Customizable interface for LTL conformance checking

Requirements engineering

Group 2

Teodora Staneva
Mohamed Amine Kooli
Ugo Detaille
Vishisht Choudhary
Daniel Henel

Aachen, November 2022

Contents

1. Requirements elicitation and development

During this chapter we will identify the stakeholders and their needs by doing detailed research. Furthermore, we will categorize all requirements according to the target groups and elaborate how we plan to inspect their quality. Finally, we will create a hierarchical structure of the said requirements to create complete clarity about the tasks the different specialists of our team have to complete. A good implementation of our structure also enables us to provide a high standard product for our client.

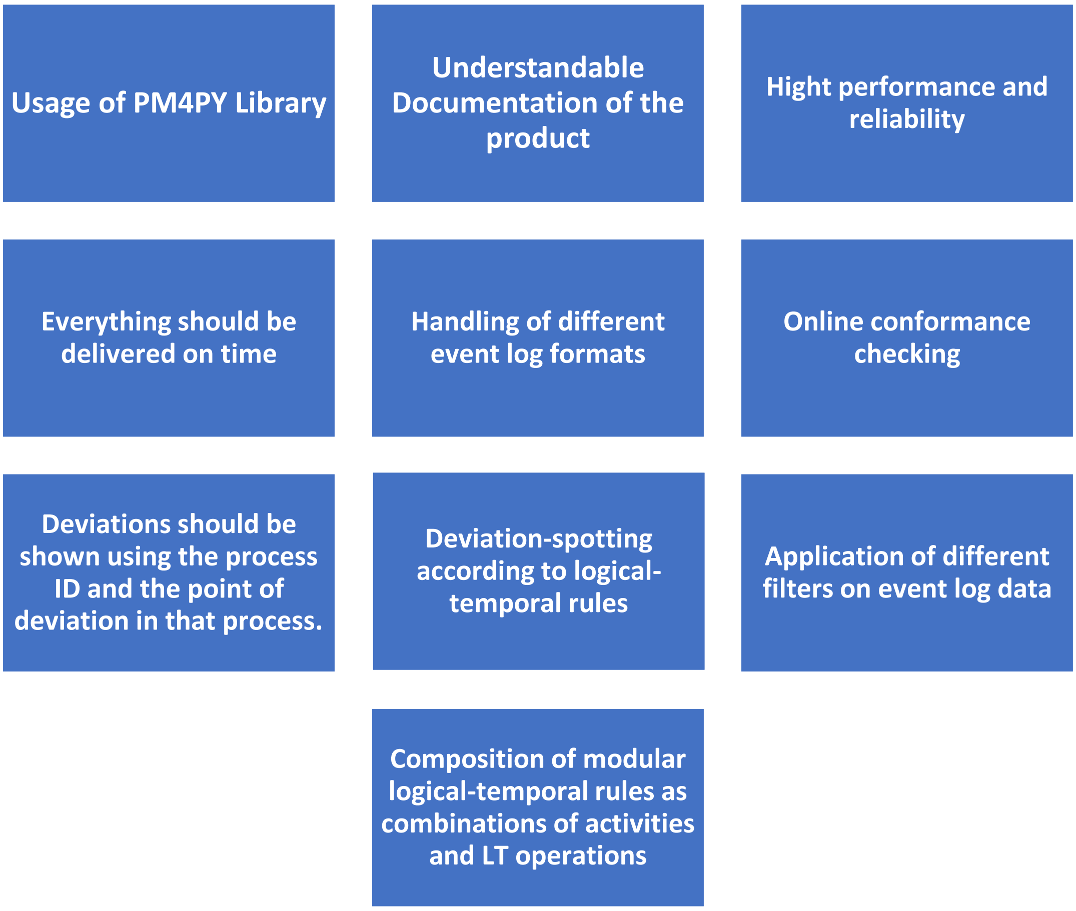
1.1. Stakeholders

After conducting thorough analysis of our project and our deliverables we have identified the following stakeholders:

- the project team
- our client (Alessandro Berti)
- the user

1.2. Raw requirements

1.2.1. Customer Requirements



Usage of PM4PY Library

Understandable Documentation of the product

Hight performance and reliability

Everything should be delivered on time

Handling of different event log formats

Online conformance checking

Deviations should be shown using the process ID and the point of deviation in that process.

Deviation-spotting according to logical-temporal rules

Application of different filters on event log data

Composition of modular logical-temporal rules as combinations of activities and LT operations

1.2.2. User Requirements

- Customizable interface for LTL Checking
- Intuitive Interface and user experience
- Software responsiveness
- Accurate results
- Understandable user manual
- User-friendly design

1.2.3. Developer Requirements

- Consistent code format according to PM4PY standards
- Work according to the Scrum methodology
- Monolithic architecture of the software
- Object oriented programming
- Usage of availiable libraries (Pandas, Numpy etc.)
- Code editing should happen with GitLab

1.2.4. System Requirements

- Support for different operating systems like Windows, MacOS, Linux.
- Python 3.10, PM4PY, Pandas
- Flask 2.2.2 and its dependencies: Jinja, MarkupSafe, Werkzeug, ItsDangerous, Click

1.3. Requirement Analysis

In order to perform a sufficient requirements analysis we had to stay in exchange with our client (Alessandro Berti). After examining the current situation and the needs of the client we were able to pinpoint the requirements of this project and structure them thematically. Furthermore we introduced a hierarchical structure enabling us to dive deeper into the details of our requirements.

1.4. Allocation and Flow-Down of Requirements

This section will present the findings of the requirement analysis in a hierarchical structure. To meet the needs of the program, hierarchically categorized requirements will specify how applications and subsystems interact.

1.4.1. High Level

HL1: The user should be able to upload event logs into the application

HL2: The user should be able to select different filtering types

HL3: The application should provide an intuitive and understandable interface

HL4: The user should have the option to save the filtered event logs to the device

HL5: The user should be able to select and combine features that need to be filtered (OR, AND)

HL6: The application should be able to show exact points of deviation with the possibility to explore the variants and the events (online conformance checking)

HL7: The user should be able to select certain features from the input data.

HL8: The code of the program should be understandable and readable

HL9: The documentation of the code should be thorough and precise

HL10: In case of an error the software should return an explanation of the origin of said error.

HL11: The software should support various recent operating systems

HL12: The combinations of filters should give correct results according to LTL and logical operations.

HL13: The software should be optimized to run efficiently.

1.4.2. Intermediate Level

IL1: The user should be able to select required columns for the filtering process. (HL7)

IL2: The code should be modular and well commented (HL8, HL13, HL9)

IL3: The software should use different packages (HL13)

IL4: The user should be able to select the format of the filtered event log. (HL4)

IL5: The Program should run on Linux, Windows and MacOS (HL11)

IL6: The functionality and performance of the code should be verified with extensive testing. (HL13)

IL7: The Software should support Python 3.10.0 (HL11)

IL8: The library should provide methods for importing a process model and event log (HL1, HL7)

IL9: The data in the originally uploaded event log should not be modified or changed (HL1, HL7)

IL10: The implementation of the error handling in our application should be understandable to the user (HL10, HL3)

IL11: The user should have the option to store the result as a PDF/CSV/XES/Parquet file. (HL4)

IL12: The application view should be responsive (HL13, HL3)

IL13: The user should be able to load event logs in CSV, XES or Parquet format into the application (HL1)

IL14: The user should be able to select LTL filters (Four eyes principle, Attribute value different persons, eventually follows) (HL2)

IL15: The library should re-use existing software where possible (HL13)

IL16: The interface should be flexible and functional (HL3)

1.4.3. Low Level

LL1: Previously imported event log can be selected without importing it again (IL8)

LL2: The software should depend on PM4Py (IL15)

LL3: The software should provide the possibility of grouping and sorting the results (IL16)

LL4: The software should show the results of different files in different tabs that can be used independently (IL16)

LL5: The user can choose language of the application (English, German + maybe our languages) (IL16)

LL6: The user should be able to filter the event log data by all features (activity, timeframe, ID etc.) (IL1)

LL7: The user should be able to filter out events containing or not containing certain activities

LL8: The backend part of the application should use the PM4PY, Pandas and Numpy library

LL9: The graphical interface is to be implemented in flask (IL12)

LL10: The user should be able to filter for outliers in the event log data(optional)

LL11: The test suite should simulate different python versions

LL12: There should be a manual on how to operate the software

LL13: The application should provide a list of all unique activities which can be used for filtering (IL14)

LL14: The test suite should ensure that the implementation works correctly (IL6, IL10, IL12)

LL15: The test suite should check for responsiveness and performance (IL6, IL12)

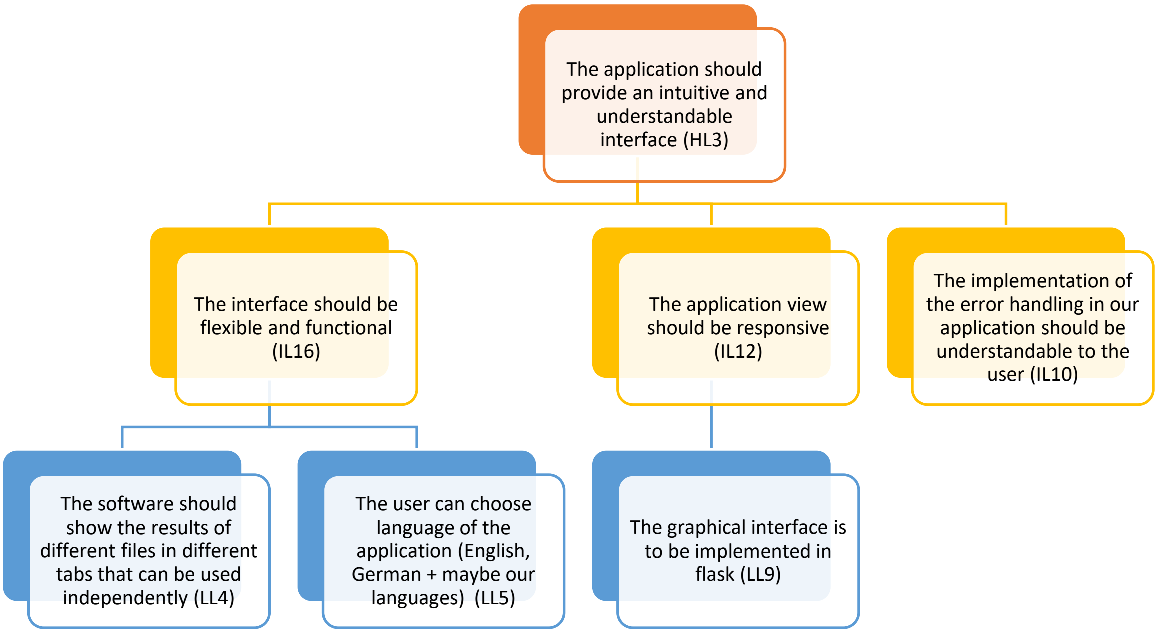
LL16: The test suite should identify bugs and based on the information they will be fixed before the final product (IL6)

LL17: The test suite will ensure the correctness of the generated model (IL6)

LL18: The user can rename selected columns.

1.4.4. Flow-down diagrams

The following diagrams graphically represent key requirements in a hierarchical form:



The application should provide an intuitive and understandable interface (HL3)

The interface should be flexible and functional (IL16)

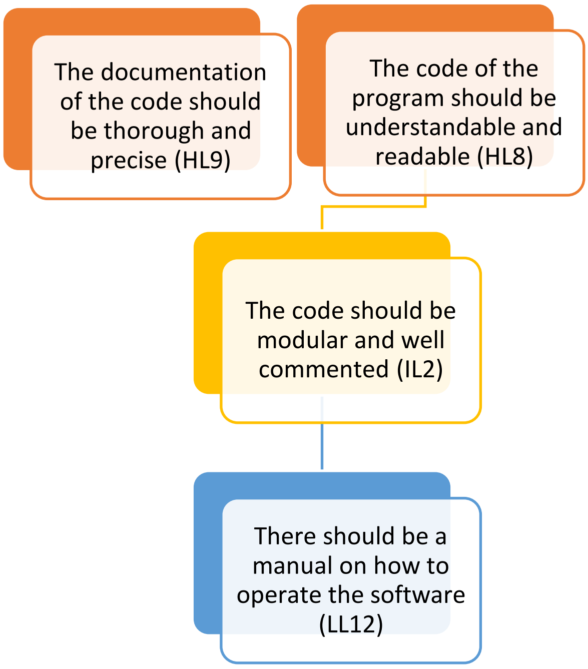
The application view should be responsive (IL12)

The implementation of the error handling in our application should be understandable to the user (IL10)

The software should show the results of different files in different tabs that can be used independently (LL4)

The user can choose language of the application (English, German + maybe our languages) (LL5)

The graphical interface is to be implemented in flask (LL9)

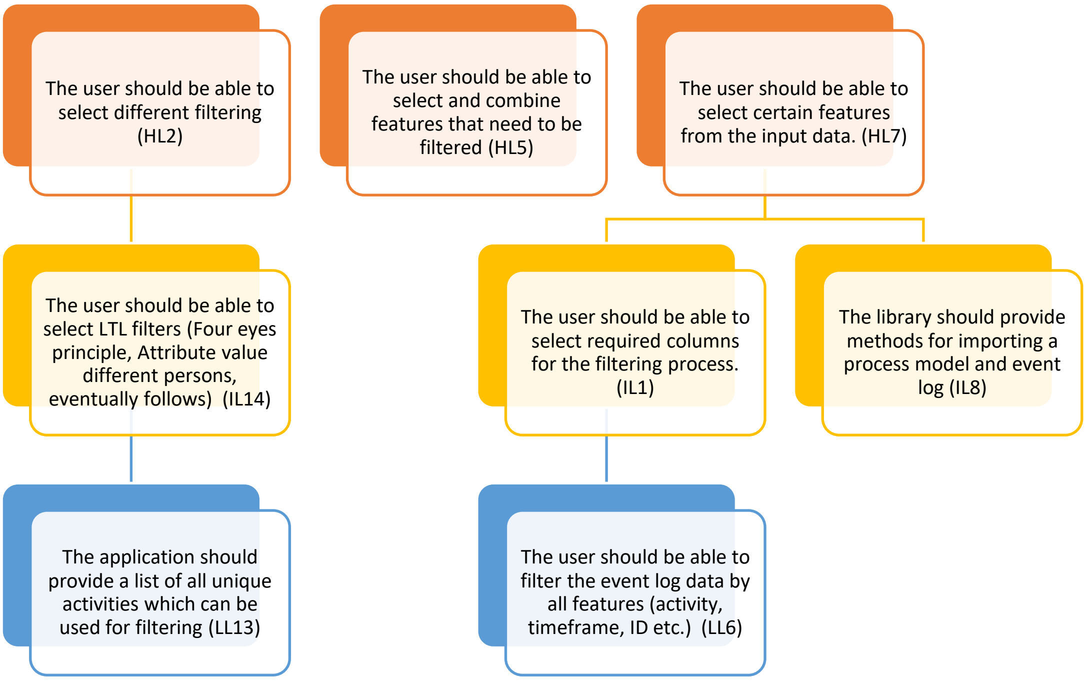


The documentation of the code should be thorough and precise (HL9)

The code of the program should be understandable and readable (HL8)

The code should be modular and well commented (IL2)

There should be a manual on how to operate the software (LL12)



The user should be able to select different filtering (HL2)

The user should be able to select and combine features that need to be filtered (HL5)

The user should be able to select certain features from the input data. (HL7)

The user should be able to select LTL filters (Four eyes principle, Attribute value different persons, eventually follows) (IL14)

The user should be able to select required columns for the filtering process. (IL1)

The library should provide methods for importing a process model and event log (IL8)

The application should provide a list of all unique activities which can be used for filtering (LL13)

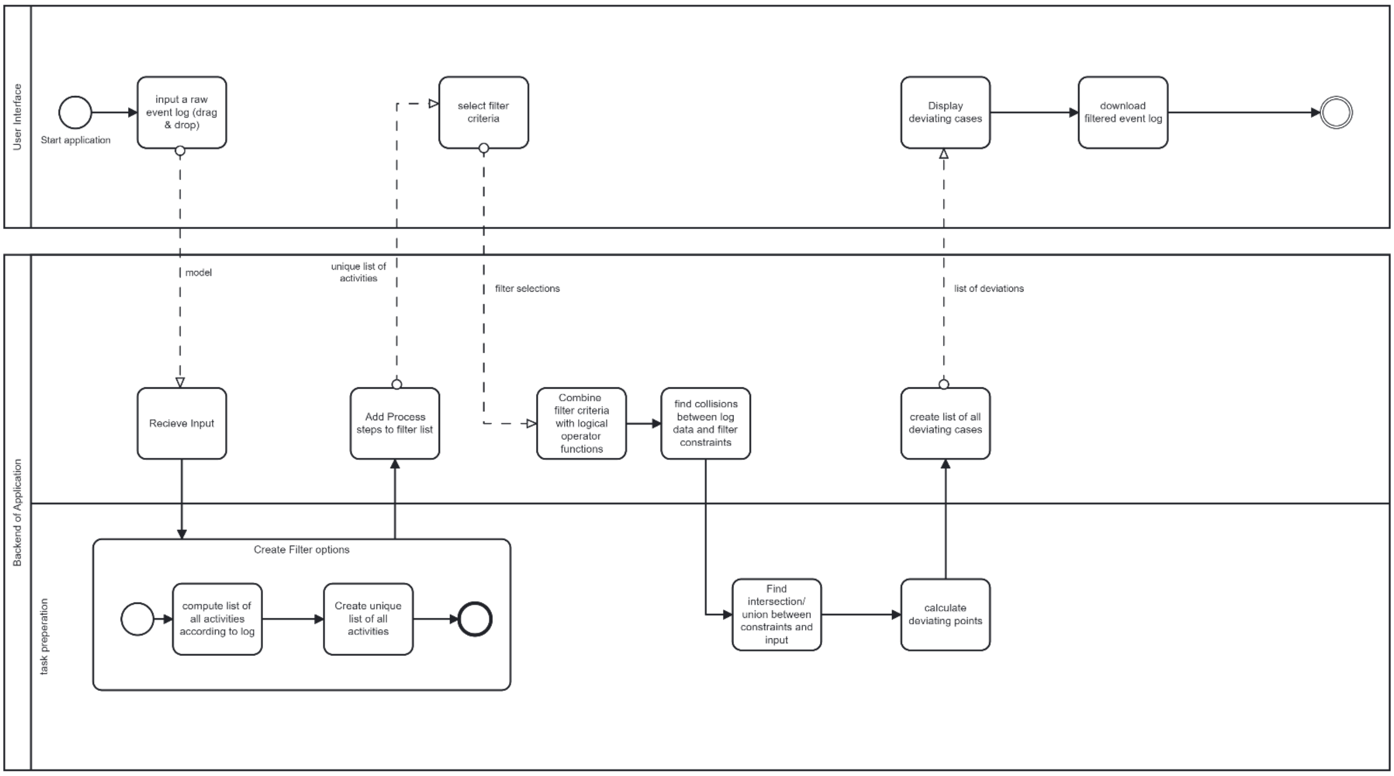
The user should be able to filter the event log data by all features (activity, timeframe, ID etc.) (LL6)

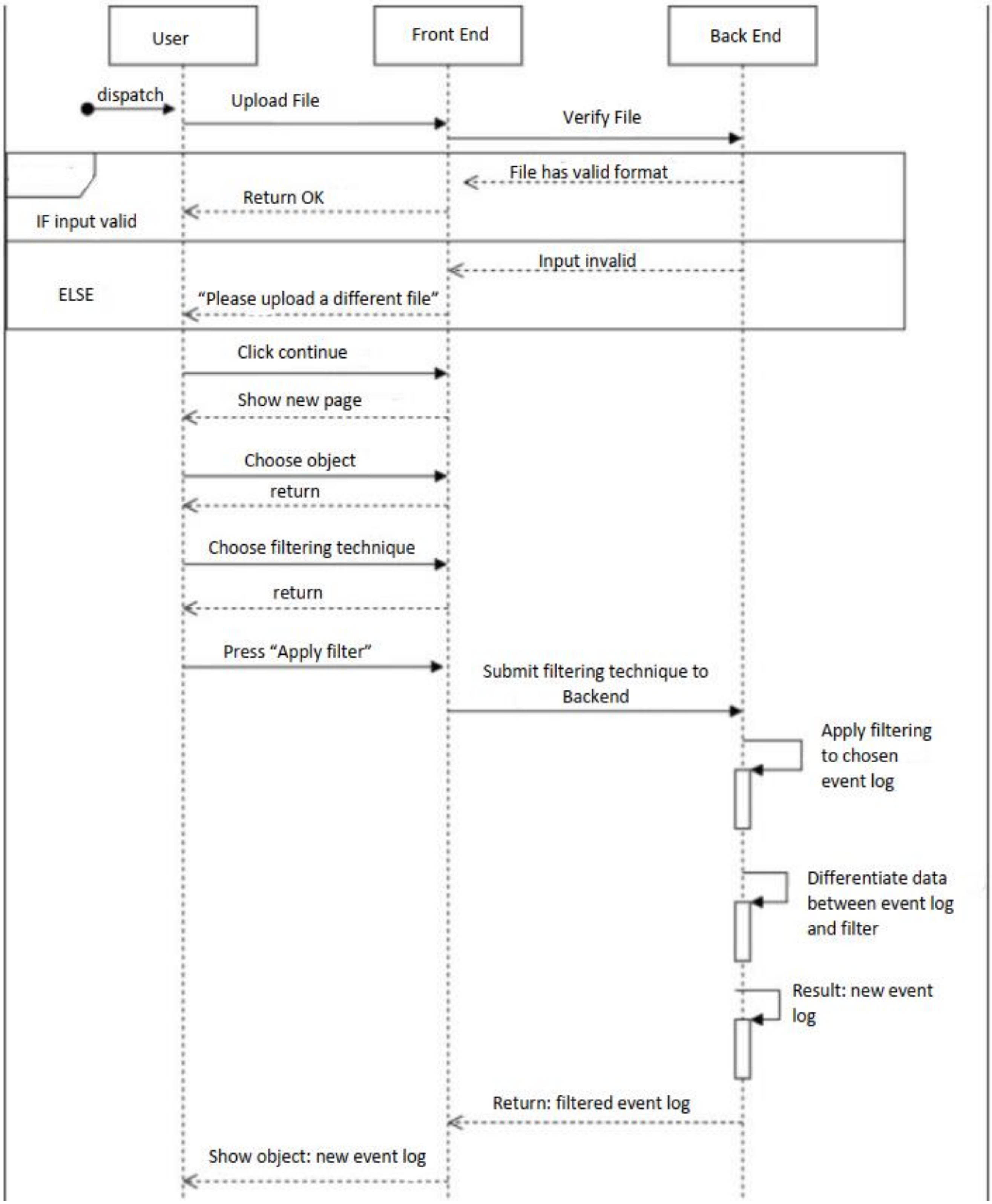
2. Functional model

In the following chapter we will establish a functional model which will help understanding a use case of our application. Such models concentrate on describing a dynamic process. We will depict an end to end run of our program in a data – flow diagram. In each step we will assign an action or datapoint to the interface or backend of our application.

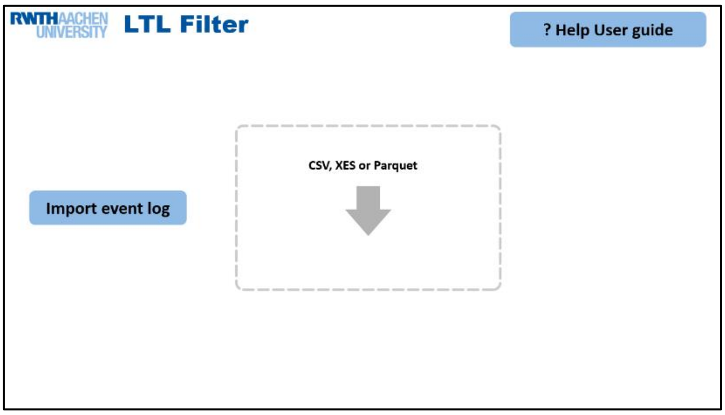
The interface: The user will be able to use the interface to upload a raw event log using drag and drop. The input data may be in XES, CSV or Parquet format. After the processing in the backend the user may select certain filters by composing LTL rules and filtering standard features of the event log such as timestamps and IDs. After setting the constraints the user will be able to see the Process IDs which deviate from the rules which have been set and download a filtered version of the input file in the format of choice.

Backend: The backend is there to process all the user selections and create the event log according to the user constraints. The first which happens here is that according to the event log, a list of all unique process logs is created. The user will be able to connect activities with LT conjugators and compose rules for filtering. Furthermore, multiple filters are combined using Intersection and Union operators (AND, OR). With these operators a final list of deviations will be passed on to the interface and displayed for the user.

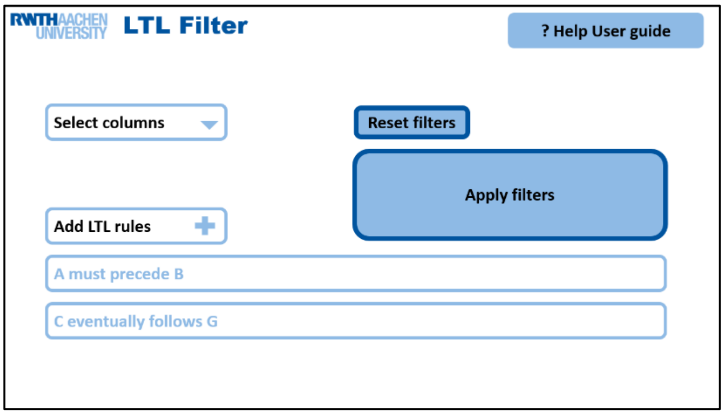

User Interface
Start application
input a raw event log (drag & drop)
select filter criteria
Display deviating cases
download filtered event log
Backend of Application
model
unique list of activities
filter selections
list of deviations
Recieve Input
Add Process steps to filter list
Combine filter criteria with logical operator functions
find collisions between log data and filter constraints
create list of all deviating cases
task preperation
Create Filter options
compute list of all activities according to log
Create unique list of all activities
Find intersection/ union between constraints and input
calculate deviating points


User
Front End
Back End
dispatch
Upload File
Verify File
File has valid format
Return OK
IF input valid
Input invalid
ELSE
“Please upload a different file”
Click continue
Show new page
Choose object
return
Choose filtering technique
return
Press “Apply filter”
Submit filtering technique to Backend
Apply filtering to chosen event log
Differentiate data between event log and filter
Result: new event log
Return: filtered event log
Show object: new event log

3. Quick design



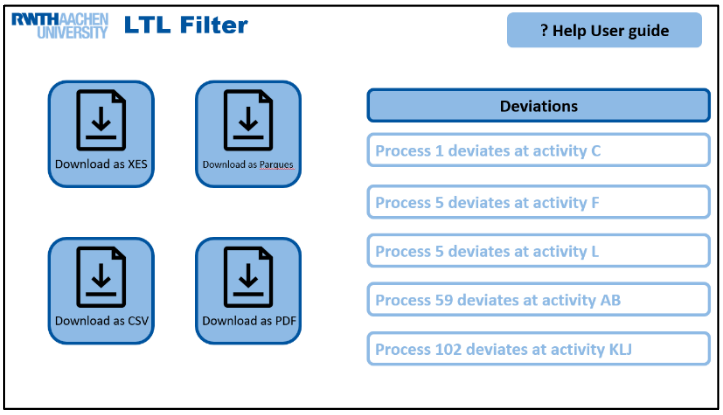
The first screen is where the user will be able to import an event log from his or her device using either drag & drop or by pressing the "import event log" button



A must precede B

C eventually follows G

After successfully uploading the event log the user can select columns from the file which he/she would like to keep and can compose LTL rules based on a list of unique activities in the log and LTL operations. The user can confirm or reset the selection by pressing either button on the right side of the screen.



Process 1 deviates at activity C

Process 5 deviates at activity F

Process 5 deviates at activity L

Process 59 deviates at activity AB

Process 102 deviates at activity KLJ

A successful run application run will take the user to the final screen, where the filtered event log can be (optionally) downloaded in different formats and can view the processes which deviated from the constraints set on the previous screen.

4. Validation and verification of requirements

It is crucial that all parties involved concur that the requirements are properly articulated to increase the quality of the project's requirements. Our team emailed the client (Alessandro Berti) in order to receive the necessary feedback regarding our requirements. He pointed out certain adjustments, and they have been implemented as follows:

1. When compared with the functionalities, development requirements such as testing, and code readability are not particularly significant in terms of customer requirements
2. Regarding the user requirements, we must anticipate that many users will not be concerned with the efficient handling of users' errors
3. Use Python 3.10 instead of Python 3.11 as PM4Py is not currently compatible with the latest Python version

5. Requirement Management Tools [1]

The requirements engineering phase is dynamic and (un)officially never-ending, as requirements and uses cases can change, be taken out of or added to a project at almost any point in time of the product development. Therefore, it is crucial for us to keep an overview of all changes we make in our requirements and everything revolving around them. Additionally, the whole team must be informed about those changes, which is why we will use multiple tools to help us during the requirements engineering phase. In the following paragraphs we will introduce some of the software tools that we could employ for this phase.

5.1. Jama Software [2] [3]

Founded in 2007 and located in Portland, Oregon, Jama Software is a software company that develops smart product development platforms for requirements, risk and test management. Their product line enables organizations to accelerate cycle times from inception to production. As a web-based platform, one of Jama’s strong sides is its availability to organizations and teams spread all over the world. Additionally, Jama offers tools to identify dependencies, enable real-time communication between all stakeholders as well as up to date testing and validation frameworks. Those can be very useful to ensure the correct behavior of not only the components of the software to be developed, but also of an organization’s work as a whole. In fact, Jama offers end-to-end traceability frameworks visualizing connections and interdependencies between and gaps within the specified requirements. Finally, Jama offers a git-like branching system that permits the parallel development of multiple variants of a product. This is just a small overview of tools Jama has to offer, but already shows why companies like NASA, Boeing, and Caterpillar use Jama in their product engineering.

5.2. Modern Requirements [4] [5]

Modern Requirements offers a single-application solution for requirements management. Within that framework called Microsoft's Azure DevOps, teams can easily create, manage and automate requirements. It allows full interconnection, as requirements, test cases and code repositories get to live under the same roof, which eliminates duplications and gaps. In fact, Modern Requirements allows users to create diagrams and use cases in self-made editors that can be analyzed by intelligent software to help identify requirements, scenarios and test cases automatically. The "Smart Docs" functionality implements the ability to build fully versioned documents, such that the requirements documents evolve at the same time (or almost) as the project itself, keeping track of all dynamics. Traceability matrices visualize relationships between all items of a project and can handle test coverage as well. Moreover, baselining captures all changes of a project throughout its existence. Obviously, the only downside is that it is a Microsoft product, haha.

5.3. Visure [6]

An easy to use, customizable solution for requirements, risk and test management is what makes Visure a key player in the requirements management software industry. Visure's key asset is its very flexible and customizable platforms that allow for stakeholders to communicate very easily. Said flexibility comes from the ability to easily integrate MS Offices features and from working very well with tools like IBM DOORS, TFS, MATLAB, Sparx EA and others. Integrating those programs permits non-Visure users to communicate well with the requirements management team without any adaptation. Many workflow models can be applied when using Visure, i.e., agile, waterfall or V-model workflow. Moreover, Visure includes end-to-end traceability, graphical representation of workflow models, as well as AI-based quality analysis of a team's requirements. Finally, Visure is one of the cheapest enterprise-level requirements management platforms on the market as of today.

5.4. Orcanos [7]

Orcanos is well known for its state-of-the-art visualization and reporting feature tools. Those include real-time, user-friendly dashboards that are very intuitive for stakeholders as well. In addition, the so called DocGen feature allows users to import and export Word files very easily. Other than that, Orcanos offers all essential requirements management tools, i.e. end-to-end traceability and collaboration and test management features. Moreover, Orcanos users praise conscious costumer support.

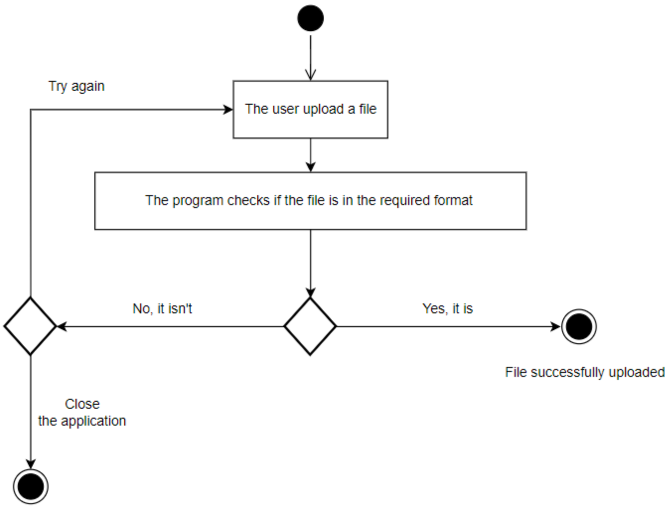
Orcanos and users state that their software is well-suited for full Quality Management Systems (QMS) and Application Lifecycle Management systems (ALM). That is due to Orcanos’ ability to simply upgrade requirements management plans to those higher-level systems. From there on, risk assessment, costumer complaint management as well as quality control are easily usable.

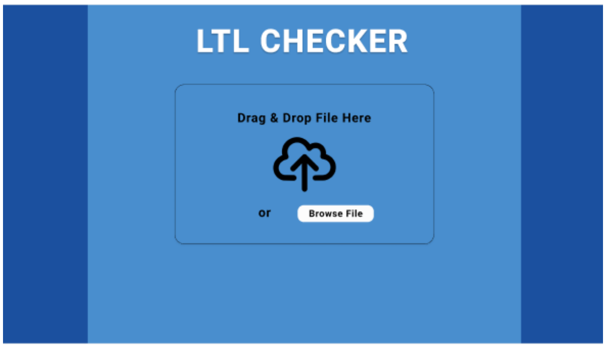
The only setback is the lack of pre-built integrations and Orcanos’ relatively high costs: At the moment, they demand 69$/month for each member of a project, where at least five members must be accounted for once the free 30-day trial has expired.

6. Prototype with activities diagrams

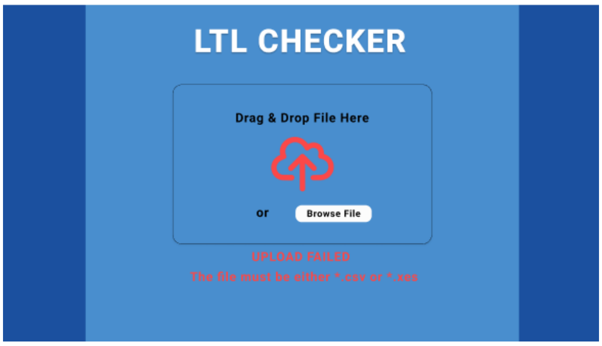
According to the requirements, we created a prototype of the interface that should be intuitive, understandable (HL3), flexible and functional (HL3).

6.1. Upload an input file


The user upload a file
The program checks if the file is in the required format
Try again
No, it isn't
Yes, it is
File successfully uploaded
Close
the application



On the start page, the user can upload an input file either in CSV or XES format (HL1, IL13).

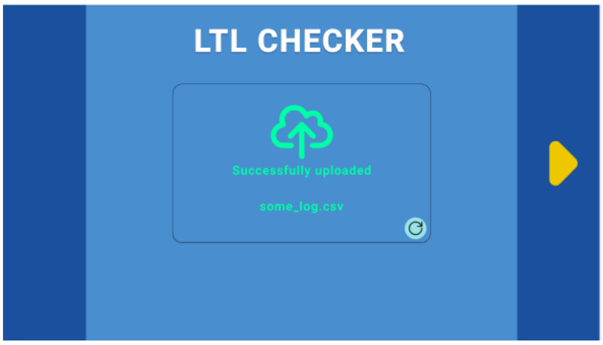


UPLOAD FAILED

The file must be either *.csv or *.xes

In case of wrong format, the software returns an error (HL10).

- The cloud icon is red.
- There is the error message "Upload failed. The file must be either *.csv or *.xes".
- The user is not able to go forward.



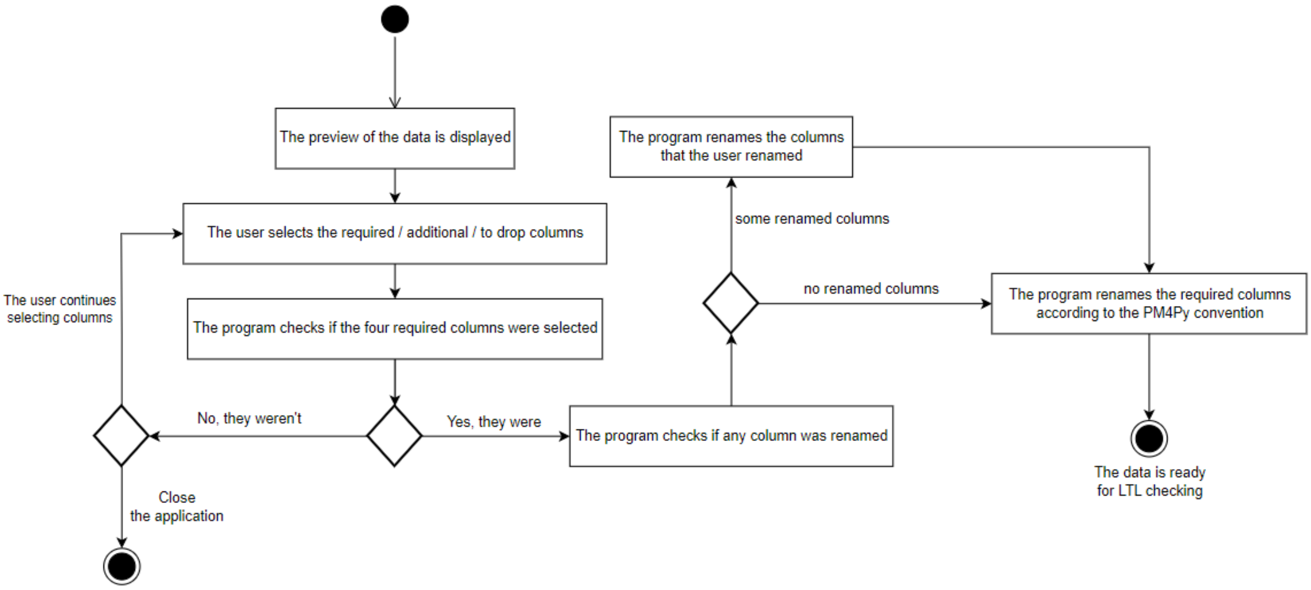
Successfully uploaded

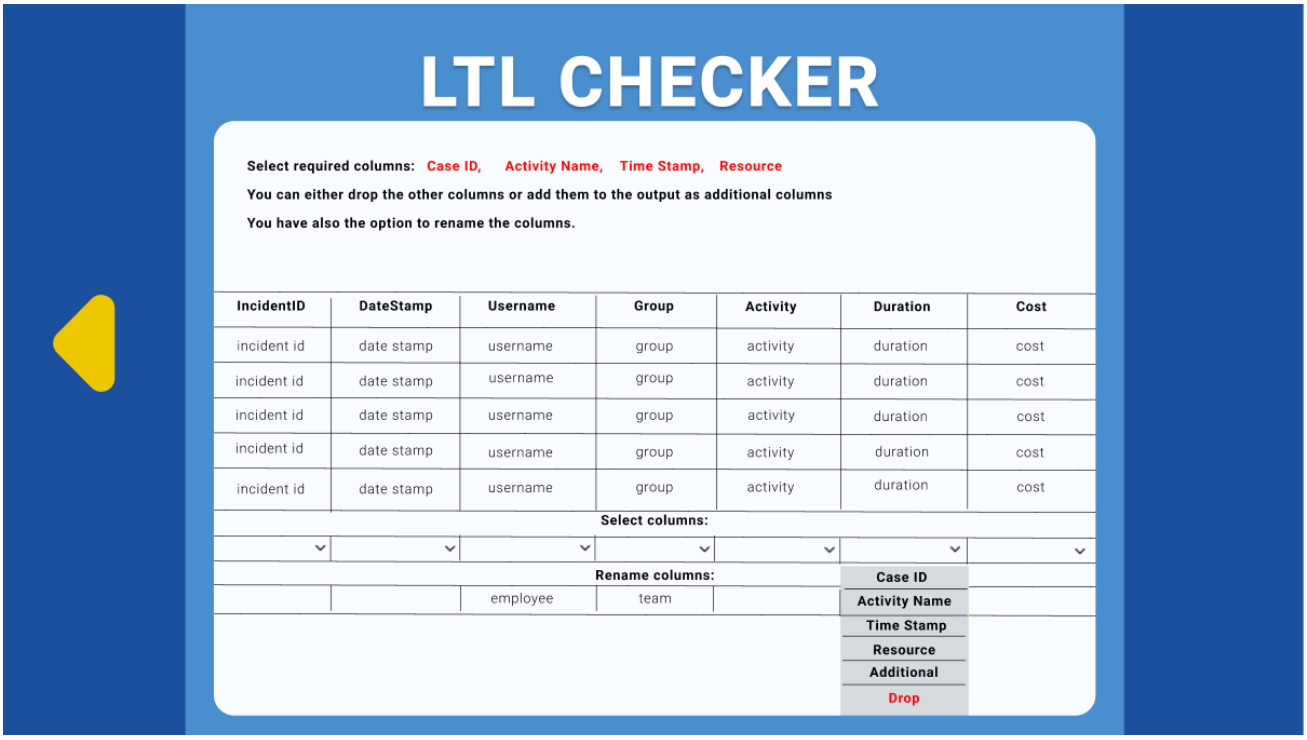
some_log.csv

File successfully uploaded.

- The cloud icon is green.
- The "next" button is shown.
- The user can change the file (reload button).

6.2. Select required columns, rename them


The preview of the data is displayed
The user selects the required / additional / to drop columns
The user continues selecting columns
The program checks if the four required columns were selected
No, they weren't
Yes, they were
Close the application
The program checks if any column was renamed
some renamed columns
no renamed columns
The program renames the columns that the user renamed
The program renames the required columns according to the PM4Py convention
The data is ready for LTL checking


LTL CHECKER

Select required columns: Case ID, Activity Name, Time Stamp, Resource

You can either drop the other columns or add them to the output as additional columns

You have also the option to rename the columns.

IncidentID	DateStamp	Username	Group	Activity	Duration	Cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost

Select columns:

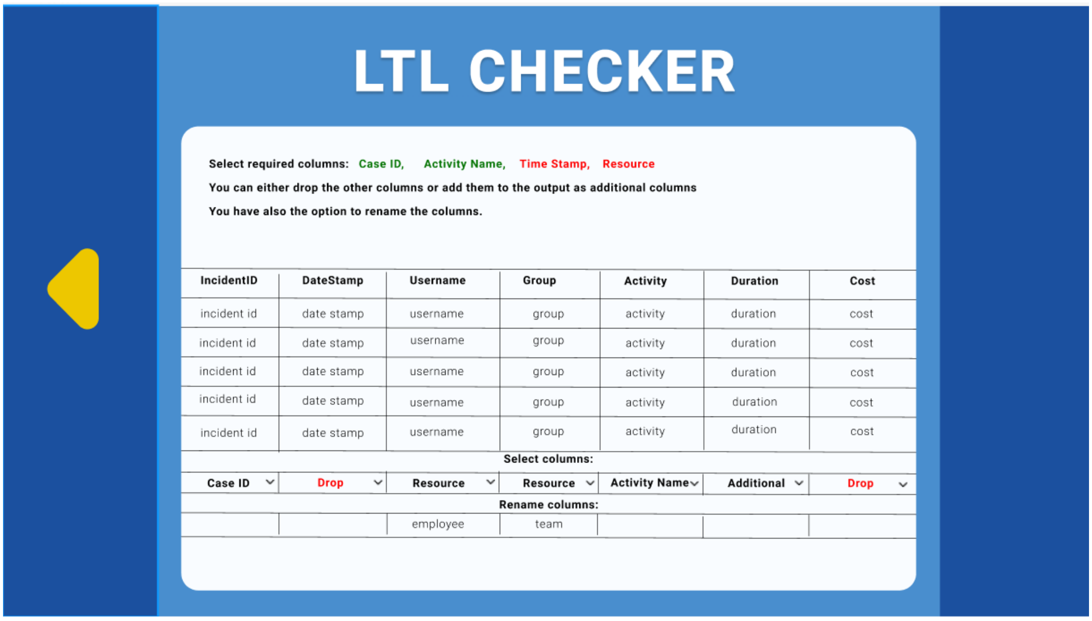
Rename columns:

		employee	team			

The user can select certain features (HL7) and required columns (IL1) from the input data and rename them (LL18).

Exactly one column should be selected for each required column.

Error handling for required columns (HL10)	
Column	**Reason**
Case ID	There is no column selected for "Case ID".
Activity Name	There is no column selected for "Activity Name".
Time Stamp	There is no column selected for "Time Stamp".
Resource	There is no column selected for "Resource".



LTL CHECKER

Select required columns: Case ID, Activity Name, Time Stamp, Resource

You can either drop the other columns or add them to the output as additional columns

You have also the option to rename the columns.

IncidentID	DateStamp	Username	Group	Activity	Duration	Cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost

Select columns:

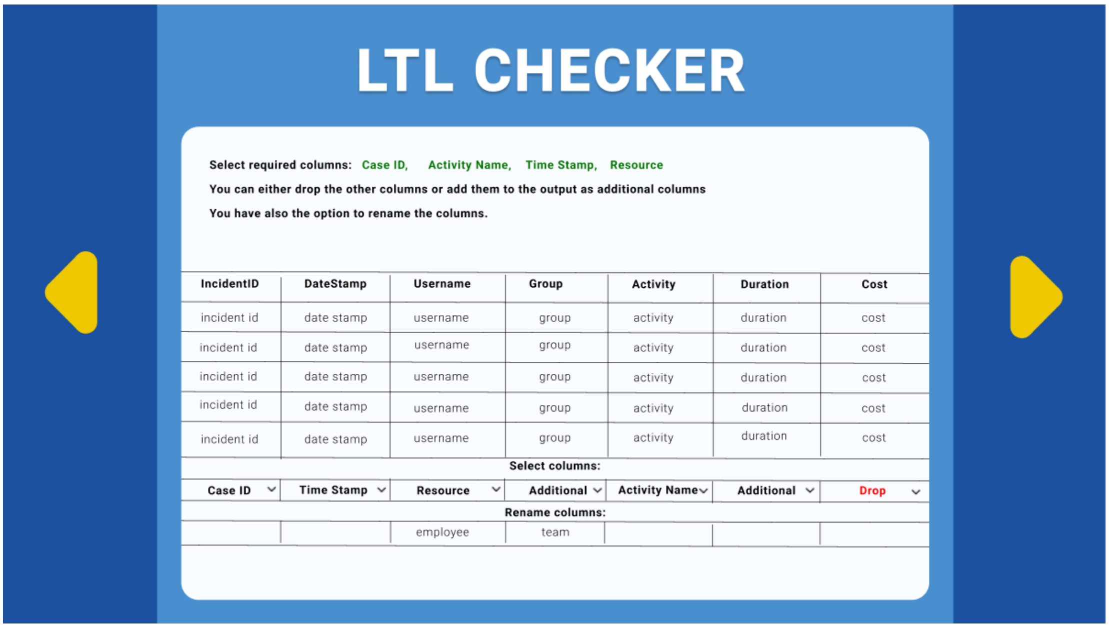
Case ID	Drop	Resource	Resource	Activity Name	Additional	Drop

Rename columns:

		employee	team			

Exactly one column should be selected for each required column.

Error handling for required columns (HL10)	
Column	**Reason**
Case ID	There is exactly one column selected for "Case ID".
Activity Name	There is exactly one column selected for "Activity Name".
Time Stamp	There is no column selected for "Time Stamp".
Resource	There is more than one column selected for "Resource".



LTL CHECKER

Select required columns: Case ID, Activity Name, Time Stamp, Resource

You can either drop the other columns or add them to the output as additional columns

You have also the option to rename the columns.

IncidentID	DateStamp	Username	Group	Activity	Duration	Cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost
incident id	date stamp	username	group	activity	duration	cost

Select columns:

Case ID	Time Stamp	Resource	Additional	Activity Name	Additional	Drop

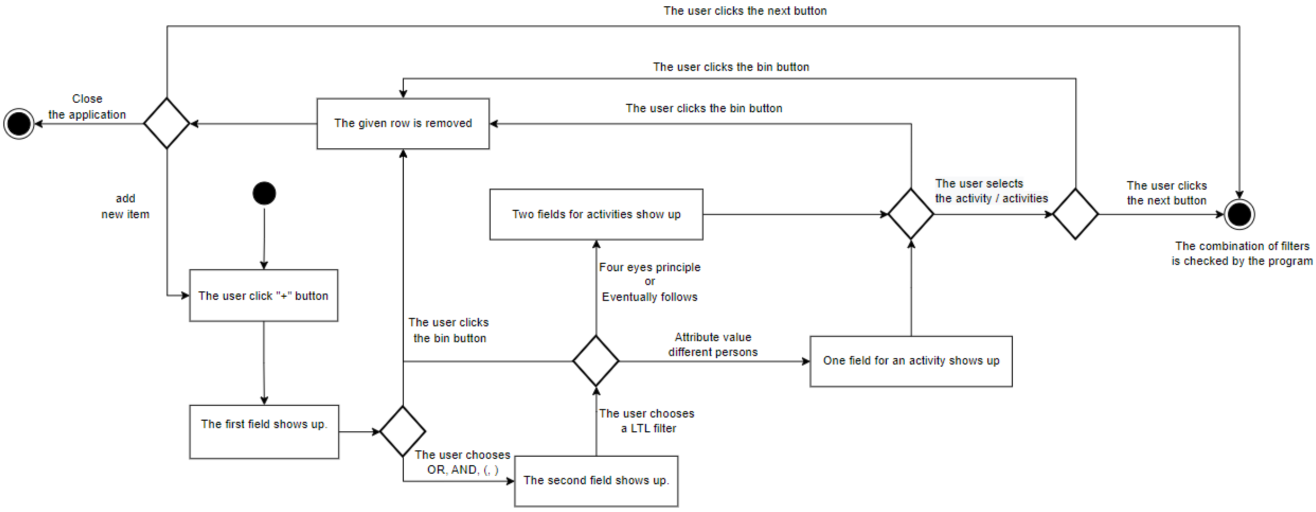
Rename columns:

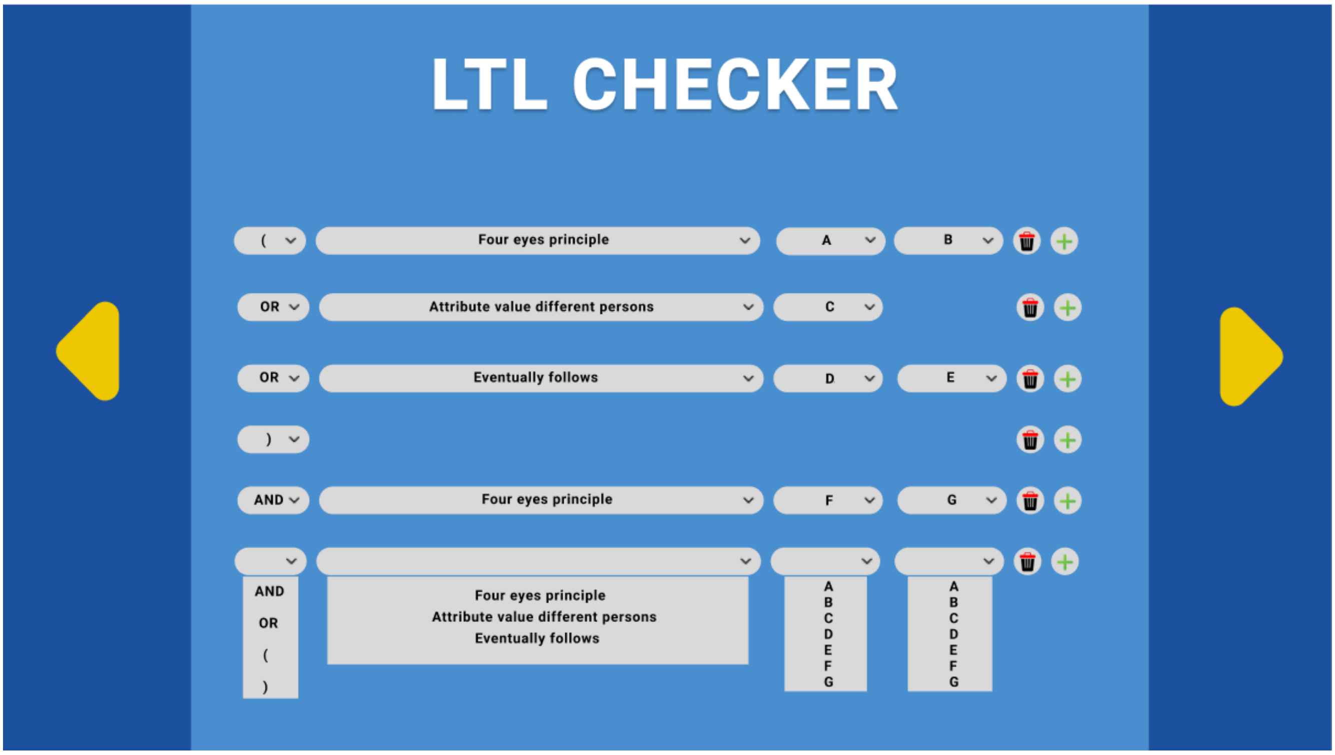
		employee	team			

Exactly one column should be selected for each required column.

Error handling for required columns (HL10)	
Column	**Reason**
Case ID	There is exactly one column selected for "Case ID".
Activity Name	There is exactly one column selected for "Activity Name".
Time Stamp	There is exactly one column selected for "Time Stamp".
Resource	There is exactly one column selected for “Resource".

6.3. Combine filters


The user clicks the next button
The user clicks the bin button
Close
the application
The given row is removed
The user clicks the bin button
add
new item
Two fields for activities show up
The user selects
the activity / activities
The user clicks
the next button
The combination of filters
is checked by the program
The user click "+" button
Four eyes principle
or
Eventually follows
The user clicks
the bin button
Attribute value
different persons
One field for an activity shows up
The first field shows up.
The user chooses
a LTL filter
The user chooses
OR, AND, (,)
The second field shows up.



LTL CHECKER

(| Four eyes principle | A | B

OR | Attribute value different persons | C

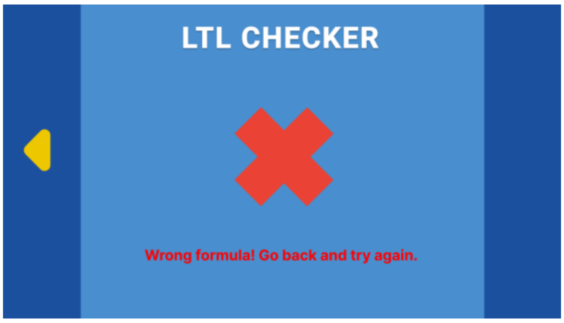
OR | Eventually follows | D | E

)

AND | Four eyes principle | F | G

The user can select different LTL filters (HL2, IL14) and combine them with logical operators like AND and OR (HL5) in a flexible way (IL16).

The software verifies the correctness of the filter combination.

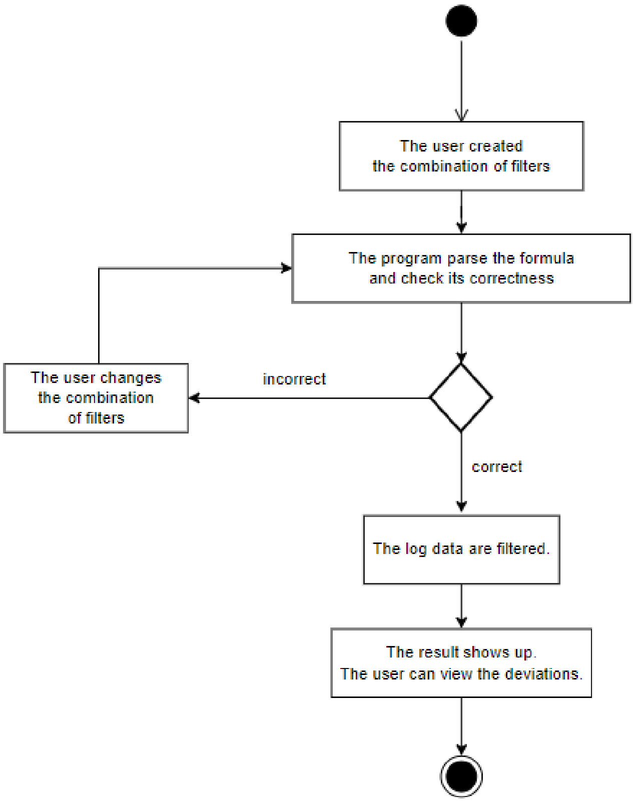


LTL CHECKER

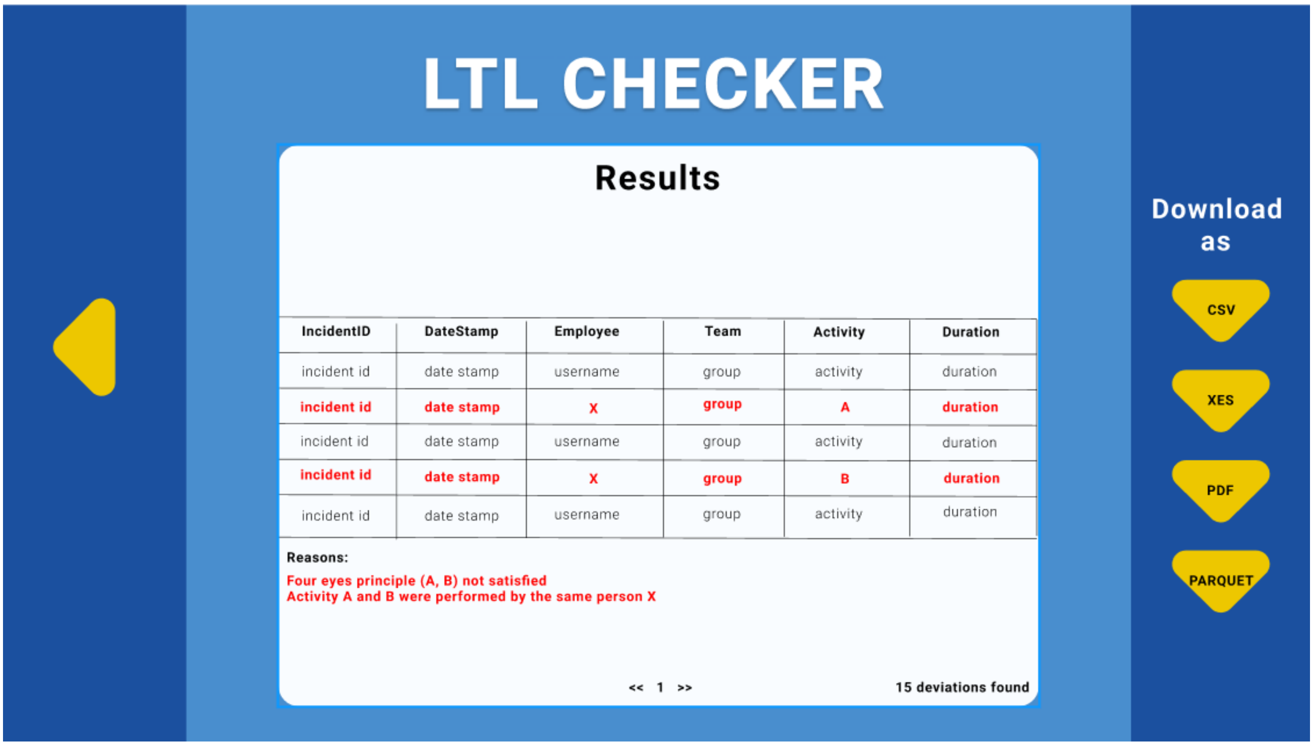
Wrong formula! Go back and try again.

LTL CHECKER

Please wait


The user created the combination of filters
The program parse the formula and check its correctness
incorrect
The user changes the combination of filters
correct
The log data are filtered.
The result shows up. The user can view the deviations.

6.4. Results



LTL CHECKER

Results

IncidentID	DateStamp	Employee	Team	Activity	Duration
incident id	date stamp	username	group	activity	duration
incident id	date stamp	X	group	A	duration
incident id	date stamp	username	group	activity	duration
incident id	date stamp	X	group	B	duration
incident id	date stamp	username	group	activity	duration

Reasons:

Four eyes principle (A, B) not satisfied
Activity A and B were performed by the same person X

The application shows the exact points of deviation (HL6).
The user can select the format of the filtered logs (IL4) and save them to the device (HL4, IL11).

7. Phase Review

7.1. Ugo Detaille

Clearly, this phase has been way harder than the previous one. First of all, defining and categorizing requirements into different levels and categories was not an easy task. Moreover, we encountered dependencies for the first time within different tasks of this phase. In order to create functional models, we had to be done with the requirements engineering first, and for the requirements engineering to be complete, we had to understand very precisely what our project was all about, not just on a vague level like in the project setup phase. So, our team faced completely new challenges this time, that I believe we mastered the right way: By communicating and distributing the workload fairly and in accordance with everyone's skills. Still, there is some room for improvement in our team. In fact, as the scrum master, I have failed to start the phase discussion early enough because I was caught up at work so much. In the future, I must make sure this works out better. But thanks to a great team that is very lenient with me, we were able to overcome the late(r) start of this phase. Let's keep it up guys!

7.2. Vishisht Choudhary

Compared to the project initiation this phase was much harder to do because we had to go into great depths of our application and how we would like to implement it. When we started working and properly distributing our tasks we were also able to do good work but I think in the next phase we should work a little more on our time and work management. We had our first meeting a little too late in my taste and also felt some pressure towards the deadline unlike last time. I think one mistake we made was to try and get everyone to be able to attend the meetings which is difficult in a group of 5. If we learn from our mistakes, I think we will be able to successfully finish our without any pressure.

7.3. Teodora Staneva

This second part of our project was a bit more challenging in comparison to the last one. We had to work more together and come up with ideas about the requirements and functional model. I was a bit worried at first, because I wasn't quite sure how we are supposed to tackle the document, but everyone was really helpful, and we managed to pull through. We started a couple of days late, but quickly managed to reorganize and do our tasks one by one. Now after we are done with this document, I think every one of us has a better understanding of what we are supposed to do in the future and are ready for the first sprint.

7.4. Mohammed Amine Kooli

Certainly, the requirements engineering phase was a little more challenging than the project initiation phase. Given that we plan our upcoming work in the project throughout the requirements engineering phase, it is essential that we produce a strong document. It is therefore crucial to properly assess the requirements and ensure their correctness. The fact that the various sections of the requirements engineering phase were interdependent—for example, the functional model could only be developed when the requirements of our program were known—posed the biggest challenge. As a result, work coordination was even more important than during the initial phase. We overcame this obstacle by first establishing the basic requirements as a team and then splitting the remaining work so that each requirement was satisfied before beginning a new section. I appreciate that everyone engaged in the phase and was receptive to outside feedback. For the project's remaining phases, I believe that we will continue to work together effectively.

7.5. Daniel Henel

In my opinion, the second phase of our project was definitely more complex and time-consuming than the previous one. We attempted to define all requirements and functionalities that should be implemented in our application.
Furthermore, their level of importance and the relationship between them were determined. Subsequently, we had to specify software processes and user interactions that had to be ordered according to the proper workflow that is presented on the functional model and activities diagrams. We also created a prototype of our application, simple design of the graphical user interface, that will be helpful in the first sprint to imagine how the program should work. After the phase, we understand our customer’s needs better and we are ready to start the programming part of the project, but to go through it smoothly, we should plan our tasks better.

8. References

[1] The Digital Project Manager. (2022, October 28). *10 Best Requirements Management Tools & Software of 2022*. The Digital Project Manager. R https://thedigitalprojectmanager.com/tools/requirements-management-tools/#best-requirements-management-tools-review

[2] Requirements management software: Jama Connect. Jama Software. https://www.jamasoftware.com/?utm_source=google&utm_medium=cpc&utm_campaign=emea-search-jama-software-brd-max-conv&utm_adgroup=software&utm_term=jama+software&utm_content=596485999428&_bm=17002130372137252146273&gclid=EAIaIQobChMInLre4OCl-wIVBIXVCh339wDwEAAYASAAEgL6nvD_BwE

[3] *Jama software company profile - office locations, competitors ... - craft*. (n.d.). https://craft.co/jama-software

[4] *Requirements management tools*. Modern Requirements. (2022, October 29). https://www.modernrequirements.com/products/modern-requirements4devops/

[5] The Digital Project Manager. (2022, October 28). *What is modern requirements4devops? detailed overview & explanation of requirements management features*. The Digital Project Manager. https://thedigitalprojectmanager.com/tools/modernrequirements4devops-overview/

[6] *Home new*. Visure Solutions. (2022, November 9). https://visuresolutions.com/home-new

[7] *Requirements Management Tool for Medical device*. ALM Software Tool – Orcanos Software – ALM And Quality Management. (2022, November 9). https://www.orcanos.com/compliance/requirements-management/

[8] GmbH, C. S. (n.d.). Cawemo. https://cawemo.com/