CRUD

# Create

Flow of events:

5.1 **Use case prioritization**

This Use case is considered ***Important***

* 1. **Use case/Services contracts**

**Create Thread**

**The following use case applies to the creating of buzz space, posts and threads.**

Use case description:

1. **User** activates “Create Thread” function by selecting the “Create Thread” option.
2. **System** responds by presenting user with a form to create thread topic etc
3. **User** fills in form 🡪 topic of thread, tags, etc and submits the form
4. **System** reviews the submitted information and verifies users status and privileges, decorates the thread as needed and checks that all requirements (rules) are followed by the user based on his/her status
5. **System** displays either an acknowledgement or error message based on the pervious checkpoint.

Participants:

Initiated by User and communicates with system

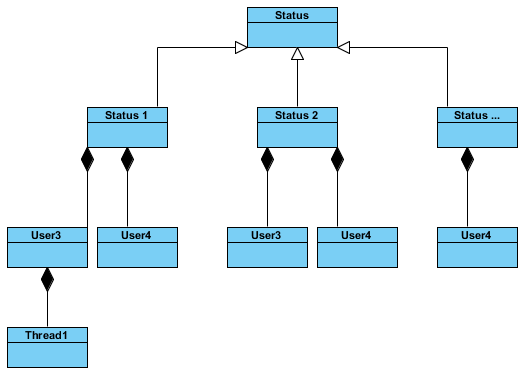
Pre-Conditions:

1. The User is logged in
   1. The user has a specific status
2. The user fills in the form to create a new thread
   1. User must have a specific status in order to perform certain functions when creating a thread

Post-Conditions:

1. System displays an acknowledgement message and create the new thread if user follows “status based rules”
2. Systems display an error message and requests user to restart if user attempts to perform a function that is not within the scope of their status privileges.

Request and Results Data Structures:



**2**

**1**

**3**

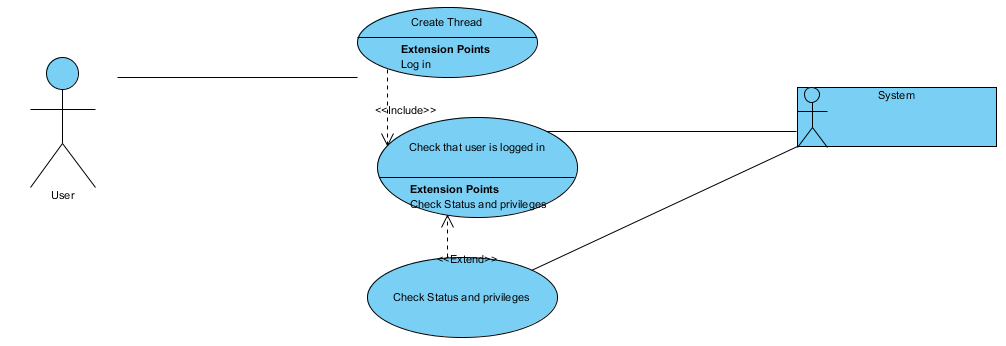
Status can be an abstract class and each sub-class will be a generalization of this class as each new status “IS-A” status.

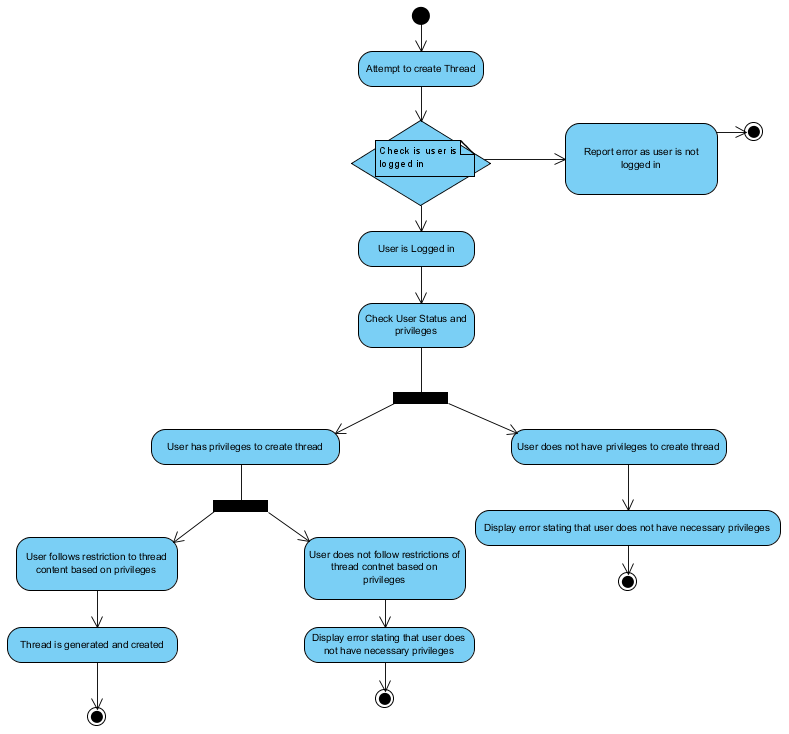
Each user “HAS-A” status and thus each new user will be a composition of each specific status class.

Each thread “HAS-A” user that started it and thus each new thread will be a composition of each user.

Steps:

1. If the user attempts to create a thread, the system will first check if the user is logged in.
   1. If so, it will go to number 1 above and check the users status
      1. The system verifies the status and at number 2 checks that the user has the privileges to create threads of the appropriate size, contents etc.
         1. If the user has the necessary privileges then the thread is created at 3
         2. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
2. If the user is not logged in, an error message is displayed telling the user that he/she is not logged in and it will redirect to the logging page.

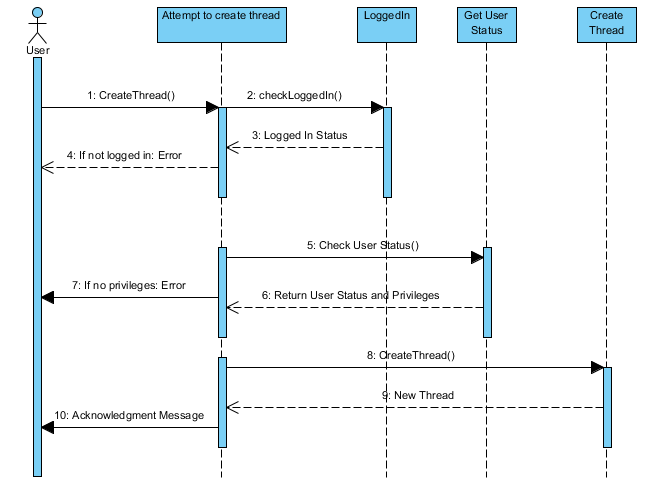
5.3 Required functionality – Use case diagram



5.4 Process specifications

1. Activity diagram

Sequence Diagram



# Read

Flow of events:

5.1 **Use case prioritization**

This Use case is considered ***Important***

* 1. **Use case/Services contracts**

**Read Thread**

**The following use case applies to the creating of buzz space, posts and threads.**

Use case description:

1. **User** activates “Read Thread” function by selecting the “A Thread” they wish to read.
2. **System** responds by presenting user with the selected thread of posts
3. **User** reads the posts and based on privileges can make changes such as update, delete, reply, tag, rate etc

Participants:

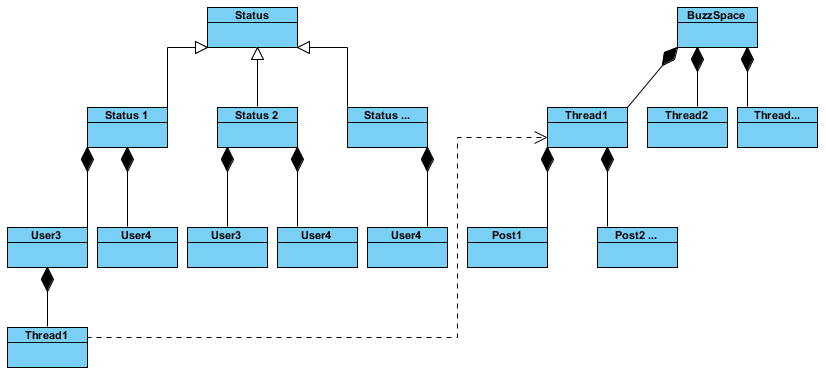
Initiated by User and communicates with system

Pre-Conditions:

1. The user must click on a thread they wish to read
   1. If User is logged in and
      1. User has status with high privileges then the user has options to change thread
         1. Calls other use cases
      2. Otherwise user is automatically logged in as “Guest” and can only view posts

Post-Conditions:

1. System displays selected thread with posts
2. If user is logged on and has privileges then an attempt to make changes such as update, delete, reply, tag, rate etc will be acknowledged.

Request and Results Data Structures:

**2**

**3**

**1**

Status can be an abstract class and each sub-class will be a generalization of this class as each new status “IS-A” status.

Each user “HAS-A” status and thus each new user will be a composition of each specific status class.

Each thread “HAS-A” user that started it and thus each new thread will be a composition of each user.

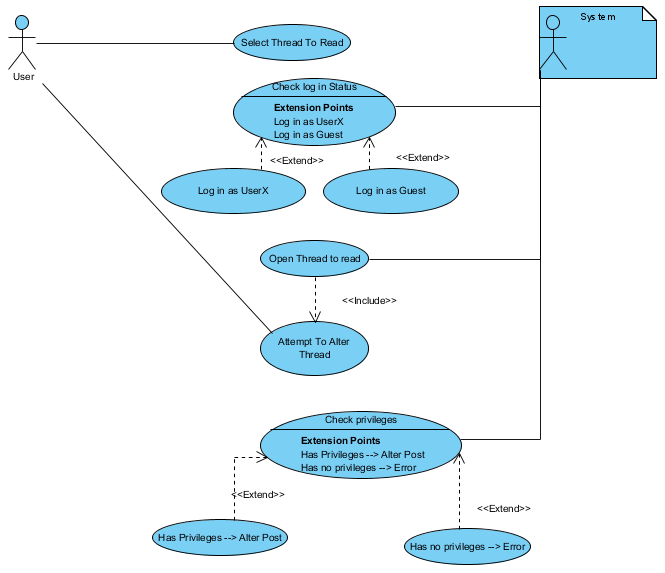
Each thread “HAS-A” BuzzSpace and thus each new thread will be a composition of each BuzzSpace.

Each Post “HAS-A” Thread and thus each new post will be a composition of each Thread

Steps:

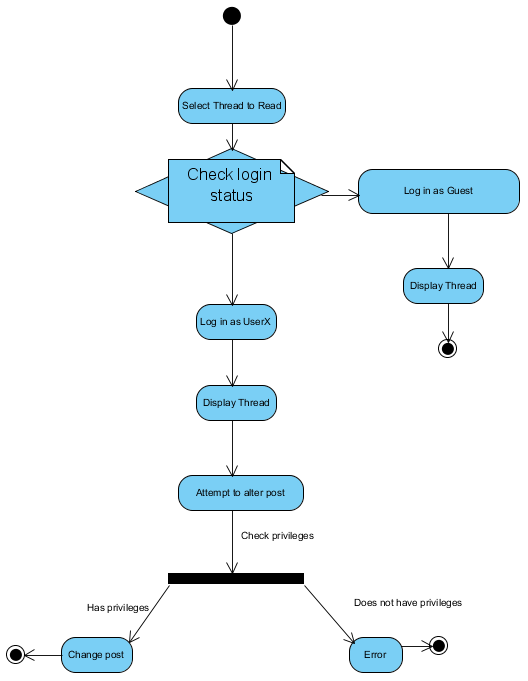
1. If the user attempts to enter a thread to read then the system will first check if the user is logged in.
   1. If so it will go to number 3 and open the selected thread to read.
      1. If user attempts to alter a post it will go to number 1 above and check the users status
         1. The system verifies the status and at number 2 checks that the user has the privileges to create threads of the appropriate size, contents etc.
         2. If the user has the necessary privileges then the post is altered at 3
         3. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
   2. If not the user will be automatically logged on as “Guest”

5.3 Required functionality – Use case diagram

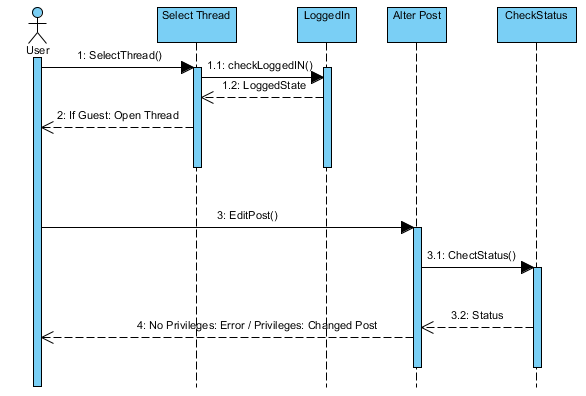


5.4 Process specifications

1. Activity diagram



Sequence Diagram



# Update

Flow of events:

5.1 **Use case prioritization**

This Use case is considered ***Important***

* 1. **Use case/Services contracts**

**Update Thread**

**The following use case applies to the creating of buzz space, posts and threads.**

Use case description:

1. **User** activates “Update Thread” function by selecting the “Update Thread” option by choosing how they wish to update the thread.
2. **System** checks to see whether user is logged in
3. **System** reviews the submitted information and verifies users status and privileges, decorates the thread as needed and checks that all requirements (rules) are followed by the user based on his/her status
4. **System** displays either an acknowledgement or error message based on the pervious checkpoint.

Participants:

Initiated by User and communicates with system

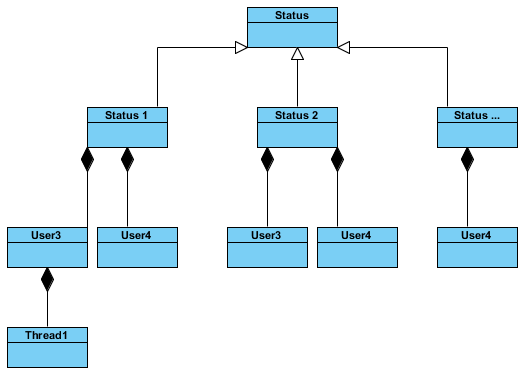
Pre-Conditions:

1. User chooses the way in which to update a thread
2. The User is logged in
   1. The user has a specific status
   2. User must have a specific status in order to perform certain functions when updating a thread

Post-Conditions:

1. System displays an acknowledgement message and updates the new thread if user follows “status based rules”
2. Systems display an error message and requests user to restart if user attempts to perform a function that is not within the scope of their status privileges.

Request and Results Data Structures:



**2**

**1**

**3**

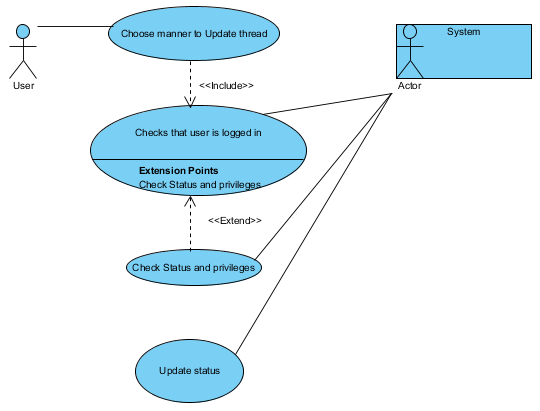
Status can be an abstract class and each sub-class will be a generalization of this class as each new status “IS-A” status.

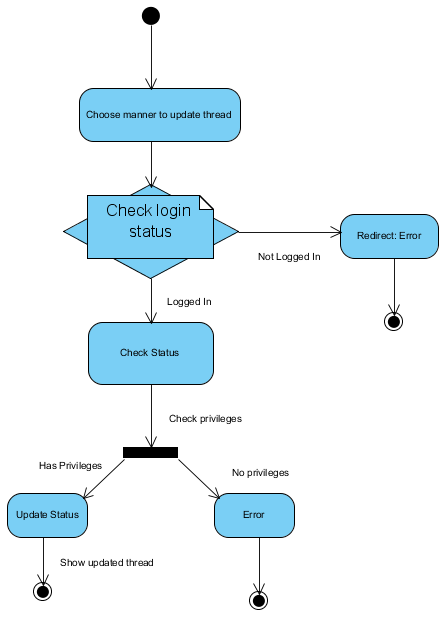
Each user “HAS-A” status and thus each new user will be a composition of each specific status class.

Each thread “HAS-A” user that started it and thus each new thread will be a composition of each user.

Steps:

1. If the user attempts to update a thread, the system will first check if the user is logged in.
   1. If so, it will go to number 1 above and check the users status
      1. The system verifies the status and at number 2 checks that the user has the privileges to update threads in the specific way chosen.
         1. If the user has the necessary privileges then the thread is updated at 3
         2. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
2. If the user is not logged in, an error message is displayed telling the user that he/she is not logged in and it will redirect to the logging page.

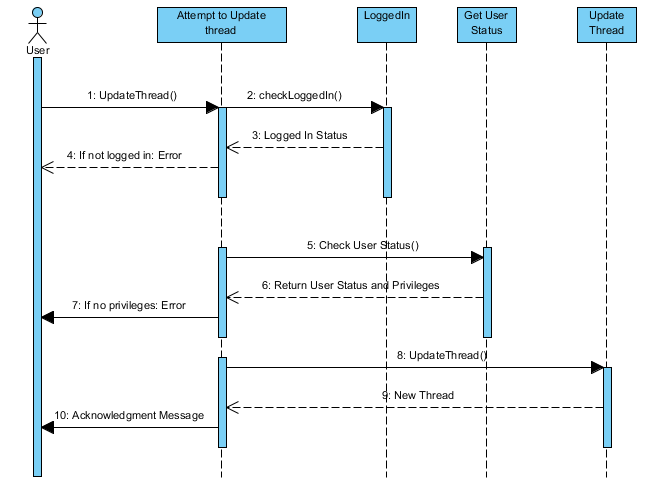
5.3 Required functionality – Use case diagram



5.4 Process specifications

1. Activity diagram

Sequence Diagram



# Delete

Flow of events:

5.1 **Use case prioritization**

This Use case is considered ***Important***

* 1. **Use case/Services contracts**

**Delete Thread**

**The following use case applies to the creating of buzz space, posts and threads.**

Use case description:

1. **User** activates “Delete Thread” function by selecting the “Delete Thread” option.
2. **System** checks to see whether user is logged in
3. **System** reviews the submitted information and verifies users status and privileges
4. **System** displays either an acknowledgement and removes the thread or displays an error message based on the pervious checkpoint.

Participants:

Initiated by User and communicates with system

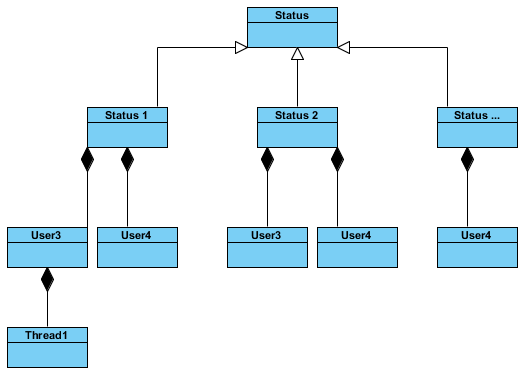
Pre-Conditions:

1. User selects “Delete Thread”
2. The User is logged in
   1. The user has a specific status
   2. User must have a specific status in order to delete a thread

Post-Conditions:

1. System displays an acknowledgement message and removes the thread if user follows “status based rules”
2. Systems display an error message and requests user to restart if user attempts to perform a function that is not within the scope of their status privileges.

Request and Results Data Structures:



**2**

**1**

**3**

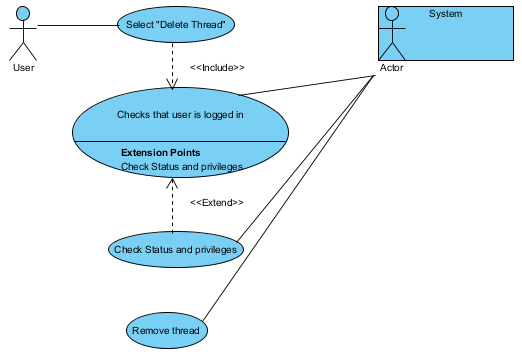
Status can be an abstract class and each sub-class will be a generalization of this class as each new status “IS-A” status.

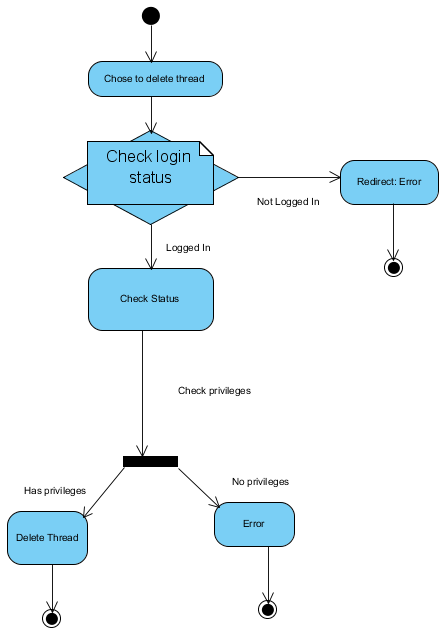
Each user “HAS-A” status and thus each new user will be a composition of each specific status class.

Each thread “HAS-A” user that started it and thus each new thread will be a composition of each user.

Steps:

1. If the user attempts to delete a thread, the system will first check if the user is logged in.
   1. If so, it will go to number 1 above and check the users status
      1. The system verifies the status and at number 2 checks that the user has the privileges to delete thread.
         1. If the user has the necessary privileges then the thread at 3 is removed from the tree
         2. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
2. If the user is not logged in, an error message is displayed telling the user that he/she is not logged in and it will redirect to the logging page.

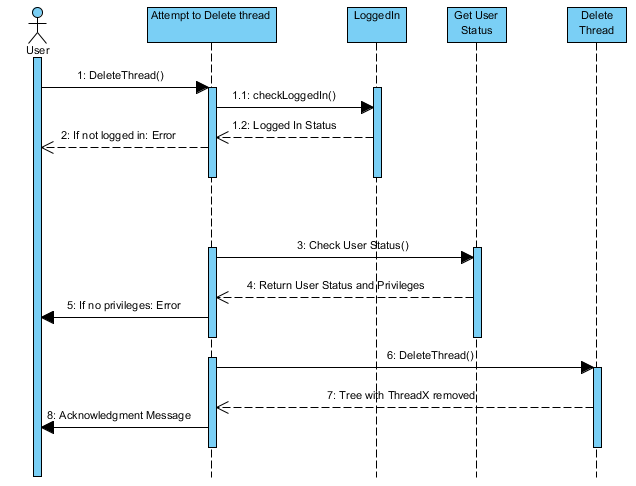
5.3 Required functionality – Use case diagram



5.4 Process specifications

1. Activity diagram

Sequence Diagram



# LogOut

Flow of events:

5.1 **Use case prioritization**

This Use case is considered ***Important***

**5.2 Use case/Services contracts**

**Logout of system**

Use case description:

1. **User** activates “Logout” function by selecting the “Logout” option.
2. **System** responds by presenting user with standard Guest interface with no profile or privileges

Participants:

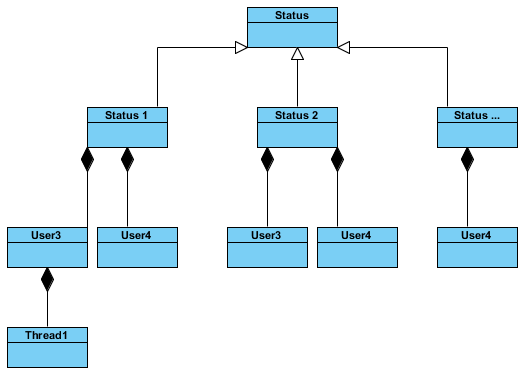
Initiated by User and communicates with system

Pre-Conditions:

1. The user must click on “Logout” function

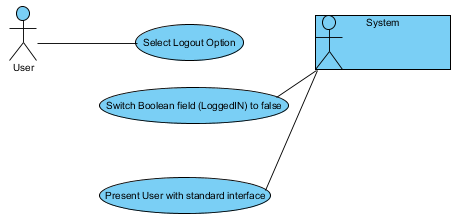
Post-Conditions:

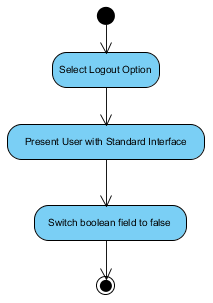
1. System displays user with standard guest interface
2. User has no privileges and status and cannot perform any function that a logged in user can

Request and Results Data Structures:

Steps:

1. User selects the logout option
   1. Boolean field in the specific user class – e.g User3 will be switched to false
2. User will be presented with Guest interface

5.3 Required functionality – Use case diagram

5.4 Process specifications

1. Activity diagram

Sequence Diagram

