User Requirements Document

2IPE0, Software Engineering Project - Q4 (2022)

Group 3

Full Name	Student ID
Bas Arendsen	1342282
Pieter van Beerendonk	1404687
Thomas Heij	1448366
Justin Kuppen	1262033
Mark Langedijk	1246976
Léniek Maas	1423878
Rick Overmeer	1424505
Isabel Rutten	1413384
Jordy Verhoeven	1001249

Project Managers: Robin Zelders, Rick Stolk
Supervisor: Huub de Beer
Customer: Rianne Conijn

Eindhoven, May 12, 2022

Abstract

This document is the User Requirements Document for Writing Dashboard. Writing Dashboard is a web-based interactive dashboard that provides students with insights into the progress on their writing skills during their stay at the TU/e. Writing Dashboard also provides researchers with insights into the actions students and participants take on the dashboard. Writing Dashboard provides the means for its users to upload their papers and essays. It will provide the user with insights about four different writing skills over their uploaded documents: language and spelling, source integration and content, cohesion and structure. These insights are provided in the form of bar and line graphs, as well as explanations about why a certain score is given.

Contents

1 Introduction **6**
1.1 Purpose . . . 6
1.2 Scope . . . 6
1.3 List of Definitions . . . 7
1.3.1 Definitions . . . 7
1.3.2 Abbreviations . . . 9
1.4 List of References . . . 9
1.5 Overview . . . 10

2 General Description **11**
2.1 Product Perspective . . . 11
2.2 General Capabilities . . . 11
2.2.1 Login . . . 12
2.2.2 Role interfaces . . . 12
2.2.3 Document Uploading and Processing . . . 12
2.2.4 Document Feedback . . . 13
2.2.5 Researchers . . . 13
2.2.6 Participants . . . 14
2.2.7 Administrator . . . 14
2.3 General Constraints . . . 14
2.3.1 Usability . . . 14
2.3.2 Environment . . . 14
2.3.3 Performance . . . 15
2.3.4 Reliability . . . 15
2.4 User characteristics . . . 16
2.4.1 Student . . . 16
2.4.2 Researcher . . . 16
2.4.3 Administrator . . . 17
2.4.4 Participant . . . 17
2.5 Environment Description . . . 18
2.6 Assumptions and Dependencies . . . 18
2.6.1 Assumptions . . . 18
2.6.2 Dependencies . . . 19

3 Specific Requirements **20**
3.1 Capability Requirements . . . 21
3.1.1 Account . . . 21
3.1.2 Document . . . 22
3.1.3 Document feedback . . . 24
3.1.4 Progress visualization . . . 26
3.1.5 Researcher . . . 27
3.1.6 User management . . . 29
3.2 Constraint Requirements . . . 30
3.2.1 Security . . . 30
3.2.2 Privacy . . . 31
3.2.3 Usability . . . 33
3.2.4 Environment . . . 34
3.2.5 Performance . . . 34

A Use Cases **35**
A.1 Student interface . . . 35
A.1.1 Student signs up for the application. . . . 35
A.1.2 Student logs into the application. . . . 36
A.1.3 Student uploads a document. . . . 37
A.1.4 Student deletes a document. . . . 38
A.1.5 Student sorts uploaded documents. . . . 39
A.1.6 Student updates feedback of documents. . . . 40
A.1.7 Student interacts with feedback of document. . . . 41
A.1.8 Student views their progress. . . . 42
A.1.9 Student zooms the progress. . . . 43
A.2 Researcher interface . . . 44
A.2.1 Researcher downloads the accumulated data. . . . 44
A.2.2 Researcher generates the participant account. . . . 45
A.2.3 Researcher changes the feedback model. . . . 46
A.3 Administrator interface . . . 47
A.3.1 Administrator grants a user a role. . . . 47
A.3.2 Administrator removes a role from a user. . . . 48

B Signing Page **49**

Document Status Sheet

Title	User Requirements Document	
Version	1.0	
Authors	Bas Arendsen	1342282
	Pieter van Beerendonk	1404687
	Thomas Heij	1448366
	Justin Kuppen	1262033
	Mark Langedijk	1246976
	Léniek Maas	1423878
	Rick Overmeer	1424505
	Isabel Rutten	1413384
	Jordy Verhoeven	1001249
Creation Date	April 26, 2022	
Last Modified	May 12, 2022	

Document History Log

Date	Changes
April 26, 2022	Added structure, chapters 1, 2.1-2.4 and B and started on chapters 3 and A.
April 27, 2022	Worked on chapters 3 and A. Some minor changes throughout the document.
April 28, 2022	Worked on chapters 3 and A. Some minor changes throughout the document.
April 29, 2022	Multiple changes throughout the whole document after first meeting with the customer.
May 2, 2022	Multiple changes throughout the whole document after obtaining first feedback from the supervisor.
May 3, 2022	Multiple changes throughout the whole document after obtaining first feedback from the supervisor and second meeting with the customer.
May 4, 2022	Worked on chapters 2 and 3 after obtaining first feedback from the supervisor and second meeting with the customer.
May 5, 2022	Worked on chapters 2, 3 and A after obtaining first feedback from the supervisor and second meeting with the customer.
May 6, 2022	Multiple changes throughout the whole document after obtaining first feedback from the supervisor and second meeting with the customer.
May 9, 2022	Some minor changes throughout the whole document after third meeting with customer.
May 11, 2022	Some minor changes to chapters 1 and A and worked on chapters 2 and 3 after obtaining second feedback from the supervisor.
May 12, 2022	Some minor changes throughout the whole document before having it signed.

Document Change Records

Version	Date	Changes
0.1	April 26, 2022	Initial version.
0.2	April 28, 2022	Version before second meeting with customer. Mostly focused on requirements, but all sections have been changed.
0.3	May 6, 2022	Version before third meeting with customer and after obtaining first feedback from the supervisor.
1.0	May 12, 2022	Version before signing of customer and supervisor.

1 | Introduction

1.1 | Purpose

This document contains the general description, user requirements and use cases for Writing Dashboard. These requirements were negotiated and agreed upon with the customer, Rianne Conijn. The requirements will be prioritized according to the MoSCoW method [2]. The acronym MoSCoW stands for the four different priorities a requirement can be given: Must have, Should have, Could have and Won't have. With the MoSCoW method, all stakeholders in the development of Writing Dashboard can provide their perspective on the product and together an agreement on the requirements can be reached such that all stakeholders are satisfied with (the priorities of) the requirements of Writing Dashboard. Any adjustments to these requirements should be made in consultation with both the developers and the customer.

1.2 | Scope

On behalf of Rianne Conijn, a researcher in the field of learning analytics from the Human-Technology Interaction (HTI) group by the Eindhoven University of Technology (TU/e), we, a group of Computer Science students at the TU/e, develop Writing Dashboard as our Software Engineering Project (SEP). The goal of this project is to develop a web-based interactive dashboard that provides students with insights into their writing progress during their stay at the TU/e. Writing Dashboard also provides researchers with insights into the actions students and participants take on the dashboard. The dashboard will be the place for students and participants to upload their documents, see their uploaded documents and see the automatically generated feedback given on the uploaded documents. On the dashboard, the researchers can get information about the actions students and participants take. The interactivity of Writing Dashboard comes from the fact that students and participants can interact with the feedback, i.e., they get more information about how the feedback was formed or what it indicates. The writing progress of users is a collection of writing skills such as language and style, source integration and content, cohesion and structure.

Students and participants upload the papers and essays they have written to Writing Dashboard to get feedback in different forms. These forms are interactive visualizations such as a bar graph and a line graph, but also annotations on the uploaded documents and explanations on why something is annotated. These visualizations will be different for a single document or for a student's or participant's progress over time. A bar graph will be provided for single documents which shows the scores for the different writing skills. A line graph will be provided for a student's progress over time, and it can show multiple writing skills and their changes in scores over time. Writing Dashboard will be developed in such a way that it has the possibility to have Learning Tools Interoperability (LTI). If LTI will be integrated at some moment in time, Writing Dashboard will get the possibility to be integrated within any Learning Management System (LMS) such as Canvas. LTI will most likely not be integrated within Writing Dashboard during our SEP, and therefore it will also most likely not be integrated within Canvas or any other LMS.

1.3 | List of Definitions

This section lists important definitions and abbreviations that are used throughout the entire document.

1.3.1 | Definitions

Definitions	Description
User Requirements Document (URD)	Document that specifies what the user expects the software to be able to do.
MoSCoW method	Prioritization technique for managing requirements.
Human-Technology Interaction (HTI) group	Group that analyzes people's interaction with technology with the aim to better understand and improve the match between technology and its users.
Writing progress	Student's progress on writing skills such as language and style, source integration and content, cohesion and structure.
Bar graph	Graph that presents categorized data with bars where the heights of the bars indicate their values.
Line graph	Graph that presents data as points that are connected by a straight line, usually to visualize a trend over time.
Learning Tools Interoperability (LTI)	Specification that specifies a method for a learning system to communicate with external systems.
Learning Management System (LMS)	Software application that supports educators in their education by providing functionalities for announcements, grading and quizzes among others.
Canvas	LMS used by TU/e.
Natural Language Processing (NLP) tools	Tools that use artificial intelligence to make human language intelligible to machines.
Administrator interface	Interface in the dashboard only accessible by users with the administrator role in which an administrator can manage user roles.
Researcher interface	Interface in the dashboard only accessible by user with the researcher role in which a researcher can generate accounts with the participant role, see their uploaded documents and download user data.

Student interface	Interface in the dashboard in which a user can upload documents and see generated feedback and visualizations.
User Requirement for Functionalities (URF)	User requirement that describes the capabilities of Writing Dashboard.
User data	Data that is generated from a user using the application, e.g. the clicks and time spent at certain sections.
Skill category	Writing skills such as language and style, source integration and content, cohesion and structure.
Feedback model	Set of algorithms to evaluate a document and give a score for each skill category.
User Requirement for Constraints (URC)	User requirement that describes the technical side of Writing Dashboard.
Single Sign-On	Authentication scheme that allows users to log in to multiple related, yet independent software systems with a single identification.
SQL injection	Web security attack that interferes with queries that are made to a database.
Cross-Site Scripting	Web security attack that injects malicious scripts on a website.
GDPR	Regulation on data protection and privacy in the European Union.

1.3.2 | Abbreviations

Abbreviation	Description
TU/e	Eindhoven University of Technology.
HTI	Human-Technology Interaction.
LTI	Learning Tools Interoperability.
LMS	Learning Management System.
URD	User Requirements Document.
NLP	Natural Language Processing
MB	Megabyte
PDF	Portable Document Format.
SDD	Software Design Document.
LTS	Long-Term Support.
XML	Extensible Markup Language.
DOCX	Office Open XML Document.
URF	User Requirement for Functionalities.
TXT	Text File.
ODT	OpenOffice Document File.
RTF	Rich Text Format.
TEX	LaTeX Source Document File.
CSV	Comma-seperated values.
URL	Uniform Resource Locator.
URC	User Requirement for Constraints.
GDPR	General Data Protection Regulation.
ASCII	American Standard Code for Information Interchange.

1.4 | List of References

[1] *2018 Reform of EU Data Protection Rules*. European Commission. May 25, 2018. URL: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.

[2] Andrew Craddock et al. *The DSDM Agile Project Framework for Scrum*. 2012.

[3] Grammarly. *About Us*. URL: https://www.grammarly.com/about. (accessed: 04-05-2022).

[4] LanguageTool. *Development*. URL: https://languagetool.org/dev. (accessed: 04-05-2022).

[5] LSD Software. *Read Aloud: A Text to Speech Voice Reader*. URL: https://chrome.google.com/webstore/detail/read-aloud-a-text-to-spee/hdhinadidafjejdhmfkjgnolgimiaplp#:~:text=To%5C%20use%5C%20Read%5C%20Aloud%5C%2C%5C%20navigate,read%5C%20before%5C%20activating%5C%20the%5C%20extension. (accessed: 26-04-2022).

[6] StatCounter. *Browser Market Share Worldwide — Statcounter Global Stats*. URL: https://gs.statcounter.com/browser-market-share. (accessed: 26-04-2022).

[7] Eindhoven University of Technology. *Language Courses and Academic Writing Skills*. URL: https://educationguide.tue.nl/career-skills/language-courses-and-academic-writing-skills/?L=2. (accessed: 26-04-2022).

[8] Eindhoven University of Technology. *Power Bi Report - Enrollment TU/e students*. URL: https://app.powerbi.com/view?r=eyJrIjoiMWQ5ZTdlODQtNWUxMC00NWY5LWExNjEtNzc2ZGNjYzg1MmI4IiwidCI6ImNjN2RmMjQ3LTYwY2UtNGEwZi05ZDc1LTcwNGNmNjBlZmM2NCIsImMiOjh9. (accessed: 26-04-2022).

[9] Tyler Walters. *Editsaurus*. URL: https://editsaurus.tylerwalters.com/. (accessed: 04-05-2022).

[10] Bang Wong. "Color blindness". In: *Nature Methods* (2011). URL: https://www.nature.com/articles/nmeth.1618.pdf.

1.5 | Overview

This User Requirements Document (URD) is organized as follows.

Chapter 2 provides a general description of the product that will be developed. This chapter is important because it highlights why Writing Dashboard is being developed and what it is capable of. In this chapter, the relation of Writing Dashboard to other (similar) systems will be described first. Secondly, the capabilities of Writing Dashboard will be listed and grouped according to different functionality categories, followed by the constraints. Together with the previous subsection, this will give a clear indication of what Writing Dashboard can and cannot do. Afterwards, the different types of users that will be using Writing Dashboard are described along with their characteristics and goals. Finally, the environment of Writing Dashboard is given and explained, which ends with the assumptions that are made and the different dependencies Writing Dashboard has to function according to its specification.

In chapter 3 the capabilities and constraints of Writing Dashboard will be described more formally. All the capabilities and constraints of Writing Dashboard will be listed as specific requirements along with their priority as agreed upon by both the customer and development team.

Appendix A lists a multitude of the most common use cases that will most likely be encountered during the usage of Writing Dashboard. This provides the reader with a clear indication of what to expect from users that are using Writing Dashboard.

Lastly, appendix B is the signing page where the customer and the supervisor are ought to sign.

2 | General Description

2.1 | Product Perspective

The TU/e has a wide range of learning goals. Next to teaching their students knowledge and engineering skills, they train students on basic professional skills that are needed in every profession. Writing is one of the professional skills that is assessed during every bachelor and master. This skill is practiced and assessed in a few selected courses of the bachelor. During these courses, students get the opportunity to get feedback, but outside these courses, students get less extensive feedback on their writing skills. Teachers only have limited time to give feedback on assignments, and mainly give feedback on the content of an assignment and don't give broad feedback on writing.

Writing Dashboard gives feedback on the writing skills of students and participants by using natural language processing (NLP) tools to analyze their uploaded documents. Students can get feedback on four different writing skills:

- language & style;
- source integration & content;
- cohesiveness;
- structure.

This feedback can give a student a better insight in their writing skills and progress in every course. This is achieved by providing the user with explanations about why a certain score is given.

On the internet, multiple tools can be found that also that make use of NLP to give feedback on pieces of text. For example, Grammarly is a website where users can input their text and get feedback on their writing [3]. Grammarly has more than 600 team members working on their tool and finding new ways to give better feedback to their users. Users can get basic writing suggestions with a free plan. Grammarly has a paid plan for users who want more advanced feedback. But, Grammarly isn't open-source. Writing Dashboard is free to use and open-source. The license used is GNU AGPLv3 and if this SEP is completed, the project will be released to the public on GitHub.

There are free and open-source alternatives for Grammarly, such as LanguageTool and Editsaurus [4, 9]. These alternatives are useful, but don't give the user insight in their writing progress over time. Writing Dashboard gives students insights in their writing progress over time by visualizing the progress of the different writing skills.

Next to that, Writing Dashboard gives researchers the possibility to retrieve user data. Researchers can use this data to improve the tool and get insights on how students use Writing Dashboard to improve their writing skills. Researchers can get basic user data from students and can get more detailed user data from paid participants using Writing Dashboard.

2.2 | General Capabilities

Writing Dashboard aims to improve the writing skills of students. This is done by allowing them to upload their documents, analyzing their files using methods used in NLP, and giving feedback on their writing skills. Students can view their progress during their studies and get insights in their writing skills. This is described further below.

Researchers can use Writing Dashboard to gain insights in how students and participants make use of it. Researchers use these insights to improve the tool for students and gain insights in how

students and participants use Writing Dashboard. Researchers can use participants to gain more insights, which will be explained further on.

Writing Dashboard is a web-based dashboard that gives student insights in their writing skills. In this section, a summary will be given of the major functions that Writing Dashboard will perform.

2.2.1 | Login

When a user navigates to the Writing Dashboard web application in their browser, they will reach the landing page. On the landing page, a user can choose to log in, if they already have an account, or choose to sign up, if they don't have an account yet. All users, except for participants which are described in a later section, can log in using their TU/e email address and a password. Participants login with a username and password.

2.2.2 | Role interfaces

After a user logs in, they see the home page. On this page, users can navigate to different pages depending on their role. Students and participants have access to the student interface. In the student interface they can go to the pages to upload documents, see uploaded documents, see feedback on the uploaded documents and see their writing progress over time. Researchers have access to the researcher interface as well. In the researcher interface it is possible to go to the pages to download data, make research projects and generate participants and get information about how to adjust and change the feedback models. Finally, administrators have access to the administrator interface as well. In the administrator interface, administrators can go to the user management page.

2.2.3 | Document Uploading and Processing

Users can upload documents using the standard way of file uploading in web browsers. Users can upload one file at a time, potentially multiple files at a time. These files can be one of these common file types: PDF, TXT or DOCX. Possibly other files types such as ODT, RTF or TEX will be supported. Uploaded files are stored in the database. Users can delete their files if they want to. The maximum file name length for an uploaded document is 256 characters. This is not a big limitation though, as most filenames are much shorter than this. If a user wants to upload a file with a file name longer than 256 characters, the system will give an error.

The application uses NLP tools to gain automated insights in the quality of the writing. The application makes use of open source NLP tools. NLP tools use different techniques to derive information from text. We determine the quality of an uploaded document by considering four different writing skills: language and style, source integration and content, cohesion and structure. The tools and models that we will use are:

- language-check, for language and style,
- NLTK, for structure and content,
- TAACO, for cohesiveness,
- Latent Dirichlet Allocation, for content.

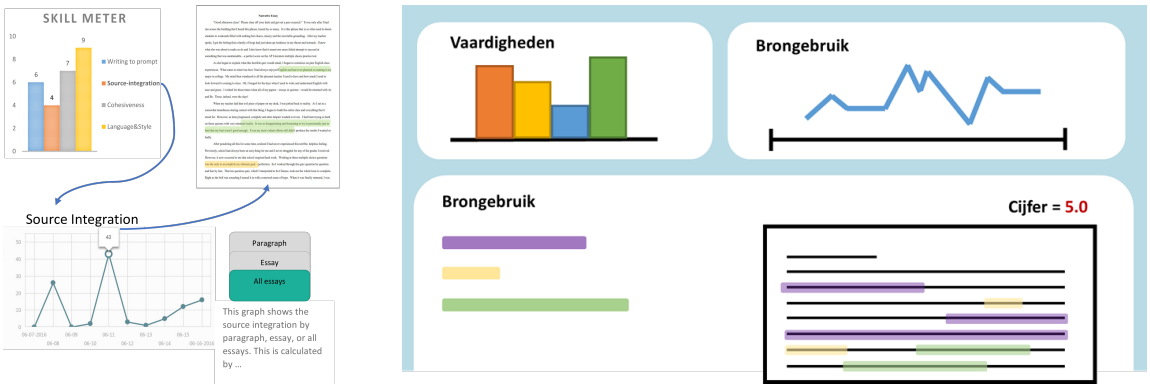
We use information that we get from these models to provide a user with feedback on their uploaded documents. This feedback is further explained in the next section.

2.2.4 | Document Feedback

The feedback derived by the NLP tools are visualized on the dashboard.
Here, a user gets feedback on documents to get insights in the writing skills in one document, but also insights in how their writing skills progressed over time. The dashboard gives the following feedback:

- For every document, a user gets feedback on the four writing skills as described earlier. Users are given a score for every writing skill.
- Students can view their document. In this view, text with feedback is highlighted. The highlighted text is colored to distinguish between different writing skills. Highlighted text is clickable, the student gets an explanation on the feedback when they click on highlighted text. Highlighted text can overlap, in this case all explanations will be shown in their corresponding color.
- Students can get an insight in their progress over time by viewing a line graph that displays scores of selectable writing skills for their documents over time. This time is the time they added as metadata to their uploaded documents.

Two prototypes from the client of the interface for this feedback can be seen in figure 2.1.



(a) Prototype **(b)** More abstract prototype

Figure 2.1: Prototype for Student Feedback

2.2.5 | Researchers

A researcher can get insights in how students and participants use the tool. This can be used to support and enhance the learning process of students. This is done via the researcher interface. Writing dashboard gathers the following data when students give permission: time, position of a click and document-id. This data can be downloaded by researchers as a CSV file. Students that don't give permission for their data being collected will still be able to use Writing Dashboard. Researchers are able to filter on a research project to only download user data from the participants in that research project.

Researchers can make changes to the tools that give feedback. This is done by changing the code of this tool in a dedicated folder on the server. The received user data can give insights on how useful a certain feedback model is for students.

Researchers are able to try outcomes of the dashboard, as they are able to see the student interface.

2.2.6 | Participants

A researcher can make use of participants. The researcher is able to make accounts with the participant role. Participants log in using a username and a password. Participants have access to the same interface and features as students, but researchers can access more data from participants. A researcher is able to access the uploaded documents and the feedback generated on those documents of participants. Researchers aren't able to do this for students.

Researchers are able to make a research project in which accounts with the participant role can be generated. The researcher has the possibility to extract the data from this group only. The researcher can select another feedback model that will be used on a research project.

2.2.7 | Administrator

Administrators have access to the administrator interface. Via the administrator interface, administrators are able to change the roles of accounts. They can add and remove the researcher and administrator role of an account. Additionally, they can remove student accounts, for example when they are no longer part of the TU/e.

2.3 | General Constraints

There are constraints on the functionalities of the product, which will be discussed in this section.

2.3.1 | Usability

It is important that Writing Dashboard can be used by as many people as possible. On the one hand, Writing Dashboard should not make it difficult for users to find where to upload documents, view feedback and interact with visualizations. This will be achieved by having clear indicators of new locations in the web application in the form of labelled buttons. On the other hand, Writing Dashboard should also make it easy for its users to interpret and understand the visualizations and feedback. This is achieved by providing the user with explanations of both the visualizations and the feedback. The explanations are adjusted to the user's document. For example, the explanation of the language and style writing skill might inform a user that a specific word is spelled wrong. These explanations are part of the feedback model and can be changed by a researcher along with the feedback model. If a new feedback model gets added, explanations will have to be added to that specific feedback model.

An issue might arise when people that have visual disabilities such as (partial) color blindness or bad eyesight want to use Writing Dashboard. They might have trouble distinguishing different colors, or lines in a line graph. Writing Dashboard tries to improve the experience of people that have visual disabilities by using colors that are suitable for color-blind users [10]. Writing Dashboard tries to prevent the issues that people with visual disabilities can experience when interacting with lines in line graphs by showing the lines using different types of lines such as dotted, dashed or solid, or a completely different line like squares or triangles.

Finally, for people that have trouble reading text, such as people with dyslexia, there is the possibility to have text read aloud in a web browser such as Google Chrome [5].

2.3.2 | Environment

Writing Dashboard is a web-based application, it is therefore important that it is supported by commonly used browser(s). Writing Dashboard supports Google Chrome version 101.0.4951 because Google Chrome is the most commonly used browser worldwide [6]. Other browsers such

as Mozilla Firefox version 100.0, Microsoft Edge version 100.0.1185.39 or Apple Safari version 15.4 might be supported if permitted by development resources.

Since English is currently the official language at TU/e, only American English is implemented in Writing Dashboard for now [7]. However, the application could later be extended to include Dutch as well, since Dutch is the secondary language of the TU/e. It would be possible to have the interface language in Dutch, additionally the text analysis and feedback language could be in Dutch if this is implemented by a developer.

2.3.3 | Performance

With students at the TU/e totaling at around 13000, of whom about 800 have a secondary enrollment, it is imperative that Writing Dashboard is able to handle a great many users [8]. Writing Dashboard should be able to have at least 1500 accounts. Besides that, Writing Dashboard should be able to handle at least 15 concurrent users, which is 1% of the total amount of accounts. Depending on deadlines, which are mainly at the end of a quartile, there will be peak moments of usage. Next to that, it is expected that each user will upload at least one document, probably several. This means that a lot of documents will be processed, resulting in a huge need of processing power. This could mean that the application slows down if this power is not provided. It is possible to scale up the system if there is a lot of traffic. However, due to development resources, Writing Dashboard will not scale up the system when there is a lot of traffic.

From our own experience, every quartile we write between five and twenty essays, ten on average, ranging in size from five to fifty pages, on average about ten pages. In this case, if all students would use the software for their whole bachelor career, we would require room for about 500,000 files. Next, the size of a document relies mainly on the amount of pictures. Comparing several large Portable Document Format (PDF) files, such as previous Software Design Documents (SDD) (more than 100 pages) gives us the reasonable estimate of at most 7 MB for an extremely large document. Comparing several documents made during the bachelor of Computer Science and Engineering, gives a reasonable estimate of an average file size of 1 MB. Another consideration is the difference in size between DOCX files and PDF files, usually DOCX files are ten times larger than PDF files. Thus, an upper limit for the amount of data storage is around half a terabyte.

Note that if at some moment in time, Writing Dashboard will be integrated within Canvas or any other LMS, it should be possible to take files directly from the LMS such that the files do not have to be uploaded and stored again. Also, it should be possible for a user and all of its uploaded documents and generated feedback to be deleted after the user has not logged in for at least one year.

2.3.4 | Reliability

With Writing Dashboard, students can track their progress in writing skills over time during their stay at the TU/e and researchers can gain insights into the actions students take on the dashboard. It is important that Writing Dashboard does not lose data or change the data too much, otherwise it might become too difficult to visualize the progress in writing skills, for example. To prevent this from happening, the data should be stored, removed and used reliably. This does not only concern the uploading of documents, but also the deletion of documents or even their entire user account. A reason for the user to remove a document could be that the user uploaded the wrong document. Removing a document will also remove the generated feedback on the document. Note that if Writing Dashboard will be integrated in Canvas (or another LMS) at a moment in time after this project, it might be possible for students to directly upload files

they submitted in Canvas to Writing Dashboard. Similarly, Writing Dashboard could directly retrieve course ID's from Canvas, which makes it unnecessary to have users supply them.

2.4 | User characteristics

Writing Dashboard considers four different types of users that make use of the application: students, researchers, administrators, and participants. Every user type is associated with one role of the same name. Having a certain role gives access to a certain interface. Also, a user with the administrator role, inherits functionalities from the researcher role and a user with the researcher role inherits functionalities from the student role. Each of the user types has certain characteristics and goals that are described below.

2.4.1 | Student

The student user type is a user that has the student role. The main goal of a student user type in Writing Dashboard is to look at and interact with the provided feedback on uploaded documents, and to look at and interact with the provided visualizations of writing skills and writing skills progress over time. The student user type is able to do the following actions:

- Give and revoke consent to the collection of their own user data.
- Upload documents to Writing Dashboard.
- Delete their own documents, including generated feedback, from Writing Dashboard.
- See all of their own uploaded documents.
- See feedback into their own writing.
- See writing progress scores for their own documents.
- See writing progress of their own writing over time.
- Select which writing skill(s) to show in the progress over time graph.
- See explanations of their own visualizations.

If a student revokes consent, the already collected user data is kept, since it was collected during the time consent was given. However, after the student revokes consent, no user data of the student will be collected anymore.

2.4.2 | Researcher

The researcher user type is a user that has the researcher role in the system. The researcher user type is able to upload and delete documents and see feedback and visualizations just like the student user type because a researcher might want to research the analysis tools on their own documents. A researcher is also able to adjust and change feedback models. This is done by changing the back-end Python code. Besides this, the researcher user type is able to do actions that comply with the two main goals of a researcher: obtaining user data from students and participants, and conducting research projects. To achieve these two main goals, the researcher user type is able to do the following actions:

- See the researcher interface.
- Access the user data of students and participants.
- Download the user data.

- Make a research project.
- Generate accounts with the participant role in a research project.
- Delete accounts with the participant role that have been generated by this researcher.
- See the documents and their corresponding feedback of the generated participant accounts.
- Remove a research project.
- Change the feedback model.

A researcher will have access to the results of a research project until the research project is removed by the researcher who made it. Note that researcher A does not have access to the generated participants and generated feedback of documents generated participants upload of researcher B.

2.4.3 | Administrator

The administrator user type is a user that has the administrator role in the system. This user's main goal is to manage other users. Since Writing Dashboard is developed for users from the TU/e, an administrator's job also concerns removing users who are no longer part of the TU/e. To achieve this goal, the administrator user type is able to do the following actions:

- See the administrator interface.
- Grant the researcher or administrator role to other users.
- Remove the researcher or administrator from other users.
- Delete user accounts.

2.4.4 | Participant

The participant user type is a user that has been given account credentials for an account with the participant role that has been generated by a researcher. The participant user type is able to do actions that comply with the main goal of a research: providing user data to a researcher. The participant user type is similar to the student user type, therefore the actions a participant user type is able to do are the same as the actions a student user type is able to do. However, the participant user type has some different characteristics, which are the following:

- The participant user type cannot register for an account with the participant role, instead account credentials are provided for this account.
- The researcher can see the uploaded documents and the corresponding feedback on those documents of the participant.

2.5 | Environment Description

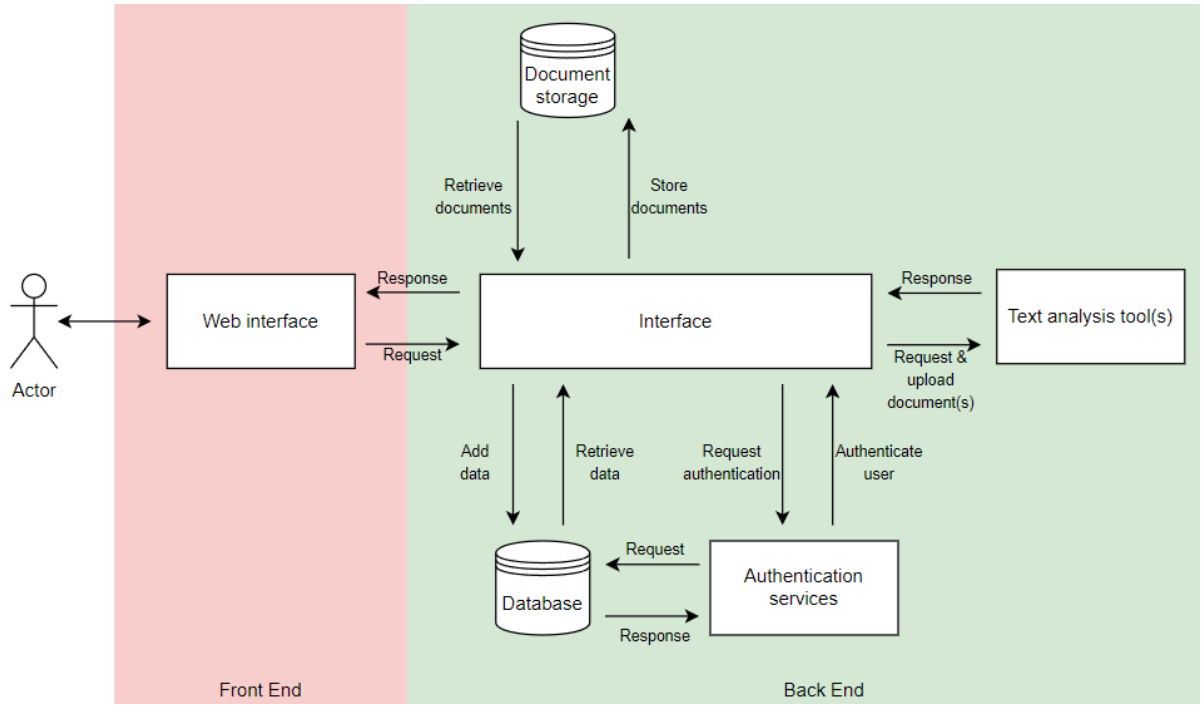


Figure 2.2: Environment model of the web application

The environment model can be seen in Figure 2.2. The main components of the environment are a front-end, back-end, including a connection to various tools to analyze texts, a database, and a second disk storage to store all the uploaded text files. This second storage is chosen to be a disk storage because it allows for quick file access and relatively large file sizes in comparison to a (No)SQL-database.

The web application works by first having a user be logged in, via the authentication services. Then, a user can upload their document to be analyzed, in a DOCX, TXT or PDF file format, optionally the ODT, RTF or TEX file formats will be supported. The uploaded document(s) will be uploaded to the document storage. After this upload, the back-end will be notified and will start using one of the built-in text analysis tools to analyze this text. The tools are implemented in such a way, that an extra tool to analyze texts can be added without much effort. This can be done by changing the back-end Python code and then restarting the server. The results of these analyses will be saved to a database and displayed to the user in the form of bar and line graphs, furthermore a researcher can also see the results for all students assigned to them, as defined in the database.

The application can be accessed using Chromium-based browsers (i.e., Chrome. Edge, etc.) and Firefox on a system with the Windows 10/11 or Ubuntu Linux 20.04+ operating system. The server is intended to run on Linux, for example Ubuntu server 22.04 LTS.

2.6 | Assumptions and Dependencies

2.6.1 | Assumptions

The web application assumes the following statements to be true in order to function according to its specification.

- The external server on which the tool is hosted, is always available and if the same code (functions/methods) is run on the same input, it will return the same result.
- A user uses Windows 10/11 or Ubuntu Linux 20.04+ to visit the website.
- The database system is available at all times.
- The user is able to understand basic English.

2.6.2 | Dependencies

In order to function according to its specification, the web application depends on the following:

- External Linux server provided by TU/e to host the web application and the database.

3 | Specific Requirements

This chapter contains the requirements as agreed upon with the customer. The requirements are ordered according to the MoSCoW model [2]. The MoSCoW model uses special methodology to define priorities of requirements as follows.

Priority	Description	Abbreviation
Must Have	Requirements that are fundamental to the solution.	M
Should Have	Important requirements for which there is a short-term workaround.	S
Could Have	Requirements that can more easily be left out.	C
Won't Have	For requirements that can be included in later development.	W

These priorities indicate the order in which the requirements will be met. All requirements with the priority 'Must Have' will be implemented in the project as they are fundamental to the solution. The 'Should Have' priority requirements will most likely be met. Some of these can be omitted if there is not enough time in the project. The requirements with priority 'Could Have' are not as necessary as the two above, so they will only be implemented when time permits. The 'Won't Have' priority requirements are the least-critical requirements, which will not be implemented during the project. They can of course be reconsidered for implementation in a later version of the application.

3.1 | Capability Requirements

This subsection contains the capability requirements. These requirements describe the capabilities that the system must provide to satisfy the needs of a stakeholder. The identification these requirements use is User Requirement for Functionalities (URF).

3.1.1 | Account

ID	Requirement	Priority
URF 1.0	Upon loading the front end of the website in the browser, the system shall display the landing page to the user.	M
URF 1.1	The user shall register itself with an email address ending in 'tue.nl' and a password.	M
URF 1.2	The user shall confirm its email address with a confirmation email before it can use the tool.	S
URF 1.3	The system shall save the email address and a password hash of the registered user to the database.	M
URF 1.4	When the user is initially registered, the system shall grant the user the student role.	M
URF 1.5	If the user is logged into the application, the system shall display the home page to the user.	M
URF 1.6	The system shall ask student accounts for consent to record their user data.	M
URF 1.7	The system shall ask participant accounts for consent to record their user data and share their documents.	M
URF 1.8	The system shall provide the means for users to change their email address and password.	M

3.1.2 | Document

ID	Requirement	Priority
URF 2.0	The system shall provide the means to upload a document of type PDF.	M
URF 2.1	The system shall provide the means to upload a document of type TXT.	M
URF 2.2	The system shall provide the means to upload a document of type DOCX.	M
URF 2.3	The system shall provide the means to upload a document of type ODT.	C
URF 2.4	The system shall provide the means to upload a document of type RTF.	C
URF 2.5	The system shall provide the means to upload a document of type TEX.	C
URF 2.6	The system shall give an error when the user tries to upload documents where the extension does not match the file type.	M
URF 2.7	The system shall give an error when the user tries to upload documents of a type other than PDF, TXT, DOCX, ODT, RTF and TEX.	M
URF 2.8	The system shall provide the user the means to delete one of their documents.	M
URF 2.9	The system shall provide the user the means to delete multiple documents at once.	S
URF 2.10	The system shall provide the user the means to update one of their documents by replacing it with a new file.	S
URF 2.11	The system shall provide the means to upload a document with a size between 0 and 10 MB.	M
URF 2.12	The system shall provide the means to upload a document with a size between 0 and 20 MB.	C
URF 2.13	The system shall provide the means to upload a document with file name up to 256 characters.	M
URF 2.14	The system shall give an error if a file name is not ASCII encoded.	M
URF 2.15	The system shall provide the means to upload multiple documents at the same time.	C
URF 2.16	The system shall save the uploaded documents to the database.	M
URF 2.17	The system shall provide the user the means to add the submission date as metadata to the document.	C
URF 2.18	The system shall provide the user the means to add the course code with a maximum length of 16 characters as metadata to the document.	C

ID	Requirement	Priority
URF 2.19	When the user does not insert the metadata of the document, the system shall set the submission date to the current date and the course code to Null.	C

3.1.3 | Document feedback

ID	Requirement	Priority
URF 3.0	The system shall display a list of all previously uploaded documents from the corresponding user.	M
URF 3.1	The system shall provide the means to sort all displayed documents by the metadata submission date.	M
URF 3.2	The system shall provide the means to sort all displayed documents by file name.	S
URF 3.3	The system shall provide the means to sort all displayed documents by file format.	S
URF 3.4	The system shall provide the means to sort all displayed documents by course code.	S
URF 3.5	The system shall provide the means to sort all displayed documents by cohesion score.	C
URF 3.6	The system shall provide the means to sort all displayed documents by structure score.	C
URF 3.7	The system shall provide the means to sort all displayed documents by language & style score.	C
URF 3.8	The system shall provide the means to sort all displayed documents by source integration & content score.	C
URF 3.9	The system shall provide the means to filter the displayed documents by searching for file names.	C
URF 3.10	When the user clicks on a document, the system shall display the document.	M
URF 3.11	When the user clicks on a document, the system shall display the corresponding feedback score for every skill category.	M
URF 3.12	When the user clicks on a document, the system shall display a bar graph that displays a score for every skill category.	M
URF 3.13	When the user uploads a document, the system shall generate feedback based on the current feedback model.	M
URF 3.14	The feedback shall be stored in the database.	M
URF 3.15	The system shall provide the option for the researcher to change the feedback model.	S
URF 3.16	When the researcher updates a feedback model, the feedback of previously uploaded documents shall be unchanged.	M
URF 3.17	The system shall allow researchers to switch between different versions of the feedback model.	C
URF 3.18	The system shall store in the database which feedback model version is used on each document.	S

ID	Requirement	Priority
URF 3.19	When a user is viewing a document which has feedback based on a feedback model that is not the current feedback model, the system shall notify the user.	S
URF 3.20	When a document is evaluated using a feedback model that is not the current feedback model, the system shall provide the option to update the feedback for this document using the current feedback model.	S
URF 3.21	The feedback shall include metrics related to the language & style of the document.	M
URF 3.22	The feedback shall include metrics related to the source integration & content of the document.	M
URF 3.23	The feedback shall include metrics related to the cohesiveness of the document.	M
URF 3.24	The feedback shall include metrics related to the structure of the document.	M
URF 3.25	The system shall highlight text, with the color depending on the skill category, in the document when there is feedback on the text.	S
URF 3.26	When the user clicks on highlighted text in the document, the system shall display explanations of the feedback on the corresponding text.	S

3.1.4 | Progress visualization

ID	Requirement	Priority
URF 4.0	The system shall provide a means for the user to view their progress throughout their academic career.	M
URF 4.1	The system shall display the progress based on the uploaded documents and their corresponding feedback in a line plot.	M
URF 4.2	The system shall display the progress of the language & style for a user.	M
URF 4.3	The system shall display the progress of the source integration & content for a user.	M
URF 4.4	The system shall display the progress of the cohesiveness for a user.	M
URF 4.5	The system shall display the progress of the structure for a user.	M
URF 4.6	When the user selects a skill category, the system shall display the progress of that skill over time.	M
URF 4.7	When the user hovers over a point in the progress over time of a specific skill category, the system shall display the file name, date and feedback model version of the corresponding document.	M
URF 4.8	When the user selects a point in the progress over time of a specific skill category, the system shall display the corresponding document and its feedback.	S
URF 4.9	The system shall provide the option to hide specific skill categories in the visualization using checkboxes.	S
URF 4.10	The progress visualizations are accessible to the colorblind.	M
URF 4.11	The system shall provide the option to zoom in on the progress visualizations.	C
URF 4.12	The system shall provide the option to zoom out on the progress visualizations.	C
URF 4.13	The system shall provide the option to select a level of zoom from predefined levels in the progress visualizations.	C

3.1.5 | Researcher

ID	Requirement	Priority
URF 5.0	The system shall record the user data of students and participants who have given consent to collection of their data.	M
URF 5.1	The user data shall include the URL of the page of each click from a user.	M
URF 5.2	The user data shall include the location on the screen of each click from a user.	M
URF 5.3	The user data shall include the timestamp of each click from a user.	M
URF 5.4	The user data shall include the document ID for each click from a user when the user is viewing a document.	M
URF 5.5	The user data shall include the keystrokes from a user on a page when not dealing with password/personal information fields.	C
URF 5.6	The researcher shall have access to the user data.	M
URF 5.7	The system shall provide the means for researchers to download all user data from all users as a CSV file.	M
URF 5.8	The system shall provide the means for researchers to download the user data as a CSV file for a selected time period.	S
URF 5.9	The system shall provide the means for researchers to download the user data as a CSV file filtered on research project name.	S
URF 5.10	The system shall provide the means for researchers to create a new research project.	M
URF 5.11	The system shall generate participant accounts with corresponding usernames and passwords.	M
URF 5.12	The system shall provide the means for researchers to generate participant accounts within a research project.	M
URF 5.13	The participant accounts created by researchers shall be connected to exactly one specific research project.	M
URF 5.14	The system shall provide the means for researchers to download the generated accounts for a research project as a CSV file.	M
URF 5.15	The system shall provide the means for researchers to remove a research project.	S
URF 5.16	The system shall provide the means for researchers to delete participant accounts.	M
URF 5.17	When researchers want to generate a participant account, the system shall provide the means for researchers to insert a project name which will be connected to that account.	S

ID	Requirement	Priority
URF 5.18	When researchers want to generate a participant account, the system shall provide the means for researchers to choose a project name from existing project names which will be connected to that account.	C
URF 5.19	When researchers have inserted and confirmed a project name, the participant account that is to be generated next shall be connected to that project name.	M
URF 5.20	When researchers have inserted and confirmed a project name, the system provides the means for researchers to insert the number of participant accounts they want to generate.	S
URF 5.21	The researcher shall have access to the documents and their corresponding feedback of participants.	M
URF 5.22	The researcher shall not have access to the documents and their corresponding feedback of students.	M
URF 5.23	The system shall provide the means for the researcher to download all documents from a single participant as a zip file.	C
URF 5.24	The student user data shall be anonymized.	M
URF 5.25	The system shall provide researchers the means to download the feedback scores as a CSV file.	C

3.1.6 | User management

ID	Requirement	Priority
URF 6.0	The user shall have exactly one role.	M
URF 6.1	There shall be an administrator role.	M
URF 6.2	There shall be a researcher role.	M
URF 6.3	There shall be a student role.	M
URF 6.4	There shall be a participant role.	M
URF 6.5	The account with the role administrator shall have access to the administrator interface.	M
URF 6.6	The account without the role administrator shall not have access to the administrator interface.	M
URF 6.7	The account with the role administrator shall have access to the researcher interface.	M
URF 6.8	The account with the role researcher shall have access to the researcher interface.	M
URF 6.9	The account with the role researcher shall have access to the student interface.	M
URF 6.10	The account with the role student shall have access to the student user interface.	M
URF 6.11	The account with the role participant shall have access to the student user interface.	M
URF 6.12	When an administrator selects a user to grant a specific role, the system shall grant the selected user that specific role.	M
URF 6.13	When an administrator selects a user to remove a specific role, the system shall remove that specific role from the selected user.	M
URF 6.14	The system shall provide the means for administrators to grant a user the administrator role.	M
URF 6.15	The system shall provide the means for administrators to remove the administrator role from a user.	M
URF 6.16	The system shall display an error message when the last administrator tries to remove its own administrator role.	M
URF 6.17	The system shall provide the means for administrators to grant a user the researcher role.	M
URF 6.18	The system shall provide the means for administrators to remove the researcher role from a user.	M

3.2 | Constraint Requirements

In this section, the constraint requirements are defined. These requirements are about the technical side of the application, which includes topics such as security, usability, and environment.

3.2.1 | Security

ID	Requirement	Priority
URC 1.0	The application shall check that the user has correct access rights before allowing access to the application.	M
URC 1.1	If the user does not provide sign-up information, namely university email address and password, the user shall not register an account.	M
URC 1.2	The user shall not register an account when the password does not contain at least 8 characters.	M
URC 1.3	The user shall not register an account when the password does not include at least 1 uppercase character.	M
URC 1.4	The user shall not register an account when the password does not include at least 1 lowercase character.	M
URC 1.5	The user shall not register an account when the password does not include at least 1 number.	M
URC 1.6	If the user provides correct login information, the user shall be logged into the application.	M
URC 1.7	If the system does not fully have the sign-up information entered, namely university email address and password, the user shall not be logged into the application.	M
URC 1.8	The system shall use Single Sign-On for logging into the application.	C
URC 1.9	The application shall allow users to log into the system via the Canvas application.	C
URC 1.10	The application shall check input fields for SQL injection.	C
URC 1.11	The application shall check input fields for Cross-Site Scripting.	C

3.2.2 | Privacy

This section is based on the General Data Protection Regulation (GDPR) [1].

ID	Requirement	Priority
URC 2.0	The system shall ask the user permission to save personal data, except for necessary sign-up information, which contain the university email address.	M
URC 2.1	The system shall ask the user permission to save user data.	M
URC 2.2	The system shall ask the user permission to save specific data again every time the processing of the data changes.	M
URC 2.3	When the user revokes permission to save personal data, user data will not be collected from that moment on.	M
URC 2.4	When the user denies permission to save personal data, user data will not be collected from that moment on.	M
URC 2.5	When the user denies permission to any data, the system shall still allow the user access to the full application.	M
URC 2.6	When the system asks the user for permission to save data, the system shall inform the user what will be done with the data.	M
URC 2.7	The system shall provide a way for users to discover what kind of data of them is saved.	M
URC 2.8	The system shall provide a way for users to discover the exact data of them that is saved.	M
URC 2.9	The system shall provide a way for users to discover what the system does with the saved data.	M
URC 2.10	The system shall provide a way for users to ask questions regarding their data, which are to be answered within a month.	M
URC 2.11	The system shall provide a way for users to ask questions regarding their data, which are to be answered within two weeks.	C
URC 2.12	The system shall provide users the option to turn user data tracking and premises on.	M
URC 2.13	The system shall provide users the option to turn user data tracking and premises off.	M
URC 2.14	The system shall provide the means for users to delete their account and with that their data.	M
URC 2.15	The system shall provide the means for users to change their email address.	M
URC 2.16	The system shall provide the means for users to change their password.	M
URC 2.17	The systems shall allow users access to the application without having to give permission for data collection.	M

ID	Requirement	Priority
URC 2.18	The system shall delete a user and its uploaded documents and generated feedback after the user has not logged in for at least one year.	S

3.2.3 | Usability

ID	Requirement	Priority
URC 3.0	A user can perform an implemented use case of moderate time without instruction within 10 minutes, which include use cases A.1.1, A.1.2, A.1.3, A.1.7, A.2.2, A.2.3, A.3.1, A.3.2.	M
URC 3.1	A user can perform an implemented use case of moderate time without instruction within 5 minutes, which include use cases A.1.1, A.1.2, A.1.3, A.1.7, A.2.2, A.2.3, A.3.1, A.3.2.	C
URC 3.2	A user can perform an implemented use case of short time without instruction within 5 minutes, which include use cases A.1.4, A.1.5, A.1.6, A.1.9. A.2.1.	M
URC 3.3	A user can perform an implemented use case of short time without instruction within 2 minutes, which include use cases A.1.4, A.1.5, A.1.6, A.1.9. A.2.1.	C
URC 3.4	The User Interface shall visually respond to interaction on implemented triggers in under 500 milliseconds.	M
URC 3.5	The User Interface shall visually respond to all interaction in under 500 milliseconds.	S
URC 3.6	The User Interface shall be consistent, i.e., act in the same way over time.	M
URC 3.7	The User Interface shall display concise page titles for clarity.	C

3.2.4 | Environment

ID	Requirement	Priority
URC 4.0	The application shall be available in English.	M
URC 4.1	The application shall be available in Dutch.	C
URC 4.2	The application shall function correctly on version 101.0.4951 of Google Chrome.	M
URC 4.3	The application shall function correctly on version 100.0 of Mozilla Firefox.	C
URC 4.4	The application shall function correctly on version 100.0.1185.39 of Microsoft Edge.	C
URC 4.5	The application shall function correctly on version 15.4 of Apple Safari.	C
URC 4.6	The application shall function correctly on version 101.0.4951.61 of Google Chrome Mobile.	C
URC 4.7	The application shall function correctly on version v17.0.1.69 of Samsung Internet Browser.	C
URC 4.8	The application shall function correctly on version 15.4 of Apple Safari Mobile.	C

3.2.5 | Performance

ID	Requirement	Priority
URC 5.0	The pages of the application shall be loaded within 100 milliseconds.	M
URC 5.1	The feedback of the documents shall be generated within 10 seconds.	M

A | Use Cases

This section describes the different use cases that are relevant for this application. For each use case, the goal, actors, preconditions, postconditions, requirements, priority and steps are reported.

A.1 | Student interface

The subsection Student interface details the use cases for the user Student.

A.1.1 | Student signs up for the application.

Goal: **The actor signs up for the web-based application using their university account.**
Actor: **Student**
Precondition: **The actor has a university account.**
Postcondition: **The actor has an account within the application.**
Requirements: **URF 1.1, URF 1.2, URF 1.3, URF 1.4, URF 1.5, URC 1.0, URC 1.1, URC 1.2, URC 1.3, URC 1.4, URC 1.5, URC 1.6**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the sign-up page of the web-based application.	
	2. The system displays the sign-up interface which requests for the sign-up information of the actor.
3. The actor inserts their sign-up information, which correspond to their university account.	
4. The actor makes a choice regarding the system saving and processing their user data.	
	5. The system checks the sign-up information.
	6. The system adds the account to the database.
	7. The system displays a confirmation message.

Alternative:
5B: In case the provided email address is not a university account, the system displays an error message. 6B: In case the database cannot be reached, the system displays an error message.

A.1.2 | Student logs into the application.

Goal: **The actor logs into the web-based application using their university account.**
Actor: **Student**
Precondition: **The actor has signed up to the application.**
Postcondition: **The actor is logged into the application and can fully access the application within their rights.**
Requirements: **URF 1.5, URF 1.6, URF 1.7, URF 6.11, URC 1.0, URC 1.6, URC 1.7**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the login page of the web-based application.	
	2. The system displays the login interface which requests for the login information of the actor.
3. The actor inserts their login information.	
	4. The system checks the login information.
	5. The system displays the home interface with the information corresponding to the actor.

Alternative:
4B: In case the login information is incorrect, the system displays an error message.
5B: In case the database cannot be reached, the system displays an error message.

A.1.3 | Student uploads a document.

Goal: **The actor uploads a document, which is an academic paper, onto the application, after which the system also saves the corresponding feedback.**
Actor: **Student**
Precondition: **The actor is logged into the application. The document is an academic paper.**
Postcondition: **The document, including its feedback, has been uploaded to the application.**
Requirements: **URF 2.0, URF 2.1, URF 2.2, URF 2.3, URF 2.4, URF 2.5, URF 2.6, URF 2.7, URF 2.10, URF 2.11, URF 2.12, URF 2.13, URF 2.15, URF 2.16, URF 2.17, URF 2.18, URF 2.19, URF 3.13, URF 3.14, URF 3.18, URF 3.21, URF 3.22, URF 3.23, URF 3.24**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the document upload page.	
2. The actor selects the button to upload a document.	
	3. The system displays a pop-up window in which the actor can select a document.
4. The actor selects the document to be uploaded.	
	5. The system retrieves the document.
	6. The system saves the document in the database.
	7. The system generates the corresponding feedback based on the current feedback model.
	8. The system displays a confirmation message.

Alternative:
2-4B: The actor drags and drops a document into the given field. 4B: The actor selects multiple documents. (URF 2.12, Could Have)
4C: The actor can also optionally add metadata to a document, namely the submission date and the course code. (URF 2.14 and URF 2.15, Could Have)
5B: In case the file cannot be retrieved, the system displays an error message.
6B: In case the file cannot be saved in the database, the system displays an error message.
7B: In case the system cannot generate feedback, the system displays an error message.

A.1.4 | Student deletes a document.

Goal: **The actor deletes a document which they have uploaded onto the application.**
Actor: **Student**
Precondition: **The actor is logged into the application. The document has been uploaded onto the application by the actor.**
Postcondition: **The document has been removed from the database of the application.**
Requirements: **URF 2.7, URF 3.0, URF 6.11**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the document page.	
	2. The system retrieves the documents of the user.
	3. The system displays the documents of the user.
4. The actor selects a document.	
5. The actor clicks the delete button.	
6. The actor confirms its deletion	
	7. The system retrieves the document.
	8. The system deletes the document and its corresponding feedback in the database.
	9. The system displays a confirmation message.

Alternative:
2B: In case the files cannot be retrieved, the system displays an error message.
4B: The actor can find a file by sorting the displayed documents. (URF 3.1-3.7, Could Have)
8B: In case the file cannot be retrieved, the system displays an error message.
9B: In case the file cannot be saved in the database, the system displays an error message.

A.1.5 | Student sorts uploaded documents.

Goal: **The actor sorts the documents in a specific order.**
Actor: **Student**
Precondition: **The actor is logged into the application. The student has uploaded at least one document.**
Postcondition: **The system sorts the uploaded documents according to the selected order.**
Requirements: **URF 3.0, URF 3.1, URF 3.2, URF 3.3, URF 3.4, URF 3.5, URF 3.6, URF 3.7, URF 6.11**
Priority: **Could Have**
Steps:

Actor actions	System actions
1. The actor navigates to the document page.	
2. The actor selects one specific ordering, from those implemented, which includes the file name, the file format, the course, and the different feedback scores.	
	3. The system displays the documents according to the selected ordering.

Alternative: -

A.1.6 | Student updates feedback of documents.

Goal: **The actor selects a button due to which the system updates the feedback of all documents that the user uploaded to that of the current feedback model.**
Actor: **Student**
Precondition: **The actor is logged into the application. The student has uploaded at least one document.**
Postcondition: **The system updates the feedback of the uploaded documents accordingly with the current feedback model.**
Requirements: **URF 3.24, URF 6.11**
Priority: **Could Have**
Steps:

Actor actions	System actions
1. The actor navigates to the document page.	
2. The actor selects the button to update all documents.	
	3. The system retrieves all documents of the actor
	4. The system deletes the current feedback of the documents.
	5. The system generates feedback based on the current feedback model for the documents.
	6. The system displays a confirmation message.

Alternative:
3B: In case the database cannot be reached, the system displays an error message.

A.1.7 | Student interacts with feedback of document.

Goal: **The actor views and interacts with the feedback of a selected document.**
Actor: **Student or Participant**
Precondition: **The actor is logged into the application. The document on which they want to receive feedback is uploaded to the application.**
Postcondition: **The system displays the document and its corresponding feedback.**
Requirements: **URF 3.10, URF 3.11, URF 3.12, URF 3.13, URF 3.21, URF 3.22, URF 3.23, URF 3.24, URF 3.25, URF 3.26**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the documents page.	
2. The actor selects the document for which they want to view feedback.	
	3. The system retrieves the document and its feedback from the database.
	4. The system displays the document and its corresponding feedback.
5. The actor views the feedback.	
6. The actor selects specific feedback.	
	7. The system displays the corresponding explanation.
8. The actor continues exploring the feedback.	

Alternative:
3B: In case the database cannot be reached, an error message is shown.
4B: In case the feedback is not based on the current feedback model, the system displays a popup in which it is suggested to update the document with the current feedback model by pressing a corresponding button. (URF 3.21 and URF 3.23, Should Have).
7B: In case the database cannot be reached, an error message is shown.

A.1.8 | Student views their progress.

Goal: **The actor views their progress regarding academic writing, based on their uploaded documents.**
Actor: **Student or Participant**
Precondition: **The actor is logged into the application.**
Postcondition: **The system displays the progress of the actor.**
Requirements: **URF 4.0, URF 4.1, URF 4.2, URF 4.3, URF 4.4, URF 4.5, URF 4.6, URF 4.7, URF 4.8, URF 4.9, URF 4.10**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the progress page.	
	2. The system retrieves the feedback for the actor from the database.
	3. The system displays the bar graph with the average score for different skill categories based on the found data.
4. The actor views the bar graph.	
5. The actor selects a skill category.	
	6. The system displays the progress over time for that specific skill category in a line chart.
7. The actor views the progress visualization.	
8. The actor interacts with the progress visualization by hovering over a point.	
	9. The system displays the title, date and feedback model version of the corresponding document.

Alternative:
2B: In case the actor has not uploaded any documents, the system displays an error message.
2C: In case the system cannot access the database, the system displays an error message.
4B: The actor can hide specific skill categories in the visualization using checkboxes. (URF 4.9)
8B: In case the user selects a document, the system shall display the corresponding documents and its feedback. (URF 4.8, Should Have)

A.1.9 | Student zooms the progress.

Goal: **The actor zooms the progress visualization, which is based on their uploaded documents.**
Actor: **Student or Participant**
Precondition: **The actor is logged into the application. The actor has uploaded at least one document.**
Postcondition: **The system zooms the progress of the actor.**
Requirements: **URF 4.11, URF 4.12, URF 4.13**
Priority: **Could Have**
Steps:

Actor actions	System actions
1. The actor navigates to the progress page.	
	2. The system retrieves the feedback for the actor from the database.
	3. The system displays the progress based on the found data.
4. The actor views the progress visualization.	
5. The actor zooms the progress visualization.	
	6. The system zooms the progress visualization according to the chosen zoom.

Alternative:
3B: In case the actor has not uploaded any documents, the system displays no progress data.

A.2 | Researcher interface

The subsection Researcher interface details the use cases for the user Researcher. The researcher interface also includes the use cases as described in the student interface (URF 6.9).

A.2.1 | Researcher downloads the accumulated data.

Goal: **The actor downloads the filtered user data that has been accumulated by tracking the users.**
Actor: **Researcher**
Precondition: **The actor is logged into the application.**
Postcondition: **The actor has downloaded the data.**
Requirements: **URF 5.0, URF 5.1, URF 5.2, URF 5.3, URF 5.4, URF 5.5, URF 5.6, URF 5.7, URF 5.8, URF 5.9 URF 5.14, URF 5.15, URF 5.16, URF 6.8, URF 6.9**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the download page.	
2. The actor sets the user data filters regarding time and users.	
3. The actor selects the button to download the user data.	
	4. The system retrieves the selected user data.
	5. The system sends a copy of the data to the actor.
6. The actor downloads the data.	

Alternative:
4B: In case the database cannot be reached, the system displays an error message.
4C: In case there is no data to be downloaded, the system displays an error message.

A.2.2 | Researcher generates the participant account.

Goal: **The actor generates a participant account which is connected to a specified project.**
Actor: **Researcher**
Precondition: **The actor is logged into the application.**
Postcondition: **The system has generated a participant account which is connected to a specified research project.**
Requirements: **URF 5.10, URF 5.11, URF 5.12, URF 5.13, URF 5.14, URF 5.18, URF 5.19, URF 5.20, URF 5.21 URF 6.8**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the Account Generating page.	
2. The actor inserts the project name of the project to which the account to be generated should be connected to.	
3. The actor selects the button to generate the participant account.	
	4. The system retrieves the project name.
	5. The system generates a user account with the participant role, including username and password.
	6. The system saves the user account to the database.
	7. The system displays the user information to the actor.

Alternative:
2B: In case the actor navigated to the Account Generating page inside a project, this step can be skipped.
2C: In case the function is implemented, the actor can also specify the number of accounts to be generated here.
4B: In case the system cannot retrieve the project name, the system displays an error message.
5B: In case the system cannot generate a user account, the system displays an error message.
6B: In case the database cannot be reached, the system displays an error message.

A.2.3 | Researcher changes the feedback model.

Goal: **The actor changes the model, which is used to provide feedback on the documents.**
Actor: **Researcher**
Precondition: **The actor is logged into the application.**
Postcondition: **The changes to the feedback model are implemented.**
Requirements: **URF 3.13, URF 3.16, URF 3.17, URF 3.18, URF 3.21, URF 3.22, URF 3.23, URF 3.24, URF 6.8, URF 6.9**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the feedback model page.	
2. The actor follows the steps given on the feedback model page.	
3. The actor inserts their changes to the feedback model.	
	4. The system processes the given changes.
	5. The system saves the changes of the feedback model.

Alternative:
4B: In case the system cannot save the changes of the feedback model, the system displays an error message.

A.3 | Administrator interface

The subsection Administrator interface details the use cases for the user Administrator. The administrator interface also includes the use cases as described in the researcher interface (URF 6.8).

A.3.1 | Administrator grants a user a role.

Goal: **The actor grants a selected user a specific role.**
Actor: **Administrator**
Precondition: **The actor is logged into the application. The selected user has an account in the system. The selected account does not have the specific role yet.**
Postcondition: **The selected user has been granted the specific role.**
Requirements: **URF 6.0, URF 6.1, URF 6.2, URF 6.3, URF 6.4, URF 6.5, URF 6.6, URF 6.13, URF 6.15, URF 6.18**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the administrator page.	
2. The actor selects the user for which they want to grant a role.	
3. The actor selects the role that they want to grant.	
4. The actor confirms their choice.	
	5. The system processes the request and grants the specific role to the specific user.
	6. The system displays a confirmation message.

Alternative:
5B: In case the request cannot be processed, an error message is shown.

A.3.2 | Administrator removes a role from a user.

Goal: **The actor removes a specific role from a selected user.**
Actor: **Administrator**
Precondition: **The actor is logged into the application. The selected user has an account in the system. The selected account has a specific role.**
Postcondition: **The selected user does not have the specified role anymore.**
Requirements: **URF 6.0, URF 6.1, URF 6.2, URF 6.3, URF 6.4, URF 6.5, URF 6.6, URF 6.14, URF 6.16, URF 6.17, URF 6.19**
Priority: **Must Have**
Steps:

Actor actions	System actions
1. The actor navigates to the administrator page.	
2. The actor selects the user for which they want to remove a role.	
3. The actor selects the role that they want to remove.	
4. The actor confirms their choice.	
	5. The system processes the request and removes the specific role from the specific user.
	6. The system displays a confirmation message.

Alternative:
5B: In case the request cannot be processed, the system displays an error message.
5C: In case the last administrator tries to remove its own administrator role, the system displays an error message.

B | Signing Page

Customer

Rianne Conijn

Date

12-05-2022

Signature

Supervisor

Huub de Beer

Date

2022-05-12

Signature