university of groningen // PASTAQ

Requirements Document

Software Engineering Project Feb 2022
Students: Tudor Dragan, Teresa Ferreira, Cristian Iacob, Mohammed Nacer Lazrak, Björn Schönrock, Dominic Therattil & Kaitlin Vos
First Supervisor: Lars Andringa
Second Supervisor: Dr A. Capiluppi

Contents

1 Abstract

The PASTAQ-GUI already provides a decent means of using the PASTAQ code, however, certain elements and functionalities of the GUI could be further improved in order to allow for better and more accessible usage of PASTAQ. The goal of our project is then to improve the GUI in order to cover the aforementioned issues such that they are fixed. This document serves to provide insight into the scope, stakeholders, and requirements of our project along with some minor technical details such as the programming languages and technologies that we are concerned with. As far as the requirements are concerned we further explain them by exploring their respective use cases along with their diagram. The primary targets of this document are researchers, data scientists, chemists, biologists and our client; however, readers with all levels of experience that have an interest in the PASTAQ-GUI may find the document useful in assessing our overall project.

2 Introduction

The project presented in this document is a Graphical User Interface (GUI) which aims to ease the use of the PASTAQ pipeline for users that do not wish to interact with it from a terminal and need an easy to use application to process their data. To be able to understand the context in which our GUI exists we need to take a closer look at liquid chromatography coupled with tandem mass spectrometry (LC-MS/MS).

LC-MS/MS is an advanced analytical chemistry technique that aims to identify and quantify proteins and their peptides in a sample. The instruments used in this technique output large amounts of data that need to first be converted through msConvert into two formats: .mgf and .mzXML. The former is then run through another suite of software meant to identify the peptides in the sample and then converted into an .mzID file. This together with the .mzXML files are fed into PASTAQ.

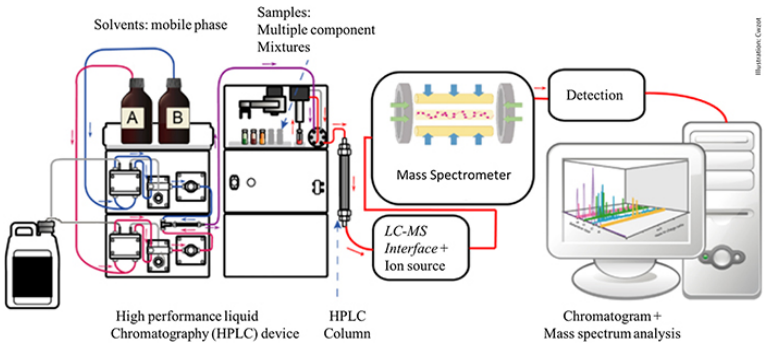

Figure 1: Liquid chromatography - mass spectrometry device

PASTAQ's intended use is in metabolomics and proteomics as it employs certain specialized algorithms. The pipeline outputs data as .csv files that identify proteins and features of the sample. These results are then meant to be interpreted through Python or R. It also outputs quality control plots as .png files. The application is available as a library that can be imported and used in C++ or Python. For users that are not familiar with programming, the GUI offers a simple alternative that also includes quality-of-life functionalities.

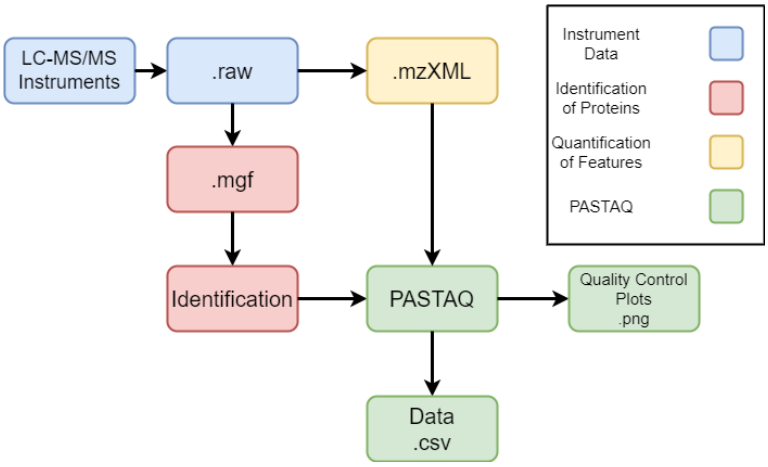

Figure 2: Overview of the ecosystem where PASTAQ is used

3 Scope

The scope of our project is to enhance and maintain the GUI that the PASTAQ project makes use of. The enhancements will be both functional and non-functional in nature. The GUI we are building upon is now in a functional albeit unpolished state. There are minor bugs present and it is missing certain features that would greatly benefit the user experience such as tool tips.

The central improvement (Figure 3: Automatic Identification) we will bring to the GUI is integrating the identification process of proteins. This is usually done through complex external applications and the researcher has different alternatives in which software to use (see Functional Requirement 3). In this way, the researcher will not have to worry about manually converting the files through the terminal, and will be able to select all wanted parameters inside the GUI making it easier to use and centralising the process.

Since our main goal is to improve the use of the GUI, we will first have to refactor the code such that currently known bugs and crashes are removed. It is also important to maintain the software usable on all major operating systems (Windows, Linux, Mac).

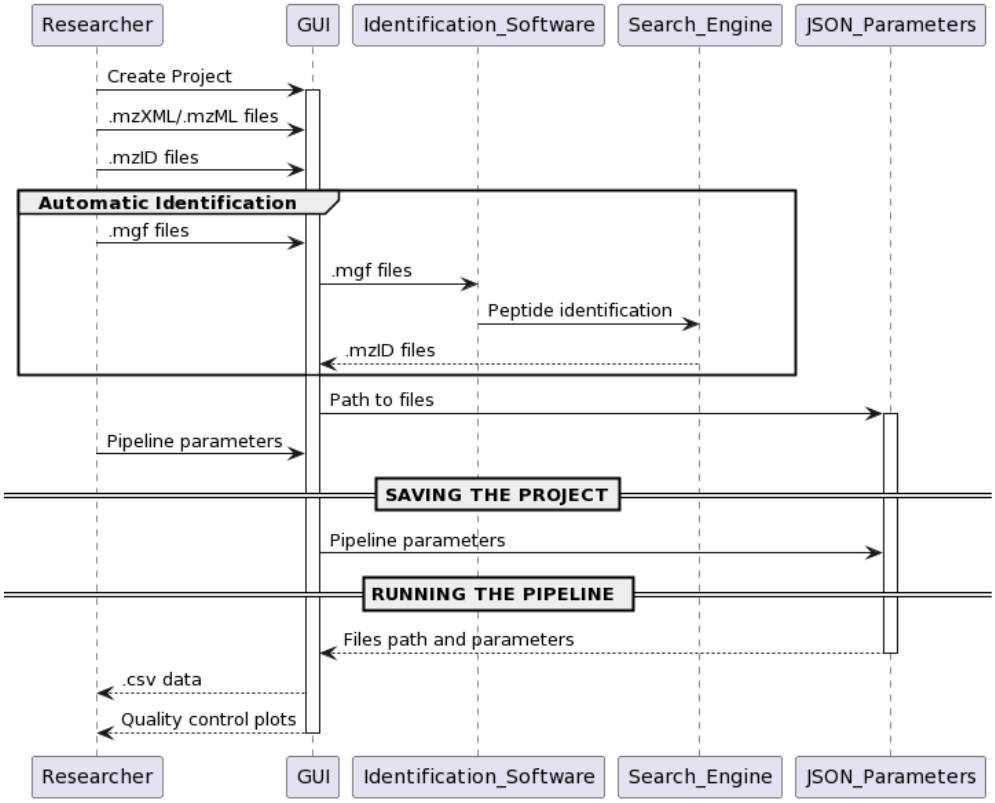

Figure 3: Overview of the GUI functionality

4 Stakeholders

4.1 Team - University of Groningen

- **Members**: Tudor Dragan, Teresa Ferreira, Cristian Iacob, Mohammed Nacer Lazrak, Björn Schönrock, Dominic Therattil, Kaitlin Vos
- **Supervisor**: Lars Andringa
- **Course Coordinator**: Dr A. Capiluppi

4.2 Client

- Alejandro Sánchez Brotons
- Prof. Dr. Peter L. Horvatovich

4.3 General Users

- Researchers, as PASTAQ automatically generates quality control plots, which allow the assessment of the overall dataset similarity between samples, as well as the distribution of peak widths, the retention time and the mass shifts.
- Data scientists, as PASTAQ provides full data traceability from the beginning to end of the analysis, which allows for post hoc analysis and data exploration.
- Chemists, as PASTAQ detects and quantifies compounds which are separated by liquid chromatography based on their chemical properties (e.g. hydrophobicity).
- Biologists, as PASTAQ can help identify the proteins of peptides through the use of tandem mass spectrometry.

5 Requirements list

We have decided to separate the requirements list based on whether said requirements are functional or non-functional. The functional requirements are further split according to the "MOSCOW" technique, such that the priority of each requirement is clearly specified. The last part of this section will describe the importance of each requirement. To further ensure the appropriateness of our requirements we considered the SMART technique (specific, measurable, attainable, realisable, testable).

The requirements are color coded according to the following schema to display which use case belongs to which requirements.

Table 1: Requirements user case schema

Related Use Case	Color
UC1	Red
UC2	Blue
UC3	Green
UC4	Violet
UC5	Orange
UC6	Teal
UC7	Olive
UC8	Purple

5.1 Functional Requirements

Identifier: R

1. **Must have**:
   - 1.1. [UC1] The GUI must accept .mzID files.
   - 1.2. [UC2] The GUI must accept .mgf files.
   - 1.3. [UC2] The GUI must allow the user to set the parameter values for the MSFragger process.
   - 1.4. [UC2] The GUI must allow the user to input the path to the MSFragger .jar file.
   - 1.5. [UC2] The GUI must allow the user to input the path to the idconvert.exe file.
   - 1.6. [UC2] The GUI must allow the user to input the path to the FASTA format protein database.
   - 1.7. [UC2] The GUI must create a .param file with the given MSFragger parameter values and the path to the protein database.
   - 1.8. [UC2] When given an .mgf file, the paths and the MSFragger parameters, the GUI must use MSFragger to process .mgf the file with the help of the newly created .param file into a .pepXML format.
   - 1.9. [UC2] After processing the file into a .pepXML, the GUI must use idconvert to convert the file into an .mzID file.
   - 1.10. [UC2] The intermediary .pepXML file must be discarded after creating the .mzID file.
   - 1.11. [UC2] When given an .mzXML/.mzML and an .mgf file, the GUI must execute the pipeline and output .csv data as well as .png quality control plots.

1.12. [UC3] The GUI must have tool-tips for the parameters to explain its functionalities to the user.

1.13. [UC7] Running the GUI must not display the cancel button after completion.

1.14. [UC7] Attempting to close the GUI must display a pop up dialog box to clarify, whether to save the project and close.

1.15. [UC7] Attempting to close the GUI must display a pop up dialog box to clarify, whether to discard any changes and close.

1.16. [UC7] Attempting to close the GUI must display a pop up dialog box to clarify, whether cancel the action and not close the GUI.

1.17. [UC7] The GUI must offer a drop down navigation menu where the user can choose to save the current project or open another project.

1.18. [UC7] The GUI must have a button to reset parameters to their default values.

1.19. [UC7] The GUI must have a button to select all the given files.

1.20. [UC7] The GUI must have a shortcut to select all the given files.

1.21. [UC7] The GUI must have a button to remove all the given files.

1.22. [UC7] The GUI must have a shortcut to remove all the given files.

2. **Should have**:

2.1. [UC1] The GUI should have a means of inputting files through a drag and drop feature.

2.2. [UC4] Canceling the PASTAQ pipeline while it is running should not cause the GUI to crash.

2.3. [UC7] The GUI should have a shortcut to save the project.

2.4. [UC7] The GUI should have a shortcut to open a new project.

2.5. [UC8] An executable file for Windows should automatically be generated using Github CI.

2.6. [UC8] An executable file for MacOS should automatically be generated using Github CI.

2.7. [UC8] An executable file for Linux should automatically be generated using Github CI.

2.8. [UC8] The Github CI pipeline should run a test suite for every new release to ensure that the user does not get incorrect code.

3. **Could have**:

3.1. [UC2] The uploaded .mgf files could be processed using SearchGUI and PeptideShaker, instead of MSFragger and idconvert.

3.2. [UC7] The GUI could have a splash page with the ability to select or make a new project.

5.2 Non-Functional Requirements

Identifier: NFR

4.1 [UC5] The code quality must be improved by refactoring code smells.

4.2 [UC5] The code must have comments that are descriptive and accurate.

4.3 The GUI can be understood and used by people outside of the research field.

4.4 [UC7] Buttons should respond in under 1 second when the GUI is used on an environment with 8GB RAM and a 2GHz processor.

4.5 [UC7] The GUI window must have its own custom icon.

4.6 [UC7] The GUI's elements could have their color changed to become more intuitive.

4.7 [UC7] The GUI could have a dark mode theme which offers a darker color palette.

4.8 [UC7] The GUI could display a progress bar with an estimated loading timer when running a process.

4.9 [UC7] The GUI should have collapsible parameters tab.

4.10 [UC7] The GUI window must be resizable.

4.11 [UC7] The GUI window must support a fullscreen option.

4.12 [UC6] Maintain support for Windows.

4.13 [UC6] Maintain support for MacOS.

4.14 [UC6] Maintain support for Linux.

5.3 Won't Have

5.1 The GUI will not have built-in data visualisation tools.

5.4 Importance Justification

In this section, we will provide justifications as to why we categorized our requirements the way we did.

- *MUST* - Having the possibility of inputting the unprocessed files and having them turned into the appropriate formats is crucial for simplifying the workflow of researchers. Tool-tips also play a very important role in speeding up the parametrization of the pipeline, thus cutting on work time. The features tied to the ease of use of the program are a must to implement as they cover important functionalities such as the saving, creating and editing of files and projects. Currently the GUI displays the OK button overlayed with the cancel button after a completed run, we must find a way to modify the GUI so as to only show the OK button.
- *SHOULD* - Currently, stopping the pipeline while it is running causes the application to crash. We should find a way to handle the multi-threading correctly and pass back control to GUI thread. The Drag and Drop feature to input files is something that should be added as it provides very accesible file input. The shortcuts for the creation and saving of projects should be implemented as they cover important functionalities with rather low usage frequency. Since the project is open source and its purpose is to aid researchers, maintaining the software usable for as many users as possible is fairly high on our importance list, therefore, we should try to make executables available for all major operating systems.
- *COULD* - Features like offering various paths for processing the files are not necessary but could be a welcome addition to the GUI, since each researcher has their own preferred type of software that they use. A splash screen for the creation or opening of a project could also be implemented, however, this does not add any extra functionality so it's of low importance.
- *NON-FUNCTIONAL* - The non-functional requirements are concerned with the accessibility, the look and feel of the interface. We would like these to be as friendly as

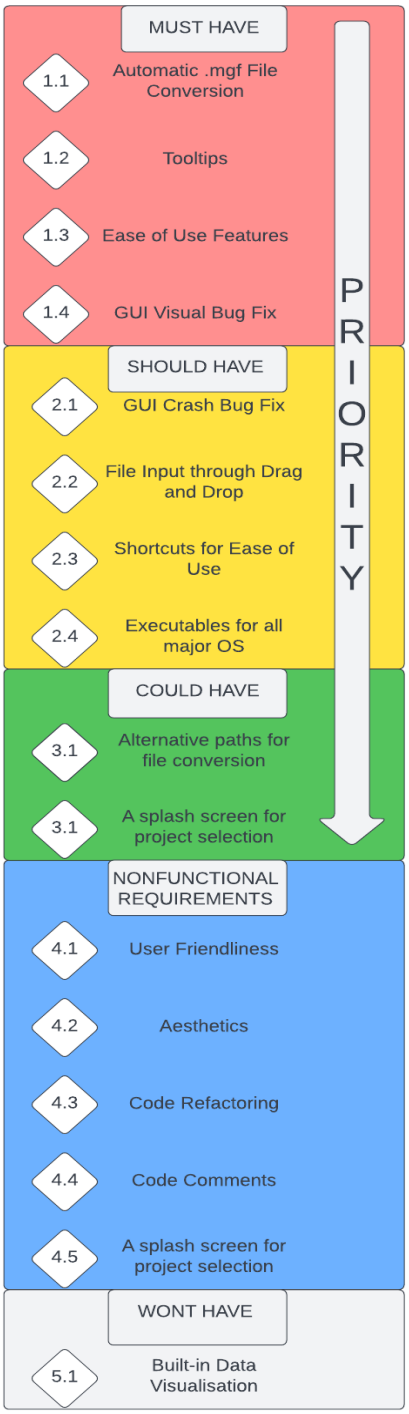

Figure 4: Requirement Priority

possible so that the tool is used by more researchers. We should also refactor the code base and add comments to make it manageable for future developers. Respectively since we're looking into making the project usable by as many users as possible, we'll have to maintain the software on all major operating systems.

- *WON'T* - Advanced data visualization tools are available out there and introducing this to the GUI will be an unnecessary effort.

6 Use Cases

All requirements for this single system can be related to the same GUI. This led us to choose use cases over user stories because we think they give a better structure to our requirements. This section is divided into eight different use cases, which we will consider individually, state the use case specification and present the use case diagram.

6.1 File Acceptance | UC1

Requirement:

- R1.1
- R2.1

Primary Actors:

- General Users

User Story:
As a general user, I would like to upload an .mzID file to the GUI so that I can run the PASTAQ pipeline.

Context:
The general user wishes to be able to upload an .mzID file.

Main Success Scenario:

- The user is able to upload the .mzID file

Alternative Scenarios:

- The .mzID file fails to get uploaded
- An error message is displayed
- Prompted to try again

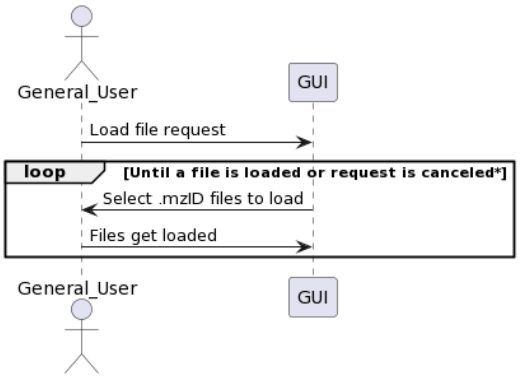

Figure 5: File Acceptance | UC2

Preconditions:

- .mzID file is uploaded

Postconditions:

- .mzID file is accepted and loaded into the GUI

Frequency of Occurrence:
High usage frequency, since whenever the users accesses the GUI they could be uploading an .mzID file.

6.2 File Processing | UC2

Requirement:

- R1.2 - R1.11
- R3.1

Primary Actors:

- General Users

User Story:
As a general user, I would like to upload an .mgf to the GUI and have it converted automatically, so that I do not need to manually process it with MSFragger and idconvert into an .mzID.

Context:
The general user wishes to avoid the tedious process of manually converting the .mgf files using the command line or other applications, instead the general user can upload .mgf files to the GUI, which will then be processed automatically to .mzID files with MSFragger and idconvert.

Main Success Scenario:

- The user is able to upload the .mgf file
- The .mgf file gets processed to an .mzID file with MSFragger and idconvert
- The newly converted .mzID file gets loaded into the GUI

Alternative Scenarios:

- The .mgf file fails to get processed to .mzID
- An error message is displayed
- Prompted to try again or manually convert the file to an .mzID file

Preconditions:

- .mgf file is uploaded

Postconditions:

- File is processed to an .mzID and loaded into the GUI

Frequency of Occurrence:
High usage frequency, since whenever the users accesses the GUI they could be uploading an .mgf file.

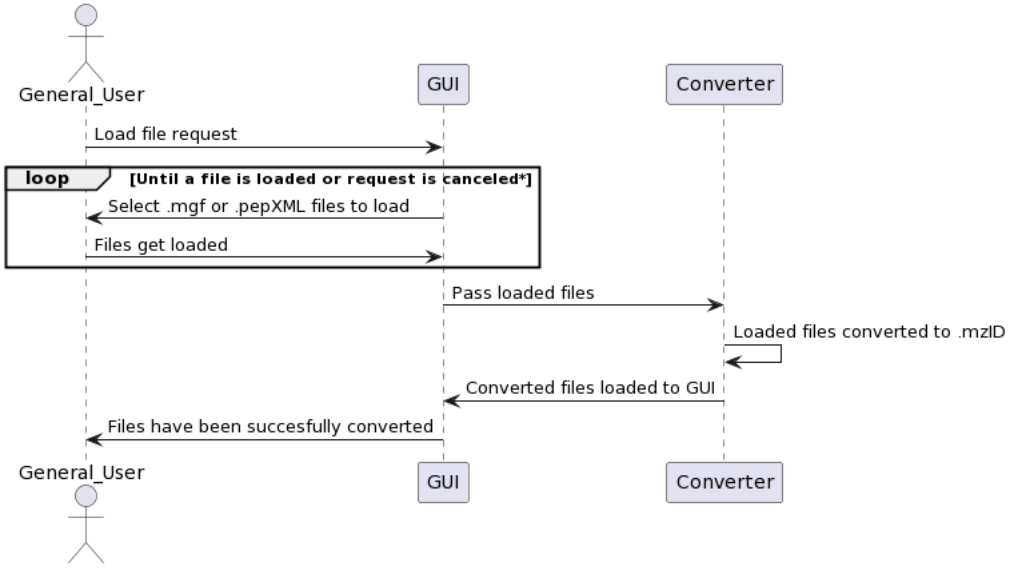

Figure 6: File Conversion | UC2

6.3 Tool-tips | UC3

Requirement:

- R1.12

Primary Actors:

- Inexperienced Users

User Story:
As an inexperienced user, I would like to be able to get tips and hints when setting parameters for the PASTAQ-GUI, so that I can understand how they will modify the analysis.

Context:
The user wants to access the PASTAQ-GUI, however, the user does not know how to fully work with the parameter options.

Main Success Scenario:

- The user is able to follow the tooltips and correctly modify the analysis of his data.

Alternative Scenarios:

- The tooltips are not simple enough, the user gets confused
- The tooltips are too simple, they do not contain enough information to achieve the users task

Preconditions:

- Tooltips written and explained to informed users
- User has enough knowledge to work with the GUI and understand the tooltips

Postconditions:

- The tooltips help the user understand the parameters
- User successfully processes his data

Frequency of Occurrence:
Moderate usage frequency, whenever a beginner user uses the parametrization tab of the GUI and as the user gets more experienced they will use this feature less.

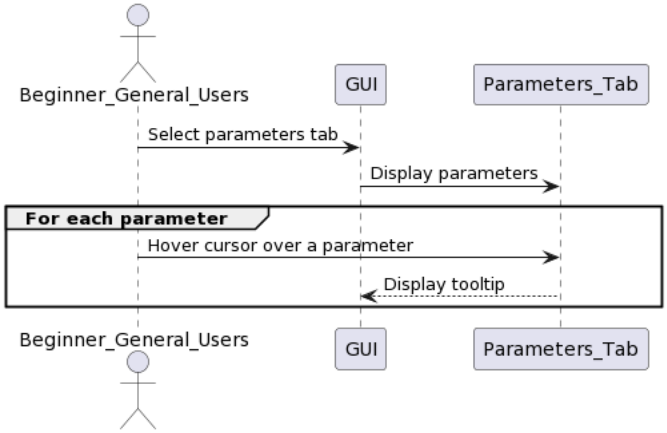

Figure 7: Display Tool Tips | UC3

6.4 Process interrupting errors | UC4

Requirement:

- R2.2

Primary Actors:

- General Users

User Story:
As a general user, I want all processes to terminate without errors by pressing the cancel button when the pipeline is running, so that I can close the PASTAQ-GUI without any errors.

Context:
In the currently existing version of the GUI, if the pipeline is started, it will have an option to cancel the processing of the data. This causes an error that makes the GUI crash and exit.

Main Success Scenario:

- The pipeline is ended and the GUI still runs.

Alternative Scenarios:

- The program crashes upon shutting down the pipeline
- The general user is provided with the appropriate crash information

Preconditions:

- GUI is running
- The user starts the pipeline
- The user wishes to stop the pipeline and presses the cancel button.

Postconditions:

- The pipeline is stopped
- The GUI is still running

Frequency of Occurrence:
Moderate usage frequency, since this will take place only when the user starts the pipeline and decides it does not want to let it finish.

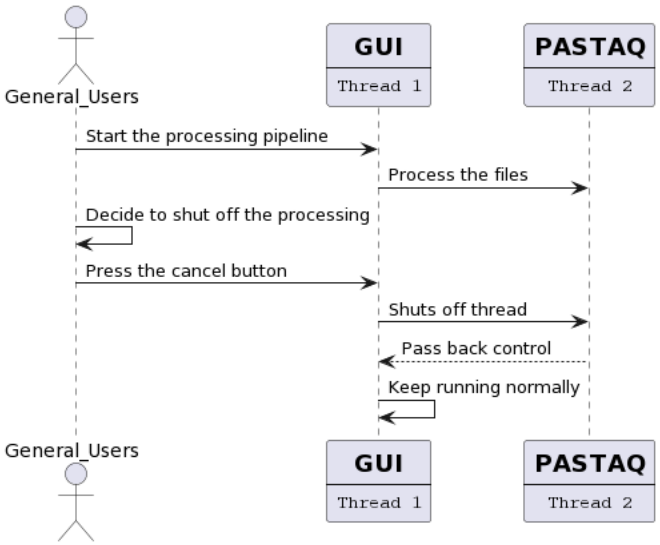

Figure 8: Thread Control Crash Avoidance | UC4

6.5 Code Smells | UC5

Requirement:

- NFR4.1-NFR4.2

Primary Actors:

- PASTAQ-GUIs Developers

User Story:
As a developer of PASTAQ-GUI, I want the code to be error-free and properly structured and documented, so that it is easily understandable and extendable.

Context:
Since the PASTAQ-GUI is an open-source project, there can be users with an understanding of programming who wish to fork the project and adjust the code for their own specific needs.

Main Success Scenario:

- The user is able to easily understand the structure of the program and implement their features into the PASTAQ-GUI

Alternative Scenarios:

- The user is unable to understand the code and can raise an issue on the repository
- The user encounters a bug and can open a pull request to fix it

Preconditions:

- The user is able to understand the code
- The user is experienced with programming

Postconditions:

- The user is able to successfully extend the GUI with their new functionality

Frequency of Occurrence:
This would occur every time someone decided to fork the project. Moderate to low frequency as there are not that many people that have experience with programming and do research in chemistry and biology.

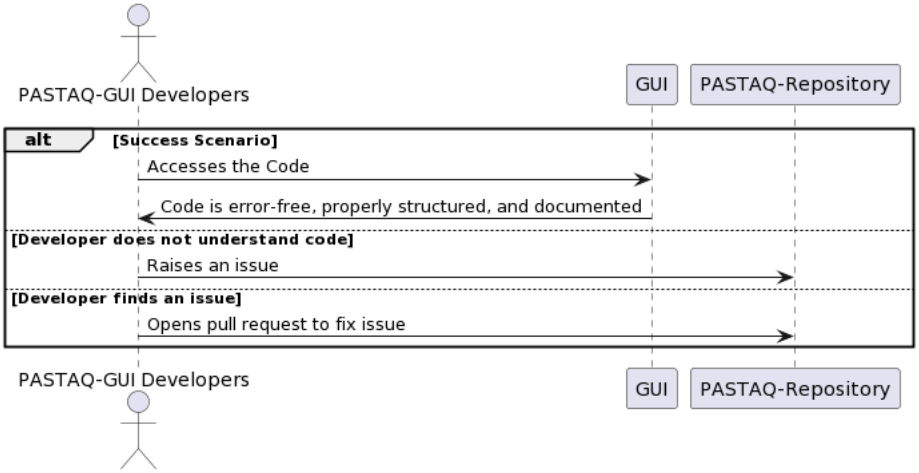

Figure 9: Code Smells | UC5

6.6 Operating System | UC6

Requirement:

- NFR4.3

Primary Actors:

- PASTAQ-GUI General Users
- Developers
- The Product Owner

User Story:

- **PASTAQ-GUI General Users**
  As a general user, I would like to be able to run the PASTAQ-GUI on any operating system, so that it ensures versatility and consistency.
- **Developers**
  As a developer, I would like to be able to deploy the application on any operating system, such that I can work on the project.
- **The Product Owner**
  As the product owner, I would like the software to be available on as many platforms as possible, so that it can be successful.

Context:
The user wants to use the PASTAQ-GUI in any Windows, Linux or MacOS- based operating system.

Main Success Scenario:

- The user is able to run the GUI on Windows, Linux and MacOS

Alternative Scenarios:

- The user is informed with an error message as to why the program is not running

Preconditions:

- Program runs on the users operating system

Postconditions:

- Program Terminates successfully

Frequency of Occurrence:
Every time the PASTAQ-GUI is run.

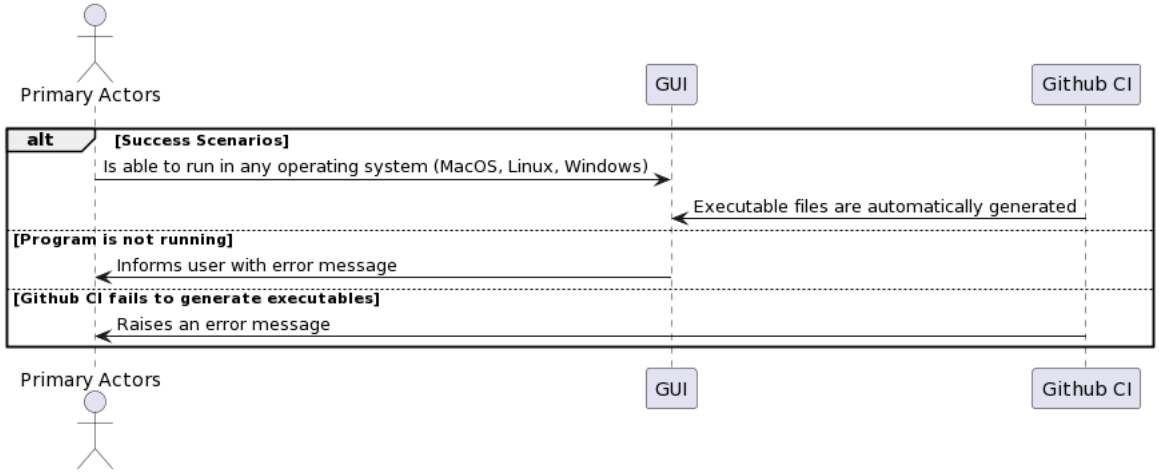

Figure 10: Operating System | UC6

6.7 Ease of Use | UC7

Requirement:

- R1.13-R1.22
- R2.3-R2.4
- R3.2
- NFR4.4-NFR4.11

Primary Actors:

- General Users
- Inexperienced users

User Story:
As a general user, I want the GUI to be aesthetically pleasing, so that I will be more inclined to choose it over other software.
As an inexperienced user, I want to see an intuitive and responsive so that I can get my work done without having to spend time on understanding the system.

Context:
The user wants the current implementation of the GUI to be responsive, more pleasing to look at and easier to use. Aside from necessary GUI changes, we have also identified a minor bug which is that the cancel button doesn't get removed from the GUI after the completion of a run.

Main Success Scenario:

- The user found the GUI to be pleasing to look at.
- The user found the GUI responsive and intuitive to use.

Alternative Scenarios:

- The user found the GUI displeasing to look at.
- The user found the GUI to be unintuitive.
- The user found the GUI to be unresponsive.

Preconditions:

- The GUI is intuitive to use.
- User has enough knowledge to work with the GUI.

Postconditions:

- The GUI helps the user process his data more easily.
- User successfully processes his data.

Frequency of Occurence: Very high, every time the PASTAQ-GUI is accesed by the user.

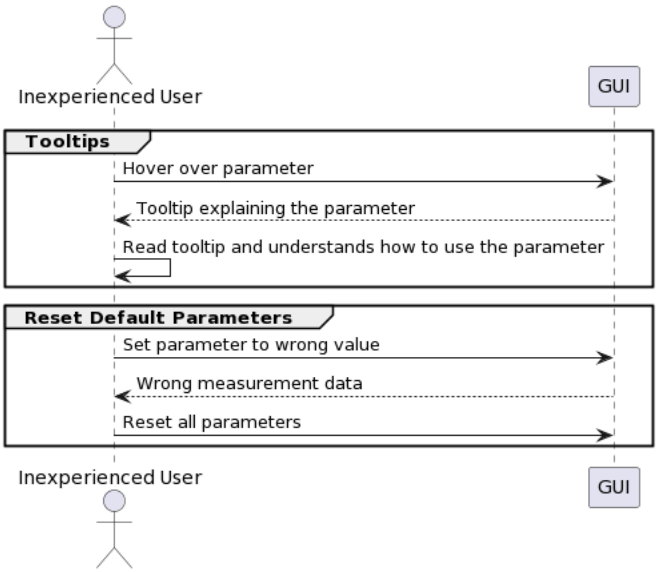

Figure 11: Different Ease of Use Features | UC7

6.8 Continuous Integration & Executables | UC8

Requirement:

- R2.5 - 2.8

Primary Actors:

- PASTAQ-GUI General Users
- The Product Owner

User Story:

- **PASTAQ-GUI General Users**
  As a general user, I would like to be able to download a working executable file, so that I do not have to perform complex installation steps.
  As a general user, I would like to get a working executable, so that I do not have to deal with bugs.
- **The Product Owner**
  As the product owner, I would like to have automatically generated executables, so that I do not have to do that manually for every update.

Context:
The user wants to run the PASTAQ-GUI in any Windows, Linux or MacOS- based operating system without having to perform complex installation steps.

Main Success Scenario:

- Executable files for Windows, Linux and MacOS are automatically generated through Github CI

Alternative Scenarios:

- Github CI fails to generate executables and raises an error message.
- An executable generated by Github CI fails to run and raises an error message.
- The new release fails the tests and some parts have to be fixed.

Preconditions:

- An updated version of the code is pushed to Github.

Postconditions:

- Executable files are generated by Github CI.

Frequency of Occurrence:
Every time a new release of PASTAQ-GUI is made.

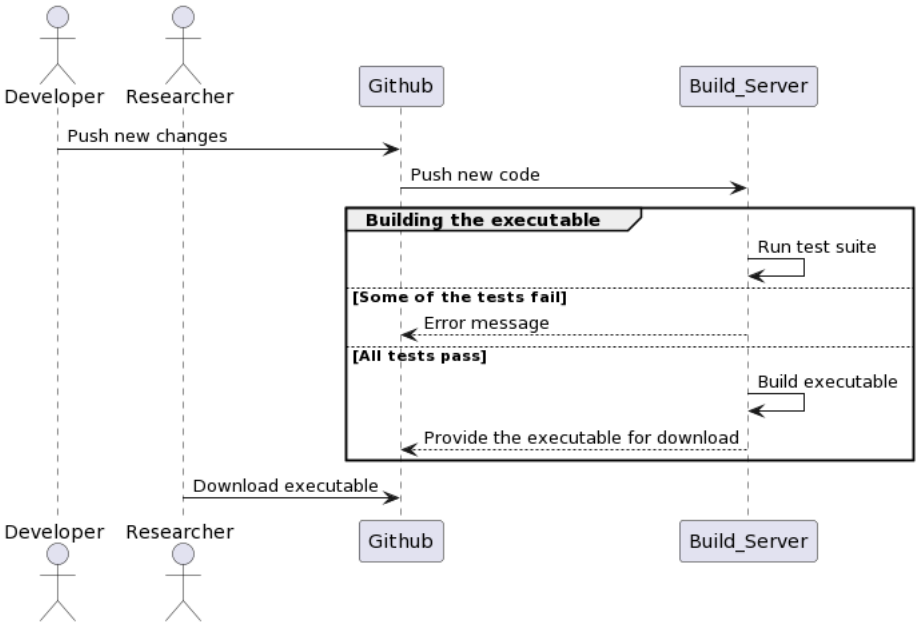

Figure 12: Continuous Integration | UC8

7 Languages and Technologies

Languages:

- Python, the language in which our PASTAQ-GUI is written in

Technologies:

- MSFragger, generates .pepXML files from .mzXML/.mzML or .mgf files
- idconvert, converts .pepXML files to .mzID so they can be parsed
- PASTAQ, takes the .mzID and .mzXML files and outputs .csv files and .png files for quality control

8 Terminology

Glossary

.mgf Mascot generic format files are the standard format to store protein data. 3, 6, 7, 12, 26

.mzID A parser for the mzIdentML file format defined by HUPO. 3, 6, 11, 12, 26

.mzXML An XML (eXtensible Markup Language) based common file format for proteomics mass spectrometric data. 3, 6, 26

.pepXML An open data format developed at the SPC/Institute for Systems biology for the storage, exchange, and processing of peptide sequence assignments of MS/MS scans. 6, 26

FASTA A text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. 6

idconvert A command line tool for converting between various file formats. 6, 26

liquid chromatography A technique in analytical chemistry used to separate, identify, and quantify each component in a mixture. 3, 5

metabolomics The scientific study of chemical processes involving metabolites, the small molecule substrates, intermediates and products of cell metabolism. 3

msConvert A command-line utility for converting between various mass spectrometry data formats, including from raw data from several commercial companies (with vendor libraries, Windows-only). 3

MSFragger Database search tool for peptide identification in mass spectrometry. 6, 26

PASTAQ Pipelines and Systems for Threshold-Avoiding Quantification; PASTAQ provides a set of tools for high-performance pre-processing of LC-MS/MS data. 2–5, 14, 16, 18, 20, 24, 26

peptide Short chains of amino acids linked by peptide bonds. 3, 5

PeptideShaker A search engine independent platform for protein identification. 7

post hoc analysis A type of statistical analysis that is conducted following the rejection of an omnibus null hypothesis. 5

proteomics The large-scale study of proteins. 3

SearchGUI Open-source common interface for configuring and running protein data search. 7

tandem mass spectrometry A technique in instrumental analysis where two or more mass analyzers are coupled together using an additional reaction step to increase their abilities to analyse chemical samples. 3, 5

9 Client Meeting Log

Table 2: Client Meeting Log

Meeting Date	Content
2022-02-25	• General introduction • Discussed available projects • Short overview of client system
2022-03-02	• Discussed project choice • Client provided overview of the whole system • Discussion about specific requirements (fix bugs, small improvements etc.) • Brainstorm possible requirements (integration of visuals and statistics)
2022-03-09	• Reported error that occurs when trying to install PASTAQ • Cleared up uncertainty about all the files and programs involved • Discussed scope of the project • Agreed on the requirements of the project • Corrected the languages and technologies • Assisted us on setting up the necessary tools for the project

Continued on next page

Table 2: Client Meeting Log (Continued)

Meeting Date	Content
2022-03-18	• Reported issues we encountered when trying to install PASTAQ • Went through the installation step-by-step • Encountered errors throughout and solved them together with the client • successfully installed PASTAQ • Asked leftover questions

10 Change log

Table 3: Change Log

Date	Time	Team Member	Changes	Version
12.2.2022	13:58	Kaitlin Vos	Inserted change log table	1.0
15.2.2022	12:16	Dominic Therattil	Cover page draft	1.1
16.2.2022	16:45	Tudor Dragan	Client Meeting Log	1.1.1
19.2.2022	13:39	Dominic Therattil	Updated Cover Page	1.1.2
01.03.2022	13:39	Dominic Therattil	Updated banner for pastaq	1.1.2
02.03.2022	12:55	Björn Schönrock	Updated meeting log	1.1.3
04.03.2022	12:30	Kaitlin Vos	Meeting log changed to table format and second meeting notes entered	1.1.4
04.03.2022	13:00	Björn Schönrock	Writing requirements	1.1.5
04.03.2022	13:00	Mohammed Nacer Lazrak	Writing Requirement first draft	1.1.5
05.03.2022	08:00	Cristian Iacob	Added stakeholders, system's scope and frameworks	1.1.6
05.03.2022	13:00	Kaitlin Vos	More detailed requirements	1.1.6
05.03.2022	22:00	Tudor Dragan	Introduction	1.1.8
06.03.2022	19:00	Dominic Therattil	Added use cases	1.1.9
06.03.2022	19:00	Teresa Ferreira	Added use cases	1.1.10
06.03.2022	19:00	Dominic Therattil	Added terminology	1.1.11
06.03.2022	23:41	Kaitlin Vos	Fixed grammar and fine-tuned introduction	1.1.12
07.03.2022	12:00	Cristian Iacob	Moved system scope to scope and added to it	1.1.13
07.03.2022	12:21	Kaitlin Vos	Terminology glossary	1.1.14
08.03.2022	11:39	Kaitlin Vos	Terminology extended and general clean-up	1.1.15
08.03.2022	12:04	Kaitlin Vos	Fixed images and added use case diagrams folder	1.1.16
09.03.2022	13:36	Nacer Lazrak	Fixed requirements & syntax review	1.1.17
09.03.2022	14:00	Tudor Dragan	Improved the scope	1.1.18
09.03.2022	14:30	Cristian Iacob	Cleaned up the scope and frameworks based on feedback	1.1.19
09.03.2022	22:13	Kaitlin Vos	Client meeting log updated	1.1.20

Continued on next page

Table 3: Change Log (Continued)

Date	Time	Team Member	Changes	Version
11.03.2022	11:52	Kaitlin Vos	Terminology updated	1.1.21
11.03.2022	11:52	Tudor Dragan	Improved requirements & added justification section	1.2
12.03.2022	22:00	Cristian Iacob	Moved team & client to stakeholders, added justification for users	1.2.1
13.03.2022	15:55	Kaitlin Vos	Fixed all inconsistencies, grammar issues and updated terminology	1.2.2
14.03.2022	10:00	Björn Schönrock	Small fixes and cleanup, clarified requirements	1.2.3
14.03.2022	12:11	Kaitlin Vos	Upgraded tables	1.2.4
15.03.2022	12:11	Dominic Therattil	Small fixes and Corrections	1.2.5
16.03.2022	20:00	Teresa Ferreira	Updated Use Cases	1.2.6
16.03.2022	22:00	Tudor Dragan	Added Diagrams for Scope and Requirements	1.2.7
17.03.2022	15:03	Kaitlin Vos	Image placement	1.2.8
17.03.2022	16:00	Dominic Therattil	Updated Use Cases	1.2.9
20.03.2022	20:00	Tudor Dragan	Improved UC 3,4,5,6	1.2.10
21.03.2022	05:44	Cristian Iacob	Color coded the requirements based on respective use case	1.2.11
21.03.2022	11:41	Kaitlin Vos	Color coding completed	1.2.12
22.03.2022	12:00	Dominic Therattil	Improved use cases and fixed alignment/formatting issues	1.2.13
22.03.2022	12:00	Dominic Therattil	Added diagrams for UC1 & UC2	1.2.14
22.03.2022	17:00	Björn Schönrock	Added CI requirement	1.2.15
23.03.2022	11:00	Dominic Therattil	Updated UML diagram for UC3 & 4 and cover page	1.2.16
23.03.2022	19:33	Teresa Ferreira	Resized UML Diagrams for UC1, UC2, UC3 & UC4	1.2.17
23.03.2022	19:34	Teresa Ferreira	Added UML Diagram for UC5	1.2.18
23.03.2022	19:47	Teresa Ferreira	Divided User Story for UC6	1.2.19
23.03.2022	16:07	Teresa Ferreira	Added UML Diagram for UC6	1.2.20
25.03.2022	16:00	Kaitlin Vos	Changed justification layout and updated client meeting log	1.2.21
25.03.2022	16:17	Kaitlin Vos	Rewrote requirements 2.1 - 2.7 and edited UC6	1.2.22

Continued on next page

Table 3: Change Log (Continued)

Date	Time	Team Member	Changes	Version
29.03.2022	20:30	Cristian Iacob	Added new requirements and broke larger requirements into smaller more specific ones	1.3
30.03.2022	14:10	Cristian Iacob	Added use case 7 and updated requirements accordingly	1.3.1
30.03.2022	14:10	Björn Schönrock	Added use case 8 and updated requirements	1.3.2
30.03.2022	20:10	Cristian Iacob	Updated importance justification of requirements and moved some requirements to non functional	1.3.3
30.03.2022	21:10	Cristian Iacob	Added Abstract section	1.3.4
30.03.2022	22:06	Cristian Iacob	Updated Requirements Priority Diagram	1.3.5
30.03.2022	22:30	Tudor Dragan	Added diagram for UC7 and UC8	1.3.6
01.04.2022	10:23	Cristian Iacob	Fixed formatting issues	1.3.7
01.04.2022	13:46	Cristian Iacob	Changed UC4 & UC7, updated requirements accordingly	1.3.8
02.04.2022	12:00	Tudor Dragan	Improved UC8 & refactored small mistakes	1.3.9
03.04.2022	23:52	Kaitlin Vos	Added requirements for consistency	1.4
04.04.2022	11:01	Kaitlin Vos	Proof-read entire document, fixed typos, syntax, logic	1.4.1