

COS301: MINI PROJECT PHASE 1

GROUP 5 A

Khathutshelo Matidza 11072157

Renaldo van Dyk 12204359

Andreas du Preez 12207871

Sean Hill 12221458

Kgomotso Sito 12243273

Hlavutelo Maluleke 12318109

Siboniso Masilela 10416260

Semaka Malapane 13081129

[Github](#)

Contents

1	User Management	4
1.1	Use case prioritization	4
1.1.1	Critical	4
1.1.2	Important	4
1.1.3	Nice-To-Have	4
1.2	Use case/Services contracts	4
1.2.1	Pre-Conditions	4
1.2.2	Post-Conditions	4
1.3	Required functionality	5
1.4	Process specifications	6
1.4.1	User Management	6
2	Admin Management	6
2.1	Use case prioritization	6
2.1.1	Critical	6
2.1.2	Important	7
2.1.3	Nice-To-Have	7
2.1.4	Pre-Conditions	7
2.1.5	Post-Conditions	7
2.2	Required functionality	8
2.3	Process specifications	9
2.3.1	User Management	9
3	Threads	9
3.1	Use case prioritization	9
3.1.1	Critical	9
3.2	Important	10
3.2.1	Nice-To-Have	10
3.3	Use case/Services contracts	10
3.3.1	Pre-Conditions	10
3.3.2	Post-Conditions	10
3.4	Required functionality	11
3.5	Process specifications	12
3.5.1	Create Threads	12
3.5.2	Delete Threads	13
3.5.3	Read Threads	14

3.5.4	Read Tracking	15
4	User Ranking	15
4.1	Use case prioritization	15
4.1.1	Critical	15
4.1.2	Important	15
4.1.3	Nice-To-Have	16
4.2	Use case/Services contracts	16
4.2.1	Pre-Conditions	16
4.2.2	Post-Conditions	16
4.3	Required functionality	16
4.4	Process specifications	17
4.4.1	Rating Post	17
4.4.2	Mark Allocation	18
4.4.3	Manage Ranking	19
5	Social Tagging	19
5.1	Use case prioritization	19
5.1.1	Critical	19
5.1.2	Important	19
5.1.3	Nice-To-Have	20
5.2	Use case/Services contracts	20
5.2.1	Pre-Conditions	20
5.2.2	Post-Conditions	20
5.3	Request and Results Data Structures	21
5.4	Process specifications	22
5.4.1	Social Tagging	22

1 User Management

1.1 Use case prioritization

1.1.1 Critical

- 1.1 updateAccountInformation
- 1.2 viewProfile
- 1.3 logIn
- 1.4 logOut

1.1.2 Important

- 1.5 directMessage

1.1.3 Nice-To-Have

1.2 Use case/Services contracts

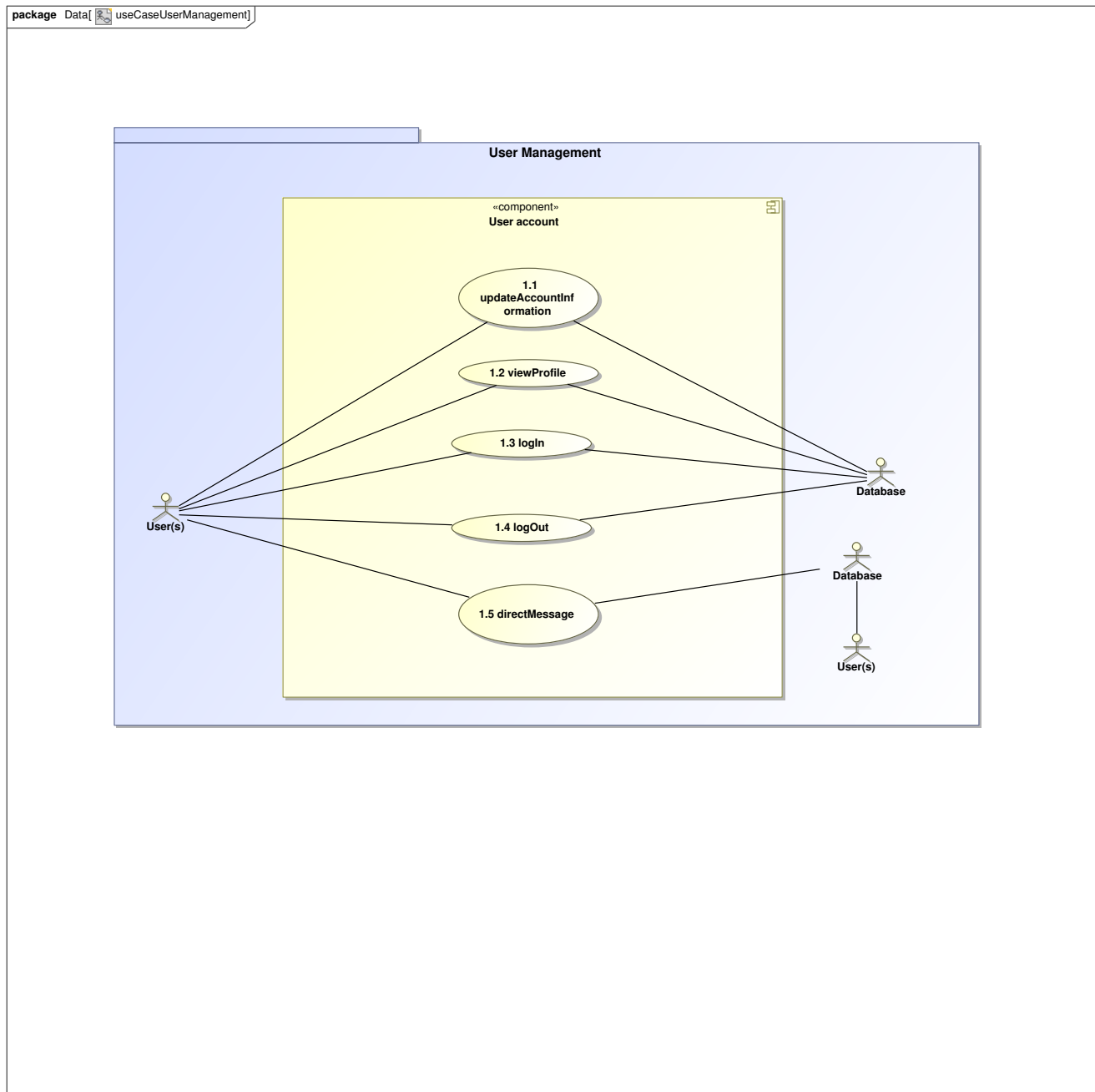
1.2.1 Pre-Conditions

- 1.1 New information must be valid.
- 1.4 User must be logged in.
- 1.5a User must have a message.
- 1.5b User must have (a) valid recipient(s).

1.2.2 Post-Conditions

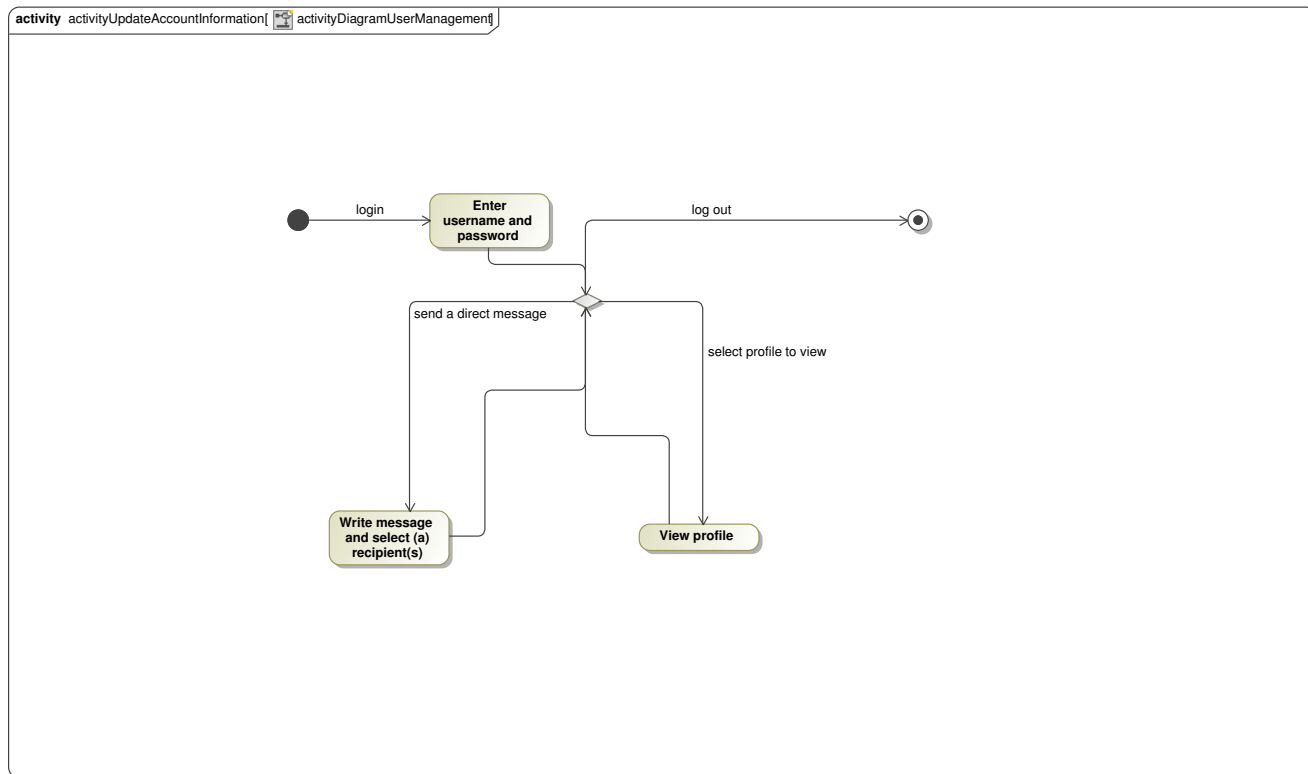
- 1.1 Information is updated on database.
- 1.4 User is logged out.
- 1.5 (The) recipient(s) receive message.

1.3 Required functionality



1.4 Process specifications

1.4.1 User Management



2 Admin Management

2.1 Use case prioritization

2.1.1 Critical

- 2.1 createAccount
- 2.2 deleteAccount

2.1.2 Important

2.1.3 Nice-To-Have

- 2.3 customizeInterface
- 2.2 summarizeThreads

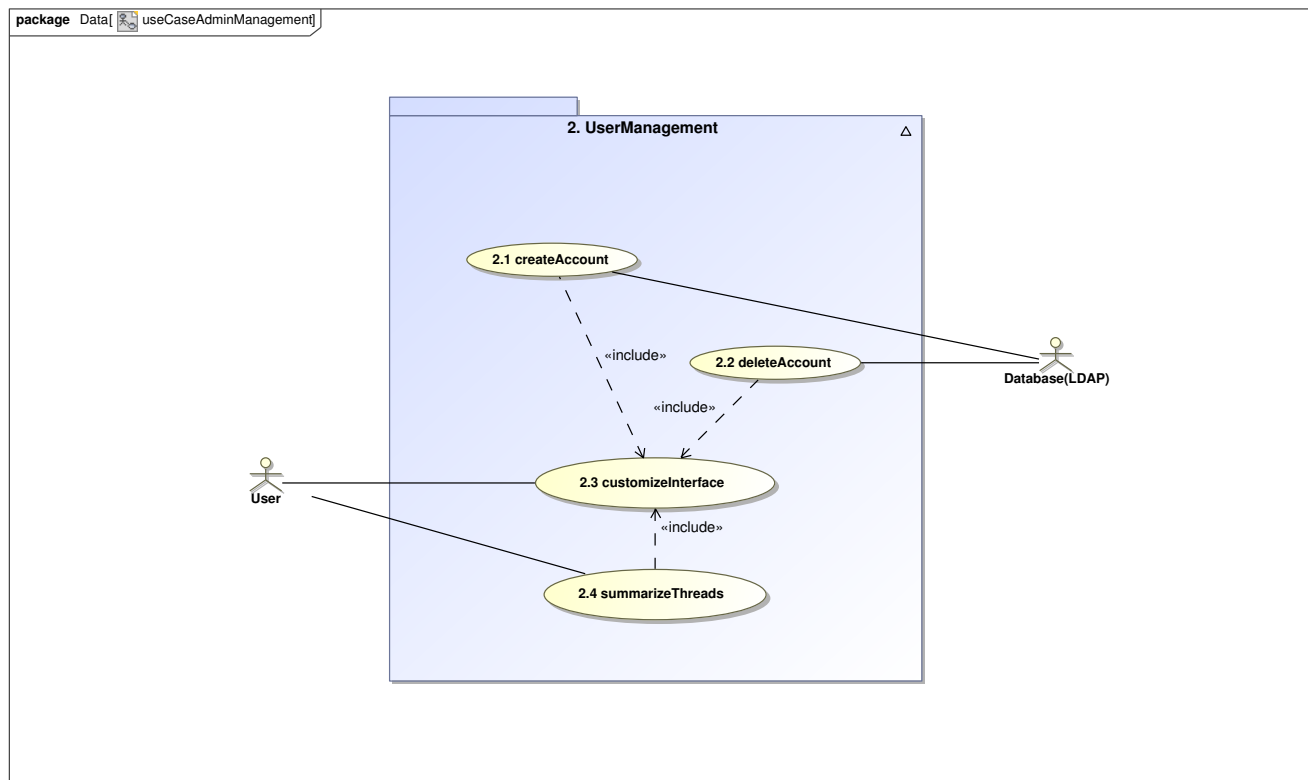
2.1.4 Pre-Conditions

- 2.1 User must be a registered with the university
- 2.2 User must no longer be a student at the university
- 2.3 User must be of a specific status level(authority)
- 2.4 User must be of a specific status level(authority)

2.1.5 Post-Conditions

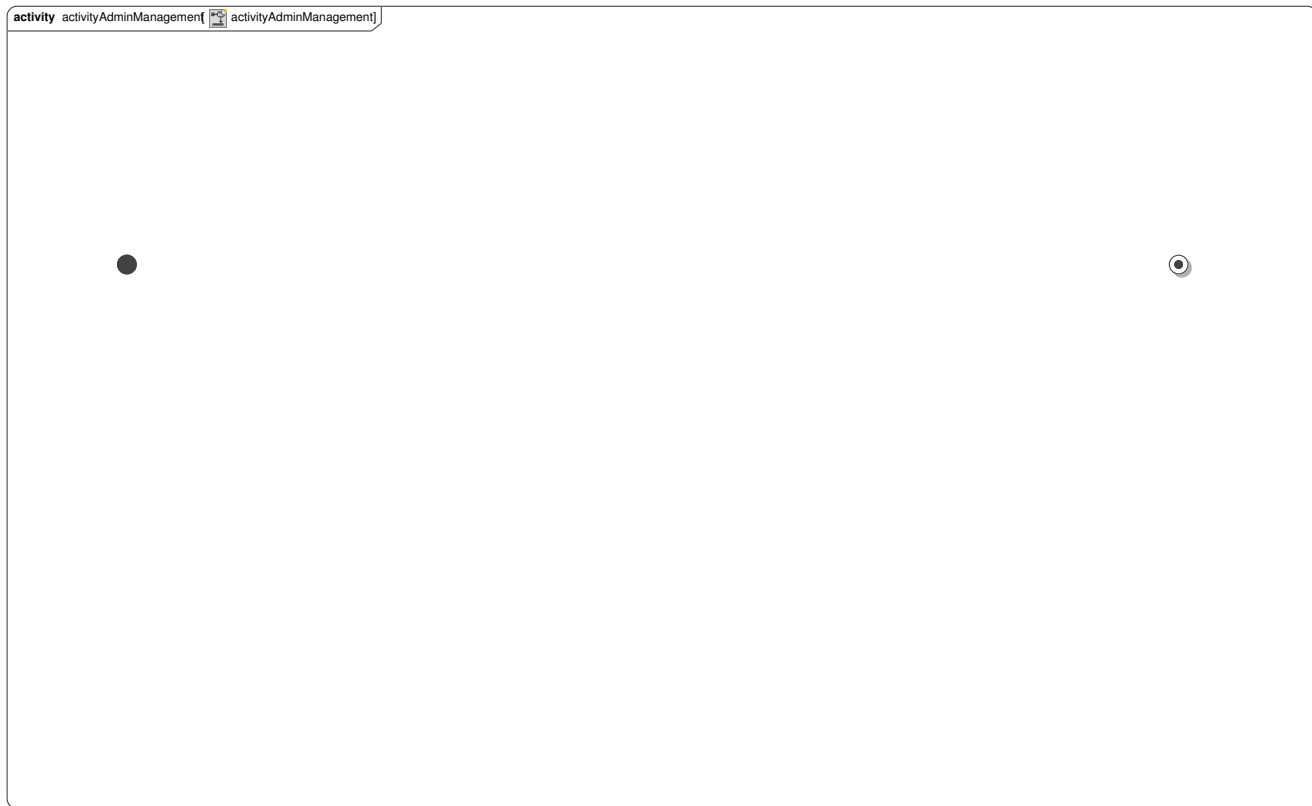
- 2.1 Account must exist
- 2.2 User account should be deleted along with all activities associated with it
- 2.3 Posts should be moved around and changes be visible
- 2.4 Summaries of threads should be created

2.2 Required functionality



2.3 Process specifications

2.3.1 User Management



3 Threads

3.1 Use case prioritization

3.1.1 Critical

- 3.1 createRootThread
- 3.1.1 createSubThread
- 3.2 deleteThreads

- 3.2.1 archiveThreads
- 3.3 updateThreads
- 3.4 readThreads

3.2 Important

- 3.5 Read Tracking

3.2.1 Nice-To-Have

3.3 Use case/Services contracts

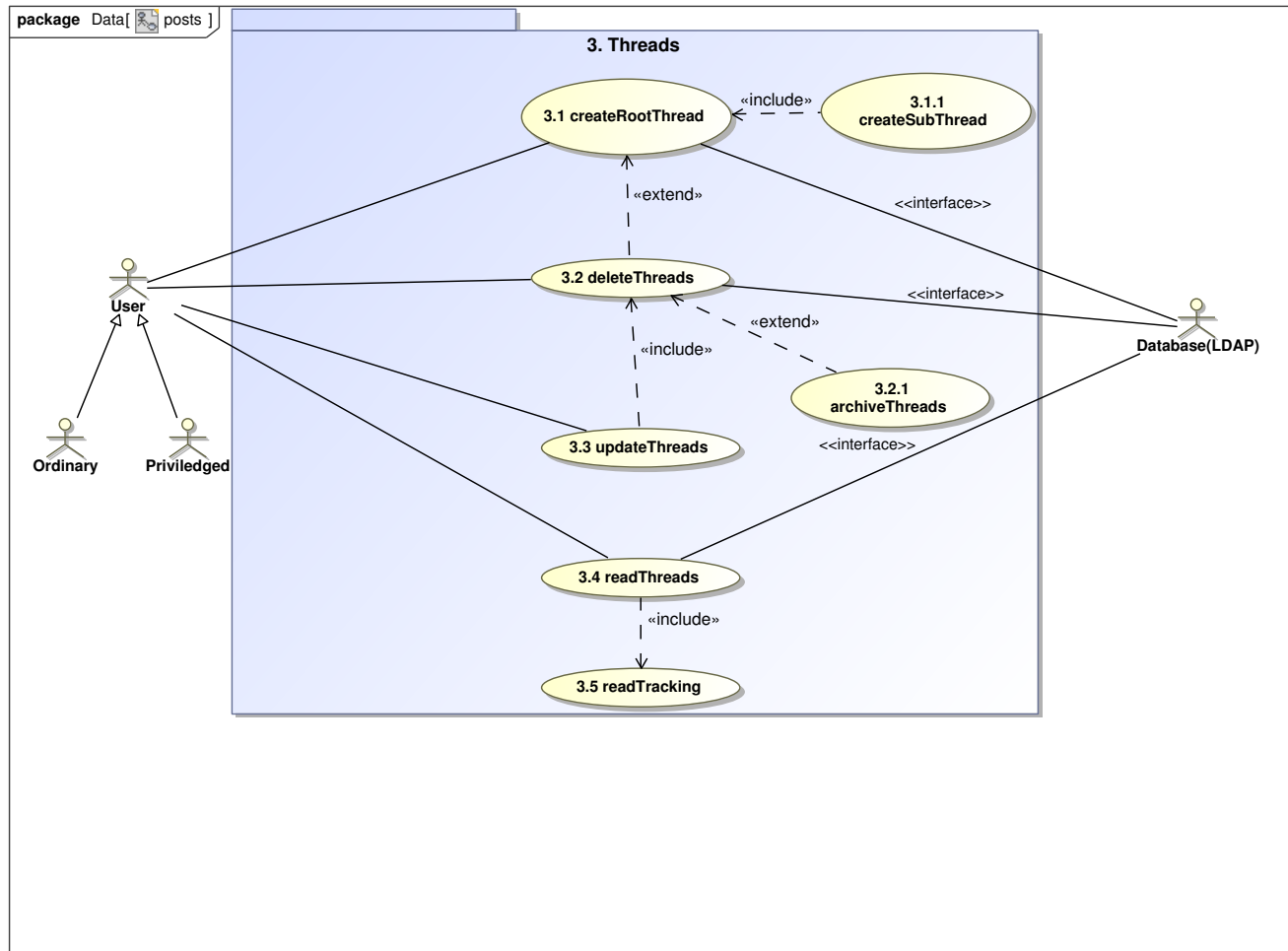
3.3.1 Pre-Conditions

- 3.1 a) User must be logged into the system to create root threads
- 3.2 a) Thread must be read from database before it can be deleted
- 3.2.1 a) Thread must be deleted before it can be archived
- 3.3 a) User must be logged into the system to update the database
- b) Before the database can be updated values must be changed
- 3.4 a) User must be logged into the system to read values from database
- 3.5 a) User must be logged into the system to see unread messages in bold

3.3.2 Post-Conditions

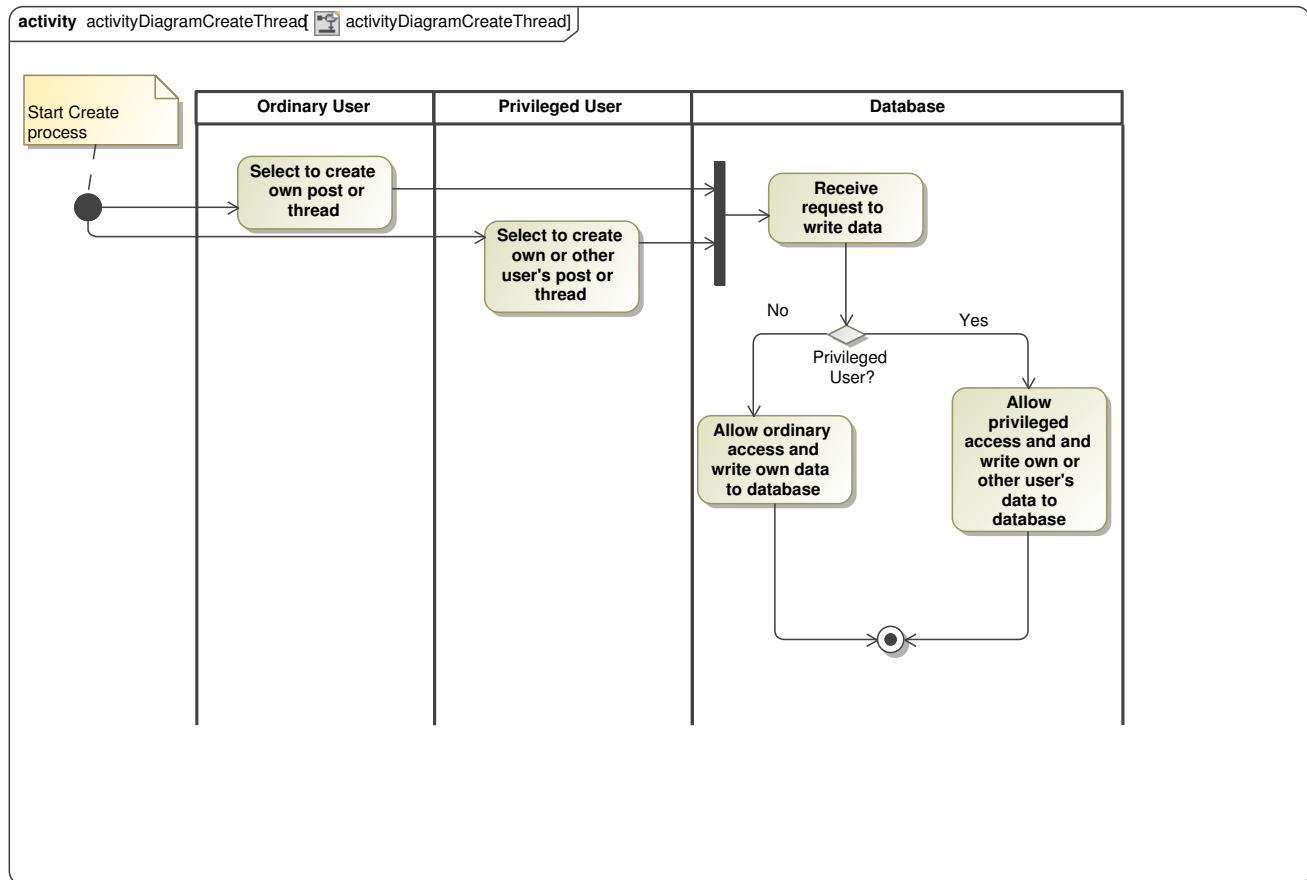
- 3.1 a) All threads will be visible to users who are logged into the system
- b) Threads will exist in database
- 3.2 a) Threads will be archived.
- b) Threads will not be visible to users
- 3.3 a) Updated threads will be visible
- 3.5 a) Thread is marked as unread

3.4 Required functionality

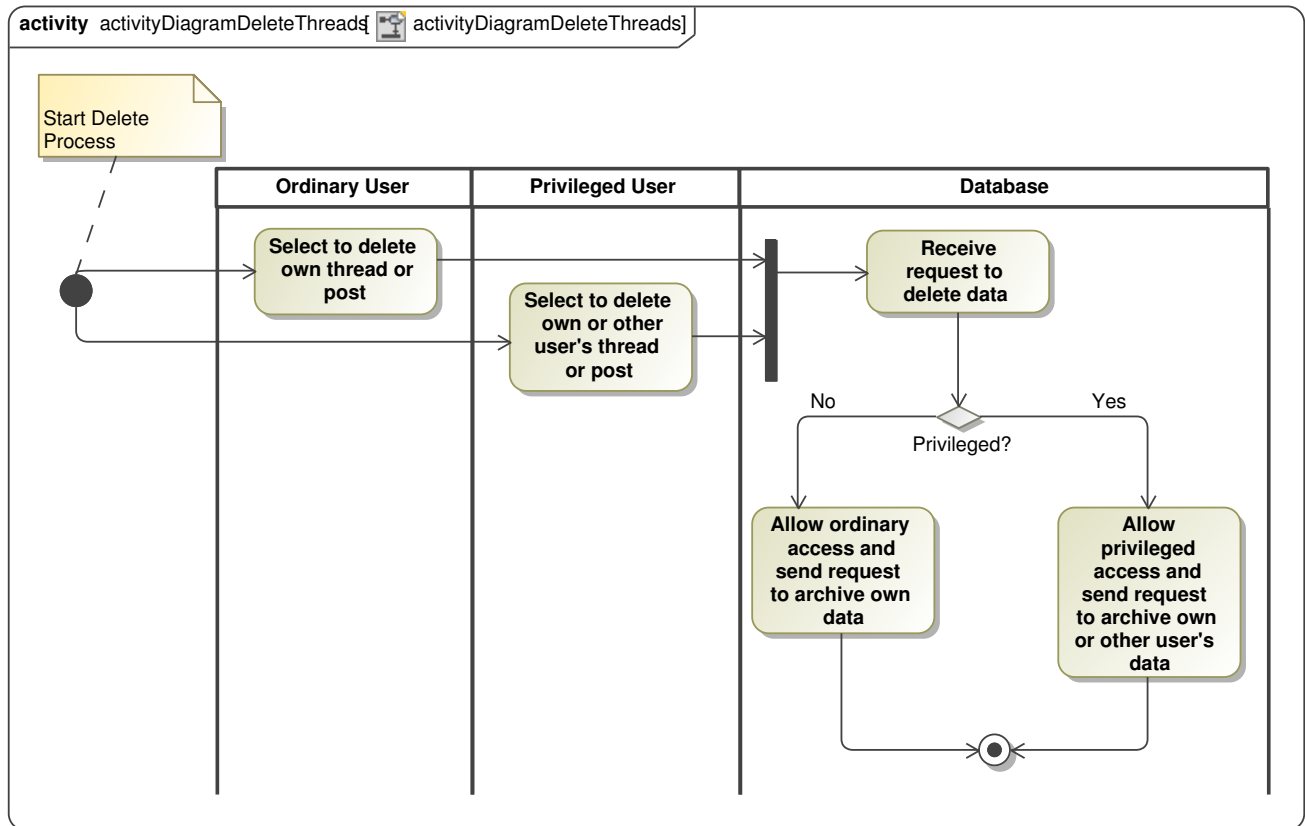


3.5 Process specifications

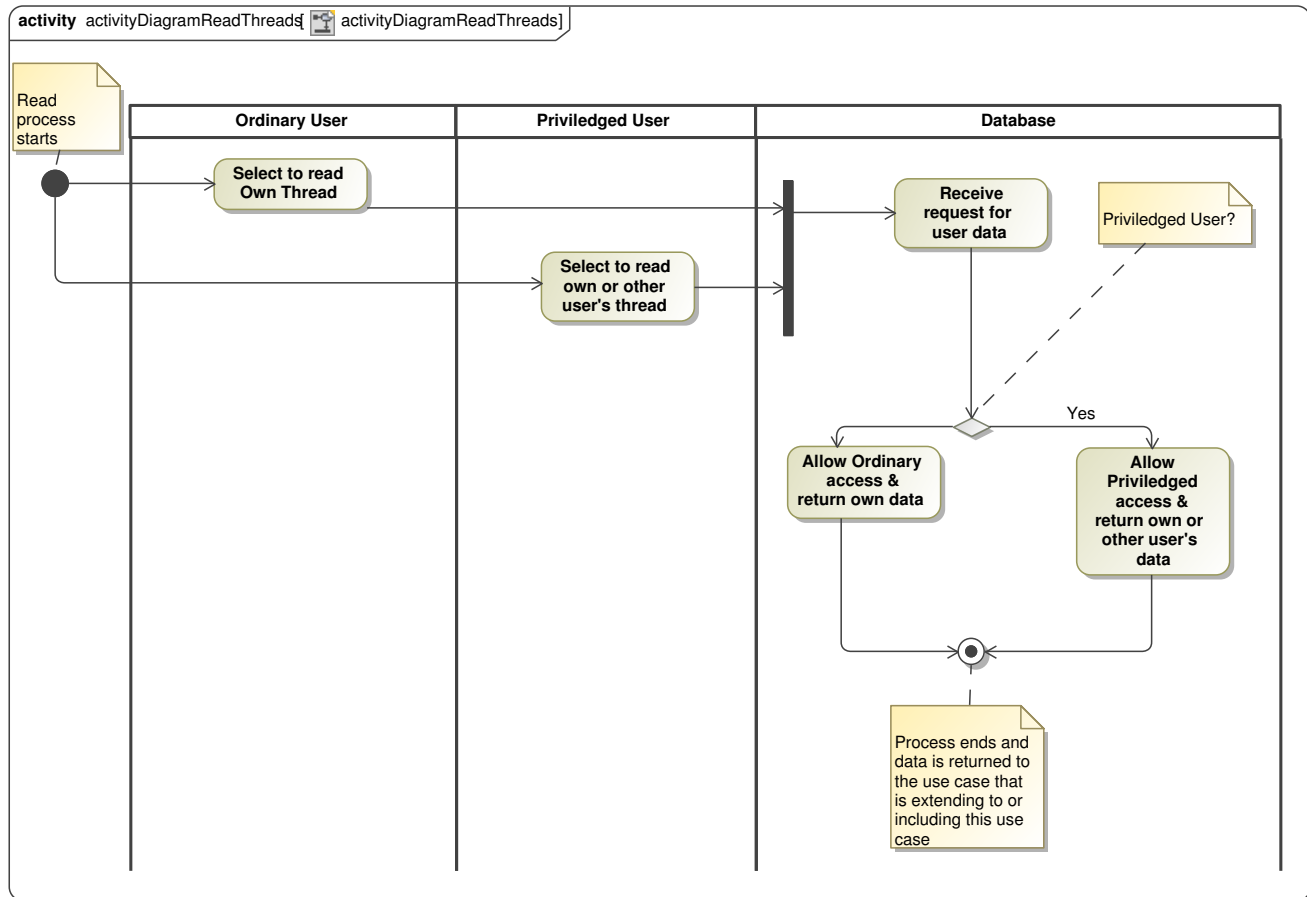
3.5.1 Create Threads



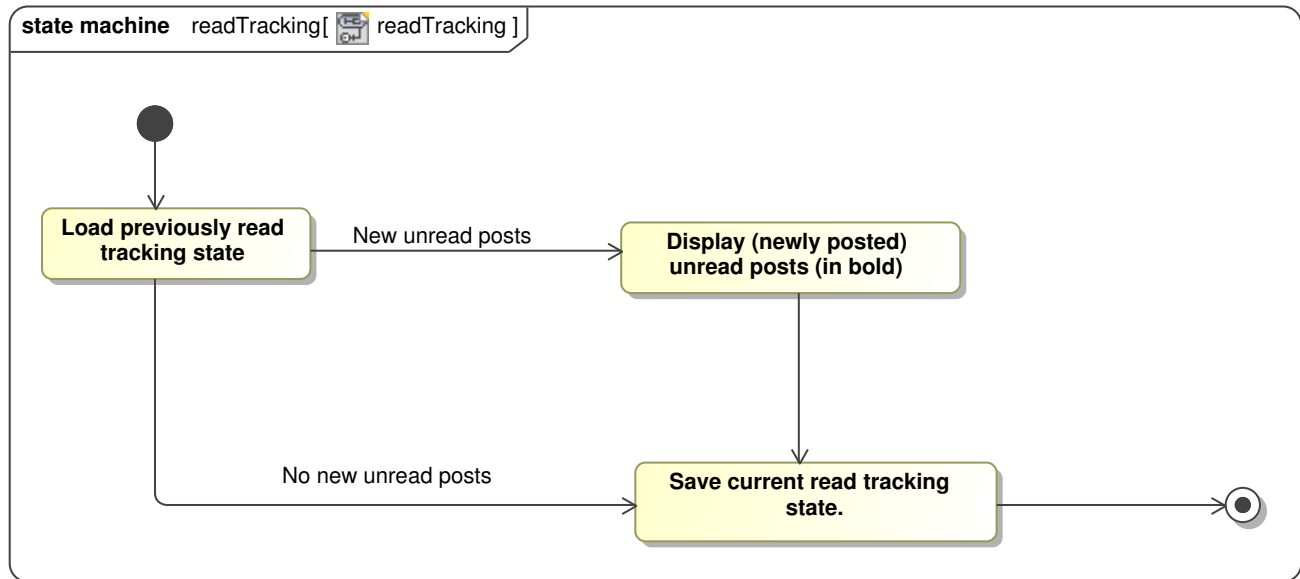
3.5.2 Delete Threads



3.5.3 Read Threads



3.5.4 Read Tracking



4 User Ranking

4.1 Use case prioritization

4.1.1 Critical

- 4.1 Rate post
- 4.3 Manage User Ranking

4.1.2 Important

- 4.2 Mark Allocation

4.1.3 Nice-To-Have

4.2 Use case/Services contracts

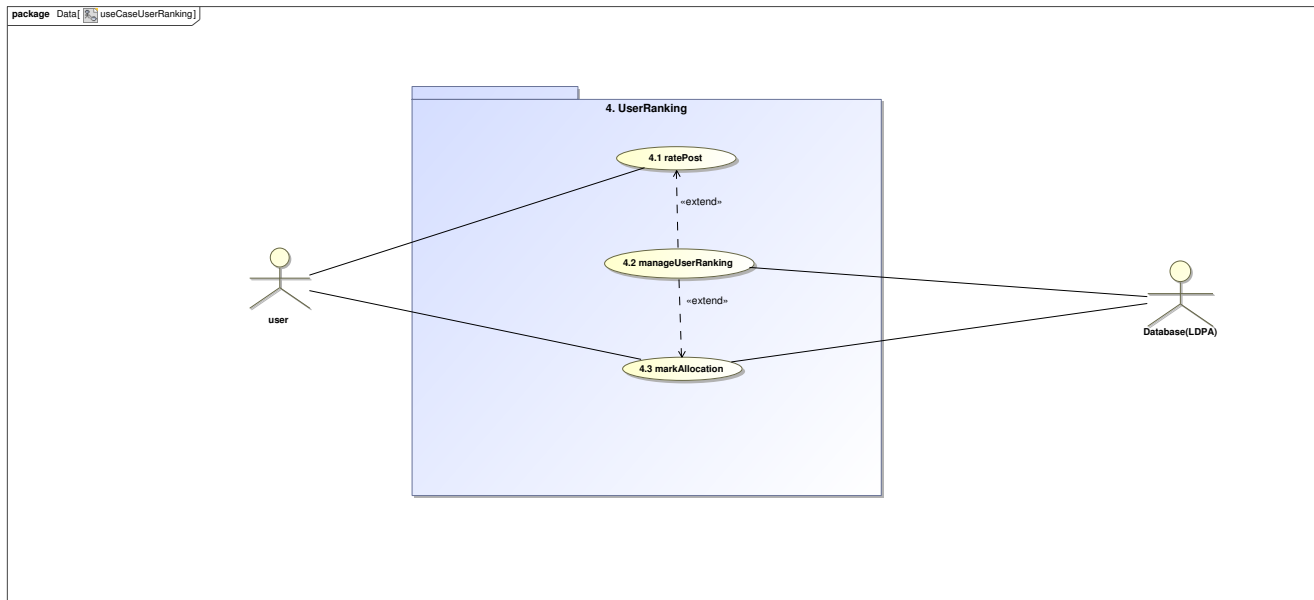
4.2.1 Pre-Conditions

- 4.1 Must be logged in
- 4.2 Must be logged in and also a lecturer or TA
- 4.3 Must be logged in and also a lecturer or TA

4.2.2 Post-Conditions

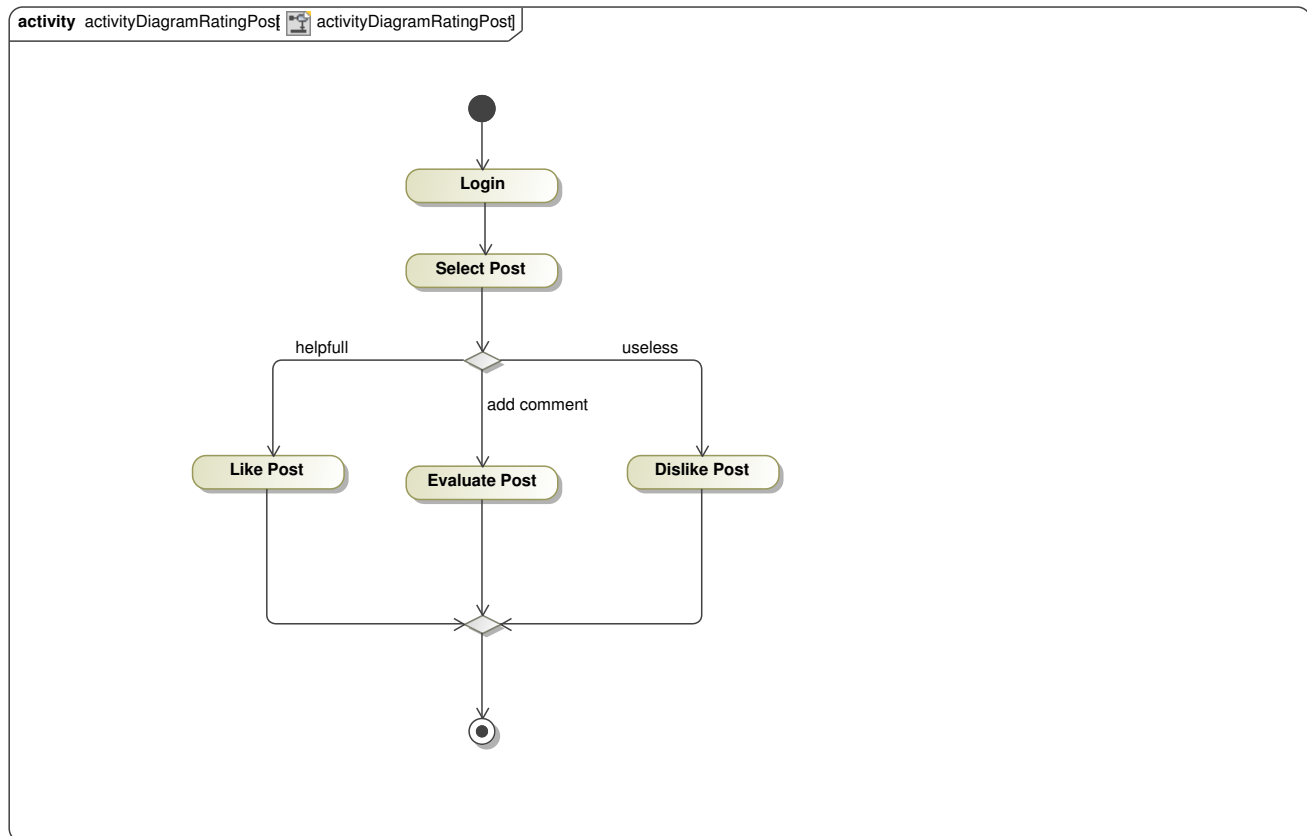
- 4.1 Post is rated
- 4.2 Must be logged in and also a lecturer or TA
- 4.3 Must be logged in and also a lecturer or TA

4.3 Required functionality

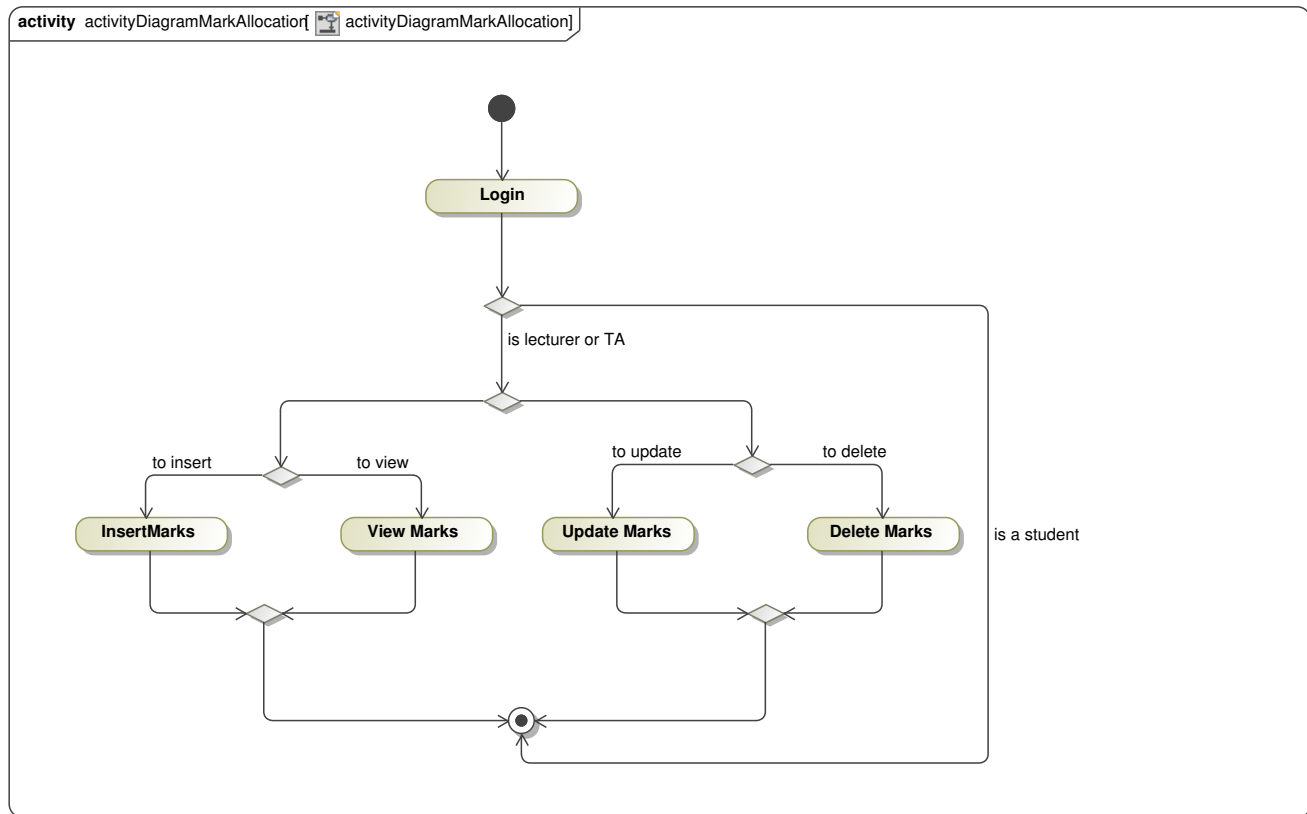


4.4 Process specifications

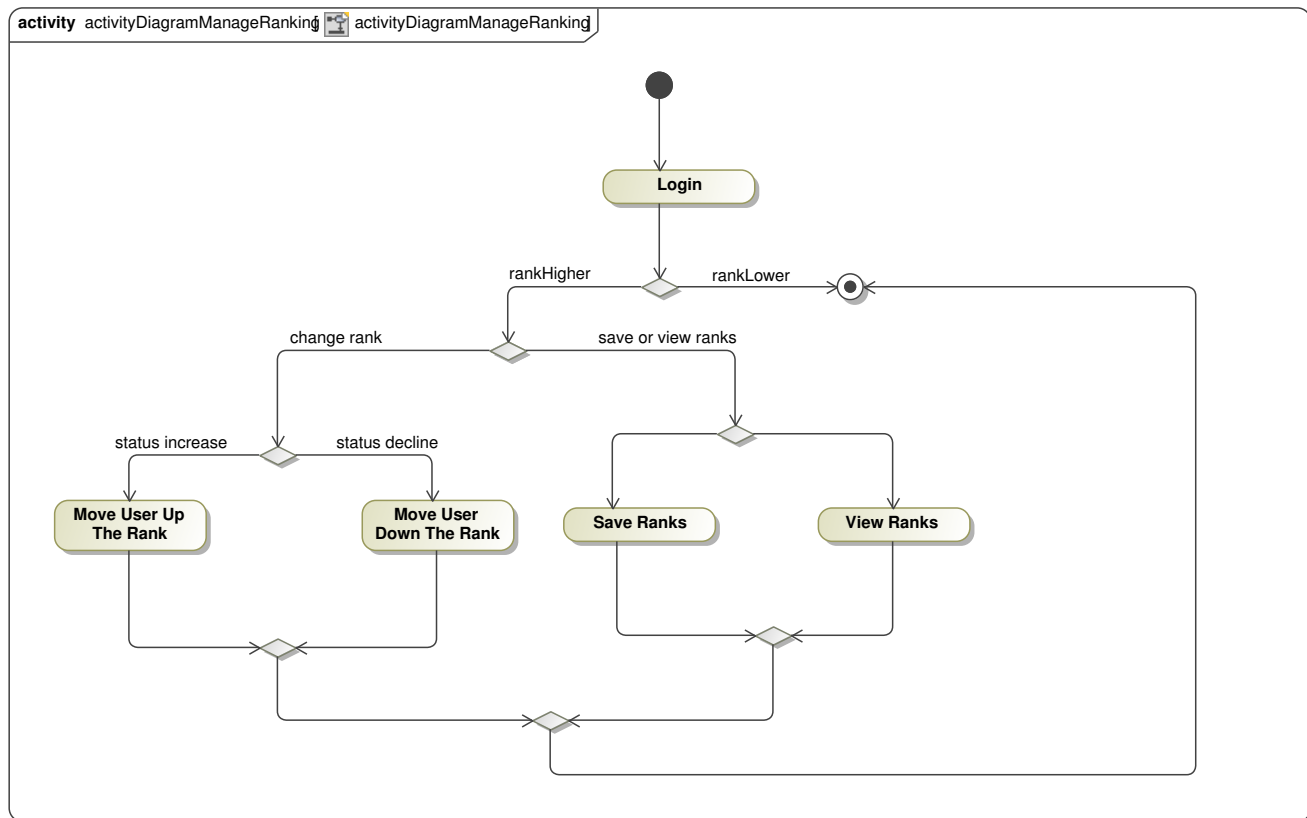
4.4.1 Rating Post



4.4.2 Mark Allocation



4.4.3 Manage Ranking



5 Social Tagging

5.1 Use case prioritization

5.1.1 Critical

- 5.1 createATtag
- 5.3 deleteTag

5.1.2 Important

- 5.2 searchTag

- 5.4 updateTag
- 5.5 viewTag

5.1.3 Nice-To-Have

- notify

5.2 Use case/Services contracts

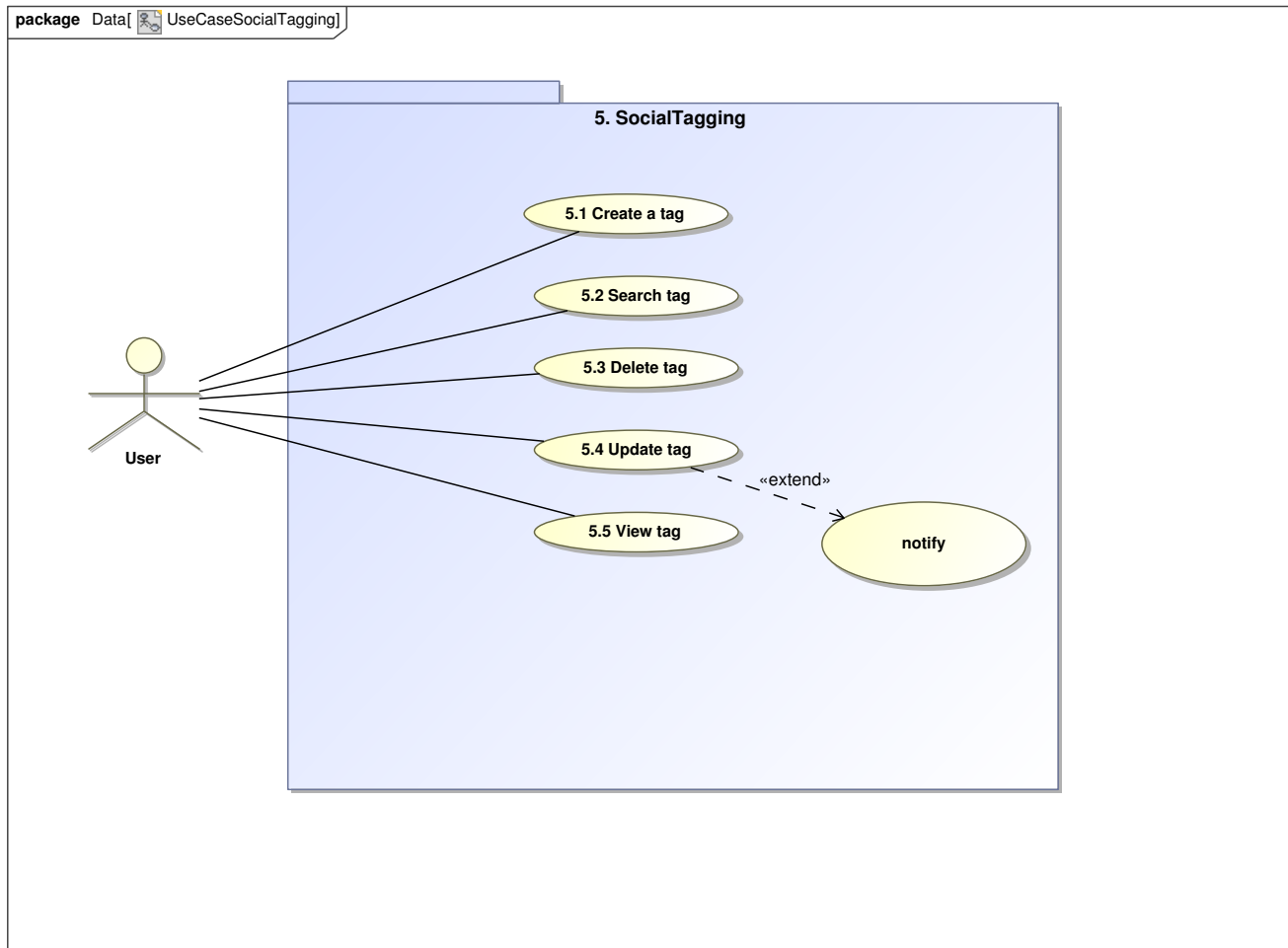
5.2.1 Pre-Conditions

- 5.1 User needs to be registered for the buzz space to be able to create tags

5.2.2 Post-Conditions

- 5.1 User should be able to access and or search for posts

5.3 Request and Results Data Structures



5.4 Process specifications

5.4.1 Social Tagging

