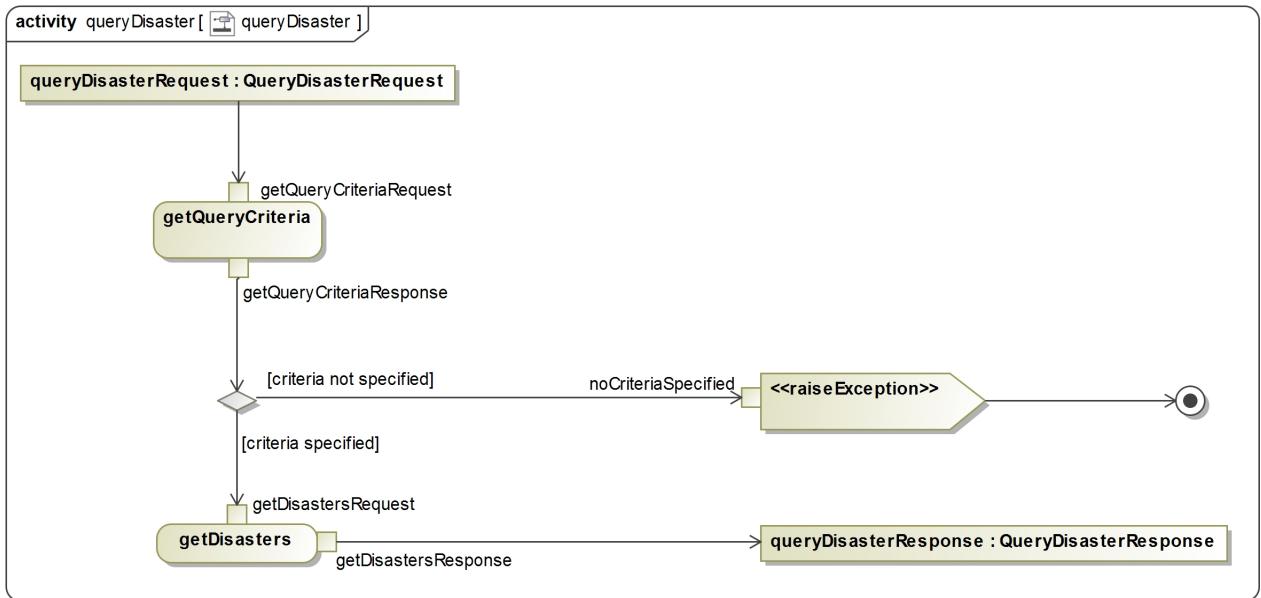


Figure 8: The activity diagram for `queryDisaster`

#### 5.1.4 notifyNewDisaster

A user will be able to be notified of any new disasters that are occurring as of the time they are using the system as long as that disaster is not visualized on the system yet. Below are the service contract, activity diagram and functional requirements diagram for `notifyNewDisaster`.

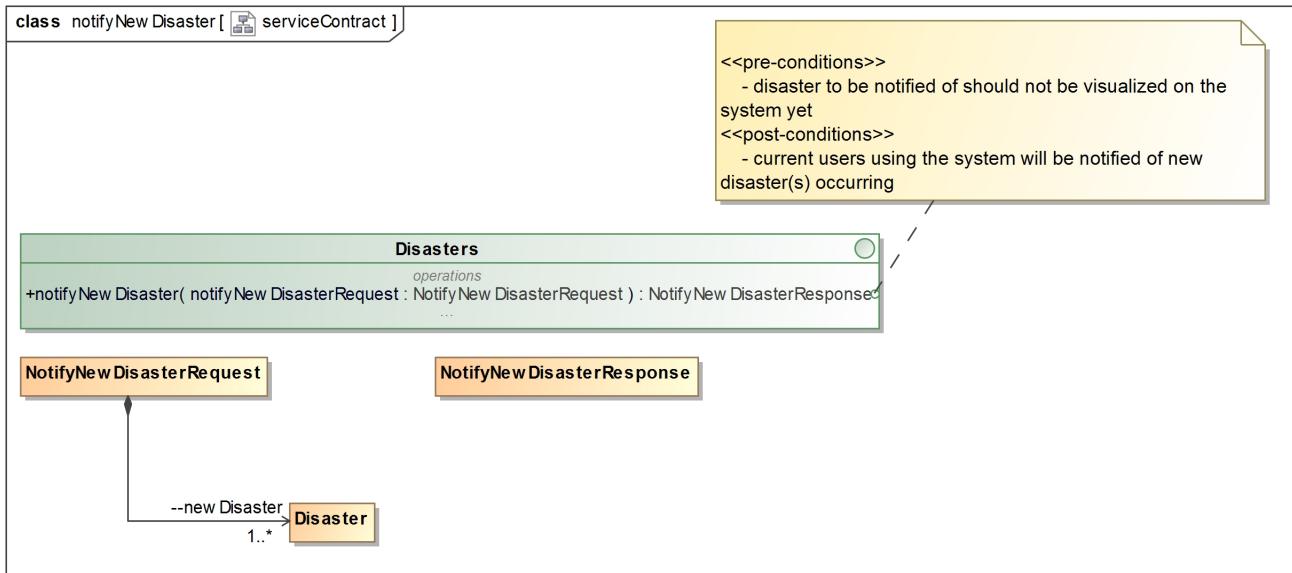


Figure 9: The service contract for notifyNewDisaster

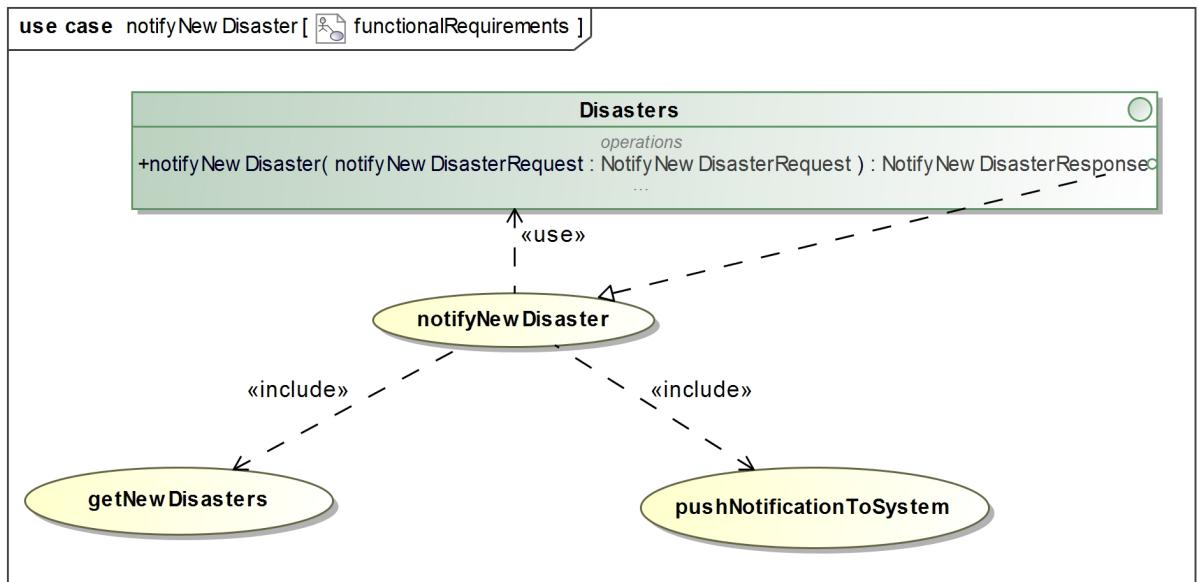


Figure 10: The functional requirements diagram for notifyNewDisaster

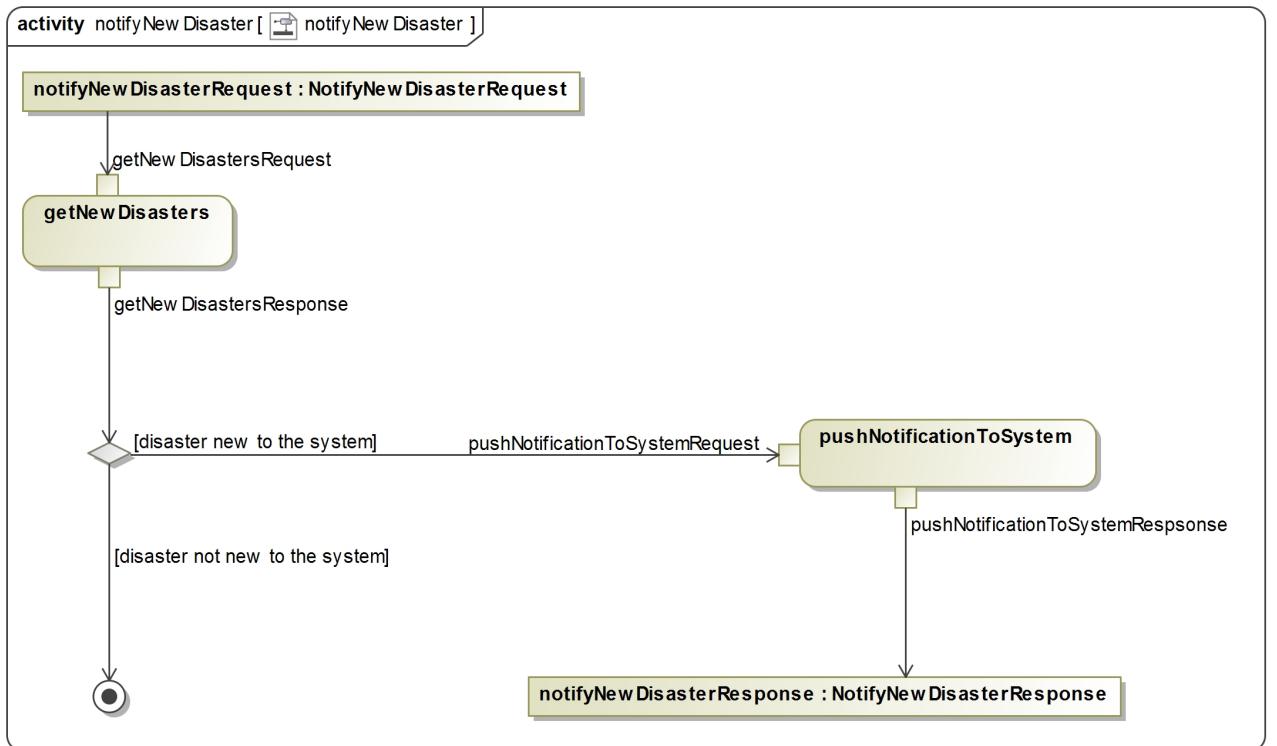


Figure 11: The activity diagram for `notifyNewDisaster`

### 5.1.5 trackDisaster

A user is able to track a disaster as long as the disaster is still active and data for the disaster is updated regularly. Below are the service contract, activity diagram and functional requirements diagram for `trackDisaster`.

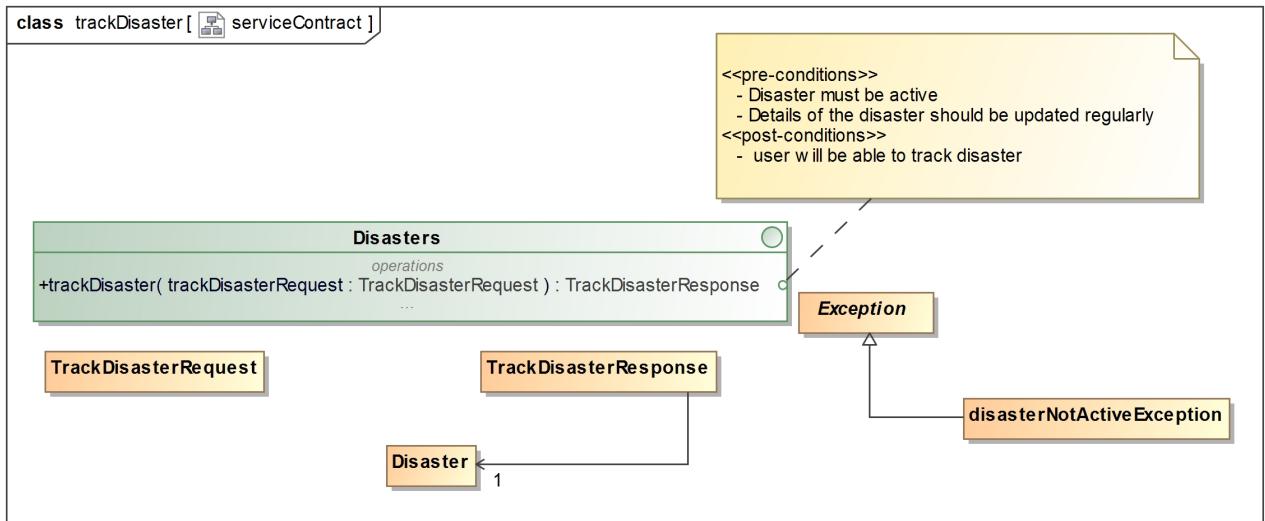


Figure 12: The service contract for trackDisaster

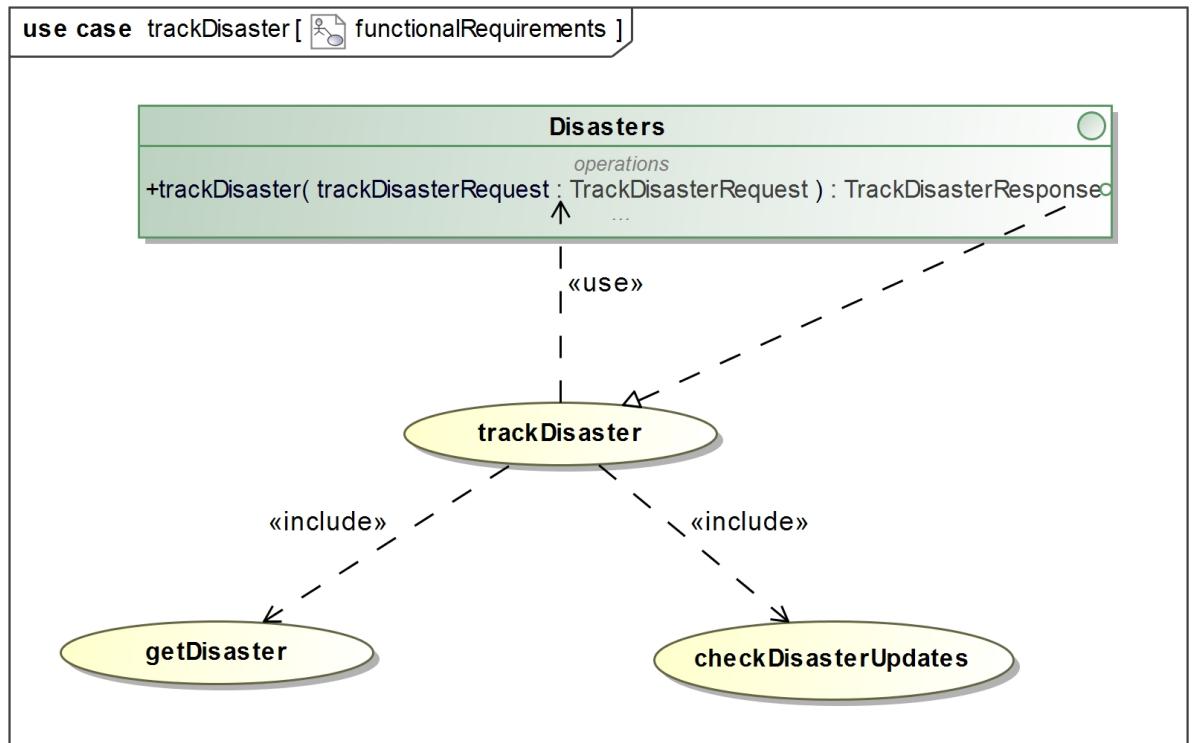


Figure 13: The functional requirements diagram for trackDisaster

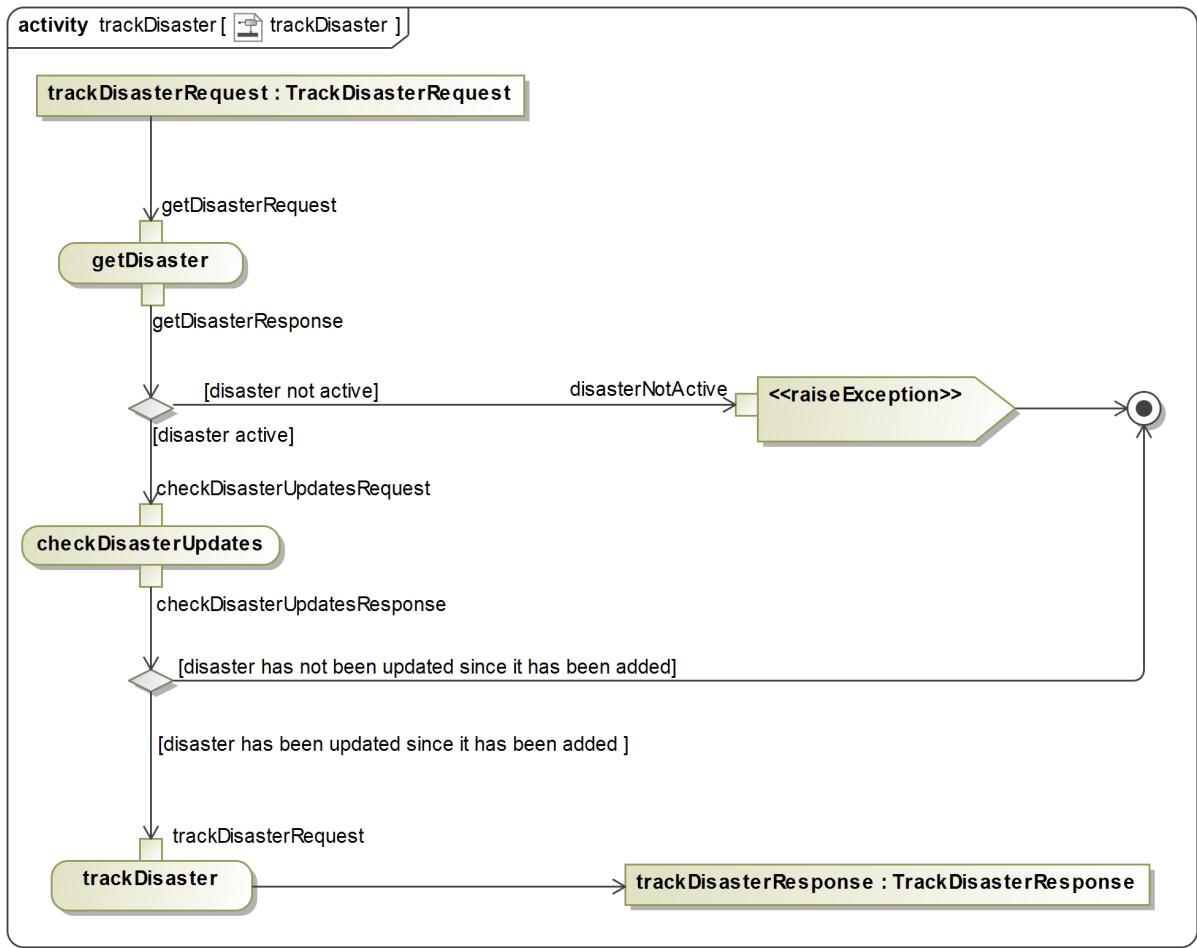


Figure 14: The activity diagram for `trackDisaster`

### 5.1.6 addDisasterType

Adding a disaster type requires that the user has admin rights and that there should not be a disaster type of the type being added that exists. If the user is admin and the disaster type being added is unique then a new disaster type will be added to the system. Below are the service contract, activity diagram and functional requirements diagram for `addDisasterType`.

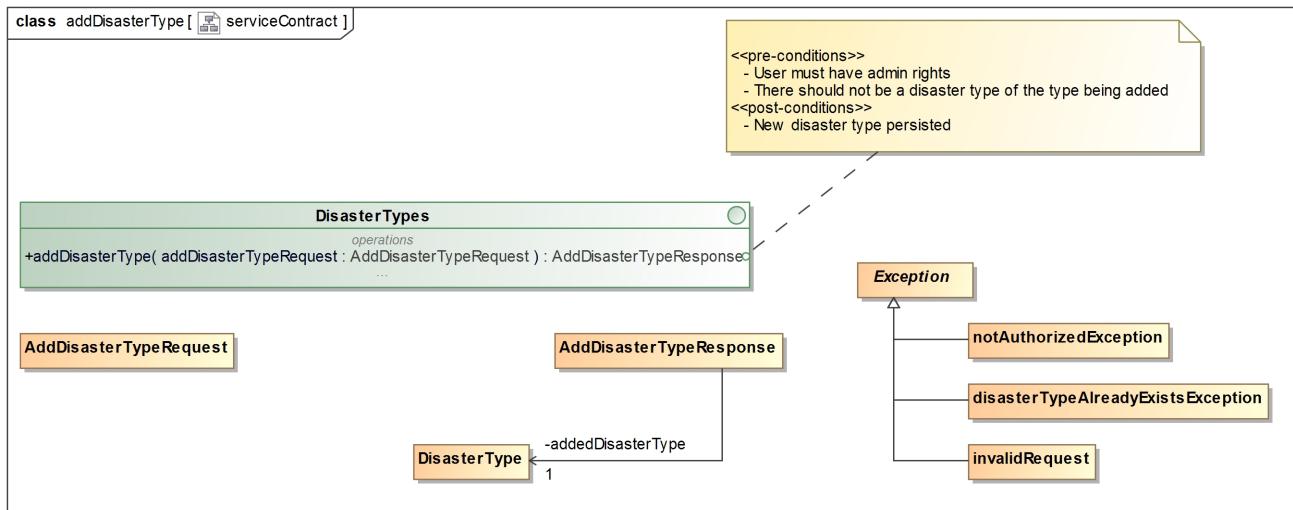


Figure 15: The service contract for addDisasterType

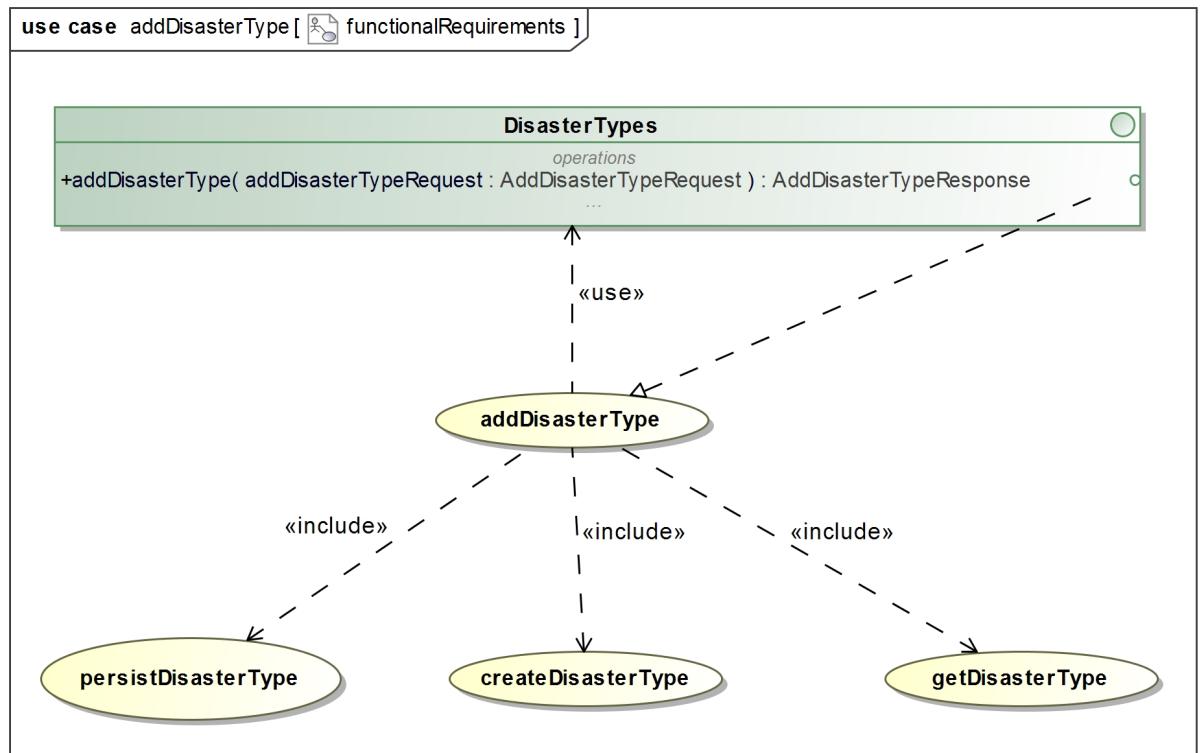


Figure 16: The functional requirements diagram for addDisasterType

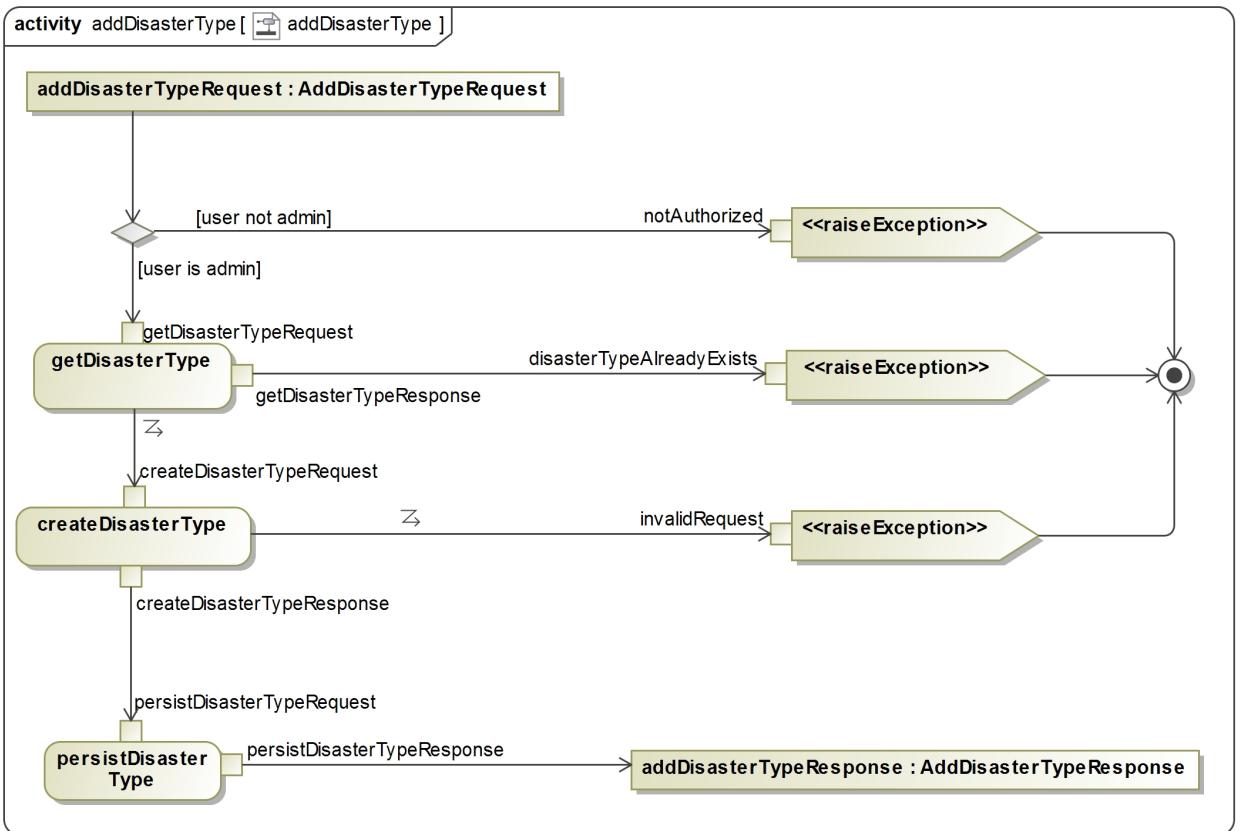


Figure 17: The activity diagram for `addDisasterType`

### 5.1.7 modifyDisasterType

Modifying a disaster type requires that the user has admin rights and that there should not be a disaster type of the same type as the modified one. If the user is admin and the modified disaster type is unique then the modified disaster type will be persisted to the database. Below are the service contract, activity diagram and functional requirements diagram for `modifyDisasterType`.

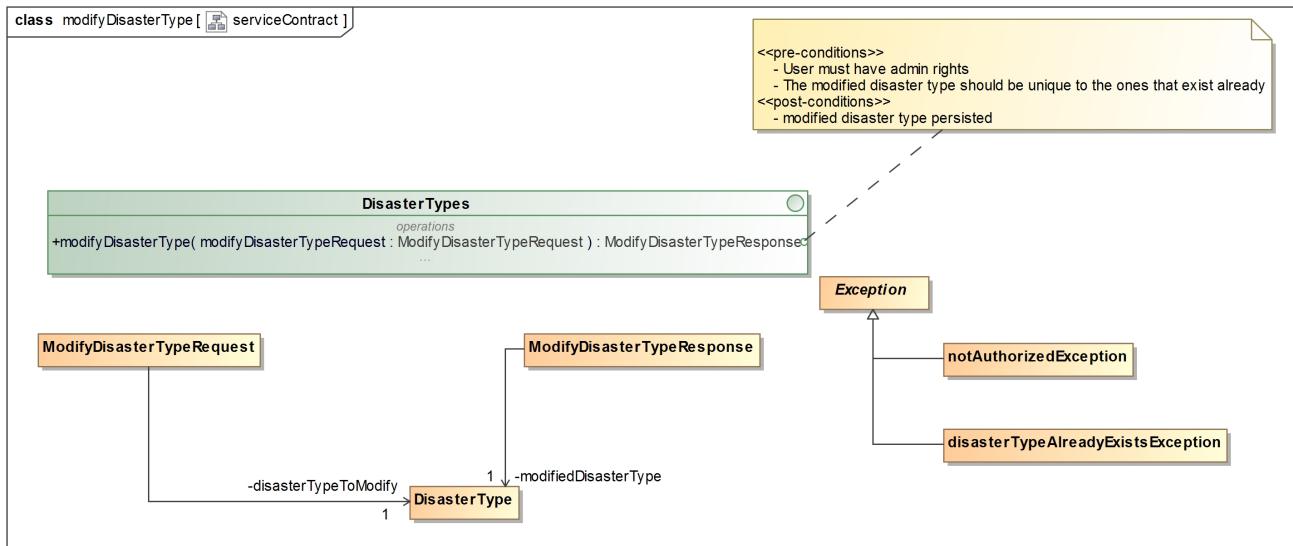


Figure 18: The service contract for modifyDisasterType

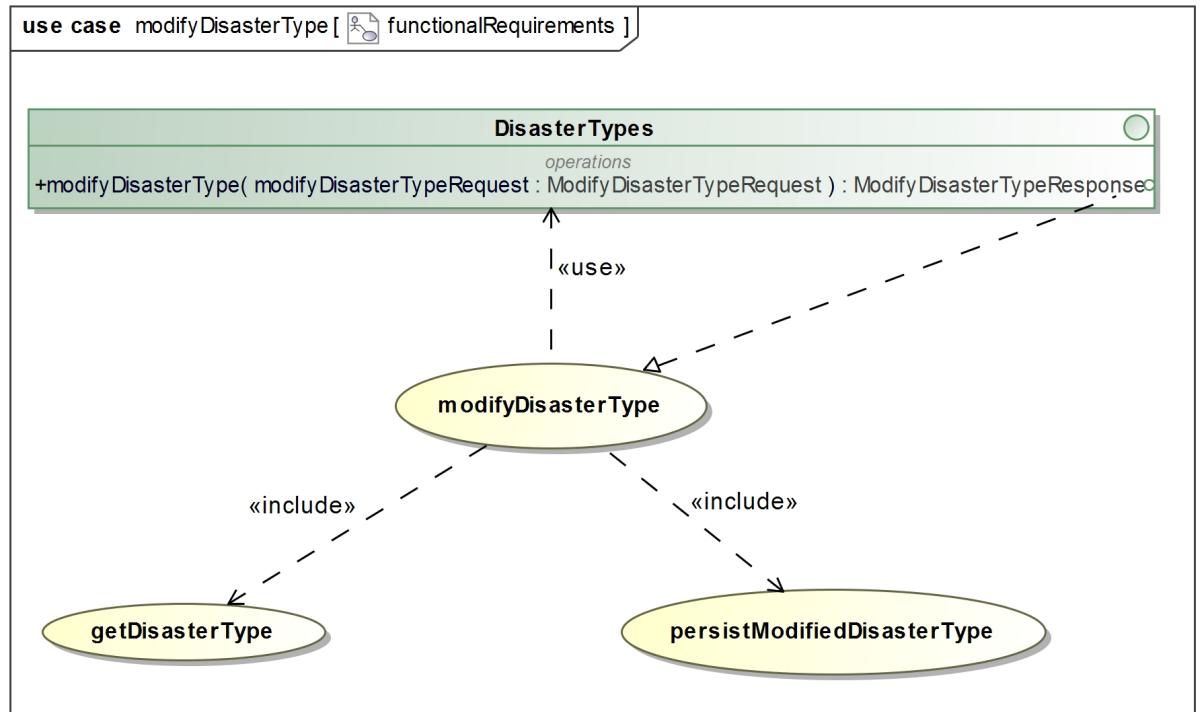


Figure 19: The functional requirements diagram for modifyDisasterType

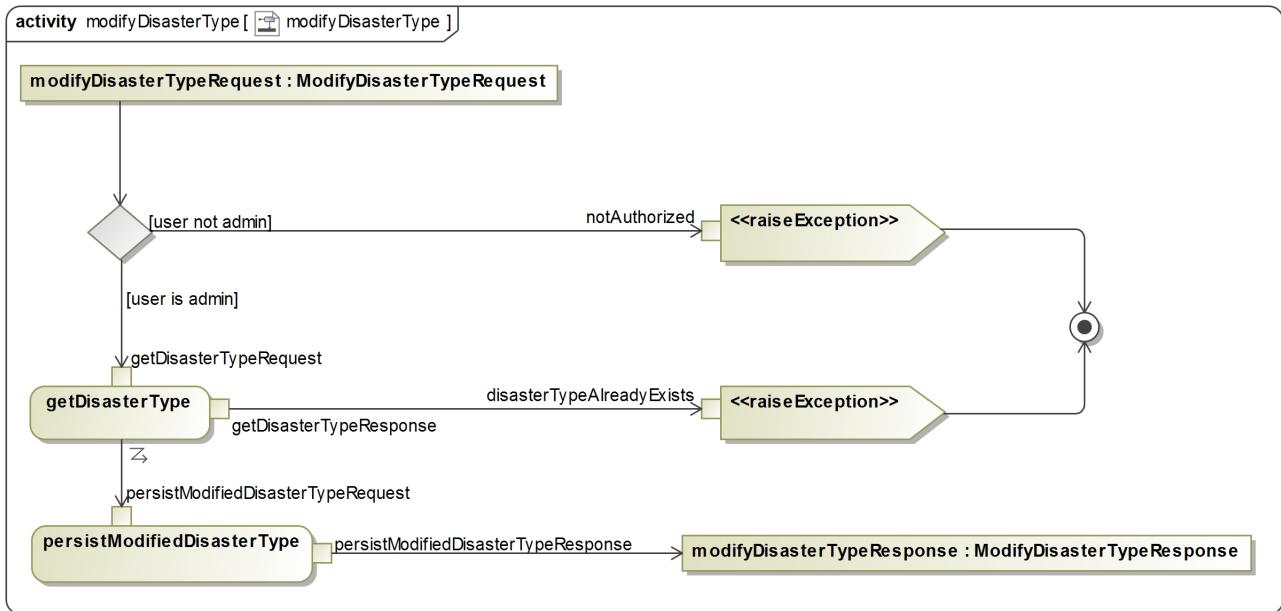


Figure 20: The activity diagram for `modifyDisasterType`

## 5.2 Climate Management Module

### 5.2.1 Scope

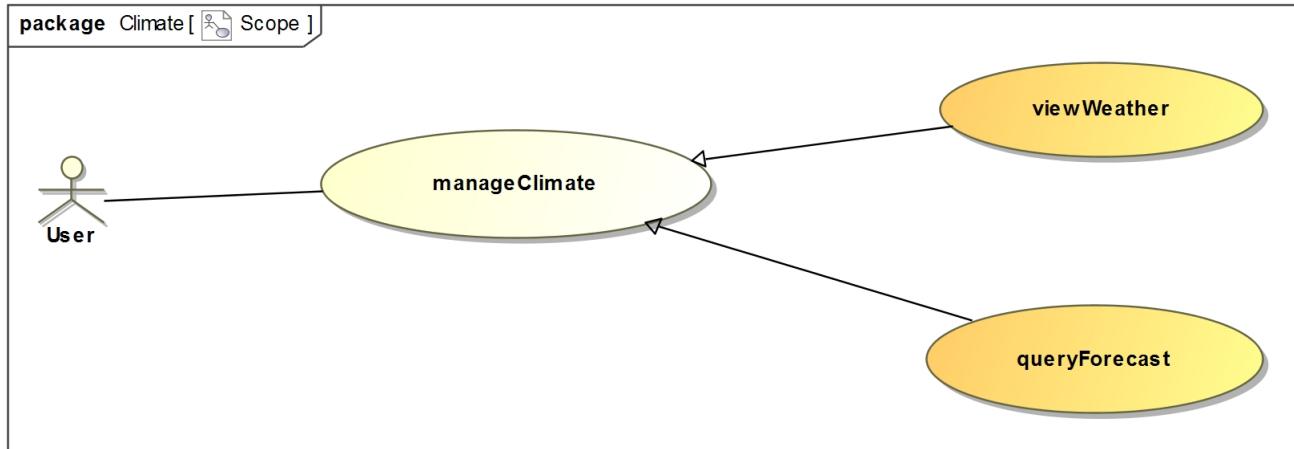


Figure 21: The scope of functionality required from the climate module

Users are able to view weather and query forecasts.

### 5.2.2 viewWeather

A user can view weather immediately as they go into the system or by going on to the weather tab on the system. The user is able to see weather for the entire world or for a specific place by specifying the name of that place in a search bar. Below are the service contract, activity diagram and functional requirements diagram for viewWeather.

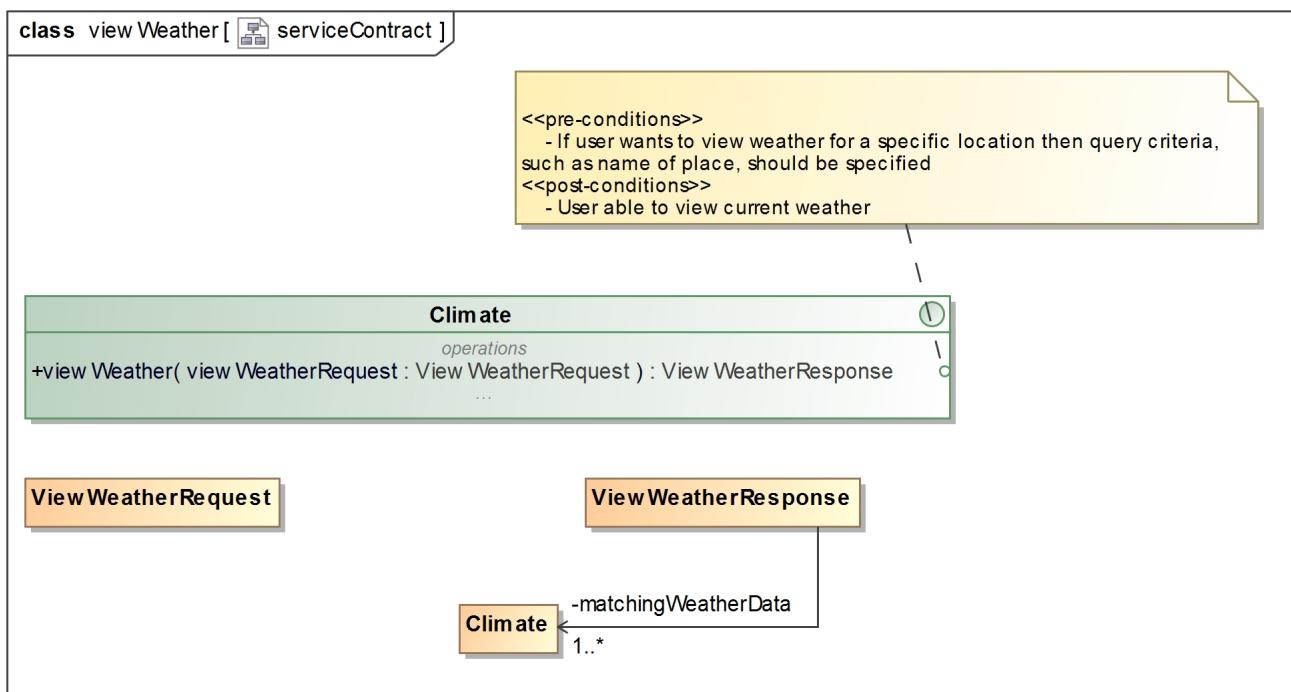


Figure 22: The service contract for viewWeather

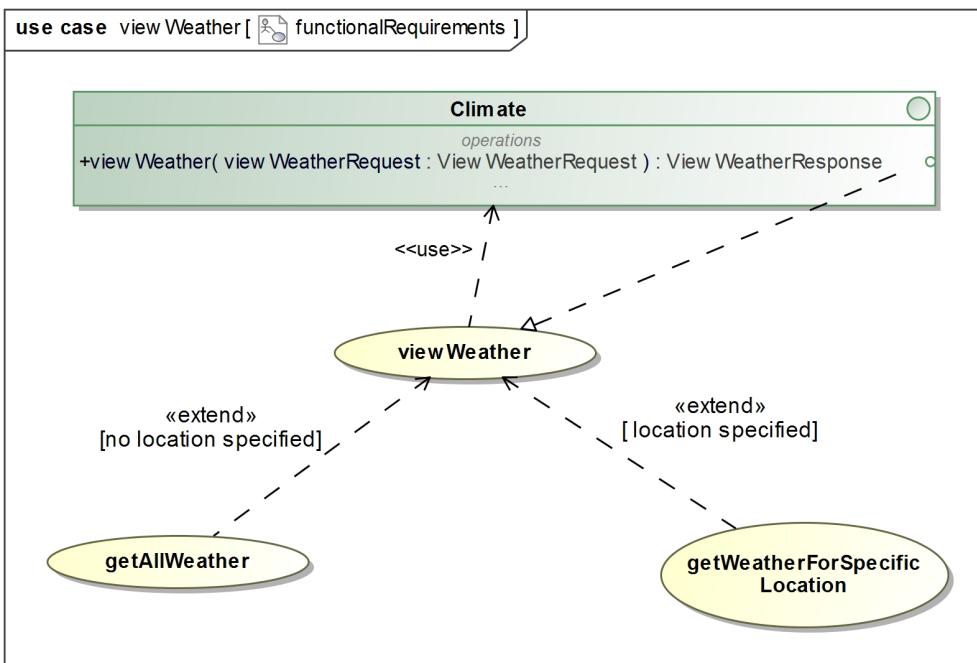


Figure 23: The functional requirements diagram for viewWeather

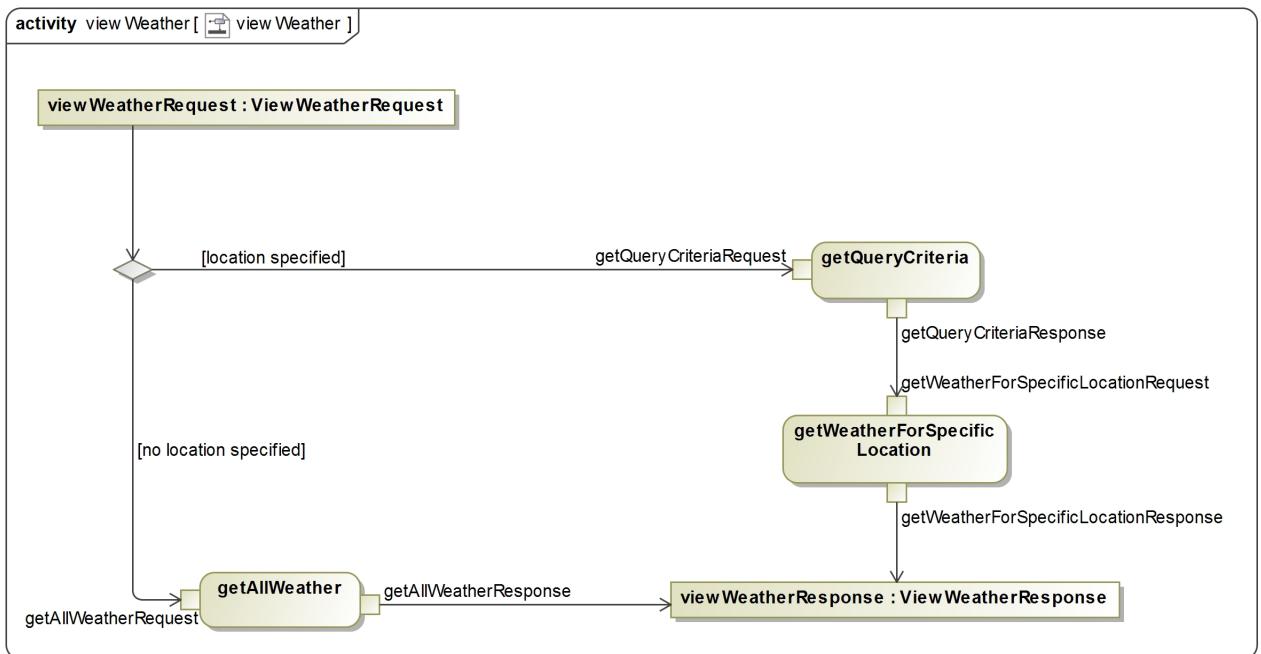


Figure 24: The activity diagram for `viewWeather`

### 5.2.3 queryForecast

A user is able to view the weather of a specific location for the next 5 days, given that they provide the latitude and longitude of that location. Below are the service contract, activity diagram and functional requirements diagram for `queryForecast`.

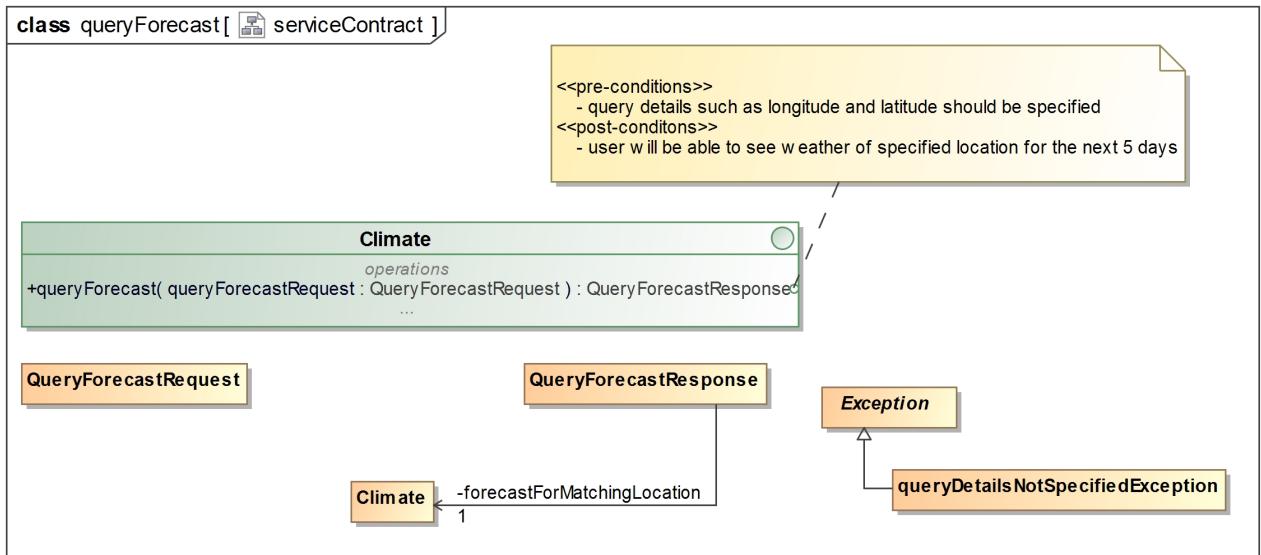


Figure 25: The service contract for queryForecast

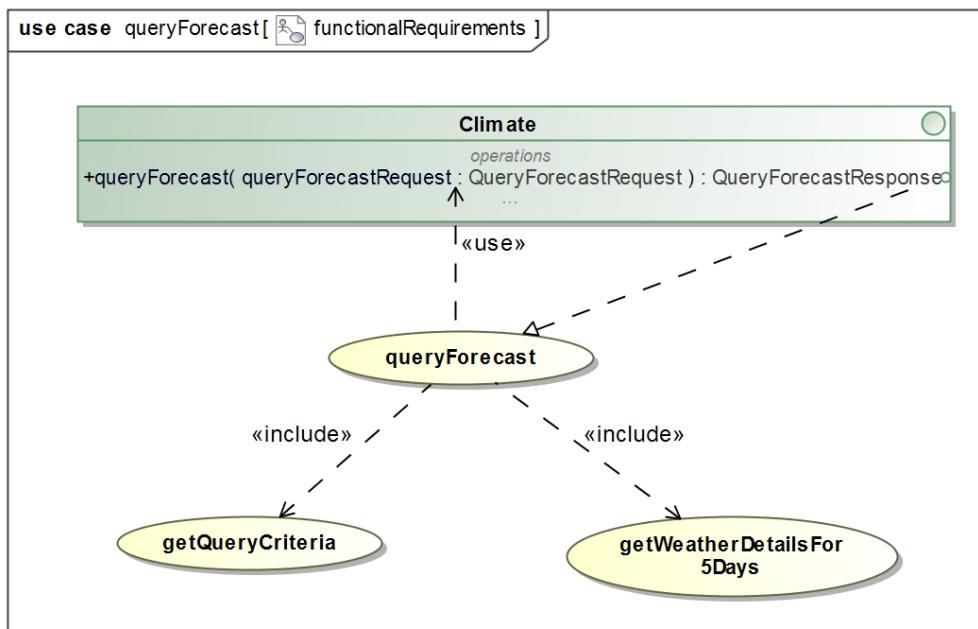


Figure 26: The functional requirements diagram for queryForecast

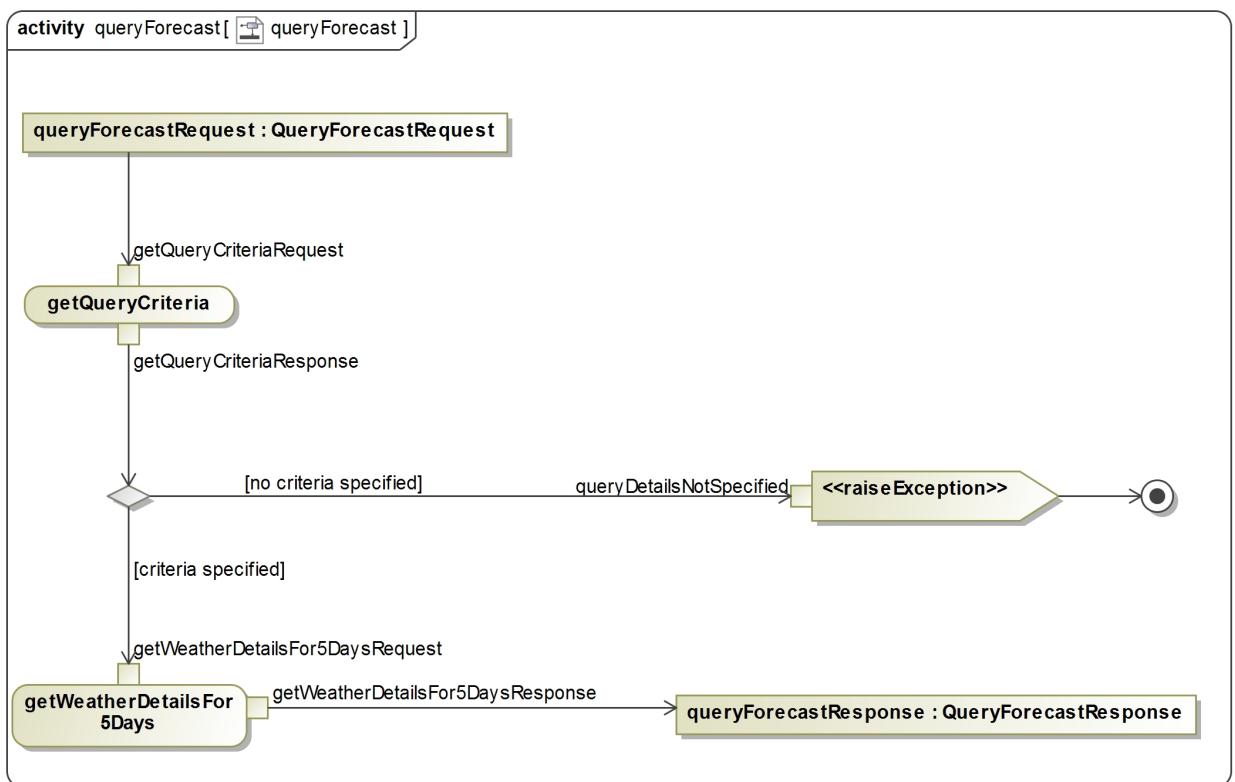


Figure 27: The activity diagram for `queryForecast`

## 5.3 User Module

### 5.3.1 Scope

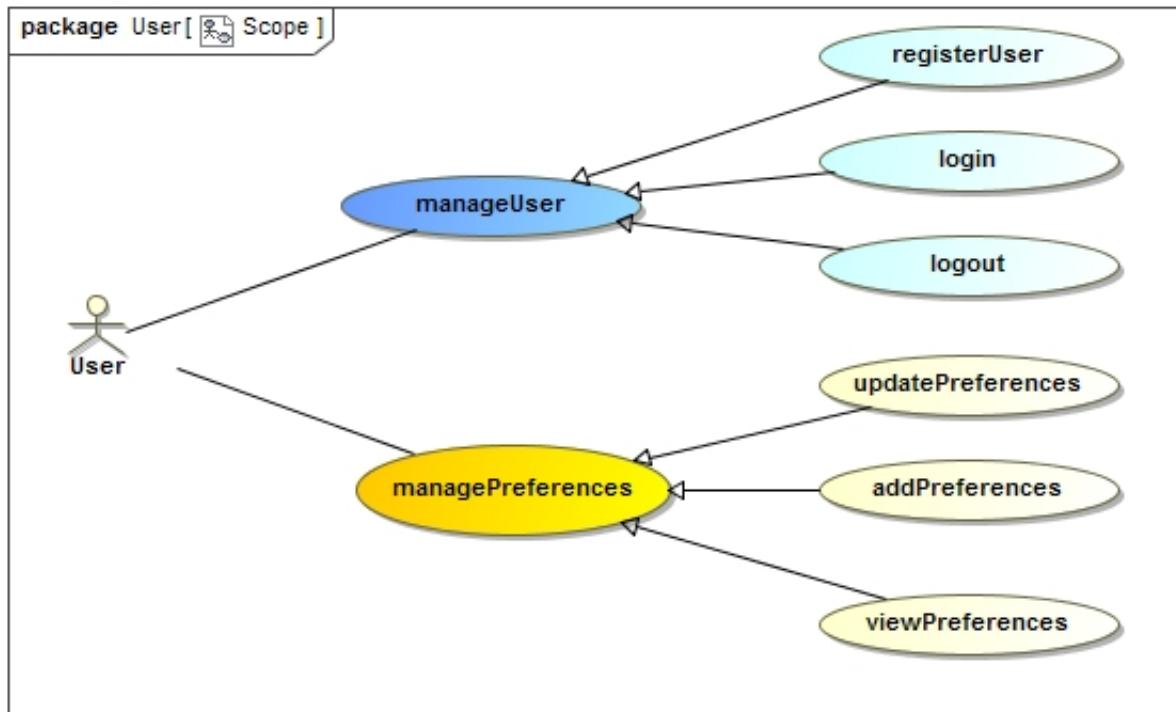


Figure 28: The scope of functionality required from the user module

A user can register and login in to the system to get access to functionality such as adding,viewing and editing preferences of weather or disaster data that they would like to be at their disposal. A user is also able to logout of the system once done.

### 5.3.2 registerUser

A user can register with the system as long as they are not already registered. Below are the service contract, activity diagram and functional requirements diagram for registerUser.

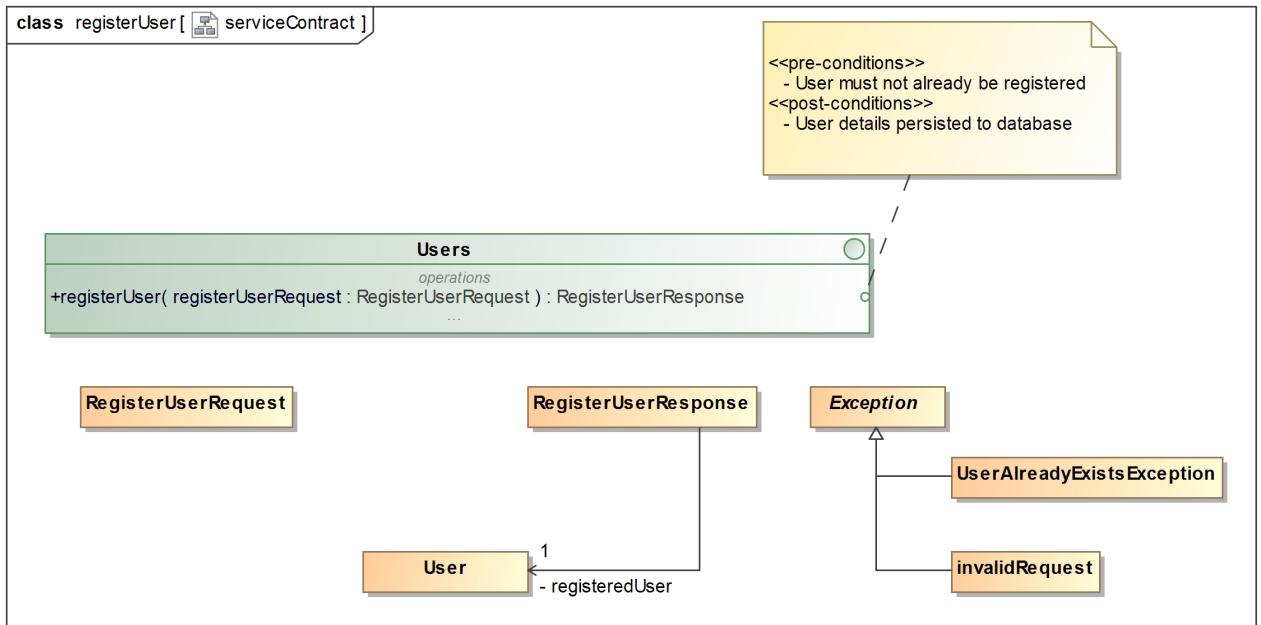


Figure 29: The service contract for registerUser

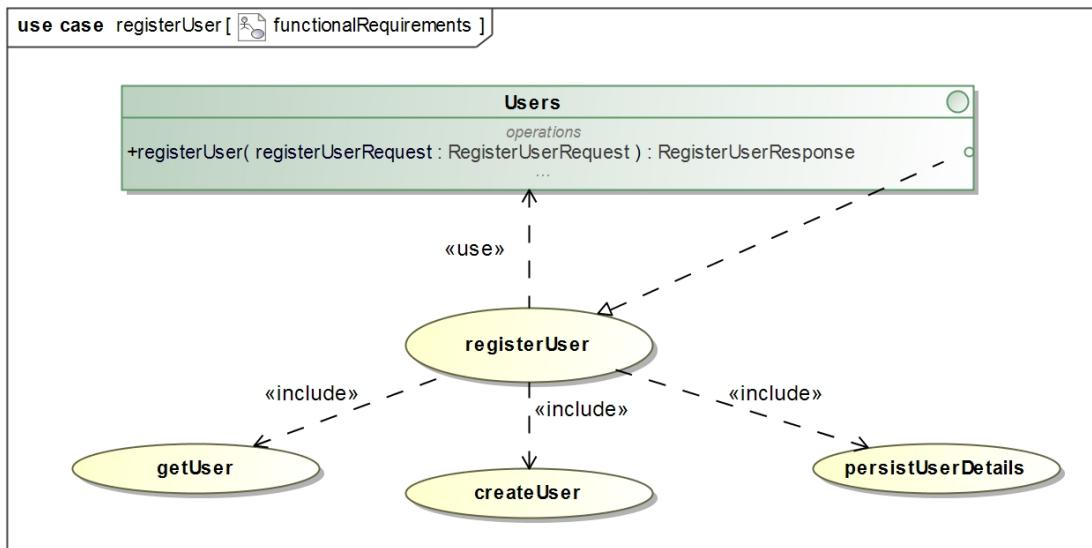


Figure 30: The functional requirements diagram for registerUser

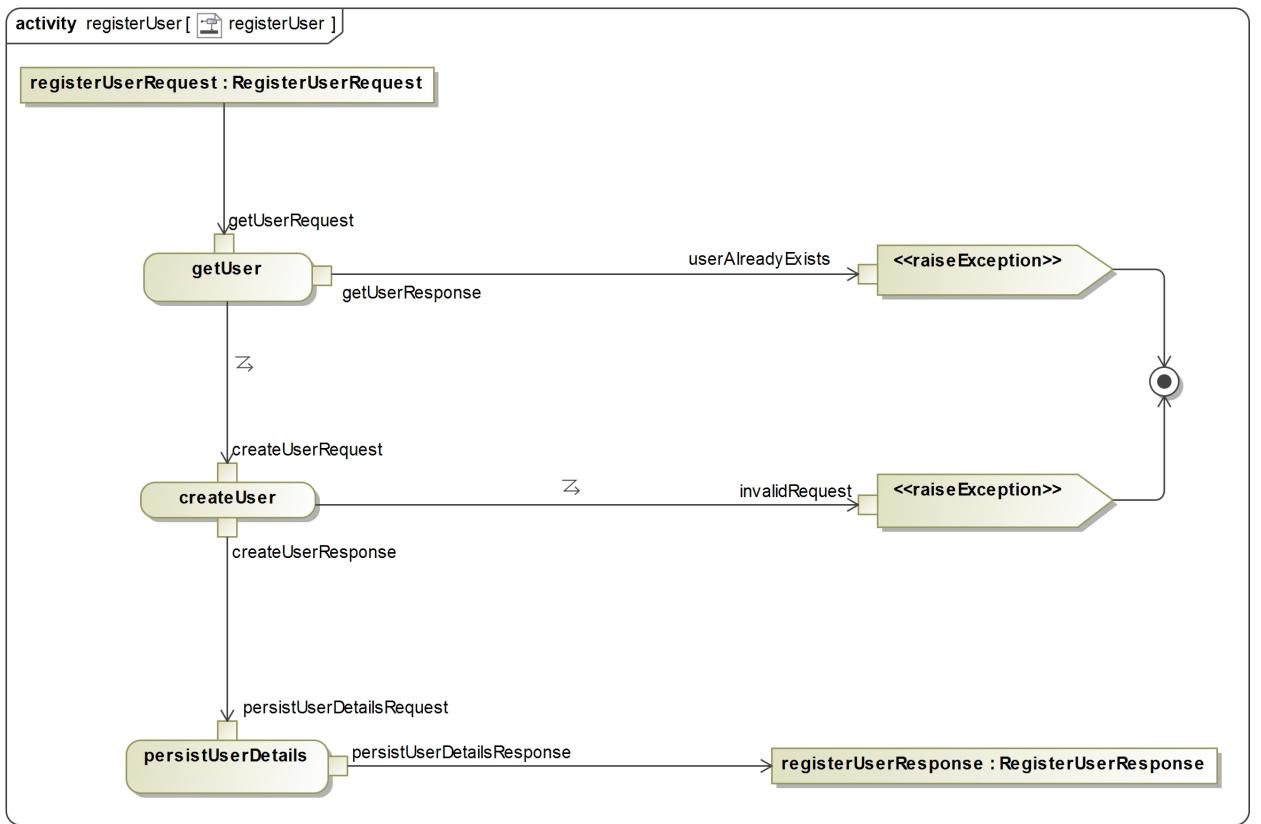


Figure 31: The activity diagram for registerUser

### 5.3.3 login

A registered user, given that they provide correct details to authenticate them and they are not already logged in, is able to access functionality such as adding and editing weather and disaster preferences once logged in. Below are the service contract, activity diagram and functional requirements diagram for login.

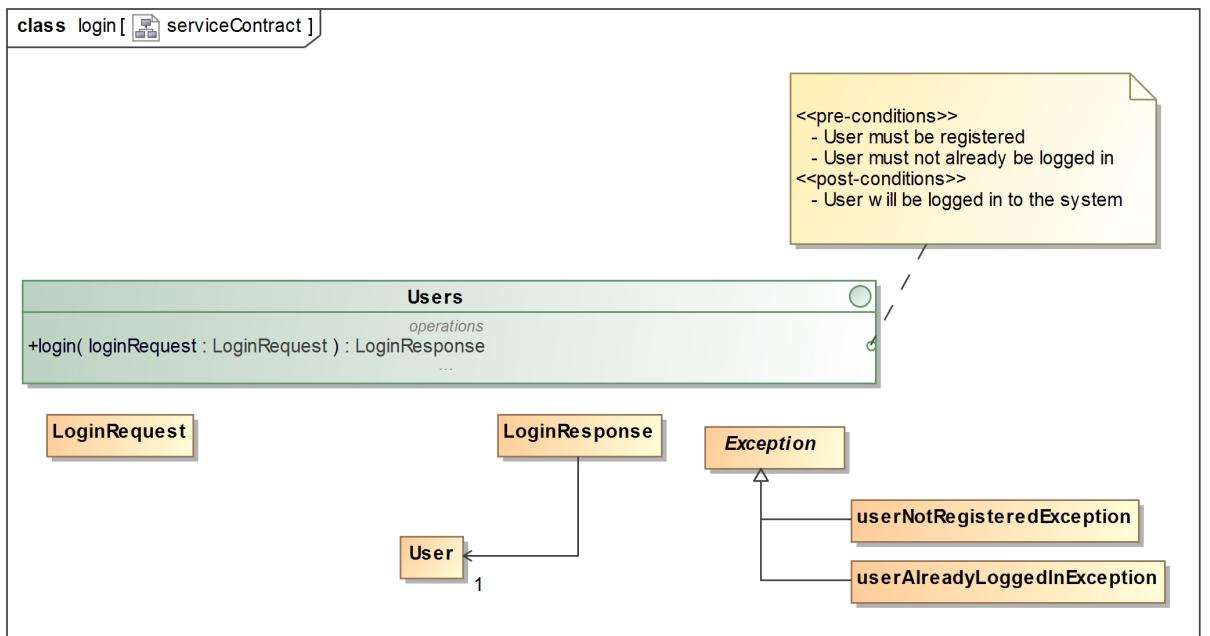


Figure 32: The service contract for login

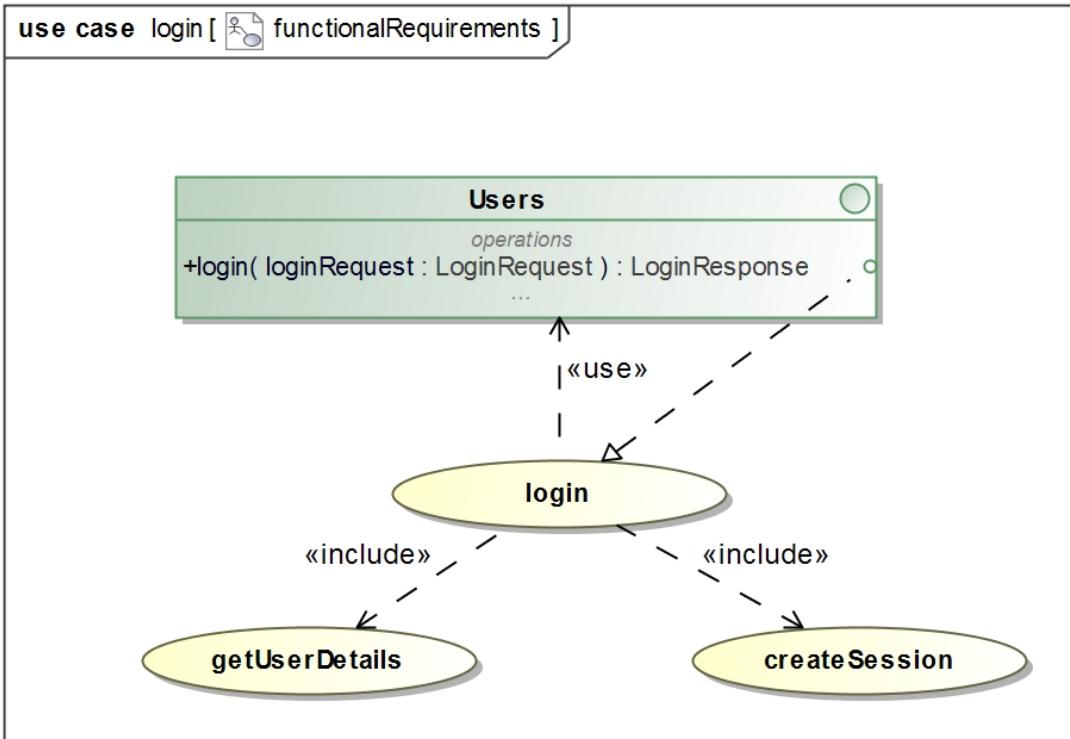


Figure 33: The functional requirements diagram for login

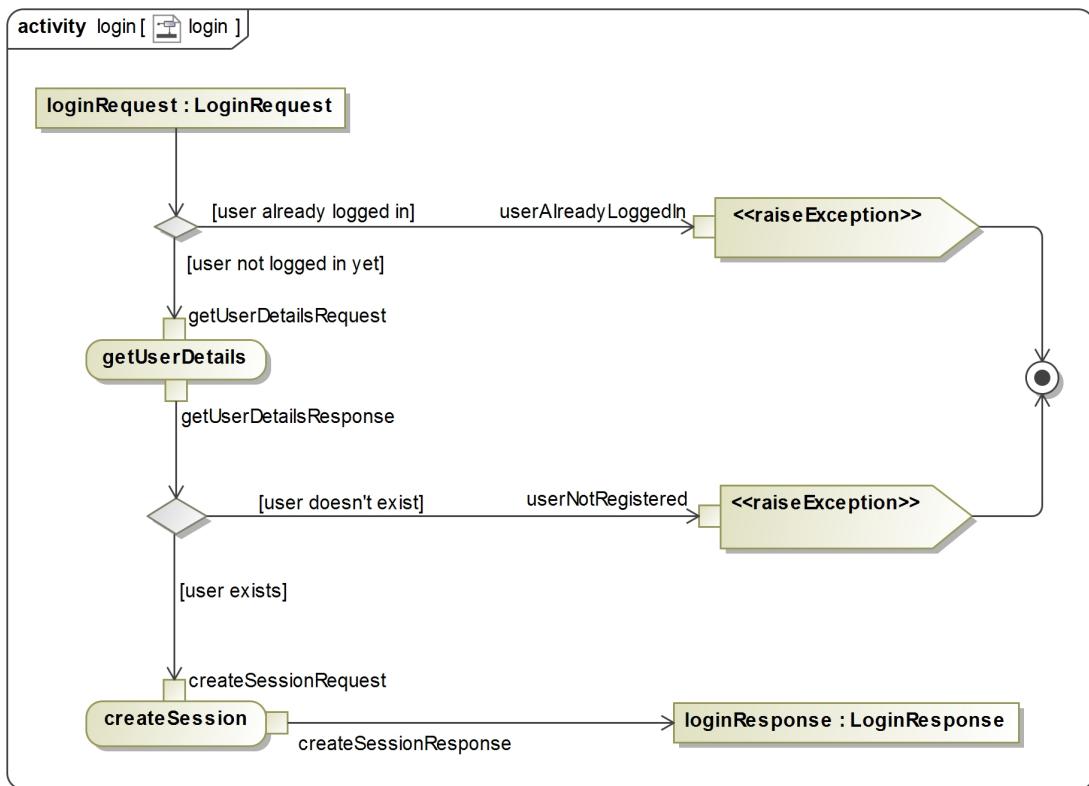


Figure 34: The activity diagram for login

#### 5.3.4 logout

A user, given that they are logged in to the system, is able to logout of the system once done. Below are the service contract, activity diagram and functional requirements diagram for logout.

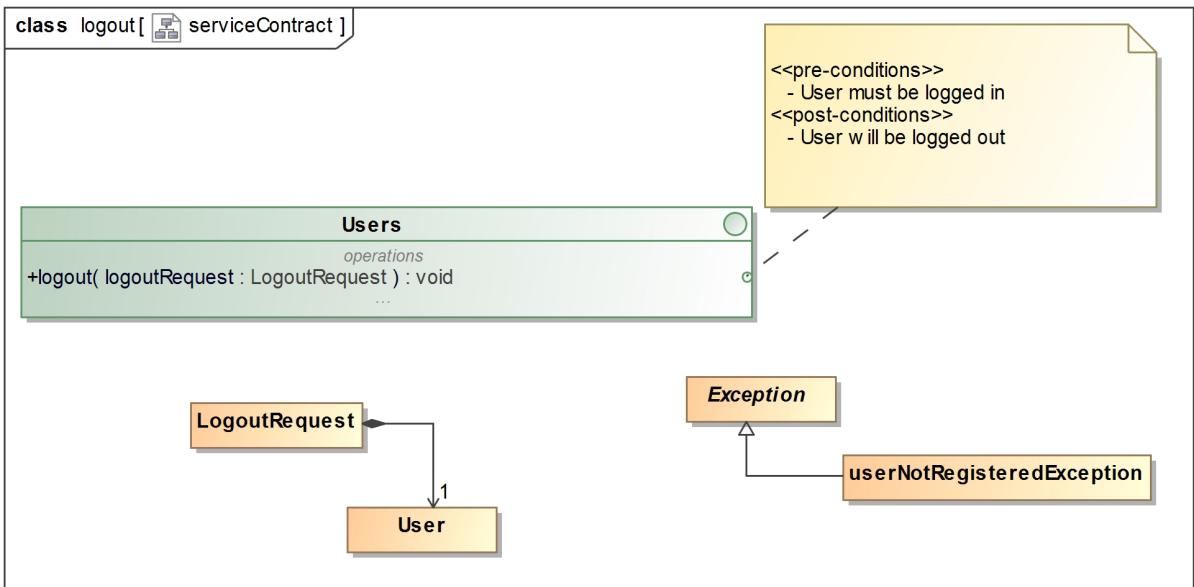


Figure 35: The service contract for logout

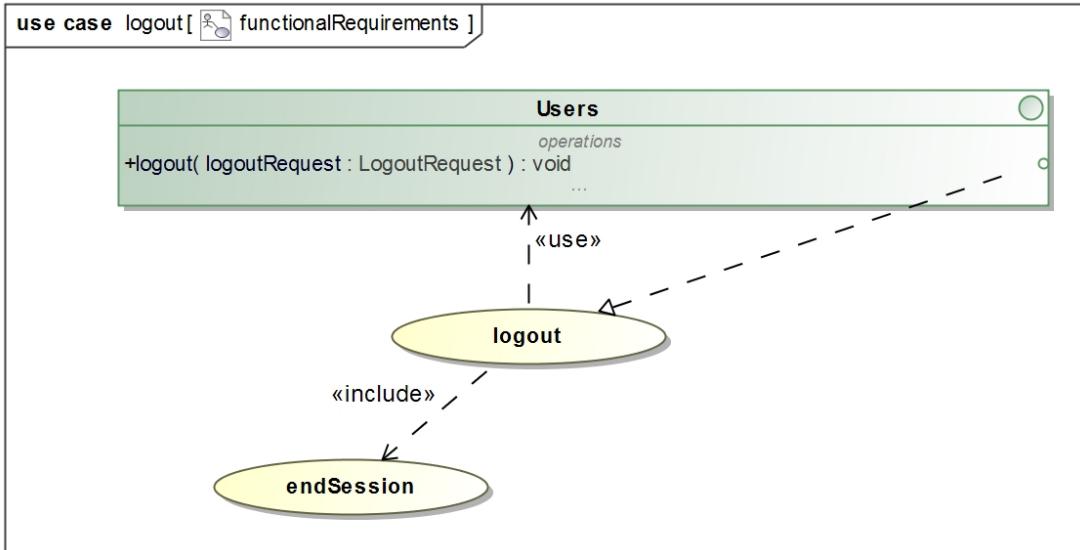


Figure 36: The functional requirements diagram for logout

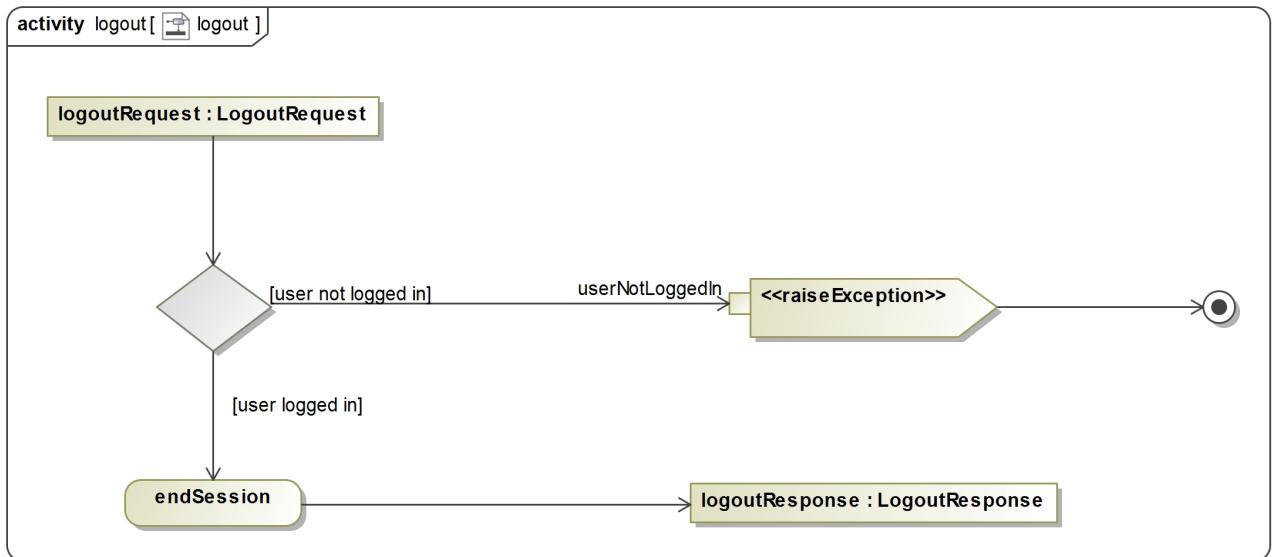


Figure 37: The activity diagram for logout

### 5.3.5 addPreferences

A user, given that they are logged into the system, is able to add their weather and/or disaster data preferences as long as the user has specified the preference details correctly and those preference details do not already exist for the user. Below are the service contract, activity diagram and functional requirements diagram for addPreferences.

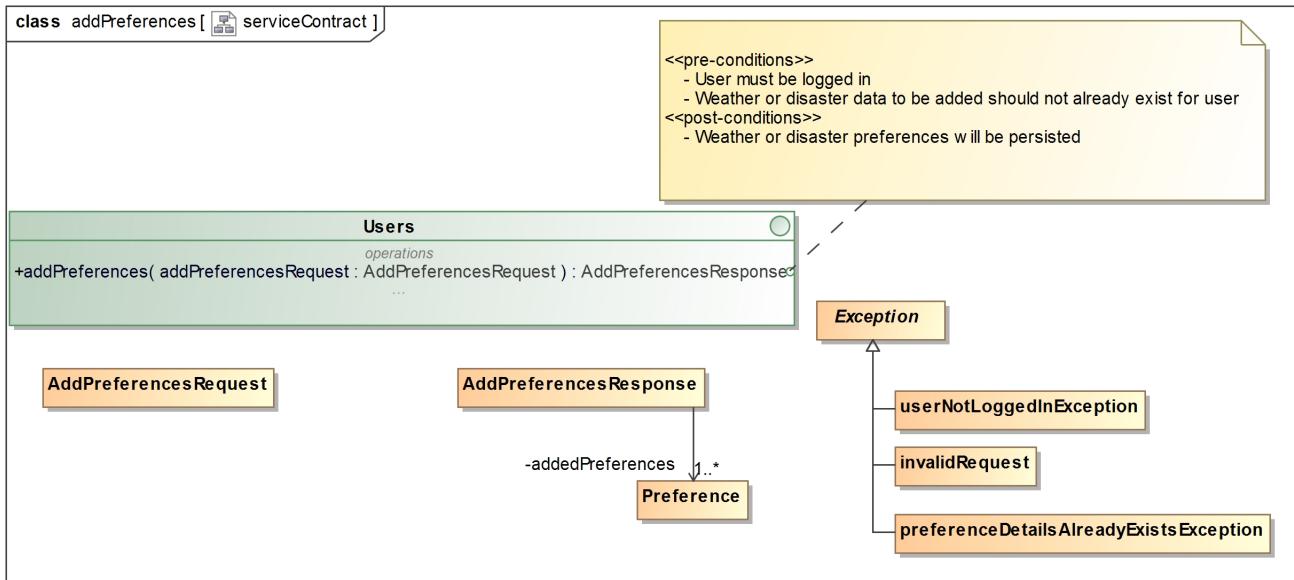


Figure 38: The service contract for addPreferences

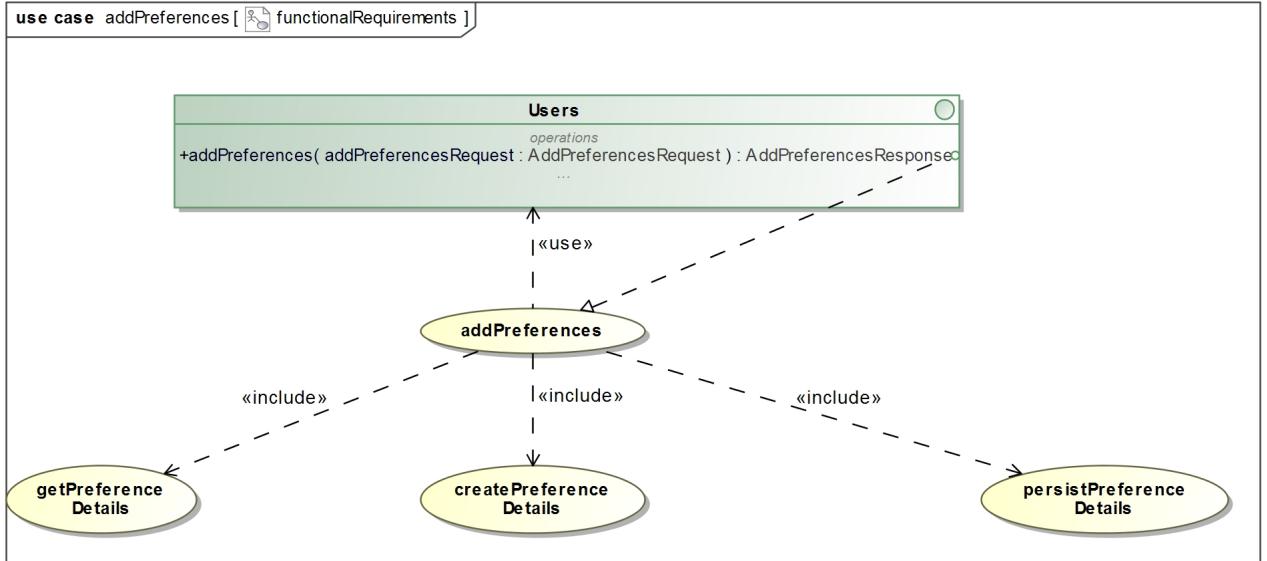


Figure 39: The functional requirements diagram for addPreferences

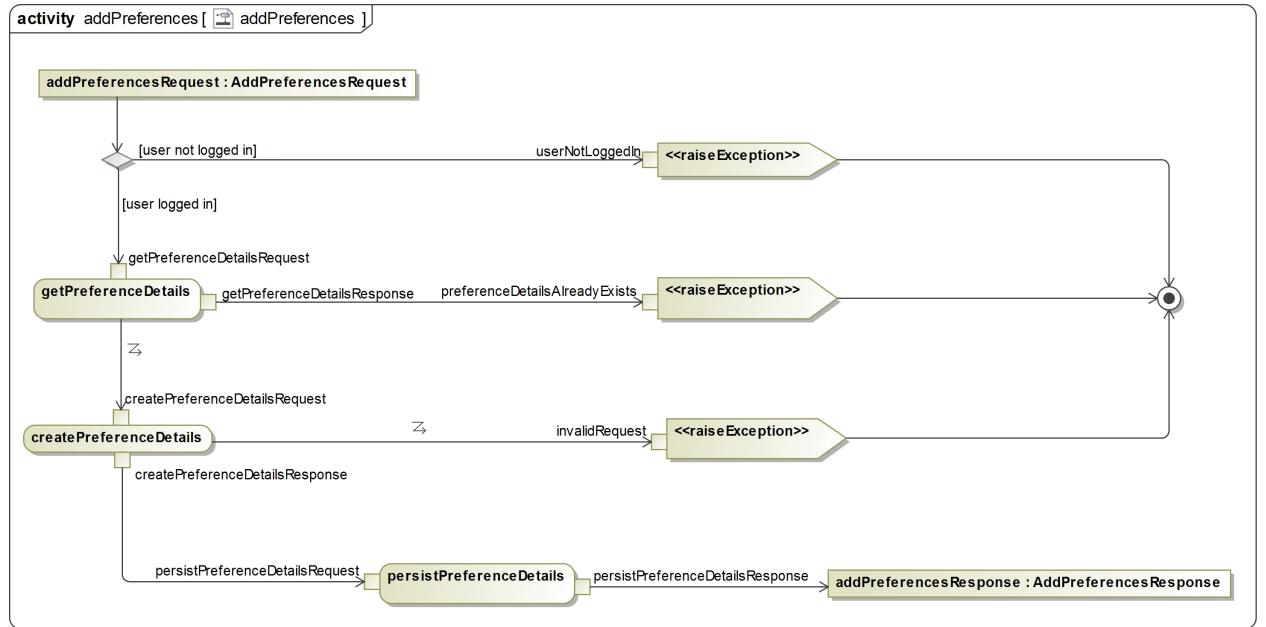


Figure 40: The activity diagram for `addPreferences`

### 5.3.6 editPreferences

A user, given that they are logged into the system, is able to edit any of their weather and/or disaster data preferences as long as the preference details to update exist and that the preference details when updated are not like any of the ones that already exist for the user. Below are the service contract, activity diagram and functional requirements diagram for `editPreferences`.

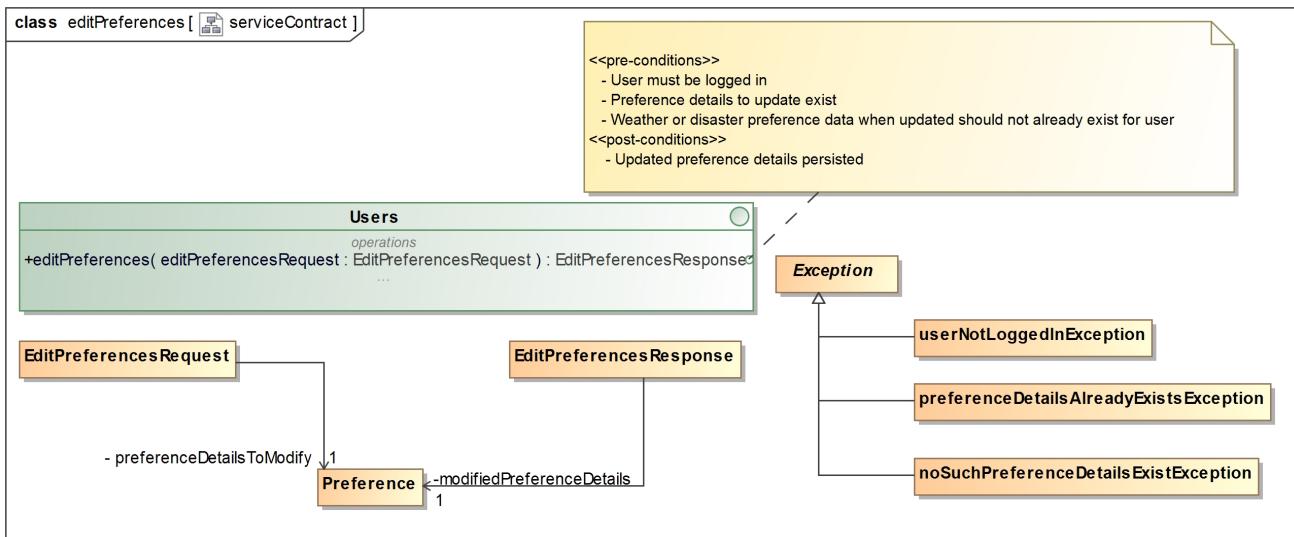


Figure 41: The service contract for editPreferences

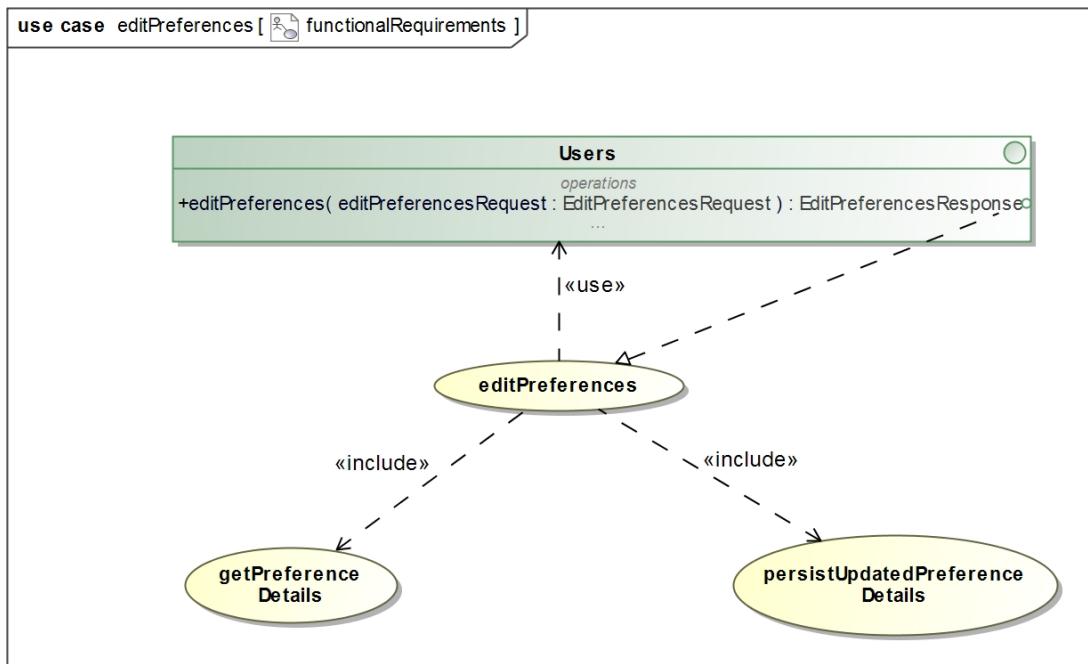


Figure 42: The functional requirements diagram for editPreferences

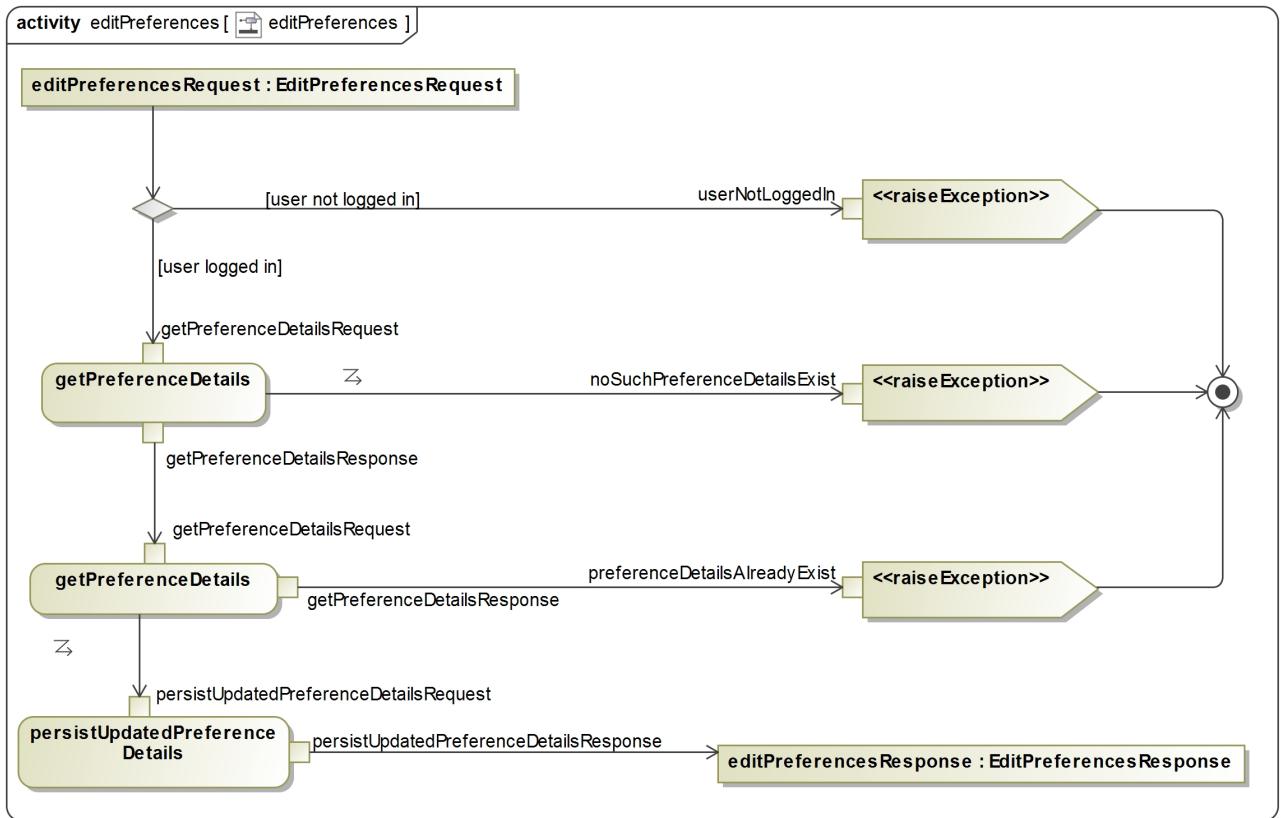


Figure 43: The activity diagram for `editPreferences`

### 5.3.7 viewPreferences

A user, given that they are logged into the system, is able to view any of their weather and/or disaster preferences as long as they have already added those preferences. Below are the service contract, activity diagram and functional requirements diagram for `viewPreferences`.

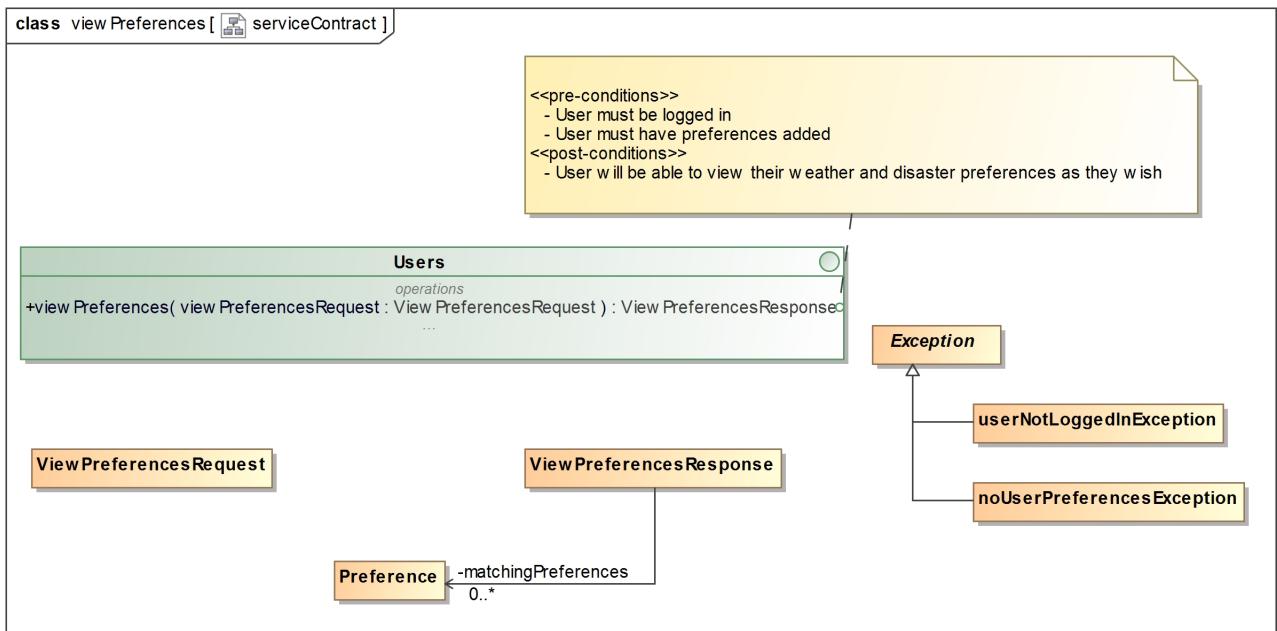


Figure 44: The service contract for viewPreferences

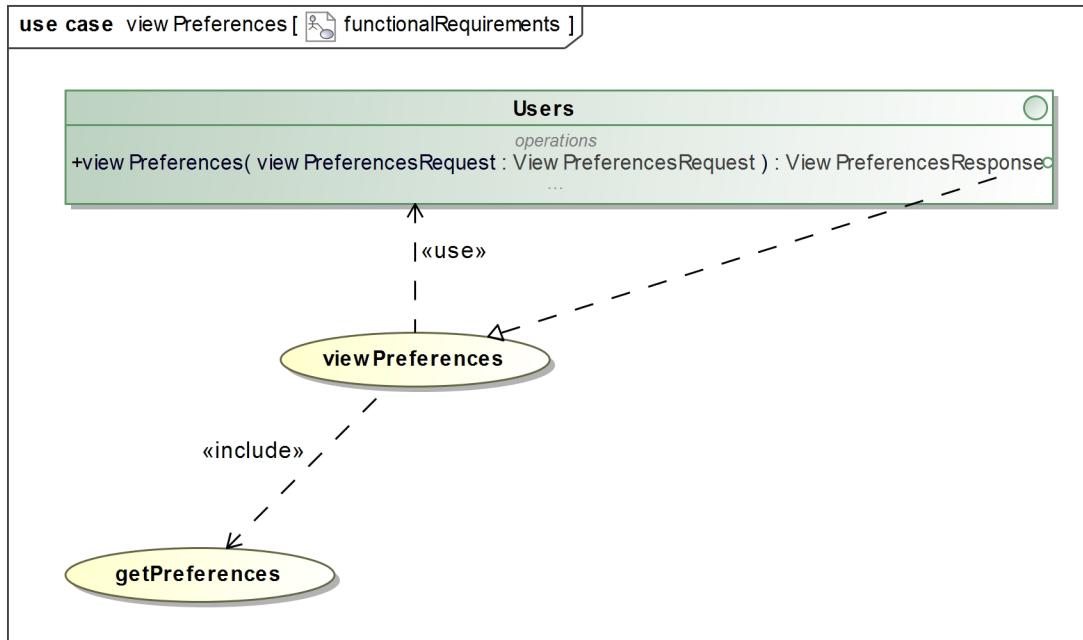


Figure 45: The functional requirements diagram for viewPreferences

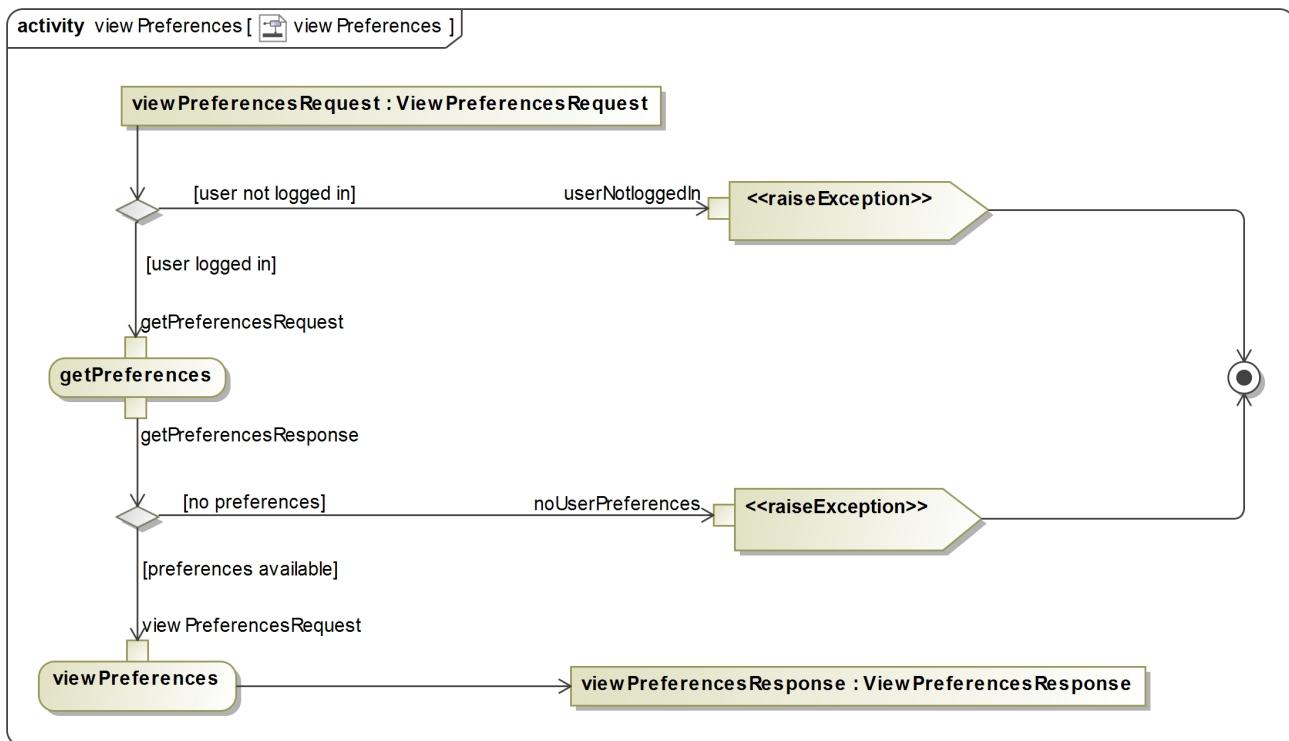


Figure 46: The activity diagram for viewPreferences

## 5.4 Domain Model

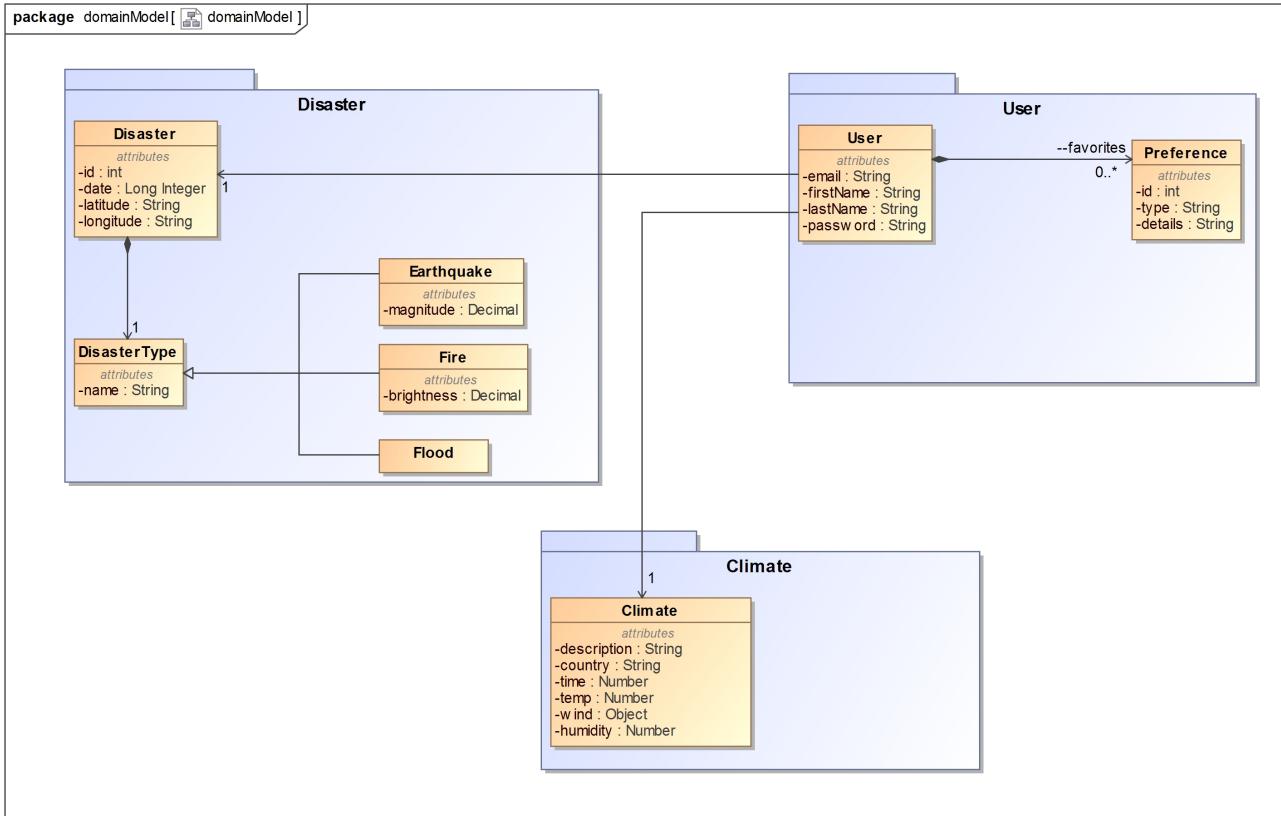


Figure 47: The domain model for the climate, disaster and user modules

Each user has a first name, last name, password and an email address. A user can also have 0 or more favourites that they can save by specifying the type of preference (disaster or weather) as well as other required details identifying the type of preference. A user, who is a client, is able to request services from the disaster and climate system components. A disaster which has an id, latitude, longitude, date and extra details also has a type. A disaster type could either be a fire, earthquake or a flood. Climate, on the other hand has a description, and values describing the country, time, temperature, wind and humidity of a specific climate.

## 6 Open Issues

There are no issues

## List of Figures

1	Overview of the System . . . . .	4
2	The scope of functionality required from the disaster module	6
3	The service contract for viewDisaster . . . . .	7
4	The functional requirements diagram for viewDisaster . . . . .	8
5	The activity diagram for viewDisaster . . . . .	9
6	The service contract for queryDisaster . . . . .	10
7	The functional requirements diagram for queryDisaster . . . . .	10
8	The activity diagram for queryDisaster . . . . .	11
9	The service contract for notifyNewDisaster . . . . .	12
10	The functional requirements diagram for notifyNewDisaster . .	12
11	The activity diagram for notifyNewDisaster . . . . .	13
12	The service contract for trackDisaster . . . . .	14
13	The functional requirements diagram for trackDisaster . . . . .	14
14	The activity diagram for trackDisaster . . . . .	15
15	The service contract for addDisasterType . . . . .	16
16	The functional requirements diagram for addDisasterType . .	16
17	The activity diagram for addDisasterType . . . . .	17
18	The service contract for modifyDisasterType . . . . .	18
19	The functional requirements diagram for modifyDisasterType	18
20	The activity diagram for modifyDisasterType . . . . .	19
21	The scope of functionality required from the climate module .	19
22	The service contract for viewWeather . . . . .	20
23	The functional requirements diagram for viewWeather . . . . .	21
24	The activity diagram for viewWeather . . . . .	22
25	The service contract for queryForecast . . . . .	23
26	The functional requirements diagram for queryForecast . . . . .	23
27	The activity diagram for queryForecast . . . . .	24
28	The scope of functionality required from the user module . .	25
29	The service contract for registerUser . . . . .	26
30	The functional requirements diagram for registerUser . . . . .	26
31	The activity diagram for registerUser . . . . .	27
32	The service contract for login . . . . .	28
33	The functional requirements diagram for login . . . . .	29
34	The activity diagram for login . . . . .	30
35	The service contract for logout . . . . .	31
36	The functional requirements diagram for logout . . . . .	31
37	The activity diagram for logout . . . . .	32
38	The service contract for addPreferences . . . . .	33
39	The functional requirements diagram for addPreferences . .	33

40	The activity diagram for addPreferences . . . . .	34
41	The service contract for editPreferences . . . . .	35
42	The functional requirements diagram for editPreferences . . .	35
43	The activity diagram for editPreferences . . . . .	36
44	The service contract for viewPreferences . . . . .	37
45	The functional requirements diagram for viewPreferences . . .	37
46	The activity diagram for viewPreferences . . . . .	38
47	The domain model for the climate, disaster and user modules	39