Unplagged Developers Manual

Building the Plagiarism Detection Cockpit

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

Department Economics II
HTW Berlin – University of Applied Sciences

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

Contents

List of Figures

Introduction

After Minister Guttenberg had to resign, because of the plagiarisms found in his doctoral thesis, the big media coverage and interest in plagiarism in Germany has very much subsided (Google, 2012). However, the initial idea for the creation of the "Unplagged" project, whose development approach will be described here, can be found in this very case of plagiarism. Related to it were the formation of the GuttenPlag and it's descendent VroniPlag. Both are Wiki-based communities that are collaboratively discovering and collecting plagiarism in their respective cases and are kind of the role models for the way the Unplagged system is developed.

The project idea and context were provided by our professor Dr. Debora Weber-Wulff and the two-term master project, every media informatics student at the HTW-Berlin has to take. Professor Weber-Wulff is a well known expert in Germany on the topic of plagiarism. As she has also done research in this field for over ten years and is actively involved in the VroniPlag community under her synonym "WiseWoman"(Spiegel-Online, 2011), she came up with the idea to build a dedicated system — a "Plagiarism Detection Cockpit"(Weber-Wulff, 2011) — that is modeled after the experiences that were made with the workflow used in VroniPlag and GuttenPlag.

So, to put it in a catchy marketing phrase, here is what Unplagged aims to become:

> **Unplagged is a simple, web-based, collaborative system to help discover, collect and document plagiarism in scientific papers.**

To make things a bit more conceivable, we also often refer to it as a mixture of a very specialised text editor, with a focus on comparing texts and marking passages and a modern project management tool like Redmine or JIRA, to manage the collaborative

aspects of the system. The big distinction we make to other plagiarism software on the market is, that the approach is not to autodetect plagiarism, but focused on aiding the workflow of the users while searching for plagiarized fragments inside a scientific paper, a homework assignment or any other kind of probable textual plagiarism.

This present document will be the handbook that gets you started if you are interested in helping us with the development of this open source project, which is licensed under the GNU GPLv3.

Chapter Overview

One of the biggest problems we faced at the start was, that none of the team members had written a longer scientific text than a bachelors thesis and therefore the experience we got with actual scientific writing was very limited and very specific to the field of computer science. We understand the ethical problems, that come with the betrayal of good scientific practice of plagiators, but we simply can not relate easily to the amount of work that has to be put into a PhD., or be as passionate about plagiarism as Prof. Weber-Wulff always is, because we never experienced it ourselves.

That is why we had a lot of catching up to do on the most important history behind VroniPlag, the different types of plagiarism, different citation styles and the research Prof. Weber-Wulff and others had already done on systems that try to help finding plagiarism. Chapter 1, A Plagiarism Primer, will give a brief overview of the most important topics to get you up to speed with the domain of the software, if you are not already familiar with it.

Chapter 2, Project Workflow and Requirements, will be the place, where the development process is described and a collection and description of the parts of the system, that already exist or that we identified as necessary parts of Unplagged will be given. As the system is developed with an agile project management style, this will be done primarily based on the product backlog.

If you know all those things already and simply want to get started working and coding,

you should probably jump to 3, Developing Unplagged. This chapter will give the technical insights into the system, the basic installation steps and all necessary tools for you as a developer.

Conventions

To markup important words in the text, the following typographical conventions are used:

Italic
: First used technical terms

`Constant Width`
: Programm code, file names, paths

`Bold Constant Width`
: Variables that have to be changed by the user

1. A Plagiarism Primer

In this chapter we will give a brief overview of the most important topics to get you up to speed with the domain of the software, if you are not already familiar with it.

1.1. Plagiarism definition

1.2. Basic Classification of Plagiarisms

In this part, we will try to cover common classifications of plagiarism. First, we want to figure out the purpose of the classification.

There are many reasons explaining why people plagiarize. Generally those include, that they didn't have the time, energy or the ability to do the work by themselves or that they try to steal other's work on purpose with the hope that others will not recognize it. This kind of plagiarism is done intentionally and named *deliberate plagiarism*(UEfAP, 2012).

Another kind is *accidental plagiarism*. This occurs when texts from some sources are copied or rephrased, but no reference is given. The reason is that the writer didn't know that it is a plagiarism because of "carelessness or lack of skill"(UEfAP, 2012) while writing.

Anyway, it is important to know, that working with the source carelessly may cause plagiarism. Classification of plagiarism is also a good way to help distinguish typical types of plagiarism, so that people are aware and able to reference the sources carefully and to avoid plagiarizing.

Classification of plagiarism is also good for professors or the plagiarism detection community such as VroniPlag, because it gives them a common terminology, which enables them to communicate faster and more efficiently. It also helps them to realize what category and how many percent the text is plagiarized if there is any suspicion, so that statistical data can be created.

So the classification is about the way to detect plagiarism while conducting a citation. But what is a citation, and what kinds of citation are there? Understanding of citations and the way to cite is an important thing in order to avoid and detect plagiarism.

1.2.1. Definition of Citation

> "A citation is a credit or reference to another document or source"
> (Wikipedia, 2011)

According to this definition, citation is your reference to a source of information, which was generated by someone and should be indicated properly when this source is used in your work.

Based on Wiredprof (2010), there are 3 forms of citation:

Direct quotation:

That is an exact word-to-word copy from one source. In this case, in order to cite you have to mark the text with quotation marks and indicate a parenthetical referencing, which includes the book's name and the page number where you found the source.

Paraphrase:

That is your own explanation of someones idea. Many people think it is not a plagiarism because the text is written with your own words. But it is still a plagiarism because basically that is not your idea or your opinion, but the author's himself. In this case of citation, some keywords are still kept in the work. Therefore the parenthetical reference must be given.

Summary:

That is the same as paraphrase, but this kind of citation is likely a summary of the text. In this case you have to give the parenthetical referencing as well.

Now we can see that citation has a lot of forms. You are free to use the source but make sure to cite sources properly or you are applying a plagiarism.

After understanding what citation is and its form, now we can try to classify the plagiarism. The criteria which we choose for classification are based on the VroniPlag's plagiarism categories, because this project is implemented based on the workflow of the VroniPlag

community.

By VroniPlag, it does not matter what citation standard system is used in the work, but it is important to know how the citation is documented and if the reference is given properly.

VroniPlag's classification of plagiarism

According to VroniPlag there are the following plagiarism categories. The categories are originally written in German, and the translation or the similar category in English is written right after the german expression.

Komplettplagiat/Copy & Paste:

The name of this category already indicates how the text is plagiarized. The plagiariser just copies from the source exactly word-by-word and does not leave any proper reference to the source intentionally. Some is too lazy so that they just copy existing mistakes, formats etc... without checking, which later can become a proof of plagiarism.

The category is differentiated by the way of conducting a citation. Source is not cited: the original text is completely copied but the source reference is not given intentionally. Source is cited but not completely: reference is given but not correctly.

Verschleierung/Paraphrasing:

The texts from different sources are rephrased and mixed together. The plagiariser try to hide his stealing by changing some word orders, replacing words with synonyms ... The source is not given with the hope, that the text is considerably generated by the plagiariser himself and therefore the plagiarism will not be detected.

Übersetzungsplagiat/translations:

This kind of plagiarism occurs when the sources in foreign languages are translated. The plagiariser pretend that this translated text is his own work. Sources are oft not given properly. This is a well-liked way in researching because there are many translation tools which can be found easily in Internet such as Babelfish, Google

Translate ... and it is not easy to detect the plagiarism with existing plagiarism detection tools.

Alibi-Fußnote/The forgotten Footnote:

in this category, the source is cited but the real location of the source is hidden.

Bauernopfer und. Verschärftes Bauernopfer:

The source reference is embedded in footnote part but it redirects to another text part of the source which has no relation with the plagiarized text.

Other categories

Halbsatzflickerei/The Labor of Laziness:

The plagiariser takes sentences, or just parts of sentence from different sources, tries to reword them and mixes all so that they look fit together. Source reference is not or not correctly given.

Shake & Paste:

The sentence or paragraph from one source is mixed with one from another source. This plagiarism can be detected by changes of writing styles. Source reference is also not given properly.

1.3. How to detect plagiarism

Plagiarism detection means a lot of effort and hard work, because a lot of documents, books and papers must be found, scanned and compared line for line. Often it is really complicated to find available sources in libraries or scientific database systems and plenty of pages must be copied by hand. During the last ten years a lot of software companies came into market to automatize this process and developed algorithms to find plagiarisms automatically. These systems are called "Plagiarism detection systems (PDS) ". There are two different approaches for plagiarism detection systems to identify plagiarism in text documents:

1. **Corpus based analysis**
   means to "compare suspicious documents against a set of potential original documents" PAN'07 to find similar text passages.

2. **Intrinsic analysis**
   "identifies potentially plagiarized passages by analyzing the suspicious document with respekt to changes in writing style". PAN'07

These computer assisted detection systems alone are not appropriate to find all plagiarisms without human judgement.

We found out that the most practical way is to combine both approaches - the first step is to use the computer-based detection to find similarities between the suspicious documents and the original papers. The second step is to examine the results, validate them and continue the search in a deeper level. (see figure 4 stage)

(Culwin, Fintan; Lancaster, Thomas (2001), "Plagiarism issues for higher education", Vine 31 (2): 36-41, doi:10.1108/03055720010804005)

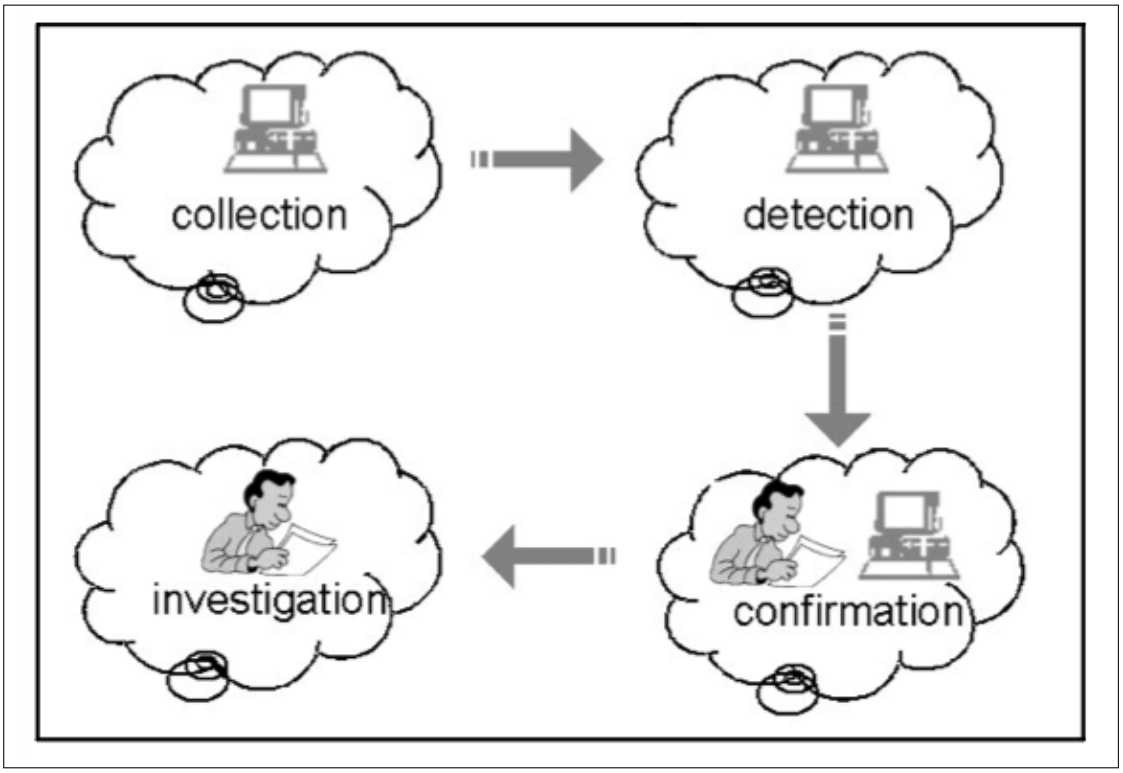

Figure 1.1.: 4-stage plagiarism detection process

1.3.1. Commercial software systems

Name	No free access	Free version available	Free demo available	Website
Plagaware		x		www.plagaware.de
Turnitin			x	www.turnitin.com
Ephorus			x	www.ephorus.com
Plagscan	x			www.plagscan.com
Urkund			x	www.urkund.com

Figure 1.2.: Overview of Commercial Detection Systems

Several computer companys developed commercial software systems to faciliate the detection of plagiarism. They offer different terms of pricing and online as a service or offline programs. The software system compare digital content from the internet or different types of databases. They seek for similarities, report suspicious parts and try to answer the question if the present text is a plagiarism or not.

This chapter covers parts of the results of the big "Plagiarism Detection System Test 2010". The following benchmarking was done by Prof. Debora Weber-Wulff at the University of Applied Sciences HTW Berlin and her plagiarism team in 2010. The entire benchmarking includes 26 of the 47 available systems on the market and gives an overview of the strenghts and weaknesses of these systems in finding plagiarism.

We used the top 5 "partially useful" software systems in this benchmarking to overview the market and find the best usable features for our own system to simplify the daily work of plagiarism finders.

PlagAware

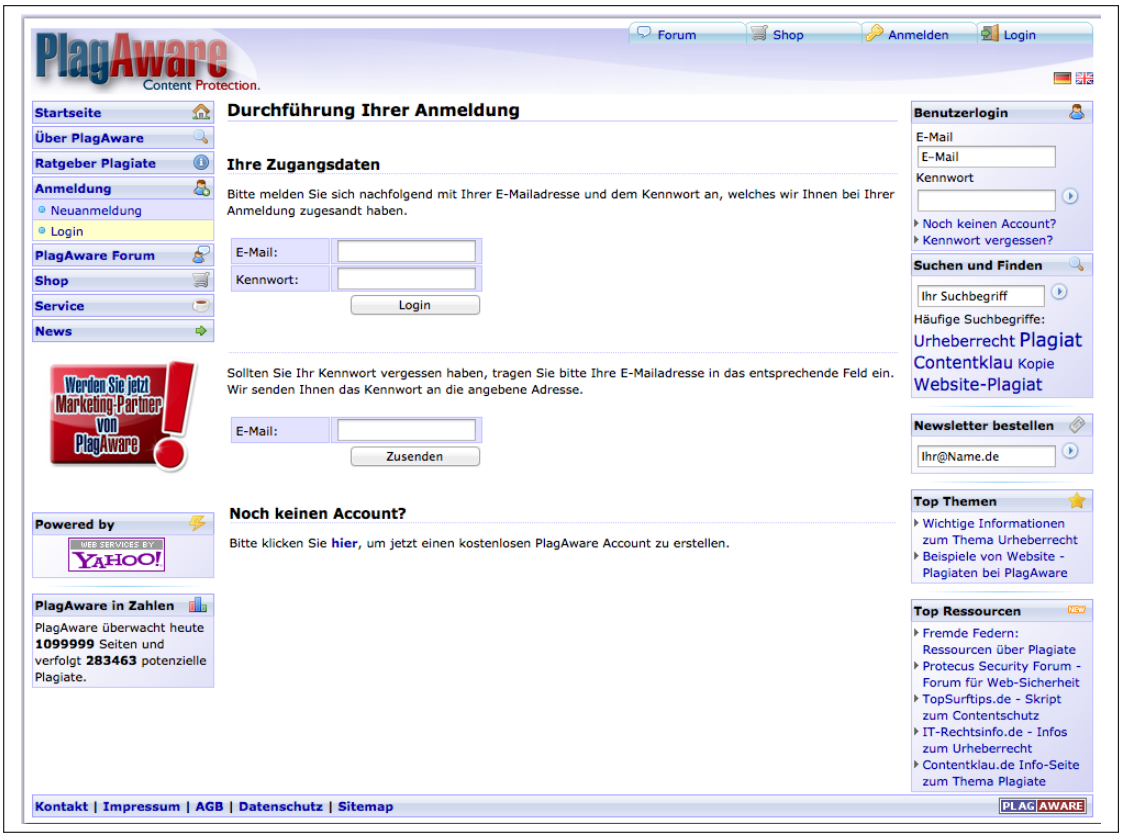

Figure 1.3.: Plagaware Website

Plagaware Business Promotion

"PlagAware is an online-service, which offers services around the topics Searching, finding, analysing and tracing of plagiarisms. The central element of PlagAware is a search engine, which is specialised in detecting identical contents of given texts. Contrary to the plagiarism scanning with classical search engines, the places of finding are not directly transferred to the user, but analysed on the rate and the type of analogy, before a message to the user is written. By this the differing result reports of PlagAware allow to recognise very fast the percentage and the distribution of the copied text contents, thus permitting an efficient and secure rating of a possible plagiarism."

“PlagAware Business Promotion”(**?**) PlagAware

PlagAware is software company from Ulm in Germany. The website is online for nearly 5 years and is the top-ranked system in the HTW ”Plagiarism Detection System Test” 2010, but in fact it still detect only 61,11% of the plagiarism cases. Although PlagAware ”produces excellent documentation of the plagiarism found, highlighting the commonalities in a side-by-side presentation. However, its usefulness at university is limited, as each file must be uploaded individually - no ZIP file or student-submission is possible. The system was not designed to be used in a university setting, but rather to find plagiarisms of online texts, which is important for sites trying to optimize their search machine ranking, as plagiarism will contribute to downranking.” Plagiat Website

Figure Overview shows an overview of all fragments and the results of the plagiarism detection. In figure Side-by-side shows a nice side-by-side fragment view, where all found plagiarisms are shown with different colors. Anomalies in the text are highlighted and the barcode-view is available.

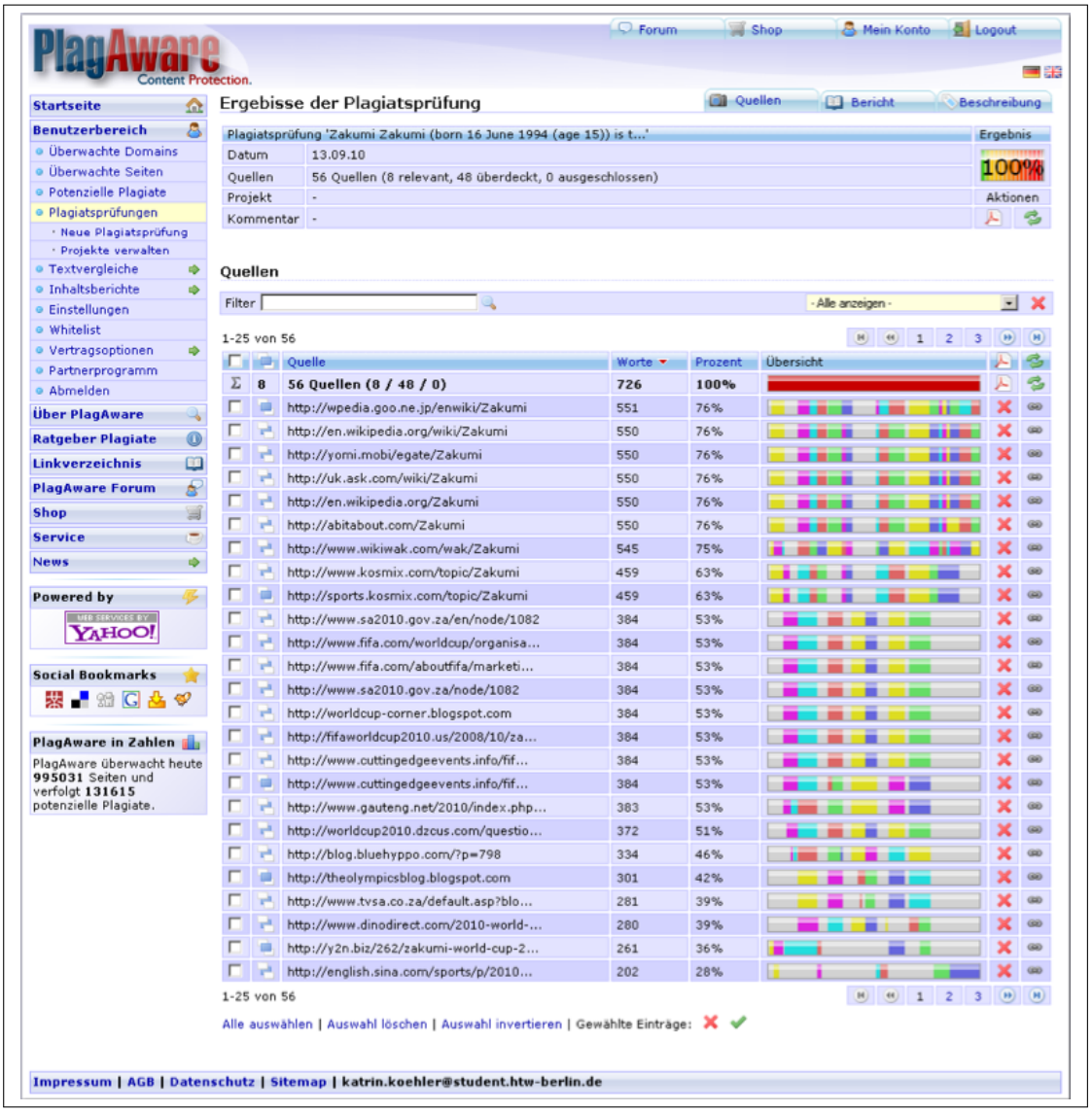

Figure 1.4.: Plagaware overview

PlagAware Costs

PlagAware has four different payment-models:

1. **Free**

   30 scans/month for free. Every additional scan costs 3,0 ct.

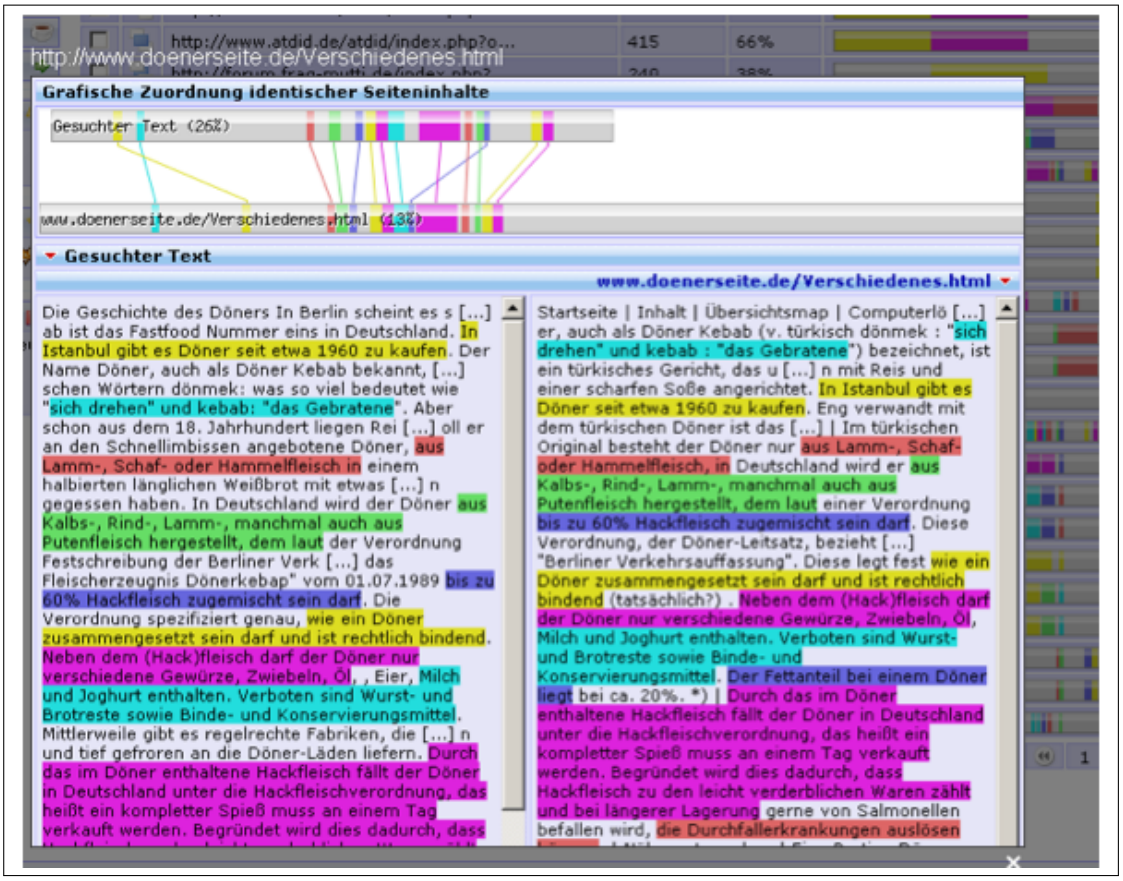

Figure 1.5.: Plagaware side-by-side view

2. **Light**

   EUR 2,99/month. 150 scans/month included. Every additional scan costs 2,0 ct. Mininum term of 6 month.

3. **Standard**

   EUR 7,49/month. 500 scans/month included. Every additional scan costs 1,5 ct. Mininum term of 6 month.

4. **Premium**

   EUR 14,99/month. 1500 scans/month included. Every additional scan costs 1ct. Mininum term of 6 month.

Turnitin

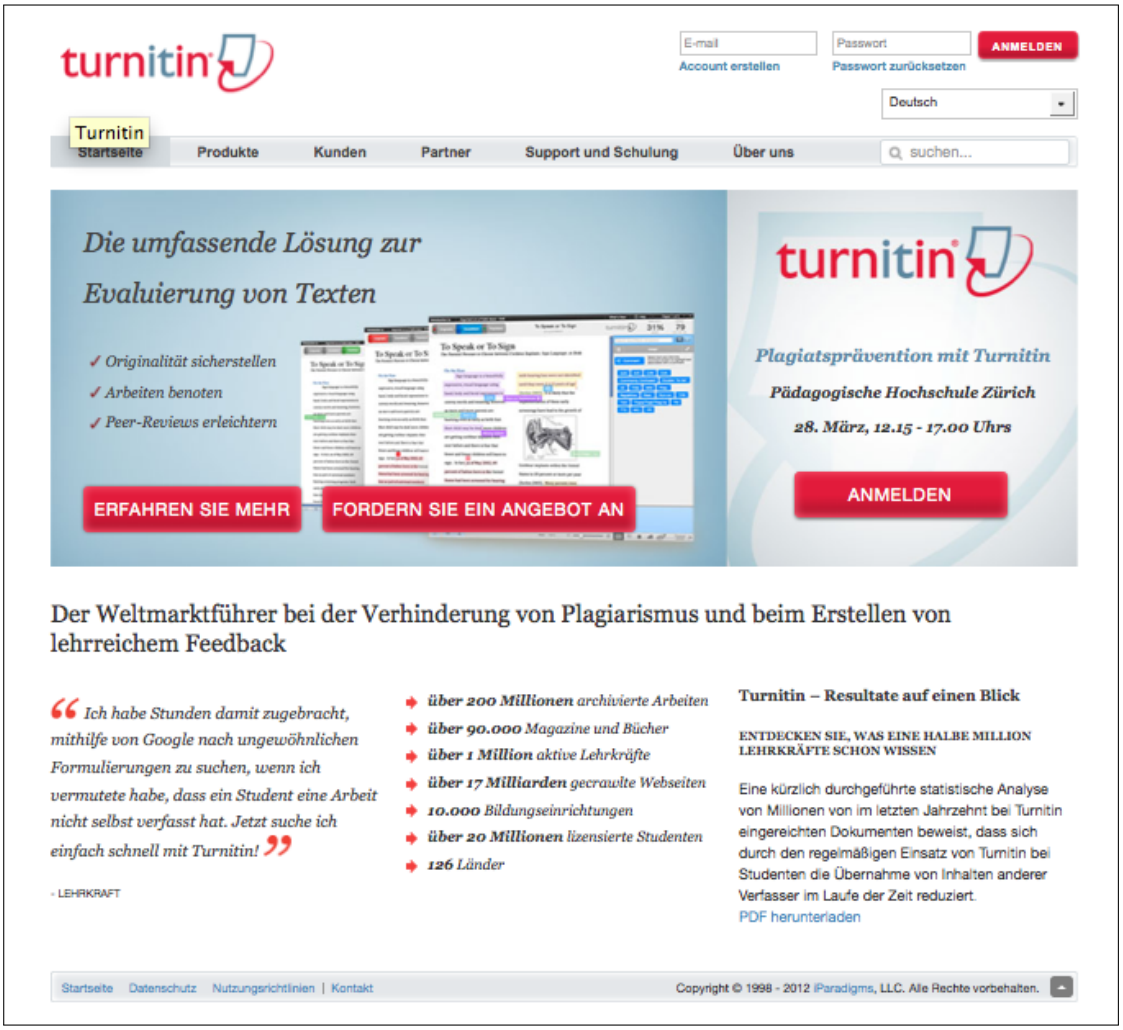

Figure 1.6.: Turnitin Website

Business Promotion

> ”Our award-winning solution discourages plagiarism and facilitates rich, meaningful feedback that improves writing skills, promotes critical thinking, and streamlines grading.”

“Turnitin”(**?**)Turnitin

Turnitin is a product by a company called iParadigms. It is a well-known US plagiarism software system and one of the most used plagiarism detection systems in the education sector. The website is online for nearly 13 years and the system is at the second position in the HTW "Plagiarism Detection System Test" 2010.

The best results can be achieved with material which is already in the database.

In the past the had a lot of problems to deal with umlauts, and having a complex setup. They improved a lot of parts especially the german translation. Still a big problem for european countries is the copyright policy of Turnitin. They still storing copies of user material in their database without a permission. Plagiat Website

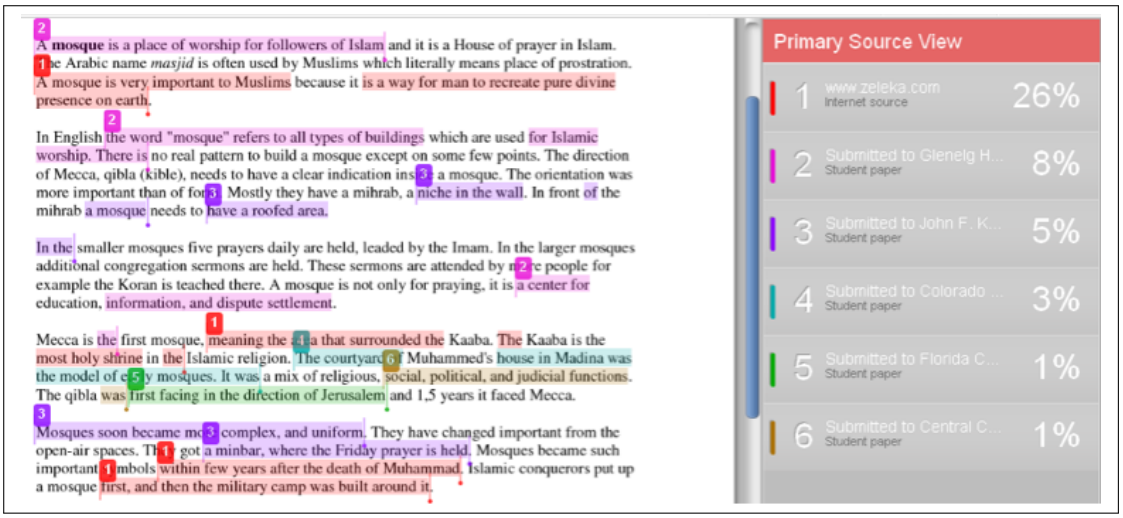

Figure 1.7.: Turnitin - A lot of little matches can't be found, if the sensibility has not been raised.

In 2008 the system was placed at the 13th position. The reason of this change is that the others systems have gotten worse. Turnitin has still problems of flagging spam sites especially when this sites are not safe for work (e.g. site with pornography content).(figure porn) On the other hand the search algorithm of Turnitin is storing sites in their database although they are still not exist. Plagiat Website

Costs

There are different license models for the education sector and the cost depends on the amount of users .

Figure 1.8.: Turnitin - A lot of spam-sites are reported. Not all sufficient to use at work. This is one of the harmless examples.

Ephorus

Figure 1.9.: Ephorus Website

Business Promotion

> "Never search for plagiarism yourself again? An end to all irritations and qualitatively better papers? No problem. With Ephorus, you can prevent plagiarism with no extra effort. Moreover, with this anti-plagiarism market leader, you will be assured of the best service and the lowest prices. With Ephorus, teaching will be fun again! Would you like to try out Ephorus?

"Ephorus Business Promotion"(**?**) Ephorus

The third position in the HTW ”Plagiarism Detection System Test” 2010 is Euphorus. It’s a plagiarism detection system from the netherlands and the website is online for nearly 8 years. 2007 it took the first place in the test, 2008 it was only position 8. Now they redisigned and reorganized the system and old problems were solved. The usability of reports and the whole handling of the system very good. (figure: report)But their still problems with umlauts and the european copyright problematic like in Turnitin. (figure: umlauts)

Figure 1.10.: Ephorus report - gives a great overview of the results

Ephorus costs

Not stated.

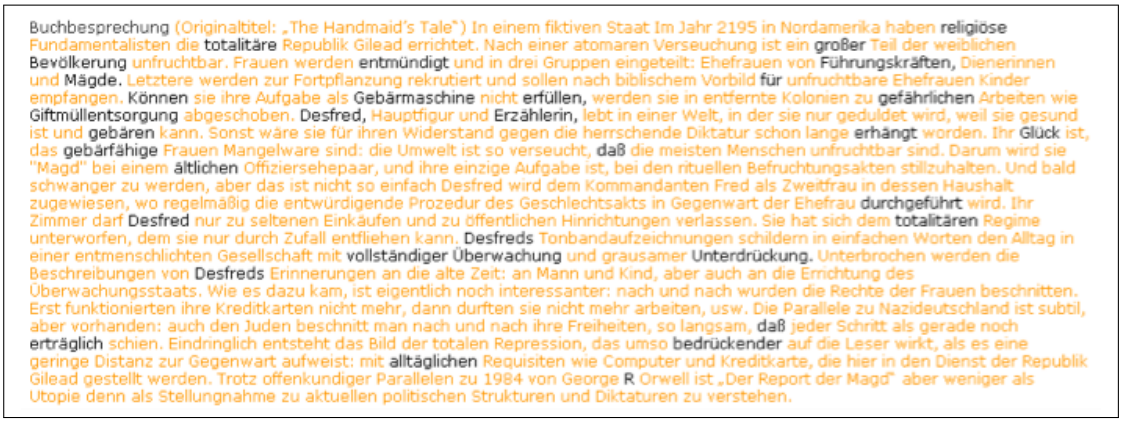

Figure 1.11.: Ephorus Problem with umlauts

PlagScan

Business Promotion

PlagScan stands for professionalism

- All documents are treated 100% confidential
- You control whether your document is checked against others, or not
- Integration via API in your existing CMS or learning management system possible

Plagiarism check as easy as pie: PlagScan

- Annotations directly in the document, check without additional work
- No installation - complete functionality in every browser
- All popular formats can be processed

Save time with PlagScan

Figure 1.12.: Plagscan Website

- Check several documents in parallel.
- Fully automated document analysis.
- No use of your resources, all computation is carried out on our servers.

"Plagscan Business Promotion"(**?**) Plagscan

Plagscan is a software company from Mainz, Germany. It placed at position number 4 in the HTW "Plagiarism Detection System Test" 2010. The website is online for 3 years and in the preview check 2008 it came to the 10th position. As a user you have to buy "Plag Points" (PP). One test costs 1 PP per 100 words. The administrator sets up users and assigns them points for use. There are three different kinds of reports - a list of possible sources with links to click on, the submitted document with the suspicious areas linked to a possible source, and a docx file with the sources in comments. There's no side-by-side presentation, so it's not possible to compare the fragments. Although there are still problems, PlagScan was first place in usability, but only 8th place in overall effectiveness with only 60% of the points awarded for finding plagiarisms.

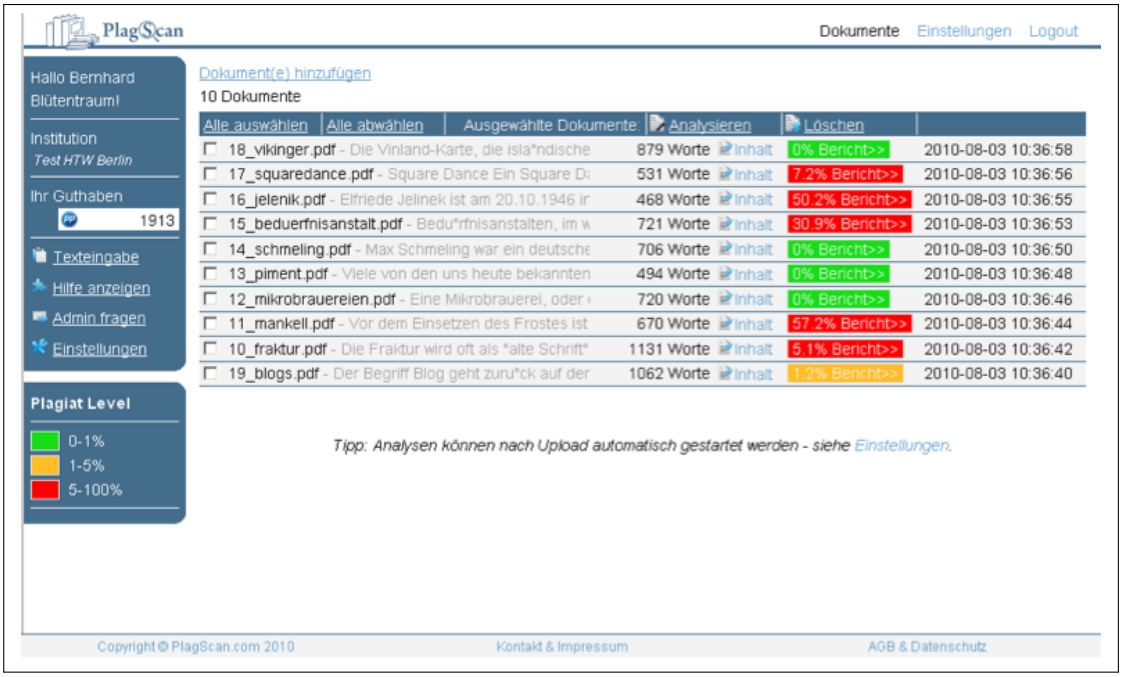

Figure 1.13.: Plagscan report is clear and tidy.

Plagscan costs

PlagScan has four different payment-models without a contract:

1. **9 Euro**
   500 Plagpoint - 50.000 words - 200 Sites.

Figure 1.14.: Plagscan reports are not self-explanatory.

2. **19 Euro**

   1.250 Plagpoints - 125.000 words - 500 Sites.

3. **29 Euro**

   2.000 Plagpoints - 200.000 words - 800 Sites. .

4. **69 Euro**

   5.000 Plagpoints - 500.000 words - 2.000 Sites.

Urkund

Figure 1.15.: Urkund Website

Business Promotion

> ”URKUND was born from the academic world. A team of teachers developed the idea of a web based service that would help them detect and deter plagiarism and URKUND was born in the fall of 2000. The problem of plagiarism received much attention in the media and more and more began realise the scope of the problem and the need of a tool to support the pedagogical work. URKUND continued to grow and develop over the years and came to be recognised as Sweden’s foremost anti plagiarism service.
>
> Today, URKUND is present in our neighbouring countries and continental Europe as well as the USA, Asia and the Middle East.
>
> URKUND is a natural part of the educational work of the academic world today. Both faculty and students are aware of the immediate and long term benefits of our system.”

“Urkund Business Promotion”(**?**) Urkund

Urkund is the last system in our comparison of partially useful systems. It ranked at the 5th position in the HTW ”Plagiarism Detection System Test” 2010. The company is from sweden and started their business in 2000. In this test it ranked high in effectiveness but on the other side it’s not easy to use. It has problems in the translation and after the redesign 2008 the usability was going worse. Overall ”the navigation is confusing, the layout at times catastrophic with texts overlapping fields, the printed reports could be better, the error messages are cryptic, and the link descriptions are unclear.”“Plagiat Team HTW”(**?**)

Urkund costs

Not stated.

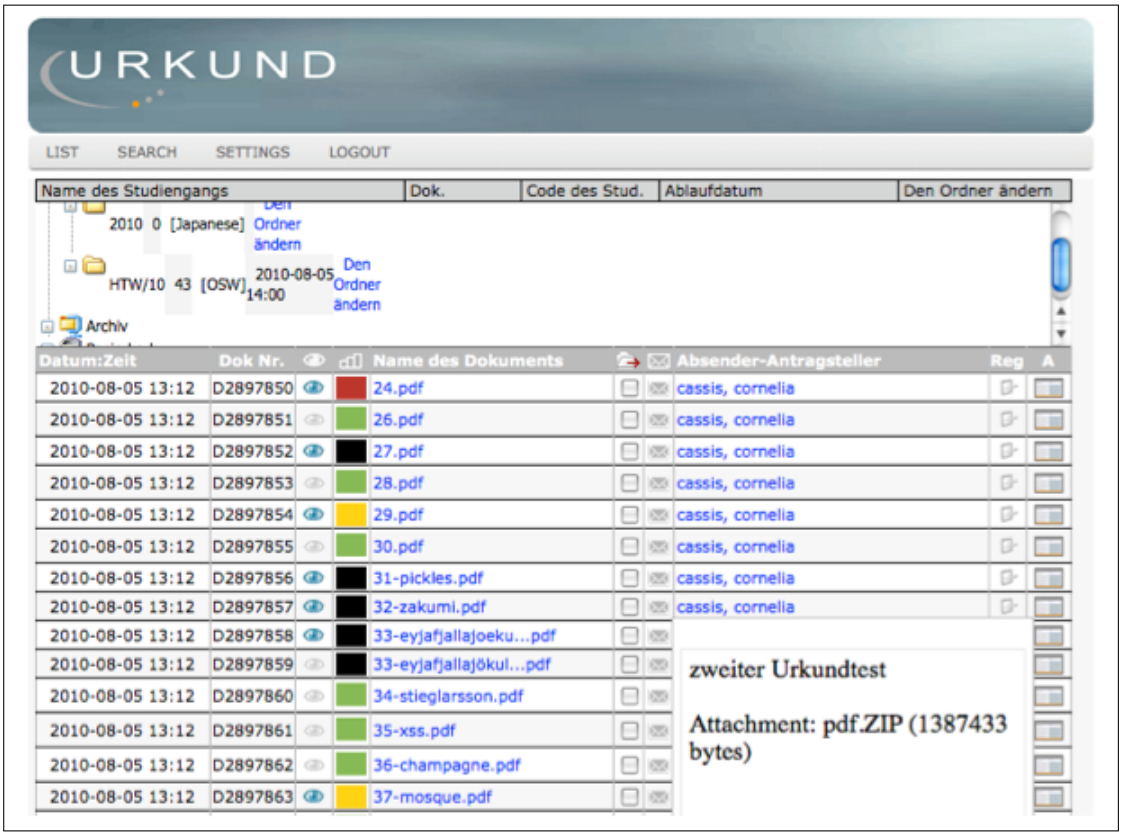

Figure 1.16.: Urkund List view

Resume commercial software systems

Although over the years there are more software detection systems that claim to check text reliable if it's plagiarism or not, but the quality of the systems decreases. A big problem that some of the tested systems offer "ghostwriting".

"In 2008 the systems fared slightly better on the test, but in 2010 many have lost the ability to detect plagiarisms that have been slightly edited - word orders switched, words dropped or added, or synonyms used. The best systems only reached a grade of C-, not quite reaching 70% of the possible points.

The 2010 test included not only short essays in German, but also ones in English and Japanese. Additionally, a usability metric was calculated that took into account aspects

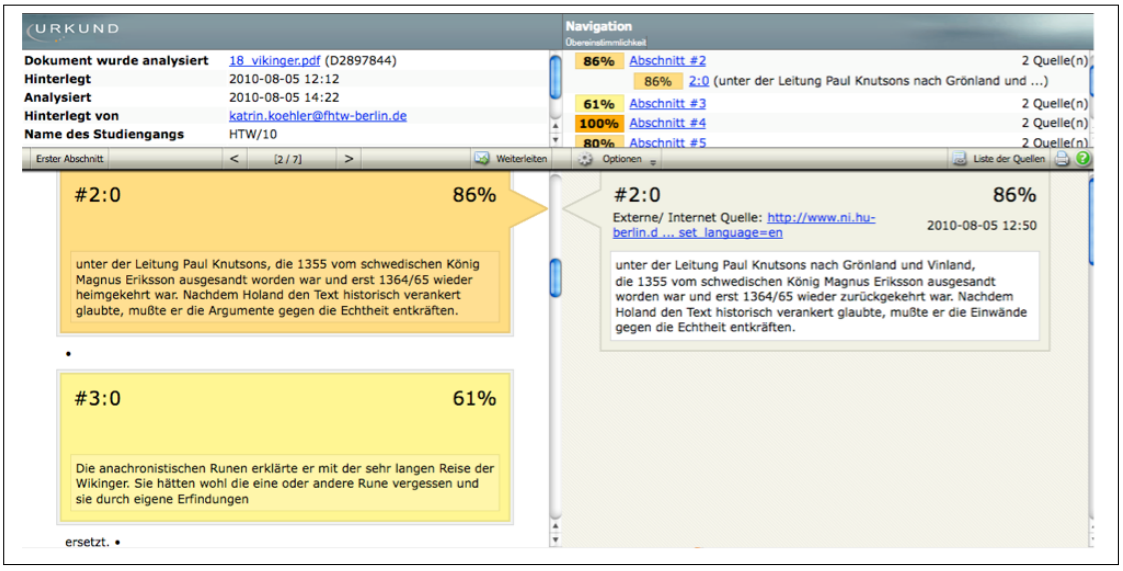

Figure 1.17.: Urkund Report

such as design, language consistency, navigation, print quality of the reports, and how well the system fits into the workflow of a university. A new professionalism metric includes giving a real street address and the name of a contact person; not advertising for paper mills or ghostwriting services; answering the phone during normal business hours and not installing malware on the computer under the guise of installing the detection software.

The systems were categorized as partially useful, barely useful, and useless for university purposes. The best systems with between 60 and 70% effectiveness are PlagAware, Turnitin, Ephorus, PlagScan and Urkund.

Our recommendation: Only use these systems when suspicions of plagiarism arise that cannot be found with 3-5 words in a search machine. The focus should be on teaching students about plagiarism and how to avoid it instead of investing time in using software. Most of the work involved later is in preparing a plagiarism case and dealing with the plagiator, and for this good documentation is needed. Very few systems provide good documentation." (plagiats)

1.3.2. Free tools and techniques

simtext Google-Search

Figure 1.18.: Google Search

1.4. Vroni Plag

VroniPlag is a wiki platform which many volunteers, who are ready to give their free time, their money or their books/resources etc. work on. They collaborate with each other in order to detect plagiarism in dissertations or habilitations.

Das VroniPlag Wiki ist ein am 28. Maerz 2011[2] auf Wikia gegruendetes Wiki, das verschiedene Hochschulschriften - hauptsaechlich Dissertationen - untersucht, die unter Plagiatsverdacht geraten sind. Die Untersuchungen fuehrten in mehreren Faellen zur Aberkennung des Doktorgrades. Der Gruender von VroniPlag ist Martin Heidingsfelder. Das ist nicht genau bekannt, da man sich fuer eine Mitarbeit nicht anmelden muss. Es gab am 18.3.2012 6335 Seiten im Wiki, 189 verschiedene Accounts haben schon mal editiert, 21 Leute sind als Admins eingetragen, und es gibt 3 Buerokraten, die Rechte vergeben koennen.

VroniPlag is named after the first case which they published. The case was the one of Veronica Saß.

Because it is a wiki page, which means open to all, every one could join in. If somebody is interested in a public case, he can feel free to edit the page without asking for an allowance. In the chat portal of this community, one could ask for more help or to take a look at the list of waiting fragments.

Before starting detection, there are some information that a collaborator might know.

1.4.1. Plagiarism detection steps

A suspicious case is called *candidate case*. This case is still not public for all. If a user has detected some suspicious part of a dissertation, he/she may first go to Chat portal to indicate his/her suspicion. He/she has also to give at least one original text source as proof for it. If it is well reasoned that there is an existing plagiarism, he/she must see if there are enough collaborators, who are ready to spend their time to work with.

The candidate case has been given an anonymous name and not public until there is proof, that there is at least 10% of the pages, in which plagiarized texts are found. After that the case will be published with the name of the author.

During the detection process, the candidate case will be divided into smaller fragments. The fragments will be checked carefully if there is a plagiarism found. If there is, a report is created which shows in which fragment the plagiarism is found. The original source is also correspondingly given.

After checking, fragment's state is changed to "to be proofed." That means, this fragment must be checked again by a second inspector. The result will be classified in corresponding category (see VroniPlag's classification of plagiarism).

After all fragments are checked, an overview of detected plagiarism will be generated. The overview includes the following parts:

Part 1:

The whole page numbers with two different colors. The page number includes also a link redirecting to corresponding fragment.

- Grey: page in which there is no plagiarism found or not checked yet.
- Blue: page with plagiarised texts found. The link goes to the plagiarized text in comparison with the source as well.

Part 2:

The second part is a generated barcode label which performs the percent of plagiarism found in one page with different colors.

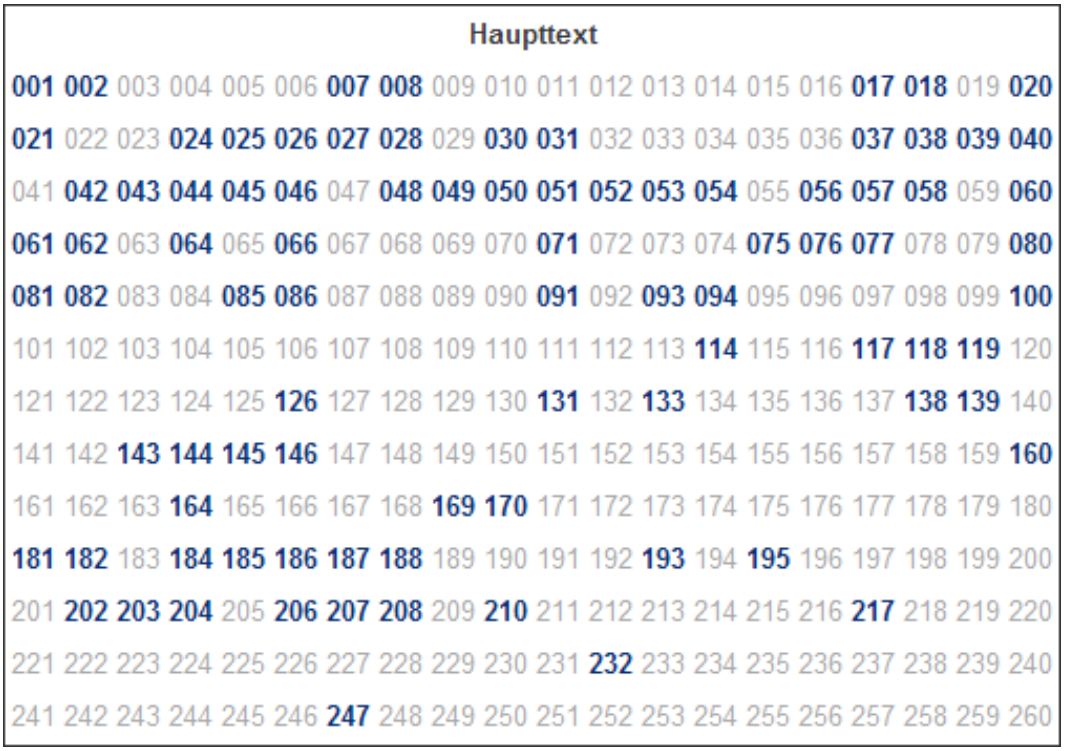

Figure 1.19.: Source: http://de.vroniplag.wikia.com/wiki/Lm, 19/03/2012, 08:53

- Blue: pages which are not calculated in the dissertation such as Index, Appendix, literature list
- Grey: suspiciously plagiarized
- Black: verified that 100% of the page is plagiarized
- Brown: verified that more than 50% of the page is plagiarized
- Red: verified that more than 75% of the page is plagiarized

If there is more than 10% of the whole pages plagiarized, the case will be public on wiki. Then a report will be sent to the university, where the dissertation is finished.

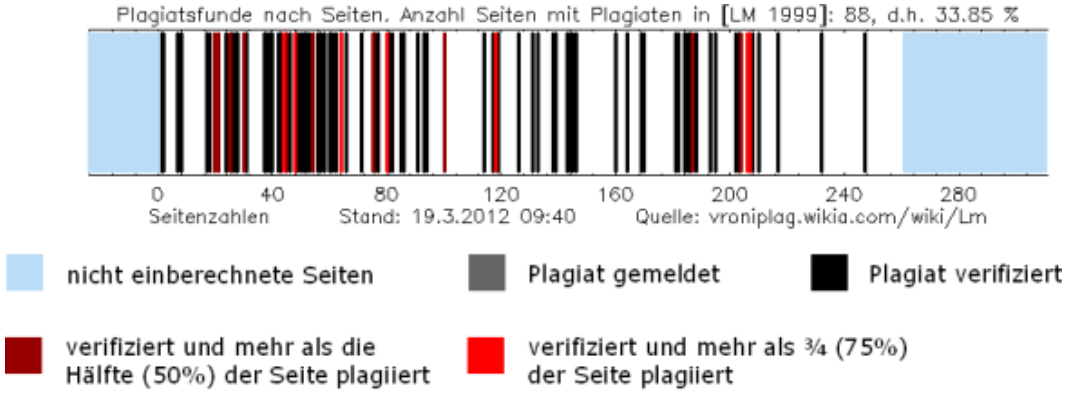

Figure 1.20.: Source: http://de.vroniplag.wikia.com/wiki/Lm, 19/03/2012, 08:54

1.4.2. Technical support

Most of the work by VroniPlag is done by hand. For example collaborators could use Google to search for sources, or they borrow books from the library and scan the texts. Generally there is no special software to help detect plagiarism, but collaborator could feel free to choose some of the existing software to help work faster.

1.4.3. Pros & Contras

Figure 1.21.: colors

2. Project Workflow and Requirements

First of all, we've got some kind of confession to make: Unplagged is like a big playground of new workflows and technologies for us, as we are aiming to incorporate "best-practices" wherever possible, or at least what we currently consider to be best-practices.

We believe this approach is necessary, because of the fact, that we are essentially trying to incubate Unplagged as a real open source project and this will only work if it is well crafted and if cutting-edge workflows and technologies are used. Nearly all of the team members are also working in some kind of web related side job, so we all got enough experiences with the problems that can occur during the maintenance of badly designed software.

Most of the times this works pretty well, but sometimes we are still trying to figure out how to get everyone up to speed with every technology and part of the system or how to divide the responsibilites carefully.

To start this project, we opted to use *Scrum*[1] as our agile development approach. If you are familiar with this methodology, you may notice, that there could be a few problems when considering, that the team is working mostly distributed without a common office and with very different time tables for each of the members.

We struggled a bit to tweak the workflow that is required by Scrum to fit the situation we faced, but you will see in the following what we came up with.

[1] A nice introduction into Scrum is "The Scrum Primer" of the Scrum Alliance: http://www.scrumalliance.org/resources/339

2.1. The Workflow

To make it possible to work efficiently together in this kind of environment, we chose to use Redmine as our project management tool, which you can access under:

- http://tickets.unplagged.com

If you register there, an administrator should grant you access to the tickets and the wiki, so that you can participate in solving the problems at hand.

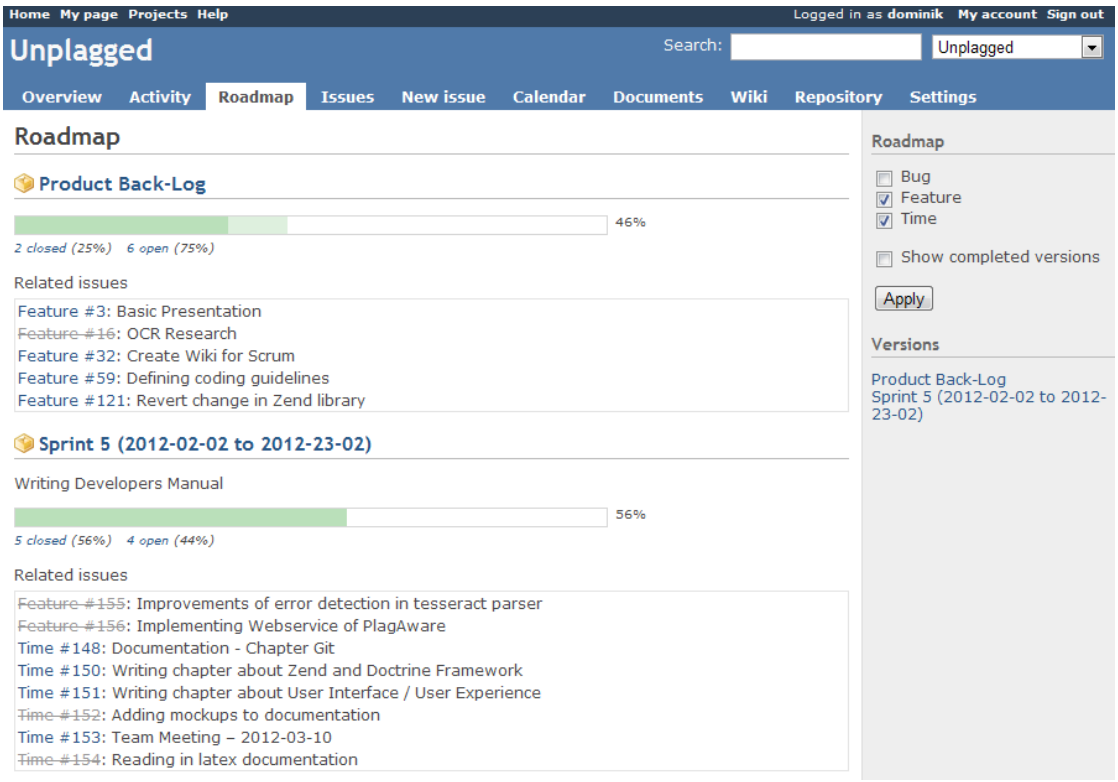

Figure 2.1.: Redmine Roadmap

What we are doing there is to map every *Sprint* and the *Product Backlog* to Redmine's notion of "Version" and every identified *User Story* to an "Issue".

You can see in figure 2.1 the view of the roadmap, with the current sprint 5 designated to create this very document at the bottom and a not very well filled product backlog at

the top.

Normally every identified user story that is not part of the current sprint should be in the product backlog (and we got plenty), but as we were still working in our small group at this point, the hassle of filling in all the tickets seemed unnecessary. This is something that will be fixed in the near future, so that you are able to see where the development is going.

Currently we are working mostly with four week long sprints, to overcome the problem that we are not working fulltime on the tickets, which is something that scrum normally assumes.

To have a nice statistical overview and more planning security for the "scrums", it is required to log the time that was spent on an issue within redmine.

2.1.1. Product Owner — "The Debbie Meetings"

Figure 2.2.: Scrum Meeting

To figure out the user stories we mostly rely on what we internally call "Debbie Meetings". Normally at the end of every sprint, the members of the team meet with Prof. Weber-Wulff, who we state to be our *Product Owner*. We simply sit down there and talk about

what should be implemented in the next sprint and collect it on cards as you can see in figure 2.3.

We consider this to be just a temporary way of handling this, because we hope that when we eventually have a prototype that we consider to have enough "business value" to be shown to more people, the focus will shift away from Prof. Weber-Wulff to a more broader understanding of the product owner.

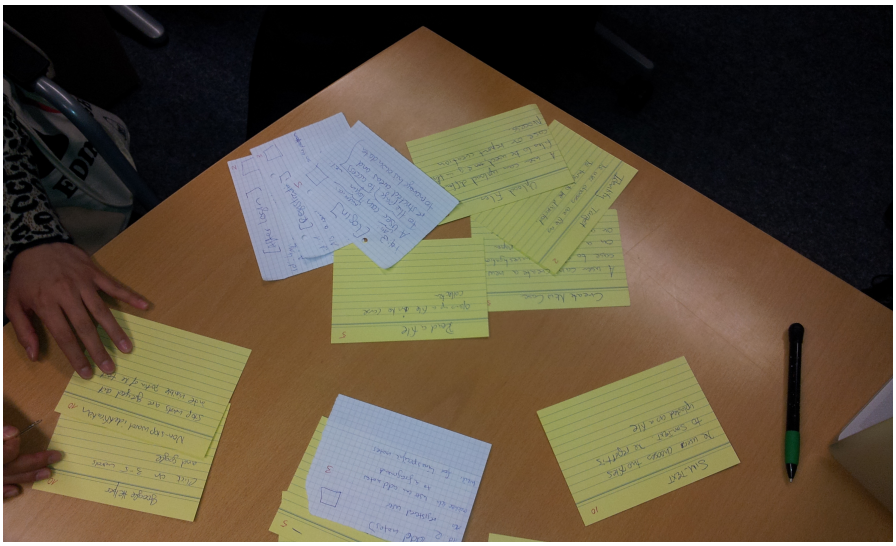

Figure 2.3.: User Stories

This means, that we want to open up our ticket system to directly collect new user stories and gather feedback for example from the VroniPlag group, which can be reached through their IRC channel `#VroniPlag` on:

- http://webchat.freenode.net/

2.1.2. Team Meetings

The Unplagged version of the *Daily Scrum* is currently a weekly meeting.

2.2. Target Group

2.3. User roles

As the Unplagged system will provide a permission based user system, our goal is to make it possible, to create custom user roles from an administration area and make it possible for users to have multiple user roles in one case and also different roles for different cases. The standard roles which will be provided by the system are:

Guest A user without a valid login can only see the parts of cases that are set to be public.

Registered Registered users can get "promoted" to higher roles and contribute to publicly editable cases.

Collaborator Collaborators are registered users who were granted access to a specific case. Collaborators can access and edit these projects.

Case-Manager Case-Managers can set up new cases and manage colloborators for their cases and project versions. They may have the permission to add or remmove project members.

Admin An admin owns all permissions, such as user administration or project administration. They also hvae the ability to to block/unblock an existing case.

2.4. Basic functionalities

2.5. Document Parser

2.6. Detection Modes

2.7. Plugin Architecture

2.8. Use Cases

3. Developing Unplagged

Coming from the Project Workflow and Requirements here we have yet another set of requirements, before we can start with the actual description of the used technologies within the system. This time it's about what we believe will be helpful or sometimes even necessary prerequisites.

First of all, the programming languages mostly used in Unplagged are PHP and JavaScript, both of which in conjuction with a framework. Teaching programming languages is, as you probably can imagine, well beyond the scope of this document, but we will at least try to cover the most important concepts of the frameworks as they occur.

The used frameworks are jQuery for Javascript and ZEND for PHP respectively. jQuery is kind of the industry standard for unobtrusive scripting with about 50% of the Top 10.000 websites using it according to Built With Trends (2012) and the Zend framework is also well established and brings a lot of features, that are useful to this project.

For most of the other topics, we will give you some (hopefully) helpful resources on the way, if it isn't covered thoroughly by us. But just to let you know, here is a list of the buzzwords, err technologies that will be mentioned:

- HTML5 and CSS3
- Continuous Integration
- Responsive Webdesign
- Progressive Enhancement

- Git, Netbeans, Redmine
- Tesseract, Imagemagick, Simtext

As said before, the system is developed, so that it should work on multiple platforms. This makes it sometimes difficult to describe certain installation processes in a way that would work for everybody. As it's often most problematic, to get some Linux software running on Windows, we will mostly concentrate on the way those things are done on this platform and give the instructions for other operating systems as an aside if necessary.

3.1. Development Environment

The following section will mostly focus on the way you can get a development version of unplagged up and running on your system.

3.1.1. Git

The source code and files of all parts of the Unplagged project are managed through Git, with the repository hosted at Github. Git is a distributed version control system, that exists since 2005 and gained more and more track in recent years. Many developers prefer it over other version control systems, because it is much easier to create different branches and merge them again or simply initalize local repositories. This made it also interesting for us to use it for Unplagged.

However, none of the team memebers had ever used Git before in a bigger context so it was a challenge to get it running on all the systems. But we took it, to explore all the features Git offers.

Installing Git Bash

First of all let's find out, how to install the Git console application, called Git Bash. Unfortunately all the GUIs we were evaluating didn't work consistently, so we decided to use it from the command line only. A very good instruction on how to install the Git Bash can be found on the website of the github project:

Windows: http://help.github.com/win-set-up-git/

Linux: http://help.github.com/linux-set-up-git/

Mac OS X: http://help.github.com/mac-set-up-git/

Getting the source code of the unplagged project

Now it is time to get the project source code on your machine. As said before, the whole unplagged project is hosted on github, so if you want to be able to contribute source code later on, you first need to create an account there:

- https://github.com

This isn't necessary, if you simply want to look into the source code, which can be accessed via the repository URL:

- https://github.com/benoertel/unplagged

If you haven't been granted write access to the above mentioned repository by a project member (which is very likely when you are reading this document for the first time), you will need to do a fork of the Unplagged project right at github, like described in:

- http://help.github.com/fork-a-repo/

After this, the following steps are mostly the same for everybody, with the distinction of the project URIs, which should be the one of your newly created fork.

Open up the Git Bash and switch to the directory where you want the project to be located and clone the repository as you can see in listing 3.1.

Listing 3.1: Cloning a repository

```
1 cd Sites/unplagged.local
2 git clone
      https://<username>@github.com/benoertel/unplagged.git
```

After this you should have a local copy of all the repository data in the specified directory.

The most important git commands

You are now ready to use Git! Here are some more instructions on the most important commands and how to properly use it. However, if the given instructions in this manual are not enough, feel free to checkout the whole Git manual on:

- http://schacon.github.com/git/user-manual.html

The Unplagged project consists of several branches, which are used to develop and store code independently of the other developers. Once a new feature is done, it is merged into the master branch. The master branch usually includes only fully tested and deployable source code.

As a new developer, it is important to create an own branch before doing anything else and switch to it.

Listing 3.2: Creating branches

```
1 git branch mynewfeature
2 git checkout mynewfeature
```

Now anything in the repository can be changed. At any point changes can be versioned in the repository by using the `git commit` command. If new files were created, `git add` has to be executed as well.

Listing 3.3: Adding all new files and commiting

```
1 git add .
2 git commit -m "A message that describes the changes."
```

When the feature is fully working and approved, it has to be merged back to the master branch, in order to get deployed to the staging environment. To do this, the master branch has to be checked out, updated with `git pull` and then all changes have to be merged from the new feature into the master branch. The feature branch can then be removed.

Listing 3.4: Merging branch into master

```
1 git checkout master
2 git pull
3 git merge mynewfeature
4 git branch -d mynewfeature
```

In comparison to Subversion for example, Git has one more step to really write back to the remote source repository. After a `git commit`, a `git push` has to be executed, each push can include multiple commits.

Listing 3.5: Pushing to the server

```
1 git push origin master
```

This is nearly it, the changes to the repository have been pushed to the master branch. The only thing, that probably has to be done now, is to open up a pull request on github, if you developed on your own fork of the project. This means, that you are asking the project members who have access to the "real" Unplagged github account, to integrate your changes into the actual project sources. A nice description of how this process is done can be found at github again:

- http://help.github.com/send-pull-requests/

Handling conflicts in merging process

It is possible, if two developers were working on the same part of a file, that a conflict is found during the merge. Such a conflict could look like this:

Listing 3.6: Merge conflict

```
1 CONFLICT (content): Merge conflict in readme.txt
2
3 To https://github.com/benoertel/unplagged.git
4  ! [rejected]        master -> master (non-fast-forward)
5 error: failed to push some refs to 'https://github.com/
      benoertel/unplagged.git'
6 To prevent you from losing history, non-fast-forward updates
      were rejected
7 Merge the remote changes (e.g. 'git pull') before pushing
      again.  See the
8 'Note about fast-forwards' section of 'git push --help' for
      details.
9
10 # Unmerged paths:
11 #   (use "git add/rm <file>..." as appropriate to mark
      resolution)
12 #
13 #    both modified:      readme.txt
14 #
```

To resolve the issues, open the files listed in the error message, in this case *readme.txt* and decide how the correct version should look like, by removing or changingn all the "< < < < < < < HEAD" and "> > > > > > > b478801d68267ef479acc5ca54544634c52c545c" parts accordingly or using a dedicated merge tool, that is able to show you the changes that were made

Here is an example of how this process would work:

Listing 3.7: Conflicted file

```
1 <<<<<<< HEAD
2 The goal of this project is the creation of an easy-to-use,
      web-based
3 system to document and detect plagiarism in scientific papers.
4
5 hello world
6 =======
7
8 The goal of this project is the creation of an easy-to-use,
      web-based
9 system to document and detect plagiarism in scientific papers.
10
11 >>>>>>> b478801d68267ef479acc5ca54544634c52c545c
12 Just a change for educational purposes.
```

It could look like this after merging:

Listing 3.8: Fixed conflict after merging

```
1 The goal of this project is the creation of an easy-to-use,
      web-based
2 system to document and detect plagiarism in scientific papers.
3
4 hello world
5
6 Just a change for educational purposes.
```

3.1.2. Local Deployment

This subsection will describe how to configure a virtual host properly. A virtual host is a domain that is mapped to the local web server. It is assumed that Apache, MySQL and PHP are already running on the machine. If not, here are some tutorial to get them all running:

Windows:

http://www.apachefriends.org/de/xampp-windows.html#1098

Mac OS:

http://www.djangoapp.com/blog/2011/07/24/installation-of-mysql-server-on-mac-os-x-lion/
http://www.quarkstar.at/index.php/2009/05/18/webserver-aktivieren-und-konfigurieren-in-mac-os-x/

Most Linux distributions should already have this kind of server stack installed.

The main goal to make the system run, is to create a local domain and add the virtual host from listing 3.12 to the vhost config.

In Max OS X this can be done via the command line:

Listing 3.9: Mac OS X: Creating virtual host

```
1 sudo vi /private/etc/hosts
2 #add the following line:
3 "127.0.0.1 unplagged.local"
4
5 sudo vi /private/etc/apache2/extra/httpd-vhosts.conf
```

On Windows you need to open up your *hosts* file, which is mostly located in *C:\WINDOWS\system32\drivers\etc\hosts*, and add the following line on the bottom:

Listing 3.10: New host declaration

```
1 127.0.0.1 unplagged.local
```

Now you need to open your apache configuration file *C:\xampp\apache\conf\httpd.conf* and remove the hash symbol(uncomment) from the following line

Listing 3.11: httpd.conf

```
1 #Include conf/extra/httpd-vhosts.conf
```

Add the following configuration to the httpd-vhosts.conf file you just included:

Listing 3.12: Apache configuration

```
1 <VirtualHost *:80>
2   ServerName unplagged.local
3   DocumentRoot "/Users/me/Sites/unplagged.local/public"
4   SetEnv APPLICATION_ENV "development"
5
6   <Directory "/Users/benjamin/Sites/unplagged.local/public">
7     Options +Indexes +FollowSymLinks +ExecCGI
8     DirectoryIndex /index.php
9     AllowOverride All
10    Order allow,deny
11    Allow from all
12  </Directory>
13 </VirtualHost>
```

You can tryout your new configuration by entering *unplagged.local* in your browser.

3.1.3. Netbeans

Configuring Tests

Documentation

Unplagged uses Apigen (2012) for the generation of a HTML page of all the source code documentation comments, because of it's superior and much more beautiful user interface in comparison to the older PHPDocumentor (2012).

Sadly the automatic generation is not yet supported by Netbeans, but as it will be soon(Heise, 2012), we are currently only generating this server-side as described in section 3.1.5.

This section will be enhanced, when the Netbeans user interface becomes available. If you are interested, you could install the software for yourself and use it over the command

line.

3.1.4. Additional Software

As we currently have no installer or script that checks for installed software, you still have to install some additional dependencies to make some parts of the system work. Those are mainly command line tools that we use for the optical character recognition or text comparison.

Most of the times the software wouldn't break completely if those dependencies were not installed, but some parts would silently fail, which is of course one of the more annoying problems to debug.

Tesseract

Tesseract is an open source OCR[1] software, that is currently used to "figure out" texts from scanned images a user can upload.

The big idea is to build the system in a way that enables users to plugin their favourite OCR system, but to have one default system that produces satisfying results in Tesseract.

You can download the latest installation files from:

- http://code.google.com/p/tesseract-ocr/downloads/list

After installing Tesseract, the easiest way to make it work with Unplagged, is to put it on your systems execution path, so that it could be called from the command line in any directoy.

If for some reason you don't want to do this, you can also change the following line in `/application/configs/application.ini` to the path where you installed

[1] Optical character recognition

Tesseract to:

Listing 3.13: Tesseract executable path

```
1 parser.tesseractPath = 'tesseract'
```

You should be able to click the "parse" icon on files now:

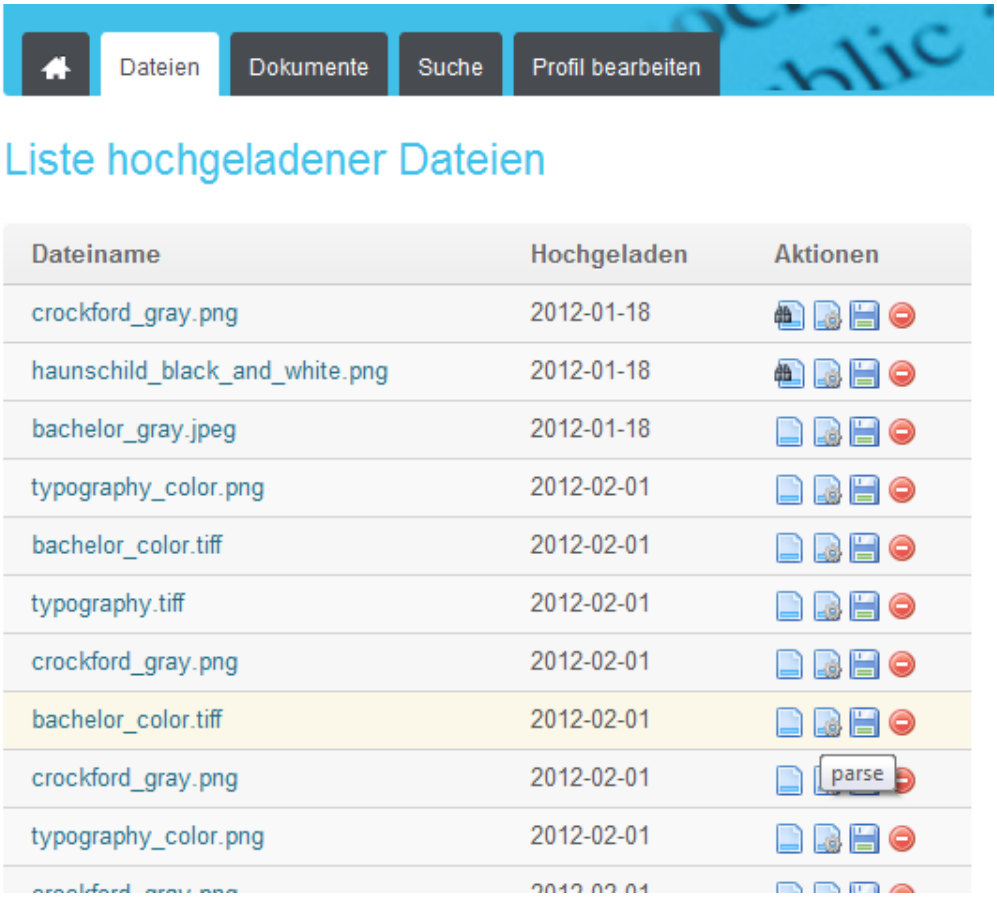

Figure 3.1.: Parsing Files with Tesseract

Imagemagick

Because Tesseract and probably other OCR systems that are provided via a plugin won't work with every image format a user decides to upload, we integrated Imagemagick as a

tool to convert images from one format to another. To install it you can simply follow the installation instructions provided here:

- http://imagemagick.org/script/binary-releases.php?ImageMagick=9dd9ttmq4g67oh5sbk4e6rjj70

Similar as with Tesseract, you can change the ini directive `parser.imagemagickPath`, if you chose not to include the executable in your path.

If you installed it properly, you can try to upload an image file other than *.tiff* and parse it with Tesseract, which should work now.

Simtext

Simtext is a text comparison tool, that is able to output nice data of the differences between texts. In figure 3.2 for example, you can see the default side-by-side comparison and in figure 3.3, the output in *diff* format is shown.

Installing Simtext can sadly be a bit tricky, because only the sources and no executables(at least none that worked for us) are distributed. We already provided a Windows 64bit EXE and an executable that runs on our Ubuntu staging environment in `/library/SIM/bin`, but if you use any other environment, you will have to compile the C sources of the system for yourself. The sources can be found here:

- http://dickgrune.com/Programs/similarity_tester/

We assume, that Linux users will be familiar with the installation steps that are described in the readme file. The only thing that can be easily overlooked there is the necessity to install *Flex* first. Windows however needs some special treatment to make the compilation work:

1. Download and install "Make for Windows" from http://gnuwin32.sourceforge.net/packages/make.htm and "Flex for Windows" from http://gnuwin32.sourceforge.net/packages/flex.htm

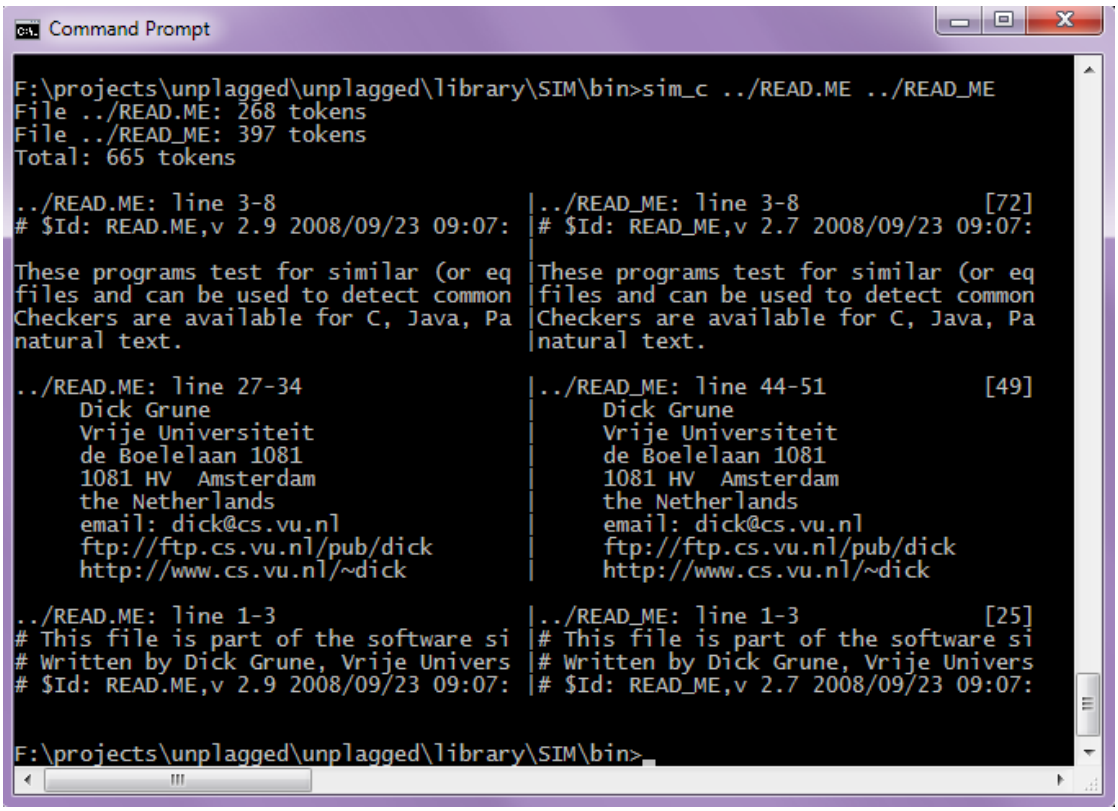

Figure 3.2.: Default Simtext output

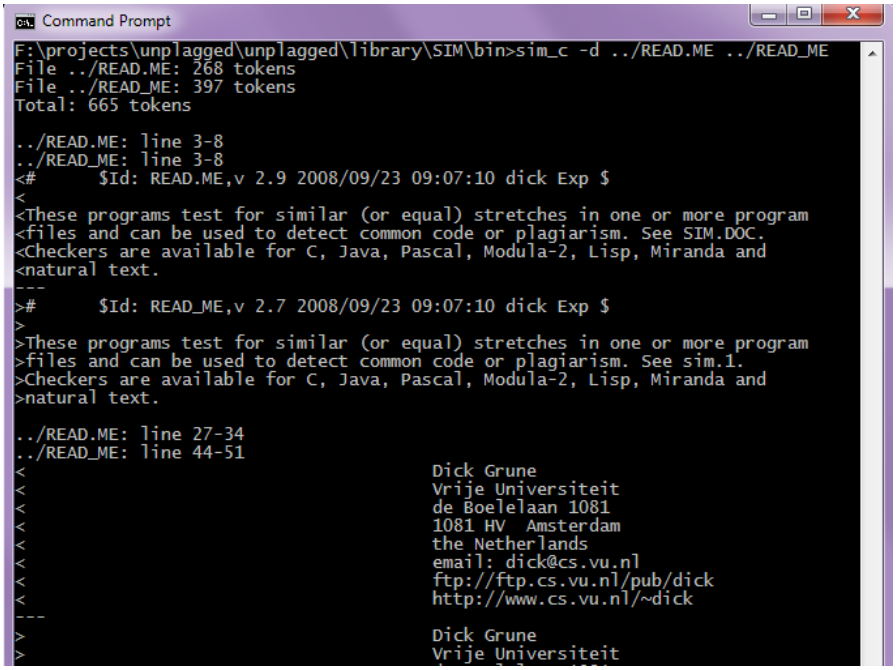

Figure 3.3.: Simtext output in diff format

2. Add the folder of the binaries to the path (should be something like `C:\Program Files (x86) \GnuWin32\bin`)

3. Rename “flex.exe” to “lex.exe”

4. Unzip and open Simtext directory on the command line and type:

Listing 3.14: Installing and checking simtext

```
1 >make
2 >sim_c --help
3 >sim_c READ.ME READ_ME
```

And again, to make it work you can set the ini directive `simtext.simtextPath` to the appropriate path.

3.1.5. Continuous Integration

To always have a running version of the latest code, we use an automated workflow, that always deploys everything that has been pushed to the Github repository on the Unplagged staging server. The machine this is done with, is a simple Ubuntu web server, that is also used for hosting our collaboration tools and the webpage.

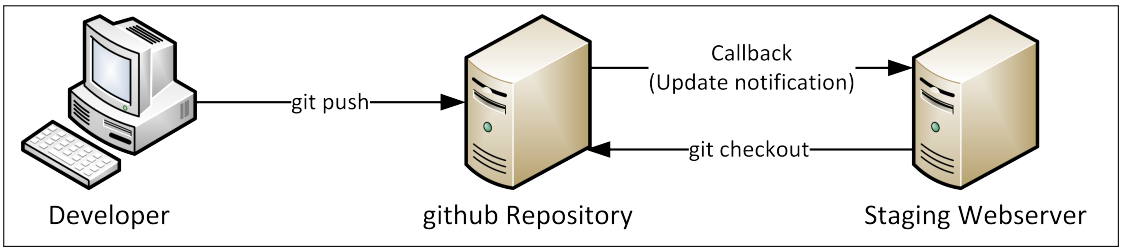

Figure 3.4.: Deployment workflow

As you can see in figure 3.4 the mechanism used for this is a callback, the *post-receive hook* of git, which github employs to let it's users enter a *post-receive URL* to call a URL after someone has pushed to the repository. The URL that gets called is located on our staging server and gets answered by a Redmine plugin called "redmine_github_hook", that would normally only call a checkout on the server-side repository, so that the newest sources can be seen via the Redmine web frontend.

Listing 3.15: Changes to redmine_github_hook.rb

```
1  # Fetches updates from the remote repository
2  def update_repository(repository)
3    command = git_command('fetch origin', repository)
4    if exec(command)
5      command = git_command("fetch origin '+refs/heads/*:refs/
           heads/*'", repository)
6      exec(command)
7
8      #custom checkout to preview area
9      system('sh /usr/local/etc/scripts/buildUnplaggedPreview.sh
           ')
10   end
11 end
```

However, we tweaked the source code of this plugin slightly, as you can see on line 9 in listing 3.15 so that it also calls the below bash script(listing 3.16) to initiate the deployment process.

Listing 3.16: Deployment script

```
1  #!/bin/bash
2
3  cd /var/git/unplagged.git/
4  GIT_WORK_TREE=/var/www/preview.unplagged.com git checkout -f
5
6  cd /var/www/preview.unplagged.com
7  #generate phpdoc
8  apigen -s application/ -s library/Unplagged/ -d docs/phpdoc --
       title "Unplagged Documentation" --todo yes
9
10 #run database build scripts
11 cd scripts/build
12 php initdirectories.php
13 php doctrine_staging.php
14
15 cd /var/www
16 chown www-data:www-data preview.unplagged.com
```

The bash script is then used to do a "clean checkout"(without hidden `.git` folders) of the repository and to run "Apigen", an engine to process the PHPDoc comments inside the project. Those two things can be accessed by the already prepared vHosts on the server:

- http://preview.unplagged.com/
- http://phpdoc.unplagged.com/

If you would like to get access to the preview areas, you need to obtain the password and username from a team member.

Possible Improvements

The above described workflow is, as we believe, already on a good way, but it still has a lot of room for improvement. First of all, it would be nice to only let the deployment go through, if the unit tests ran successfully on the server and to have some sort of email notification mechanism if this wasn't the case.

Another improvement would also be to have a separation into a staging environment with the newest commits and an actual preview environment, that can be deployed to a known stable state/commit of the system in a simple manner.

3.2. User Interface / User Experience

This chapter will explain the progress and development of the user interface of our project. As we tried to follow the typical project workflow, we first drew a lot of mockups in the beginning, which represented the main features of Unplagged. At first the wireframes, or also called mockups were drawn by hand, before we digitalized them. As an example the mockups of the 'new case' page are shown below in figure 3.5. All the other mockups can be found in the appendix B.

After we had a basic idea how the main interface should be structured, we created a first screen in Photoshop. Therefore we got several helpful suggestions from the website PremiumPixels:

- http://www.premiumpixels.com/.

The next step, before the HTML template got created, we defined the key features, our user interface should take care of:

- Reponsive Layout – optimized layouts for different devices
- Cross-Browser-Compatibility

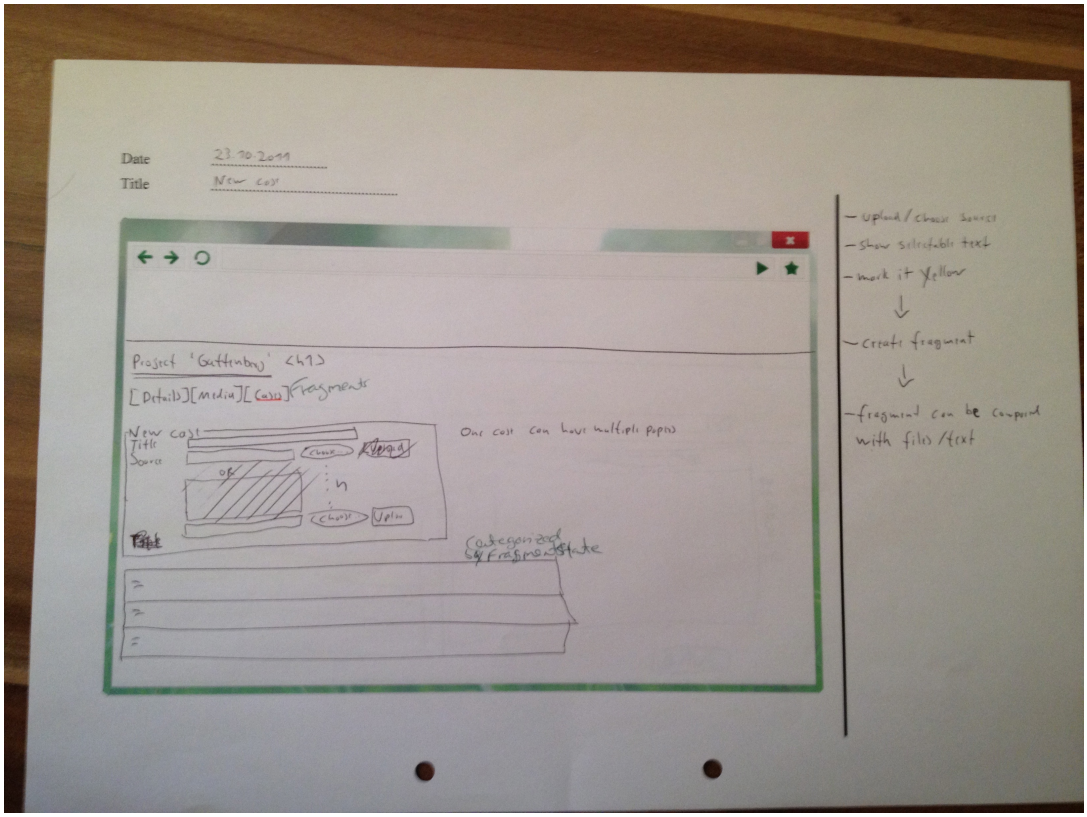

Figure 3.5.: Mockup – New case – hand-drawn

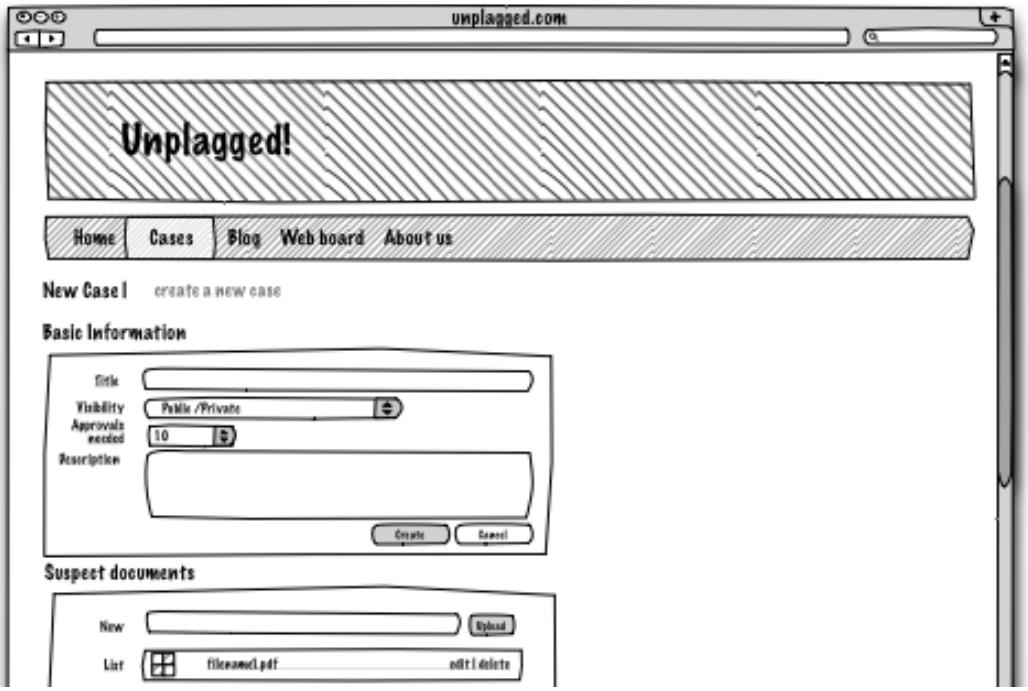

Figure 3.6.: Mockup – New case – digitalized

- Light-weight and w3c-conform HTML5 Markup
- Progressive enhcancement with CSS3 – CSS instead of images where possible

3.2.1. Responsive Layout using CSS3 Media Queries

Since the worldwide amount of different mobile and desktop devices is growing very fast, Ethan Marcotte coined the term *Resposive Webdesign*(Marcotte, 2011) for websites that are not optimized for any devices at all, but simply for different screen resolutions. And this is what we do.

Some functionality as uploading a file, doesn't work on iOS at all, but at least all functions that are working on mobile devices, should work. So the goal is to create a user interface that uses the same markup, but displays differently according to the device it is viewed on. Therefore CSS media queries are used, these are basically conditions that apply the CSS only if the condition is true. The most prominent conditions are the following:

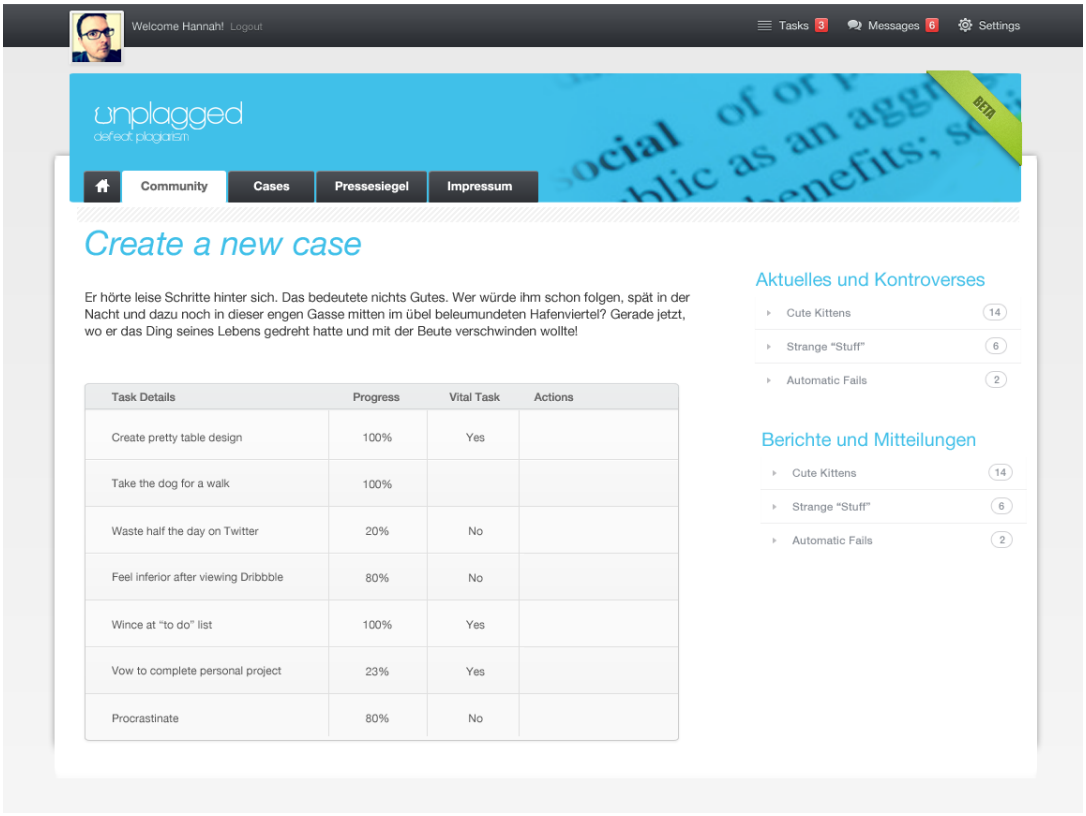

Figure 3.7.: Initial Screen PSD

Listing 3.17: CSS Media Queries

```
1 max-width: 600px  # max browser width 600px
2 min-width: 300px  # min browser width 300px
3 orientation:landscape # current orientation landscape mode
4 orientation:portrait # current orientation portrait mode
5 -webkit-min-device-pixel-ratio: 2 # min pixel ration (iPhone
      4, Retina)
```

These media queries can be combined in any order to display an optimized page for each device. Since only the CSS changes, the HTML does not have to be touched. An example query looks like this:

Listing 3.18: CSS Media Query

```
1 @media only screen and (max-width: 600px) { }
```

Media queries are a CSS2 feature which most modern browsers, except IE9, implement. This isn't a problem though, because browsers that don't support them, just display the default css outside of the media queries, which is optimized for a width of about 1000 pixels in our case.

3.2.2. Javascript and fallbacks

Even though many websites require Javascript as mandatory for using the whole functionality range of the page, it is important to provide as much functionality as possible, when Javascript is turned of, to ensure accessibility(Zeldman u. Marcotte, 2010, page 323). A short example will show how to implement a pagination with and without Javascript.

Usally when Javascript is disabled, the whole page will be reloaded, when the user changes to another page of the paginated content. The URL in this case will look something like this: http://unplagged.local/document/list/page/2. When Javascript is enabled, it is much faster to only refresh the area of the page, that really needs to reload, in this case the table with the content of the next page. It is only possible to change the hash

of an URL, the part after the hash key (#), using Javascript, the changed URL will be: http://unplagged.local/document/list/#page/2. An event called 'hashchange' can trigger a change of this part of the url and then reload the content through an AJAX request.

The pagination has the same HTML markup with and without Javascript, but with Javascript enabled it is much more convenient.

Listing 3.19: Javascript Pagination

```
1  $(".pagination a").live("click", function() {
2          var href = $(this).attr("href");
3          if(href) {
4            var substr = href.split('/');
5            var hash = substr[substr.length-2] + "/" + substr[
                 substr.length-1];
6
7            window.location.hash = hash;
8          }
9          return false;
10     });
11
12     $(window).bind('hashchange', function(){
13         var newHash = window.location.hash.substring(1);
14
15         if (newHash) {
16             var substr = newHash.split('/');
17             var hash = substr[substr.length-2] + "/" + substr[
                   substr.length-1];
18
19             var url = window.location.pathname;
20             if(url.charAt(url.length-1) != '/') {
21               url += '/';
22             }
23             url += hash;
24             $("#main-wrapper").load(url + " #main");
25         };
26
27     });
```

3.3. Frameworks

When we first discussed which programming language and frameworks, the Unplagged project should be built on, we figured out, that everyone was familiar with PHP. Since programming in a group requires a much better structure, than programming on your own, we needed a framework that requires a comfortable Model-View-Controller structure, we decided to use the Zend Framework. And to get rid of all the database issues as well as getting a flexibility in the used database system, we decided to use an Object-Relational-Mapping(ORM) framework, called Doctrine. What an ORM is, will be discussed later on.

3.3.1. Zend Framework

The Zend Framework is a typical PHP Framework using the Model-View-Controller pattern. Due to it's pre-defined directory structure, it is easy to get well seperated code.

The directories and their meaning:

- application — includes controller, models and view
- data — currently only includes i18n stuff (language stuff)
- docs — PHP Documentation and Developers Manual
- library — external and internal frameworks and extension to the Zend framework
- public — files that are accessible directly through the browser
- scripts — build scripts, deploying scripts
- temp — data that is overridden at any deployment

- tests — PHP Unit tests

The Zend framework offers RESTful[2] URLs that follow a fixed pattern: `unplagged.local/controller/a`
Each controller is defined as ControllerNameController.php in the `application/controllers` directory and includes all possible actions. For example a file controller will look like this:

Listing 3.20: Persisting an object to the database in Doctrine

```
1 class FileController extends Zend_Controller_Action{
2   public function init(){
3   }
4
5   public function indexAction(){
6   }
7
8   public function uploadAction(){
9   }
10
11 <<<<<<< HEAD
12   public function listAction(){
13 =======
14   public function listAction(){
15 >>>>>>> 7a2ccf18eebad5d280d777f496be384bdf2a24f8
16   }
17
18   public function downloadAction(){
19   }
20 }
```

By default, if no action is defined, the `indexAction` is called. For each action the appropriate view is by default rendered in the `application/views/scripts/conntrollerName/actionNa` file.

The `models` directory includes all the objects, this will be discussed in more detail in the following chapter about Doctrine.

[2]See for example "RESTful Web Services" of Richardson and Ruby for more information

3.3.2. Doctrine

The whole database connection management of Unplagged is implemented using the Doctrine Framework in version 2. It consists of two layers, a database abstraction layer (DBAL) and an object relational mapping framework (ORM). The DBAL uses PDO, a PHP framework for encapsulating database statements. The DBAL manages the communication with any kind of SQL database and offers an own query syntax. This has the advantage, that the database behind the framework can be changed from MySQL, to OracleSQL, PostgreSQL or SQLite at any time. The DBAL is the agent between PDO and the ORM. The ORM is the connection between PHP objects and the DBAL.

Before the use of Doctrine is described, it will be explained, how the database on a new machine can be created and how the database structure can be updated, whenever something changed in the structure. Actually this is very easy, it is required to have a local MySQL database at this point having root' as a username, no password and a database called 'unplagged'. It is also possible to create a new configuration in the application/configs/application.ini file, although this step will not be described in this chapter. If the database is created, the build script can be executed:

Listing 3.21: Updating database structure

```
1 php unplagged.local/scripts/build/doctrine.php
```

Now, the database is created or updated. If the database already existed, the data in it will not be removed! As described below, the big advantage of ORM is, that the programmer can stay in the PHP object context at any time. The only thing that has to be done additionally, is adding comments to the member variables of a class, that define the fields in the database:

Listing 3.22: Defining a class in Doctrine

```
1 /** @Entity */
2 class UserClass
3 {
4   /** @Column(type="integer") */
5   private $id;
6   /** @Column(length=30) */
```

```
7   private $username;
8 }
```

The whole syntax documentation of doctrine can be found here:

- http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/index.html

To persist a new object to the database, the persist method on this object has to be called, this writes the object into the doctrine cache. It stays in the cache, until the flush method is called, which actually executes all the previous operation since the last flush. These cann be deleting, updating, or editing an object.

Listing 3.23: Persisting an object to the database in Doctrine

```
1 $user = new User();
2 $user->setUsername('Max');
3 $em->persist($user);
4 $em->flush();
```

As the previous examples show, no database programming is necessary to create or persist a new object to the database, everyhting can be done in the PHP object context.

4. Summary and Outlook

During our first project semester, we already immerged into the field of plagiarism detection very deeply. After we read about exisiting websites and talked to Prof. Dr. Weber-Wulff about her experience and missing features at VroniPlag, we defined our list of requirements. In the next step we developed first user interface mockups and a basic layout, before we started the development of some requirements.

The application of a modified version of Scrum as our agile software development method was a very interesting and new experience to all of us. With the use of Redmine for keeping track of all issues, repository changes and time logs, everyone in the team was at anytime able to take a look on the current project state.

All in all the coditions for a succesful development were allocated properly and we already have implemented many features of the requirements list, that can be improved in the next semester.

Though the employed development processes in this semester were not as consistent as they should have been. The in theory defined rules, how to programm and how to test source code properly, were not always applied. So one of the main goals for the second semester is to improve this proccess. We need to focus on test driven development and to increase our velocity during the sprints, in order to get stable code more easily.

Another issue we need to work on is the staging environment we are using. Some features behave differently on Mac OS and Windows machines. Since we have only one staging environment, which updates on every commit to our git repository, it sometimes happens that a feature working on Windows crashes the staging environment. As a solution we should setup a second pre-staging envirnoment which updates at every commit to the

repository automatically and another more stable server, which gets updated manually as soon as the pre-staing environment is testet properly.

The documentation of our work, which is represented in this developers manual was done in the end of the semester. In the next semester we are planning to extend and keep the manual up to date at every sprint, which means preferably more time for development and less time for documentation in the end.

A. Logged Time As Of March 11, 2012

The following tables are some example reports generated from the logged time in Redmine. To find the most recent version of these reports or to generate custom data analysis you can use the "Report" tool found in Redmine on the "Overview" page.

Table A.1.: Overview By Member and Month

Member	**2011-10**	**2011-11**	**2011-12**	**2012-1**	**2012-2**	**2012-3**	**Total**
Dominik Horb	27.30	10.00	55.75	51.00	28.00	16.80	188.85
Benjamin Oertel	34.00	29.00	19.00	57.50	4.50	29.00	173.00
Elsa Mahari	9.50	10.00	23.50	14.50	7.00		64.50
Tien Nguyen	1.00	24.00	13.00	12.00			50.00
Heiko Stammel	16.00	28.00	13.50	11.00			68.50
Total	91.30	101.00	124.75	146.00	39.50	45.80	548.35

Table A.2.: Overview By Member and Issue

Issue	Member	Total
none		18.30
	Dominik Horb	18.30
Feature #1: Trac aufsetzen		2.00
	Dominik Horb	1.00
	Dominik Horb	1.00
Feature #5: Create Wiki List of Interesting Plagiarism Papers		1.00
	Dominik Horb	1.00
Feature #6: Logo		8.00
	Benjamin Oertel	7.00
	Elsa Mahari	1.00
Feature #8: Configure Email Notification in Trac		1.50
	Dominik Horb	1.50
Feature #9: Github in Trac integrieren		1.00
	Dominik Horb	1.00
Bug #11: Umstellung von Trac auf Redmine		3.50
	Dominik Horb	3.50
Bug #12: Zeitmanagement in Trac integrieren		4.80
	Dominik Horb	4.80
Bug #13: Initalize basics in wiki and issues		1.00
	Dominik Horb	1.00
Bug #14: Setup Github in Redmine		2.50
	Dominik Horb	2.50

Table A.2.: Overview By Member and Issue

Issue	Member	Total
Bug #15: Fremde Federn finden		3.00
	Dominik Horb	3.00
Feature #16: OCR Research		3.50
	Dominik Horb	3.50
Feature #20: Create Wiki Page for UserRoles		1.00
	Tien Nguyen	1.00
Time #21: Meeting 24.10.11		12.50
	Dominik Horb	2.50
	Benjamin Oertel	2.50
	Elsa Mahari	2.50
	Tien Nguyen	2.50
	Heiko Stammel	2.50
Time #22: Meeting 17.10.11		10.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
	Tien Nguyen	2.00
	Heiko Stammel	2.00
Time #23: Meeting 10.10.11		10.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Tien Nguyen	2.00
	Heiko Stammel	2.00
Time #24: Debbie-Meeting 18.10.11		7.00
	Dominik Horb	1.00
	Benjamin Oertel	1.00
	Elsa Mahari	2.00
	Tien Nguyen	1.00
	Heiko Stammel	2.00
Feature #28: Resume: Projectmeeting with HTW-Plag-Team		4.00
	Heiko Stammel	4.00
Feature #29: Projectmanagement Research		7.00
	Heiko Stammel	7.00
Feature #30: Scrum Research		6.00
	Heiko Stammel	6.00
Feature #31: Create Wiki for Project Management Steps		2.00
	Heiko Stammel	2.00
Feature #32: Create Wiki for Scrum		1.00
	Heiko Stammel	1.00
Bug #33: Mockup hand-drawng		6.00
	Benjamin Oertel	6.00
Bug #34: Mockup digital version		11.00
	Benjamin Oertel	11.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
Bug #35: Textvergleich-Algorithmen rechechieren		1.00
	Tien Nguyen	1.00
Time #39: Meeting 31.10.11		12.50
	Dominik Horb	2.50
	Benjamin Oertel	2.50
	Elsa Mahari	2.50
	Tien Nguyen	2.50
	Heiko Stammel	2.50
Time #43: Meeting 2011-11-04 (Martin Heidingsfelder)		8.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
	Tien Nguyen	2.00
	Heiko Stammel	2.00
Feature #44: Setting up initial Zend Framework + Doctrine environment		2.00
	Benjamin Oertel	2.00
Feature #45: User Stories		1.00
	Tien Nguyen	1.00
Time #47: Meeting 07.11.11		10.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
	Tien Nguyen	2.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Heiko Stammel	2.00
Time #48: Meeting 14.11.11		10.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
	Tien Nguyen	2.00
	Heiko Stammel	2.00
Time #49: Wiki Editing		2.00
	Dominik Horb	2.00
Bug #50: Meeting Plagiatsteam HTW 3.11.11		6.00
	Dominik Horb	1.50
	Benjamin Oertel	1.50
	Heiko Stammel	3.00
Time #53: Debbie Meeting 15.11.11		7.50
	Dominik Horb	1.50
	Benjamin Oertel	1.50
	Elsa Mahari	1.50
	Tien Nguyen	1.50
	Heiko Stammel	1.50
Feature #56: User registration		6.50
	Benjamin Oertel	6.50
Feature #57: Profile information update		3.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Benjamin Oertel	3.00
Time #58: Meeting 28.11.11		12.50
	Dominik Horb	2.50
	Benjamin Oertel	2.50
	Elsa Mahari	2.50
	Tien Nguyen	2.50
	Heiko Stammel	2.50
Feature #59: Defining coding guidelines		0.50
	Benjamin Oertel	0.50
Feature #61: Meeting 24.11.11 - Setting up developing environment		12.00
	Dominik Horb	3.00
	Benjamin Oertel	3.00
	Elsa Mahari	3.00
	Tien Nguyen	3.00
Feature #62: Identify target		0.50
	Benjamin Oertel	0.50
Feature #63: Read a file		1.00
	Benjamin Oertel	1.00
Feature #64: Upload files		8.00
	Elsa Mahari	8.00
Feature #65: User Login		6.50
	Benjamin Oertel	1.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Tien Nguyen	5.50
Feature #66: Create a new case		2.00
	Dominik Horb	2.00
Feature #67: Setting up PHPUnit		7.50
	Dominik Horb	3.50
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
Time #68: Meeting 05.12.11		4.00
	Dominik Horb	2.00
	Elsa Mahari	2.00
Time #69: Reading DocBook-XML		4.00
	Dominik Horb	4.00
Feature #70: Writing git tutorial		1.00
	Benjamin Oertel	1.00
Time #71: Meeting Online 07.12.11		16.00
	Dominik Horb	4.00
	Benjamin Oertel	4.00
	Elsa Mahari	4.00
	Tien Nguyen	4.00
Time #72: Meeting preparation PHPDoc		2.00
	Dominik Horb	2.00
Feature #73: Layout design		10.50

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Benjamin Oertel	10.50
Bug #74: protocoll meeting minuts 05.12.2011		0.50
	Elsa Mahari	0.50
Time #75: Creation of Preview area on server		3.00
	Dominik Horb	3.00
Time #76: Meeting 12.12.2011		7.50
	Dominik Horb	1.50
	Benjamin Oertel	1.50
	Elsa Mahari	1.50
	Tien Nguyen	1.50
	Heiko Stammel	1.50
Time #77: create Wiki page for MeetingMinutes 15.11.2011		0.50
	Tien Nguyen	0.50
Feature #78: Setting up the unplagged-blog		17.00
	Heiko Stammel	17.00
Feature #79: Google Helper		10.00
	Elsa Mahari	10.00
Feature #80: SIM-Text		13.00
	Dominik Horb	3.00
	Tien Nguyen	10.00
Feature #81: Correct/edit OCR-scanned Text		4.00
	Benjamin Oertel	4.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
Feature #82: De-hyphanator		8.00
	Benjamin Oertel	8.00
Feature #83: Non-stop word identification		8.00
	Dominik Horb	2.00
	Heiko Stammel	6.00
Feature #84: Display text		3.50
	Dominik Horb	1.00
	Benjamin Oertel	2.50
Feature #85: Standard OCR		18.00
	Dominik Horb	12.00
	Benjamin Oertel	6.00
Feature #86: Change file list view columns		0.50
	Benjamin Oertel	0.50
Feature #87: Decrease line-height in tables.		0.50
	Benjamin Oertel	0.50
Time #89: create Wiki page for MeetingMinutes 13.12.2011		0.50
	Tien Nguyen	0.50
Time #90: Prepare automatic preview area		11.25
	Dominik Horb	11.25
Time #101: Preparation of Database Mocks		3.50
	Dominik Horb	3.50
Feature #102: Basics for responsive layout		9.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Dominik Horb	9.00
Time #103: Meeting 10.01.12		3.00
	Dominik Horb	1.50
	Elsa Mahari	1.50
Bug #104: Skype Meeting 4.01.12		1.00
	Elsa Mahari	1.00
Bug #105: Switching Menu to Zend_Navigation		3.00
	Dominik Horb	3.00
Feature #106: Add Imagemagick to convert to different formats		10.00
	Dominik Horb	6.00
	Benjamin Oertel	4.00
Bug #107: Meeting 2012-01-09		1.50
	Tien Nguyen	1.50
Time #108: Skype Meeting 12.01.12		4.00
	Dominik Horb	2.00
	Elsa Mahari	2.00
Time #109: Cleaning up directory structure		2.50
	Dominik Horb	2.50
Feature #110: Implementing Zend_Acl for user access control		13.50
	Dominik Horb	11.50
	Benjamin Oertel	2.00
Feature #111: Improving interface workflow		8.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Benjamin Oertel	8.00
Time #115: Debbie Meeting 18.01.12		6.00
	Dominik Horb	3.00
	Elsa Mahari	3.00
Feature #116: Logger		4.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
Feature #117: Create Paging mechanism		6.00
	Benjamin Oertel	6.00
Bug #118: create Wiki page for MeetingMinutes 18.01.2012		0.50
	Tien Nguyen	0.50
Bug #119: Fixing several css issues		5.00
	Benjamin Oertel	5.00
Feature #120: Beautifying forms, tables and buttons		5.00
	Benjamin Oertel	5.00
Time #122: 23.01.12 Meeting		6.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00
	Elsa Mahari	2.00
Time #123: 30.01.12 Meeting		6.00
	Dominik Horb	2.00
	Benjamin Oertel	2.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
	Elsa Mahari	2.00
Time #128: LaTeX preparation		2.00
	Dominik Horb	2.00
Time #142: License research and inclusion		6.00
	Dominik Horb	6.00
Bug #143: FIxing tests after last refactorings		1.50
	Dominik Horb	1.50
Time #144: Reading Zend Doku for Bootstrap Plugins		4.00
	Dominik Horb	4.00
Time #145: Meeting 06.02.12		2.00
	Dominik Horb	2.00
Time #146: Basic TOC and split up of LaTeX document		1.50
	Dominik Horb	1.50
Feature #147: Preface Developers Manual		7.00
	Dominik Horb	7.00
Time #148: Documentation - Chapter Git		4.50
	Benjamin Oertel	4.50
Time #149: Meeting 04.03.12		5.00
	Dominik Horb	2.50
	Benjamin Oertel	2.50
Time #150: Writing chapter about Zend and Doctrine Framework		5.00
	Benjamin Oertel	5.00

Table A.2.: Overview By Member and Issue

Issue	Member	Total
Time #151: Writing chapter about User Interface / User Experience		6.00
	Benjamin Oertel	6.00
Time #152: Adding mockups to documentation		2.00
	Benjamin Oertel	2.00
Time #153: Team Meeting – 2012-03-10		1.50
	Benjamin Oertel	1.50
Time #154: Reading in latex documentation		3.00
	Benjamin Oertel	3.00
Feature #155: Improvements of error detection in tesseract parser		2.00
	Benjamin Oertel	2.00
Feature #156: Implementing Webservice of PlagAware		6.00
	Benjamin Oertel	6.00
Bug #157: Zend Acl – allowing/denying access to actions, not only controllers		1.00
	Benjamin Oertel	1.00
Total		548.35

Table A.3.: Overview By Sprints

Version	2011-10	2011-11	2011-12	2012-1	2012-2	2012-3	Total
none	7.00		28.75	47.00	33.00	19.30	135.05
Sprint 1 (2011-10-01 - 2011-11-14)	81.80	67.50					149.30
Product Back-Log	2.50	0.50	3.00				6.00
Sprint 2 (2011-11-15 to 2011-12-13)		33.00	72.00				105.00
Sprint 3 (2011-12-14 to 2012-01-17)			21.00	63.00			84.00
Sprint 4 (2012-01-18 to 2012-02-01)				36.00	2.00		38.00
Sprint 5 (2012-02-02 to 2012-23-02)					4.50	26.50	31.00
Total	91.30	101.00	124.75	146.00	39.50	45.80	548.35

B. Mockups

B.1. Hand-Drawn

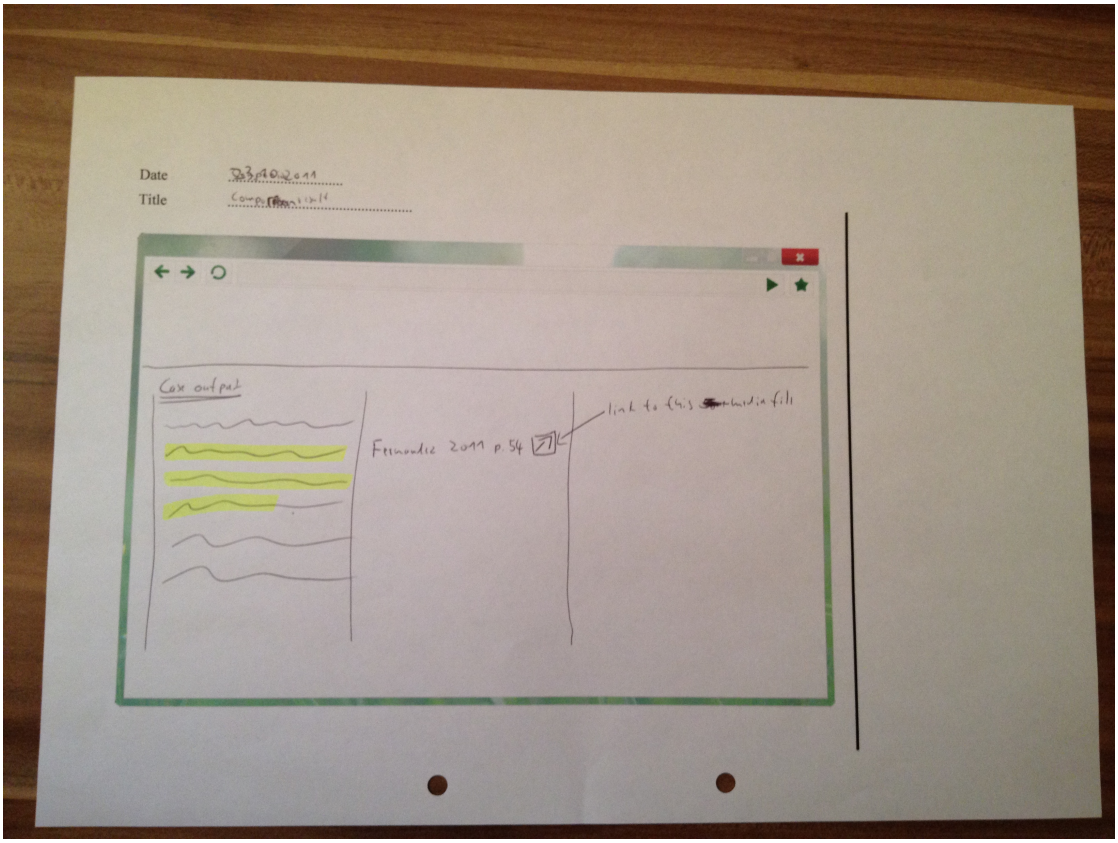

Figure B.1.: Mockup – Compare results – digitalized

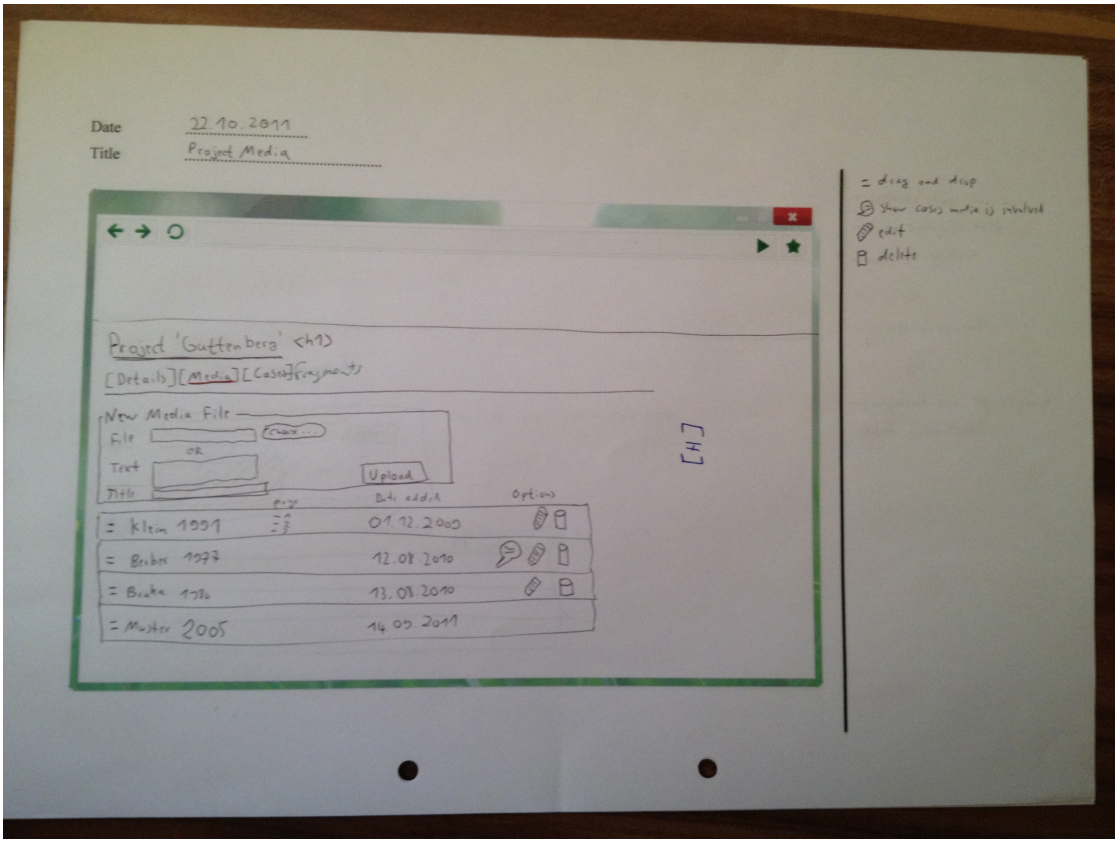

Figure B.2.: Mockup – Media list – digitalized

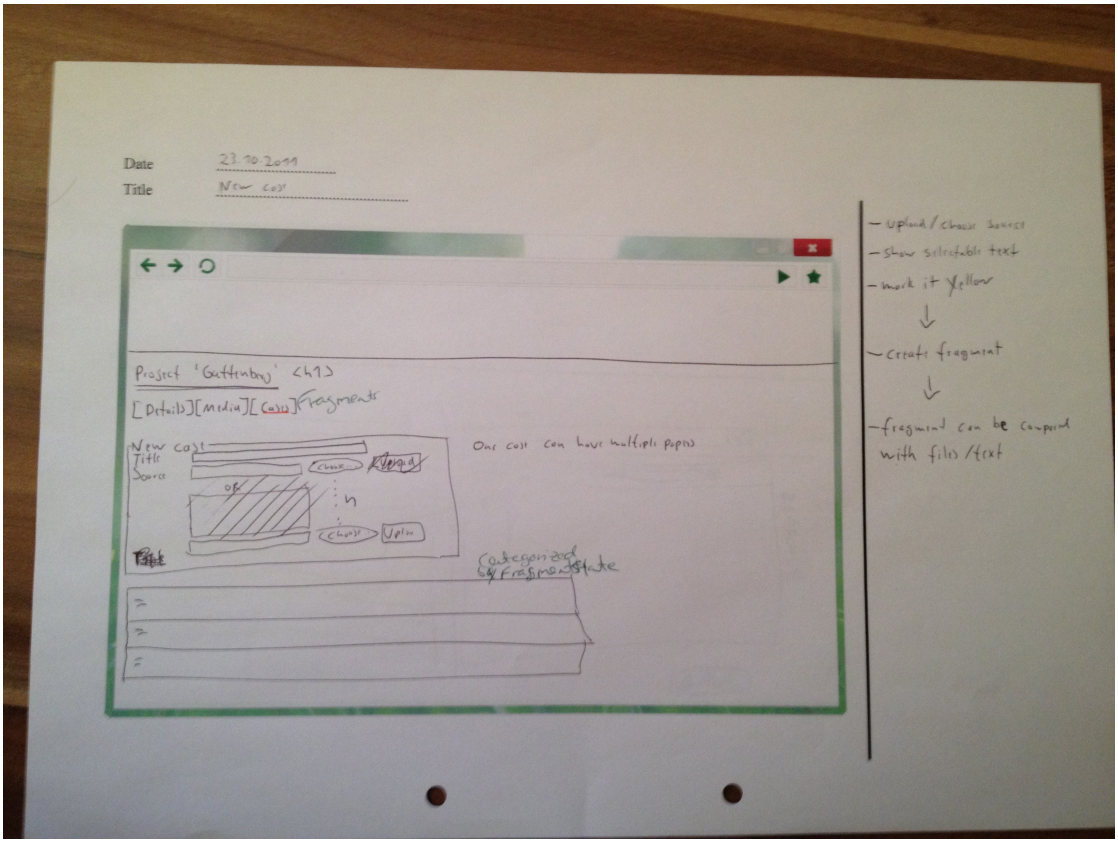

Figure B.3.: Mockup – New case – digitalized

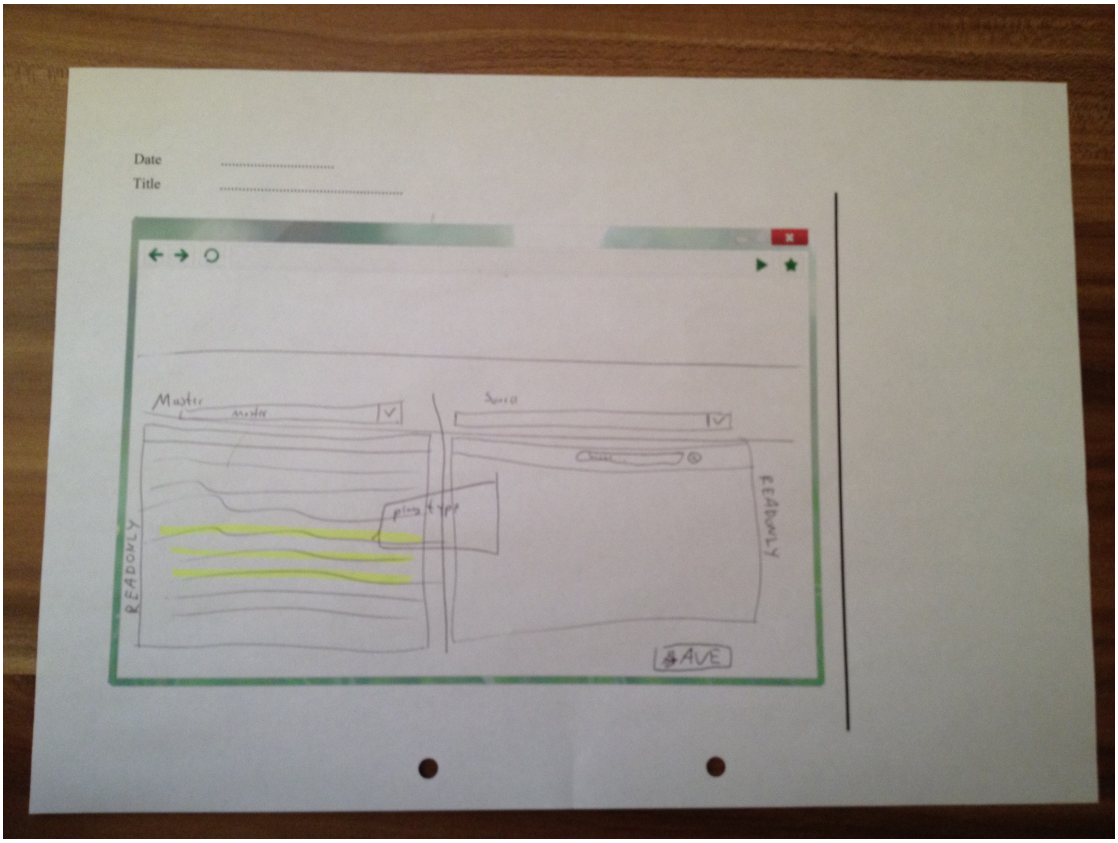

Figure B.4.: Mockup – New fragment – digitalized

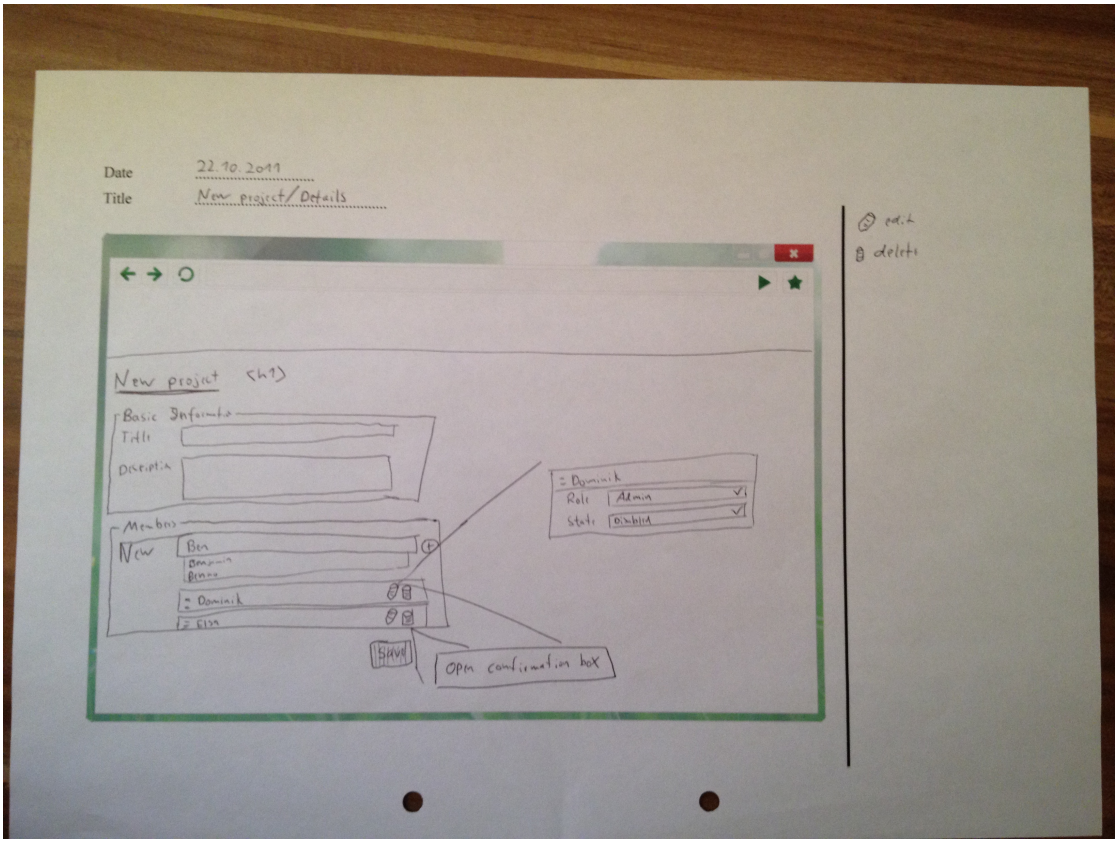

Figure B.5.: Mockup – New project – digitalized

B.2. Digitalized

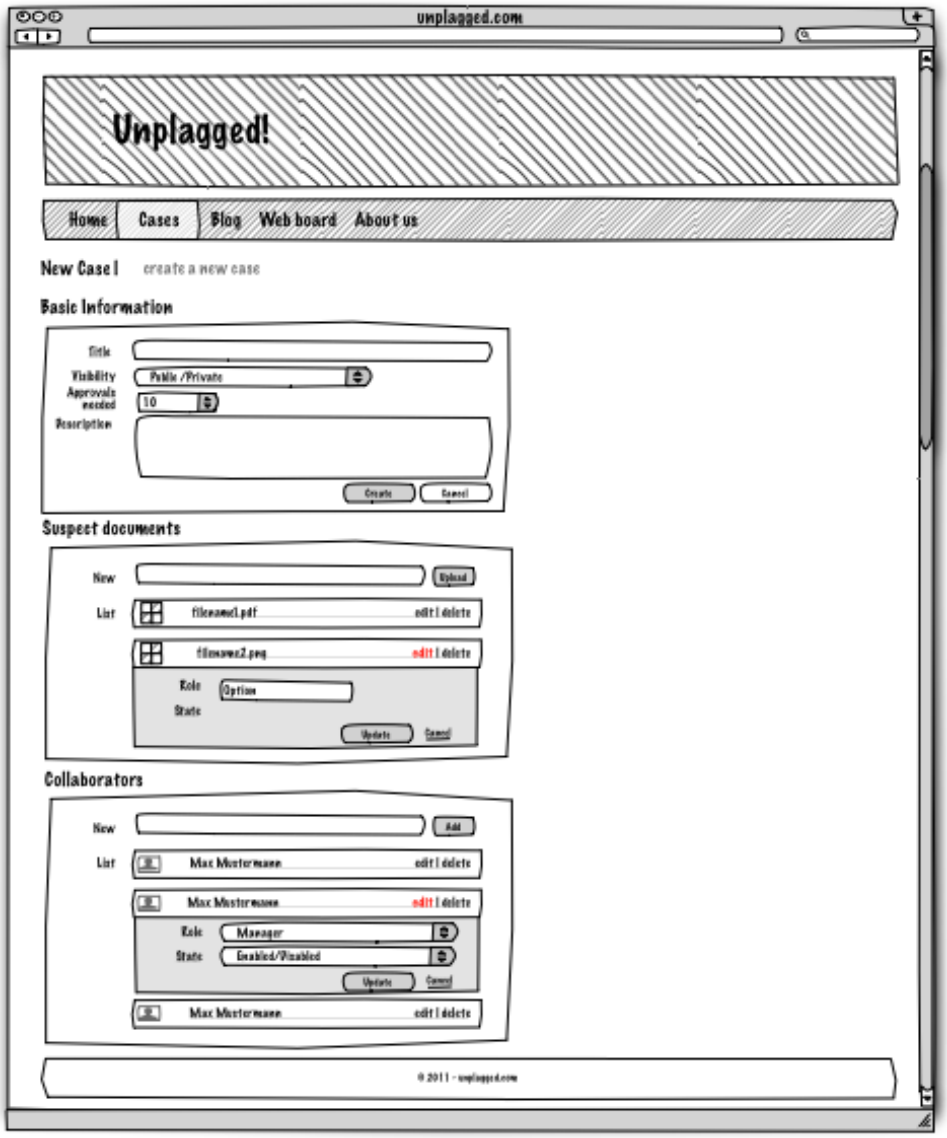

Figure B.6.: Mockup – New case – digitalized

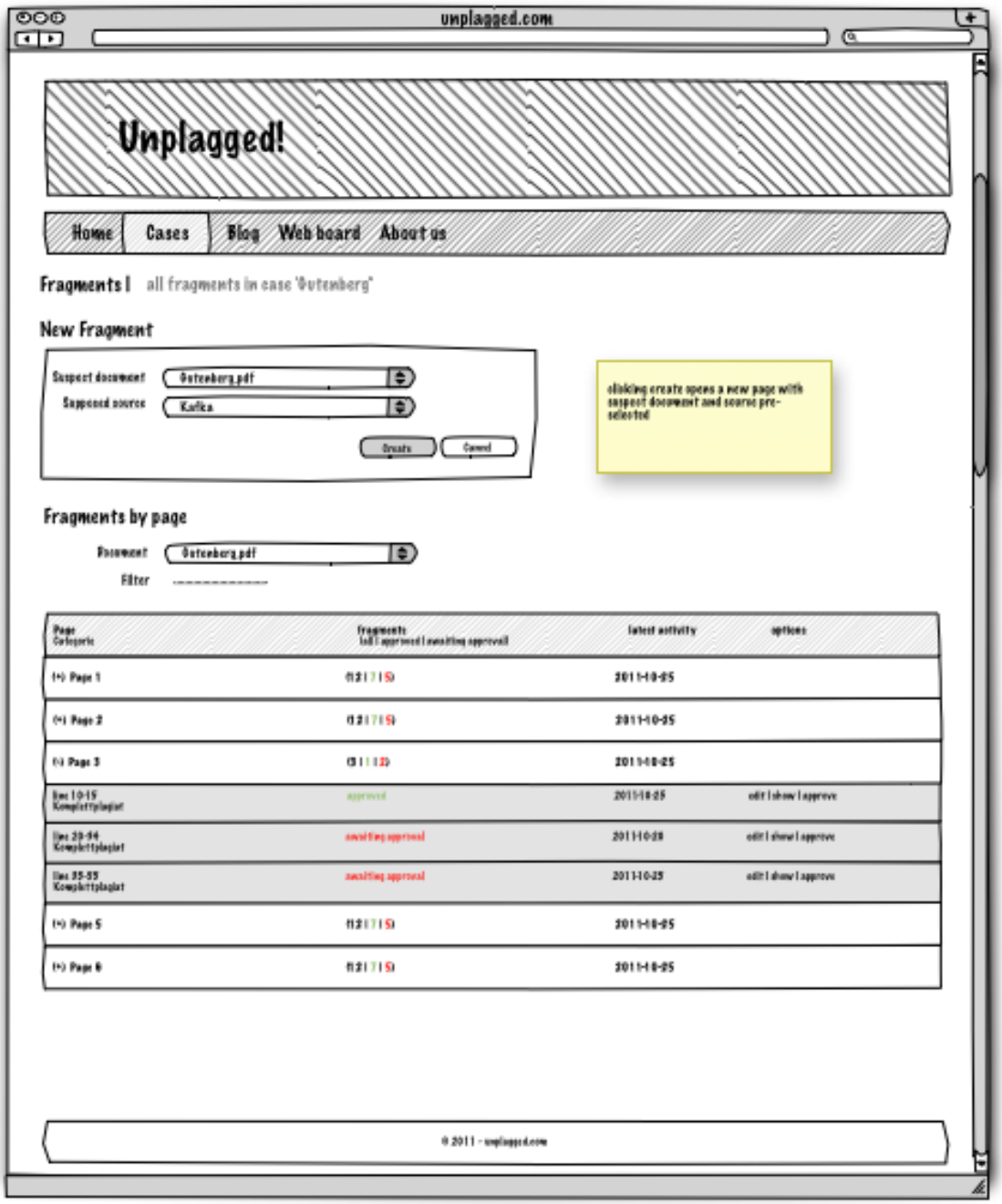

Figure B.7.: Mockup – List fragments – digitalized

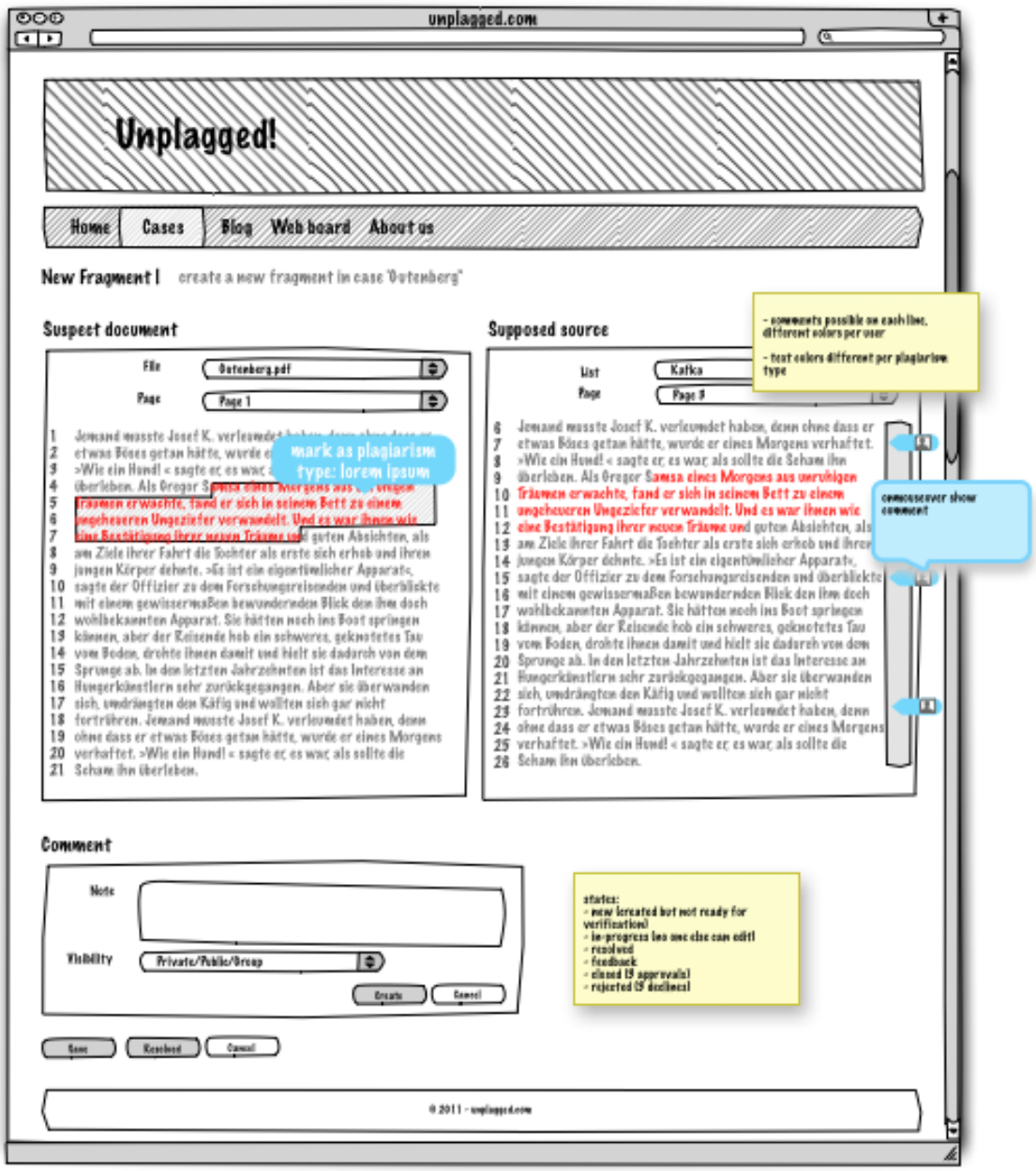

Figure B.8.: Mockup – New fragment – digitalized

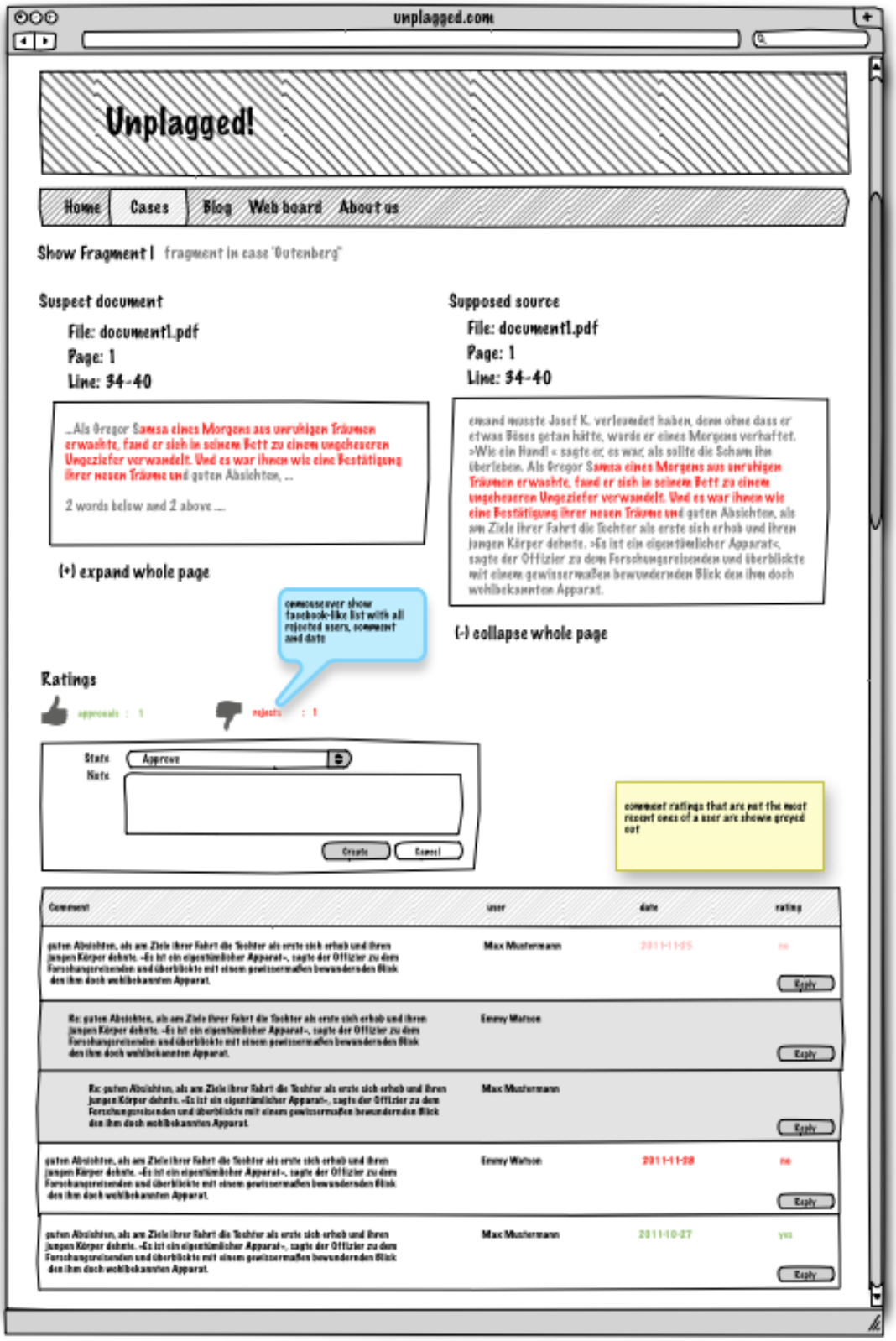

Figure B.9.: Mockup – Show fragment for approval – digitalized

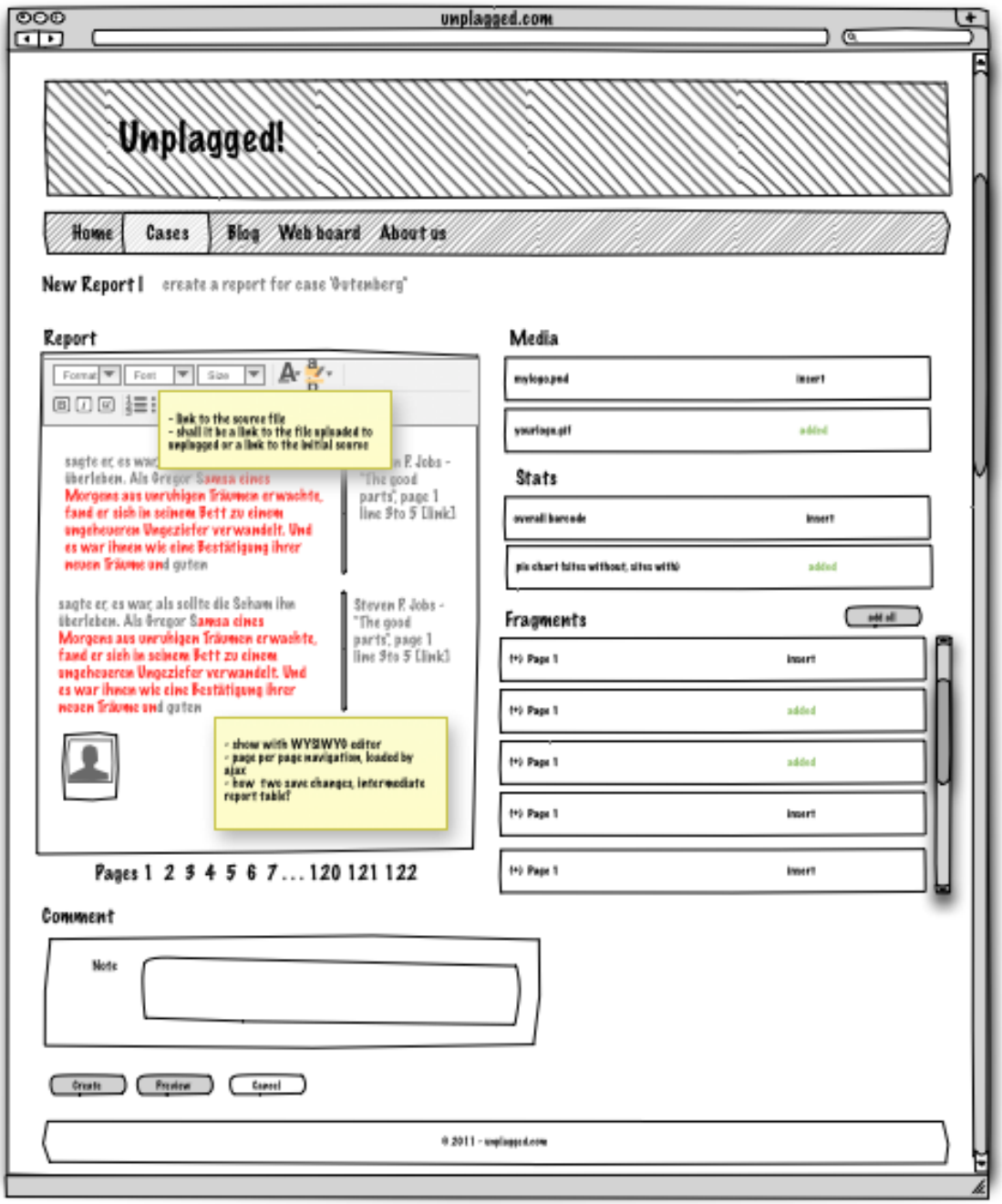

Figure B.10.: Mockup – New report – digitalized

Bibliography

[Apigen 2012] Apigen: *api documentation generator for PHP 5.3+*. http://apigen.org/, 2012. – [Online; accessed 08.03.12]

[Built With Trends 2012] Built With Trends, dummy: *jQuery Usage Trends*. http://trends.builtwith.com/javascript/jQuery, 2012. – [accessed online 01.03.12]

[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]

[Heise 2012] Heise: *API-Dokumentation mit NetBeans und ApiGen*. http://www.heise.de/developer/meldung/API-Dokumentation-mit-NetBeans-und-ApiGen-1410250.html, 2012. – [Online; accessed 08.03.12]

[Marcotte 2011] Marcotte, Ethan: *Responsive Web Design*. A Book Apart, 2011. – 150 S.

[PHPDocumentor 2012] PHPDocumentor: *phpDocumentor: The complete documentation solution for PHP*. http://www.phpdoc.org/, 2012. – [Online; accessed 08.03.12]

[Spiegel-Online 2011] Spiegel-Online: *Streit über VroniPlag-Gründer "Verprellter Liebhaber oder SPD-Mitglied, das ist egal"*. http://www.spiegel.de/unispiegel/wunderbar/0,1518,778626,00.html, 2011. – [Online; accessed 26.02.12]

[UEfAP 2012] UEfAP: *Avoiding plagiarism*. http://www.uefap.com/writing/plagiar/plagiar.htm, 2012. – [Online; accessed 16.03.12]

[Weber-Wulff 2011] Weber-Wulff, Dr. D.: *Master Project: Plagiarism Detection Cockpit*. http://www.f4.htw-berlin.de/~weberwu/classes/HTW/projects/Plagiarism-Detection-Cockpit.shtml, 2011. – [Online; accessed 26.02.12]

[Wikipedia 2011] Wikipedia: *Citation (disambiguation) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Citation_(disambiguation)&oldid=432313019. Version: 2011. – [Online; accessed 20-March-2012]

[Wiredprof 2010] Wiredprof: *Plagiarism, Paraphrasing, And Scholarly Integrity*. http://www.wiredprof.com/100/lectures/Plagiarism.htm. Version: 2010. – [Online; accessed 20-March-2012]

[Zeldman u. Marcotte 2010] Zeldman, Jeffrey ; Marcotte, Ethan: *Designing with web standards*. 3rd Ed. New Riders, 2010