Software Requirements Specification

ReviewOger Reloaded

Prepared by:

Hannah Meinhardt, Jakob Rechberger, Bastian Schindler, Nicolas Stoll, Richard Wünsch

Supervisor:

Prof. Dr. Daniel Kulesz

Version 0.3

29.03.2023

Table of Contents

Revision History

Revision	Date	Author(s)	Changes
0.1	29.03.2023	Jakob, Richard, Nicolas	Initial Version.
0.2	31.03.2023	Jakob, Richard, Bastian, Hannah, Nicolas	Harmonize naming conventions, correction of spelling errors, changed user groups
0.3	12.04.2023	Jakob, Richard, Bastian, Nicolas	Improve Use Cases, added additional features, changed AB-Review definition, rephrase quality requirements
1.0	28.04.2023	Nicolas	Extended AB-review definition, added email template requirement, added data formats to imports/export, slight changes in UC-Diagram

1. Introduction

1.1 Purpose

The software will be designed for the organization of a Technical Review, with special emphasis on the computer science sphere. However, anyone who may want to organize a technical review can use this product.

This tool covers all steps necessary to arrange the groups based on roles and individual preferences by the organizer of the Review.

1.2 Document Conventions

The user of ReviewOger (Reloaded) will be referred to as the ‘User’.

The predecessor of the new product will be referred to as ReviewOger. The new software will be called ReviewOger Reloaded.

Functional requirements will be sorted by their corresponding use case. If lower-level requirements have a high priority, that will be assumed for their associated high-level requirements as well. The priority levels in this document are ‘high’, ‘medium’ and ‘low’.

1.3 Intended Audience and SRS Structure

The document is addressed to:

- The whole project team
- stakeholders of the project
- anyone who wants to participate in this open source project

The SRS contains a short overview on the scope and functionality of ReviewOger Reloaded. Second, the design will be proposed as well as interfaces needed by the product. Third, we list all the requirements - functional and non-functional. The following chapter contains all Use Cases of ReviewOger Reloaded drawn in a Use Case Diagram and in writing. Lastly, the appendix contains additional information we refer to in the document.

1.4 Product Scope

This Software will be primarily used for people who are using Technical Reviews as a part of the development process of one's product. The tool will make it easy to sort multiple people into different technical review groups.

Additionally the tool offers functionality specifically designed to assist faculty staff at universities.

The goal is to implement this tool with available standard frameworks and tools as well as publishing all its content as an open source project.

2. Overall Description

2.1 Product Perspective

ReviewOger has been in use for several years. However, it is now outdated and desperately in need of a rewrite. Our Product, ReviewOger Reloaded, will be a modern successor for the old product, providing the same functionality as well as multiple new features. The product will be a progressive web app (PWA), making the application portable and making an installation optional. ReviewOger Reloaded will be a stand-alone software - i.e. it is not part of a larger system.

2.2 Product Functions

This product should make the process of organizing technical reviews easier by allowing faculty staff to import lists of participants from moodle or ilias and automatically assigning them to a group and role. It also supports the user scheduling the place and date for their respective group meetings.

After a successful creation of groups and dates the author can inform the participants via mail, as well as create a file that can be imported into RevAger, a technical review execution tool.

2.3 User Classes and Characteristics

Our user class is anyone planning to organize technical reviews. Specifically emphasis will be on faculty staff. We can assume a high level of education and a solid understanding of what the software should be used for. Furthermore we can assume a certain affinity to the usage of web apps. Therefore this user class should be able to use the software without any training or supervision.

2.4 Operating Environment

ReviewOger Reloaded is a progressive web app (PWA). The app should be able to run on any device that has a browser installed and a working internet connection for the first request to the app. Some PWA functionality may not be supported by all browsers or need a current version of a browser.

2.5 Design and Implementation Constraints

The product will be implemented as a Progressive Web App (PWA). Therefore it has to meet several requirements to be seen as such (e.g. platform independence, offline availability, installability).
The use of a web framework such as React is recommended, but not required. Usage of JavaScript/Typescript, CSS and HTML is mandatory. Optionally some parts may be written in e.g. Rust or C to improve performance.
Due to security and maintenance concerns, the app cannot have a backend. All data may only be stored on a local device. The project does not include the maintenance of ReviewOger Reloaded.
Since ReviewOger Reloaded is an open source project it will be published under a public license.

2.6 User Documentation

User documentation will consist of a README providing a guide to the basic functionality of the product. Furthermore a more detailed user documentation will be provided on readthedocs.org

2.7 Assumptions and Dependencies

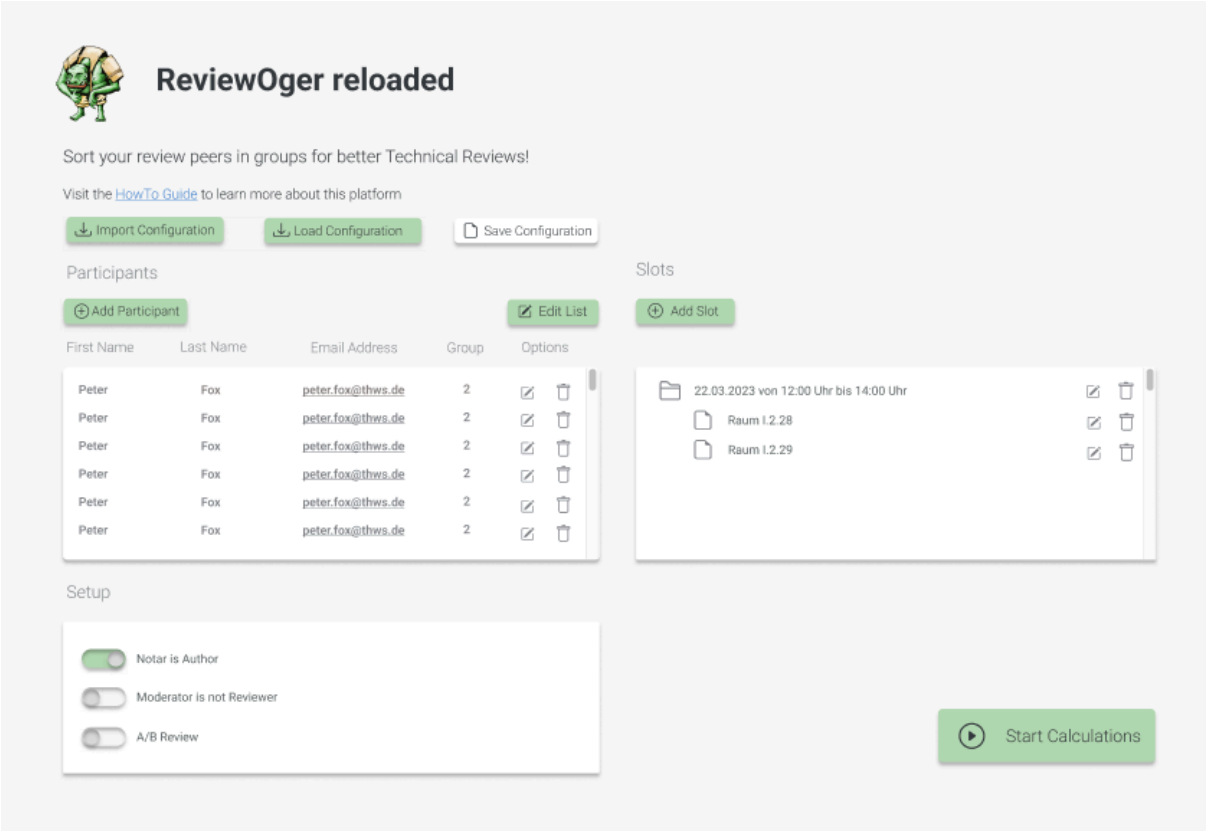
As the Algorithm will be implemented in a new language (JavaScript) some dependencies or features may not behave like the predecessor although known limitations will be accounted for in the project.

3. External Interface Requirements

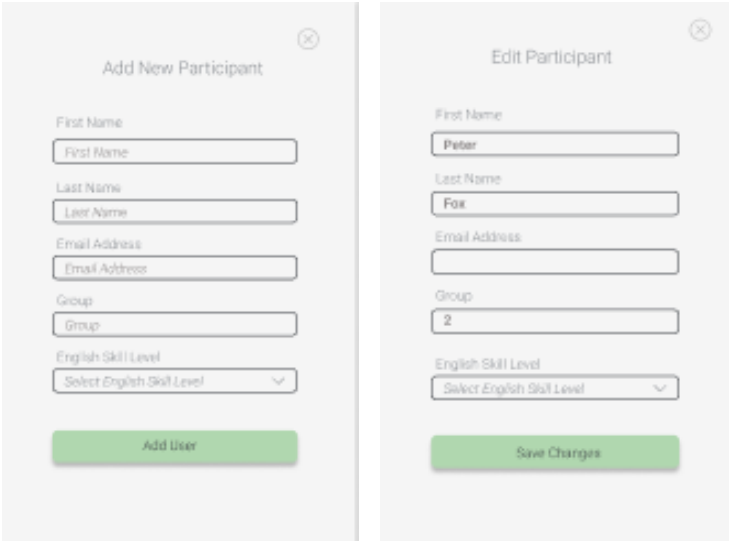
The following chapters contain information on different interfaces in our project. It provides an overview of our User Interfaces (UI) as well as Hardware, Software and communication Interfaces

3.1 User Interfaces

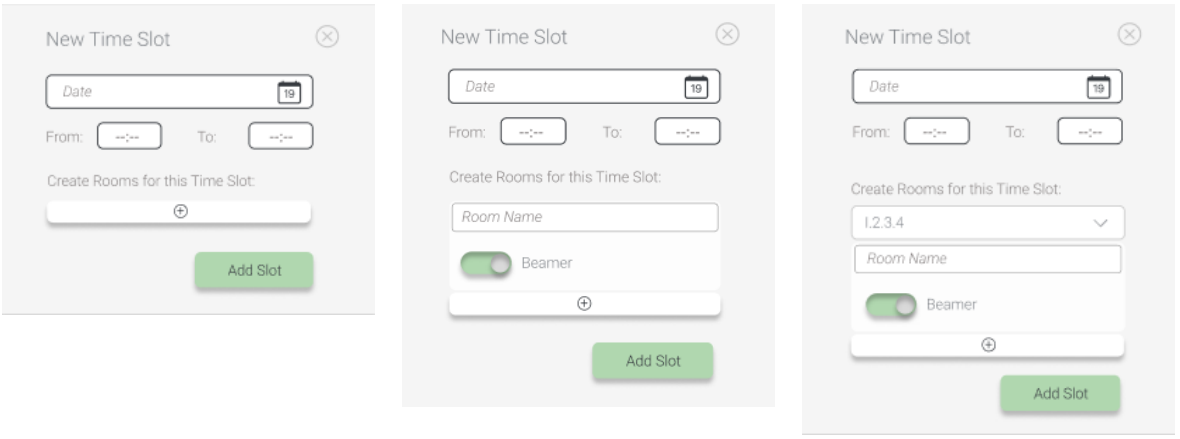
The user interface will be designed in Figma. Please check our Figma project for reference: ReviewOger – Figma



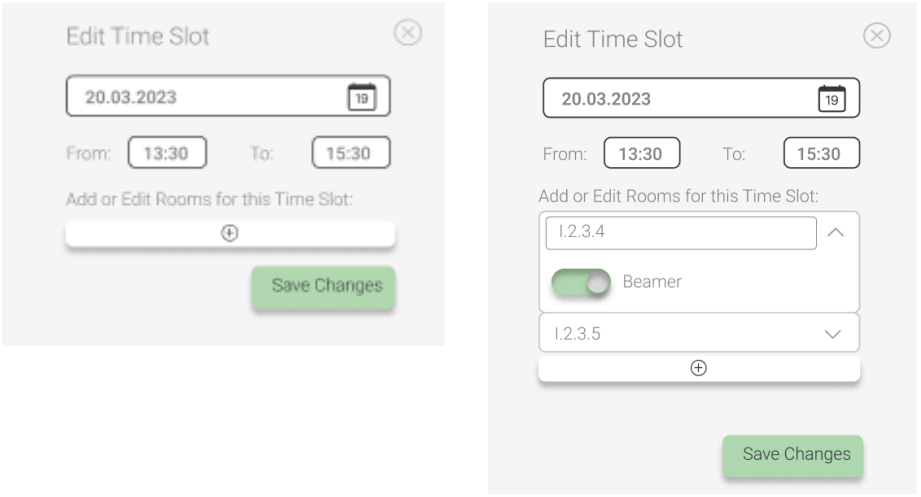
Landing page



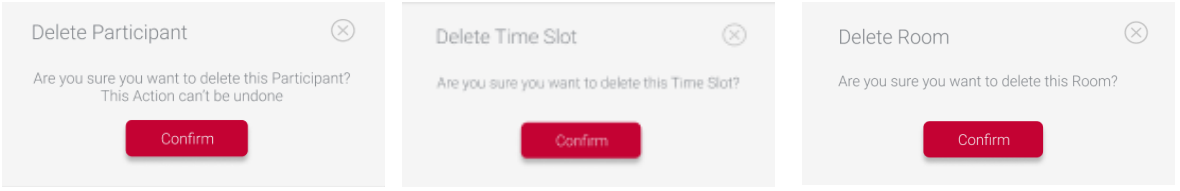
Modals to add and edit Participants



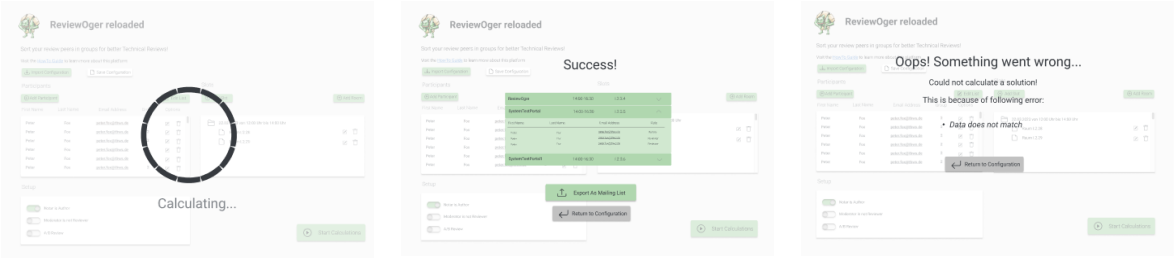
Modals to Add a New Time Slot and allocate Rooms to this Slot



Modals to edit a selected Time Slot



Verify Participant Deletion *Verify Time Slot Deletion* *Verify Room Deletion*



Loading Screen and Success/Fail message

3.2 Hardware Interfaces

Not applicable.

3.3 Software Interfaces

ReviewOger Reloaded will be able to create files designed to be imported in RevAger or RevAger Lite. Those files will consist of information on the team composition and role distribution.

3.4 Communications Interfaces

The product is required to be installable. This may only be achieved under certain constraints. Chromium-based browsers on Android require the PWA to be served on HTTPS.

4. Functional Requirements

The following chapters describe the general requirements of the product.

4.1 Import List of Students:

In order to enable seamless usage with e-learning platforms, it is necessary to have the capability to correctly import files provided by these platforms.

4.1.1 Data Import

ID	**F-1**
Description	The user must be able to import a file provided by an e-learning platform (THWS e-learning or ilias) into the tool. The import format is csv (and JSON in later versions).
Priority	high
Dependencies	-

4.1.2 Import data formatting

ID	**F-2**
Description	The tool must be able to filter and format relevant information from an imported file containing student information.
Priority	high
Dependencies	F-1, F-3

4.1.3 Preview imported data

ID	**F-3**
Description	Before importing data from the file, the tool should display a preview of the data. The user can confirm, the tool chooses the right data fields for the import.
Priority	medium
Dependencies	-

4.2 Manage Participants

To ensure an intuitive management of participants, it is essential to have the ability to create, delete, and update room details. Furthermore the model needs to be extended.

4.2.1 Create Participant

ID	**F-4**
Description	The user can create instances of people that are in the Technical Review. Each instance has a name, surname, email address and a group.
Priority	high
Dependencies	-

4.2.2 Add attribute 'likelihood of being in multiple reviews'

ID	**F-5**
Description	The organizer of the Review can manually set how likely it is for one instance to be placed in multiple reviews.
Priority	medium
Dependencies	-

4.2.3 Add attribute German proficiency’

ID	**F-6**
Description	An additional field contains the German proficiency level of the participant based on the international grading scheme (A-C).
Priority	medium
Dependencies	-

4.2.4 Edit Participant

ID	**F-7**
Description	The user is able to change all information of a participant.
Priority	high
Dependencies	-

4.2.5 Delete Participant

ID	**F-8**
Description	The user is able to delete a participant from the list of entries.
Priority	high

Dependencies	-

4.2.6 Mass edit Participants

ID	**F-9**
Description	The user should be able to edit information of multiple participants simultaneously.
Priority	low
Dependencies	-

4.3 Manage Slots

To ensure an intuitive management of (time)slots, it is essential to have the ability to create, delete, and update room details. The requirements are listed in this section.

4.3.1 Create Slot

ID	**F-10**
Description	The user can create instances of slots that are available in the Technical Review. Each slot has a date, start and end time.
Priority	high
Dependencies	-

4.3.2 Update Slot

ID	**F-11**
Description	The user is able to change the information on a slot.
Priority	high
Dependencies	-

4.3.3 Delete Slot

ID	**F-12**
Description	The user is able to delete a slot with its corresponding rooms.
Priority	high
Dependencies	-

4.4 Manage Rooms

To ensure an intuitive management of rooms, it is essential to have the ability to create, delete, and update room details. The requirements are listed in this section.

4.4.1 Create Room

ID	**F-13**
Description	The user can create instances of rooms. A room has a room designation, start and end time. Rooms also have an attribute stating whether a beamer is available.
Priority	high
Dependencies	F-9

4.4.2 Update Room

ID	**F-14**
Description	The user is able to change the information of a room.
Priority	high
Dependencies	-

4.4.3 Delete Room

ID	**F-15**
Description	The user is able to delete a room.
Priority	high
Dependencies	-

4.5 Create RevAger (Lite) export

In order to enable seamless usage with RevAger, it is necessary to have the capability to create exports in RevAger (Lite) format.

4.5.1 Create RevAger (Lite) file

ID	**F-16**
Description	The tool must be able to create a RevAger File, that can be imported by the notary into RevAger to start the Technical Review.
Priority	high
Dependencies	F-17

4.5.2 Choose file format

ID	**F-17**
Description	The user must be able to choose between the RevAger and RevAger Lite format, depending on which tool he wants to use for the review.
Priority	high
Dependencies	-

4.6 Invoke mail client

To streamline the invitation process for the review we want the system to create an invitation addressed to the participant.

4.6.1 Create email

ID	**F-18**
Description	The tool must be able to create an email, addressed to every member of a review group. It should contain the role distribution of the given group as well as the RevAger File.
Priority	high
Dependencies	F-16

4.6.2 Create email templates

ID	**F-19**
Description	The user should be able to create templates for the email content.
Priority	low
Dependencies	-

4.7 Safe intermediate result

In order to prevent data loss and ensure seamless continuation of work, this section defines following requirements:

4.7.1 Save Progress

ID	**F-20**
Description	The user is able to store the current progress at any time. The progress will be stored in a file on the local device as a JSON file.
Priority	high
Dependencies	-

4.7.2 Restore Progress

ID	**F-21**
Description	The tool must be able to load a locally stored file to continue working on a previously started review plan. The data format is JSON.
Priority	high
Dependencies	F-20

4.8 Sort participants into review groups

To facilitate an effective and organized review process, a crucial requirement is to have the ability to sort participants into appropriate review groups, as outlined in this section.

4.8.1 Create Review Groups

ID	**F-22**
Description	The tool must contain an algorithm that can sort all people into groups containing one moderator, one notary, one author and multiple reviewers.
Priority	high
Dependencies	F-23, F-24, F-25

4.8.2 Form international review teams

ID	F-23
Description	The tool should be able to group participants based on their language proficiency (german). Those with a bad german language level can form an international team.
Priority	low
Dependencies	F-6

4.8.3 Choose review type

ID	F-24
Description	The user may choose between a Technical Review and an A/B-Review. Based on the review type the outcome of the algorithm may change.
Priority	medium
Dependencies	-

4.8.4 Static data inspection

ID	F-25
Description	The tool gives meaningful feedback to the user based on the current input. This static inspection should run before the grouping algorithm, to allow detection of possible exceptions or impossible to group input data (e.g. only one participant, …).
Priority	medium
Dependencies	-

4.8.5 Create Analytics on participants

ID	F-26
Description	The tool should be able to give the user some insight into the participants e.g. in how many reviews they take part in, as well as average, maximum and minimum number of reviews the participant has to attend.
Priority	low
Dependencies	-

4.8.6 Minimize participant wait time

ID	F-27
Description	After sorting the review teams the algorithm should look for the optimal arrangement of review meetings to minimize the time each participant has to spend waiting for their next meeting.
Priority	low
Dependencies	-

4.9 Export room plan

The room plan provides the user of ReviewOger Reloaded with an overview on the review groups and their associated rooms.

4.9.1 Create room export

ID	F-28
Description	The user should be able to export a room plan containing information on room usage and the people assigned. Free rooms should be highlighted so they can be reassigned for other purposes.
Priority	medium
Dependencies	-

4.9.2 Create door signs

ID	F-29
Description	The tool should be able to create door signs for printouts.
Priority	low
Dependencies	-

5. Nonfunctional Requirements

This chapter delves into the non-functional requirements. We further divided them into subcategories.

5.1 Performance Requirements

In order to ensure that ReviewOger operates efficiently, we need to consider the performance of the software.

5.1.1 High performance of algorithm

ID	NF-1
Description	The algorithm should run as fast as possible. The old implementation could take seconds or even minutes to find a solution, leading to frustration.
Priority	low
Dependencies	F-18

5.2 Security Requirements5

Since personal data is stored about the participants(e.g. first name, last name, language level), this data must be protected. However, since the application is implemented through a PWA, has no backend service and the data is stored exclusively through an export function on the user's local computer, no further measures need to be taken here.

5.3 Software Quality Attributes

Of course all qualities of the software quality tree should be taken into account. The following list however, contains only those qualities our customer thought of as especially important.

5.3.1 Device independence

ID	NF-2
Description	The tool is able to run on any device with a browser installed. This will help our users to plan reviews faster and independent from access to a specific computer or OS and making the switch from an old device to a newer one easier.
Priority	high
Dependencies	-

5.3.2 Understandability

ID	NF-3
Description	The user quickly understands how to operate the software, ideally without the need to look into the user manual.

Priority	high
Dependencies	-

5.3.3 Correctness

ID	**NF-4**
Description	The solution provided by the algorithm actually needs to actually be correct. This includes several aspect such as: ● no overlapping reviews for participants ● groups of roughly the same size ● every artifact will be reviewed at least once This is essential for the tool to provide any benefit.
Priority	high
Dependencies	-

5.4 Other Nonfunctional Requirements

This chapter contains requirements that don't fit in one of the categories above. However, they are important as well and therefore listed here.

5.4.1 Installability

ID	**NF-5**
Description	The user must be able to install the tool on his local device. The installation will prove useful in case of loss of internet connectivity or the server being down. Since ReviewOger Reloaded doesn’t rely on a backend it is fully functional even in those cases.
Priority	high
Dependencies	-

5.4.2 Consistency

ID	**NF-6**
Description	The tool should have a consistent user interface throughout, with predictable interactions and a uniform visual style. This helps users navigate through the app avoiding mistakes and is especially important since we want our users to be able to use the app without a manual. Consistency rules include: ● Delete buttons are always red ● users always need to confirm before making irreversible changes

	● buttons to add something always contain a ‘+’ symbol ● invalid user input prompts error messages directly below the input field ● hovering over buttons leads to the display of a short help text
Priority	high
Dependencies	-

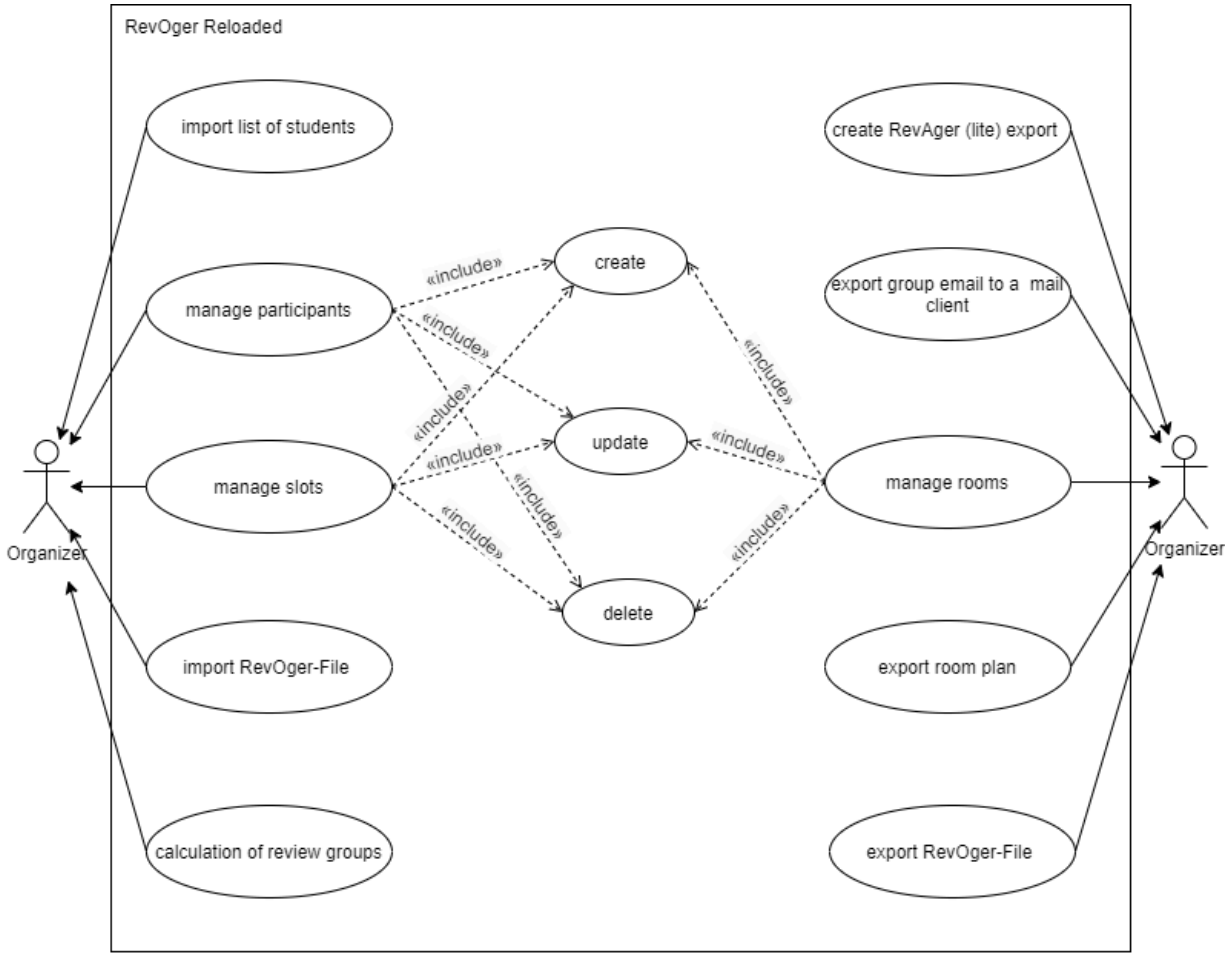
5.4.3 Input Validation

ID	**NF-7**
Description	The tool should validate user input before saving or updating data. This includes checking for required fields, format restrictions etc. High quality input is important to decrease the risk of the algorithm running into errors. This is important since the legacy ReviewOger often failed to create review groups.
Priority	high
Dependencies	-

6. Use Cases

This chapter provides a summary of all Use Cases of ReviewOger.

6.1 Use-Case Diagramm



6.2 Use-Case Description

Detailed description of the individual Use-Cases:

Name	**calculation of review groups**
ID	UC1
Target	The organizer will be able to generate groups for reviews regarding his input.
Precondition	-The organizer has completed all ReviewOger steps -The organizer has entered valid input
Postcondition	Different review groups are formed.
Postcondition in case of exception	The organizer will be informed if the algorithm has no output.
Actors	Organizer
Normal process	1. The organizer enters all needed information 2. The ReviewOger generates review groups 3. ReviewOger asks what to do with the result
Exceptions	1a There are errors in the entered data fields. An error message should be displayed with the reason for the error. 2a The ReviewOger can not find a solution.The organizer should receive an error message and be prompted to repeat the group generation.

Name	**Import list of students**
ID	UC2
Target	The organizer will be able to import a list of students from for example Moodle in the ReviewOger
Precondition	-The organizer has data that he wants to import -The data is in the needed format and a .csv-File
Postcondition	The students are imported successfully

Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the load students function 2. ReviewOger asks the organizer where he stored the current state 3. The organizer chooses the student .csv-file 4. ReviewOger includes the data from the file
Exceptions	3a The selected file is not in the right csv-structure. An error message should be displayed. 3b The selected file is not in .csv-format. An error message should be displayed.

Name	**Manage participants**
ID	UC3
Target	The organizer can create, update and delete participants
Precondition	-
Postcondition	A participant is created, updated or deleted
Postcondition in case of exception	-
Actors	Organizer
Normal process	
Exceptions	

Name	**Create participant**
ID	UC3.1
Target	The organizer can create participants.
Precondition	-
Postcondition	A participant is created.

Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the create participant functionality 2. The ReviewOger shows the window to create a participant 3. The organizer enters the participant information 4. The ReviewOger verifies the information and adds the participant to the list
Exceptions	4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not add the participant to the list. An error message with the reason should be shown.

Name	**Update participant**
ID	UC3.2
Target	The organizer can update a participant
Precondition	The participant to be updated exists
Postcondition	A participant is updated
Postcondition in case of exception	The participant will not be updated in case of an exception.
Actors	Organizer
Normal process	1. The organizer selects the participant to be updated 2. The ReviewOger shows the window to edit a participant 3. The organizer enters the participant information 4. The ReviewOger verifies the information and updates the participant in the list
Exceptions	4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not update the participant in the list. An error message with the reason should be shown.

Name	**Delete participant**
ID	UC3.3
Target	The organizer can delete a participant
Precondition	The participant to be deleted exists.
Postcondition	A participant is deleted
Postcondition in case of exception	The participant will not be deleted in case of an exception
Actors	Organizer
Normal process	1. The organizer use the delete functionality for one participant 2. The ReviewOger displays the participant information and asks the organizer if he really wants to delete this participant 3. The organizer confirms 4. The ReviewOger deletes the participant
Exceptions	1a The organizer does not select a participant. An error message should be shown with the information that the organizer must select a participant. 3a The organizer does not confirm. The delete process should be canceled. 4a The ReviewOger can not delete the participant in the list. An error message with the reason should be shown.

Name	**Delete more than one participant**
ID	UC3.4
Target	The organizer can delete more than one participant at the same time
Precondition	The participants to be deleted exist.
Postcondition	The selected participant are deleted
Postcondition in case of exception	The participants will not be deleted in case of an exception
Actors	Organizer

Normal process	1. The organizer selects the participants to be deleted 2. The organizer selects the “delete selected” functionality 3. The ReviewOger displays the participant information and asks the organizer if he really wants to delete this participants 4. The organizer confirms 5. The ReviewOger deletes the participant
Exceptions	1a The organizer does not select a participant. An error message should be shown with the information that the organizer must select at least one participant. 4a The organizer does not confirm. The delete process should be canceled. 5a The ReviewOger can not delete one or more participants in the list. An error message with the reason should be shown.

Name	**Manage slots**
ID	UC4
Target	The organizer can create, update and delete slots
Precondition	-
Postcondition	A slot is created, updated or deleted
Postcondition in case of exception	-
Actors	Organizer
Normal process	
Exceptions	

Name	**Create slots**
ID	UC4.1
Target	The organizer can create slots.
Precondition	-

Postcondition	A slot is created.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the create slot functionality 2. The ReviewOger shows the window to create a slot 3. The organizer enters the slot information 4. The ReviewOger verifies the information and adds the slot to the list
Exceptions	4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not add the slot to the list. An error message with the reason should be shown.

Name	**Update slots**
ID	UC4.2
Target	The organizer can update slots
Precondition	The slot to be updated exists
Postcondition	A slot is updated
Postcondition in case of exception	The slot will not be updated in case of an exception.
Actors	Organizer
Normal process	1. The organizer selects the slot to be updated 2. The ReviewOger shows the window to edit a slot 3. The organizer enters the slot information 4. The ReviewOger verifies the information and updates the slot in the list
Exceptions	1a The organizer does not select a slot. An error message should be shown with the information that the organizer must select a slot.

	4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not update the slot in the list. An error message with the reason should be shown.

Name	**Delete slots**
ID	UC4.3
Target	The organizer can delete slots
Precondition	The slot to be deleted exists.
Postcondition	A slot and its rooms are deleted
Postcondition in case of exception	The slot will not be deleted in case of an exception
Actors	Organizer
Normal process	1. The organizer selects the slot to be deleted 2. The organizer selects the delete functionality 3. The ReviewOger displays the slot information and asks the organizer if he really wants to delete this slot 4. The organizer confirms 5. The ReviewOger deletes the slot
Exceptions	1a The organizer does not select a slot. An error message should be shown with the information that the organizer must select a slot. 4a The organizer does not confirm. The delete process should be canceled. 5a The ReviewOger can not delete the slot in the list. An error message with the reason should be shown.

Name	**Manage rooms**
ID	UC5
Target	The organizer can create, update and

	delete rooms
Precondition	A slot in which the room is used must exist
Postcondition	A slot is created, updated or deleted
Postcondition in case of exception	-
Actors	Organizer
Normal process	
Exceptions	

Name	**Create rooms**
ID	UC5.1
Target	The organizer can create rooms.
Precondition	A slot in which the room is used must exist
Postcondition	A room is created.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the create room functionality 2. The ReviewOger shows the window to create a room 3. The organizer enters the room information 4. The ReviewOger verifies the information and add the room to the slot
Exceptions	4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not add the room to the slot. An error message with the reason should be shown.

Name	**Update rooms**
ID	UC5.2
Target	The organizer can update rooms

Precondition	The room to be updated exists
Postcondition	A room is updated
Postcondition in case of exception	The room will not be updated in case of an exception.
Actors	Organizer
Normal process	1. The organizer selects the room to be updated 2. The ReviewOger shows the window to edit a room 3. The organizer enters the room information 4. The ReviewOger verifies the information and update the room to the slot
Exceptions	1a The organizer does not select a room. An error message should be shown with the information that the organizer must select a room. 4a The organizer enters incorrect input. An error message should show information about what is wrong. 4b The ReviewOger can not update the room in the slot. An error message with the reason should be shown.

Name	**Delete rooms**
ID	UC5.3
Target	The organizer can delete rooms
Precondition	The room to be deleted exists.
Postcondition	A rooms is deleted
Postcondition in case of exception	The room will not be deleted in case of an exception
Actors	Organizer
Normal process	1. The organizer selects the room to be deleted 2. The organizer selects the delete functionality 3. The ReviewOger displays the slot information and asks the organizer if he really wants to delete this room

	4. The organizer confirms 5. The ReviewOger deletes the room from the slot
Exceptions	1a The organizer does not select a room. An error message should be shown with the information that the organizer must select a slot. 4a The organizer does not confirm. The delete process should be canceled. 5a The ReviewOger can not delete the room from the slot. An error message with the reason should be shown.

Name	**Create RevAger (lite) export**
ID	UC6
Target	The organizer can create an export for RevAger to share it with the participants so that they can use it for the review.
Precondition	The organizer has completed all ReviewOger steps.
Postcondition	The export is run successfully
Postcondition in case of exception	If there is an error via the export, the organizer should be informed.
Actors	Organizer
Normal process	1. The organizer selects the export functionality 2. The organizer is asked were he wants to save the file 3. ReviewOger creates the file in the needed format
Exceptions	2a The organizer closes the “How you want to save”-Dialog. Nothing should happen. 3a The export can not be performed. The organizer should get an error message.

Name	**Export group email to a mail client**
ID	UC7
Target	ReviewOger should prepare a Mail with all

	needed information for the participants to send it via a mail client.
Precondition	The organizer has completed all ReviewOger steps.
Postcondition	The Mail is prepared.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer chooses the send via Mail functionality 2. The ReviewOger prepares the Mail 3. The organizer can send the mail via his mail client
Exceptions	2a ReviewOger can not prepare the mail. The organizer should get an error message.

Name	**Export ReviewOger-File**
ID	UC8
Target	For saving the current status of his work or to use the current status again (e.g. as template), the organizer can export the current view.
Precondition	-
Postcondition	An export file with the current status is generated.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer chooses the export current state functionality 2. The organizer is asked were he wants to save the file 3. ReviewOger generates the export-file
Exceptions	2a The organizer closes the “How you want to save”-Dialog. The organizer should be informed that he will lose his current state. 3a The export can not be performed. The organizer should get an error message.

Name	**import ReviewOger-File**
ID	UC9
Target	The organizer should be able to import a current state to continue his work.
Precondition	A file with the current state in the needed format should exist.
Postcondition	The organizer can continue his work.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the import current state functionality 2. ReviewOger asks the organizer where he stored the current state 3. ReviewOger performs the import
Exceptions	2a The file is in the wrong format. The organizer should be informed with an error message. 3a ReviewOger can not perform the import. The organizer should be informed with an error message.

Name	**Export room plan**
ID	UC10
Target	The organizer should get a room plan to see which group is in which room and when.
Precondition	The organizer has completed all ReviewOger steps.
Postcondition	The export is run successfully.
Postcondition in case of exception	-
Actors	Organizer
Normal process	1. The organizer selects the functionality to generate the room plan 2. ReviewOger asks the organizer

	where he want to store the current state 3. ReviewOger generates the room plan
Exceptions	2a The organizer closes the “How you want to save”-Dialog. Nothing should happen. 3a ReviewOger can not perform the export. The organizer should be informed with an error message.

7. Appendix

The following appendix provides additional information and resources that are relevant to the main content of the document.

7.1 Glossary of Terms

This section explains terms used in the SRS that need clarification. Please refer to these definitions in case of uncertainty.

Term	**Technical Review**
Description	A Technical Review is a static white-box testing technique. The review is documented and its results are recorded. The roles involved include author, moderator, notary, reviewer.
Validity	-
Label	-
Uncertainty	-
Cross-reference	Notary, Author, Moderator, Reviewer

Term	**A/B - Review**
Description	Teams are writing specifications for two topics to which they are randomly assigned. The authors of topic A are only allowed to participate as authors for Topic A, and must assume other roles only for Topic B. The goal is to prevent people from plagiarizing ideas from other groups working on the same topic.
Validity	-
Label	-

Uncertainty	-
Cross-reference	-

Term	Notary
Description	The notary takes the expertise from the reviewer and writes it down. He is completely neutral. He can ask the moderator to slow down the discussion if he can't take notes that fast.
Validity	-
Label	-
Uncertainty	-
Cross-reference	Reviewer, Moderator

Term	Author
Description	The author is somebody from the development team. He knows the product deeply. The other review roles can ask him if there is something completely unclear. However, he should not answer every little question, because ambiguities are also points of criticism.
Validity	-
Label	-
Uncertainty	-
Cross-reference	Reviewer, Moderator, Notary

Term	Reviewer
Description	A reviewer is an expert in some part of the review. He provides valid arguments if he has other opinions.
Validity	-
Label	-
Uncertainty	-
Cross-reference	-

Term	Moderator
Description	Moderate the discussion and gather the consensus. He asks experts/reviewers explicitly for their agreement and keeps the discussion going.
Validity	-
Label	-
Uncertainty	-
Cross-reference	Reviewer

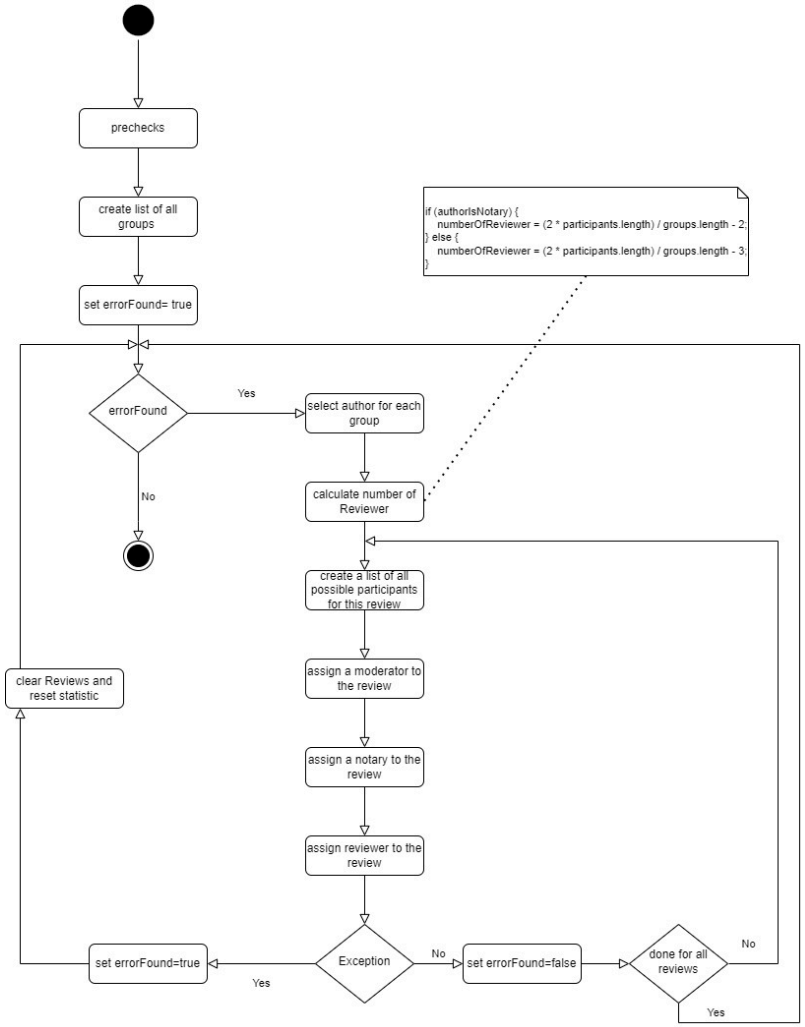
Term	Organizer
Description	The organizer is the person using ReviewOger (Reloaded) to plan a review. He can have a double role, if he is a participant as well.
Validity	-
Label	-
Uncertainty	-
Cross-reference	-

Term	Participant
Description	Participants are all people that take part in the review. All participants are assigned at least one role, required in the review process (Moderator, Author, Notary, Reviewer).
Validity	-
Label	-
Uncertainty	-
Cross-reference	Moderator, Author, Notary, Reviewer

Term	Slot
Description	A defined time interval in a schedule. A time slot can have 0-n rooms associated that are available in that time interval.
Validity	-

Label	-
Uncertainty	-
Cross-reference	-

Appendix A: Algorithm Flowchart



Appendix B: Quantity Structure

Entity	Constraint
Participants	200 Participants
Slots	20 Slots
Rooms	10 Rooms/Slot