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

0.1 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.

0.2 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.

0.3 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 form 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.

0.4 Functional Requirements and application design

0.4.1 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.

   ¡¡¡¡¡¡¡ HEAD

0.4.2 Use case prioritization

CRUD

Critical

Message Restrictions

Critical

Message Read Tracker

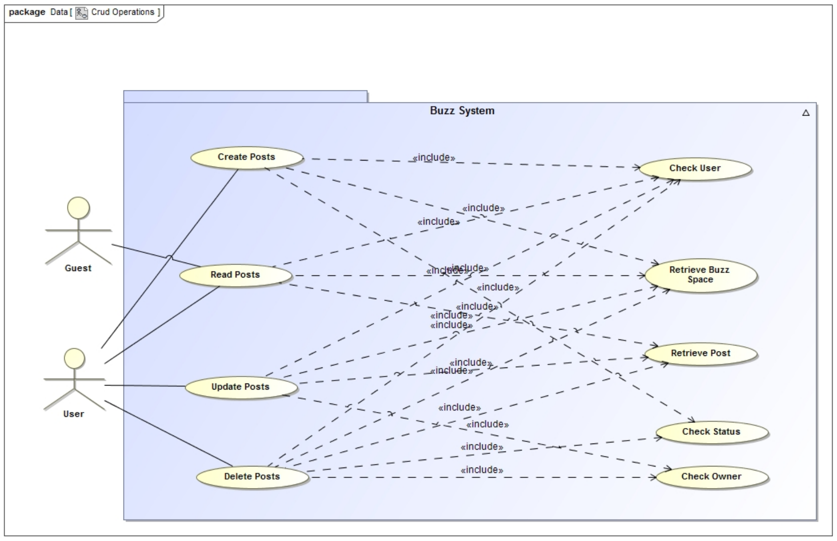
Important

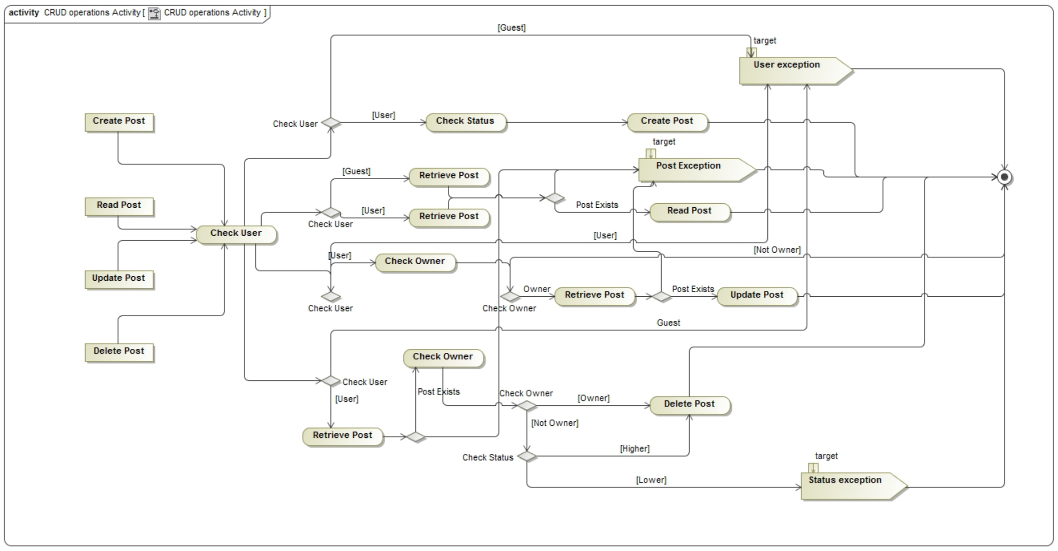
0.4.3 Use case/Service contracts

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:





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.

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.

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.

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.

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.

   =======

0.4.4 Use case prioritization

CRUD

Critical

Message Restrictions

Critical

Message Read Tracker

Important

0.4.5 Use case/Service contracts

CRUD

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

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.

3. Request and Result Data Structures:

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.

3. Request and Result Data Structures:

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.

3. Request and Result Data Structures:

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.

3. Request and Result Data Structures:

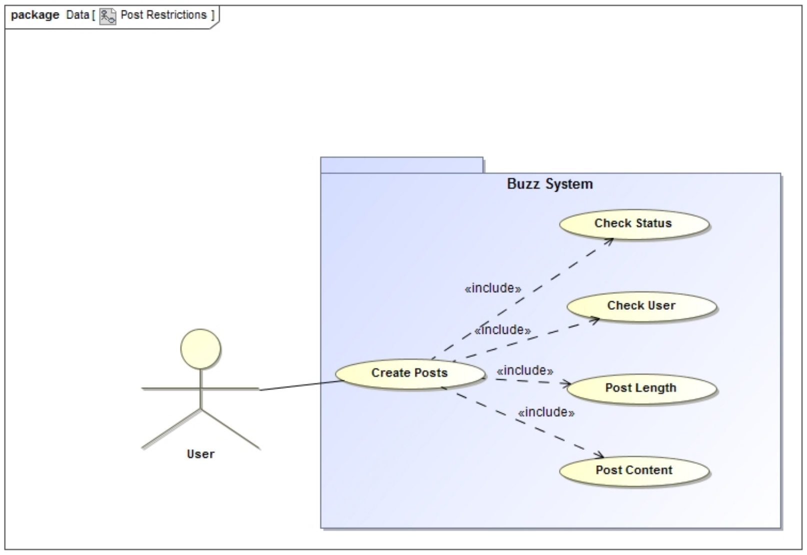
Message Restrictions

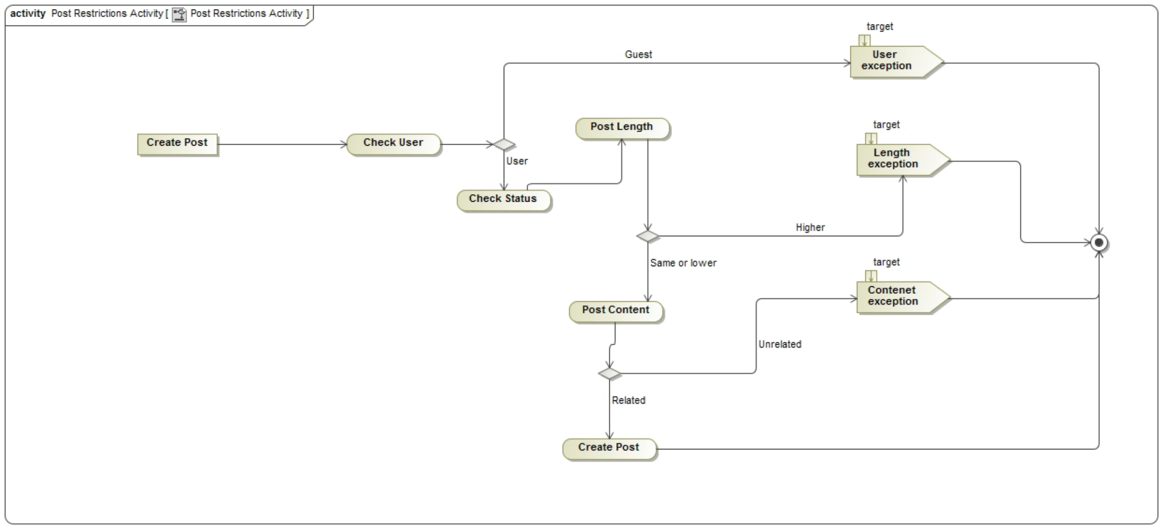
1. Pre-conditions: User must be registered on the buzz system. The buzz system needs to be available and a connection to the buzz system possible. The user must be logged in to the buzz system. User must be able to create posts. Length and content of posts must be specified by the buzz system of which the user is registered. Must have levels of posts.

   ¿¿¿¿¿¿¿ master

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:





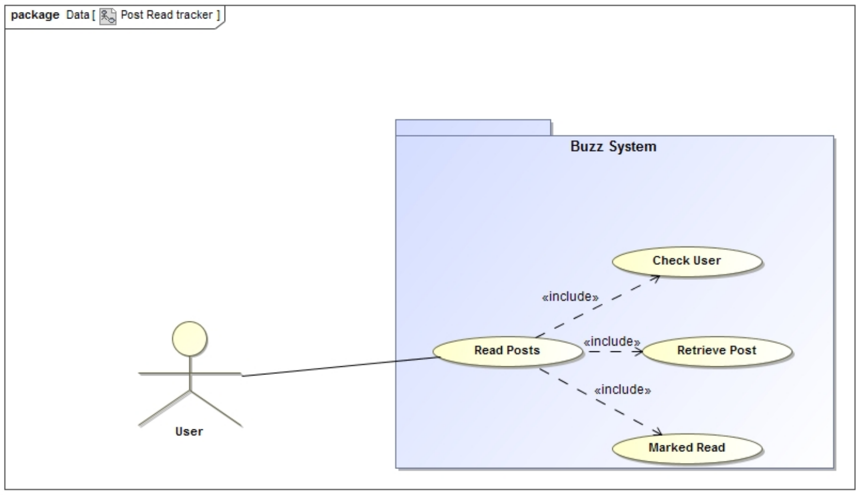
Message Read Tracker

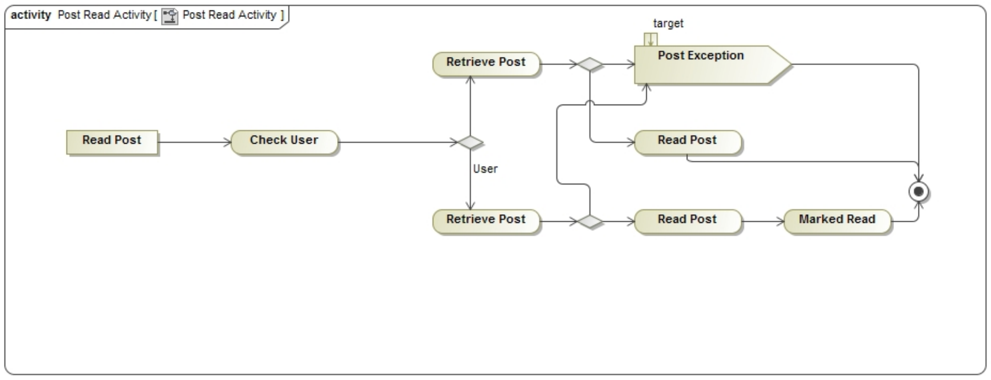
1. Pre-conditions: User must be registered on the buzz system. The user must be logged in to the buzz system. Messages must exist to be read, marked as unread and must be seen as read by other users.

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:





0.4.6 Required functionality

0.4.7 Process specifications

0.4.8 Domain Model

0.5 Open Issues

¡¡¡¡¡¡¡ HEAD

0.6 Searching and Filtering

0.6.1 Description:

Users will make use of such functionality to locate hard to find information like specific phrases within posts or category headings. The user will also use this functionality be able to sieve through the forum to track down other users and/or topics.

0.6.2 Actors:

Active forum members, administrators, evaluators will be the most common users of this functionality. Guests and other passive forum members will be the uncommon users.

0.6.3 Preconditions:

Guests are able to search/filter but a limited search results would be imposed depending on the search criteria. Higher level users (forum members and ranked forum members) would have less restricted search capability based on the user ranking system. Super users (Administrators and Evaluators).

0.6.4 Basic Flow:

The actor will type in the search term and select advanced criteria (optional, less restricted based on actor rank). The search is processed, this involves database querying and third party search engines. Results displayed in a categorical fashion. Categories (posts, subjects, topics, members etc.) from the results are scanned by the actor. A Result selection leads the actor directly to the selected results origin. A search rating mechanism would occasionally ask for a search rating.

0.6.5 Alternative Flows:

For uncommon users the search is less intensive and restricted thus shallow search processing will occur. Security questions would have to be answered occasionally. The browser checked cached search results. The results would still be presented in a categorical form, but would obviously omit restricted results.

0.6.6 Exception Flows:

Unintelligible or foreign language search would be severely limited. However technical jargon in search terms will be accommodated to an extent. Search results with restricted content would require user login to display.

0.6.7 Post Conditions:

Search results are directly linked to their displayed source. A search/filter with no hits will display related content and help for bettering the search.

0.6.8 User Case Diagram

`SearchAndFilter.png`

0.7 Post Evaluation and Voting

0.7.1 Description:

Users would occasionally be approached to assess other users and their posts on the forum, users can also willingly rate posts or topics they have read through. Evaluation criteria would include the integrity of the information in posts and the posters rank and forum activity.

0.7.2 Actors:

Only forum members can participate in post evaluation and voting. There are possible exceptions like security cleared +1 members (temporary members linked to a current forum member) or forum guests that are identifiable experts on a topic/subject.

0.7.3 Preconditions:

The actor must be a forum member or have qualified to vote based on the above mentioned exceptions. The voter must be of a certain rank to access advanced voting functionality. The voter must be clear of any bans.

0.7.4 Basic Flow:

The user will click on the voting mechanism linked to the post. The voting window will appear and the voters profile and history will be checked. The voting will then commence and allow for more or less voting capabilities based on the voters details. The vote will be submitted and the optional evaluation (for ranked voters only) will be sent to the post owner and displayed alongside the post.

0.7.5 Alternative Flows:

The exception user will place a vote on a post. The linked forum member will be informed of this action. The vote and evaluation will be submitted upon the forum members approval.

0.7.6 Exception Flows:

The user has been banned from voting and is denied voting and evaluation rights until he/she obtains an acceptable rank or is unbanned by the forum administrator.

0.7.7 Post Conditions:

The voter and vote receivers profiles are updated and rank re-evaluated. The vote/evaluation on the post is updated instantly.

0.7.8 User Case Diagram

`PostVoteAndEvaluation.png`

0.7.9 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.

0.7.10 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.

0.7.11 Pre-conditions

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.

0.7.12 Self-Organization Based On Social Tagging

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

0.7.13 Post-conditions

Social Tagging Functions

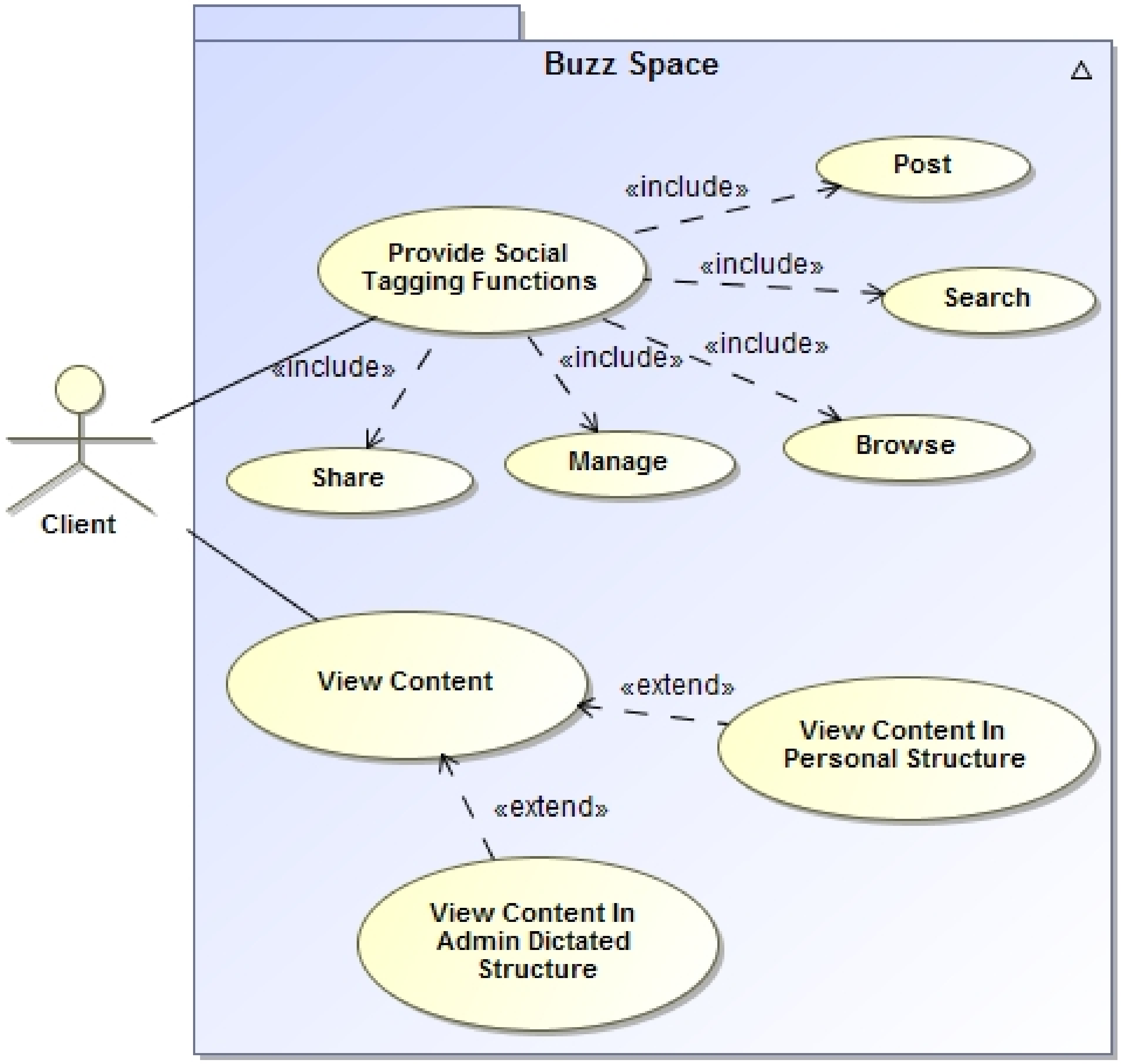
1. After the user has executed any of the functions the user state changes.
2. After the user selects their view structure, the structure must stay consistent until the user chooses otherwise.

0.7.14 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.

0.7.15 Use Cases



=======
¿¿¿¿¿¿¿ d3fa091506bd680bbda206b06f20604b39c40530