UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS 301

Department Of Computer Science

Application Design and Functional Requirements

Group Members:	*Student numbers:*
Diana Obo	u13134885
Priscilla Madigoe	u13049128
Kudzai Muranga	u13278012
Sandile Khumalo	u12031748

October 24, 2016

IMPAKD link

For further references see gitHub. October 24, 2016

Contents

1 Vision

The Property Investor Optimizer project is objective is to evaluate whether a certain rental property is worth buying. It does this by calculating the Return of Investment (ROI) of a property, which can be compared with another property's ROI, to assist a user to optimize their investment strategy according to their portfolio.

The project will assist the user by helping to answer the following questions:

- Given a certain bond (interest rate, deposit as a percentage of property value), rental (occupancy rate, agent commission, rental amount) and environmental conditions (Interest rate, inflation) what is the ROI?
- When is it better to pay a higher or lower deposit for a bond?
- Between two rental scenarios which provides the greater ROI?
- Is it better to try and pay off the bond as fast as possible by paying in extra capital?
- How does purchasing another property influence a user's ROI and at which point would this be a good idea?
- At which point does it make sense to buy another property?
- How much tax will the user have to pay?

2 Background

The project was given to us by our client, CSIR, so that we can research how the ROI of different configurations of rental properties can answer the questions listed in the Vision section of this document. Answers to these questions can be used to help users of the system choose to buy the best property that fits their portfolio and requirements with the ease of not having to manually evaluate the property themselves. The project can also be used for property-related research.

3 Functional Requirements and Application Design

3.1 Use case prioritization

Critical:

- Property
    1. updateProperty
    2. addProperty
    3. deleteProperty
    4. viewProperty
    5. compareTwoProperties
- Accounting
    1. calculateNPV
    2. calculateROI

Important:

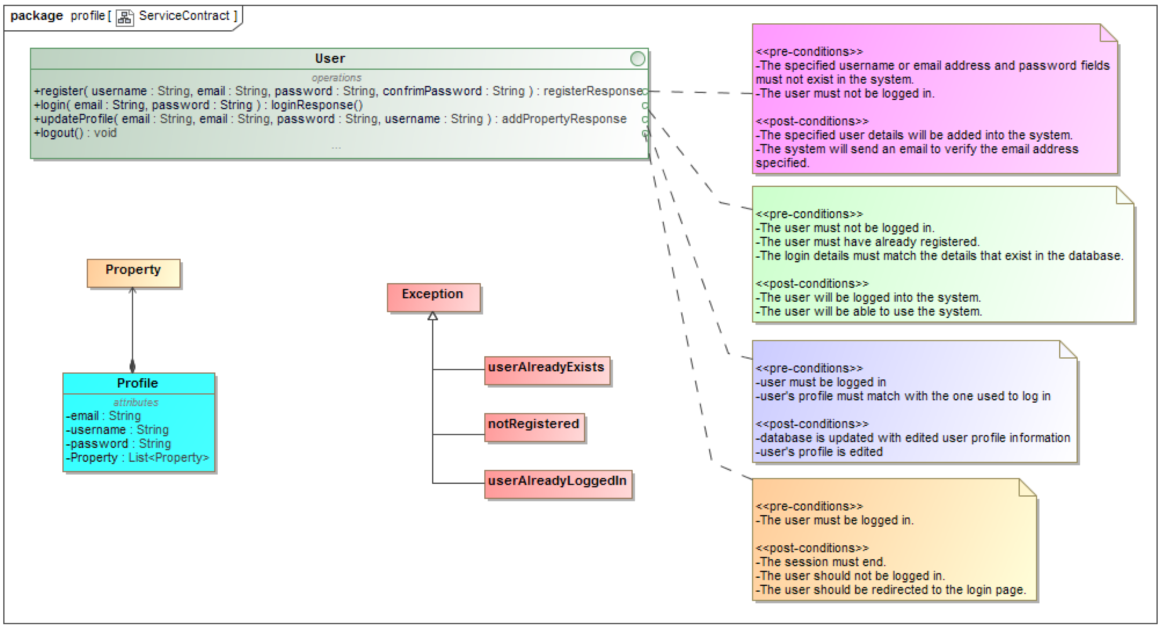
- Profile
    1. login
    2. logout
    3. register
    4. viewProfile
    5. updateProfile

Nice-to-have:

- Report
    1. generateReport
    2. producePDF
    3. viewReport

3.2 Use case/Services contracts

3.2.1 Profile


package profile [ServiceContract]
User
operations
+register(username : String, email : String, password : String, confrimPassword : String) : registerResponse
+login(email : String, password : String) : loginResponse()
+updateProfile(email : String, email : String, password : String, username : String) : addPropertyResponse
+logout() : void
...
Property
Profile
attributes
-email : String
-username : String
-password : String
-Property : List<Property>
Exception
userAlreadyExists
notRegistered
userAlreadyLoggedIn

<<pre-conditions>>
-The specified username or email address and password fields must not exist in the system.
-The user must not be logged in.

<<post-conditions>>
-The specified user details will be added into the system.
-The system will send an email to verify the email address specified.

<<pre-conditions>>
-The user must not be logged in.
-The user must have already registered.
-The login details must match the details that exist in the database.

<<post-conditions>>
-The user will be logged into the system.
-The user will be able to use the system.

<<pre-conditions>>
-user must be logged in
-user's profile must match with the one used to log in

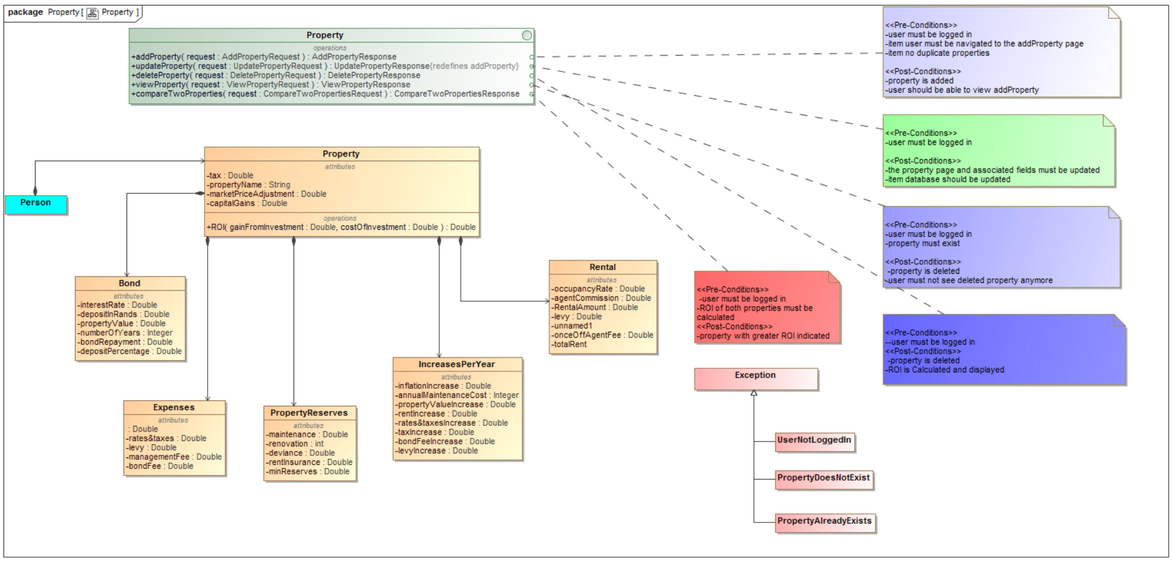
<<post-conditions>>
-database is updated with edited user profile information
-user's profile is edited

<<pre-conditions>>
-The user must be logged in.

<<post-conditions>>
-The session must end.
-The user should not be logged in.
-The user should be redirected to the login page.

3.2.2 Property

updateProperty


package Property [Property]
Property
operations
+addProperty(request : AddPropertyRequest) : AddPropertyResponse
+updateProperty(request : UpdatePropertyRequest) : UpdatePropertyResponse{redefines addProperty}
+deleteProperty(request : DeletePropertyRequest) : DeletePropertyResponse
+viewProperty(request : ViewPropertyRequest) : ViewPropertyResponse
+compareTwoProperties(request : CompareTwoPropertiesRequest) : CompareTwoPropertiesResponse
Person
Property
attributes
-tax : Double
-propertyName : String
-marketPriceAdjustment : Double
-capitalGains : Double
operations
+ROI(gainFromInvestment : Double, costOfInvestment : Double) : Double
Bond
attributes
-interestRate : Double
-depositInRands : Double
-propertyValue : Double
-numberOfYears : Integer
-bondRepayment : Double
-depositPercentage : Double
Rental
attributes
-occupancyRate : Double
-agentCommission : Double
-RentalAmount : Double
-levy : Double
-unnamed1
-onceOffAgentFee : Double
-totalRent
IncreasesPerYear
attributes
-inflationIncrease : Double
-annualMaintenanceCost : Integer
-propertyValueIncrease : Double
-rentIncrease : Double
-rates&taxesIncrease : Double
-taxIncrease : Double
-bondFeeIncrease : Double
-levyIncrease : Double
Expenses
attributes
: Double
-rates&taxes : Double
-levy : Double
-managementFee : Double
-bondFee : Double
PropertyReserves
attributes
-maintenance : Double
-renovation : int
-deviance : Double
-rentInsurance : Double
-minReserves : Double
Exception
UserNotLoggedIn
PropertyDoesNotExist
PropertyAlreadyExists

<<Pre-Conditions>>
-user must be logged in
-item user must be navigated to the addProperty page
-item no duplicate properties

<<Post-Conditions>>
-property is added
-user should be able to view addProperty

<<Pre-Conditions>>
-user must be logged in

<<Post-Conditions>>
-the property page and associated fields must be updated
-item database should be updated

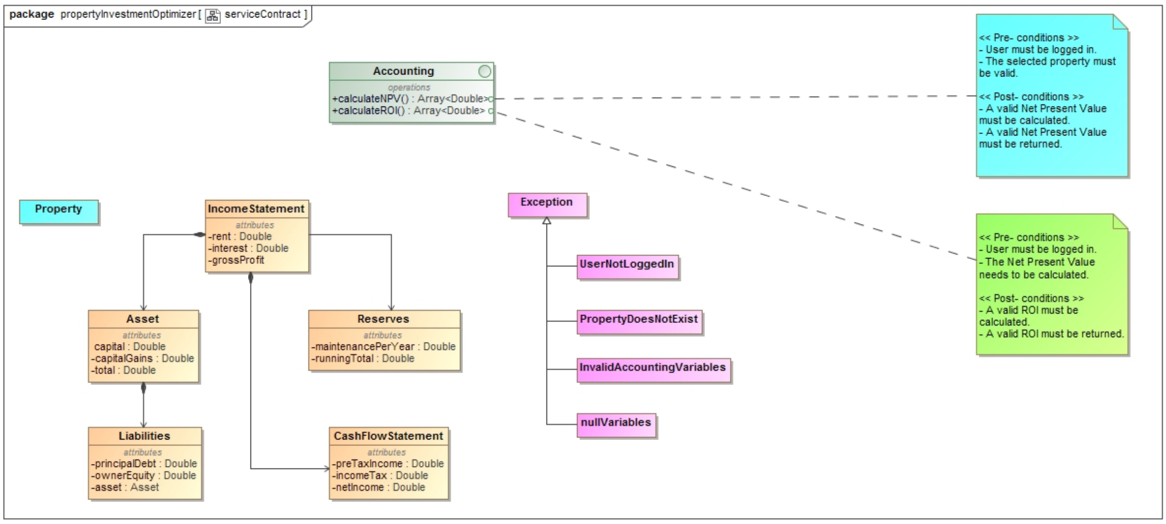
<<Pre-Conditions>>
-user must be logged in
-property must exist

<<Post-Conditions>>
-property is deleted
-user must not see deleted property anymore

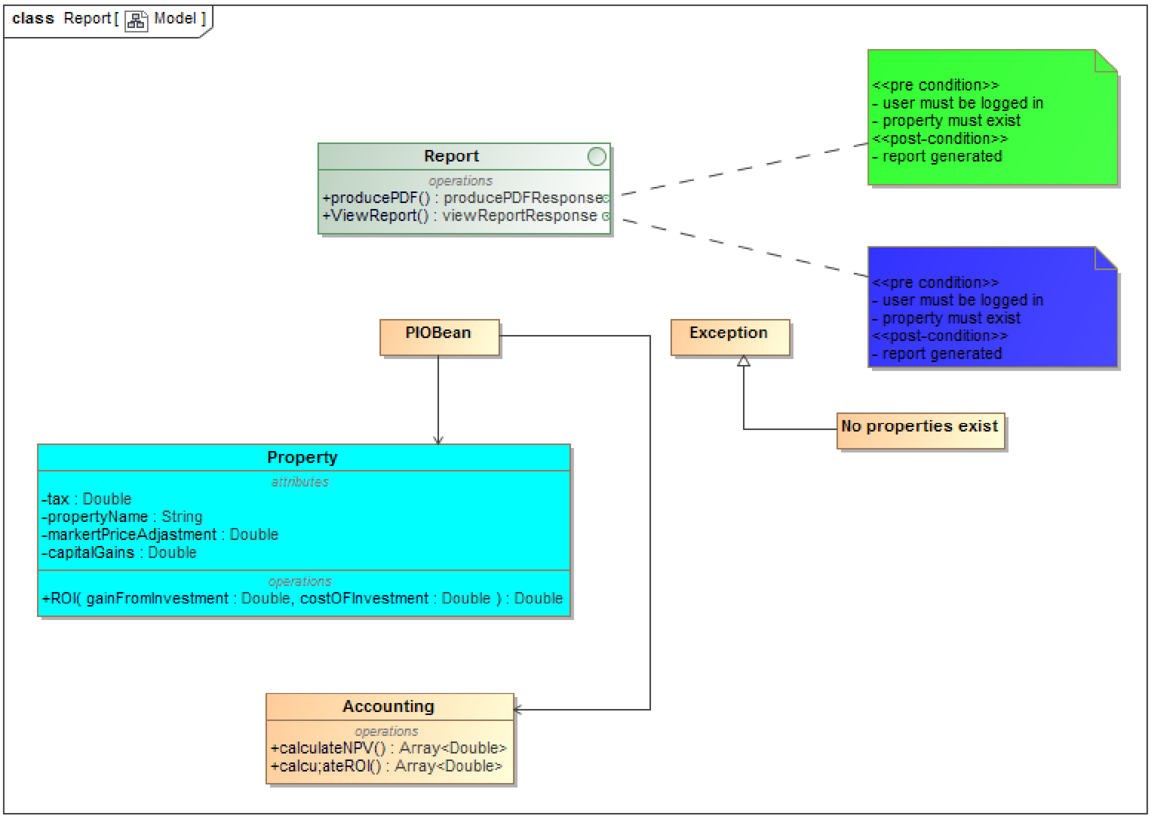
<<Pre-Conditions>>
--user must be logged in
<<Post-Conditions>>
-property is deleted
-ROI is Calculated and displayed

<<Pre-Conditions>>
-user must be logged in
-ROI of both properties must be calculated
<<Post-Conditions>>
-property with greater ROI indicated

3.2.3 Accounting

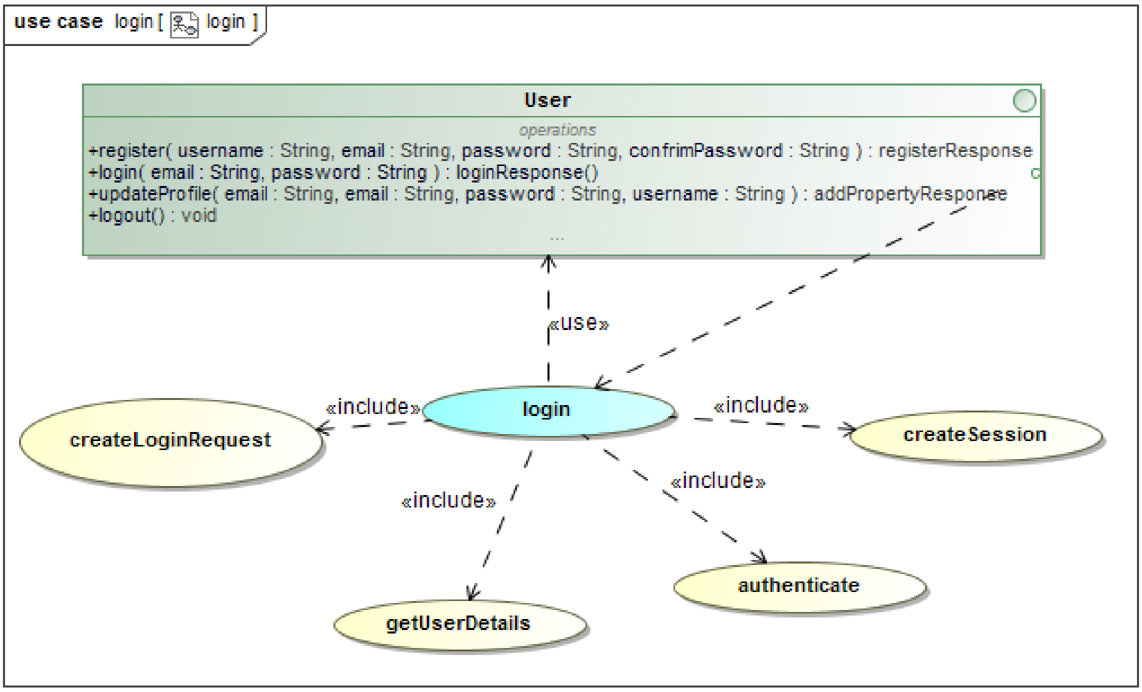


3.2.4 Report

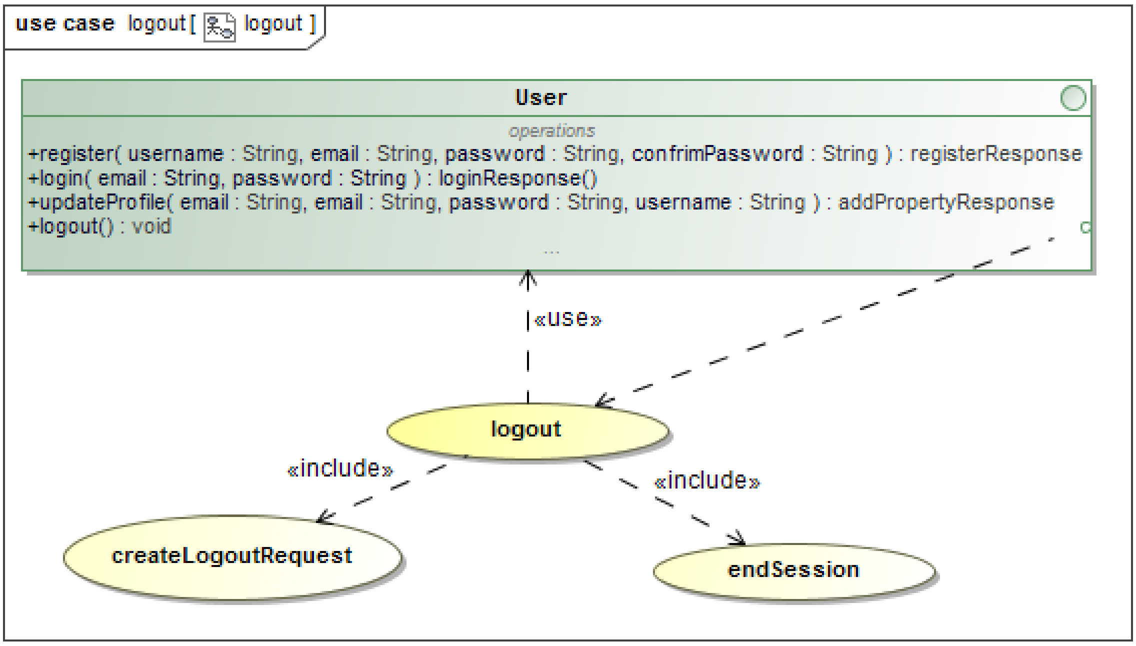


3.3 Required functionality

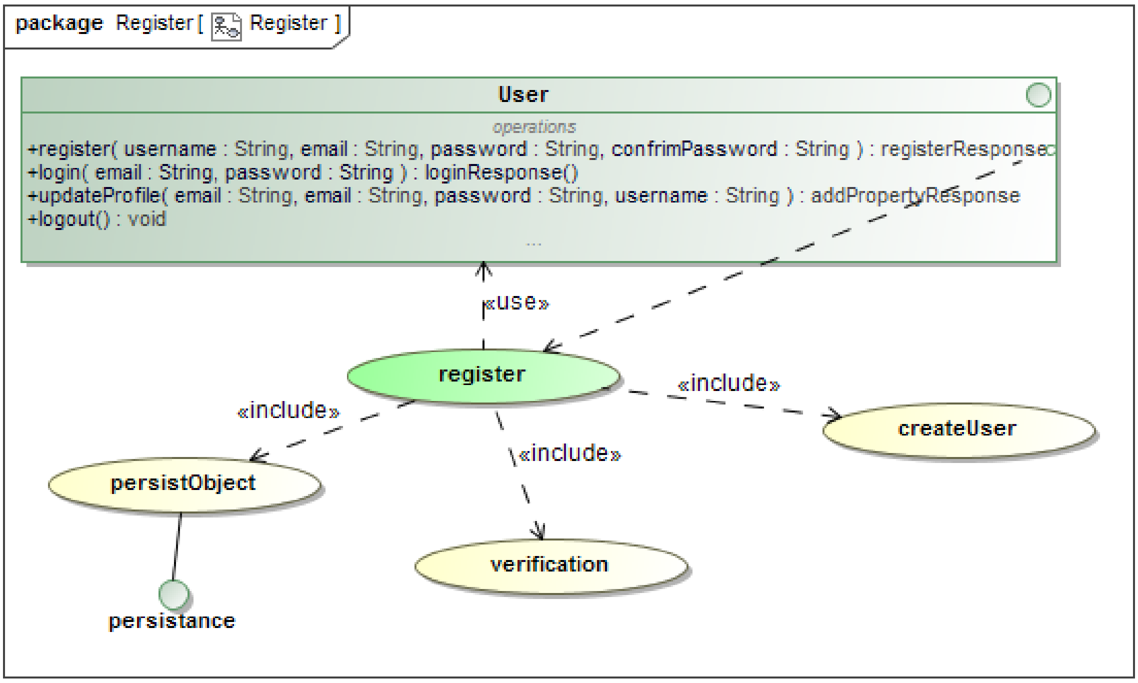
3.3.1 Login


use case login [login]
User
operations
+register(username : String, email : String, password : String, confrimPassword : String) : registerResponse
+login(email : String, password : String) : loginResponse()
+updateProfile(email : String, email : String, password : String, username : String) : addPropertyResponse
+logout() : void
...
«use»
login
«include»
createLoginRequest
«include»
createSession
«include»
getUserDetails
«include»
authenticate

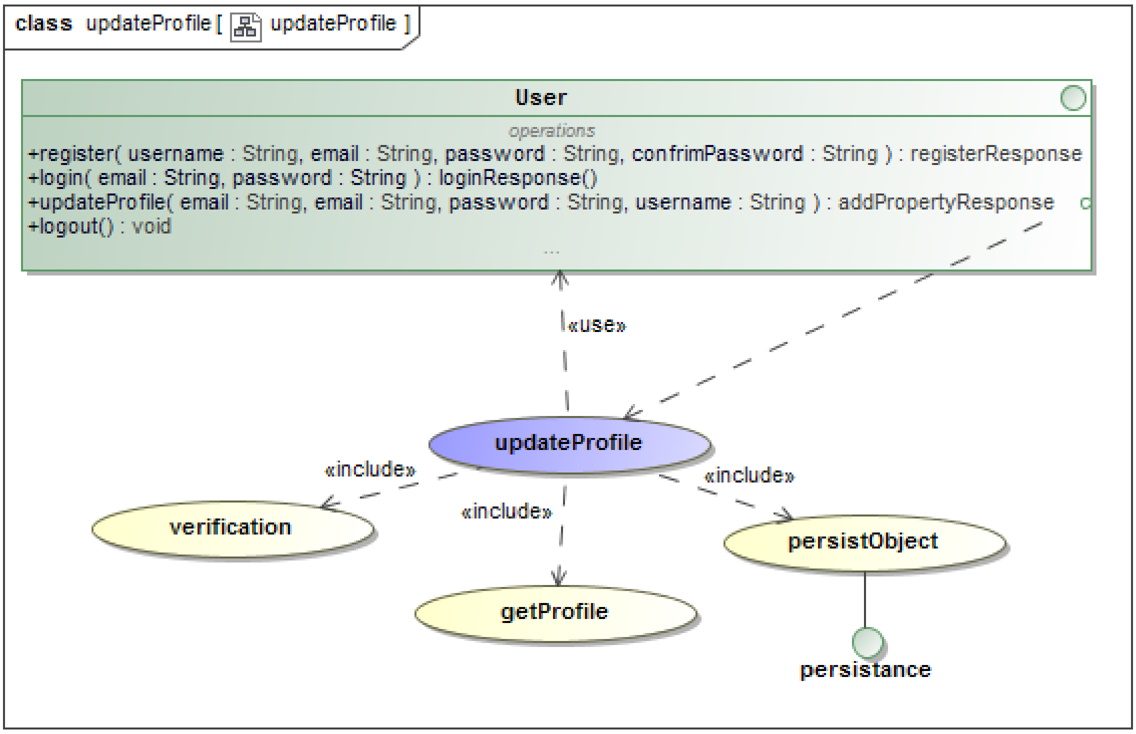
3.3.2 Logout


use case logout [logout]
User
operations
+register(username : String, email : String, password : String, confrimPassword : String) : registerResponse
+login(email : String, password : String) : loginResponse()
+updateProfile(email : String, email : String, password : String, username : String) : addPropertyResponse
+logout() : void
...
«use»
logout
«include»
createLogoutRequest
«include»
endSession

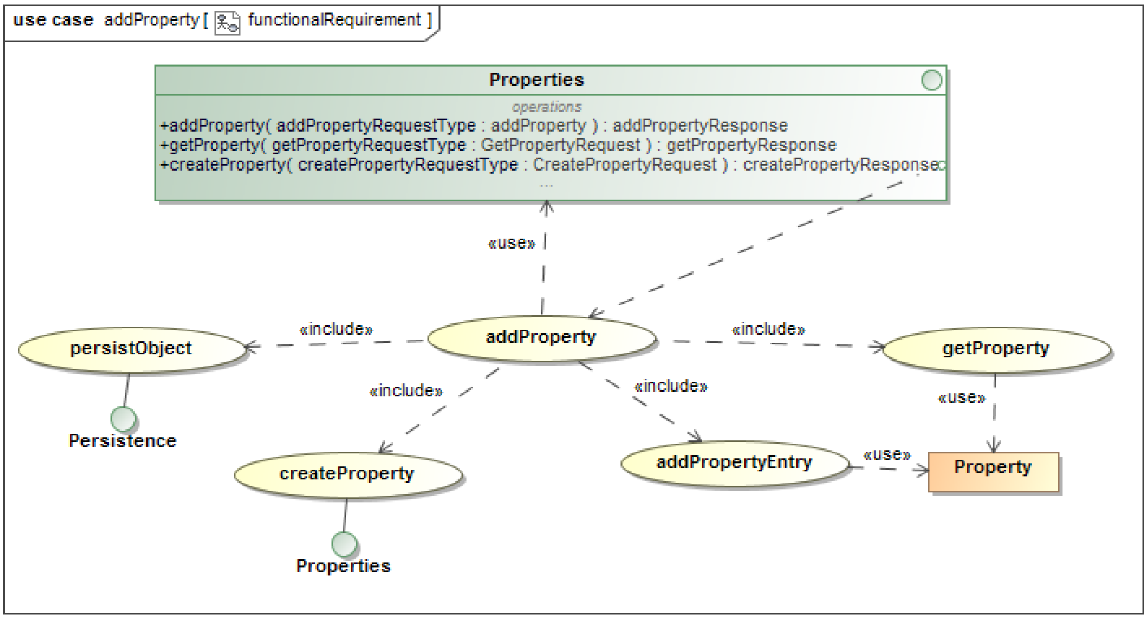
3.3.3 Register


package Register [Register]
User
operations
+register(username : String, email : String, password : String, confrimPassword : String) : registerResponse
+login(email : String, password : String) : loginResponse()
+updateProfile(email : String, email : String, password : String, username : String) : addPropertyResponse
+logout() : void
...
«use»
register
«include»
«include»
«include»
createUser
persistObject
verification
persistance

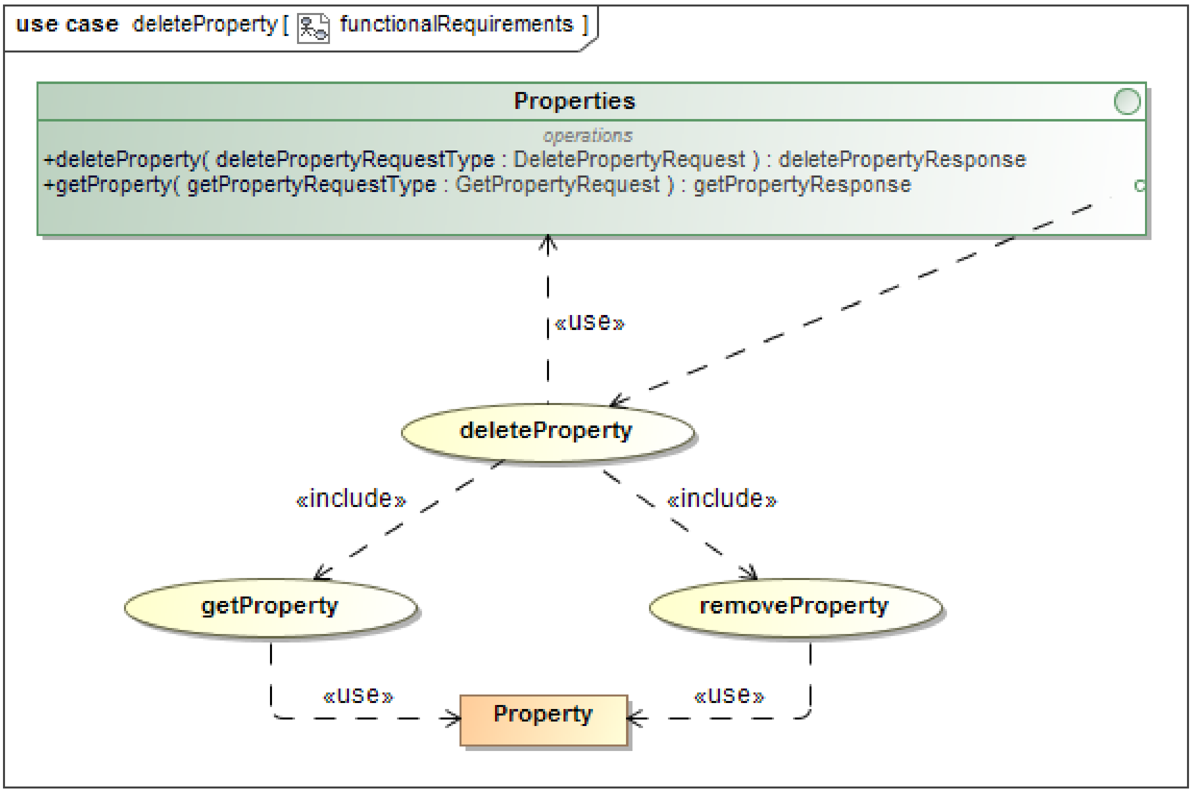
3.3.4 updateProfile


class updateProfile [updateProfile]
User
operations
+register(username : String, email : String, password : String, confrimPassword : String) : registerResponse
+login(email : String, password : String) : loginResponse()
+updateProfile(email : String, email : String, password : String, username : String) : addPropertyResponse
+logout() : void
...
«use»
updateProfile
«include»
«include»
«include»
verification
persistObject
getProfile
persistance

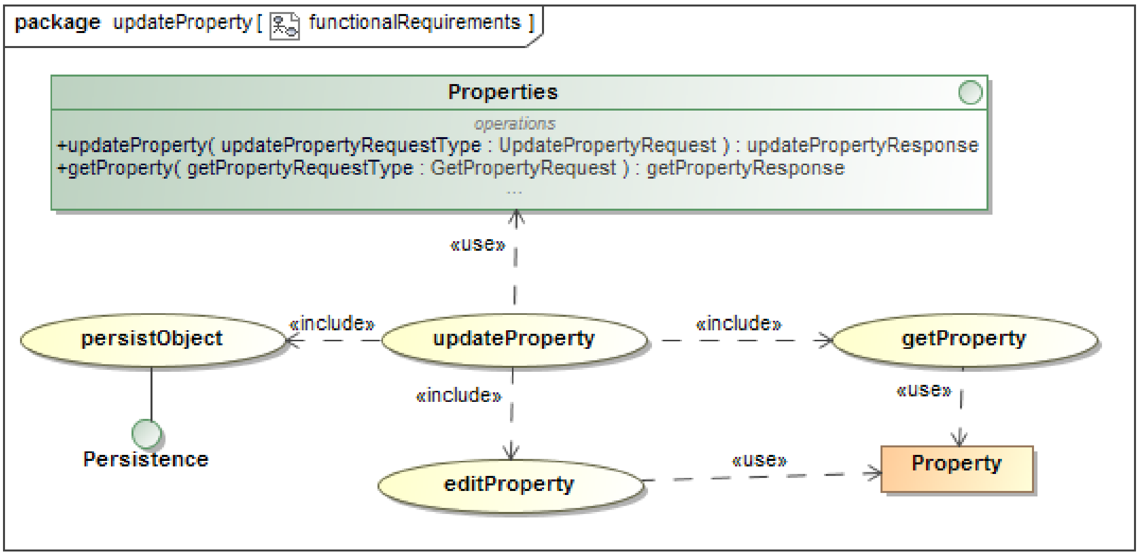
3.3.5 addProperty


use case addProperty [functionalRequirement]
Properties
operations
+addProperty(addPropertyRequestType : addProperty) : addPropertyResponse
+getProperty(getPropertyRequestType : GetPropertyRequest) : getPropertyResponse
+createProperty(createPropertyRequestType : CreatePropertyRequest) : createPropertyResponse
...
«use»
«include»
persistObject
addProperty
«include»
getProperty
«include»
«include»
«use»
Persistence
createProperty
addPropertyEntry
«use»
Property
Properties

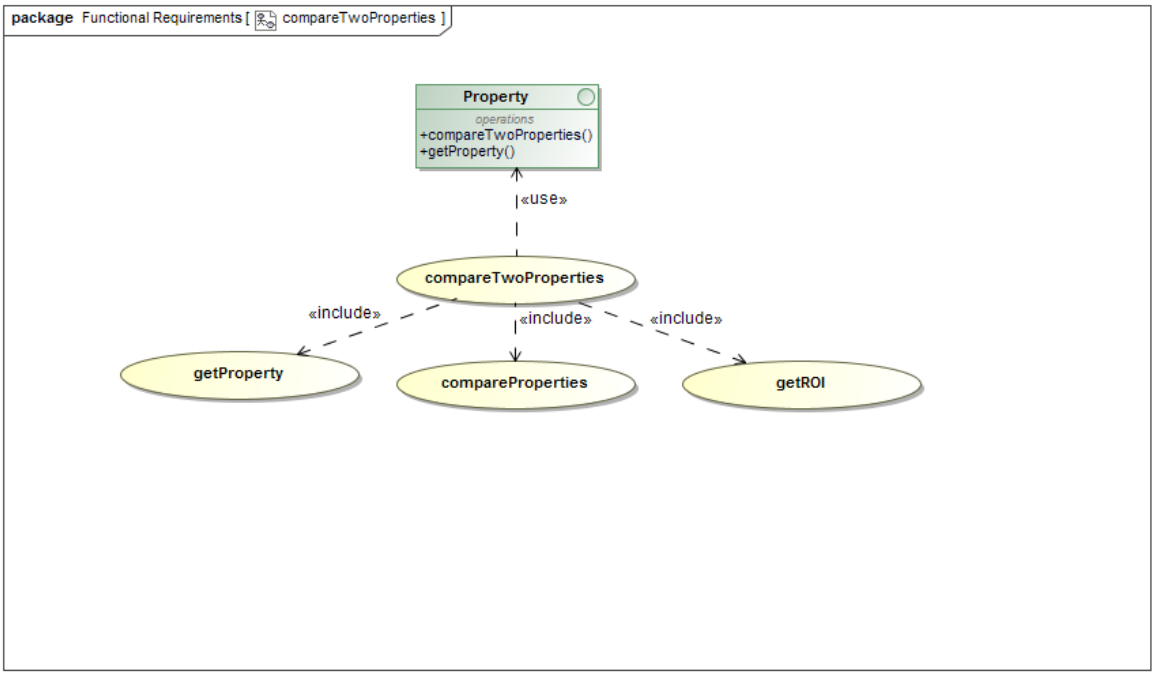
3.3.6 deleteProperty


use case deleteProperty [functionalRequirements]
Properties
operations
+deleteProperty(deletePropertyRequestType : DeletePropertyRequest) : deletePropertyResponse
+getProperty(getPropertyRequestType : GetPropertyRequest) : getPropertyResponse
«use»
deleteProperty
«include»
«include»
getProperty
removeProperty
«use»
Property
«use»

3.3.7 updateProperty

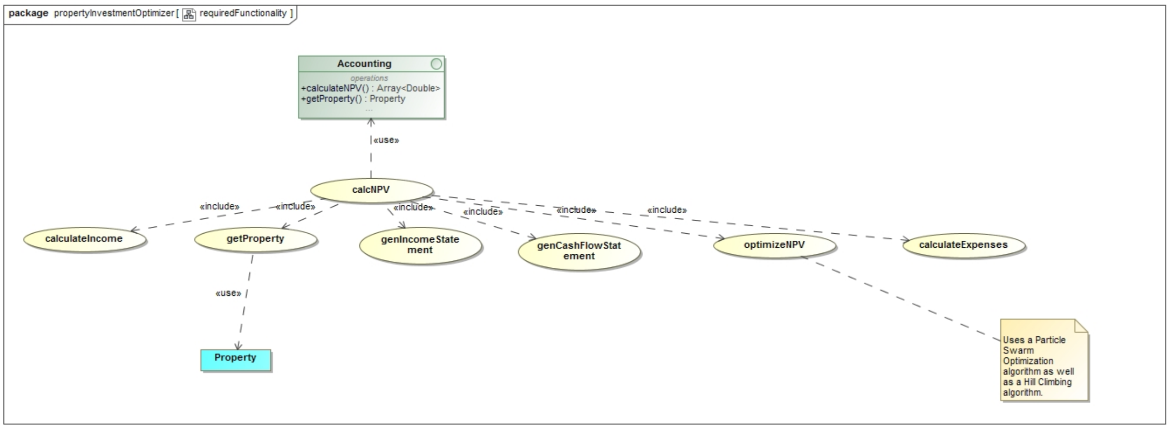

package updateProperty [functionalRequirements]
Properties
operations
+updateProperty(updatePropertyRequestType : UpdatePropertyRequest) : updatePropertyResponse
+getProperty(getPropertyRequestType : GetPropertyRequest) : getPropertyResponse
...
«use»
persistObject
«include»
updateProperty
«include»
getProperty
«include»
«use»
Persistence
editProperty
«use»
Property

3.3.8 compareTwoProperties

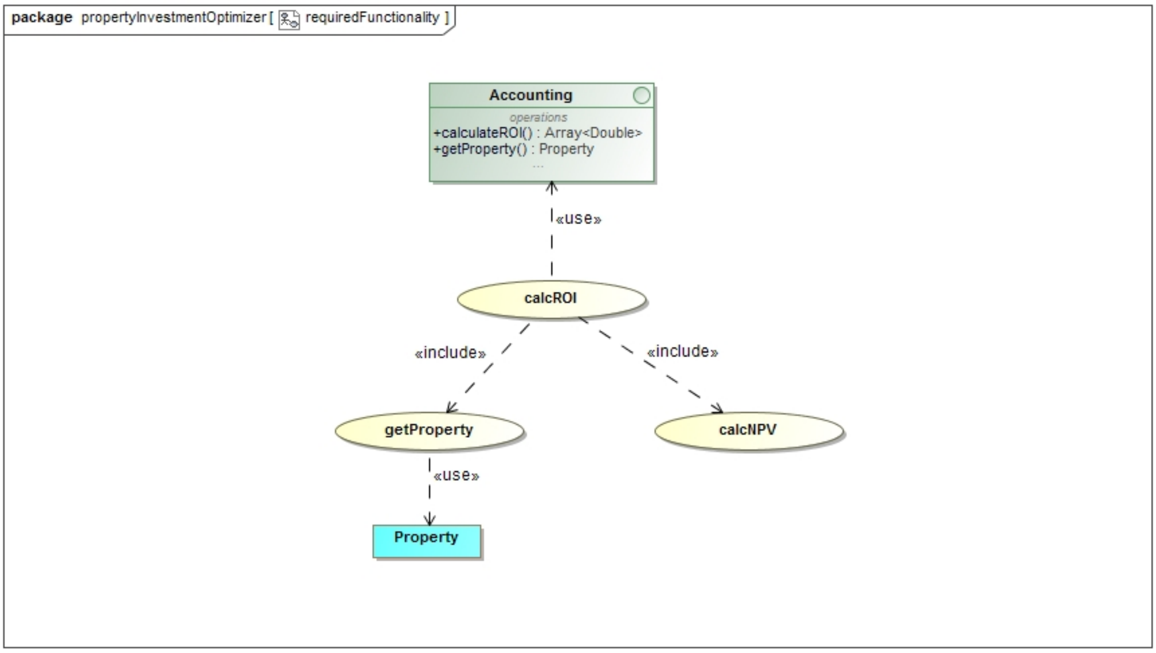


3.3.9 viewProperty

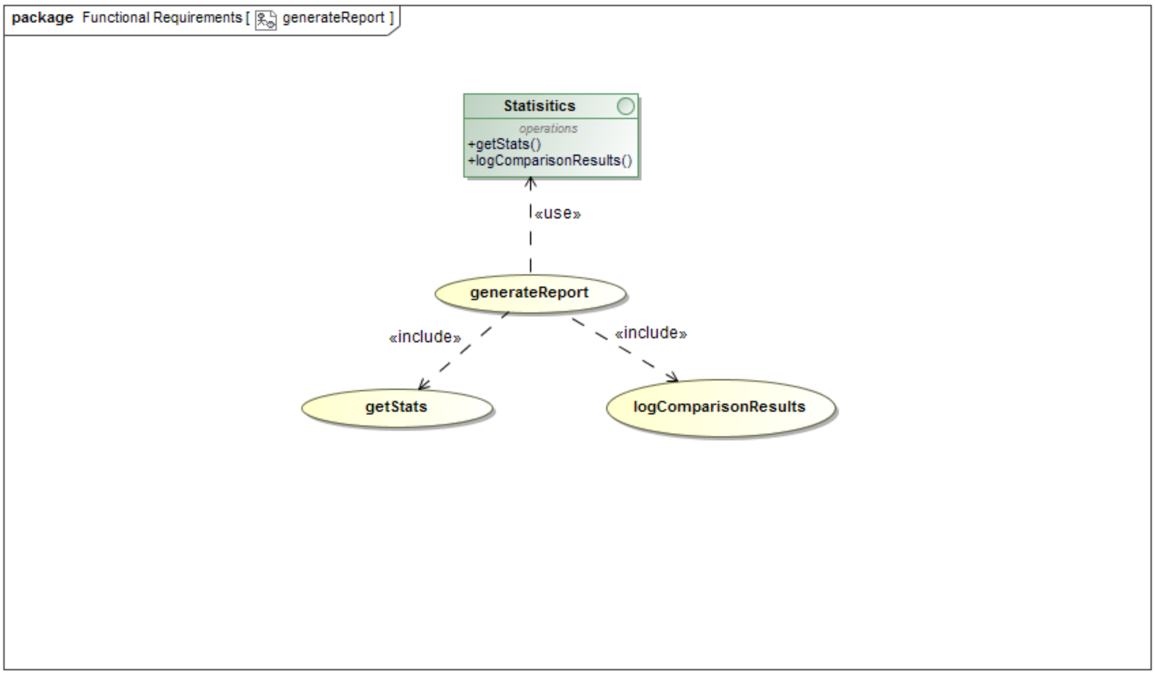
3.3.10 calucalateNPV


package propertyInvestmentOptimizer [requiredFunctionality]
Accounting
operations
+calculateNPV() : Array<Double>
+getProperty() : Property
...
«use»
calcNPV
«include»
«include»
«include»
«include»
«include»
«include»
calculateIncome
getProperty
genIncomeState ment
genCashFlowStat ement
optimizeNPV
calculateExpenses
«use»
Property
Uses a Particle Swarm Optimization algorithm as well as a Hill Climbing algorithm.

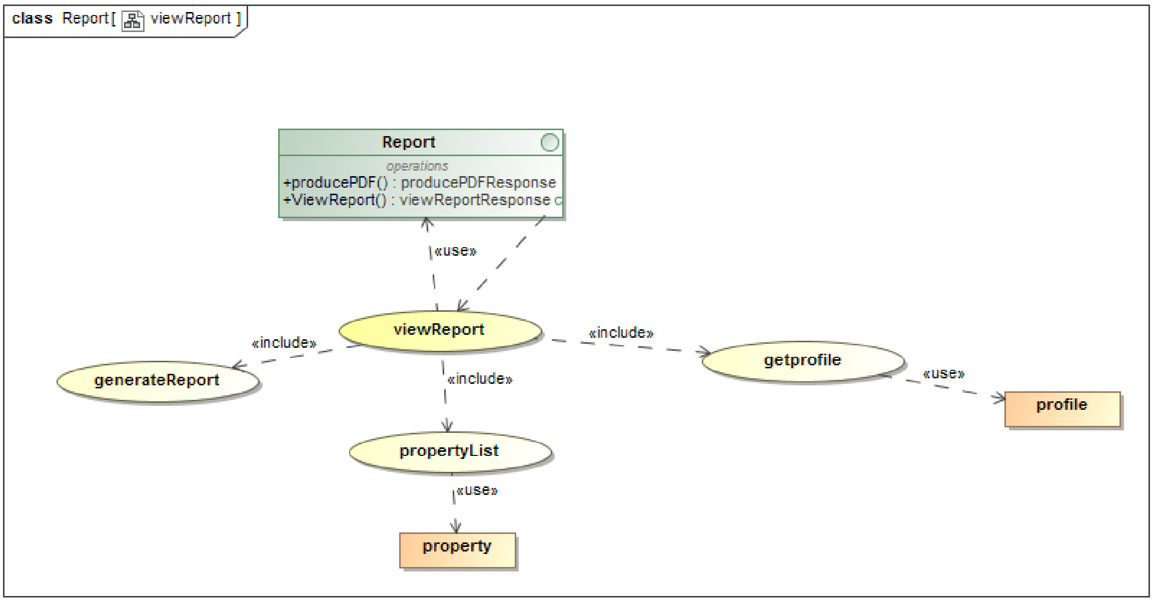
3.3.11 calucalateROI


package propertyInvestmentOptimizer [requiredFunctionality]
Accounting
operations
+calculateROI() : Array<Double>
+getProperty() : Property
...
«use»
calcROI
«include»
«include»
getProperty
calcNPV
«use»
Property

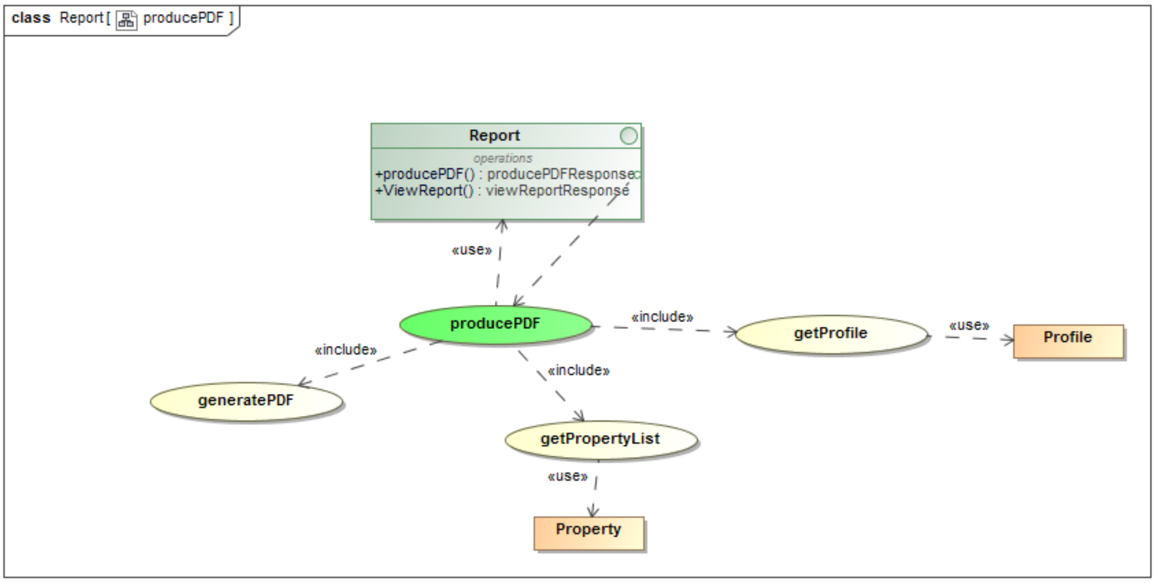
3.3.12 generateReport



3.3.13 viewReport

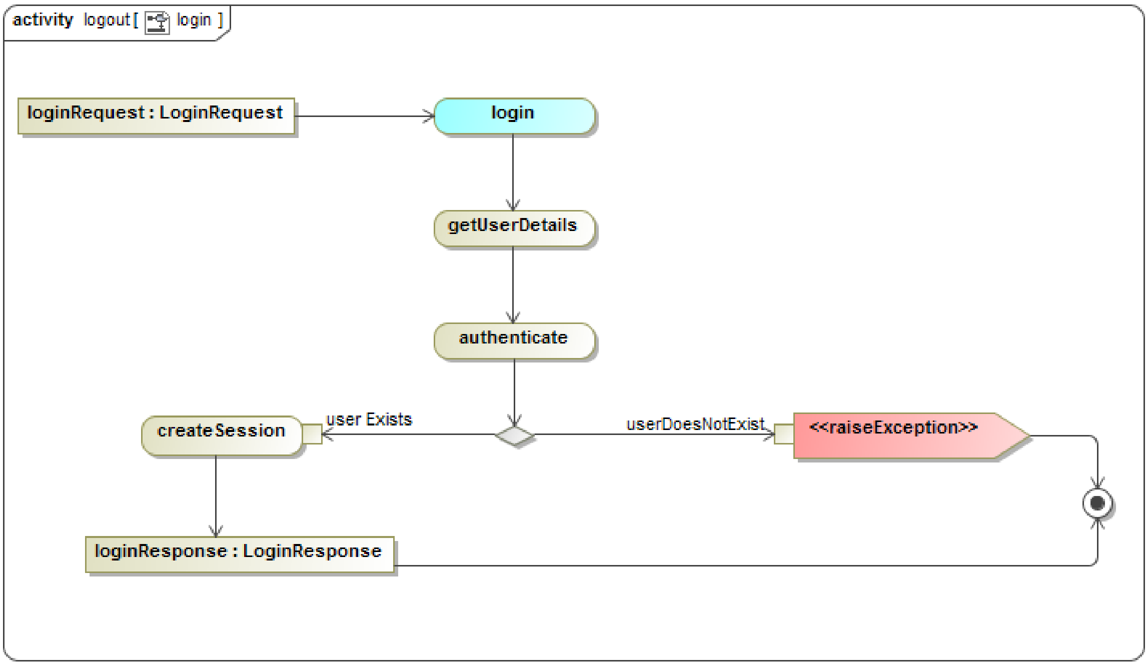


3.3.14 generatePDF

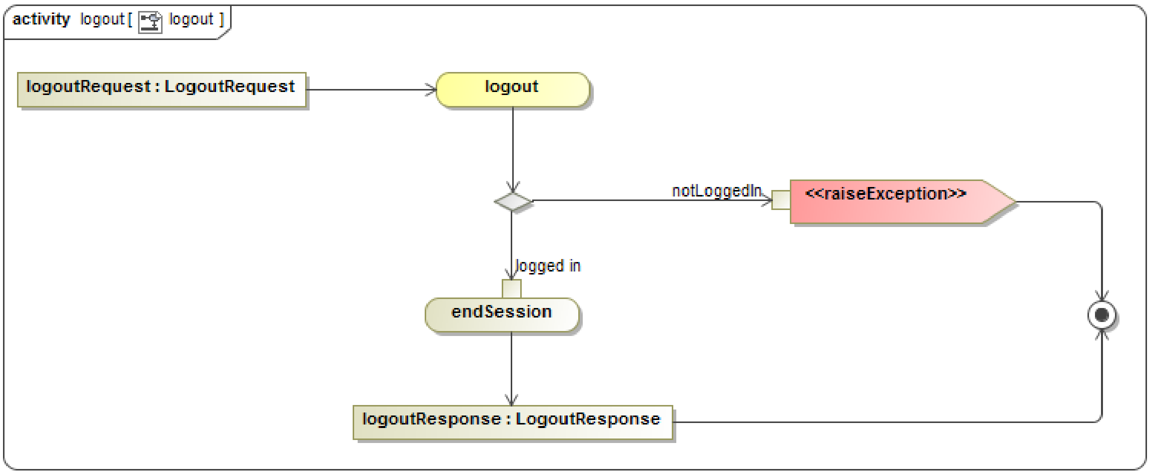


3.4 Process specifications

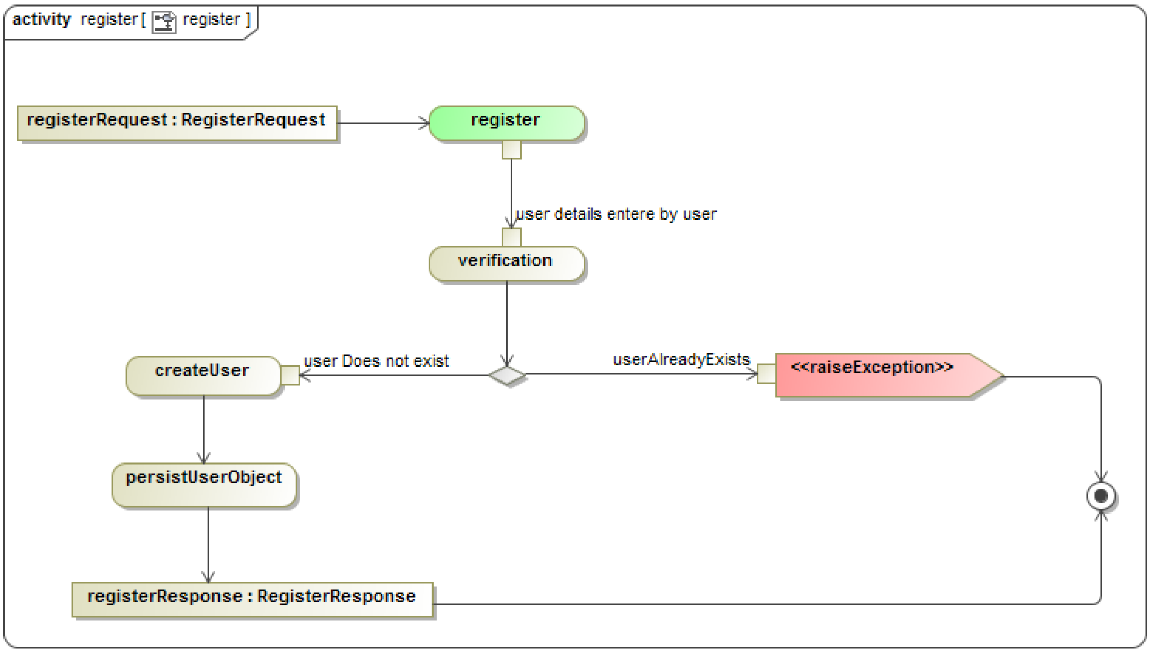
3.4.1 Login



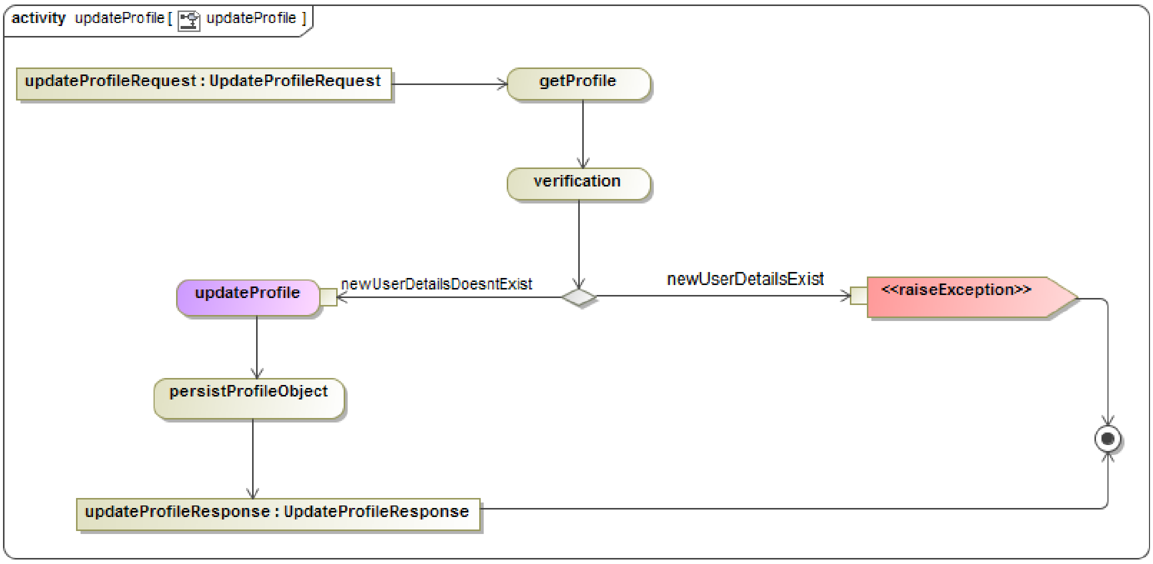
3.4.2 Logout


activity logout [logout]
logoutRequest : LogoutRequest
logout
notLoggedIn
<<raiseException>>
logged in
endSession
logoutResponse : LogoutResponse

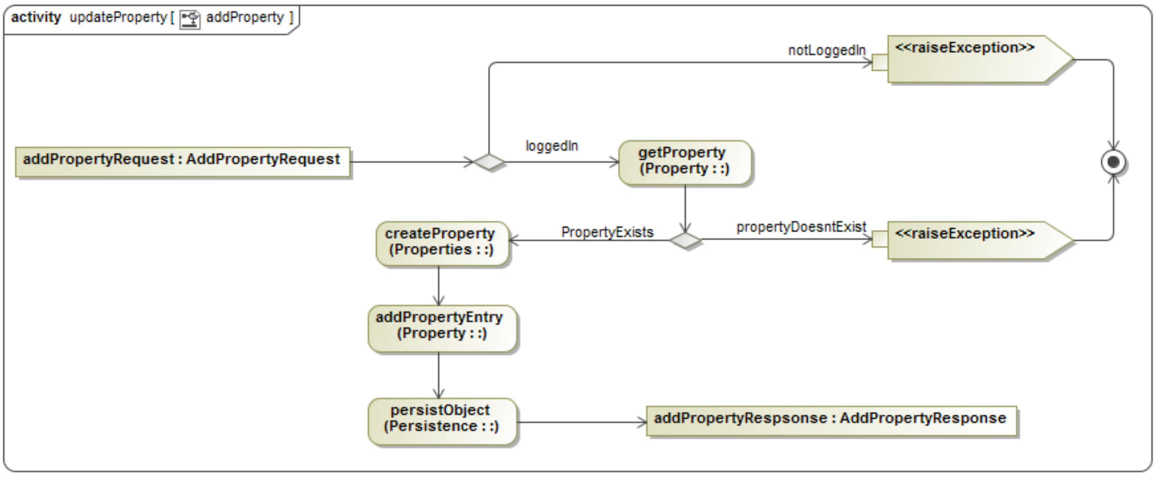
3.4.3 Register


activity register [register]
registerRequest : RegisterRequest
register
user details entere by user
verification
createUser
user Does not exist
userAlreadyExists
<<raiseException>>
persistUserObject
registerResponse : RegisterResponse

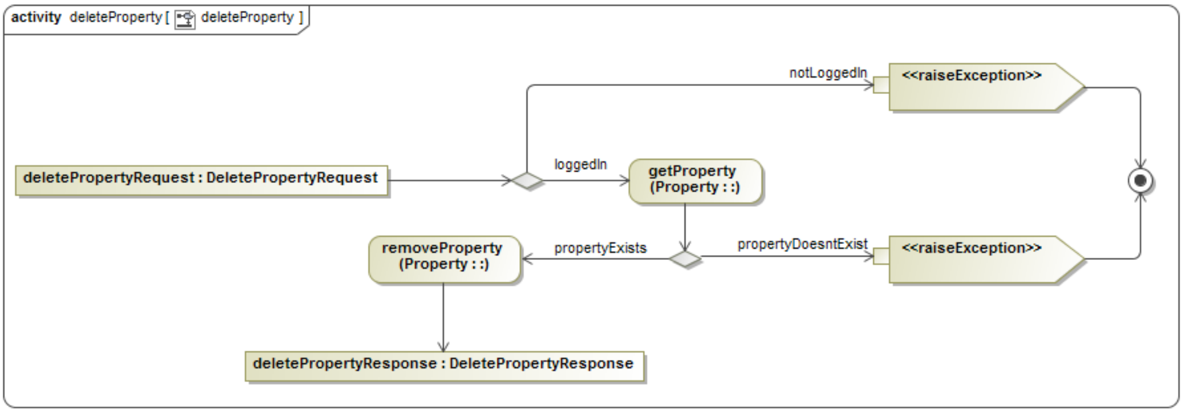
3.4.4 updateProfile


activity updateProfile [updateProfile]
updateProfileRequest : UpdateProfileRequest
getProfile
verification
updateProfile
newUserDetailsDoesntExist
newUserDetailsExist
<<raiseException>>
persistProfileObject
updateProfileResponse : UpdateProfileResponse

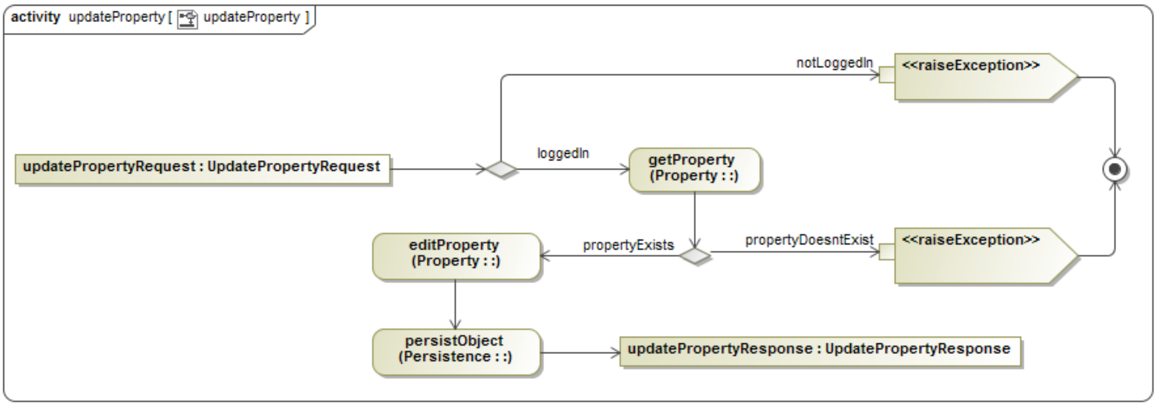
3.4.5 addProperty


activity updateProperty [addProperty]
notLoggedIn
<<raiseException>>
addPropertyRequest : AddPropertyRequest
loggedIn
getProperty (Property : :)
createProperty (Properties : :)
PropertyExists
propertyDoesntExist
<<raiseException>>
addPropertyEntry (Property : :)
persistObject (Persistence : :)
addPropertyRespsonse : AddPropertyResponse

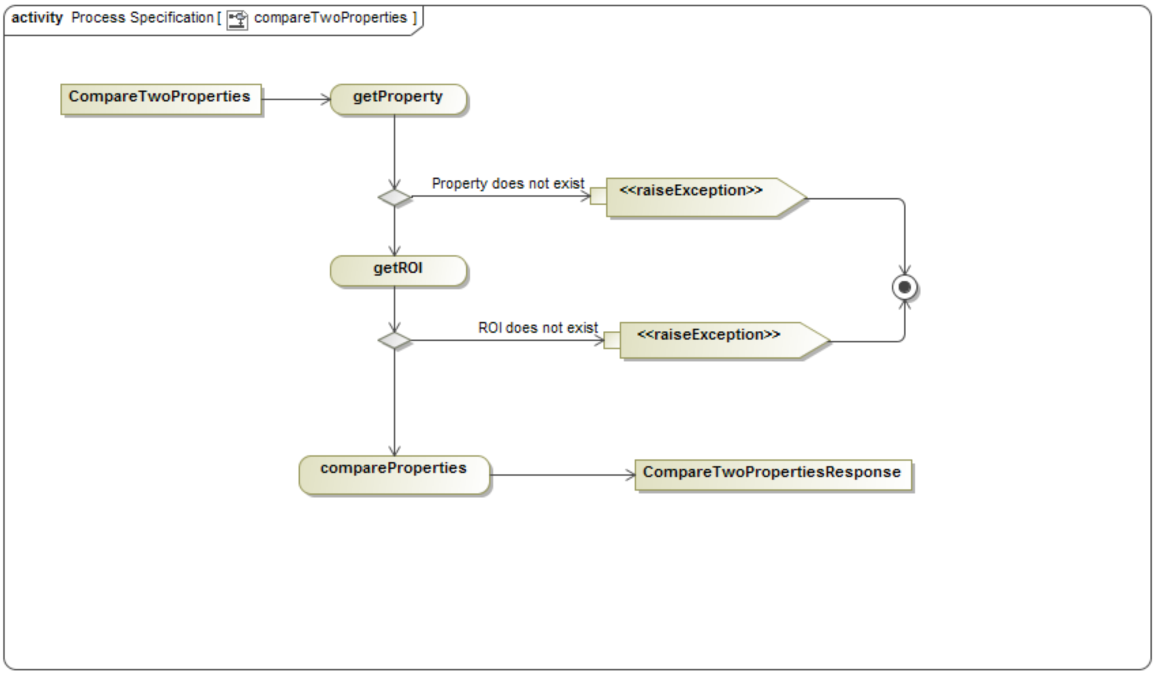
3.4.6 deleteProperty


activity deleteProperty [deleteProperty]
notLoggedIn
<<raiseException>>
deletePropertyRequest : DeletePropertyRequest
loggedIn
getProperty (Property : :)
removeProperty (Property : :)
propertyExists
propertyDoesntExist
<<raiseException>>
deletePropertyResponse : DeletePropertyResponse

3.4.7 updateProperty

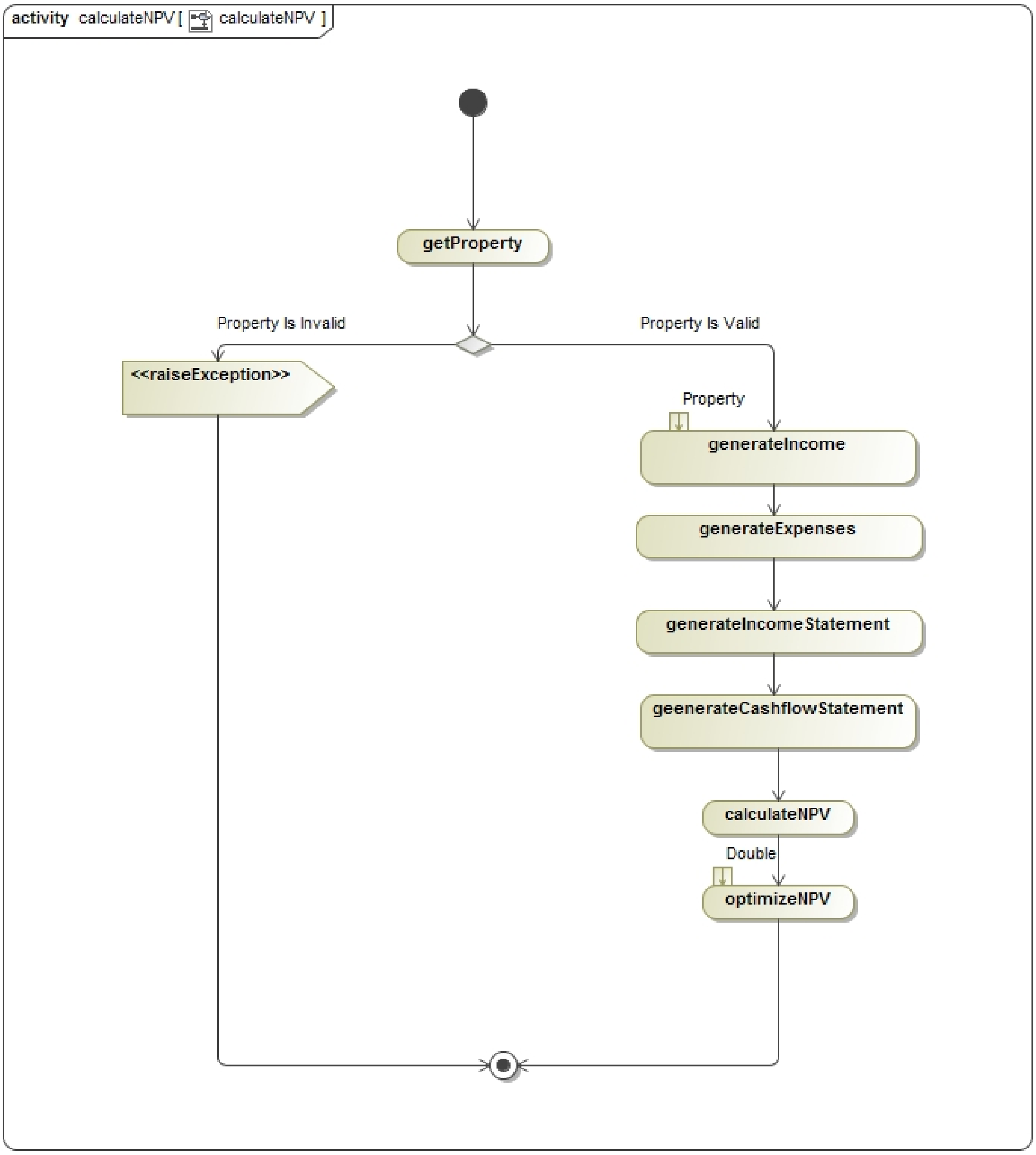

activity updateProperty [updateProperty]
notLoggedIn
<<raiseException>>
updatePropertyRequest : UpdatePropertyRequest
loggedIn
getProperty (Property : :)
editProperty (Property : :)
propertyExists
propertyDoesntExist
<<raiseException>>
persistObject (Persistence : :)
updatePropertyResponse : UpdatePropertyResponse

3.4.8 compareTwoProperties

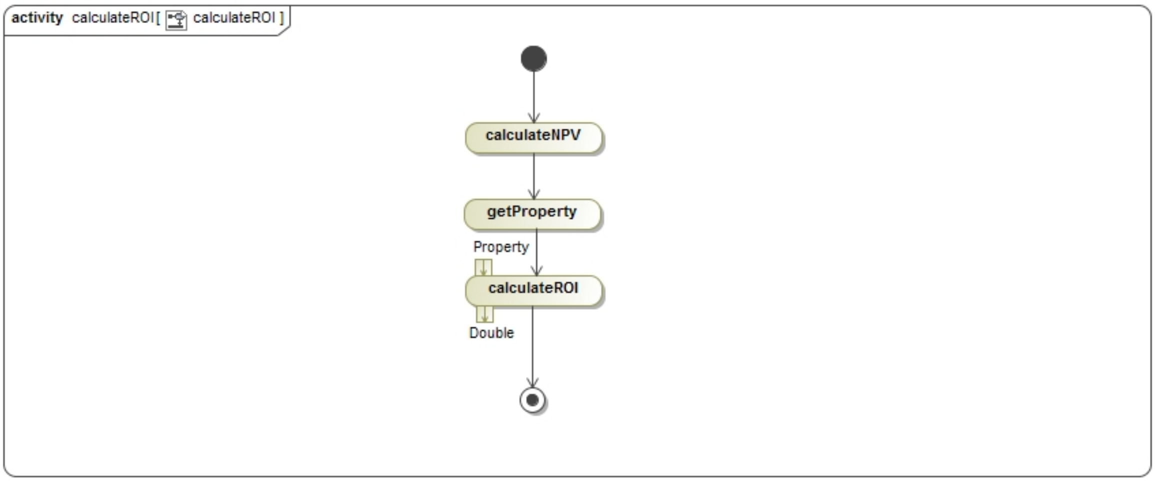


3.4.9 viewProperty

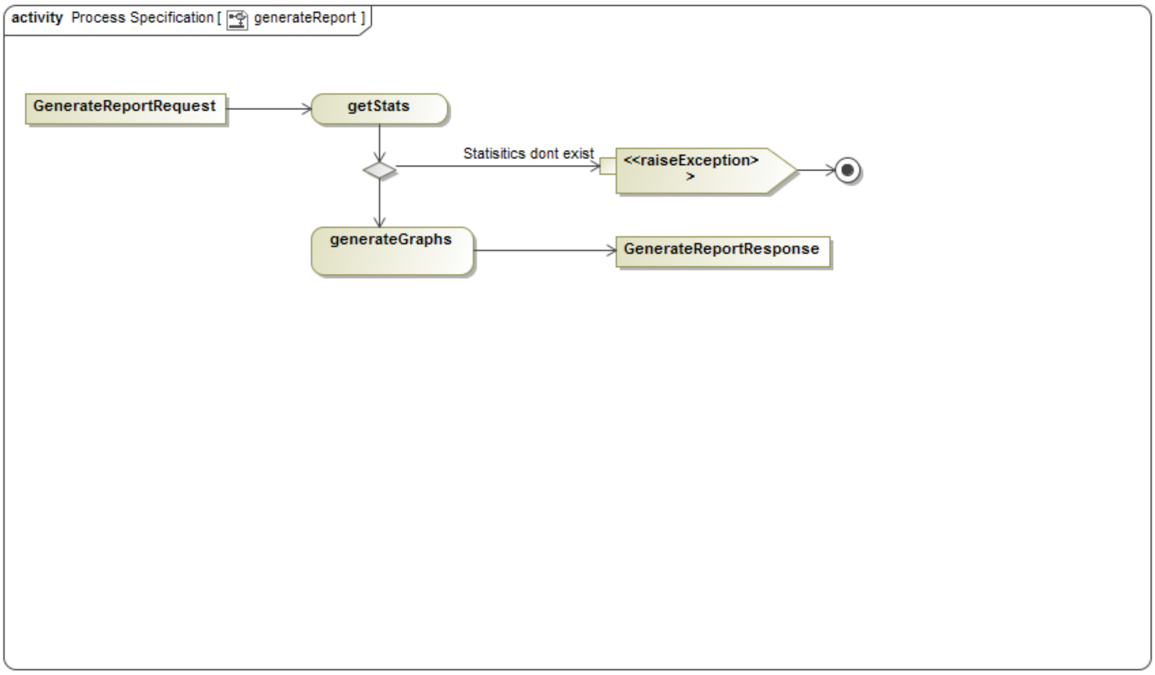
3.4.10 calucalateNPV



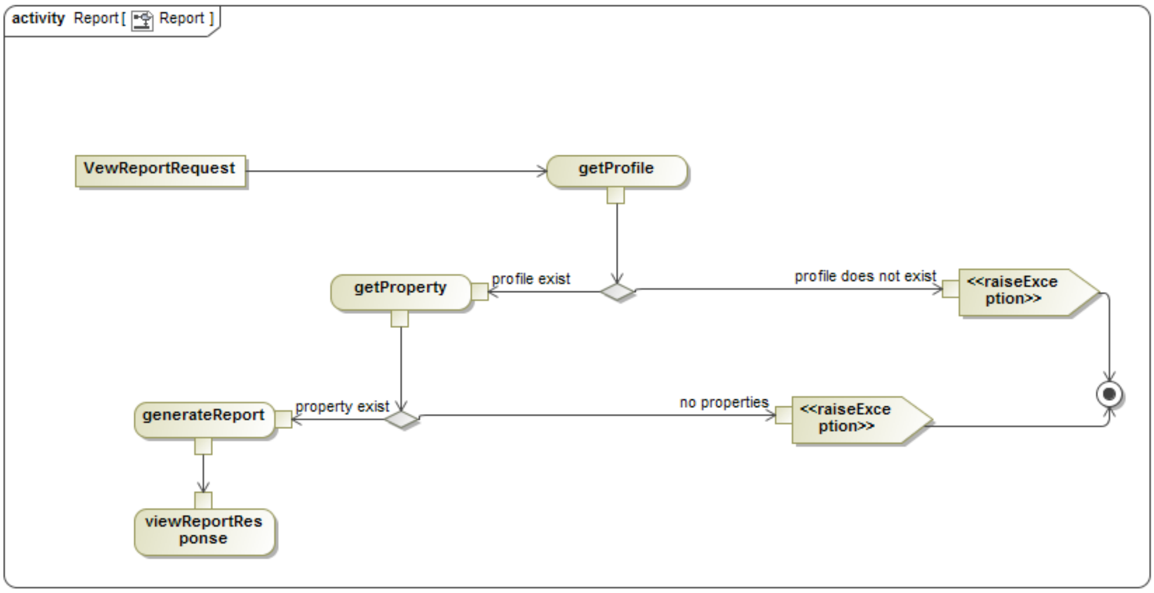
3.4.11 calucalateROI



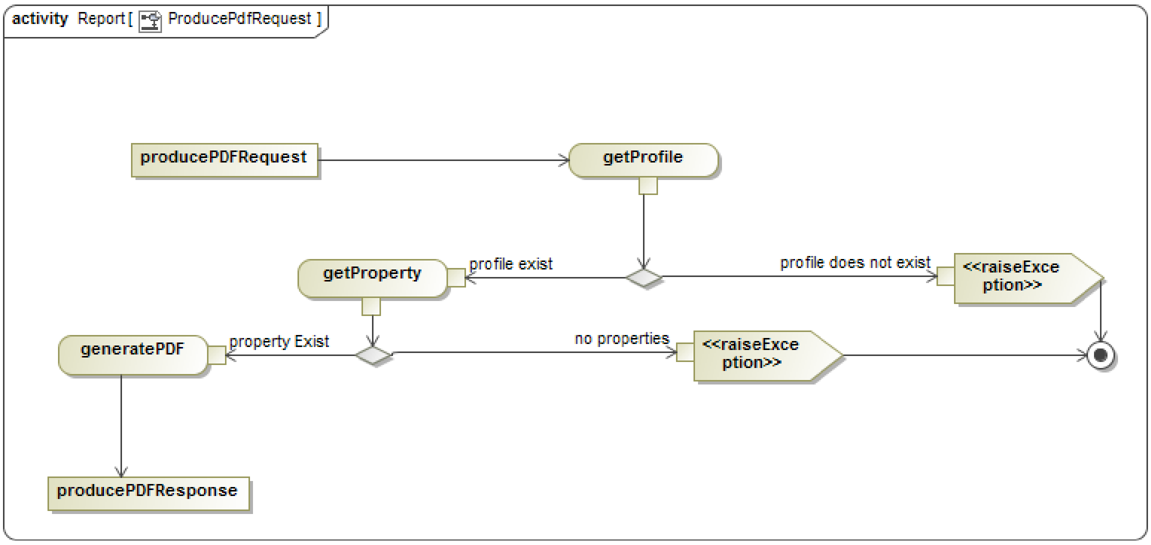
3.4.12 generateReport



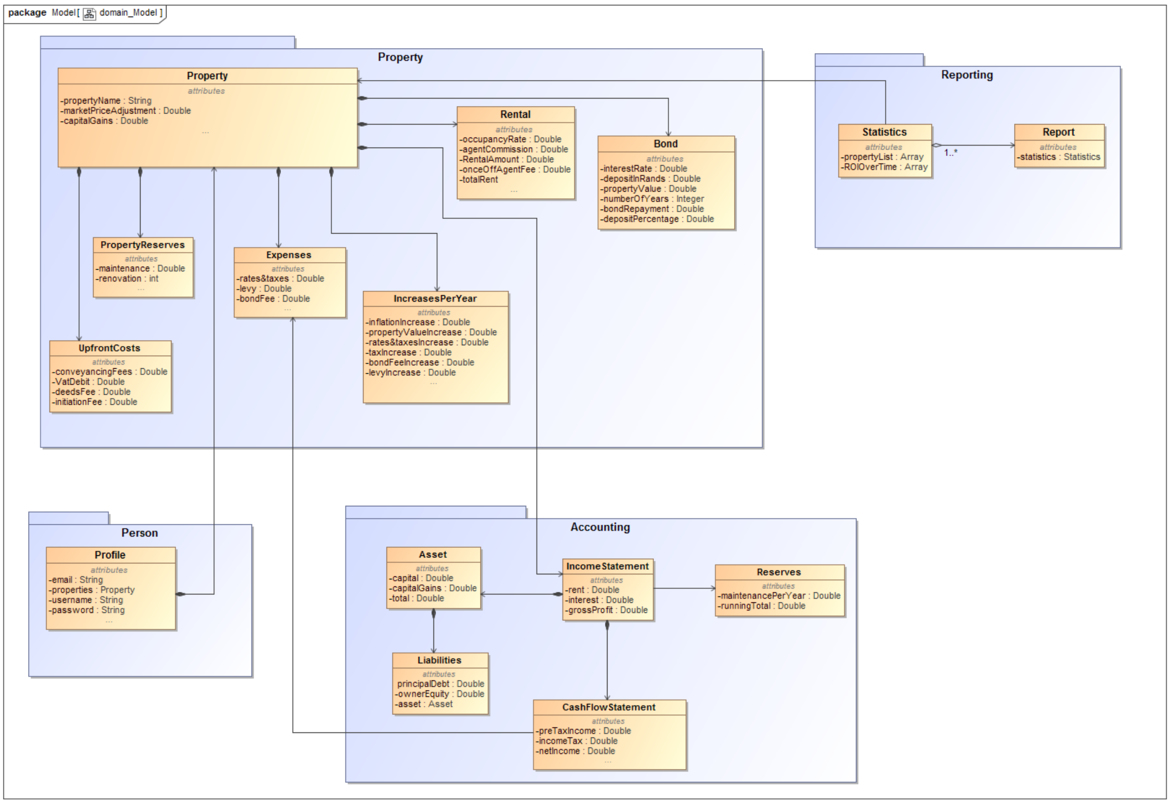
3.4.13 viewReport



3.4.14 generatePDF



3.5 Domain Model



4 Open Issues

- An appropriate Particle Swarm Optimization algorithm API and a Hill Climbing API is needed to optimize the Net Present Value calculation.
- We still need to get the EJB Container and JNDI lookup to work with our system.