



UNIVERSITEIT VAN PRETORIA  
UNIVERSITY OF PRETORIA  
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

## FUNCTIONAL REQUIREMENTS COS 301 GROUP 2 B

Byron Dinkelmann u11057638

Johan van Rooyen u11205131

Mandla Mhlongo u29630135

Ryno Pierce u12003922

Sylvester Mpungane u11241617

Molefe Molefe u12260429

Taariq Ghoord u10132806

Timothy Snayers u13397134

Github

February 2015

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Vision</b>	<b>4</b>
<b>3</b>	<b>Background</b>	<b>4</b>
<b>4</b>	<b>Functional Requirements and application design</b>	<b>5</b>
4.1	Purpose . . . . .	5
4.2	Use case prioritization . . . . .	6
4.2.1	CRUD . . . . .	6
4.2.2	Message Restrictions . . . . .	6
4.2.3	Message Read Tracker . . . . .	6
4.2.4	Check Post For Plagiarism . . . . .	6
4.2.5	Report Post For Plagiarism . . . . .	6
4.2.6	Netiquette . . . . .	6
4.2.7	Text Decorating . . . . .	6
4.2.8	Statistical gathering . . . . .	7
4.3	Use case/Service contracts . . . . .	7
4.3.1	CRUD . . . . .	7
4.3.2	CRUD-Create . . . . .	8
4.3.3	CRUD-Read . . . . .	8
4.3.4	CRUD-Update . . . . .	8
4.3.5	CRUD-Delete . . . . .	9
4.3.6	Message Restrictions . . . . .	9
4.3.7	Message Read Tracker . . . . .	10
4.4	Functions To Apply Social Tagging . . . . .	11
4.5	Apply Self-Organization . . . . .	12
4.6	Pre-conditions . . . . .	12
4.6.1	Social Tagging Functions . . . . .	12
4.7	Self-Organization Based On Social Tagging . . . . .	12
4.8	Post-conditions . . . . .	12
4.8.1	Social Tagging Functions . . . . .	12
4.9	Self-Organization Based On Social Tagging . . . . .	12
4.10	Use Cases . . . . .	14
4.10.1	Check Post For Plagiarism . . . . .	15
4.10.2	Report Post For Plagiarism . . . . .	16

4.10.3	Netiquette . . . . .	17
4.10.4	Text Decorating . . . . .	18
4.10.5	Statistical Gathering . . . . .	19
4.11	Required functionality . . . . .	20
4.12	Process specifications . . . . .	20
4.13	Domain Model . . . . .	20
<b>5</b>	<b>Open Issues</b>	<b>20</b>

# 1 Introduction

This document outlines the functional requirements of "The Computer Science Education Didactic and Applications Research" (CSEDAR) registered research project called "The use of Online Discussion in Teaching" (TODT). This document describes the "Buzz" system to be developed and gives insight to how to achieve this implementation for those implementing it.

# 2 Vision

The research projects aim is to find ways to enhance and improve the learning of students with the use of online discussion. This online forum "Buzz" will become part of the Computer Science websites modules, creating an online space for discussion for each or where relevant. Where students, teaching assistants and lecturers can engage in activities related to learning the content of the specific module while applying game concepts to motivate students to increase the quality of their participation and consequently experience deeper learning of the course content.

# 3 Background

The reason for this research project is the problem of engaging an extremely large number of first year students within The University of Pretoria, specifically the Computer Science Department students. Currently available tools for discussion forums lead to the problems that are hampering positive engagement of teaching staff and students. These problems consist of inexperienced users, unorganized content and low levels of excitement. To combat these problems the "Buzz" system intends to have the basic functionality of online forums, have automated feedback on common mistakes, have a Game-like presentation and have automated structuring.

The University of Pretoria have the opportunity to teach the COS301 "Software Engineering" students while being able to improve current systems of online discussion which can be extended to benefit all students in the future.

The research project gives the students of COS301 "Software Engineering" an opportunity to learn from the experiences of designing, creating and

developing this system, which include learning to work with colleagues, creating documentation, simplifying and improving the communication of lectures and students through a structured forum, using online systems for group work and the basic process of how software is designed, created and developed in business to name but a few.

## **4 Functional Requirements and application design**

### **4.1 Purpose**

1. **CRUD:** To create posts on the buzz system so that users in the system can share relevant information with one another and communicate. To allow registered users and non registered users such as guests to view posts created by users of the buzz system. To allow users to update posts of which they control. To allow users to delete their own posts or to allow users with sufficient status level to delete other posts based on certain factors about that specific post.
2. **Message Restrictions:** To restrict users to length, content and positioning of posts based on the users status level.
3. **Message Tracker:** To mark posts for each registered user so that they can keep track of which posts they have read and which users have read certain posts.
4. **Check Post For Plagiarism:** To use a service provider to check if a post has been plagiarised.
5. **Report Post For Plagiarism:** To enable users to be able to report posts that they suspect is plagiarised (with a link to the original), because the system couldn't detect it.

6. Netiquette: To ensure that every post submitted conforms with the netiquette rules.
7. Text decorating functions: To allow privileged users to use advanced functions when typing posts.
8. Statistical gathering: To advance users to their next level for gamification concept.

## **4.2 Use case prioritization**

### **4.2.1 CRUD**

Critical

### **4.2.2 Message Restrictions**

Critical

### **4.2.3 Message Read Tracker**

Important

### **4.2.4 Check Post For Plagiarism**

Nice To Have

### **4.2.5 Report Post For Plagiarism**

Nice To Have

### **4.2.6 Netiquette**

Nice To Have

### **4.2.7 Text Decorating**

Nice To Have

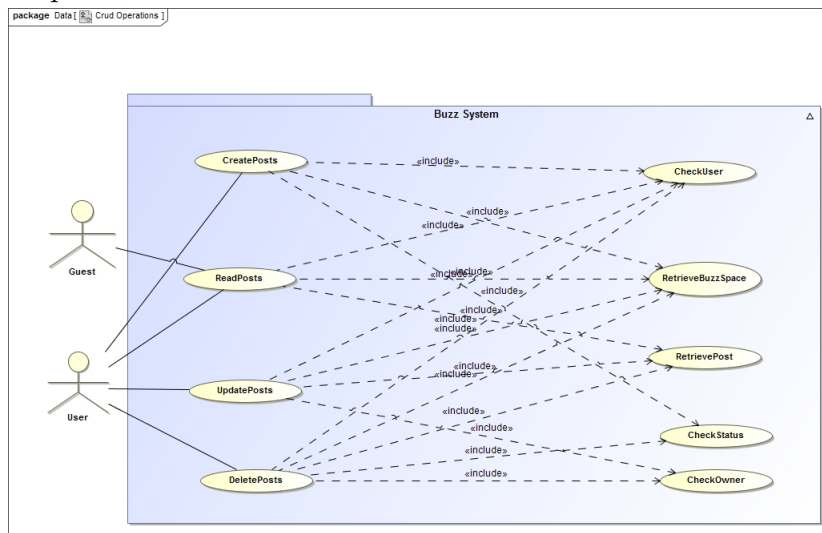
## 4.2.8 Statistical gathering

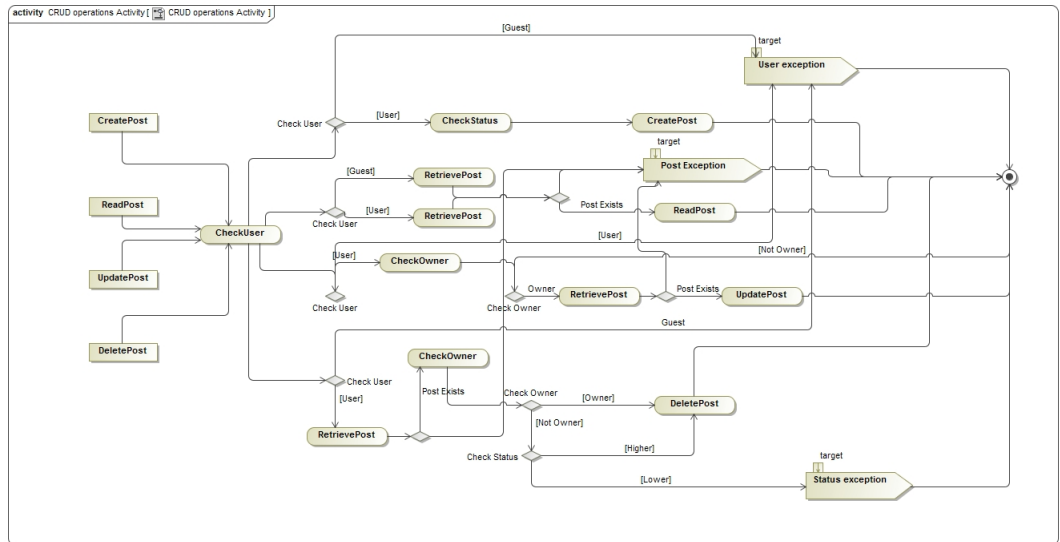
Critical

## 4.3 Use case/Service contracts

### 4.3.1 CRUD

1. Pre-conditions for CRUD: The buzz system needs to be available and a connection to the buzz system possible.
2. Request and Result Data Structures:





#### 4.3.2 CRUD-Create

1. Pre-conditions: For a user to create posts they need to be registered of the buzz system. The user must be logged in to the buzz system.
2. Post-conditions: Posts will succeed to be created and will appear in the buzz system where posted.

#### 4.3.3 CRUD-Read

1. Pre-conditions: Posts must be present to be read.
2. Post-conditions: Once a post has been viewed by a logged in registered user, the post should be marked as read for that specific user.

#### 4.3.4 CRUD-Update

1. Pre-conditions: For a user to update posts they need to be registered of the buzz system. The user must be logged in to the buzz system.



The user must be the owner of the post.

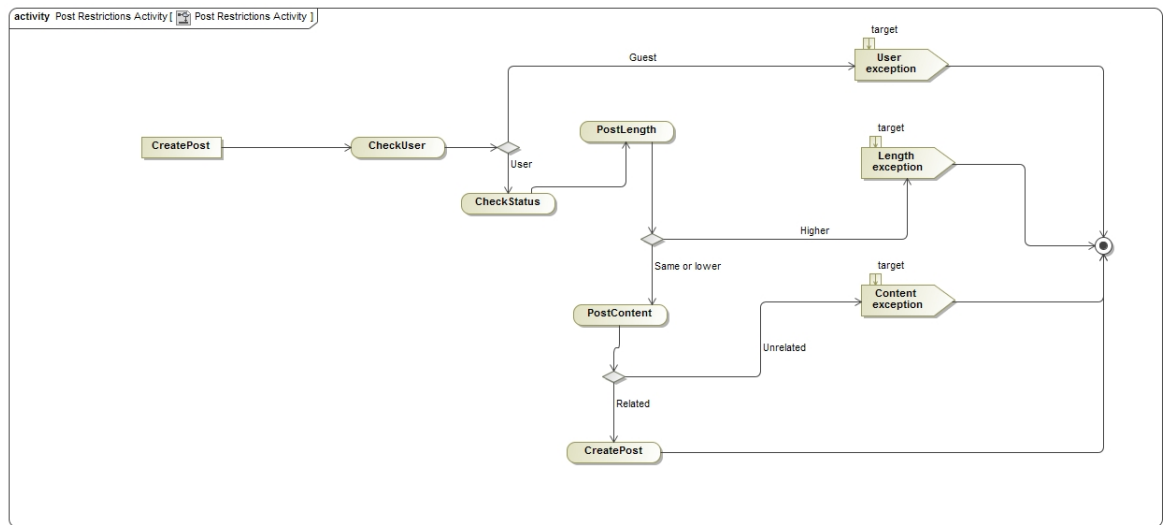
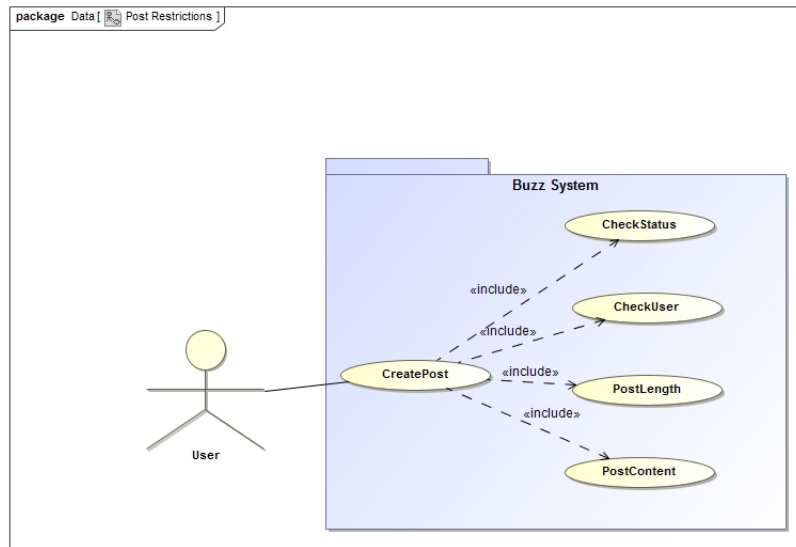
2. Post-conditions: Post will be updated and shown on the buzz system to be updated.

#### **4.3.5 CRUD-Delete**

1. Pre-conditions: For a user to delete posts they need to be registered of the buzz system. The user must be logged in to the buzz system. The user must be the owner of the post unless a user with status level "x" views the post to be unfit. The post needs to exist for it to be deleted.
2. Post-conditions: Post successfully deleted will be hidden from the other users and buzz system and archived. Posts will never be physically deleted from the system itself.

#### **4.3.6 Message Restrictions**

1. Pre-conditions: User must be registered on the buzz system. The buzz system needs to be available and a connection to the buzz system possible. The user must be logged in to the buzz system. User must be able to create posts. Length and content of posts must be specified by the buzz system of which the user is registered. Must have levels of posts.
2. Post-conditions: Post will be created and shown on the buzz system. Correct length and content of posts if buzz system is configured correctly.
3. Request and Result Data Structures:

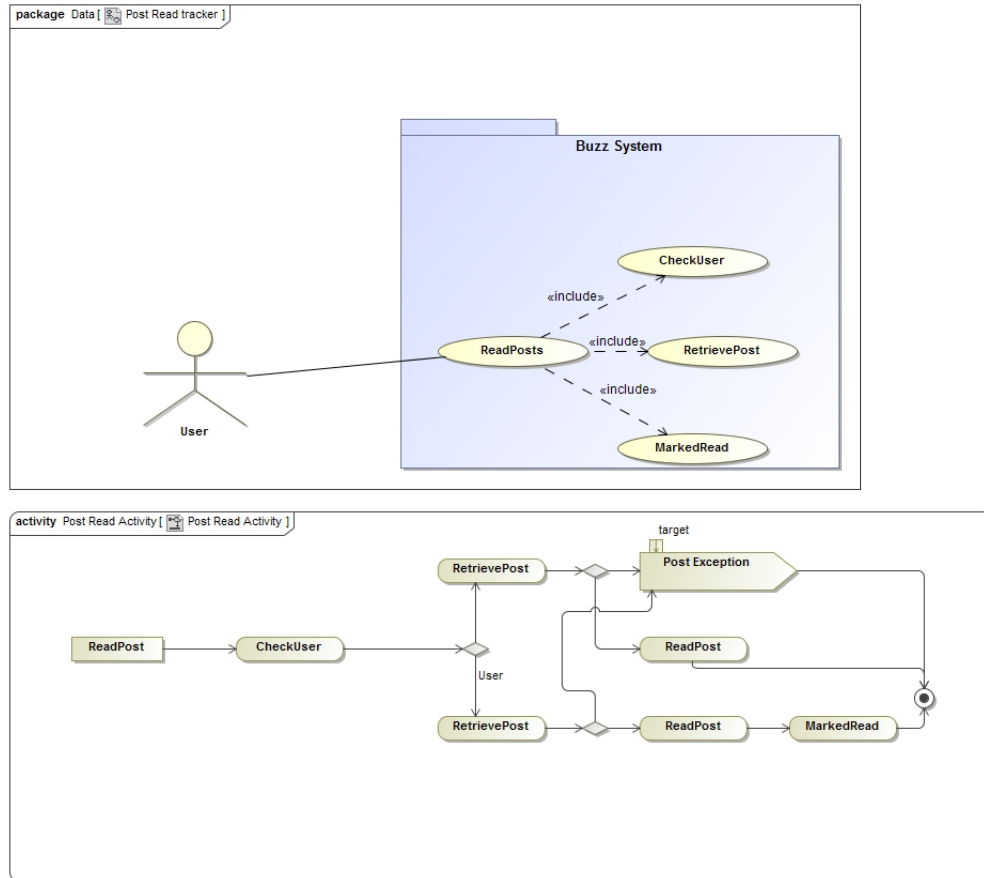


#### 4.3.7 Message Read Tracker

1. Pre-conditions: User must be registered on the buzz system. The user must be logged in to the buzz system. Messages must exist to be read, marked as unread and must be seen as read by other users.
2. Post-conditions: Post is marked as unread. Post is marked when read

and seen to be read on the buzz system.

### 3. Request and Result Data Structures:



## 4.4 Functions To Apply Social Tagging

1. The system will provide functions which through employing tags can provide flexible ways of using information that supports users with: Finding, Collecting, Storing, Organizing and Sharing Information.
2. Allow users to share their tags.
3. Allow users different views of the content. Either a personally structured view according to the user's tags or the default admin structure.

## **4.5 Apply Self-Organization**

1. The system will allow the user to organize their content themselves based on their social tags.
2. The system will allow the user to chose from different types of views either the base owns or public structure.

## **4.6 Pre-conditions**

### **4.6.1 Social Tagging Functions**

1. User is signed in and has default privileges to execute all social tagging functions functions.
2. User has already created tags before they can share them.
3. System allows for different views of the content.

## **4.7 Self-Organization Based On Social Tagging**

1. Have three different structures users can choose from for viewing.

## **4.8 Post-conditions**

### **4.8.1 Social Tagging Functions**

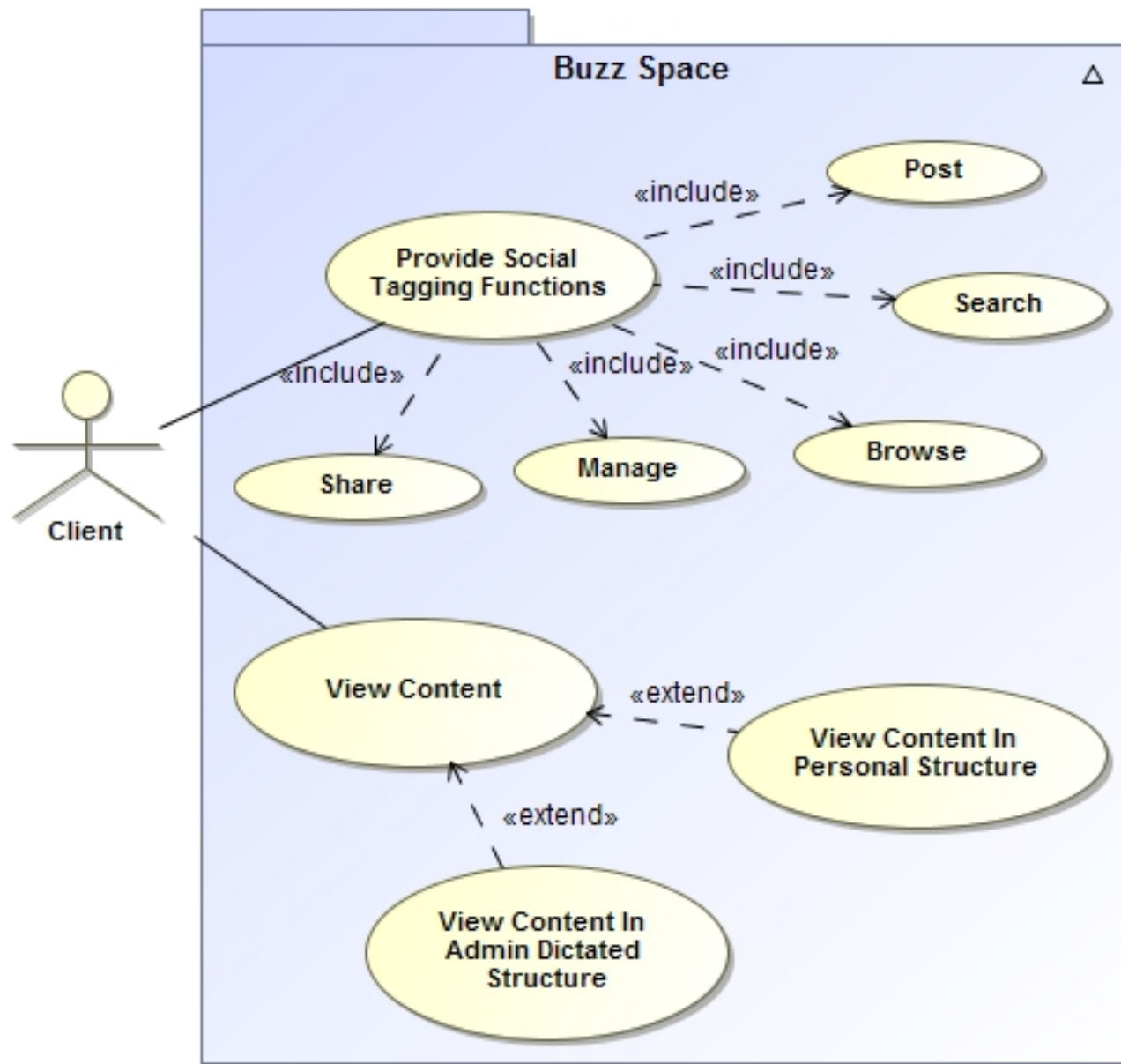
1. After the user has executed any of the functions the user state changes.
2. After the user selects their view structure, the structure must stay consistent until the user chooses otherwise.

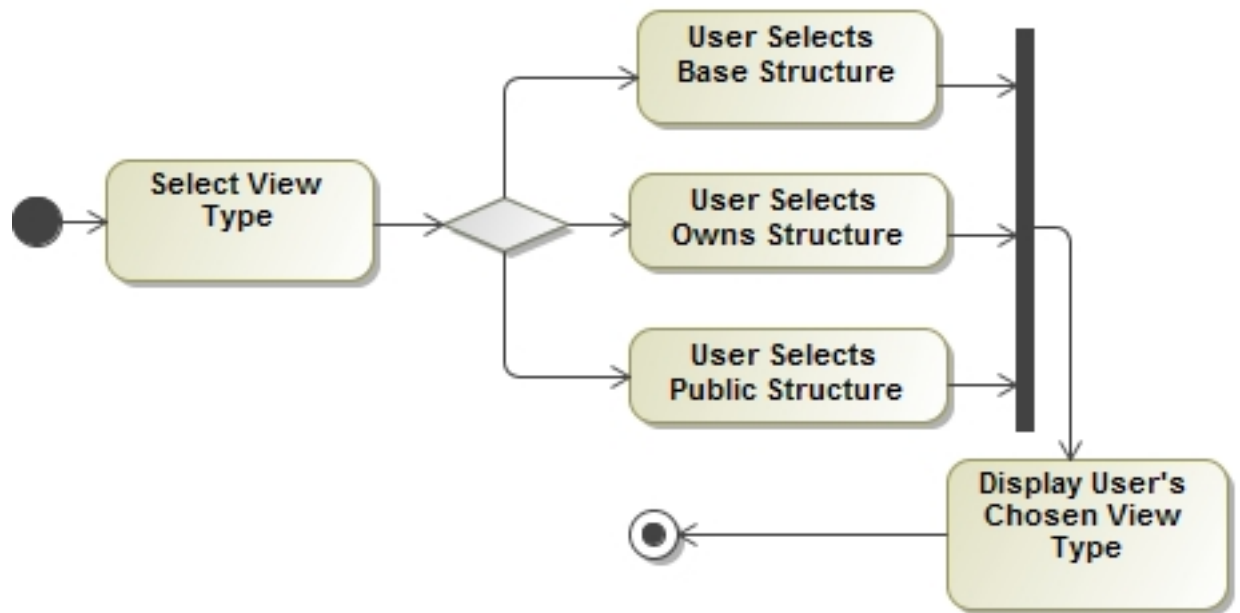
## **4.9 Self-Organization Based On Social Tagging**

1. After the user selects their view type, provide option to still be able to switch views.
2. Content must be displayed according to user's desire.



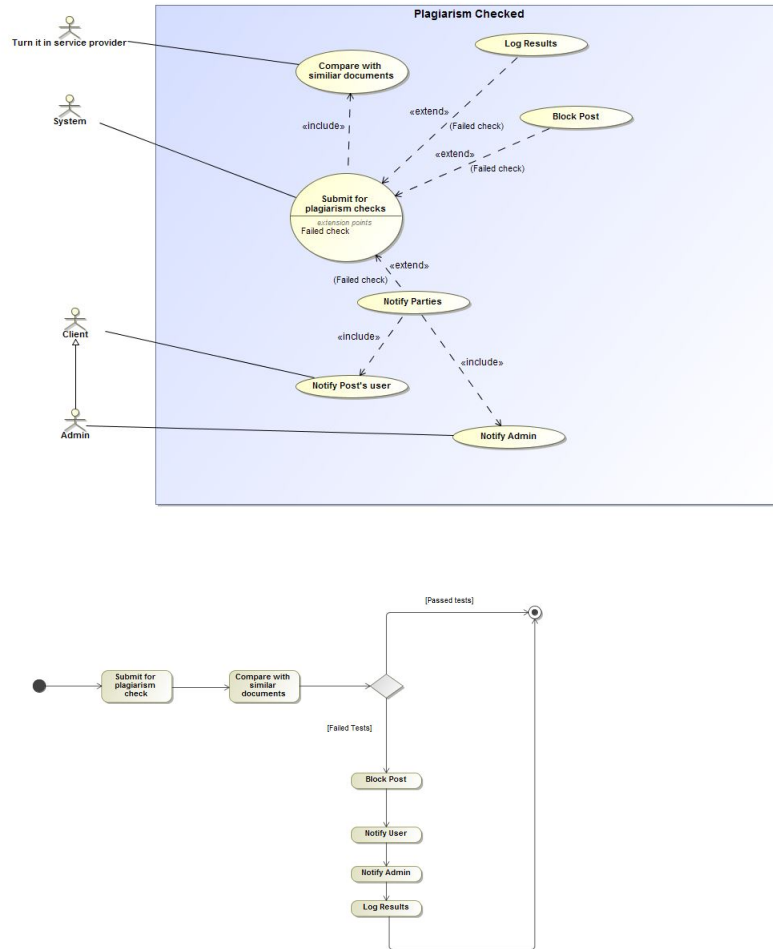
## 4.10 Use Cases





#### 4.10.1 Check Post For Plagiarism

1. Pre-conditions: A post must have been submit by a registered user.
2. Post-conditions: If the post passes the plagiarism test, the new post should be registered on the system. If the post has failed the plagiarism tests the post will be blocked, then the post's user and the module's administrator will be notified. Afterwords a log will be stored.
3. Request and Result Data Structures:



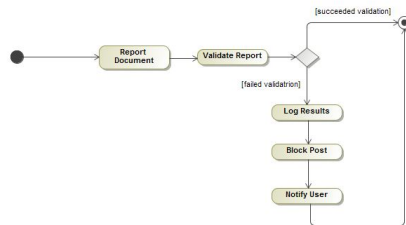
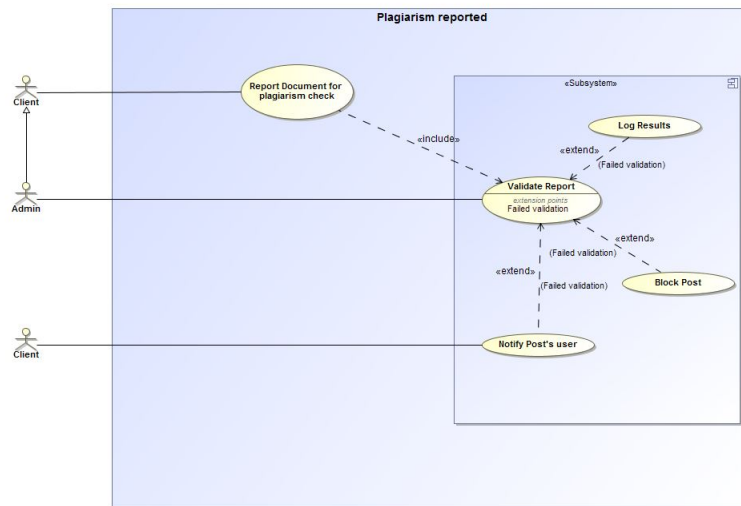
#### 4.10.2 Report Post For Plagiarism

1. Pre-conditions: A post must have been reported by a registered user that has a level X account.
2. Post-conditions: The post is sent to an administrator to follow up on the report. If the administrator finds the post not plagiarised, the post will be registered on the system. If it is found to be plagiarised the post will be blocked, then the post's user and the module's administrator



will be notified. Afterwards a log will be stored.

### 3. Request and Result Data Structures:

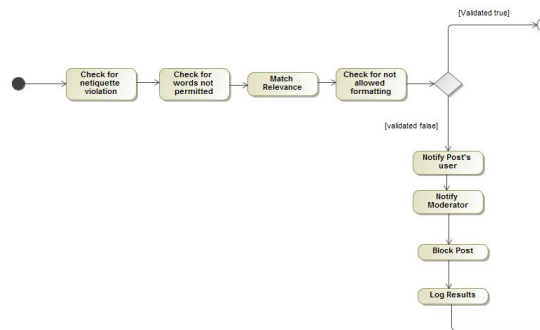
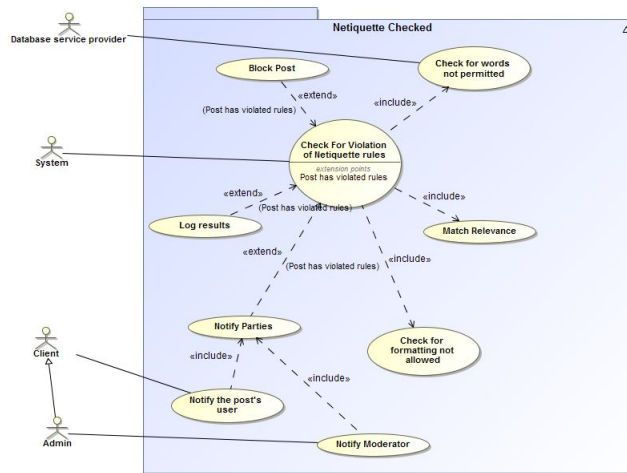


#### 4.10.3 Netiquette

1. Pre-conditions: A post must have been submit by a registered user.
2. If the post passes the netiquette test, the new post should be registered on the system. If the post has failed the netiquette tests the post will be blocked, then the post's user and the module's administrator will

be notified. Afterwards a log will be stored.

### 3. Request and Result Data Structures:

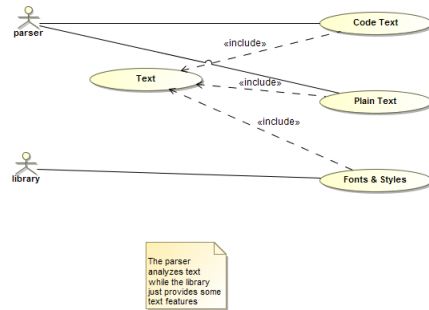


#### 4.10.4 Text Decorating

1. Pre-conditions: User must have required level.
2. Post-conditions: User can use the pretty text functionality that he/she qualifies for.

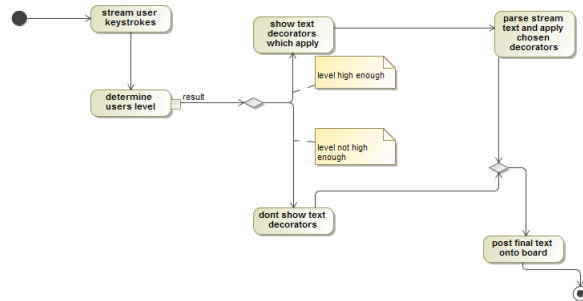
### 3. Request and Result Data Structures:

package Data [ Use Case: textDecorating ]



### 4. Request and Result Data Structures:

activity textDecoratorProcess [ textDecorator ]



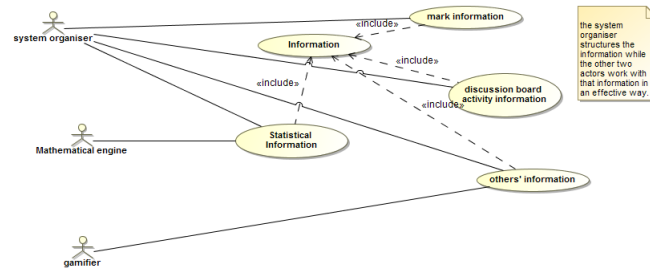
#### 4.10.5 Statistical Gathering

1. Pre-conditions: Database must be up and accessible.
2. Users want to level up to use privileged mechanisms
3. Post-conditions: Users who are students marks can be used to extract statistical information and these information can then be compared.

4. users have leveled up.

## 5. Request and Result Data Structures:

package GetInformation [ Use Case: Get Requests on information & data sorting ]



### 4.11 Required functionality

### 4.12 Process specifications

### 4.13 Domain Model

## 5 Open Issues