SAMBUG

Functional Requirements

VecTor
Software Developers

SAMBUG

October, 2015

Abrie van Aardt	13178840
Werner Mostert	13019695
Kale-ab Tessera	13048423
Keagan Thompson	13023782
Michelle Swanepoel	13066294

Contents

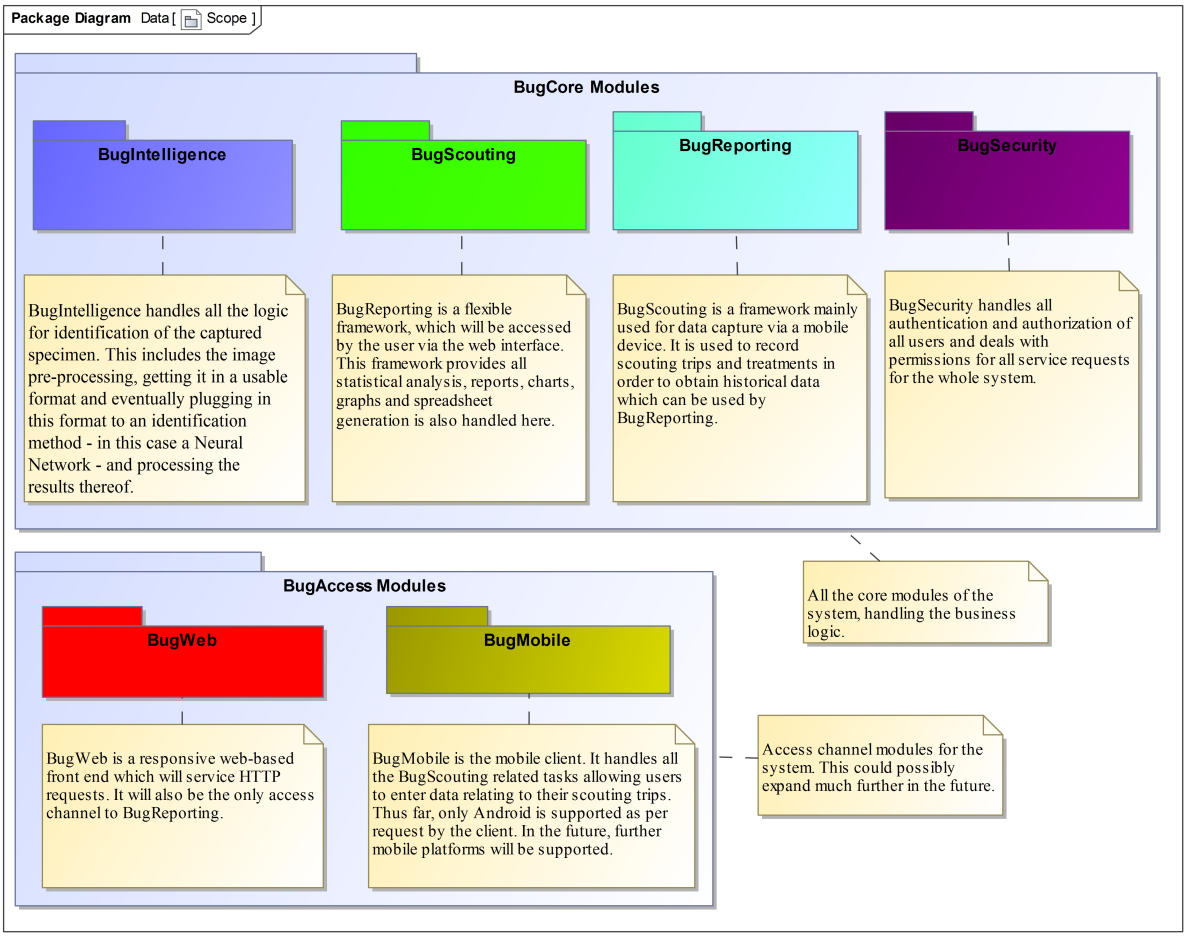
1 Background

South Africa is currently the largest producer of macadamia nuts in the world. One of the main production and quality limiting factors is the incidence of stink bug damage.
Accurate scheduling of chemical sprays rely on accurate scout data and threshold levels of the insect pests in an orchard. However, scouting for these pests has a major shortfall, namely the accurate identification of pests, despite efforts to train growers and scouts by various means.
Area wide control of pests and diseases is a concept that has been considered, but with the lack of scout data from across and within growing regions it is impossible to make such recommendations.

2 Vision

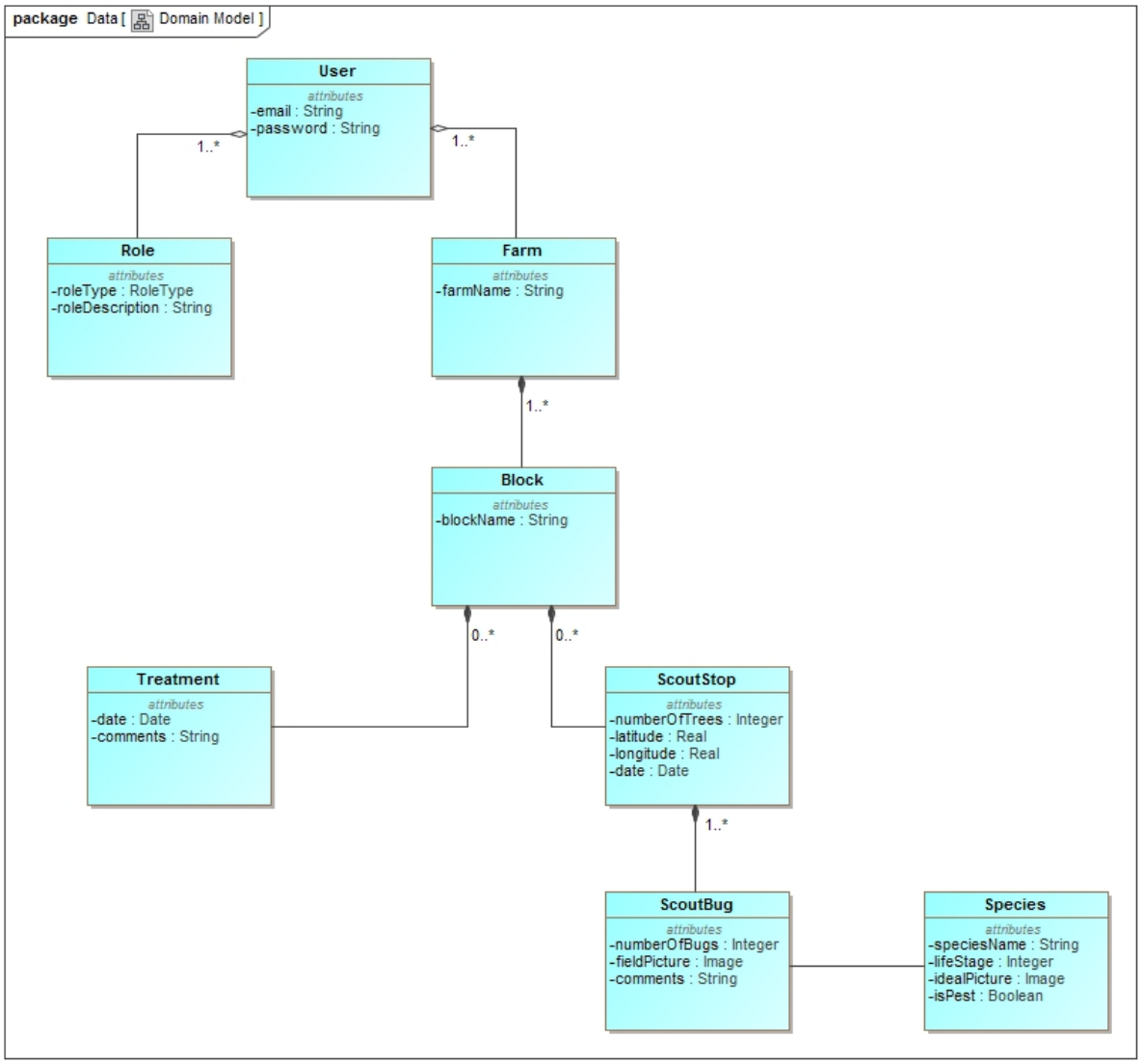
An innovative approach to handling the management and acquisition of scout data is to develop a smartphone application that is able to identify specific stink bug species by making use of the built-in camera of the smartphone. This application makes use of the smartphones built-in GPS to perform geotagging and uploading information to a central database. In addition to the data management, one of the primary goals is to supply data mining methods on historical data as obtained via the smartphone application. A visual representation of said data is therefore made available in a dynamic and easy way for the every day farmer and researcher.

3 Scope



4 Functional Requirements and Application Design

4.1 Domain Model



4.2 BugScouting

The BugScouting module has the following functionality:

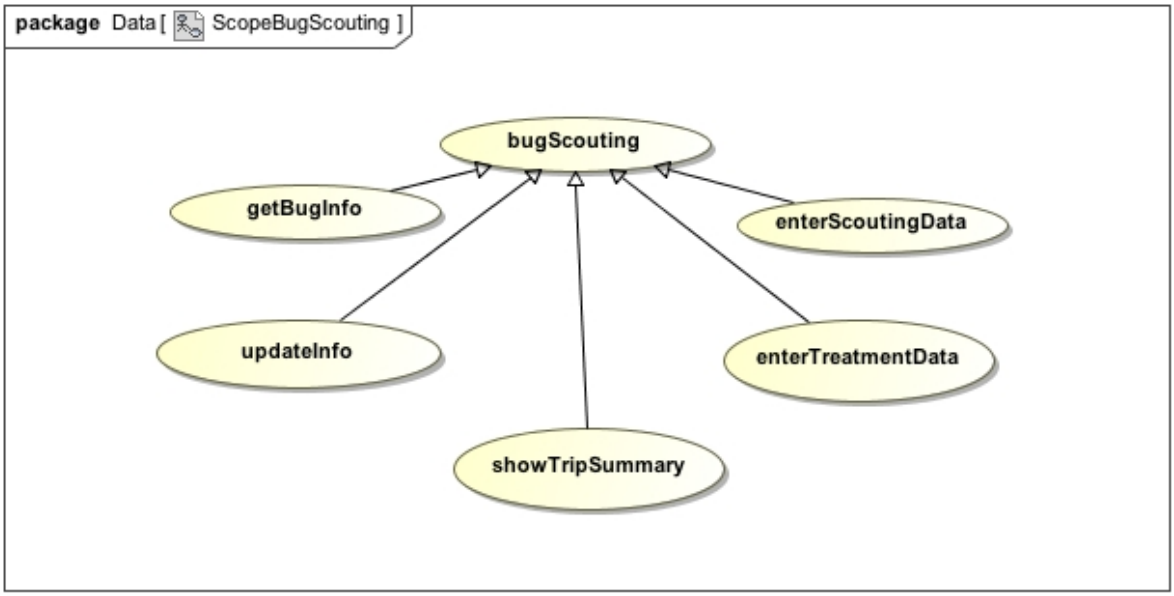
1. It provides the functionality for a user to be able to enter data related to a scouting trip - entering data such as the number of trees scouted, the number of bugs per tree and the different kind of bugs specified. After this data has been entered, a summary should be displayed for the user.

2. It provides the ability for a user to record the details related to spraying, such as the date of spraying and the specific block that was sprayed.

In many ways this module may be regarded as the "core" functionality of the system. This encompasses the main purpose of the system.

4.2.1 Module Scope

The scope of the *BugScouting* module is shown below:



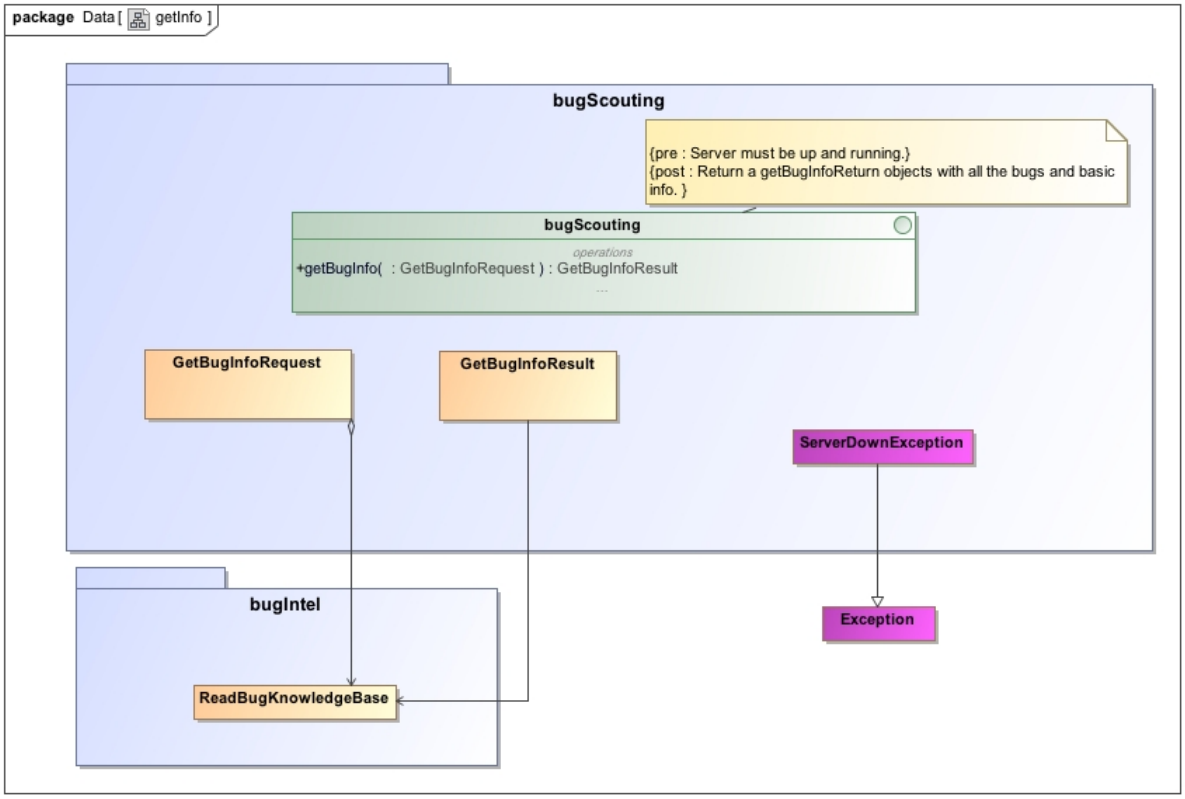
4.2.2 Use Cases

The concrete use cases for the *BugScouting* module follow:

4.2.2.1 getBugInfo ..... [Priority - Medium]

This use case uses the BugIntelligence module to retrieve information on a specific specimen which will be used to display to the user a "wiki" like piece which has the purpose of informing the user.

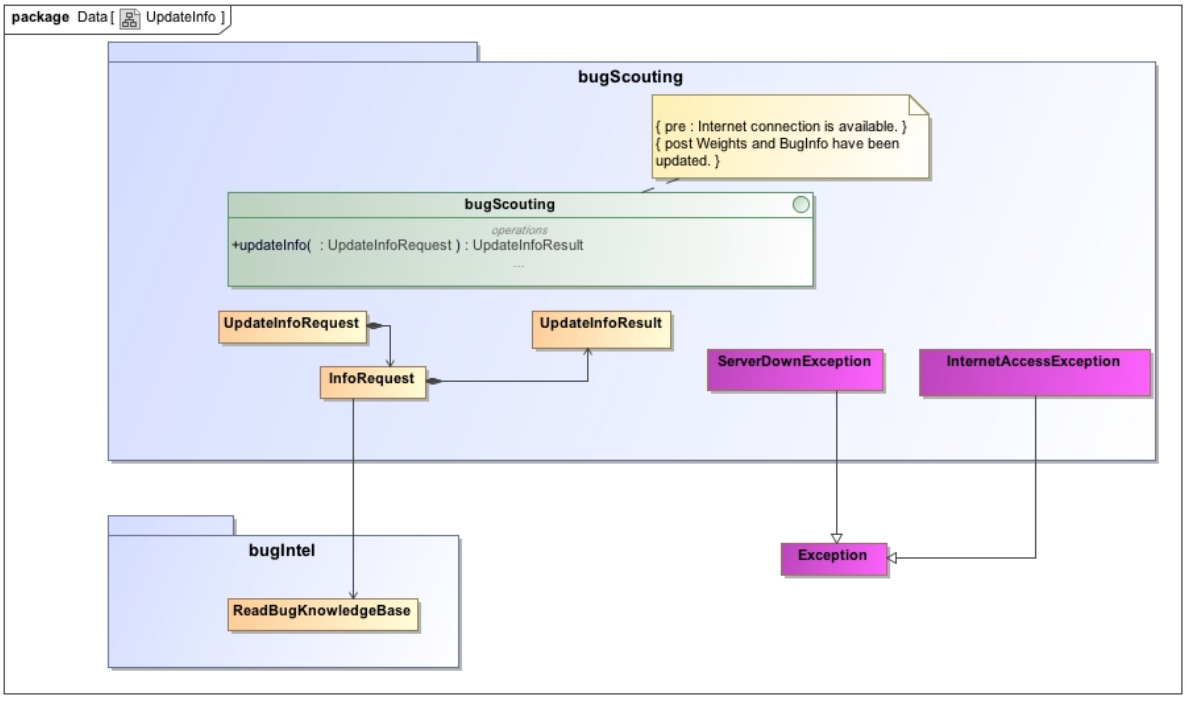
The Service Contract for the *getBugInfo* use case is shown below:



4.2.2.2 updateInfo ..... [Priority - Medium]

The objective of this use case is to provide functionality to retrieve updated bug information and identification method updates.

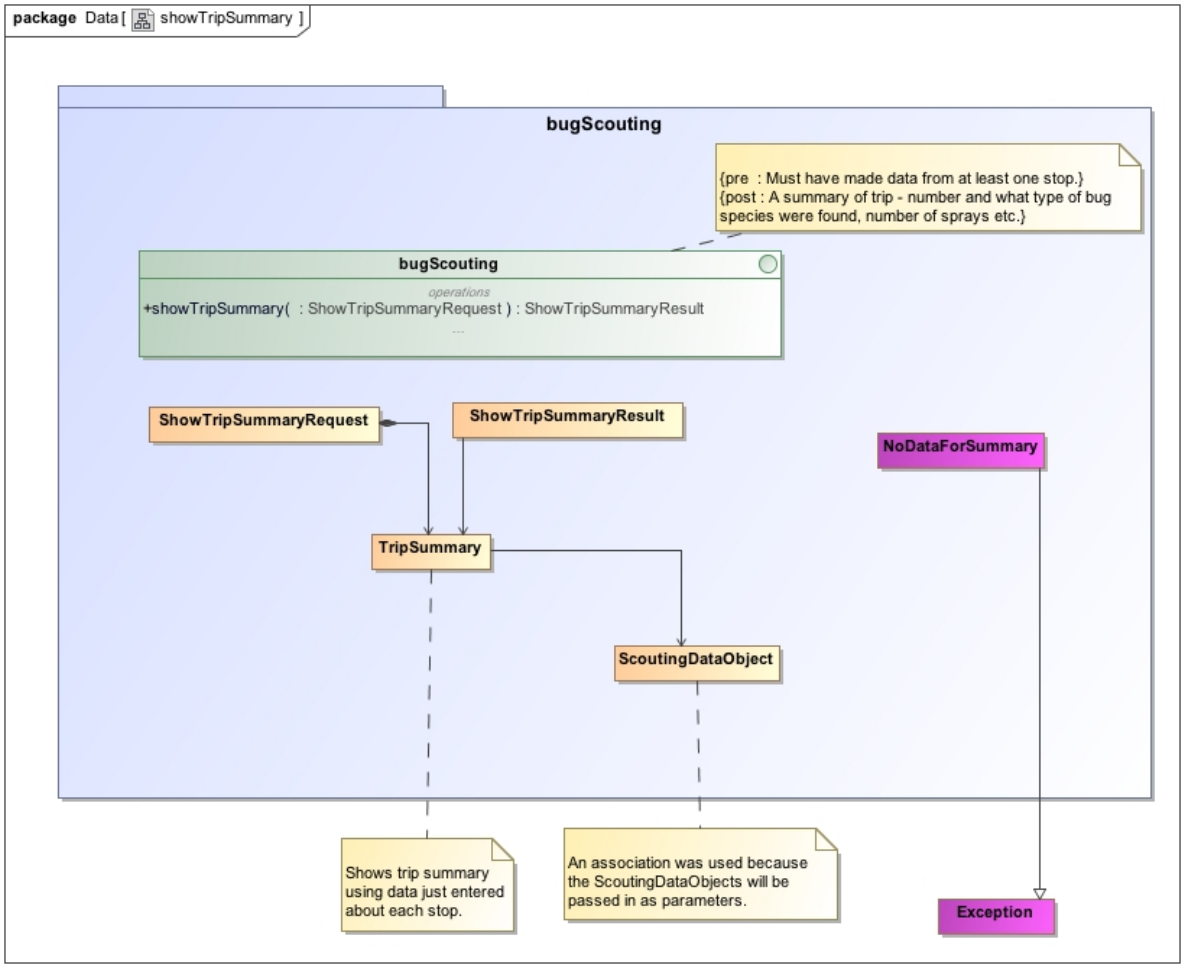
The Service Contract for the *getBugInfo* use case is shown below:



4.2.2.3 showTripSummary ..... [Priority - High]

After completing a full scouting trip, this use case should provide a summary of the entire scouting trip. One scouting trip may consist of many scouting stops and each scout stop may include the finding of various bugs. The averages for the scouting trip data is used in the summary.

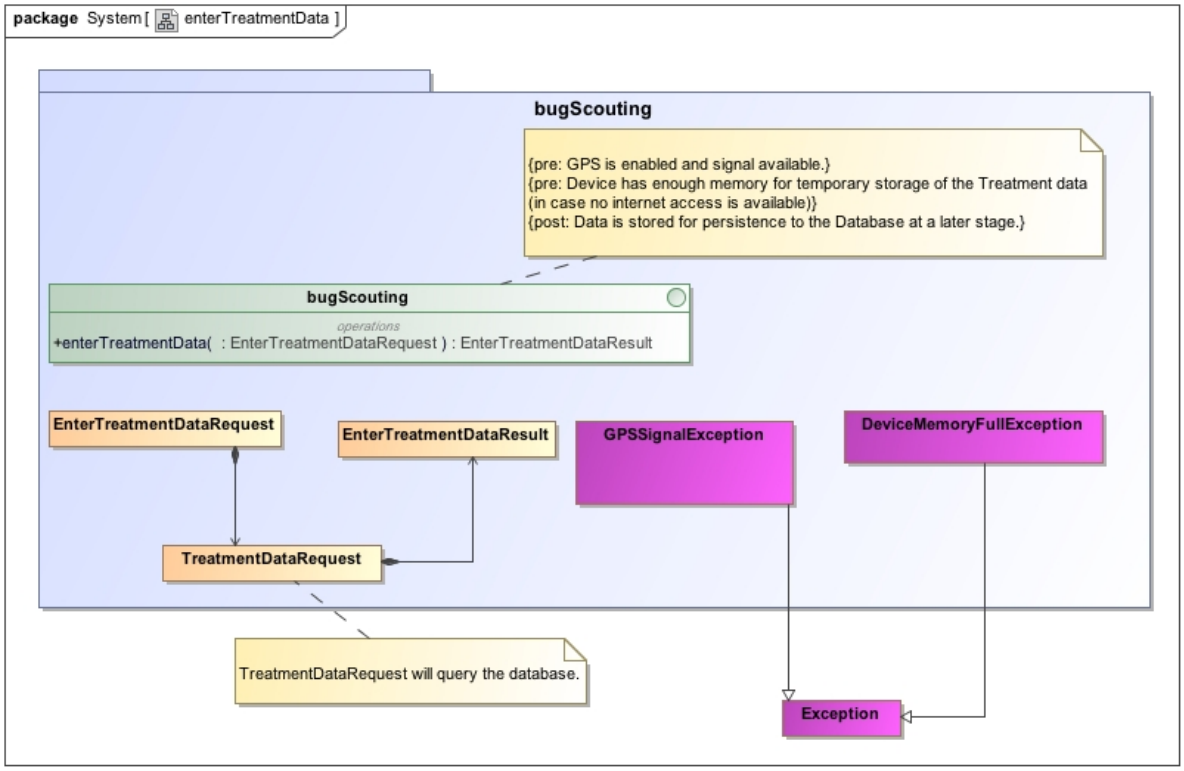
The Service Contract for the *showTripSummary* use case is shown below:



4.2.2.4 enterTreatmentData ..... [Priority - Critical]

This use case is for entering data related to spraying chemicals (pesticide). This allows for the type of chemical, the date and the block or orchard number to be recorded.

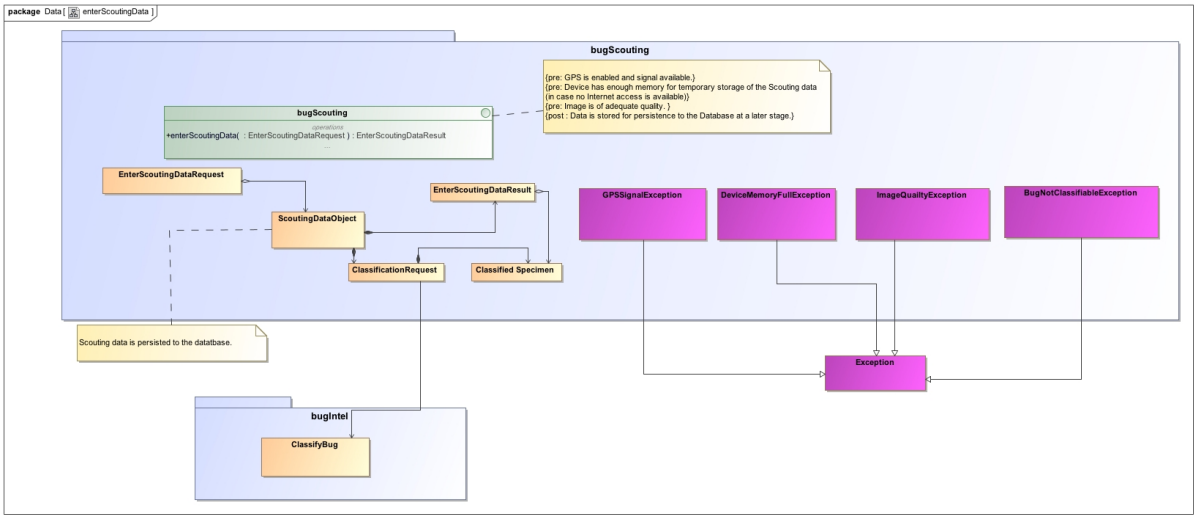
The Service Contract for the *enterTreatmentData* use case is shown below:



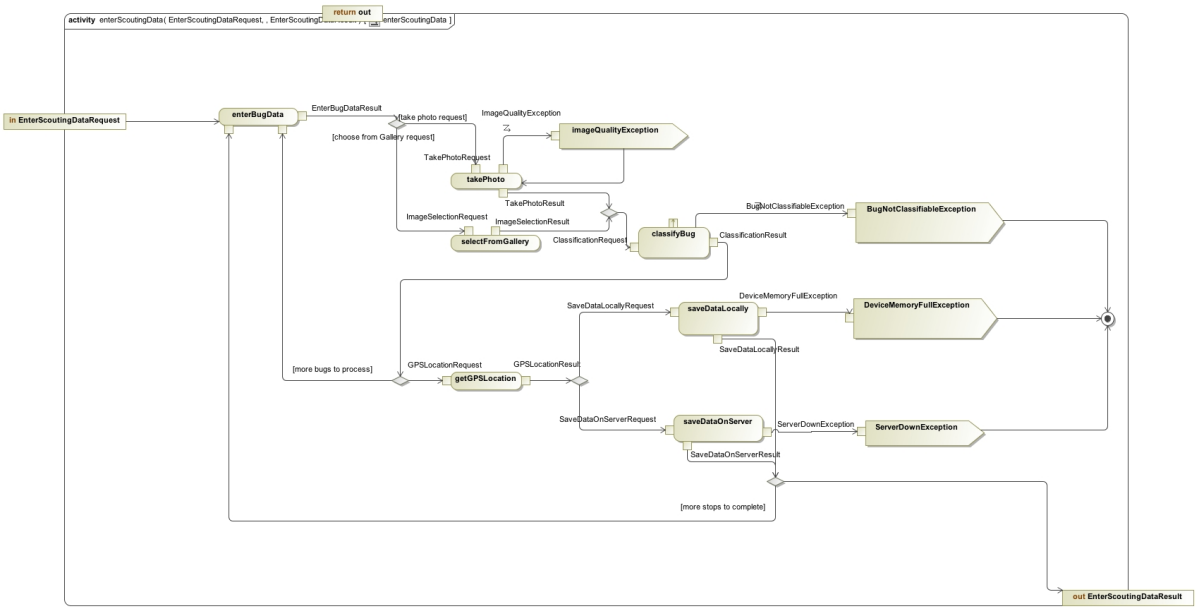
4.2.2.5 enterScoutingData ..... [Priority - Critical]

This is the core use case of this module. This use case allows entering data related to a scouting stop. Data captured should include the number of trees observed, the number of bugs encountered and the block number where the scouting took place. After entering the data, the specimen should be identified (classified), either manually or automatically, before successfully submitting it for persistence.

The Service Contract for the *enterScoutingData* use case is shown below:



The Process Specification for the *enterScoutingData* use case is shown below:



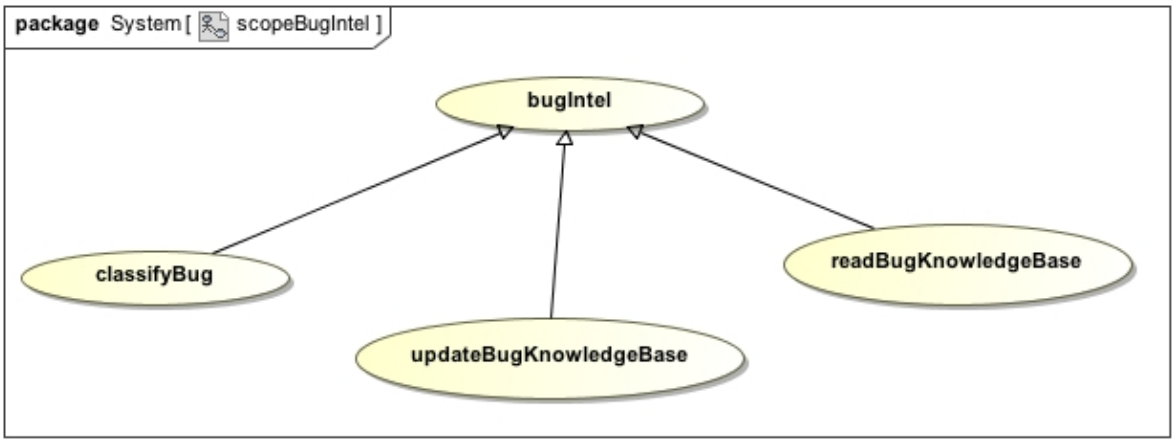
4.3 BugIntelligence

The BugIntelligence module has the following functionality:

1. It provides a pluggable method to classify a specimen according to species and life stage.
2. It provides an interface which can be used to obtain information related to any specific specimen which is identifiable by the system
3. It provides CRUD(Create Read Update Delete) functionality for bug information for specimens identifiable by the identification method

4.3.1 Module Scope

The scope of the *BugIntelligence* module is shown below:



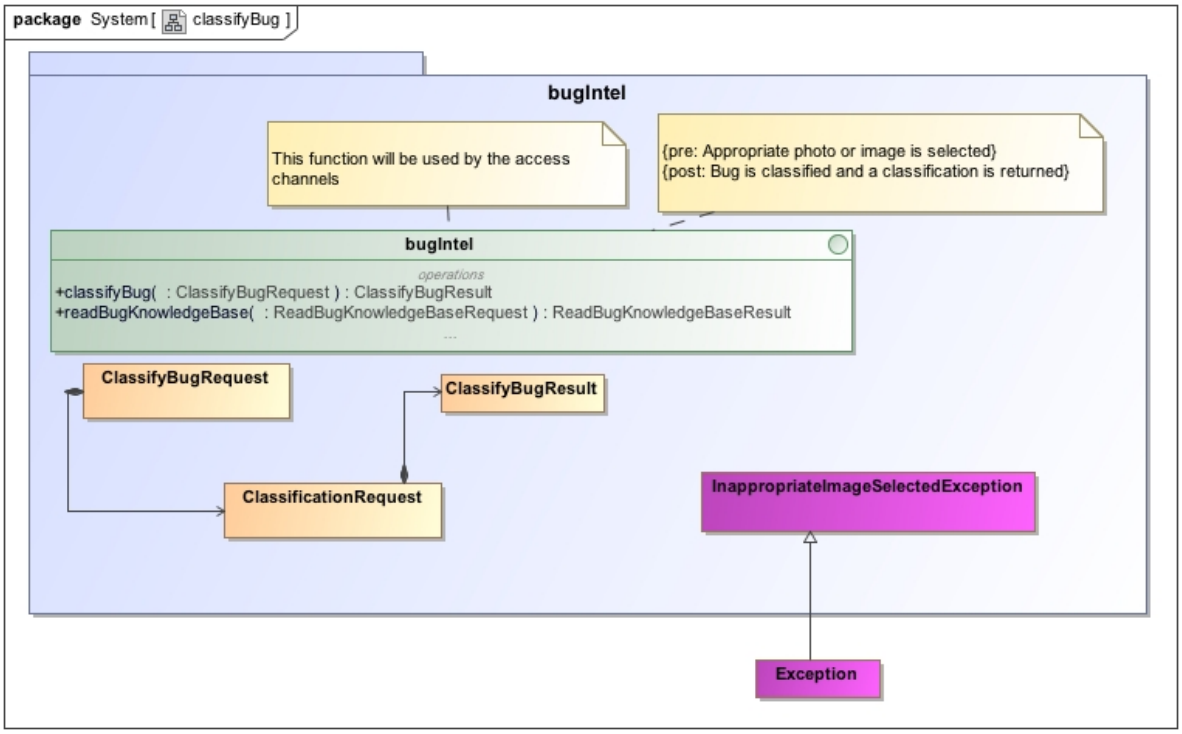
4.3.2 Use Cases

The concrete use cases for the *BugIntelligence* module follow:

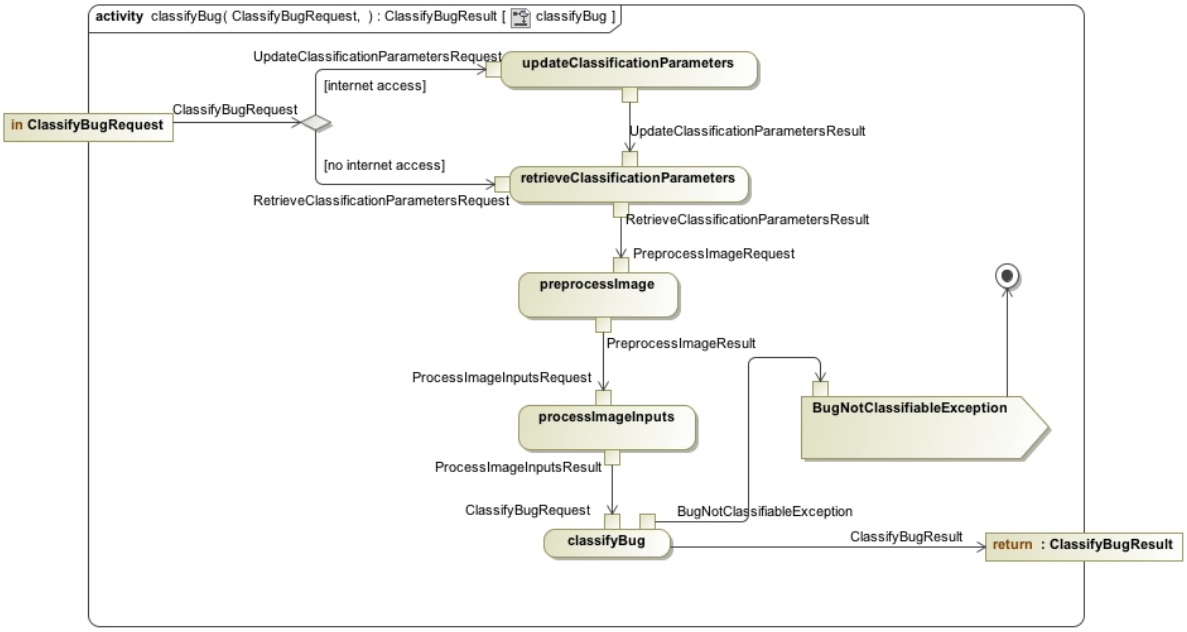
4.3.2.1 classifyBug ..... [Priority - Critical]

This use case supplies a method to classify the bug according to life stage and species. The classification method used is pluggable.

The Service Contract for the *classifyBug* use case is shown below:

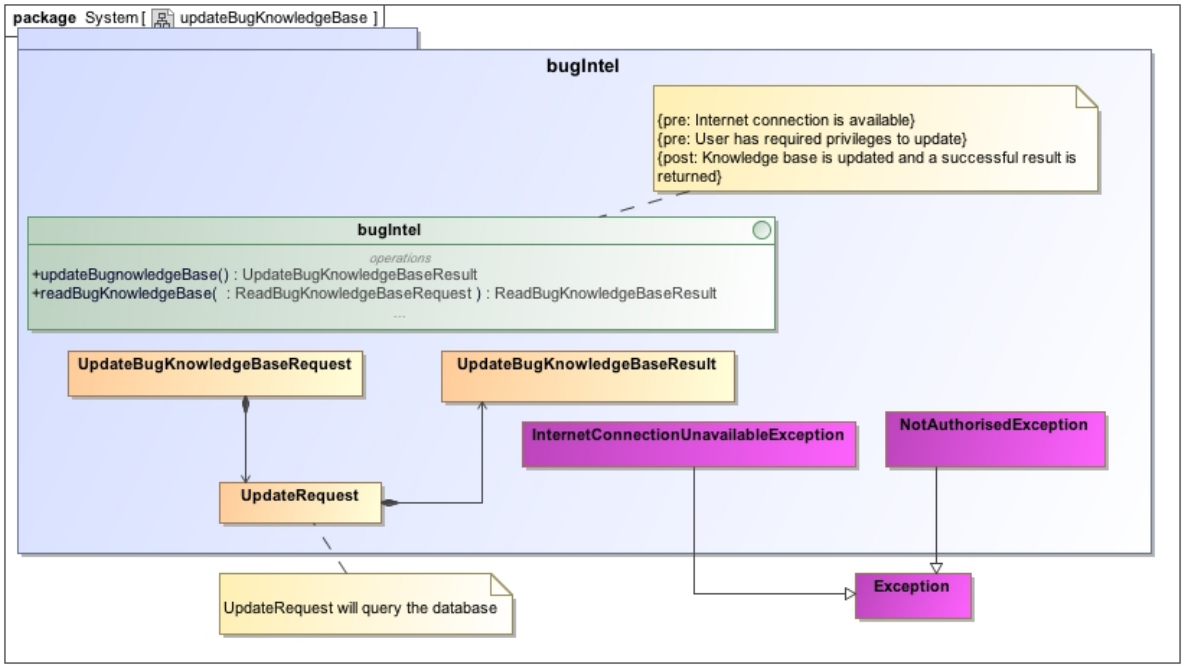


The Process Specification for the *classifyBug* use case is shown below:



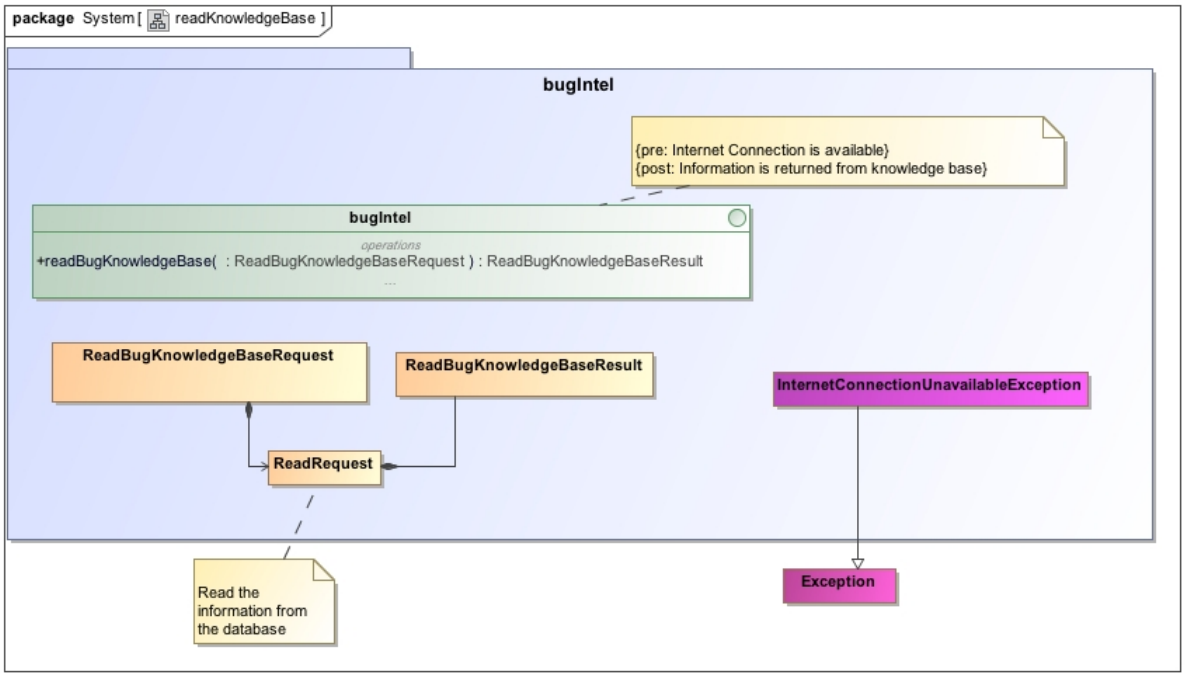
4.3.2.2 updateBugKnowledgeBase ..... [Priority - Low]

This use case provides the functionality to edit the information used for classification. As in the example of a neural network being used as a classification method, the training examples may be edited. The Service Contract for the *updateBugKnowledgeBase* use case is shown below:



4.3.2.3 readBugKnowledgeBase ..... [Priority - Critical]

This use case provides the functionality to get the information used for classification. As in the example of a neural network being used as a classification method, the training examples may be retrieved. The Service Contract for the *readBugKnowledgeBase* use case is shown below:



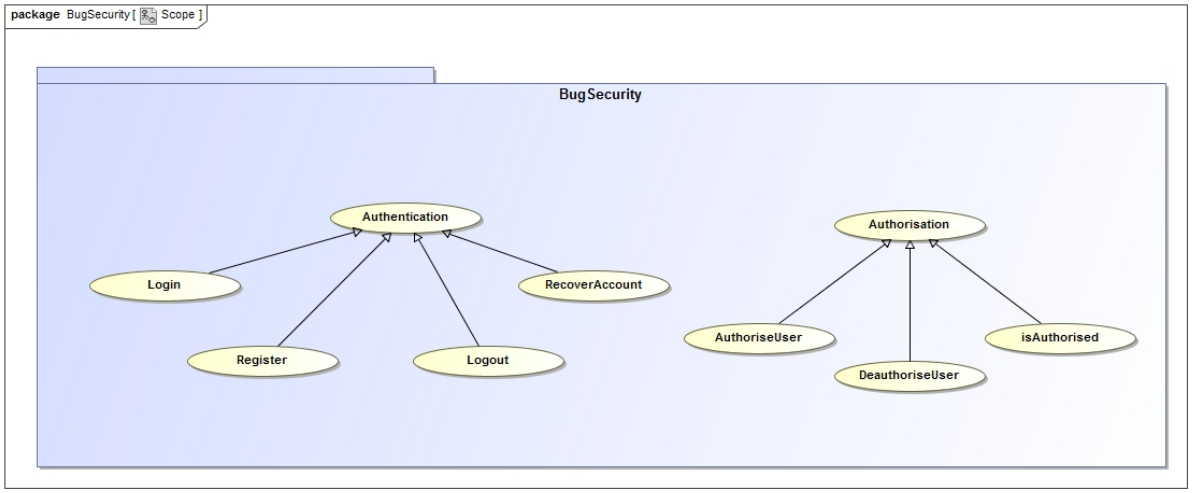
4.4 BugSecurity

The BugSecurity module allows the following functionality:

1. It provides functionality related to user accounts and user roles in order to login, register and recover your account within the capacity of a user role.
2. It provides functionality to determine whether a specific service request should be allowed.

4.4.1 Module Scope

The scope of the *BugSecurity* module is shown below:



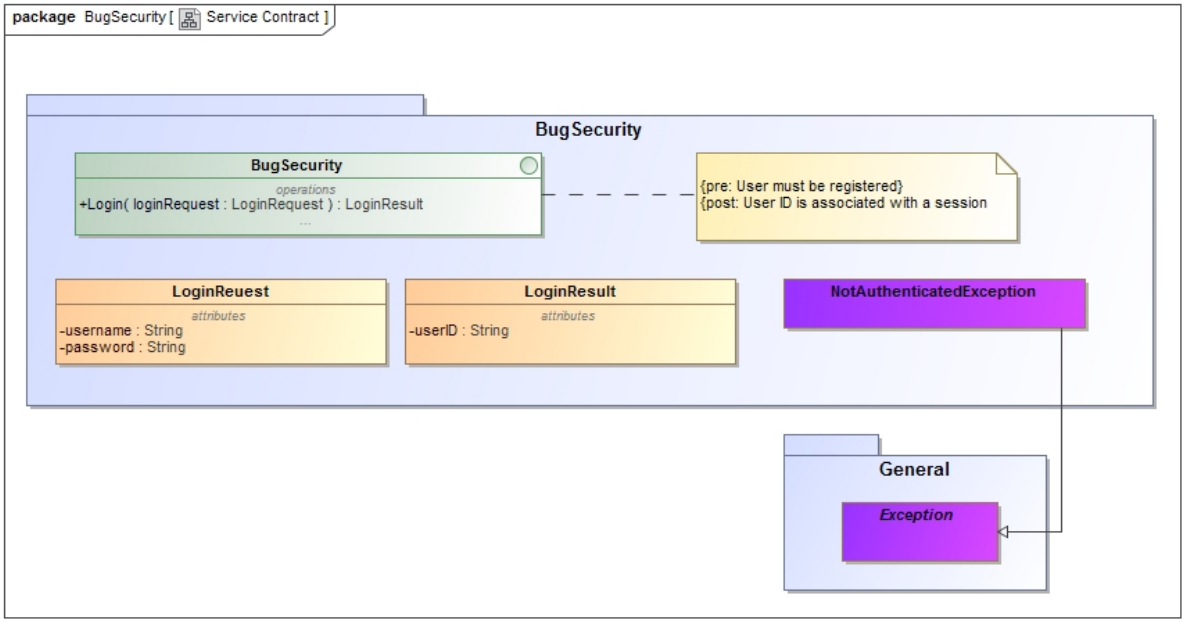
4.4.2 Use Cases

The concrete use cases for the *BugSecurity* module follow:

4.4.2.1 login ..... [Priority - Critical]

This use cases allows one to login with username and password credentials which will be validated and thus allowing the user to be authenticated.

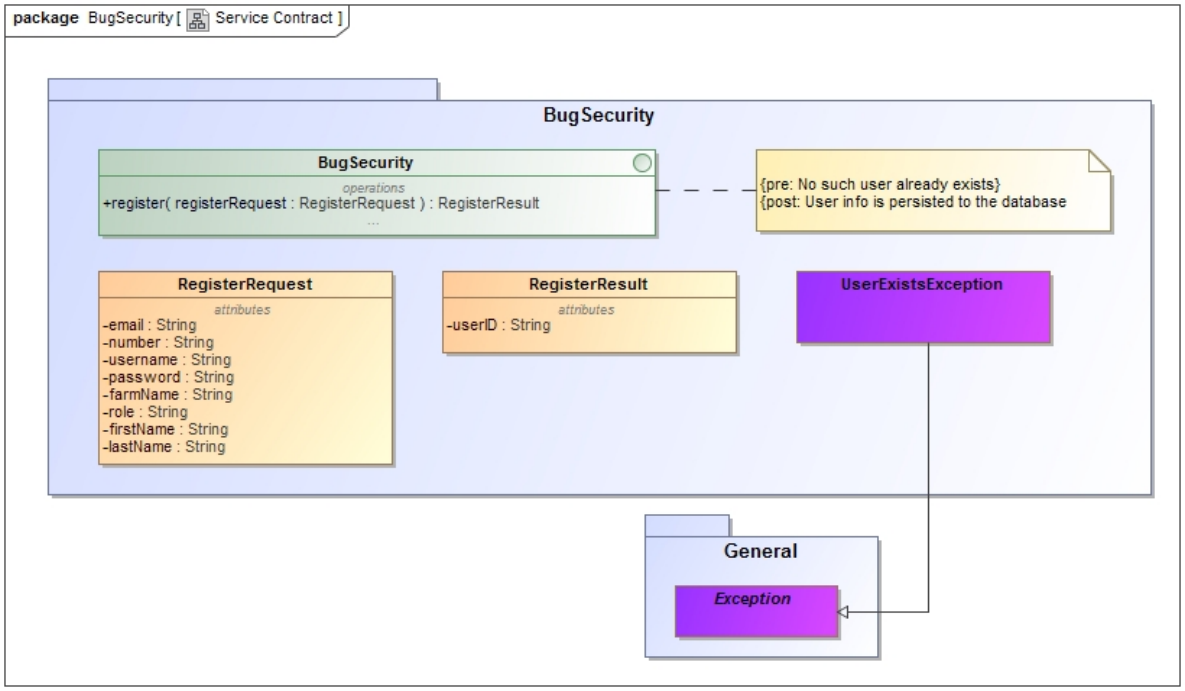
The Service Contract for the *login* use case is shown below:



4.4.2.2 register ..... [Priority - Critical]

This use case caters for registration as a new user. A user account is created which is associated with a unique user name. Currently, there are no password format requirements.

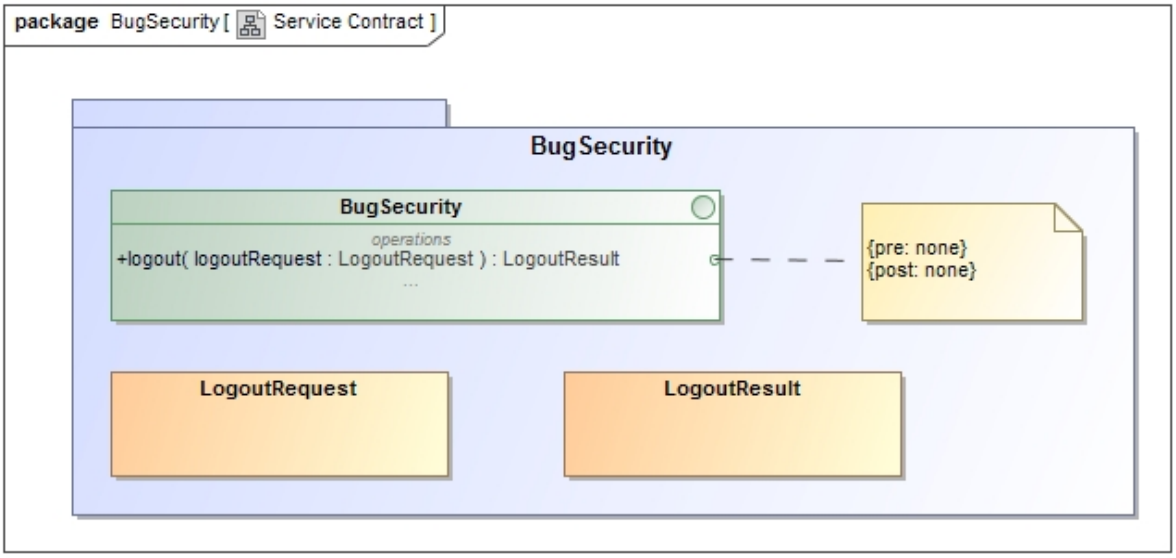
The Service Contract for the *register* use case is shown below:



4.4.2.3 logout ..... [Priority - Medium]

This use case provides functionality to log out of the system.

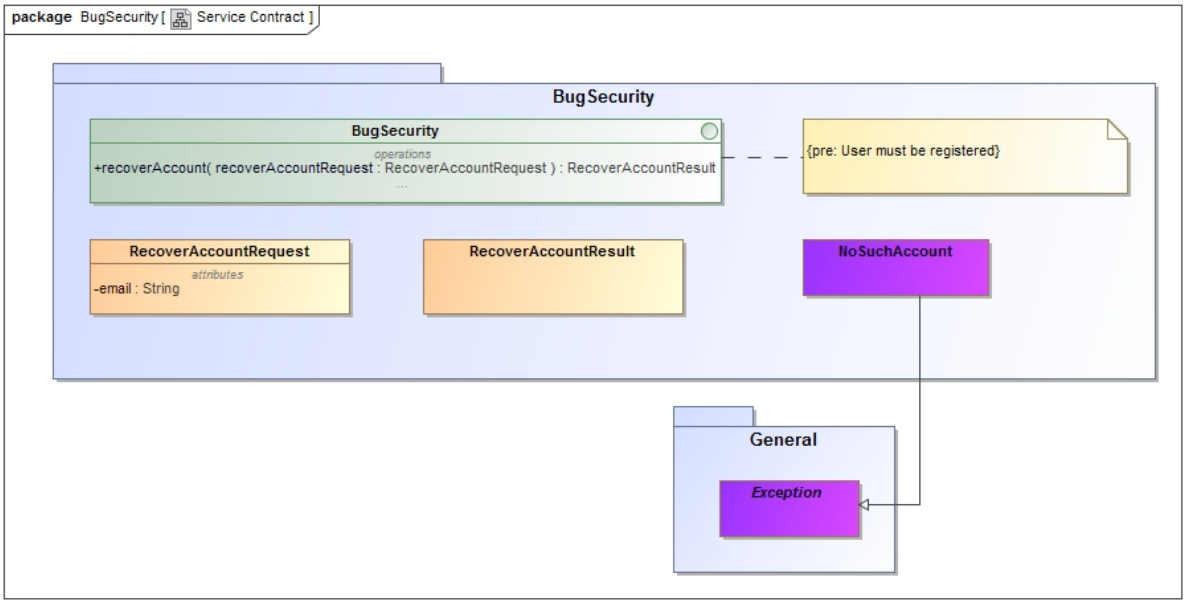
The Service Contract for the *logout* use case is shown below:



4.4.2.4 recoverAccount ..... [Priority - Low]

In the case where a user forgets or has lost his/her credentials to access the system, this use case provides functionality to recover or to generate a new password. User verification is done via the email address associated with the account.

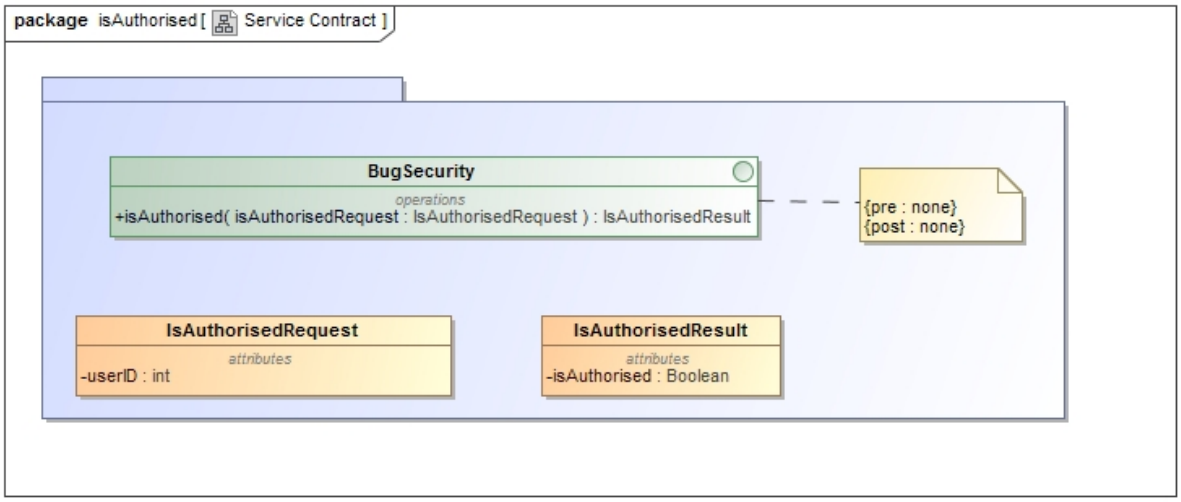
The Service Contract for the *recoverAccount* use case is shown below:



4.4.2.5 isAuthorised ..... [Priority - High]

Whenever a service which should not be accessibility to entire plethora of users is requested, authorization is required. This use case makes provision for this.

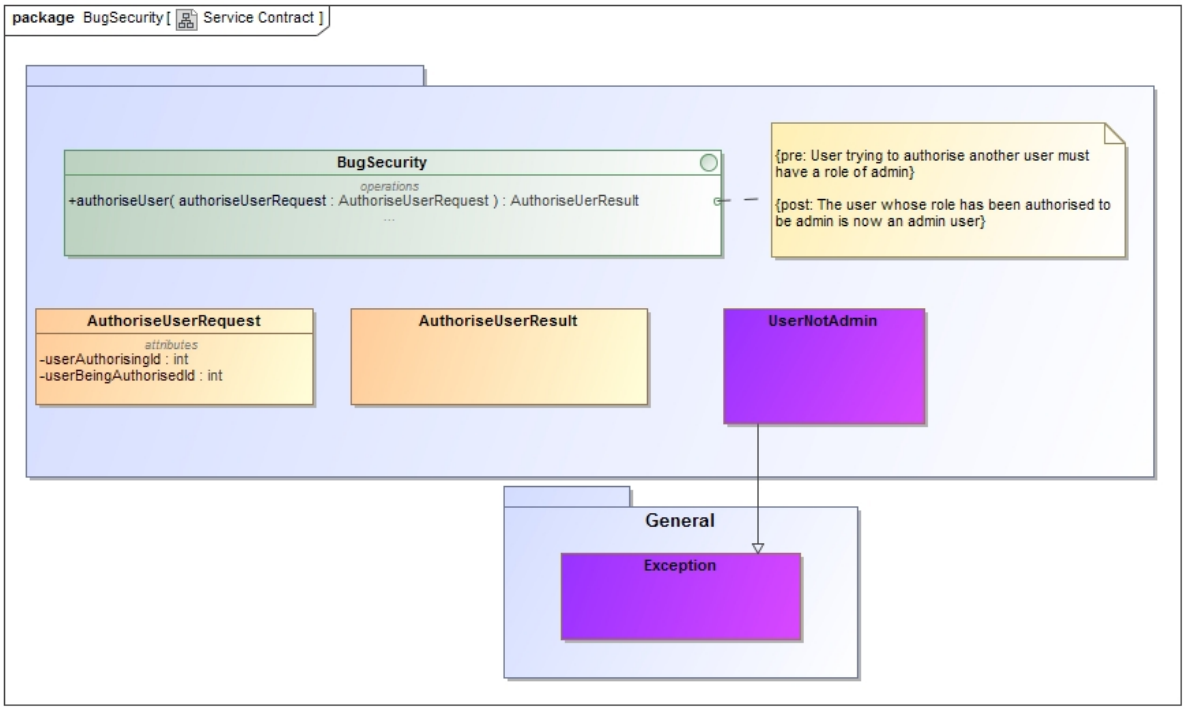
The Service Contract for the *isAuthorised* use case is shown below:



4.4.2.6 authoriseUser ..... [Priority - Medium]

This use case allows for an authorization restriction for a service to be added/

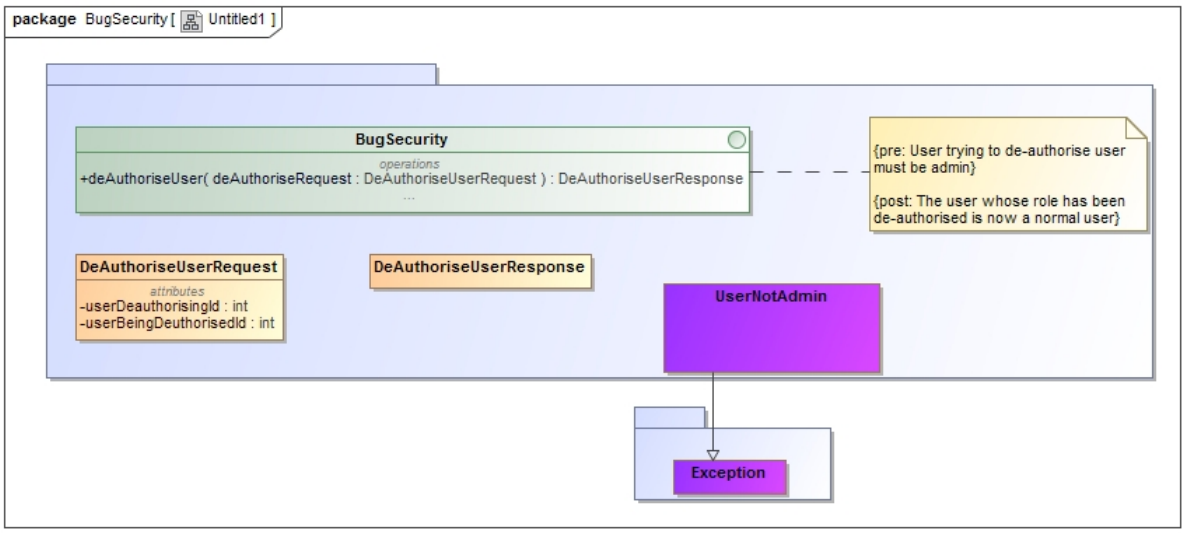
The Service Contract for the *authoriseUser* use case is shown below:



4.4.2.7 deauthoriseUser ..... [Priority - Medium]

This use case allows for an authorization restriction for a service to be removed/

The Service Contract for the *deauthoriseUser* use case is shown below:



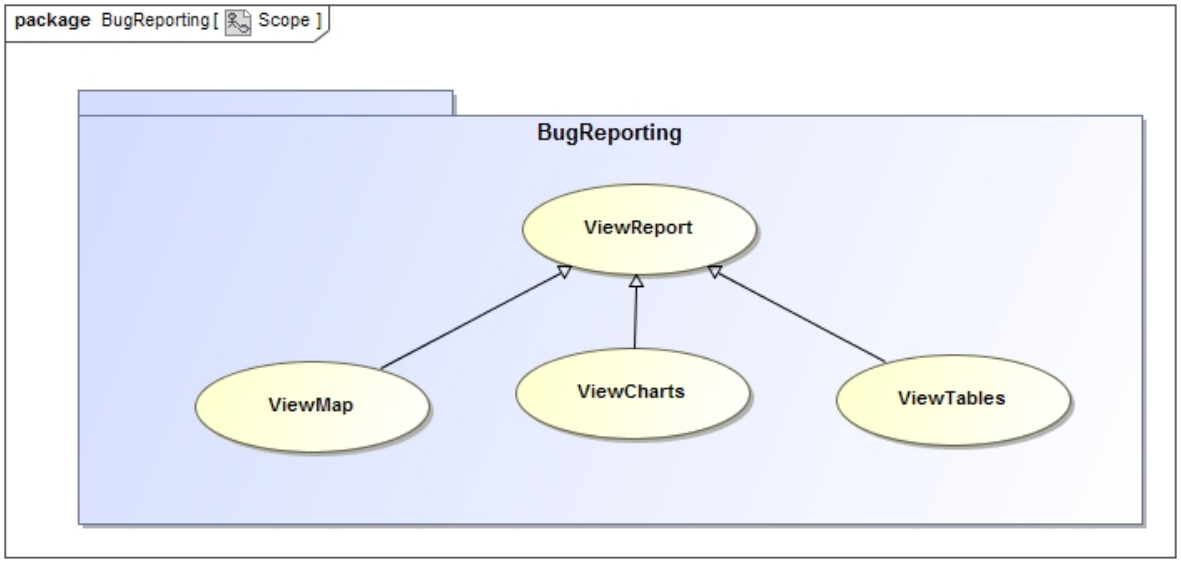
4.5 BugReporting

The BugReporting module allows the following functionality:

1. It provides functionality to generate historical data in a usable, tabular - as used by Microsoft Excel and similar software, format.
2. It provides functionality to generate a visual representation of historical data in the form of a dynamically chosen set of graphs.
3. It provides functionality to generate a heat map of the farm with regards to the population of stink bugs identified.

4.5.1 Module Scope

The scope of the *BugReporting* module is shown below:



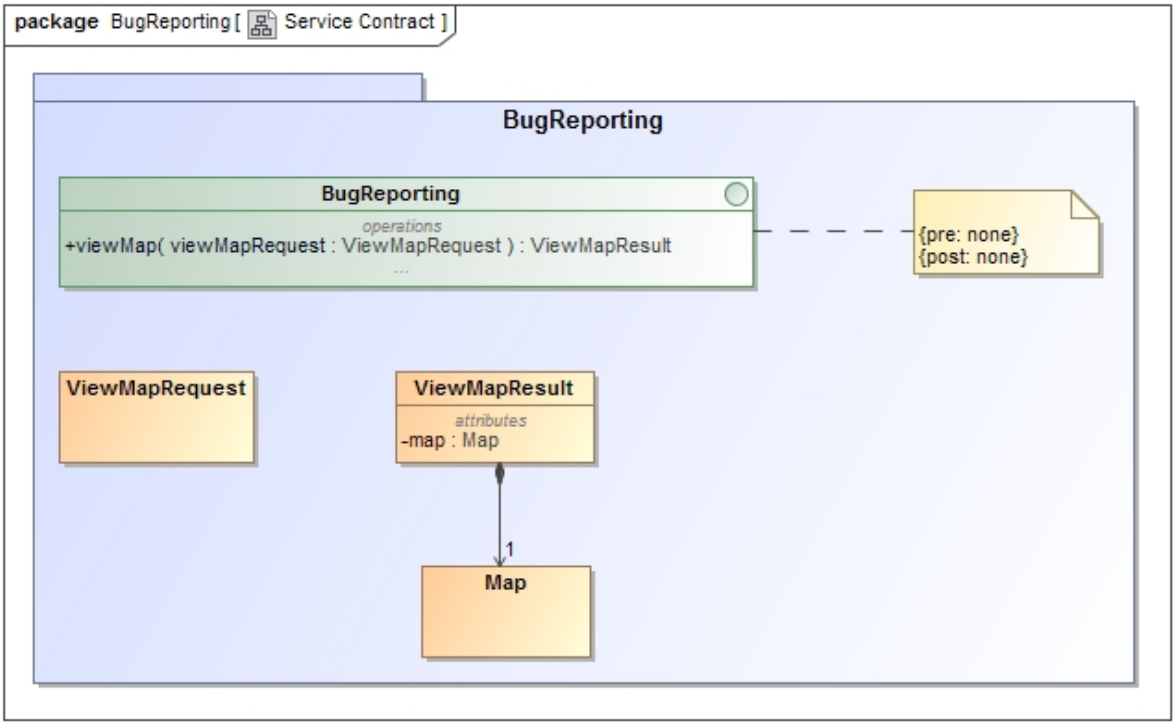
4.5.2 Use Cases

The concrete use cases for the *BugReporting* module follow:

4.5.2.1 viewMap ..... [Priority - Medium]

This use case provides the functionality to view a heat map, where the total number of pests found will be shown. Areas with more pests will show a darker area on the map. One will this be able to conclude which areas are experiencing a problem with pests. Multiple constraints and filters can be applied to show only a subset of the data available.

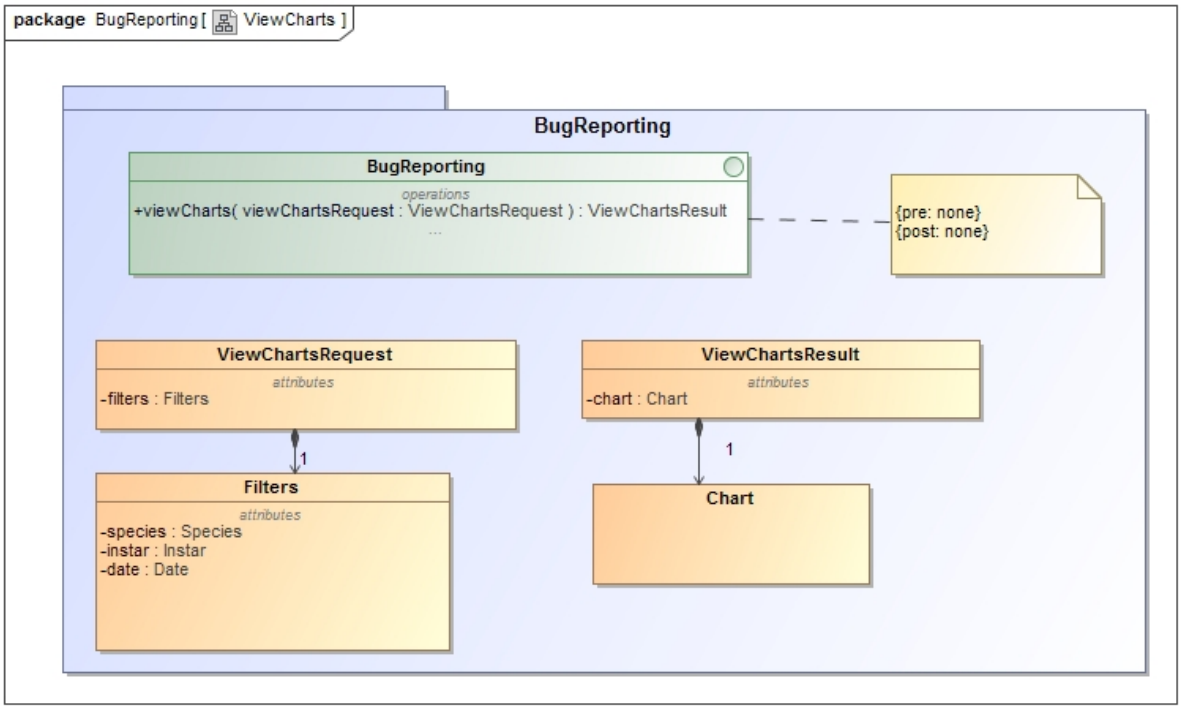
The Service Contract for the *viewMap* use case is shown below:



4.5.2.2 viewCharts ..... [Priority - Medium]

This use case provides the functionality to view data that was collected through the mobile application, specifically sccouting data. The data will be displayed in the form of dynamic charts, where one can choose multiple filters and constraints to apply to the data. One will be able to see how spraying affects the pest count.

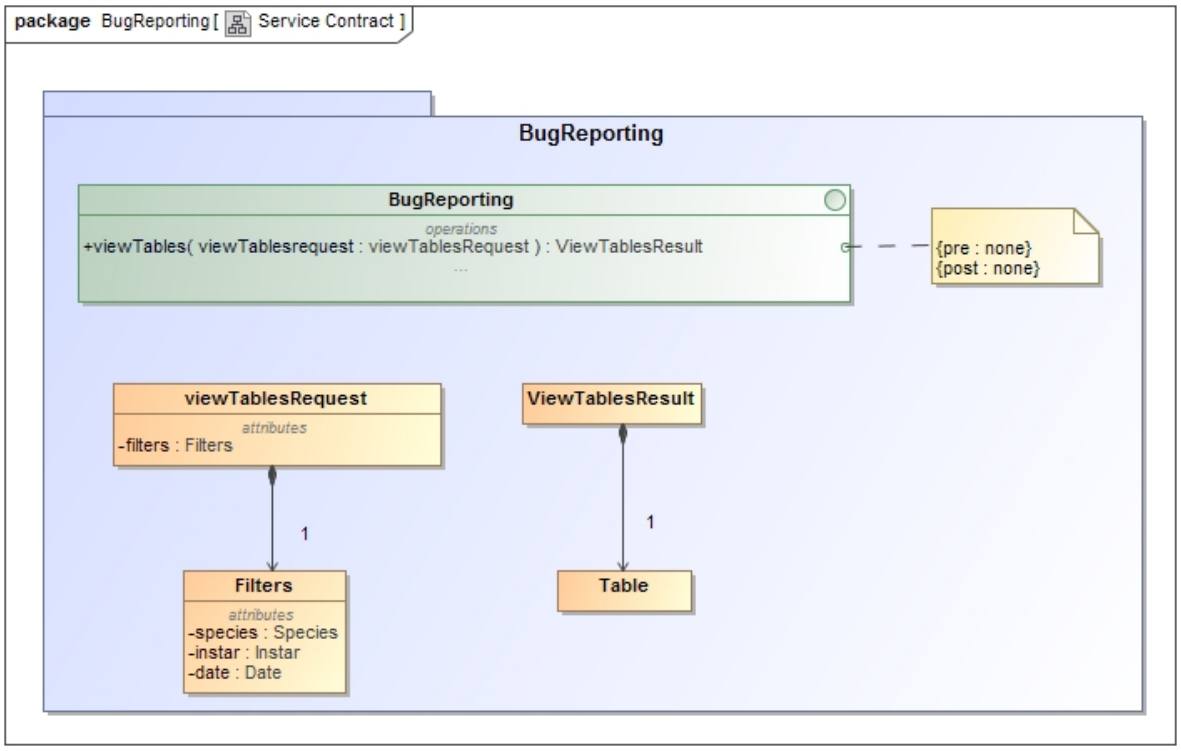
The Service Contract for the *viewCharts* use case is shown below:



4.5.2.3 viewTables ..... [Priority - Medium]

This use case allows the user to see data that has been collected through the mobile application, specifically scouting data. The data will be displayed in the form of a table and will have the functionlity to download this as an Excel spreadsheet, PDF document or to print the table. One will be able to also view spraying data. Multiple filters and constraints can be applied to the data to only view a subset of the data available.

The Service Contract for the *viewTables* use case is shown below:


package BugReporting [Service Contract]
BugReporting
BugReporting
operations
+viewTables(viewTablesrequest : viewTablesRequest) : ViewTablesResult
...
{pre : none}
{post : none}
viewTablesRequest
attributes
-filters : Filters
ViewTablesResult
1
1
Filters
attributes
-species : Species
-instar : Instar
-date : Date
Table