Functional Requirements

Group Name: Group 6_b

Jessica (JI) Lessev	13049136
Thabang (TM) Letageng	13057937
Michelle Swanepoel	13066294
Prenolan Govender	13102380
Fako (FJA) Peleha	12230830
Lutfiyya Razak	10198408
Ephiphania Munava	10624610
Maria Qumalo	29461775

Git repository link:
`https://github.com/u12230830/COS301_6b`

Date: 27 February 2015

Contents

The following modules will be discussed, each with their own set of use cases:

- BuzzSpace Handling
- Thread Handling
- Posts Handling
- Filtering
- Authorisation
- Reporting
- Profile Handling

BuzzSpace Handling
Use Cases

1 Create BuzzSpace

Description:

This use case encapsulates the ability for a user to create a new BuzzSpace that will be associated with a specific acitve module. Each BuzzSpace will have a root thread to start with. The creator of the BuzzSpace will set properties of the BuzzSpace (i.e which status is needed for a user to do what).

1.1 Prioritization:

Critical

1.2 Conditions and Data Structures:

Pre-Conditions:

- The user wanting to create a new BuzzSpace should be logged in to the system and should be registered with the module for which he/she wants to create the BuzzSapce.
- The user should have a high enough status to be allowed to create a BuzzSpace (i.e. lecturer status).
- The module for which the BuzzSpace is being created must be an active module at that time.

Post-Conditions:

- A message is displayed to inform the creator of the BuzzSpace that the creation has been successful. Users can now start more threads on this BuzzSpace.
- If this service has been rejected, an exceoption is thrown and an error message will be displayed to the user.

Requests and Results Data Structures:

1.3 Required Functionality:

1.4 Process Specifications:

2 Read BuzzSpace

Description:

This use case includes the ability to view BuzzSpaces, this is, to view the different threads in the BuzzSpace (and thus have acces to them), and also to view the properties of the specific BuzzSpace (i.e. the permissions that were set, date of creation and the creator of this thread).

2.1 Prioritization:

Critical

2.2 Conditions and Data Structures:

Pre-Conditions:

None

Post-Conditions:

- Threads can now be viewed and accessed, and the properties can also be viewed.

Requests and Results Data Structures:

2.3 Required Functionality:

2.4 Process Specifications:

3 Update BuzzSpace

Description:

This use case gives the ability for a user with a high enough status (lecturer status) to change the permissions needed for certain actions of this BuzzSpace.

3.1 Prioritization:

Important

3.2 Conditions and Data Structures:

Pre-Conditions:

- The user must be logged in and registered for the module that the BuzzSpace is in.
- The user must have lecturer status to be able to update threads.

Post-Conditions:

- A message is displayed to let the user knnow that the update has been done successfully
- If the service was rejected, an exception is thrown and an error message is displayed to let the user know.

Requests and Results Data Structures:

3.3 Required Functionality:

3.4 Process Specifications:

4 Delete BuzzSpace

Description:

This use case includes the ability to delete a BuzzSpace when the module is not active anymore, or the BuzzSpace is not used anymore. Only the creator of the BuzzSpace will be able to delete a BuzzSpace, except if he changed the property to let anyone of a certain status be abe to delete it.

4.1 Prioritization:

Critical

4.2 Conditions and Data Structures:

Pre-Conditions:

- The user must be logged in
- The user must be the creator of the thread or must have a high enough status (if the permission for this was customized).

Post-Conditions:

- The BuzzSpace is deleted and archived for future reference.

Requests and Results Data Structures:

4.3 Required Functionality:

4.4 Process Specifications:

Thread Handling
Use Cases

1 Create Thread

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.

1.1 Prioritization:

This Use case is considered Important

1.2 Conditions and Data Structures:

Participants: Initiated by User and communicates with system

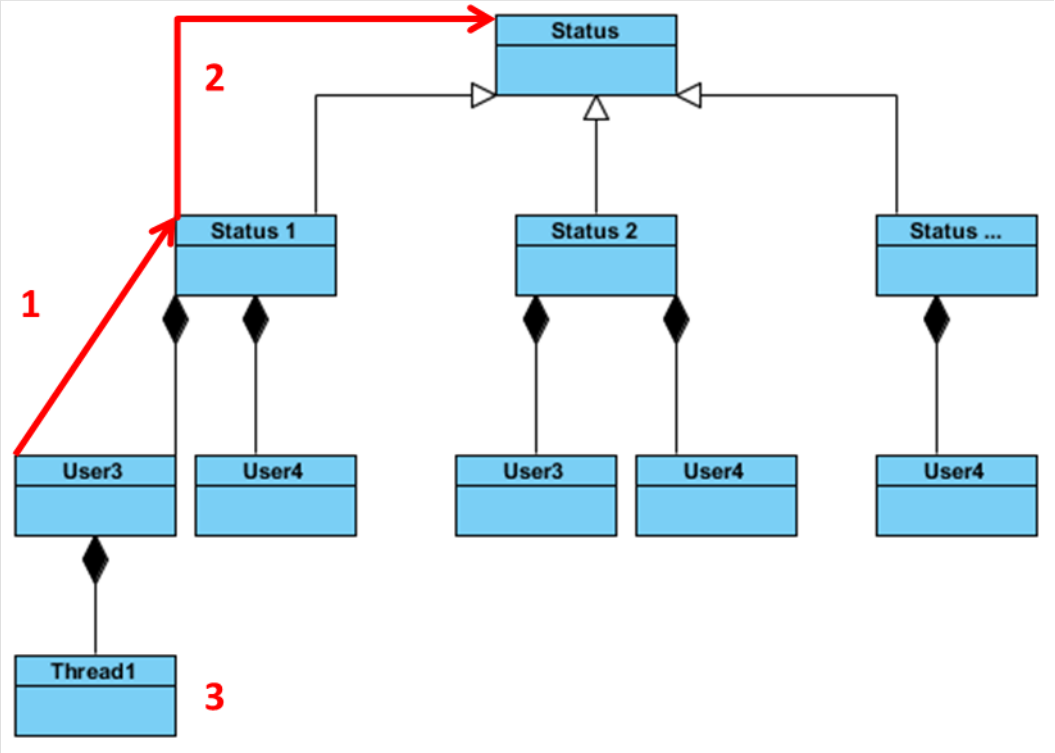
Pre-Conditions:

1. The User is logged in
a. The user has a specific status

Post-Conditions:

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

Requests and Results Data Structures:

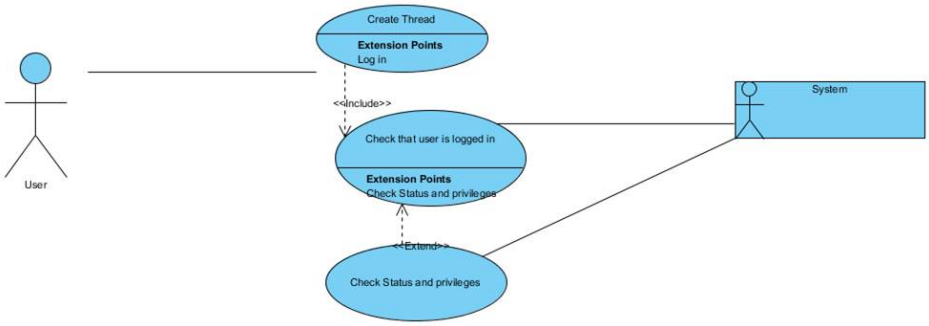


Status can be an abstract class and each sub-class will be a generalization of this class as each new status IS-A status. Each user HAS-A status and thus each new user will be a composition of each specific status class. Each thread HAS-A user that started it and thus each new thread will be a composition of each user.

Steps:

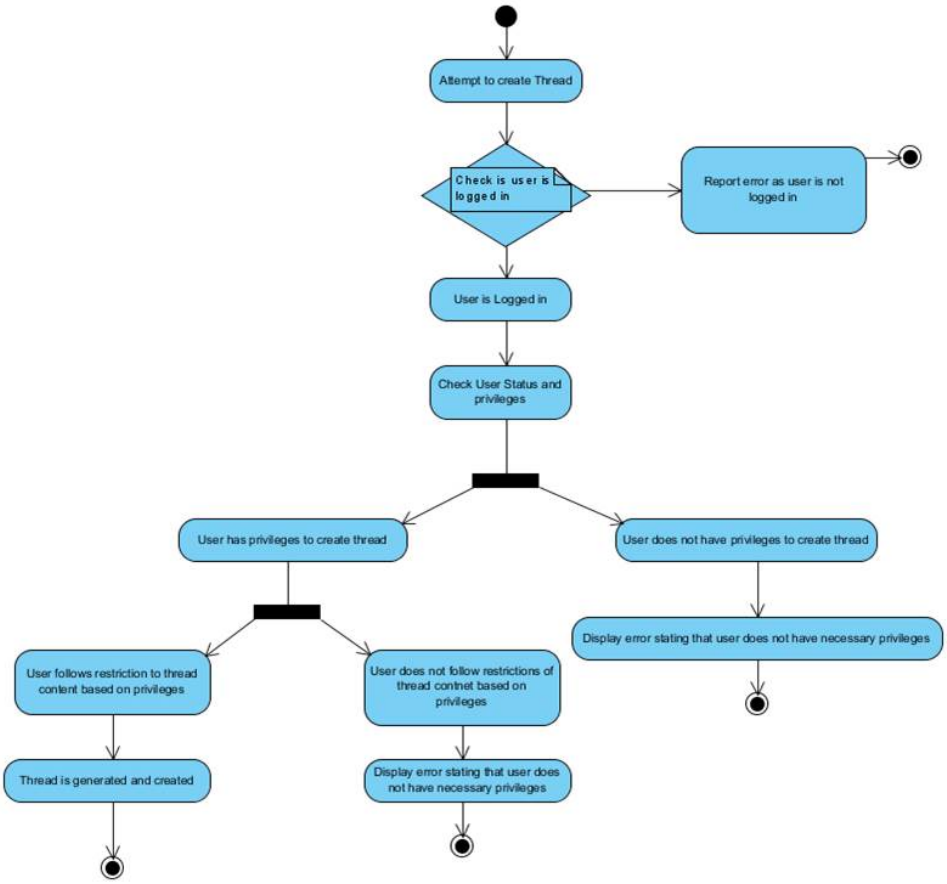
1. If the user attempts to create a thread, the system will first check if the user is logged in.
1.1. If so, it will go to number 1 above and check the users status
1.1.1. The system verifies the status and at number 2 checks that the user has the privileges to create threads of the appropriate size, contents etc.
1.1.1.1 If the user has the necessary privileges then the thread is created at 3
1.1.1.2 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.

1.3 Required Functionality:

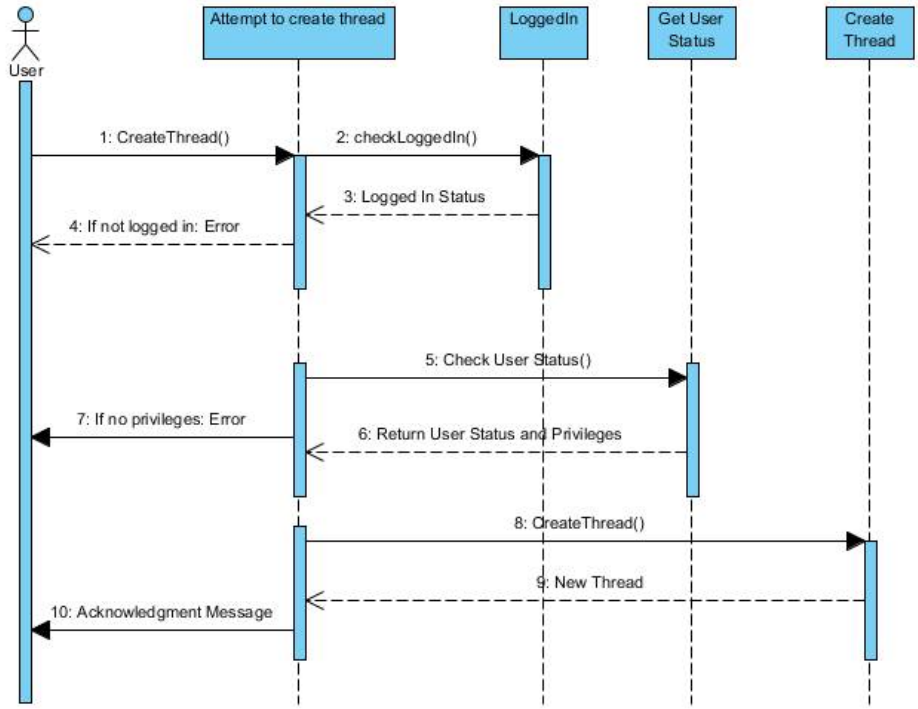


1.4 Process Specifications:

Activity Diagram



Sequence Diagram



2 Read Thread

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

2.1 Prioritization:

This Use case is considered Important

2.2 Conditions and Data Structures:

Participants:
Initiated by User and communicates with system

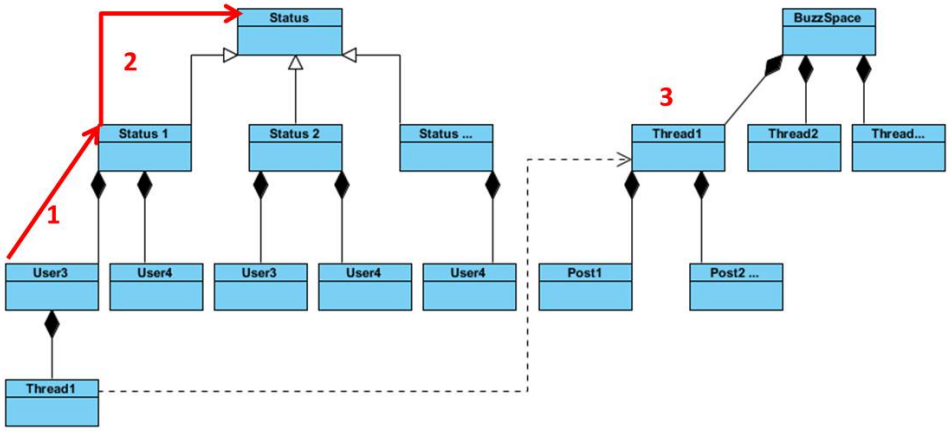
Pre-Conditions:

1. The user must click on a thread they wish to read
1.1. If User is logged in and
1.1.1. User has status with high privileges then the user has options to change thread
1.1.1.1. Calls other use cases
1.1.2. 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.

Requests and Results Data Structures:

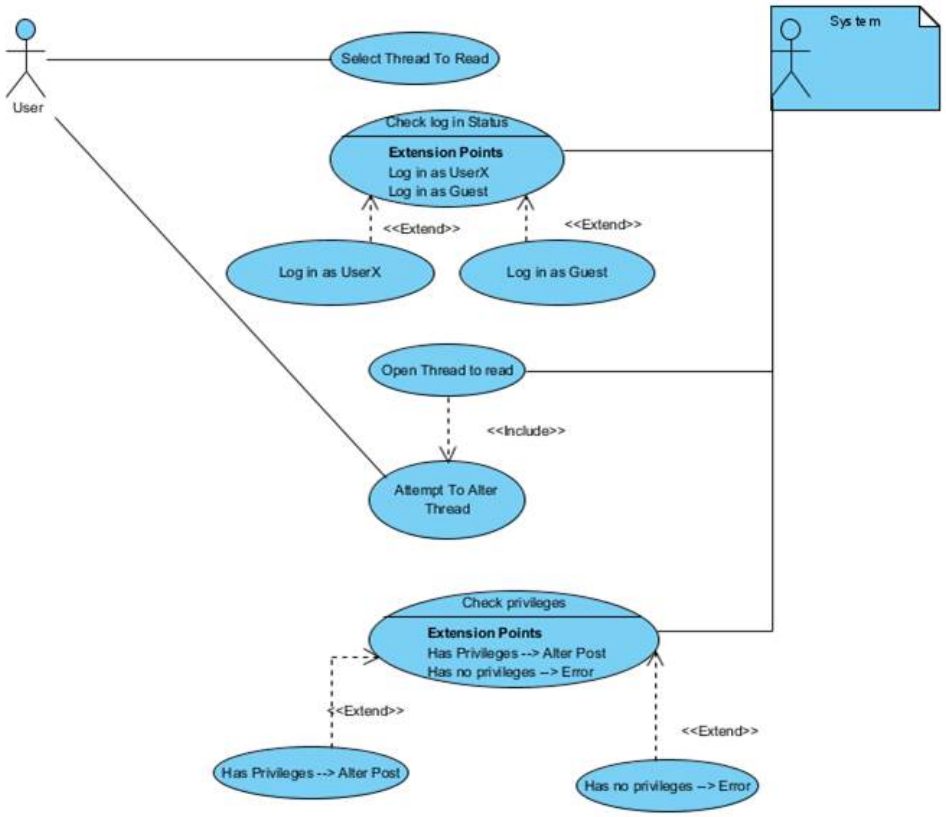


Status can be an abstract class and each sub-class will be a generalization of this class as each new status IS-A status.
Each user HAS-A status and thus each new user will be a composition of each specific status class.
Each thread HAS-A user that started it and thus each new thread will be a composition 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

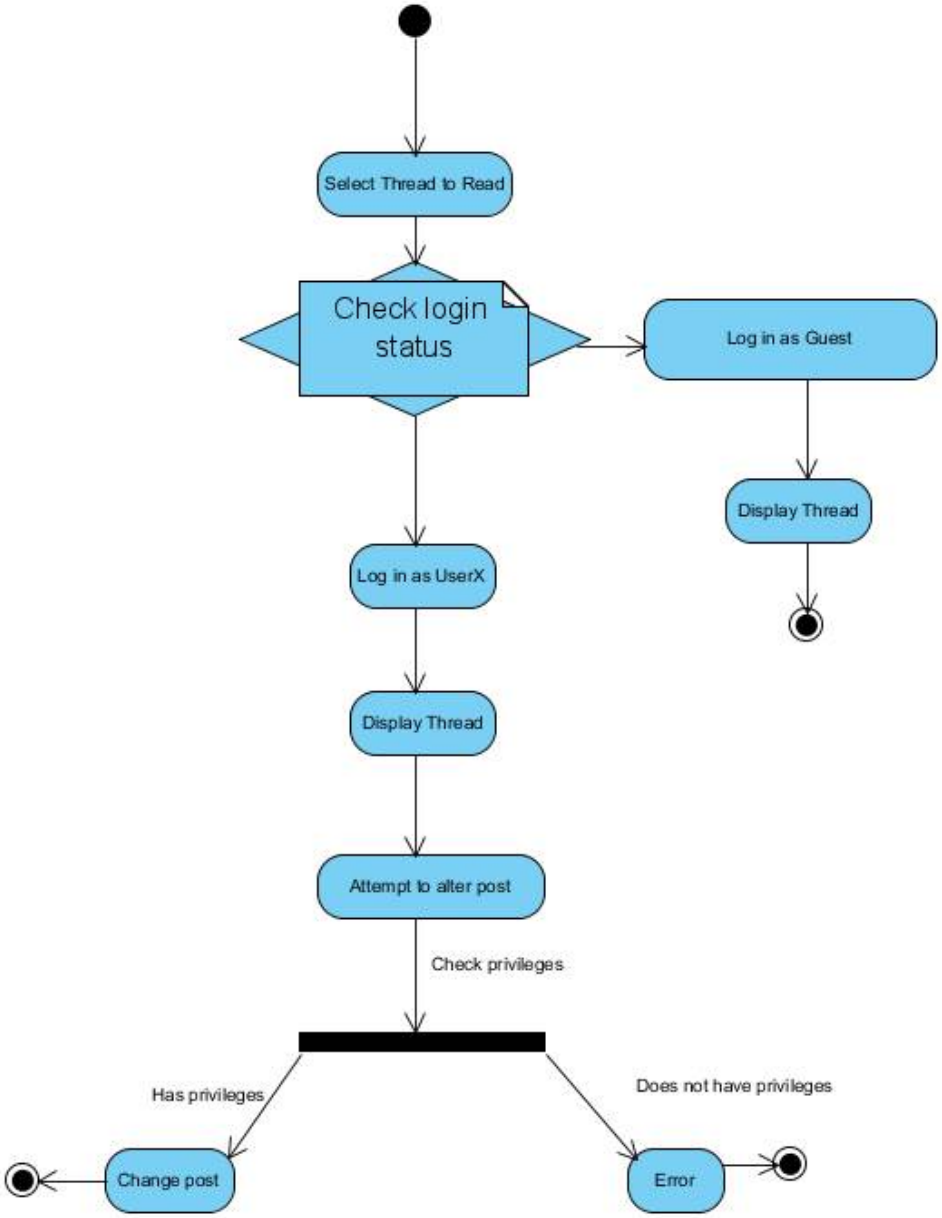
check if the user is logged in.
1.1. If so it will go to number 3 and open the selected thread to read.
1.1.1. If user attempts to alter a post it will go to number 1 above and check the users status
1.1.1.1. The system verifies the status and at number 2 checks that the user has the privileges to create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

2.3 Required Functionality:

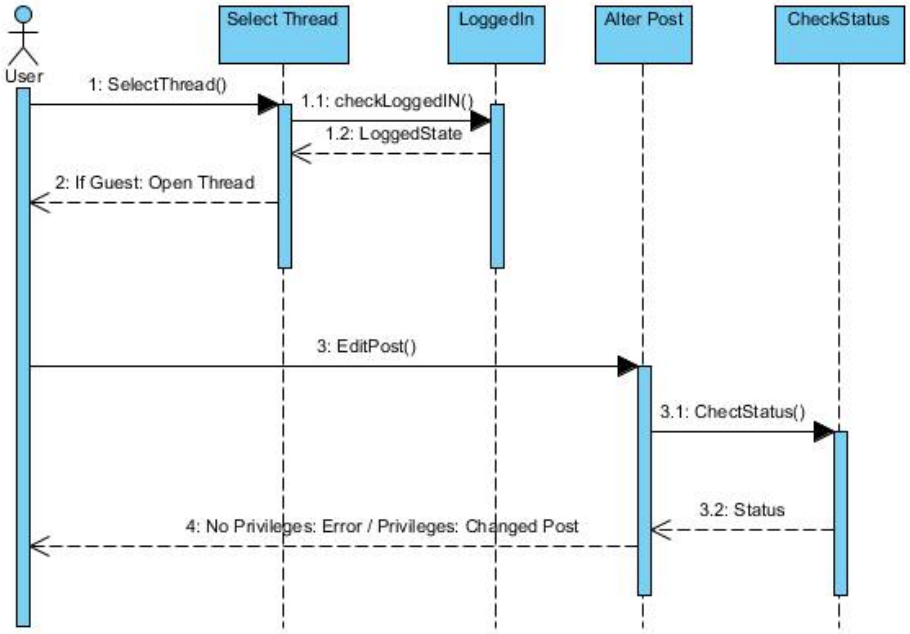


2.4 Process Specifications:

Activity Diagram



Sequence Diagram



3 Update Thread

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.

3.1 Prioritization:

This Use case is considered Important

3.2 Conditions and Data Structures:

Participants:
Initiated by User and communicates with system

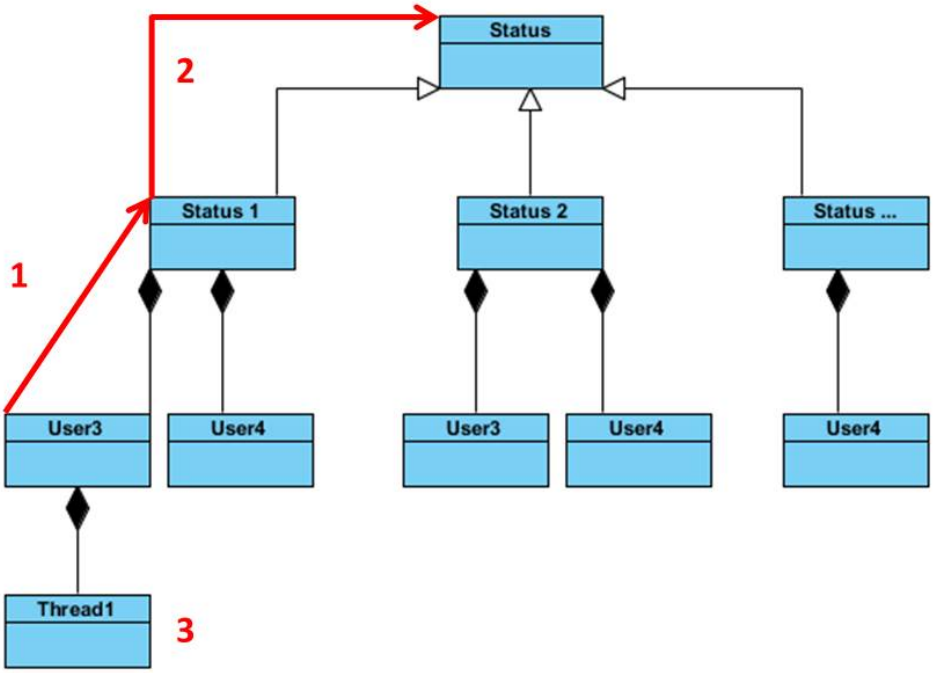
Pre-Conditions:

1. User chooses the way in which to update a thread
2. The User is logged in
2.1. The user has a specific status
2.2. 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.

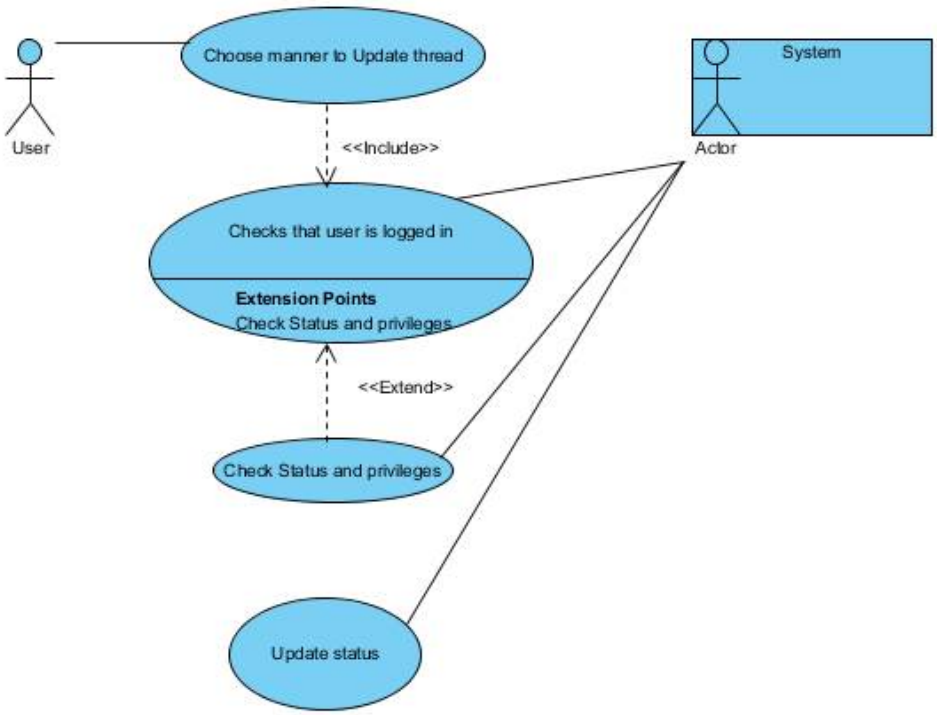
Requests and Results Data Structures:



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

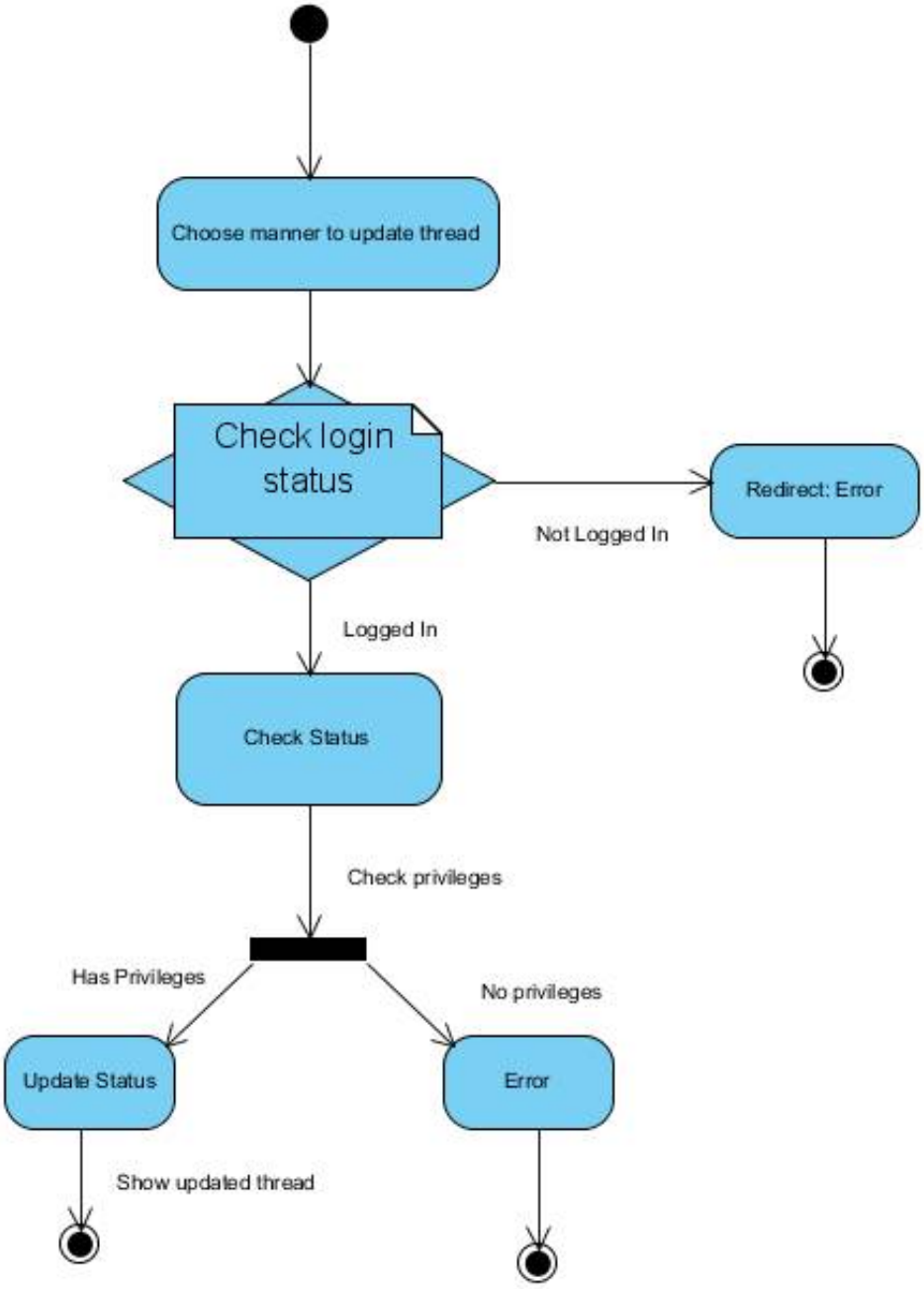
each specific status class.
Each thread HAS-A user that started it and thus each new thread will be a composition of each user.
Steps:
1. If the user attempts to update a thread, the system will first check if the user is logged in.
1.1. If so, it will go to number 1 above and check the users status
1.1.1. The system verifies the status and at number 2 checks that the user has the privileges to update threads in the specific way chosen.
1.1.1.1. If the user has the necessary privileges then the thread is updated at 3
1.1.1.2. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
1.2. 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.

3.3 Required Functionality:

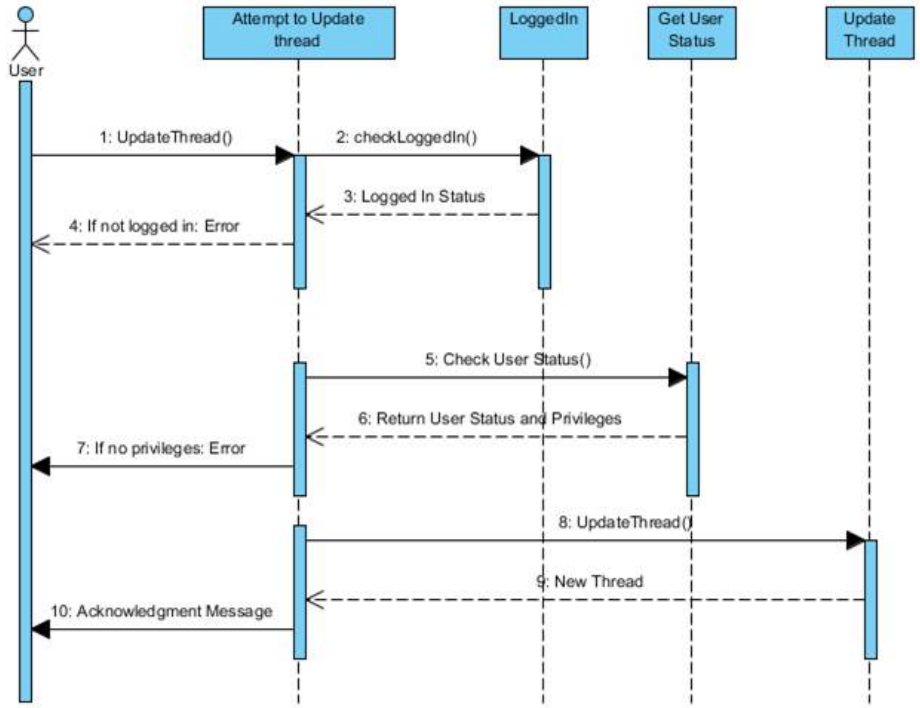


3.4 Process Specifications:

Activity Diagram



Sequence Diagram



4 Delete Thread

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.

4.1 Prioritization:

This Use case is considered Important

4.2 Conditions and Data Structures:

Participants:
Initiated by User and communicates with system

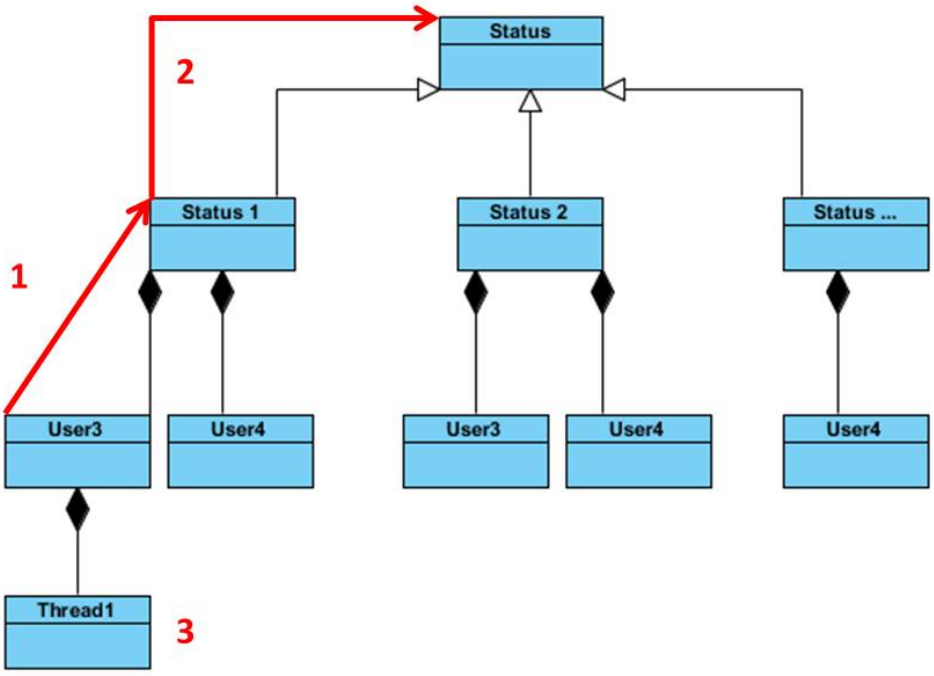
Pre-Conditions:

1. User selects Delete Thread
2. The User is logged in
2.1. The user has a specific status
2.2. 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.

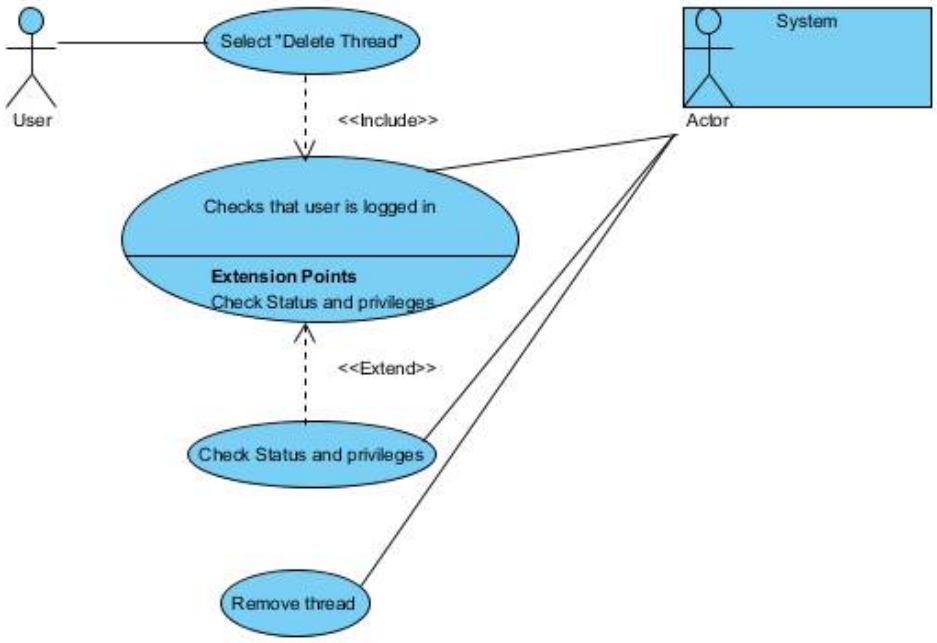
Requests and Results Data Structures:



Status can be an abstract class and each sub-class will be a generalization

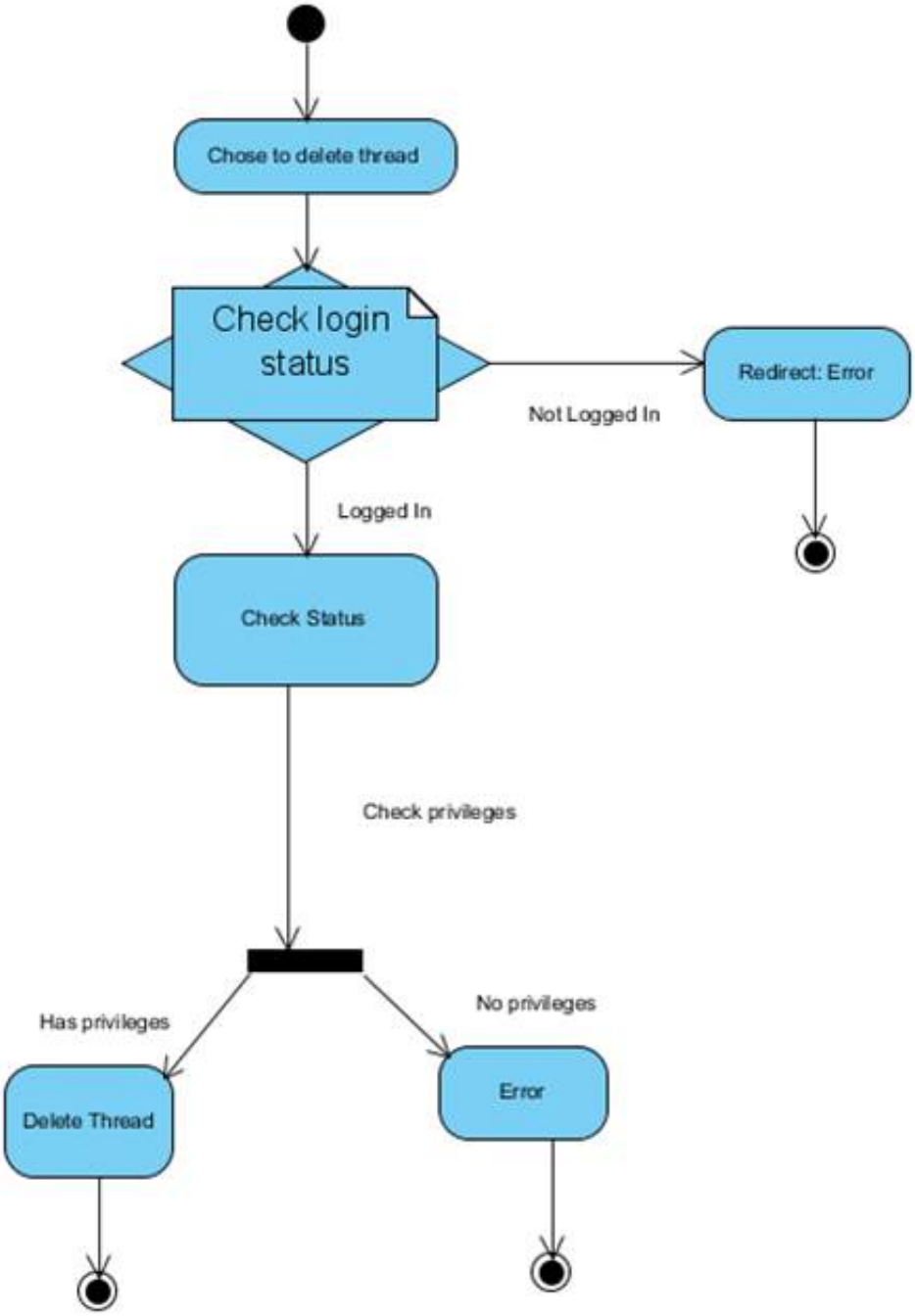
of this class as each new status IS-A status.
Each user HAS-A status and thus each new user will be a composition of each specific status class.
Each thread HAS-A user that started it and thus each new thread will be a composition of each user.
Steps:
1. If the user attempts to delete a thread, the system will first check if the user is logged in.
1.1. If so, it will go to number 1 above and check the users status
1.1.1. The system verifies the status and at number 2 checks that the user has the privileges to delete thread.
1.1.1.1. If the user has the necessary privileges then the thread at 3 is removed from the tree
1.1.1.2. Otherwise an error message is displayed telling the user that he/she does not have the necessary privileges
1.2. 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.

4.3 Required Functionality:

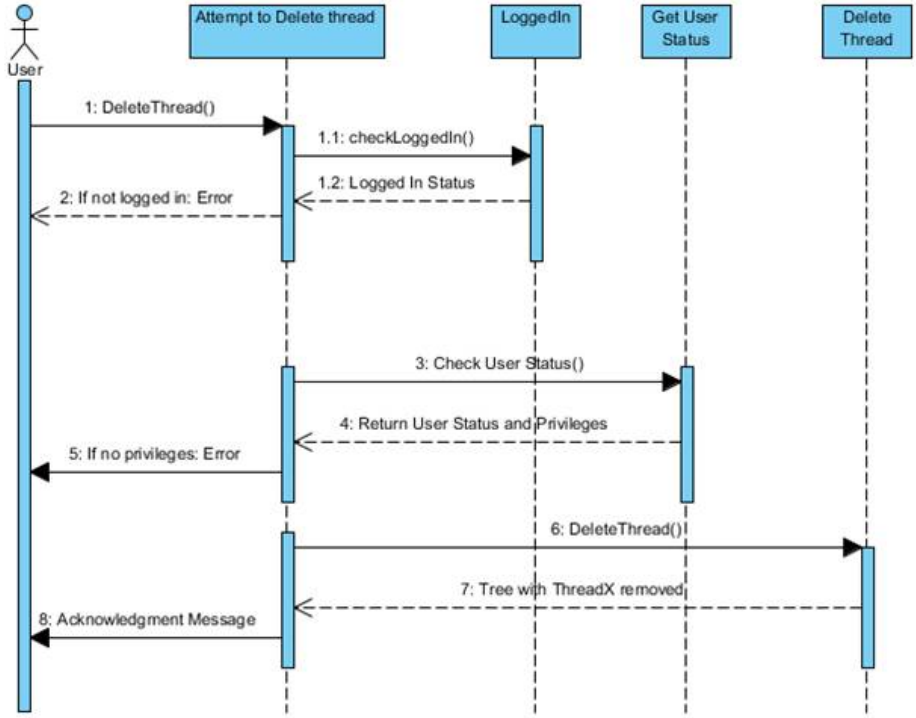


4.4 Process Specifications:

Activity Diagram



Sequence Diagram



5 Summarise Thread

Description:

5.1 Prioritization:

5.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

5.3 Required Functionality:

5.4 Process Specifications:

6 Hide Thread

Description:

6.1 Prioritization:

6.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

6.3 Required Functionality:

6.4 Process Specifications:

7 Close Thread

Description:

7.1 Prioritization:

7.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

7.3 Required Functionality:

7.4 Process Specifications:

8 Move Thread

Description:

8.1 Prioritization:

8.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

8.3 Required Functionality:

8.4 Process Specifications:

Post Handling

Use Cases

1 Create Post

Description:

1.1 Prioritization:

1.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

1.3 Required Functionality:

1.4 Process Specifications:

2 Read Post

Description:

2.1 Prioritization:

2.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

2.3 Required Functionality:

2.4 Process Specifications:

3 Update Post

Description:

3.1 Prioritization:

3.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

3.3 Required Functionality:

3.4 Process Specifications:

4 Delete Post

Description:

4.1 Prioritization:

4.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

4.3 Required Functionality:

4.4 Process Specifications:

5 Mark solution post

in the thread and should make in turn make the post easily visible as one enters the thread.

5.1 Prioritization:

Nice-To-Have: Marking a post as the best answer.

5.2 Conditions and Data Structures:

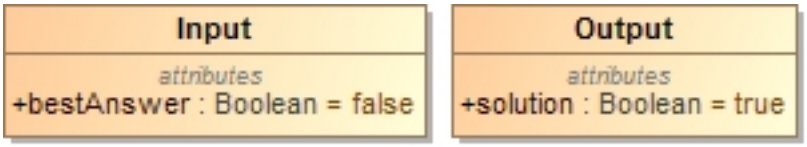
Pre-Conditions:

A user must have the privileges set by the administration of a space in order to mark a post as the best answer. The user is required to be logged in.

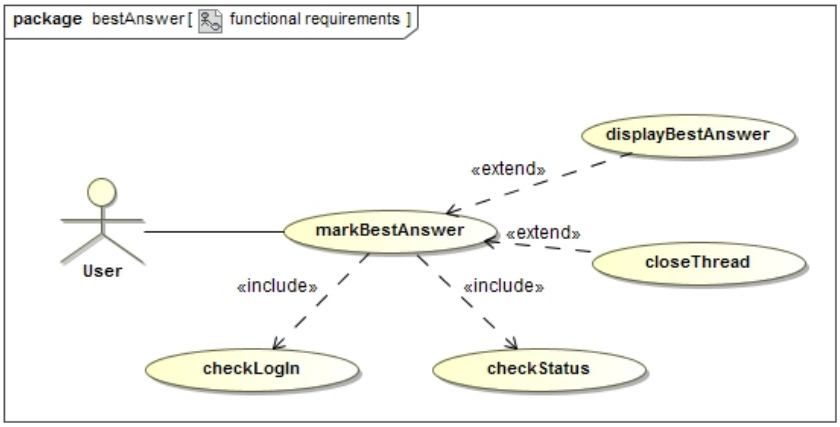
Post-Conditions:

The thread should effectively display that an answer has been selected and optionally close the thread. The poster of the best answer should acquire an increase in progress to the next status level.

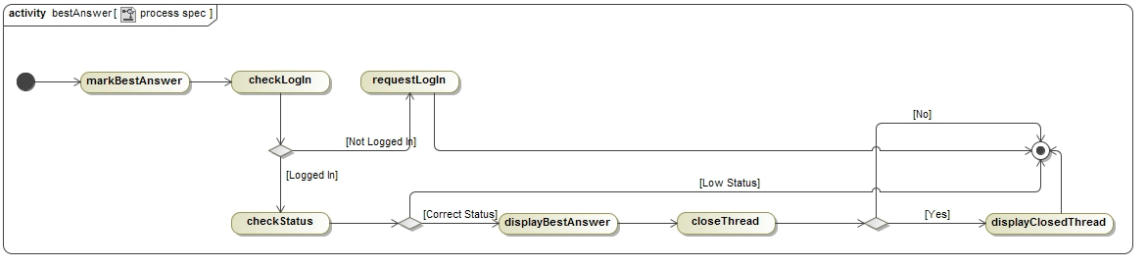
Requests and Results Data Structures:



5.3 Required Functionality:



5.4 Process Specifications:



6 Vote and Evaluate Posts

Description:

This use case refers to the ability of a user to either vote up or down a post; a higher vote count on a post indicates the usefulness of a post. It also refers to a user being able to evaluate or give feedback on another users post.

6.1 Prioritization:

Important: Voting on and evaluation of posts.

6.2 Conditions and Data Structures:

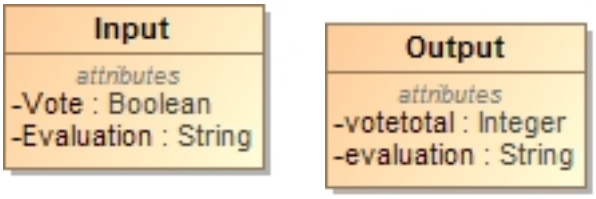
Pre-Conditions:

A post has to have been created for a user to be able to vote on or evaluate it. Any user can vote on or evaluate a post as long as the user is logged in and a part of a specific Buzz Space. An exception to this rule is when the administration of a Buzz Space set specific rules as to which users (perhaps based on status) can vote on or evaluate posts.

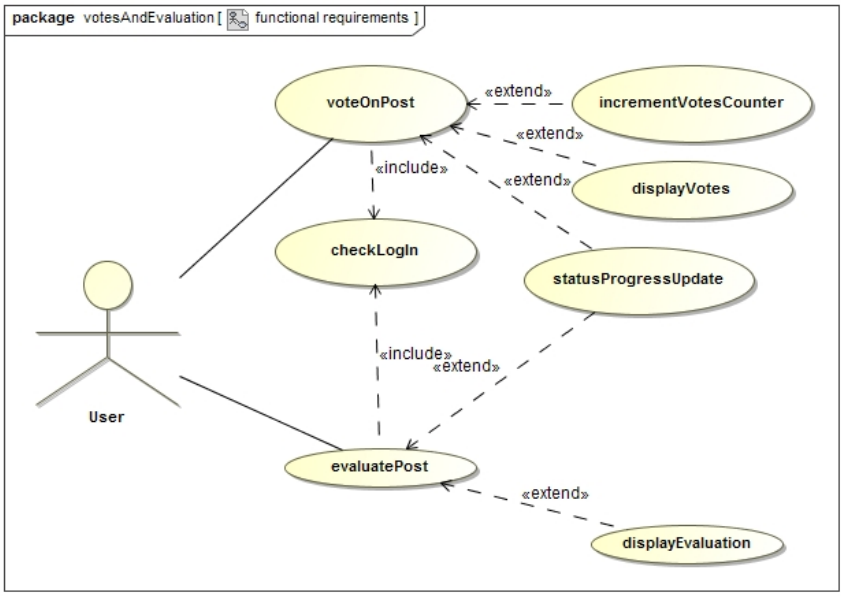
Post-Conditions:

A vote or evaluation has to be visible to every user (or only some based on the administration of that space) after it has been made on a post. A positive vote or evaluation has to reflect upon the posters progress to the next status level.

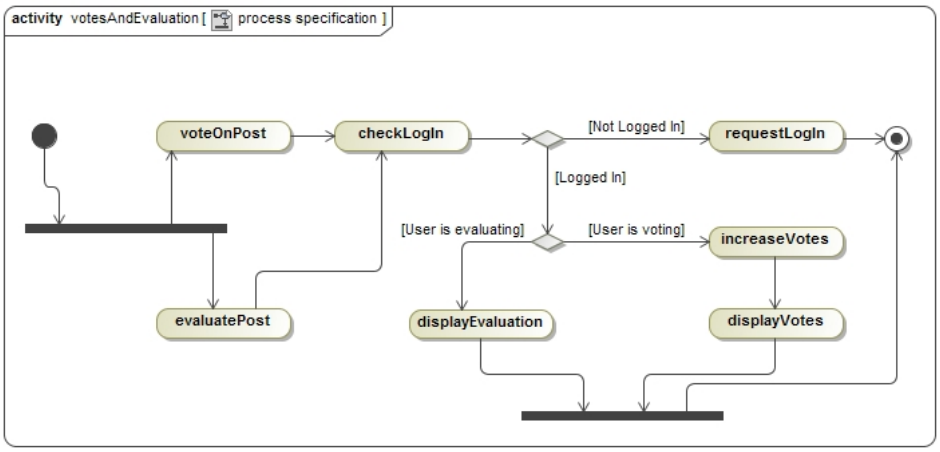
Requests and Results Data Structures:



6.3 Required Functionality:



6.4 Process Specifications:



7 Post Tagging

Description:

Brief:One very important functional requierment is the ability for the user to structure the information and content according to his own preferance and needs. Socilal tagging is the feaure that will enable him to do so

7.1 Prioritization:

Case-Priority - Important: The user should be able to tag topics of interest in the post they created, as well as posts created by other users.
Social tagging saves time and keeps user up to date with latest news and post regarding tagged topics
Users can:
View posts according to tags.
Sort posts according to tags.
Follow different authors of posts and index topics and subjects according to tags.

7.2 Conditions and Data Structures:

Pre-Conditions:

A post must not have been previously marked as a tag. If there are any keywords that look like "links" and have a hash symbol infront of them within the post, that post may be seen as already tagged.

Post-Conditions:

A post becomes a tagged post if it has words with in it that have a hash symbol in front of them.
words that are tags must change colour and become links.
The user can now index topics, follow different authors, sort and view posts according to the tagged topics.

Requests and Results Data Structures:

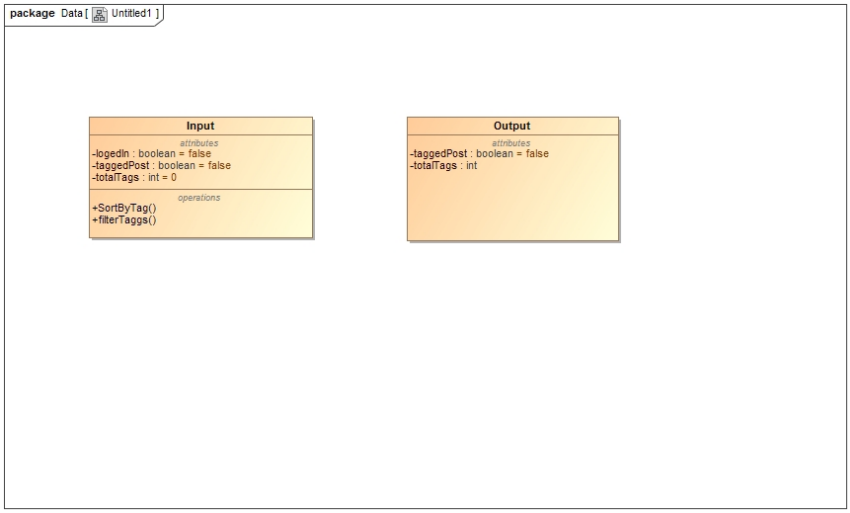


Figure 1: Input and Output classes showing potential attributes and operations *logedIn: boolean = true.

7.3 Required Functionality:

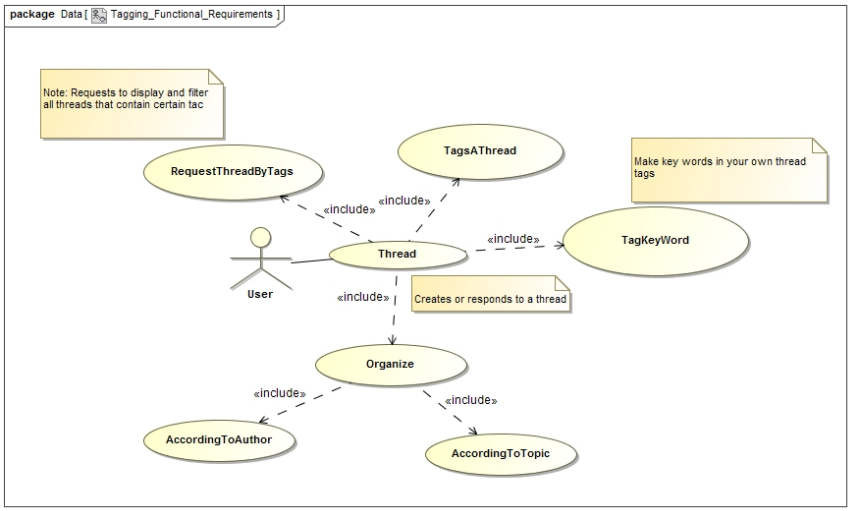


Figure 2: A UML use case diagram showing the required functionality for tagging posts.

7.4 Process Specifications:

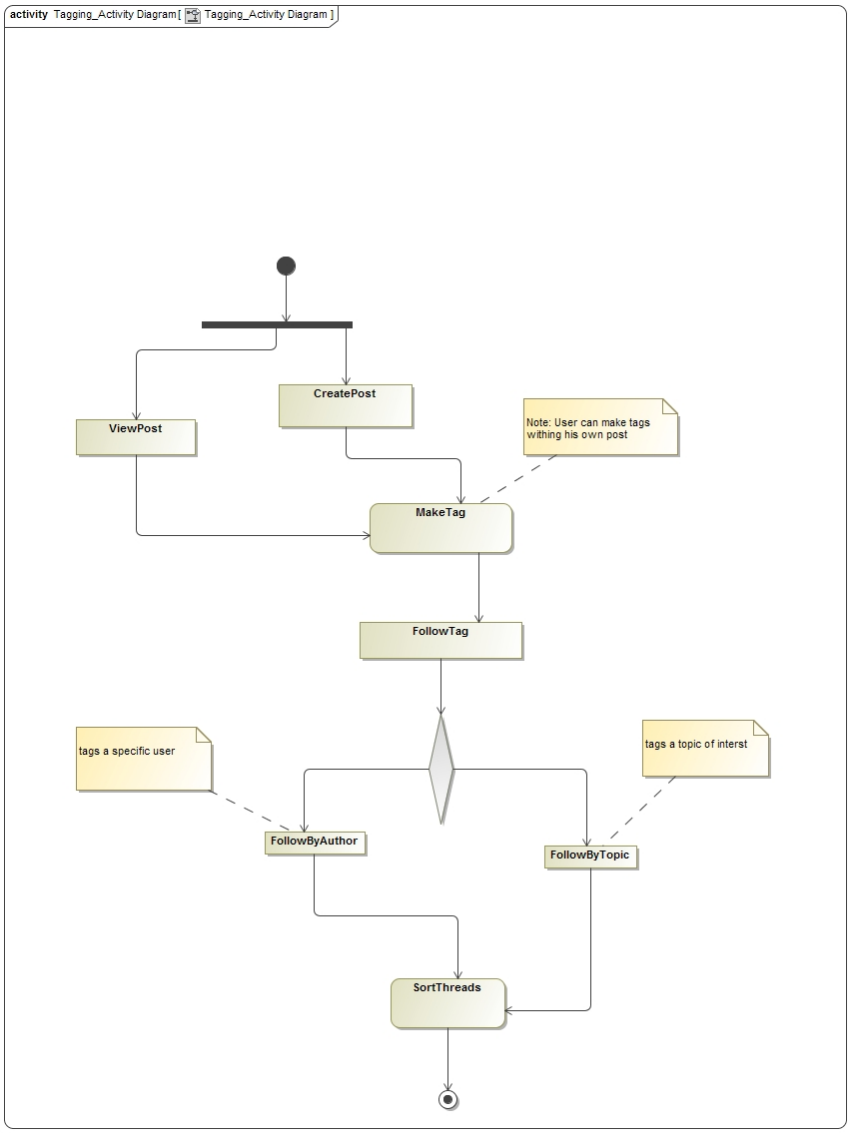


Figure 3: An activity diargram showing the process in wich tags are made and organized.

Filtering
Use Cases

1 Filter threads and buzzspaces

Description:

1.1 Prioritization:

1.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

1.3 Required Functionality:

1.4 Process Specifications:

2 Filter threads and buzzspaces

Description:

2.1 Prioritization:

2.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

2.3 Required Functionality:

2.4 Process Specifications:

Authorisation
Use Cases

1 Login

Description:

1.1 Prioritization:

1.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

1.3 Required Functionality:

1.4 Process Specifications:

2 Logout

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

2.1 Prioritization:

This Use case is considered Important

2.2 Conditions and Data Structures:

Participants:
Initiated by User and communicates with system

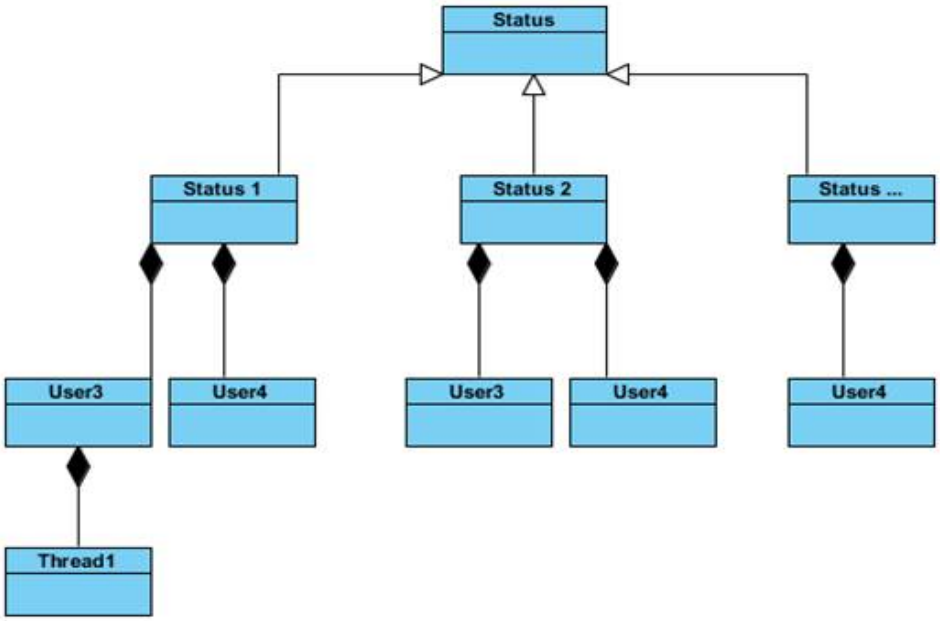
Pre-Conditions:

1. The user must click on Logout function

Post-Conditions:

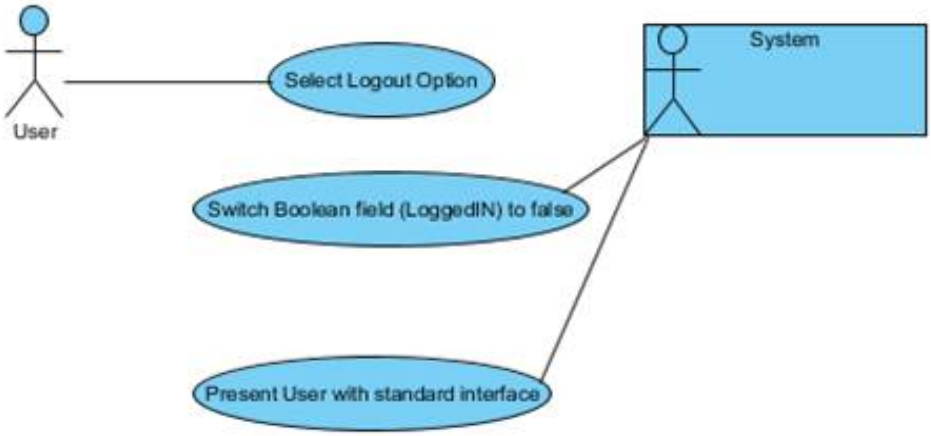
1. System displays user with standard guest interface
2. User has no privileges and status and cannot perform any function that a logged in user can

Requests and Results Data Structures:



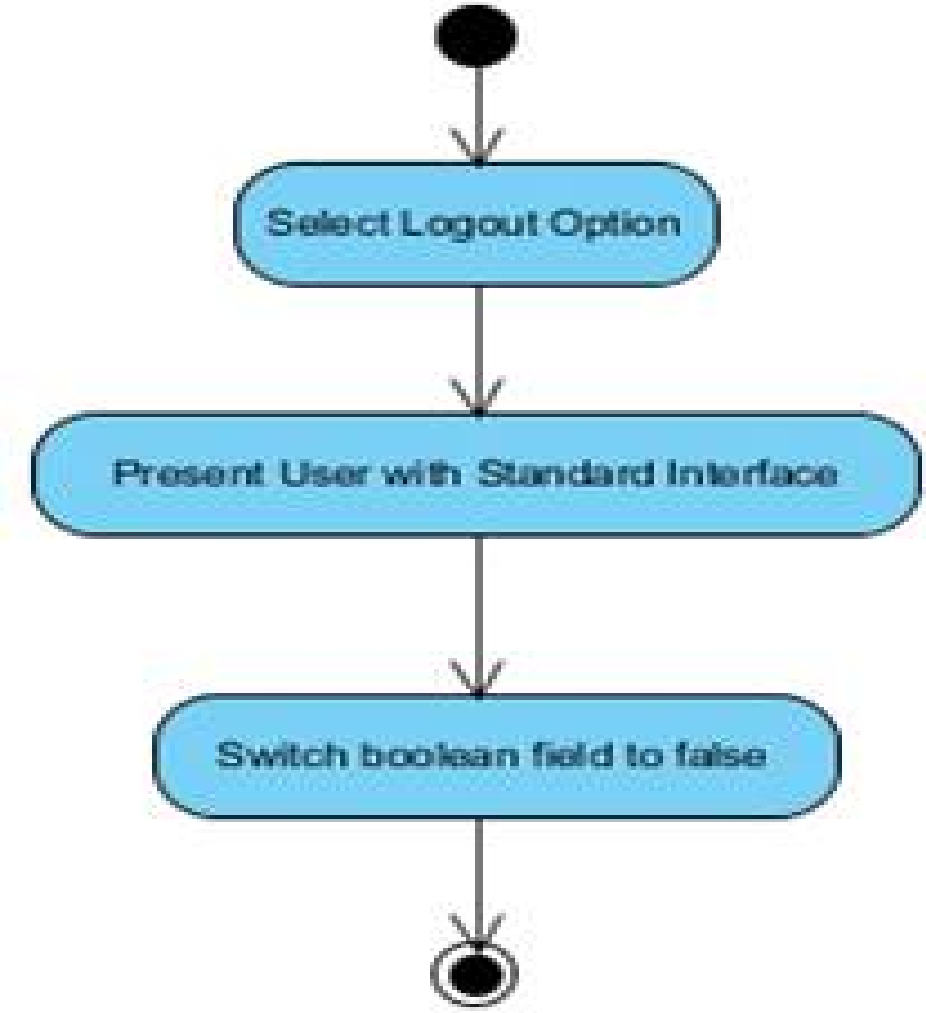
Steps:
1. User selects the logout option
1.1. Boolean field in the specific user class e.g User3 will be switched to false
2. User will be presented with Guest interface

2.3 Required Functionality:

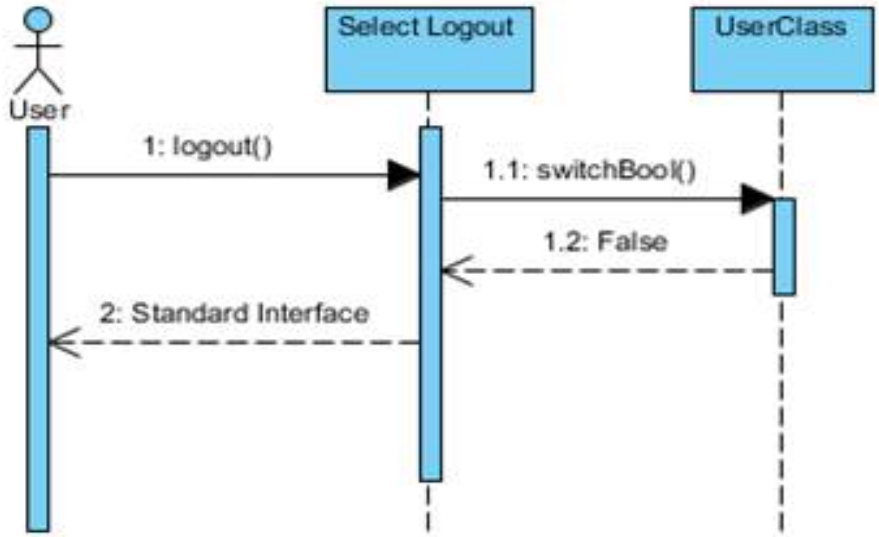


2.4 Process Specifications:

Activity Diagram



Sequence Diagram



3 Registration

Description:

3.1 Prioritization:

3.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

3.3 Required Functionality:

3.4 Process Specifications:

Reporting
Use Cases

1 Statistical Reporting

Description:

1.1 Prioritization:

1.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

1.3 Required Functionality:

1.4 Process Specifications:

Profile Handling
Use Cases

1 Modify Profile

Description:

Description: Profile modification allows the user logged into the system to modify the user profile that was previously created.

1.1 Prioritization:

Important

1.2 Conditions and Data Structures:

Pre-Conditions:

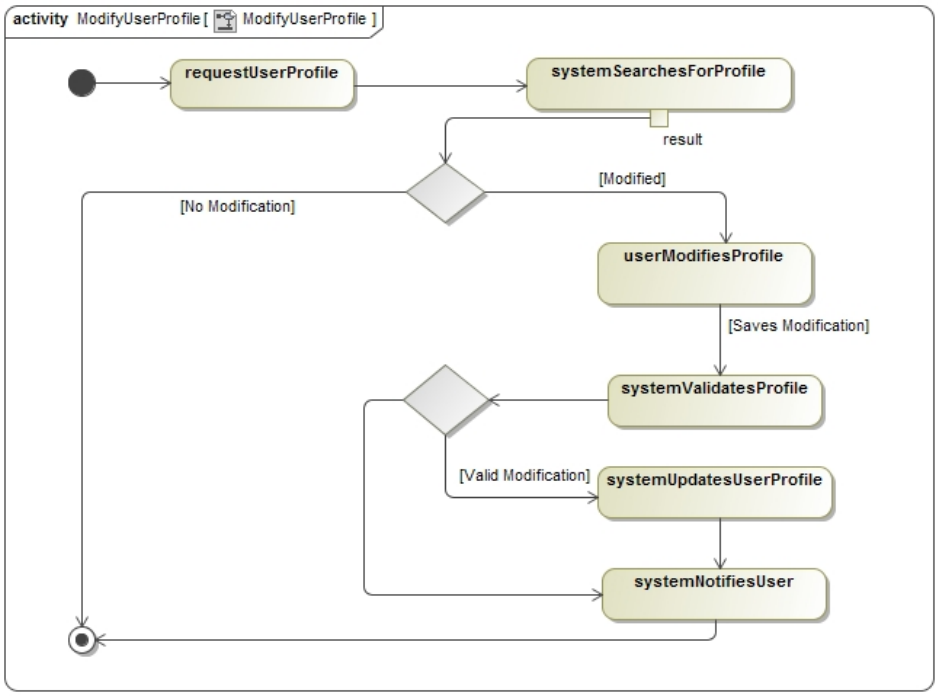
User logged into the system. User has an already existing profile.

Post-Conditions:

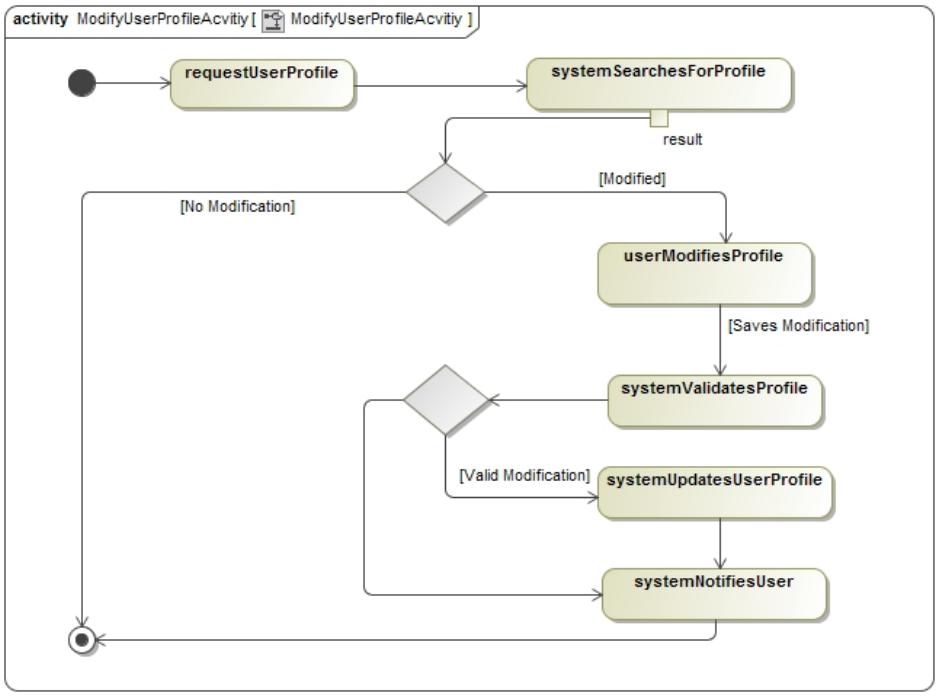
User modifies existing profile: User is logged in and user profile has been modified. User keeps already existing profile: User had logged into the system and the profile had not been modified.

Requests and Results Data Structures:

1.3 Required Functionality:



1.4 Process Specifications:



2 Create Profile

Description:

2.1 Prioritization:

Critical

2.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

2.3 Required Functionality:

2.4 Process Specifications:

3 Create Private Message

Description:

3.1 Prioritization:

3.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

3.3 Required Functionality:

3.4 Process Specifications:

4 Read Private Message

Description:

4.1 Prioritization:

4.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

4.3 Required Functionality:

4.4 Process Specifications:

5 Update Private Message

Description:

5.1 Prioritization:

5.2 Conditions and Data Structures:

Pre-Conditions:

Post-Conditions:

Requests and Results Data Structures:

5.3 Required Functionality:

5.4 Process Specifications:

6 Delete Private Message

Description:

6.1 Prioritization:

6.2 Conditions and Data Structures:

Domain Model