Unplagged Final Report

Building the Plagiarism Detection Workbench

Term paper for the master project II
Mentoring Teacher: Prof. Dr. Debora Weber-Wulff

Department Economics II
HTW Berlin – University of Applied Sciences

Elsa Mahari (s0534556) <Elsa.Mahari@gmx.de>
Dominik Horb (s0534217) <horb@htw-berlin.de>
Tien Nguyen (s0512510) <s0512510@htw-berlin.de>
Benjamin Oertel (s0522720) <contact@benjaminoertel.com>
Heiko Stammel (s0534218) <heiko.stammel@googlemail.com>

Contents

1. Features **3**

1.1. Notifications and Comments 3

1.1.1. Recent activity stream 3

1.1.2. Comments plugin 4

1.2. Fragments 6

1.2.1. Creating a fragment 7

1.2.2. Rating a fragment 7

1.3. Barcode 7

1.4. User avatar 7

1.4.1. Avatar cropping 7

1.5. Automatic Plagiarism Detection Webservice 8

1.5.1. PlagAware 8

1.6. Permission and role management 8

1.6.1. Collaborators 8

2. Showtime **9**

3. Summary and Outlook **10**

A. Meetings **11**

List of Figures

1.1. Single activity in the activity stream 4
1.2. Creating a comment on a resource 5

Introduction and Overview

Nine month after the start of development the time of the masters project Unplagged at the HTW has now come to an end for the initial team members. Hopefully though, this isn't the end of the lifecycle of the software Unplagged that was created during that stretch. Perhaps some of us will continue working on it afterwards or maybe we are even able to attract some other developers to help this open source project if we are lucky. It was a good experience after all and the foundation feels good and strong.

Although there are no direct connections, this document is at least in some ways a sequel to the Developers Manual, that was written after the first semester of this project. In some instances it assumes prior knowledge of things already described there, so that it doesn't unnecessarily need to be repeated here. All in all, we are going to give a more technical insight into the development process then in the report before, which concentrated on the project management and development environment aspects. We will also try to critically analyze the problems that were faced and the mistakes that were made over the course of those two semesters.

For all the team members, the development of Unplagged up to this point was one of the biggest projects that we ever had to start and needed to bring to a state of usable maturity on our own. Most of us came across big or even huge projects at work some time, but the process until the first release was mostly long over and replaced by bugfixing and the eventual feature release. So those famous team building phases forming, storming, norming and perfoming that Bruce Tuckman described(**?**), were something we only could see here in a full scale.

During our time of study, it was also one of the first projects, where we couldn't really envision in the beginning, how the end result would need to look like.

With other projects before, there were often enough requirements similar to some parts of software systems we already knew and just needed to adapt to the current purpose. This mostly gave us some hints of the feasibility and the best approach to solve a problem. With Unplagged this was somehow different.

First of all, the idea for the project came from outside the team with very short time to research beforehand what it would be about and second, we didn't have any experience with plagiarism research. So we started out with one abstract idea of the system outside of the development team, turned that into a different one for every team member and eventually had to break it down to one software system.

Luckily enough, the choices of technologies, architecture or even programming languages we made in the beginning, didn't came to haunt us in the long run. We eventually figured, that it would be some mix

1. Features

1.1. Notifications and Comments

The following section describes an important part for registered users, the notification system and the opportunity for adding comments basically on any resource in the system.

1.1.1. Recent activity stream

It displays the most recent events in the portal in the order they happened. Each activity (figure 1.1), one line in the stream, consists of 3 parts: meta information about the activity itself, information about the initiator and comments. The content of each of these parts is being determined automatically, when a new notification is being persisted to the database.

Besides the initiator information and comments, each activity has an own title, description and icon. These meta information texts can be edited in the portal in the 'Administration' > 'Acions' section. Although it is not possible to remove them, because the action references are being used hard-coded within the system. That means a set of notifications is provided and can be ouput anywhere in the workflow.

The following code snippet shows, how to create a new notification when a new automatic plagiarism detection report was created. The static method takes in 3 parameters: a unique name for the notification type, the content object related to the notification and a user object, as the third parameter. A list of all available notification types can either

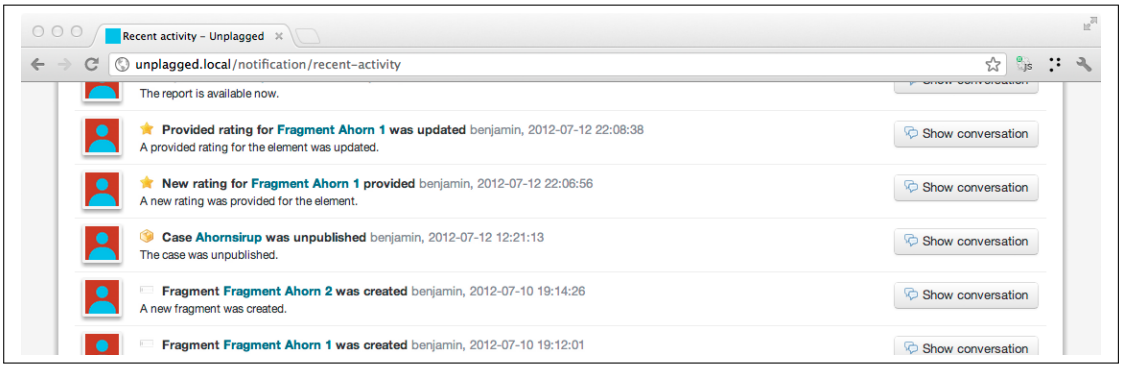

Figure 1.1.: Single activity in the activity stream

be found in the previousely mentioned actions section or in the scripts/build/initdb.php file, where all notifcation types are being declared.

Listing 1.1: Creating a notification for a created report

```
1 Unplagged_Helper::notify("detection_report_created", $report,
      $report->getUser());
```

Unplagged does have an extensive role and permission management. Therefore the grant on each resource is being verified, before it is being displayed in the activity stream. Usually the resource related to the notification is the resource where the permission check is being performed on. Although in some cases, e.g. when rating a fragment, the resource will be the rating itself, but the permission check is done on the fragment. In this case the notify-method is being called with a fourth, optional parameter, another resource, in this case the fragment. When the user has access on the fragment, all ratings can be accessed as well, automatically.

1.1.2. Comments plugin

Comments are simply a small text related to a specific user and a resource. They can be used to share ideas on an object collaboratively. The most prominent part at Unplagged, where comments are being used, is the activity stream. A comment can be added by any user having access to the notification.

Figure 1.2.: Creating a comment on a resource

For providing a better workflow to the user, the comments can be refreshed and added in place. That means, the position where the user scrolled to in the browser does not get affected. The in-place refreshing is being realized through AJAX. The comments container is being loaded empty and displaying a small loading image only. Not before the user clicks the 'show conversation' button, the comments are being fetched through a post request to the server. Whenever the result is being fetched completely, the spinner graphic is being hidden and the comments are being appended. The parsing of the comments markup is being done in Javascript as well. So the server requests are kept small and the server can return JSON only without any HTML.

Listing 1.2: Refreshing the comments of a resource

```
1   target.show();
2       conversation.hide();
3       loading.slideDown(800, function() {
4         // get the whole conversation
5         $.post('/notification/conversation', {
6           'source': sourceId
7         }, function(data) {
8           if(!data.errorcode) {
9             conversation.html("");
10            $.each(data, function(index, value) {
11              conversation.append(renderConversation(value));
12            });
13            loading.slideUp(800, function() {
14              conversation.slideDown(300);
15            });
16          } else {
17            conversation.html('<div class="comment">' + data.
                  message + '</div>');
```

```
18              loading.slideUp(800, function() {
19                conversation.slideDown(300);
20              });
21            }
22          }, "json");
```

Listing 1.3: Creating the markup of a single comment

```
1  function renderConversation(data, target) {
2      var tpl;
3
4      switch(data.type) {
5        case 'comment':
6          tpl = '<div class="comment">' +
7          '<div class="image"><img class="avatar-small" src="' +
               data.author.avatar + '" /></div>' +
8          '<div class="details">' +
9          '<div class="title"><b>' + data.author.username + '</b
               > ' + data.text +
10         ' <span class="date">' + data.created.humanTiming +
               '</span>' +
11         '</div>' +
12         '</div>' +
13         '</div>';
14         break;
15     }
16     if(!target) {
17       return tpl;
18     } else {
19       target.append(tpl);
20     }
21   }
```

1.2. Fragments

@benjamin

1.2.1. Creating a fragment

@benjamin

the-old-way

two-column view

1.2.2. Rating a fragment

@benjamin

1.3. Barcode

@benjamin

1.4. User avatar

@elsa

1.4.1. Avatar cropping

@benjamin

1.5. Automatic Plagiarism Detection Webservice

@benjamin

1.5.1. PlagAware

@benjamin

1.6. Permission and role management

@dominik types of permission, types of roles

1.6.1. Collaborators

@benjamin

new role created from global case role

2. Showtime

3. Summary and Outlook

summary and outlook

A. Meetings

The following tables show the minutes of most of the team meetings.

Bibliography

[Google 2012] Google: *News Search Interest: plagiat*. http://www.google.com/insights/search/#q=Plagiat&geo=DE&date=1%2F2011%2013m&gprop=news&cmpt=q, 2012. – [Online; accessed 07-February-2012]