Annotated Books Online

Documentation

Mathijs Baaijens, Iris Bekker, Renze Droog, Maarten van Duren,
Jeroen Hanselman, Bert Massop, Robin van der Ploeg, Tom Tervoort,
Gerben van Veenendaal, Tom Wennink

Contents

Part I

Functional

Chapter 1

Functional Designs

1.1 Definitions

1.2 Binding

1.2.1 1. Description

A Binding is one or more books bound together, wrapped in a cover.
A Binding is stored in a Library

- Research about books
- Library

1.2.2 2. Definition

2.1 Attributes

Some attributes are optional, as not always all information about a Binding is known.
A Binding has:

- Set of books - Mandatory.You cannot have a Binding without contents. This set can consist of one or more books.
- Library - Mandatory. The current owner of the Binding. This can either be a real library or a personal collection. It is mainly used for finding the real world location of the book.
- Shelfmark - Mandatory. The code with which the book is marked in its library. The code can consist of numbers and characters. If the book is in a personal collection,it should be possible to enter some kind of "not available" code. Still, under normal circumstances, the Library-Signature couple should form a unique identifier.
- Reader(s) - Optional. Someone who has owned, probably read, the book and may have written annotations in it. He or she may also have read other books, or wrote some of their own.
- Language(s) of the annotations - Optional. The language the annotations are written in. If multiple languages are used in annotating, it has to be possible to name them all.

2.1 Extra Information

The Books in the Binding are a subset of all the scans taken from the original manusript. Some pages in the manuscript may not belong to a certain book, loose pages. They should thus not be classified as part of that book, but rather as part of the manuscript, or Binding.

2.2 Loose Pages

Pages that are part of the Binding, but not part a specific Book. These are pages like the front of the book, the title page, and pages with notes on several Books. Pages with notes on a single Book which are disjoint with the Book, when it is located in another part of the Binding, are also considered loose pages.

1.3 Book

1.3.1 1. Description

A book is a collection of scans of pages.
A book is part of a Binding.

- Research about books
- Binding

1.3.2 2. Definition

2.1 Attributes

Some attributes are mandatory. One can always fill in these attributes. Others are optional because they might be unknown.
A Book has:

- Title - Mandatory. The name of the book. Even though the original title of the book may have been lost with time, the book is sure to have some sort of name.
- A set of authors - Optional. Sometimes the author of a book is unknown. Sometimes a book has several authors.
- Timeperiod - Mandatory. The exact publishing year of books is not always known. However, one is usually able to approximate the publishing year of a book. E.g. 1530 - 1580. If an exact year is known, it should be possible to enter that too.
- Place published - Optional. The place the work was published.
- Publisher/Printer - Optional. The firm or person that printed or published the book. Basicly, the book's origins.
- Edition - Optional. Whether it is it the first, second or nth print. Also known as the Version.
- Language - Mandatory. The language the book was written in.

1.4 Library

1.4.1 1. Description

A library contains a collection of books.
Library's will be used to store Bindings

- Research about books

1.4.2 2. Definition

2.1 Attributes

A Library contains:

- Name
- Address
- Link to website
- A set of Bindings

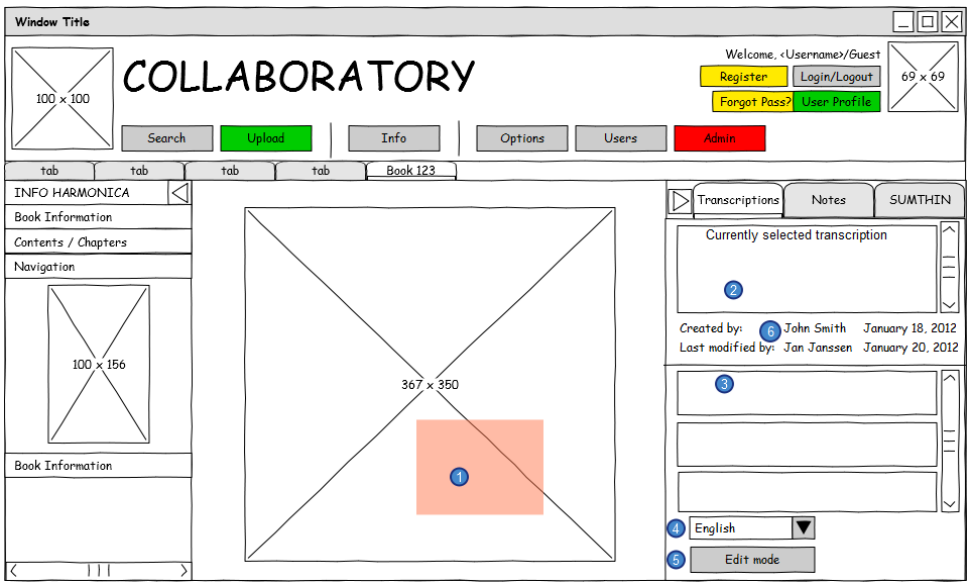
1.5 FD - Annotations

1.5.1 1. Concepts

This page describes the annotation functionality, including adding and editing annotations. When things are unclear or inconsistent, please contact someone from functional design.

1.5.2 2. Viewing annotations

Figure 2.1: Viewing annotations mockup



All functionalities of the usual viewing mode remain the same.

1. **Viewing polygons:** when a user mouses over a polygon that polygon lights up with a light blue colour. A user can select a polygon by clicking on it (edges included). (this is not really important so if it is easier to implement without the edges then that should be done). When dragging, normal panning occurs. When a polygon is selected,it lights up with a slightly darker blue colour. When a user selects a polygon, the transcription corresponding to that polygon becomes the selected transcription. When a user selects a transcription, the polygons corresponding to that transcription become selected. When multiple polygons overlap there should be some rule to determine which one is selected/moused over. When a polygon completely covers another one, the one with the smallest area will be picked. (This includes mousing over and selecting). When there exists an overlapping region, but no polygon completely overlaps the other one, the last one which was moused over will be picked when checking the overlapping region. (When none of them have been moused over, random will be fine).

2. **Selected transcription:** the transcription shown here is shown in its full length, unless this exceeds a certain number, in which case a scrollbar appears. When a user clicks on some other transcription from the transcription list (see 3), that transcription becomes the selected transcription. When a user clicks on a polygon and the polygon corresponds to a transcription, that transcription becomes the selected transcription. The user can manually adjust the size of the selected transcription box by dragging the line between the selected transcription and the others.

3. **Transcription list:** all transcriptions excluding the selected transcription are shown here. The transcriptions here have a fixed length (A character limit which we will need to determine through experiments). If the transcriptions do not all fit in the assigned space, a scrollbar appears. The transcriptions are ordered according to their order in edit mode.

4. **Language dropdown box:** in order to choose which language the user wants to see a choice needs to be made in the dropdown menu. There are two options: Original language: a transcription in the original language and English: the transcription, translated in English. (More languages may be added in the future).

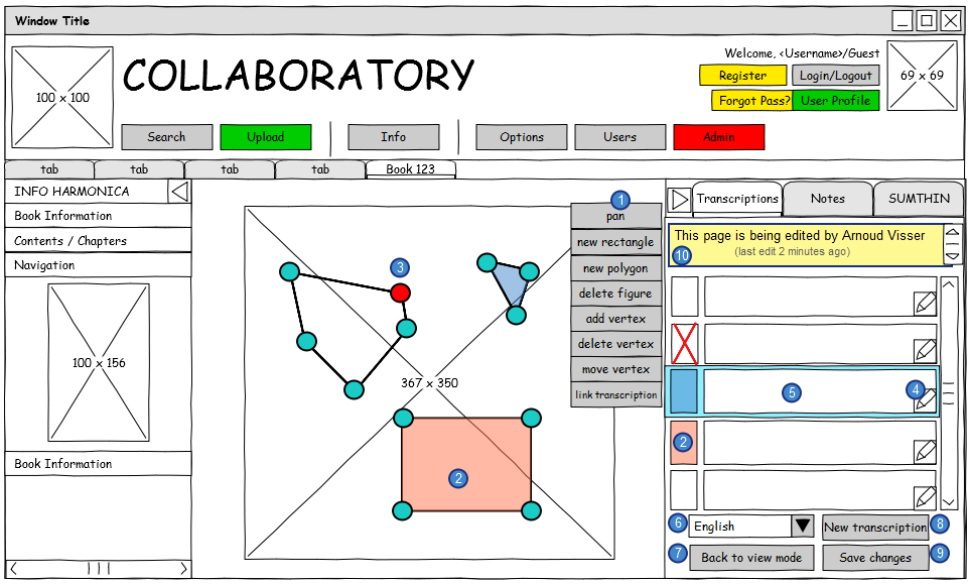
5. **'Edit mode' button:** when the user clicks this button, the user enters edit mode and the 'Transcriptions tab' changes to the one shown in figure 3.1. Who are allowed to edit is undecided as of yet. We will decide the structure of these permissions in a later FD.

6. **Creator and modifier information:** The first line shows the first name and last name of the person who originally created the transcription. Behind it is the date in the format "mmm dd, yyyy". The same applies for the second line, only this shows the user who last modified the transcription. If the transcription has never been modified yet, the second line could show exactly the same as the first line, or it could be hidden.

1.5.3 3. Editing annotations

During editing mode, when one goes to a different page (By selecting a bookmark, navigating through pages, etc.), if there are any unsaved changes, the User is shown a prompt asking him whether he wants to save these changes. If there are unlinked polygons and transcriptions, the system also tells the user that these will be removed if he chooses to save. The User can answer 'yes' (save the changes) 'no' (don't save the changes) or 'cancel' (don't save the changes and don't go to a different page). After making the decision to 'Save' or 'Discard', the system opens the selected page in editing mode.

Figure 3.1: Viewing annotations mockup



1. **Polygon tools:** in the mockup, the polygon tools are displayed as buttons. The idea is to make icons which a user can easily understand. (If this position is too difficult to implement, a different location can be chosen. We are still discussing what the best location for this toolbar is). A right click or clicking somewhere outside the viewer will generally result in selecting the pan tool, unless specified otherwise.

    - Pan tool: the default tool. While the pan tool is selected, the user can pan, zoom and rotate. Moving the mouse while the left mouse button is held results in panning. A single click results in selecting the underlying polygon. (including borders and vertices, unless this is difficult to implement).
    - New rectangle tool: with this tool, the user can make a polygon with the shape of a rectangle. When this tool is selected, the user can click (with the left mouse button) and drag in order to create a rectangle. The starting point and the end point of the dragging will determine the diagonal of the rectangle. While dragging, a light blue transparent rectangle and the vertices of the resulting polygon can be seen. The polygon is created when the user releases the left mouse button the pan tool is selected.
    - New polygon tool: with this tool, the user can create all kinds of polygons. When this tool is selected, the user can click (with the left mouse button) to create a vertex on the current location of the mouse cursor. (A preview of the vertex is shown underneath the mouse cursor) After the first vertex is created, a dotted line will be shown between the cursor (only if the cursor is on the scan) and the last created vertex. When the user right clicks or double clicks the polygon is completed by forming a line between the last and the first vertex. If a double click was used the last vertex is a new vertex at the position of the cursor. After a polygon is completed the pan tool is selected.
    - Delete figure tool: when this tool is selected, the mouse cursor is changed into a cross. If the user clicks on a polygon, the polygon is deleted. (When hovering over a polygon, the polygon lights up). Afterwards the pan tool is selected.
    - Add vertex tool: when this tool is selected, the mouse cursor is changed into a plus. If the user clicks on an edge, a new vertex will be created in the middle of the edge. (When hovering in the neighbourhood (a few pixels away) of an edge, the edge becomes slightly thicker and the new vertex can be seen). Afterwards the pan tool is selected.
    - Delete vertex tool: when this tool is selected, the mouse cursor is changed into a cross. If the user clicks on a vertex, this vertex is and the edges connected to this vertex are

deleted. A new edge is created between the vertices which have only one edge. (When there are only two vertices left, the whole polygon is deleted). Afterwards the pan tool is selected.

- Move vertex tool: when the 'move vertex' tool is selected the user can select a vertex and drag it. When a vertex is dragged the lines attached to the vertex are dragged along.
- Link tool: when the link tool is selected the user can click on a polygon to create a link between the polygon and the selected transcription. If the polygon was already linked to another transcription the link is replaced. If the transcription was already linked to a polygon, that link disappears. Afterwards the pan tool is selected.

2. **Linked transcriptions:** when a transcription is linked to a polygon, the colour of the interior of the polygon and the coloured field next to the transcription coincide. Because we do not expect a lot of transcriptions on one page , we can limit the amount of transcriptions to 128 or higher in order to generate easily differentiable colours. When a transcription is unlinked, a cross is shown in the coloured box. A polygon which is unlinked does not have a coloured interior.

3. **Polygons:** the lines between the polygons will be black and the vertices consist of a black border and a light blue area (e.g. r: 197, g: 251, b: 238). When a vertex is selected with the 'move vertex' tool (see 1), the vertex becomes red.

4. **Transcription list:** all transcriptions are shown here. The transcriptions have a fixed length (A character limit which we will need to determine through experiments). If the transcriptions do not all fit in the assigned space a scrollbar appears. The selected transcription is lit up. When a user clicks on some other transcription from the transcription list (see 5), that transcription becomes the selected transcription. When a user selects a polygon and the polygon corresponds to a transcription, that transcription becomes the selected transcription. (Which transcription is shown depends on the current language. See 6. Language dropdown box). The textfield supports unicode (All kinds of languages can appear in the "Original language" mode). A user can manually order the transcriptions in the list by dragging.

5. **Edit transcription:** when the user clicks on the pencil icon in the right corner (or double clicks on the transcription), a popup screen appears (see figure 3.2). In here he can change the contents of the transcriptions. (When mutiple languages are supported, a dropdown menu will appear above the English textfield, English will be the default option and other languages can be chosen).

6. **Language dropdown box:** in order to choose which language the user wants to edit, a choice needs to be made in the dropdown menu. There are two options: Original language: A transcription in the original language and English: The transcription, translated in English. (More languages may be added in the future).

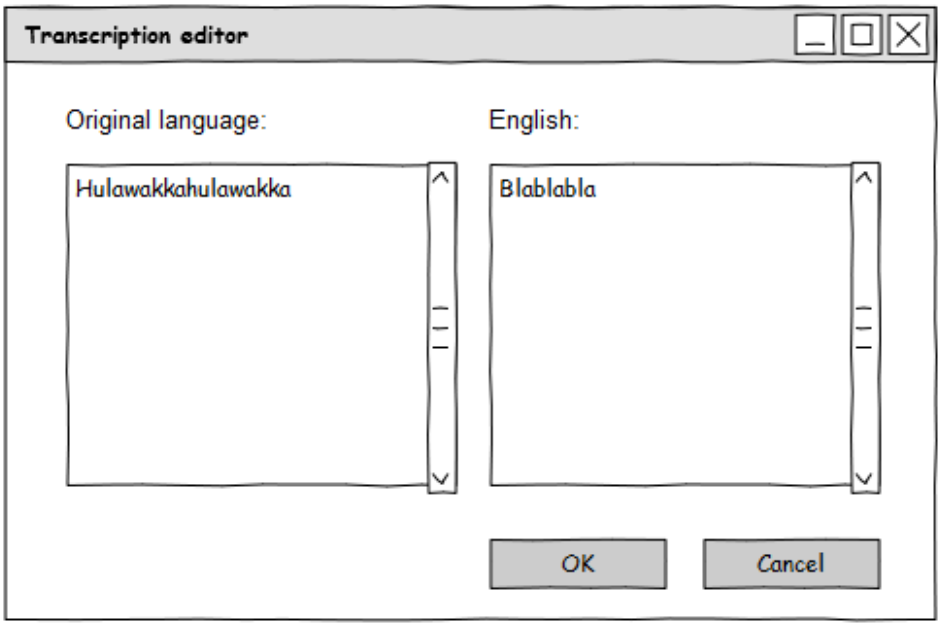
7. **'Back to view mode' button:** if the user clicks this button the editing mode ends, the 'Transcriptions' tab changes to the one shown in figure 2.1 and he is returned to the view mode. If there are any unsaved changes the user is shown a prompt asking him whether he wants to save these changes. The user can answer 'yes' (save the changes) 'no' (don't save the changes) or 'cancel' (don't save the changes and don't end the editing mode).

8. **'New transcription' button:** when a user clicks this button a new transcription is created and the transcription editor opens so that the user can enter immediately start editing it.

9. **'Save changes' button:** if the user clicks this button the changes are saved. The button is greyed out when no changes have been made since the last save or since the user started

editing. When there exist polygons and transcriptions which are unlinked. The user is shown a prompt asking whether it is ok that these will be removed. If the user selects yes, the unlinked items are removed and the result is saved.

10. **Other users are editing message:** When a user is editing the same page as another user, a message will be shown notifying him of all the users who are editing the page and how long they have been doing that. (e.g. This page is being edited by <username> (last edit 2 minutes ago).)



1.6 FD - Bookcase and bookmarks

1.6.1 1. Concepts

This page describes the functionalities that allow users to organize their books: the personal bookcase and bookmarks.

Bookcase: the personal bookcase is a user-bound, personalized collection of titles available on the website. Users can put books they find interesting or might need to read again later to the bookcase, and sort them by placing the books on different shelves for specific purposes.

The bookcase contains 3 special shelves by default. These can not be removed, merged, contain new shelves, be moved into other shelves or be renamed. They can be cleared, and books can be removed or moved out of these shelves. These are:

1. 'Bookmarked': books that are bookmarked (in the viewer) automatically are added to this shelf.
2. 'Uploaded': books uploaded by the user are automatically added to this shelf.
3. 'Unshelved': books that were on a removed shelf which contained books are moved to this (or 'pile').

Bookmarks: bookmarks are what bookmarks have always been; a simple marker for pages that the reader is interested in, or to remember where the user was when he last stopped reading. They are

stored per user, per book, and can be used to easily skip to the page they are set to when reading the book. In essence, it's nothing more than a user-specified 'jump to page number X'. Because one binding can contain multiple books, our implementation also lists the title and author of the book found at the page number the bookmark is placed at.

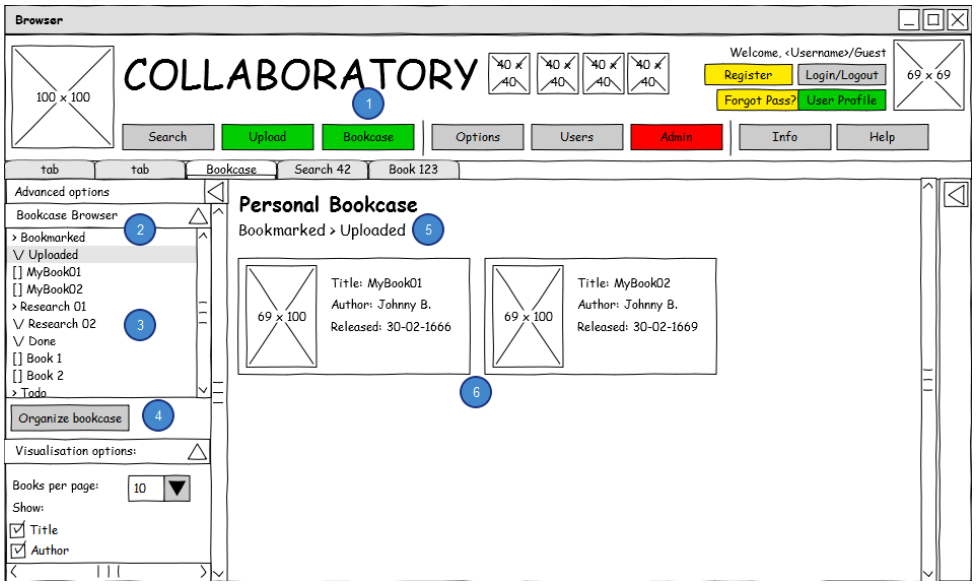
Note: bookmarks are NOT saved per 'instance' of a book as found in the personal bookcase - adding a book to the personal bookcase twice in different locations will not have them have different lists of bookmarks, as both are simply a link to the same book.

1.6.2 2. Designs

2.1. Bookcase

The personal bookcase window:

Figure 1:



List of screen objects and points of interest for Figure 1:

1. To instantly go to his/her personal bookcase, the user can press the 'Bookcase' button at the top.

   (a) This button only shows up for logged in users, as guests do not have a personal bookcase.

2. On the left-hand side is the Advanced options sidebar. On the personal bookcase page, this contains a 'Bookcase Browser' section.

   (a) It also has the same visualisation options as the search results, to change the visualisation of the books on the selected shelf (shown information, books per page).

3. The bookcase browser itself is a tree view of the bookcase, showing the (possibly nested) shelves and their contents in a collapsible tree format.

   (a) doubleclicking on a book here opens it in a new viewer tab.

4. Below the tree is a button to open the 'Bookcase organizer' popup window. See Figure 2.

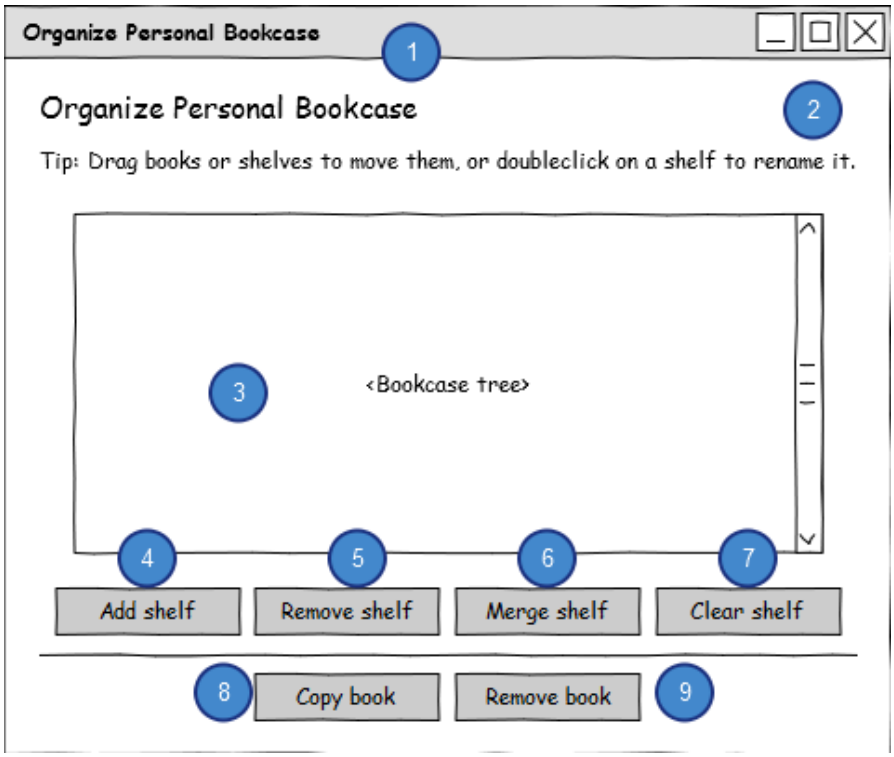
5. In the middle is the name and contents of the currently selected shelf in the personal bookcase.

(a) The name and location of the shelf is shown as breadcrumbs, to more easily identify where in the bookcase the current shelf is.

6. Clicking on a book in the overview opens it in a new viewer tab, similar to how it works on the search results page.

The organize bookcase popup window:

Figure 2:



List of screen objects and points of interest for Figure 2:

1. The window pops up over the bookcase itself, and can be closed using the 'X' button at the top.
2. A tip explaining the move and rename functionality is shown above the bookcase tree visualisation.
3. The bookcase is shown in a tree, as in Figure 1 (see bullet 3).
4. "Add shelf" button opens the "Add shelf" dialog - see Figure 3.
5. "Remove shelf" button opens the "Remove shelf" dialog - see Figure 4.
   (a) *Can only be used if a **shelf** is selected.*
6. "Merge shelf" button opens the "Merge shelf" dialog - see Figure 6.
   (a) *Can only be used if a* **shelf** *is selected.*
7. "Clear shelf" button opens the "Clear shelf" dialog - see Figure 7.
   (a) *Can only be used if a* **shelf with books** *is selected.*
8. "Copy book" copies the currently selected book and places the copy on the same shelf as the original.

(a) *Can only be used if a **book** is selected.*

9. "Remove book" button opens the "Remove book" dialog - see Figure 5.

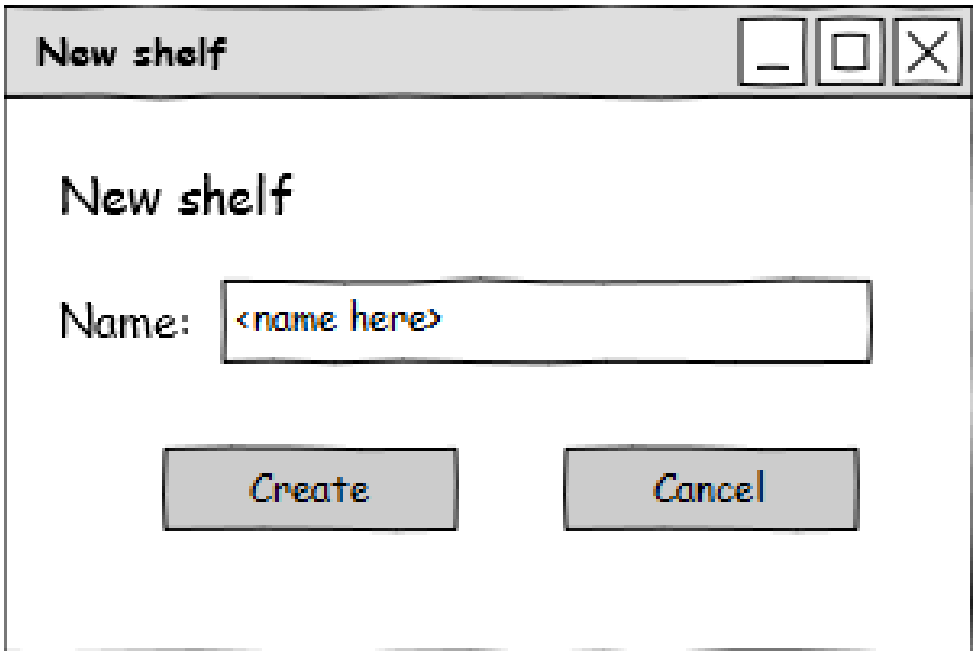
(a) *Can only be used if a **book** is selected.*

NOTE: as mentioned above (see definition of Bookcase, in Section 1 - Concepts), special rules apply to the 'Bookmarked', 'Uploaded' and 'Unshelved' shelves. Extra checks need to be performed to make sure these rules are not broken.

2.2. Organize Dialogs

The dialogs opened by pressing the buttons in the Organize personal bookcase window:

Figure 3 - New shelf:

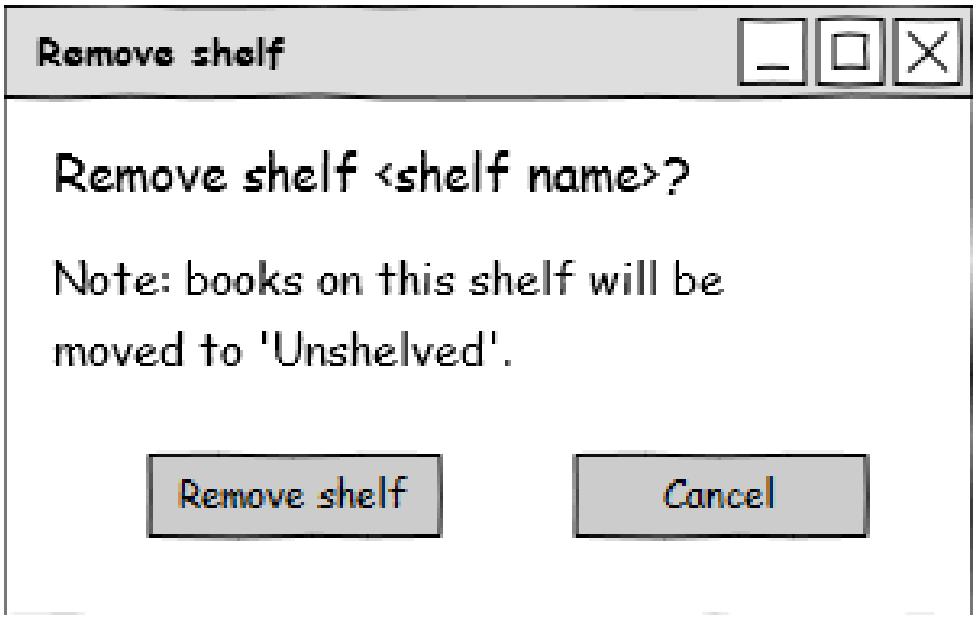


List of screen objects and points of interest for Figure 3:

1. A simple textfield with new shelf name, a create button to finish the action, and a cancel button to cancel it.

2. Pressing the 'X' in the top right also closes the dialog as if 'cancel' was pressed.

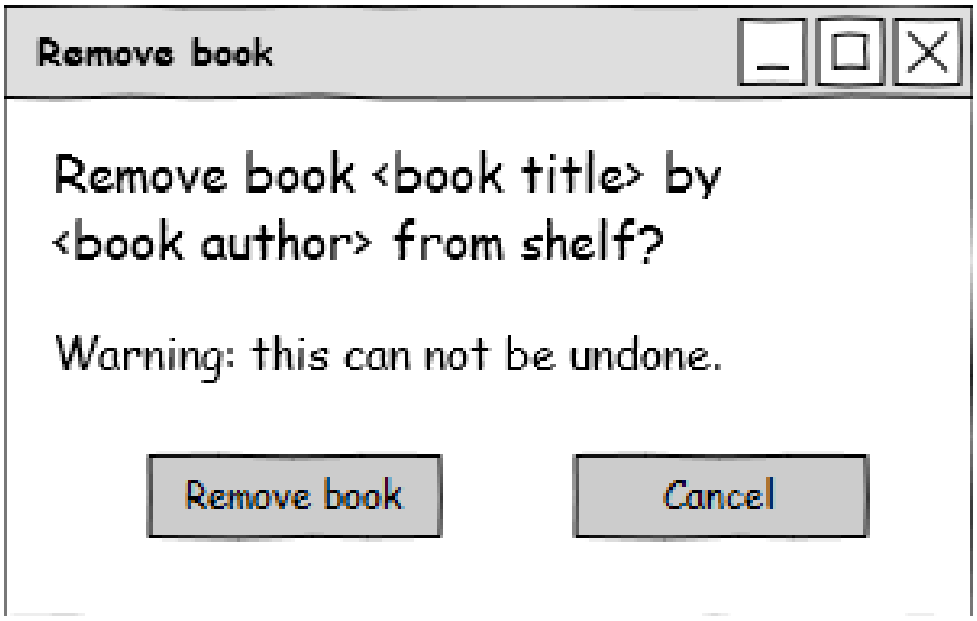
Figure 4 - Remove shelf:



List of screen objects and points of interest for Figure 4:

1. A warning stating that the books on this shelf (if any) will be moved to 'Unshelved', a remove button to finish the action, and a cancel button to cancel it.
2. Pressing the 'X' in the top right also closes the dialog as if 'cancel' was pressed.

Figure 5 - Remove book:

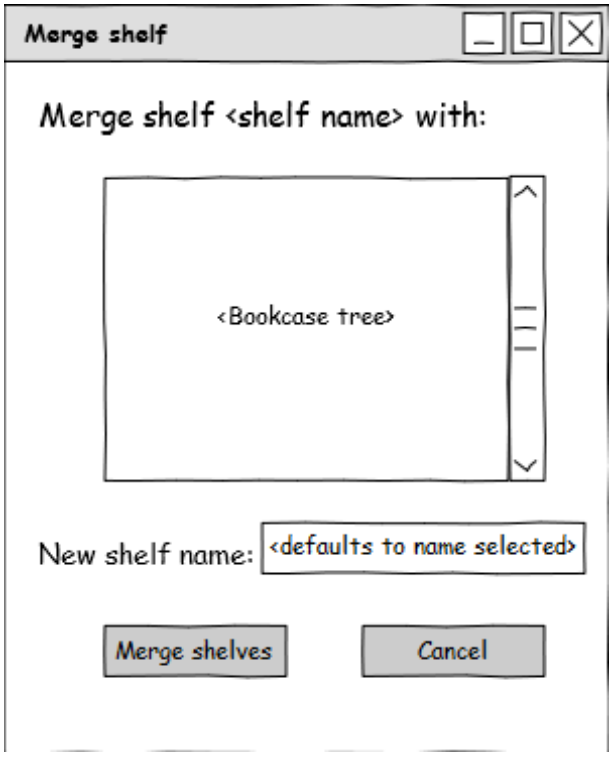


List of screen objects and points of interest for Figure 5:

1. A warning stating that the selected book (title and author shown) will be removed from the shelf it is on (which can not be undone, other than re-adding the book to the shelf), a remove button to finish the action, and a cancel button to cancel it.

2. Pressing the 'X' in the top right also closes the dialog as if 'cancel' was pressed.

Figure 6 - Merge shelf:



List of screen objects and points of interest for Figure 6:

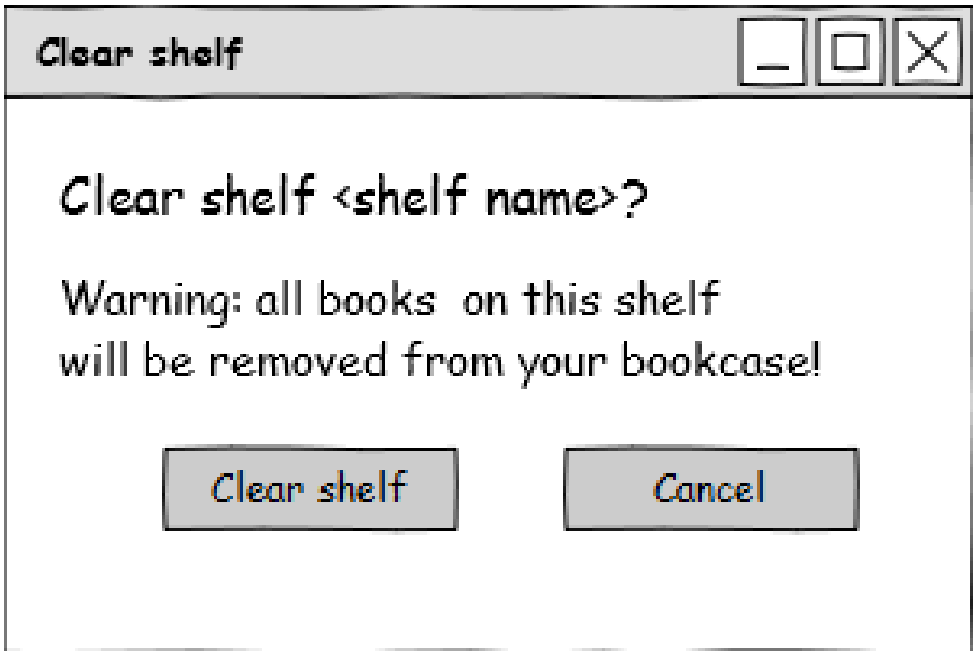
1. A 'tree' view of the bookcase (similar to that found in the sidebar for the Bookcase screen, see Figure 1, bullet 3), a textfield for the name of the 'new' shelf, a 'Merge shelves' button to finish the action, and a cancel button to cancel it.

2. The 'new' shelf is the result of combining two others, and the default name is the name of the shelf last selected to merge with (the one selected in the tree view in this dialog).

   (a) If the merged shelves both contained a copy of the same book, only one copy is saved on the new shelf.

3. Pressing the 'X' in the top right also closes the dialog as if 'cancel' was pressed.

Figure 7 - Remove shelf:



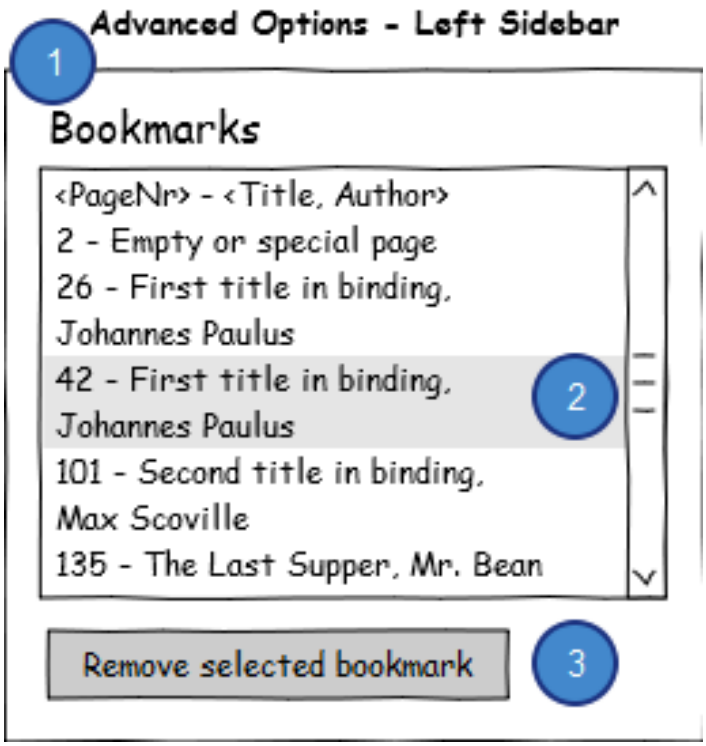
List of screen objects and points of interest for Figure 7:

1. A warning stating that the books on this shelf will be removed from the bookcase, a remove button to finish the action, and a cancel button to cancel it.

2. Pressing the 'X' in the top right also closes the dialog as if 'cancel' was pressed.

2.3. Bookmarks and Bookcase in Viewer

Figure 8 - Bookmarks 'panel' for left-hand sidebar in Viewer:



List of screen objects and points of interest for Figure 8:

1. This section, found in the left-hand sidebar of the viewer, is a bookmarks browser.

2. Double-clicking on a bookmark automatically makes the viewer jump to the page the bookmark is at.

   (a) Single-click merely selects, to allow for the remove functionality below without having to jump to the page first. After all, if the user wants to remove the bookmark, it probably is not useful enough to jump to.

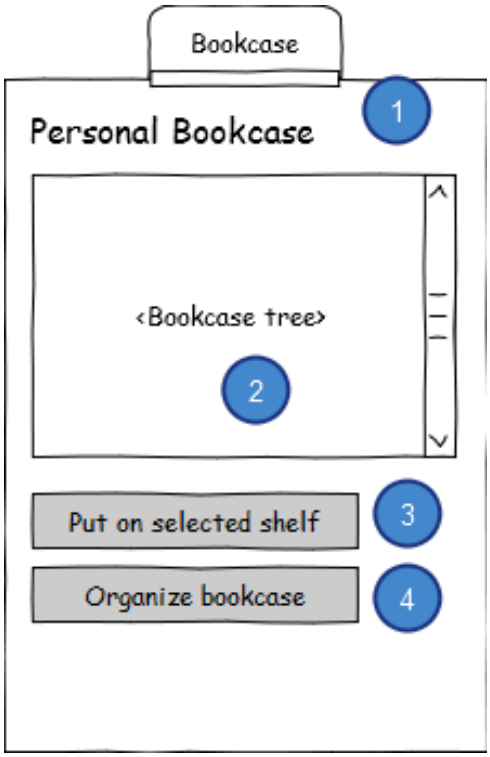
   (b) The bookmarks are formatted as follows: <**Page number**> **-** <**Title**>**,** <**Author**> or, if the page is in a binding but not part of a book (beginning or end of a book, pages for notes, etc.) as <**Page number**> **- Empty or special page.**

3. The 'Remove selected bookmark' removes the bookmark selected in the list above (bullet 2).

   (a) This button is disabled if no bookmark is currently selected.

NOTE: the viewer should also have an 'add bookmark' button at the top of the viewer. It is not included here, as having the button on two locations would clog up the screen unnecessarily, and the buttons at the top of the viewer are always in view.

Figure 9 - Bookcase tab for right-hand sidebar in Viewer:



List of screen objects and points of interest for Figure 9:

1. This tab, to be found in the Viewer in the right-hand sidebar, is called 'Bookcase' and allows the user to place a book in his personal bookcase.
2. The tab contains a bookcase tree view, to select a shelf to place the book on.
3. The 'Put book on selected shelf' button is used to place the current book on the shelf selected in the tree view above it.
   (a) Button should be disabled if no shelf is selected.
4. The 'Organize bookcase' button opens the 'Organize personal bookcase' popup, as seen in Figure 2.

1.7 FD - Book Search, Sort, Select

1.7.1 1. Concepts

This page describes the basic book search functionality, including the search options, sorting and selecting functionality.

1.7.2 2. Book Search

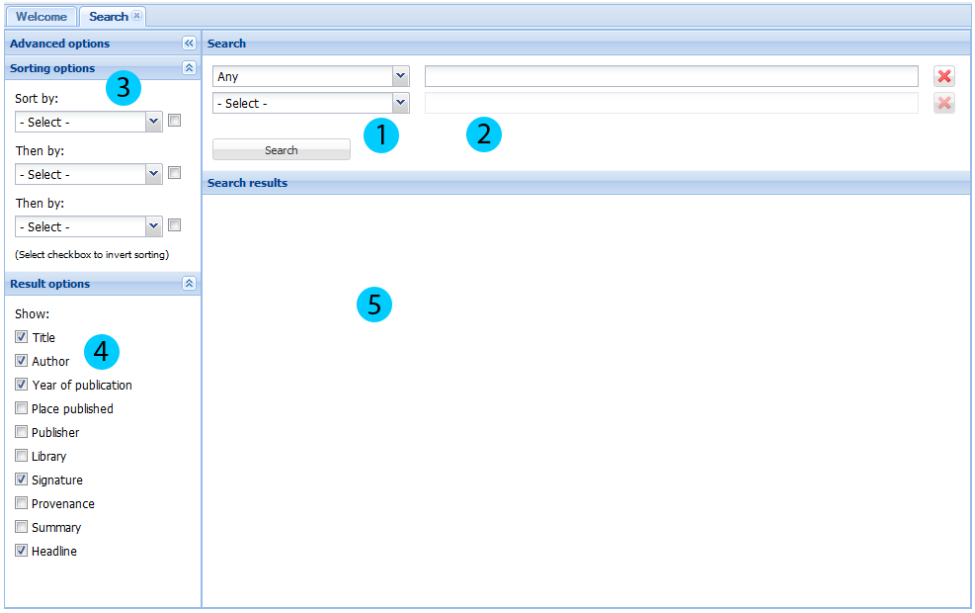
The following describes the functionality of the Book Search screen, and how it reacts to user input.

2.1. Select search terms

The search functionality allows users to specify what books they want to see listed.

For requirements, see Research about select, search and sort.

Figure 1 (screenshot taken from in-development testing version):



List of screen objects for Figure 1:

1. The user can select what attribute or field to search for (’search *type*’). Default for the first row is ’Any’, all rows after have ’-Select-’ as default. Other options are the attributes defined for Book and Binding.

2. If a type other than ’-Select-’ has been selected in the dropdown box, the field on it’s right is enabled and information to search for (’search *terms*’) can be entered. A type and terms combined make a search query.

   - For text fields, the following features will be implemented:

   (a) i. Exact matches are not required: partial matches and small typing errors should give valid results.
   ii. Spaces work as logical "AND".
   iii. A minus sign (-) works as "NOT" in front of a word, so that a search for ’-banana’ will exclude results with ’banana’ in them. Note that this only counts if preceded by either nothing or a space: ’jan-klaessen’ should not exclude ’klaessen’ as results.
   iv. Quotation marks (" ") around a term (or part of a term) will make it an exact search: partial matches and typing errors are no longer excused. Spaces within the quotation marks are also part of the exact search term.

   - Depending on the search type selected in the dropdown box on the left , the information field(s), labels, and possibly placeholder text (for ’empty’ fields) on the right may change.
     – The labels and text fields should fit with what needs to be entered (’Between - and’ for years, etc.). This only needs to change for special cases (publishing year in particular), and in most cases (such as title or author) will simply be a text field, as

described above.

- As long as the dropdown box on the left is not set to something other than '-Select-', the field on the right is disabled and nothing can be entered.
- Selecting a type other than '-Select-' results in a new line being added at the bottom, so more search options can be added. The button on the right of each line removes the line, leaving the other lines intact.

3. The sorting options are in the Advanced options sidebar, allowing up to 3 different attributes to sort by.

   - A checkbox behind the options allows the sorting to be reversed: from Z to A instead of A to Z, for instance.
   - Changes here are automatic: pressing the search button again is not necessary after making changes.
   - For more info, see **3.1 Book Search Results.**

4. The results options are also in the Advanced options sidebar, allowing the user to select what information is shown in the results listing.

   - Changes here are also automatic: pressing the search button again is not necessary after making changes.
   - MISSING ON SCREENSHOT: dropdown box to change the amount of 'results per page'.

5. After pressing the 'Search' button, the search results will be shown, on multiple pages if the list is too long.

   - Pressing 'Search' again is only necessary after changing the search terms / keywords, not the sorting and result options in the sidebar.

1.7.3 3. Book Search Results (Sorting and Selecting)

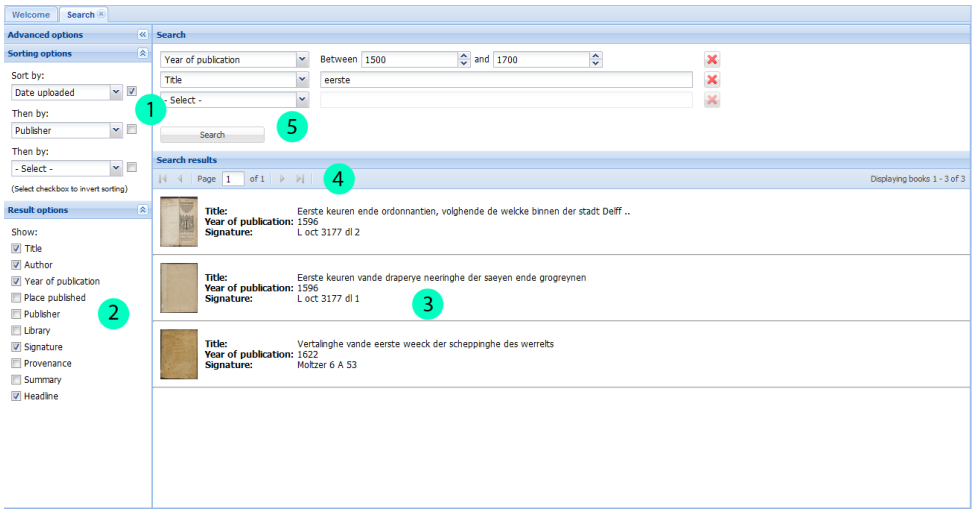
The following describes the functionality of the Book Search Results screen, and how it reacts to user input.

3.1. Sorting

The sorting functionality allows users to sort the list of search results, so that users can more easily find the book they are looking for.

For requirements, see Research about select, search and sort.

Figure 2 (screenshot taken from in-development testing version):



List of screen objects for Figure 2:

1. The dropdown boxes for sorting contain the same fields as what can be searched for (see **2.1. Select search terms**), which is to say:
   (a) The attributes for Book and Binding,
   (b) "Date last modified" and "Date uploaded".
   - The order of the sorting can be reversed (A-Z to Z-A, 0 - 999999 to 999999 - 0) by checking the checkboxes, in case a Title starts with a Z for instance.
   - Changing the sorting settings automatically changes the sorting of the list of results.
2. The Result options can be modified to change what information is shown in the list of results through checkboxes, and a dropdown box to select how many results are listed per page.
3. The search results list all 'found' books that match the search query.
   (a) Each 'result' can be clicked on (anywhere in the 'block' around the book information) to select it (see **3.2 Selecting**).
   (b) The results show the information selected under Result options (by default these are **Title**, **Author**, **Year of publication**, **Signature** and **Headline)**.
   - If available, an image for each book is shown. This can be the front cover, or first page.
4. Search results may be spread over multiple pages.
5. At the top of the page the search query can be changed, in case an error was made. After making changes, the user must press 'Search' to commit the changes.
   (a) The Sorting and Result options will stay the same.

3.2. Selecting

1. Person finds book
2. Person moves mouse over box with book details
3. Person clicks
4. Book is opened in Viewer

1.8 UC - Book Search

Use case: Book Search
ID: R001
Brief description:The user enters search queries for searching a book.
Primary actors: User
Secondary actors: None.
Preconditions 1. None.
Main flow: 1. The use case starts when the User selects the search function. 2. While the User is entering search criteria: (a) If the type of a row is not "-Select-": i. The system enables the field(s) where terms are entered. (b) If all the fields of a search contain information: i. The system enables the next search row. (c) If the User presses the "+ Add more search terms" button: i. The system adds another row. 3. The User presses "Search" or the Enter key. 4. The system performs a search with the entered search criteria. 5. If the system finds search results: (a) The system takes the User to a search results screen. 6. Else: (a) The system shows a message stating "No search results found! Please change your search terms and try again." (b) The use case resumes at 2.
Postconditions: 1. The User has been taken to a search results screen showing the books found by the search.
Alternative flow:None.

1.9 UC - Book Search Results (Sort)

Use case: Book Search Results (Sort)
ID: R002
Brief description:The user sorts the search results.
Primary actors: User
Secondary actors: None.
Preconditions 1. The system has performed a search and found results.
Main flow: 1. The use case starts when the system has found search results. 2. The system shows the User the search results. 3. While the User is entering sorting options: (a) If a sorting option is set to something other than "-Select-": i. The system enables the next sorting option. 4. The User presses the "Sort" button or the Enter key. 5. The system sorts and displays the search results according to the sort options.
Postconditions: 1. The system has displayed the resorted search results.
Alternative flow:

1.10 FD - Exporting

1.10.1 1. Concepts

This design describes the way users can export scans and Bindings to PDF.

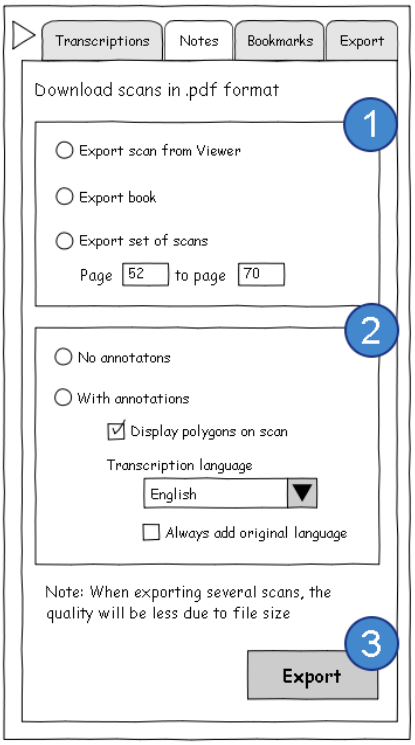
1.10.2 2. Exporting scans to PDF

The following describes the way export is accessed from the Viewer.

2.1. The Export Tab

The 'Export' tab in the Viewer

Figure 1:



1. 'Export Selection' box*. List of radio buttons for selecting which scans to export. The user can choose from three options

   (a) Export scan from Viewer - This will export only the scan currently being viewed. This is the default setting.

   (b) Export book - This will export the whole Binding including all books and loose page scans as a single .pdf.

   (c) Export set of scans - Export a part of the Binding. The user gives a start and an end page. All scans from the start up to and with the end page are exported as one .pdf.

2. 'Transcription Selection' box*. With this list of radio buttons the user can indicate if he wants to export with or without the transcriptions.

   (a) No transcriptions - No transcriptions are added to the export; it will be the bare book, a copy of the original.

   (b) With transcriptions - The transcriptions made by the users are added to the export

      i. Display polygons on scan- If the user selects to export with the annotations, she can also choose to display the scans with or without their annotation polygons.

      ii. Transcription language - The user can select the main language for the transcriptions with the drop down box.

      iii. 'Always add original language' - When an other language then original is selected in the drop down box, this checkbox becomes available. When checked, aside from the selected language, all transcriptions in the original language are also added.

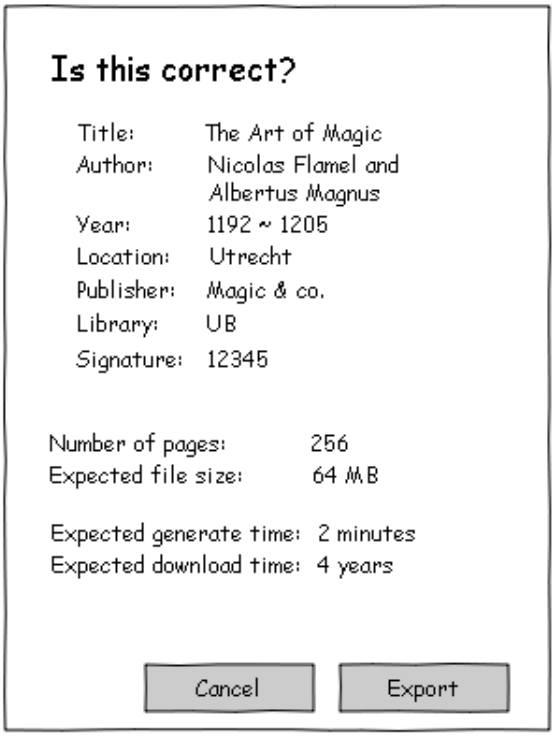
3. 'Export' button - Activates the export function. If a whole Binding is exported, there is first a acceptance popup, as a Binding is a rather big file. See Figure 2.

* 'Export Selection' and 'Translation Selection' are called boxes, but they need not be in an individual box per se. "Box" is simply shorter than "radio button list"

2.2. The Export Popup

If a whole Binding or several pages are exported a popup is given to ensure the user wants this, potentially large, file. With a single page, which is relatively small, no popup is required.

Figure 2:



The user is prompted to agree with the data, to confirm that the correct Binding is being exported. Also the expected number of pages, file size, generation time and download time. The generation time estimation makes it clear to the user that it may take some time before the download starts. Note that the number of pages is that of the PDF, not the number of scans exported.

If there are multiple authors, they should be displayed one after another, the line broken when needed. If possible, line breaks in the middle of a name should be avoided, like in Figure 2. It is not expected that a book will have many, very long-named authors.

If the export is with transcriptions, this is noted below the signature attribute, as shown in Figure 3.

Figure 3:

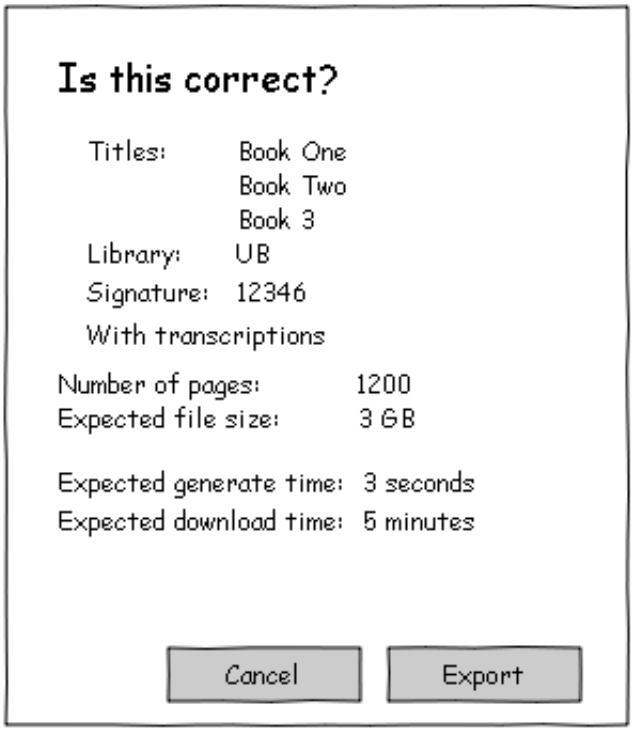


Figure 3 shows the popup given when a binding with multiple books is exported. All included titles and the library and signature are displayed, as well as wether there are transcriptions included. As every book has its own author, publication year, location and publisher, these data are not displayed in the popup.

Note that the number of pages includes the pages spend on the transcriptions.

Figure 4:

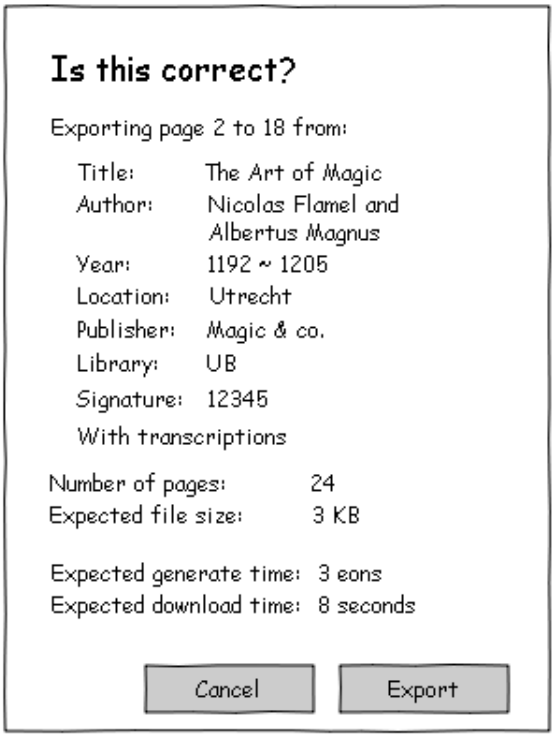


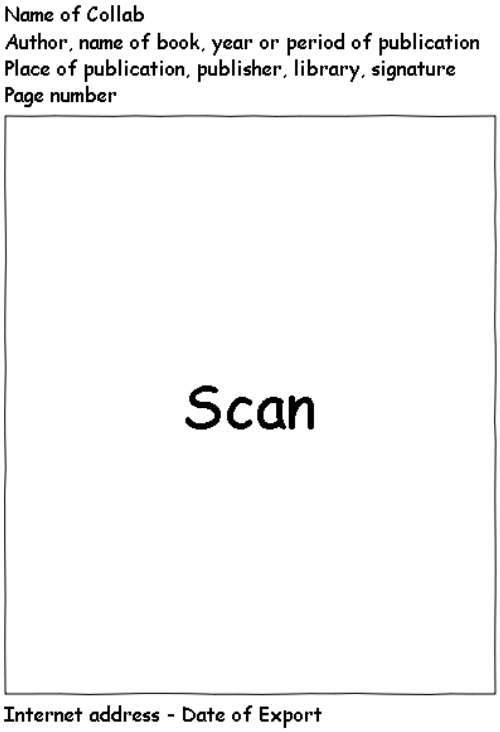
Figure 4 shows the displayed data when exporting several pages. This popup is activated since people could export large quantities of data with the pages feature.

The popup is the same as with exporting a Binding, the only difference being the top now saying "Exporting page x to y from:". Note that if the Binding contains several books, only the titles, library and signature are displayed.

2.3. Export Scan

The idea of exporting a single san, is to make a quick copy of the page currently being viewed. The page layout should be compact, so printing is fast and the user only has to look at a single page. Of course, this is without the transcriptions. See Exporting with Transcriptions for that.

Figure 5:



At the top of the page the name of the collaboratory should be visible.

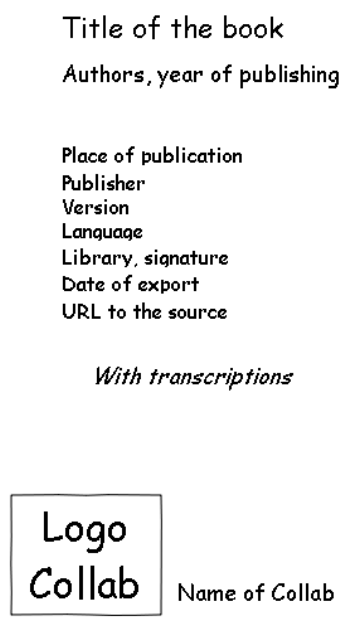
Underneath that, all the data for referencing the book. The text is concatenated to save space on the page. If something is unknown, like place of publication, it is left empty. Beneath that will be the page number referencing to the online copy.

Below the scan will be the internet address of the source and the date when the scan was exported.

2.4. Export several Scans

When exporting more than one scan, it would make the set of pages more coherent when there is a front page followed by scans which are all displayed in the same format.

Figure 6:



The front page contains a list of the information needed for referencing the unique source. If the Binding contains several books, all the titles are listen, each on a new line, together with the library and signature; as in Figure 7. The other data for books is given in the index, Figure 7. The front page also shows the name and loge of the Collab. If the export contains transcriptions, it should say so on the cover. If only a set of pages has been exported, this should also be noted on the cover.

Figure 7:

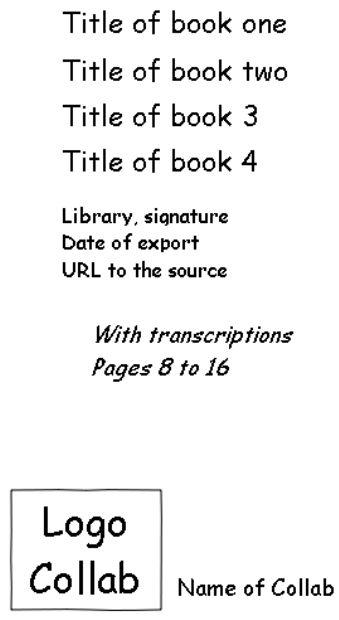
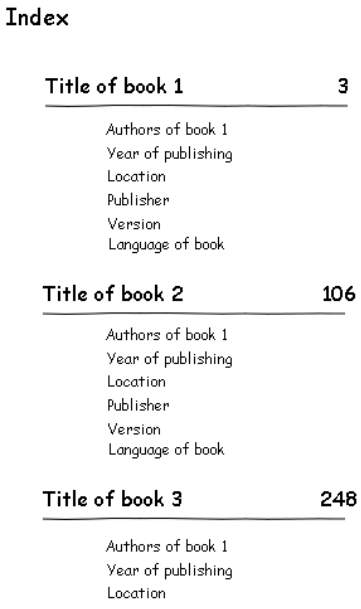


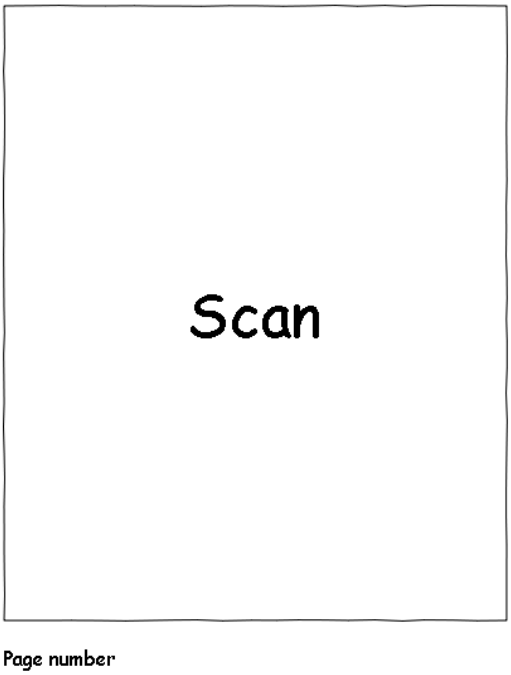
Figure 8:



An index is only added if (part of) a Binding with more than one book is exported. The numbers do not match the page numbers as used in the system, but the numbers in the PDF where to find the first page of that book.

Even when just exporting a set of pages of the Binding, a list of everything inside

Figure 9:

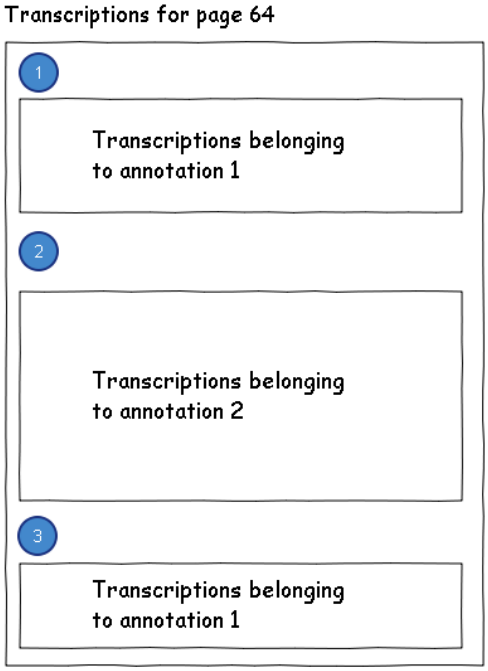


The layout for all the scans in the export. Only the scan and the page number, referencing to the page number used in the system.

2.5. Exporting with Transcriptions

If the export is done with transcriptions, they are added on the page(s) after the scan that they belong to. This is done the some for single scan and multiple scan, only with multiple scan a line saying transcriptions are included is added to the cover.

Figure 10:



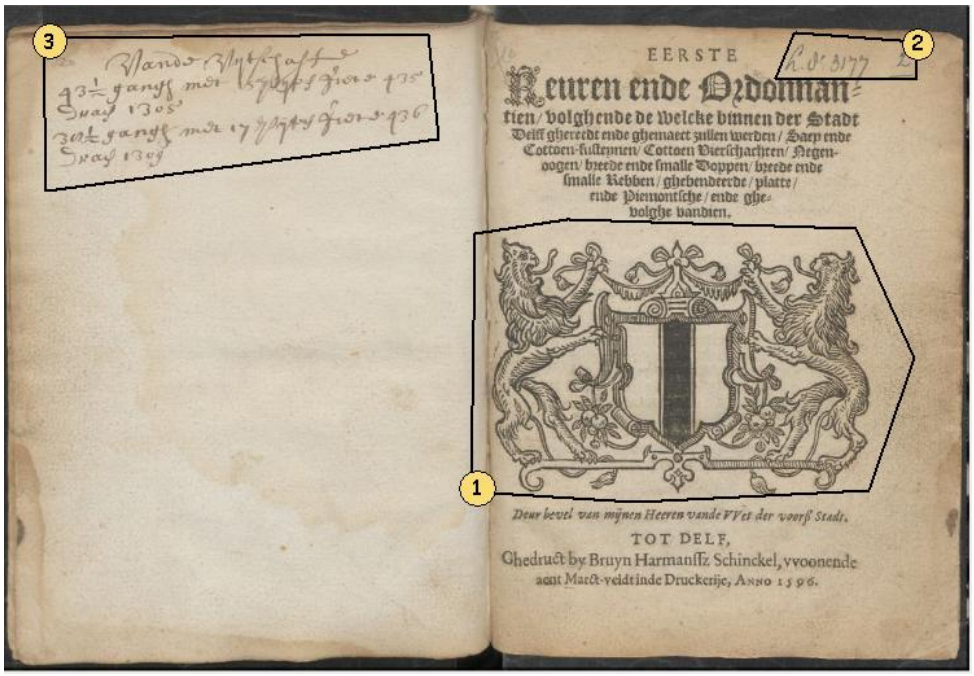
At the top of the page it says to which page the transcriptions link. If more then one page is needed for displaying the transcriptions, every next page should have "Transcriptions for page x (continue)" as headline.

The language of a transcriptions depends on the entered preference by the user. For a certain annotation, the transcription with the chosen language will be exported. If the annotation has no transcription in this language, it will add the transcription in the original language. When there are more languages to write transcriptions in, at the moment only original and English are supported, it can happen there is no transcription in the selected or the original language, when one is written in a third language. In these cases, no transcription is added to though the transcription list, but the annotation is marked on the page if the user selected for polygons.

If the user selected a preference language other then original and checked the 'always add original language' underneath the dropbox in the export tab, the transcription will always be added in the original language, if available, followed by the transcription in the selected language, again if available.

The transcriptions are sorted by annotation they are linked to. If the user has selected to display the polygons, they are shown on the scan with a number linking them to their corresponding transcriptions.

Figure 11:



EERSTE

Keuren ende Ordonnan-

tien/ volghende de welcke binnen der Stadt

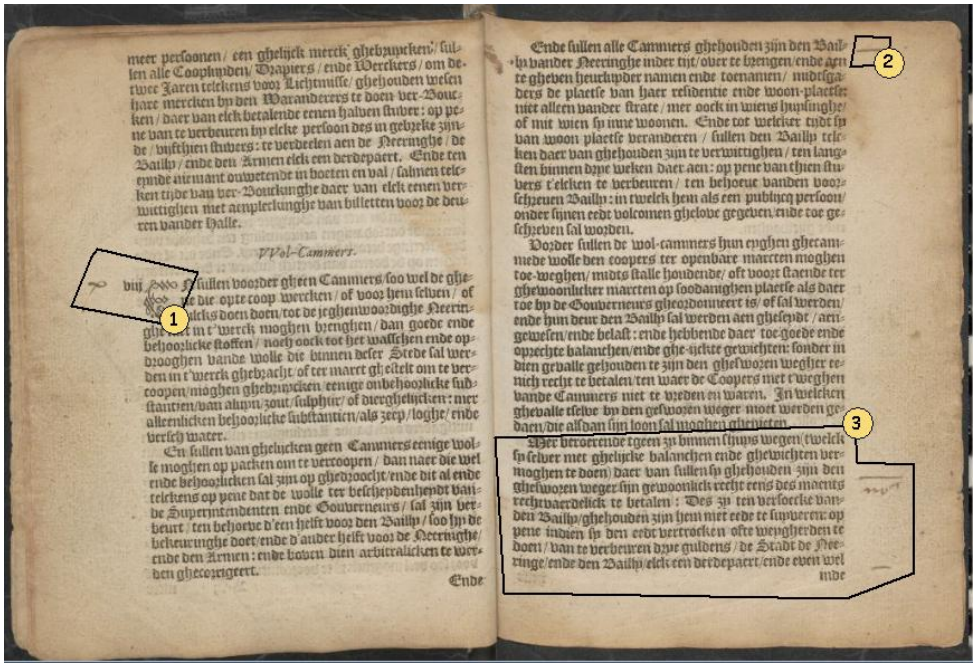
Delft gheregt ende ghemaect zullen werden/ Saey ende Cotten-hullemen/ Cotten Bierscharsten/ Negen-oogen/ breede ende smalle Doppen/ breede ende smalle Rebben/ ghebenderde/ platte/ ende Piemontsche/ ende ghe-volghe vandien.

Deur bevel van mijnen Heeren vande VVet der voors Stadt.

TOT DELF,

Ghedruct by Bruyn Harmansz Schinckel, vvoonende aent Marct-veldt inde Druckerije, Anno 1596.

Figure 12:



meer persoonen/ een gheliick merck ghebruycken/ sul-len alle Coopluyden/ Wapiers/ ende Werckers/ om de-twee Jaren te lekens voor Lichtmisse/ ghehouden weten hare mercken by den Waranderers te doen ver-Bouc-ken/ daer van elck betalende eenen halven stuber: op pe-ne van te verbeuren by elcke persoon des in gebreke zijn-de/ vijftien stuvers: te verdeelen aen de Heerinche/ de Bailiu/ ende den Armen elck een derdepaert. Ende ten eynde niemant onwetende in boeten en val/ salmen elc-ken tijde van ver-Bouckinghe daer van elck eenen ver-wittighen met aenplerkinghe van billetten voor de deu-ren vander Halle.

VVol-Cammers.

vij ende Hullen voorder gheen Cammers/ soo wel de ghe-woo-de die opte coop wercken/ of voor hem selven/ of ghe-lijcks doen doen/ tot de zeghen woordighe Heerin-ghe-ert in t'werck moghen brenghen/ dan goede ende behoorlicke stoffen/ nach ock tot het wasschen ende op-drooghen vande wolle die binnen deser Stede sal wer-den in t'werck ghebracht/ of ter maect gheleit om te ver-coopen/ moghen ghebruycken eenige onbehoorlicke sub-stantien/ van alum/ zout/ sulphur/ of dierghelijcken: mer alleenlicken behoorlicke substantien/ als zeep/ loghe/ ende versch water.

En sullen van ghelijcken geen Cammers eenige wol-le moghen op packen om te vercoopen/ dan naer die wel ende behoorlicken sal zijn op ghedroocht/ ende dit al ende telckens op pene dat de wolle ter beschepdenheydt van-de Superintendenten ende Gouverneurs/ sal zijn ver-beurt/ ten behoeve d'een helft voor den Bailiu/ soo ju de beleeninghe doet/ ende d'ander helft voor de Heerinche/ ende den Armen: ende boven dien arbitralicken te wer-den ghecorrigeert.

Ende

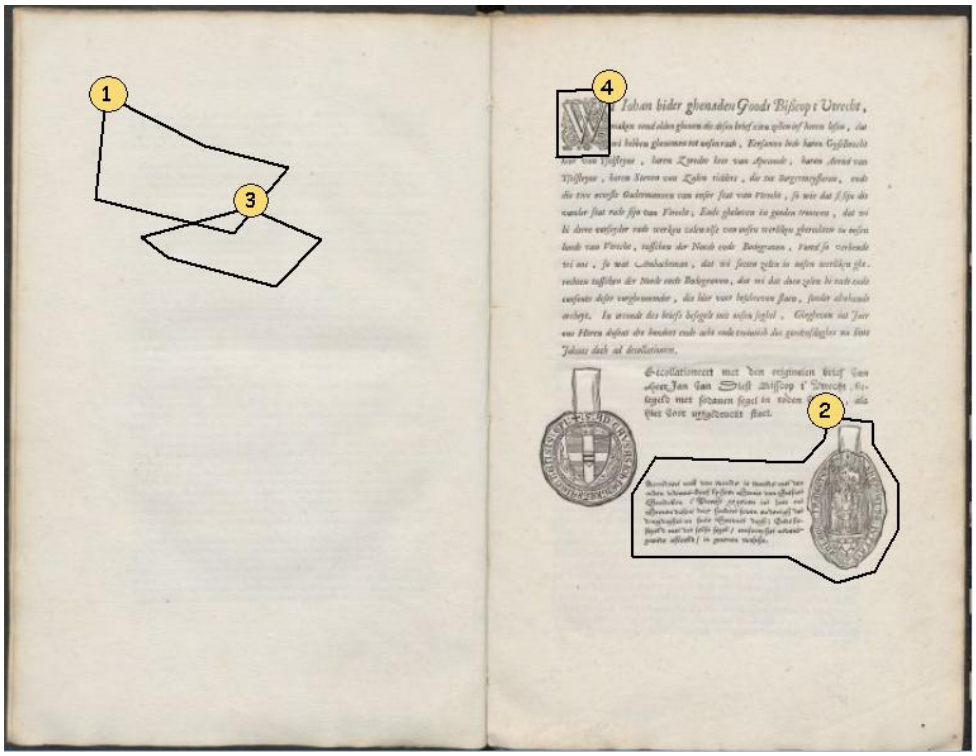
Ende sullen alle Cammers ghehouden zijn den Bail-liu vander Heeringhe inder tijt/ over te brengen/ ende aen te gheven heurluyder namen ende toenamen/ midtsga-ders de plaetse van haer residentie ende woon-plaetse: niet alleen vander strate/ mer ock in wiens huysinghe/ of nut wien sy inne woonen. Ende tot welcker tijdt sy van woon plaetse veranderen/ sullen den Bailiu telc-ken daer van ghehouden zijn te verwittighen/ ten lang-sten binnen drye weken daer aen: op pene van thien stu-vers t'elcken te verbeuren/ ten behoeve vanden voor-schreuen Bailiu: in welck heen als een publijcq persoon/ onder sinnen eedt volcomen ghelove gegeven/ ende toe ge-schreven sal worden.

Voorder sullen de wol-cammers hun eyghen gheram-mede wolle den coopers ter openbare maecten moghen toe-weghen/ midts stalle houdende/ oft voor staende ter ghewoonlicker maecten op soodanighen plaetse als daer toe by de Gouverneurs gheordonneert is/ of sal werden/ ende hun deur den Bailiu sal werden aen ghelendt/ aen-gewesen/ ende belast: ende hebbende daer toe goede ende oprechte balanchen/ ende ghe-yckte gewichten: sonder in dien ghevalle gehouden te zijn den ghelwozen wegher te-nich recht te betalen/ ten waer de Coopers met t'weghen vande Cammers niet te vreden en waren. In welcken ghevalle elcke by den gelwozen weger moet werden ge-daen/ die alsdan sijn loon sal moghen ghenieten.

Wer bevoerende tgeen zy binnen stuys wegen/ (welck sy selver met gheliicke balanchen ende ghewichten ver-moghen te doen) daer van sullen sy ghehouden zijn den ghelwozen weger sijn gewoonlick recht eens des maents rechtvaerdelick te betalen: Des zy ten verloerlie van-den Bailiu/ ghehouden zijn hem met eede te suyveren: op pene indien sy den eedt vertrocken/ ofte weygherden te doen/ van te verbeuren drye guldens/ de Stadt de Hee-ringe/ ende den Bailiu/ elck een derdepaert/ ende even wel

inde

Figure 13:



These examples also illustrate the effect of the color of the bullet. The choice landed on a soft yellow ochre(#F5E49C) in combination with a black number. The contrast with the page is low, making it pleasant for the eyes and easy to read the page near the bullet.

2.6. PDF make-up

Resolution

Because of the type of research, where books are examined in great detail, the exported PDF should have as high a resolution as possible. However, this may be problematic when exporting large sets of scans, due to size of the file.

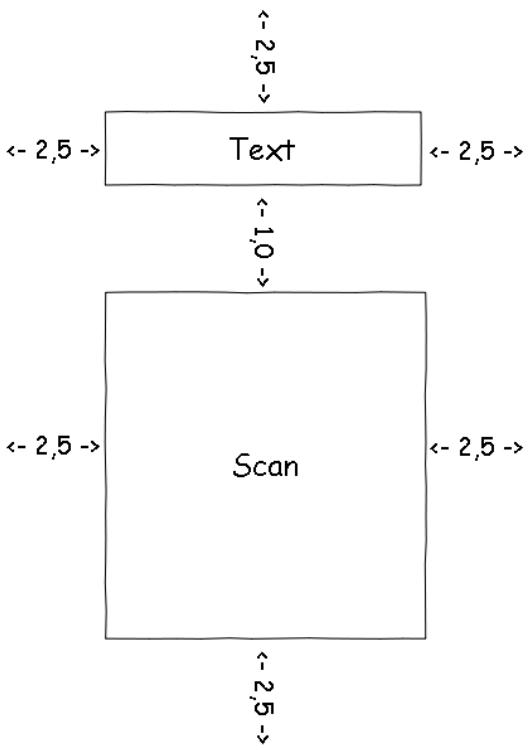
Therefore, when exporting a single scan or just a few pages, the suggestion is to get the dpi as high as is feasible. When the sets of pages get larger, the quality may decrease in favour of the file size. The user will be warned for this quality decrease in the Export tab.

Margins

The margins for pages with normal text (the front page, the index, transcriptions), are 2.5 cm on all sides. Pages displaying only scans have a margin of 1,25 cm on all sides. Even though the goal is to have the scans mimic the actual book, the framing helps to contain the scan, making it more pleasant to look at.

For the margins on the single scan, see Figure 14. All margins are in cm.

Figure 14:



Document information dictionary

- Title
  - Single scan and a set of scans - the title of the book the first scan was taken from.
  - Binding - the title of the first Book in the Binding
- Author
  - Single scan and a set of scans - the author of the book the first scan was taken from
  - Binding - the author of the first Book in the Binding
- Creator - the name of the collaboratory
- CreationDate - speaks for itself

1.11 FD - Global Layout

1.11.1 1. Concepts

This page describes the global layout, including UI elements and positioning used for general actions.

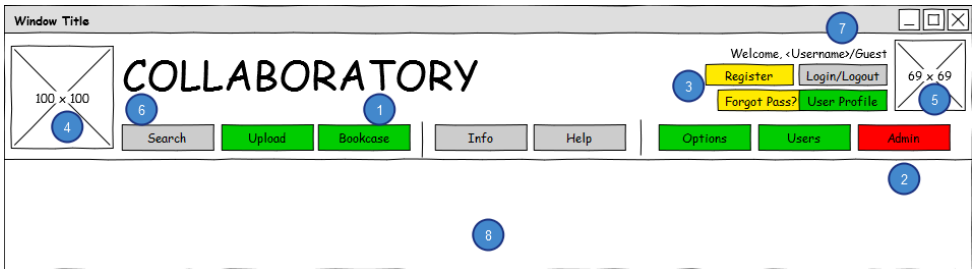
1.11.2 2. Layouts

The following describes layouts used on the website.

2.1. Header

This layout is for the header, which should be visible on all the pages of the website.

Figure 1:



List of screen objects for Figure 1:

1. Green items: only visible for logged in users (this includes admins).

2. Red items: only visible for admins.

3. Yellow items: only visible for guests (these disappear after logging in).

   (a) 'Forgot Password' / 'Account Retrieval' button takes the place of the 'User Profile' button for guests (guest has no profile, after all).

4. Image on the left: website icon

5. Image on the right: user avatar (if any)

6. Buttons are positioned next to, instead of below, the logo for a more compact header, and less waste of space at the top of the website.

   (a) Buttons are grouped in roughly 3 groups: "**Main functionality**" (Search, Upload), "**Information / Help**" (TO ADD: "HELP" BUTTON), and "**Options / Secondary functionality / Users / Admin related**".

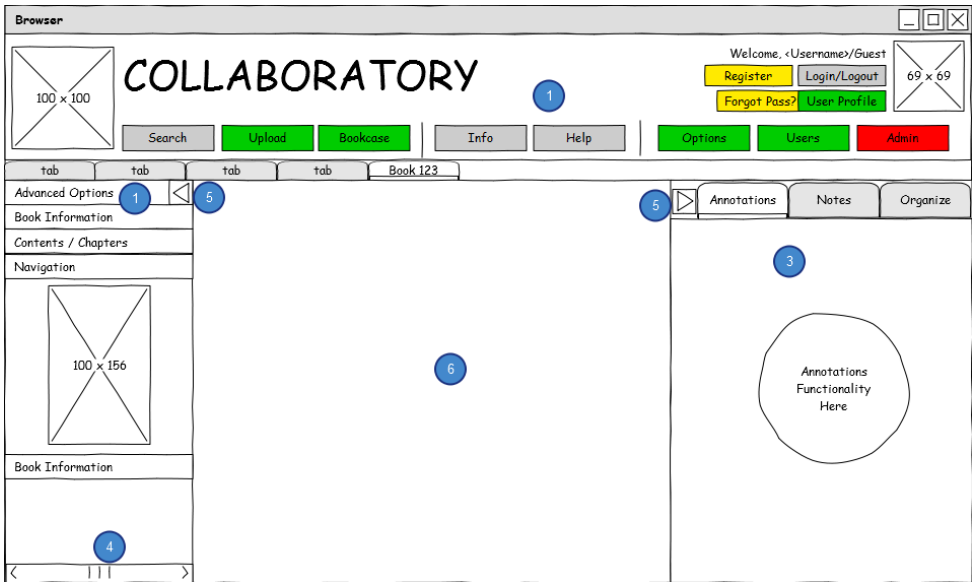
7. "Welcome, Guest" is replaced with "Welcome, <Username>" after logging in. Similarly, "Login" is replaced with "Logout". This reverts after logging out.

8. The empty space below the header is filled with tabbed pages (not shown here, to keep focus on the header).

2.2. Global Layout

The following image also includes the tabs, and sidebars (in this case, used for the viewer).

Figure 2:



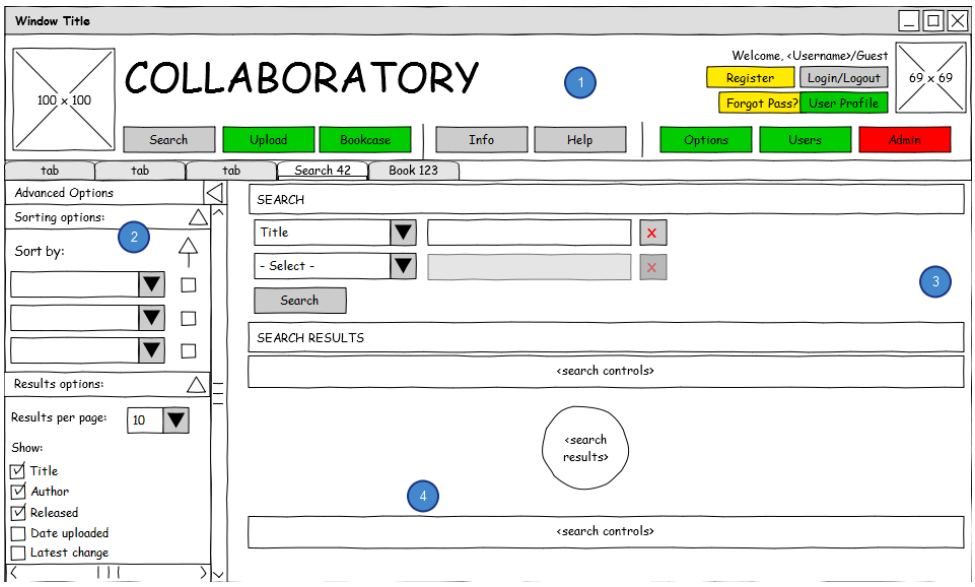
List of screen objects for Figure 2:

1. At the top, below the aforementioned header: tabs for searches and search results, open books and more.
2. On the left: information sidebar, meant purely for output from the system. This contains information and overviews, and should contain as little input fields as possible. 'Accordion' UI item.
   (a) Multiple things can be open in this at the same time.
   (b) 'Output' in this context means: 'showing, or changing the visualisation of information'. User input given here is limited to selecting checkboxes to modify what is shown in the working area and similar, short-term changes.
3. On the right: work sidebar, meant for input to the system. This is the working area for the user, including notes and annotations, and pretty much all other editing functionality. Tabbed, for quick switching while working.
   (a) 'Input' in this context means: 'changes to the data in the system'. User input here is usually saved long-term (during the session or even to the database), expands beyond the current screen, or similar.
4. If either sidebar is currently showing something that is too wide or high for the sidebar, a scrollbar should appear (as shown on the left, despite there being nothing too wide currently shown in Figure 2). These should be hidden if not necessary.
   (a) Sidebars can have a variable width where necessary, which the user can control, to make it less likely they will have to use this scrollbar.
5. Both sidebars can be folded away (and back) using the arrows, to free up space when not in use.
   (a) If a screen has nothing to put in either sidebar (or both), the would-be empty sidebars are simply not shown.
   (b) Sidebars are reopened the way they are shown by default, every time a new screen is opened. User made changes such as folding away, selected tabs, and variable width are not saved outside the screen.

6. The empty field in the middle is where the page and controls would be shown: this is left blank in Figure 2 for clarity as it is not the focus of this image. Annotations is as of yet unfinished and therefore only indicated.

Keeping the same 'tools for information presentation in the left sidebar' guideline in mind, the following image is another example showing an implementation of the concept. This time for the search and sorting functionality for books:

Figure 3:



List of screen objects for Figure 3:

1. Again, the top (header and tabs bar) is default and similar at every point of the collaboratory for consistency. The only change is that a different tab (for search) was selected.

2. On the left: the information sidebar, as shown in Figure 2. The purpose is unchanged: present information, and changing the way information is presented. Two options are shown: "Sorting options" and "Results options".

   (a) "Sorting options" is at the top, as we assume this will more frequently be used. As this only changes the order of the information presented on the right, and not the information itself, it belongs in the left sidebar.

   (b) "Results options" does something similar: it changes the amount of information that is presented. However, as all the information is already in the system and is in itself not affected or changed, this functionality too belongs in the left sidebar.

   (c) Both of the above options lists are expanded in Figure 3, resulting in a sidebar that is too high to fit everything on the screen. Therefore, a scrollbar is automatically added so that users can still see and use the options at the bottom.

3. On the right, no sidebar is presented, as there is nothing to edit or add on this page.

   (a) Note: for admin usage, contents might have to be moderated. In this case, it is a possibility that the admin could select and remove books, listed in the search results on this page. This functionality would belong in the right sidebar.

   (b) If "personal bookshelves" are implemented, adding books to (or removing them from) these shelves directly from this page would also fit in the right sidebar.

4. For the search functionality and layout itself, please refer to FD - Book Search, Sort, Select.
   (a) The only new thing here is the addition of another set of search controls (more specifically, the next/previous page buttons) at the bottom of the list, so that users won't have to scroll all the way back to go to the next (or previous) page of search results.

1.12 FD - Help

1.12.1 1. Interface

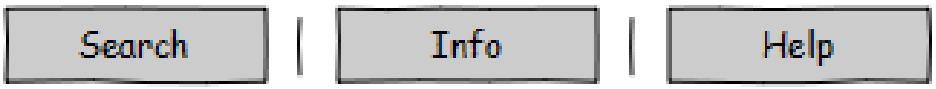
1.12.2 1.1. Design choice

We have chosen to design and implement a full-page solution rather than a sidebar solution. With the help in the sidebar things would get really crowded and cluttered really fast. With the full-page option there is more space and liberty to describe the functionality. The sidebar option was a good option because the help would be in context. The drawback in the full-page option (either the help text, or the subject material tab is visible at any time) is not that bad either. After the user navigates to the help it is very easy to navigate back to where he came from to continue initial work.

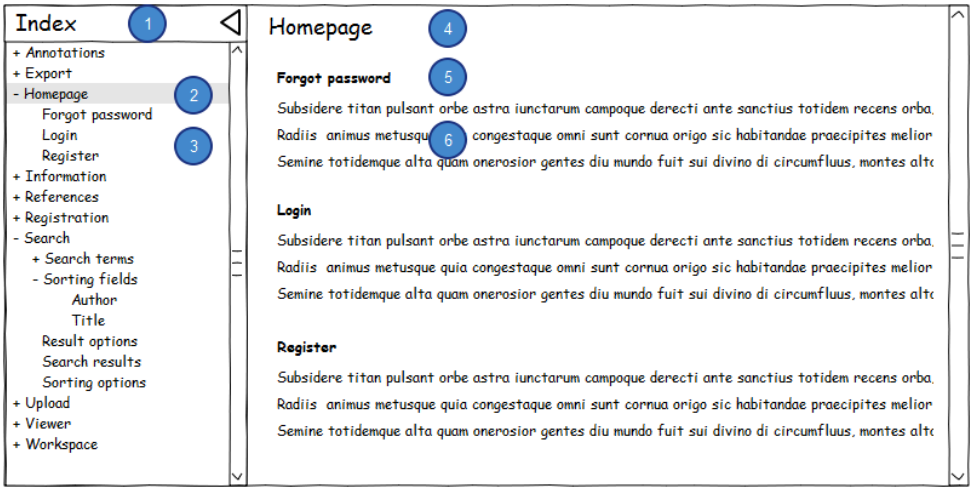
This help being a page of course means adding a link to the page. This will be a simple button "Help" on the homepage.

1.12.3 1.2. Mockups

This is the button on the homepage that will lead to the help page. It will open a tab just like other buttons. All (types of) users are able to view the help at any time.



The following image shows the help page:



List of controls:

1. Sidebar: called "Index", collapsible, just like the other sidebars.
2. Tree: page item: These are expandable items in the tree. One page item leads to the page shown in bullets 4 through 6. One page item may contain zero or more other page items and one or more control items. Collapsed it shows a "+" in front of the text, expanded it shows

a "-". Clicking the "+" or "-" will toggle between these. When expanded, all items are shown indented below the item. Items are sorted, first all page items sorted alphabetically, then all control items sorted alphabetically. Clicking on the item itself will open the page with all control items in it (bullets 4 through 6). The currently selected item will be coloured. In the right side the children control items will also be visible. First all children page items will be enumerated (in the same order as in the tree) with a description describing the entire tree item. Next are all control items (again, in the same order as in the tree) with a full description about that item.

3. Tree: control item: Clicking will open the parent page and jump to the control's title (bullet 5). These control items are not expandable. If possible they should have an icon in front of the text in the tree. This way a button (consisting of only an icon) can be quickly associated with the corresponding help item. These icons will also be shown in from of the text in bullet 5. The currently selected item will be coloured.

4. Page title. The same text as the selected page item.

5. Control title. A text, possibly with icon or small image in front of it.

6. Control text. A text, possibly with icons and images, explaining the functionality of the control. The text should also be able to contain links to other help items. So for example when a process is explained, navigating to the next control is very fast and easy.

The screen opens with all items in the tree contracted.

If possible, the tab the user was in when he pressed the help button should open in the main window and the item in the tree expanded.

Else we can make a page explaining the help which is only shown in the main window and not in the tree. So this page is only visible when the user opens the help in a new tab.

1.12.4 1.3. User management and rights

The exact content of the help tree depends on the current user. The user can only view the help items which he can actually see and use. For example, the "Login" is only visible to anonymous visitors. Likewise, "Logout" is only visible to logged in users. Another example: "Users" is only visible to an Administrator. Likewise, reverting user changes is only available to Moderators (and implicitly to Administrators).

This means every item in the tree (pages and controls) can be visible of invisible. When an entire page is invisible, all items nested in that item are invisible as well. If we construct the help pages about one page per tab or sidebar, this will all work out.

1.13 FD - Managing user generated content

1.13.1 1. Concepts

This page describes the way user generated content is managed. When things are unclear or inconsistent, please contact someone from functional design (or someone who used to belong to functional design).

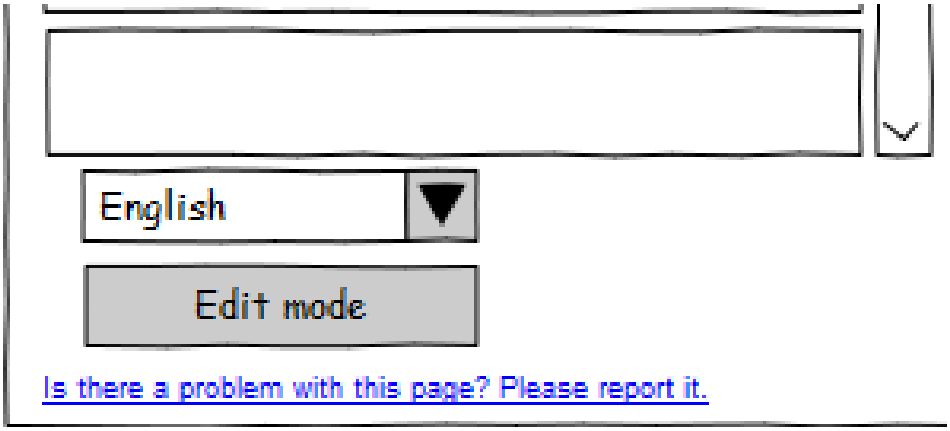
1.13.2 2. Reverting changes and making corrections in the collaboratory

2.1. Version control

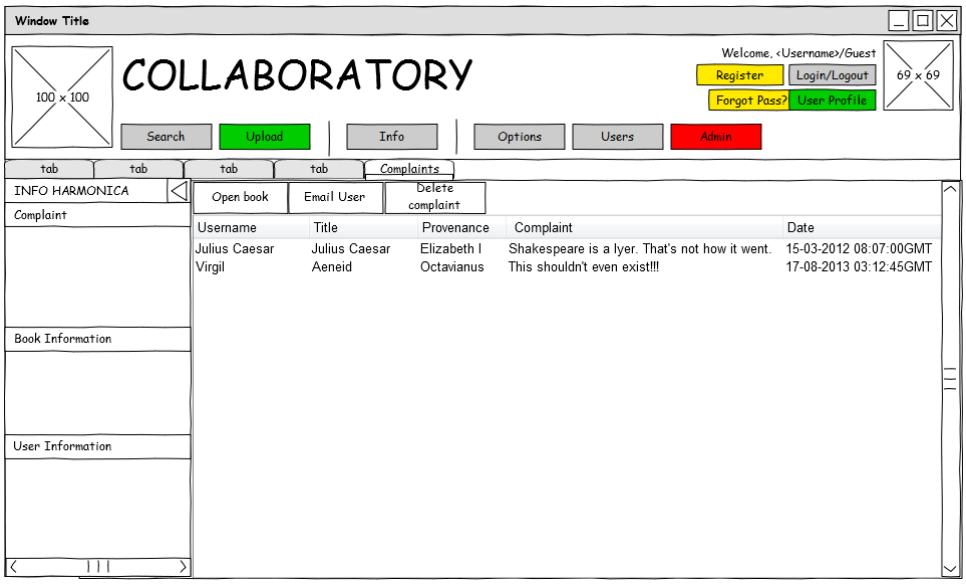
Whenever a page is saved, a new version of the page will be created and this version will be set to the current version. We will also store all previous versions, so that moderators and admins will be able to return a page to a previous version. Returning a page to a previous version will also count as a save. With each save we will store which user saved it and when he did it.

2.2. Reporting mistakes

When mistakes have been found in key book information like author, title, etc. users will be able to report this to the moderators. A hyperlink will be placed in the transcriptions sidebar which users can click on to show a popup screen. In this screen they will be able to enter their corrections in a simple textfield.

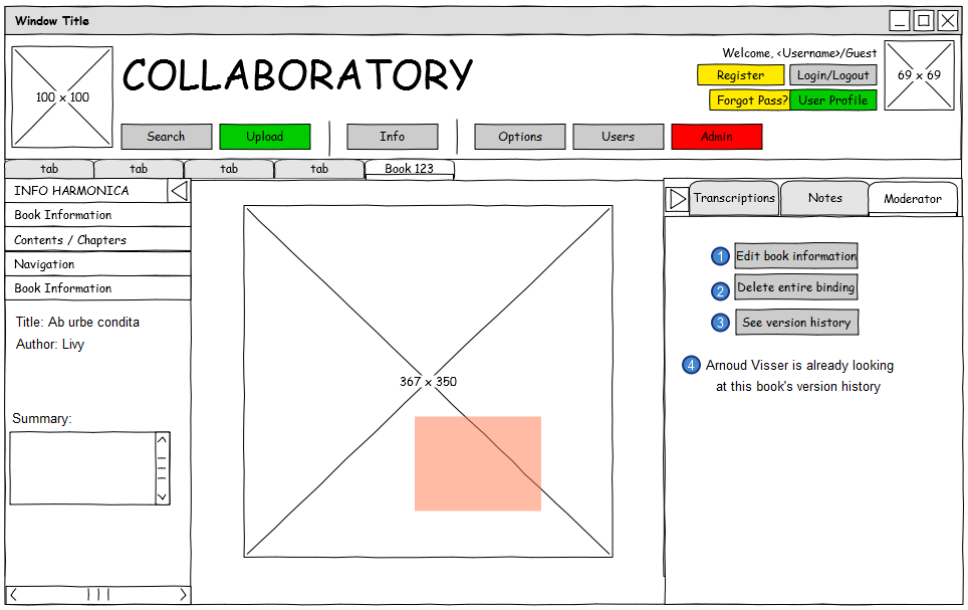


The moderator will be able to view the complaints in a complaints screen which is reachable by clicking on complaints, a field in the dropdown menu of the options button.

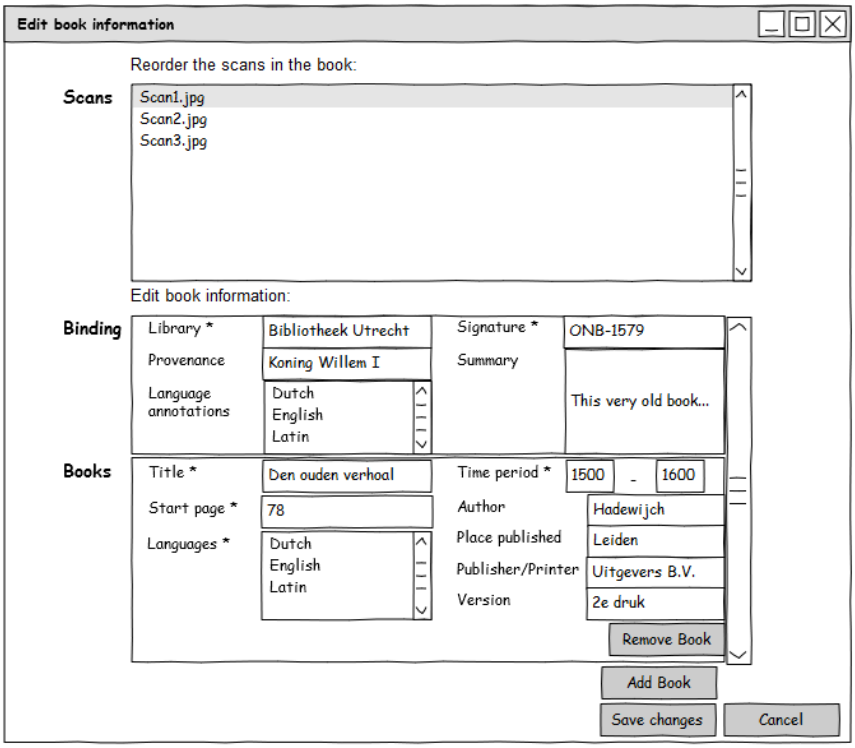


2.3. Correcting mistakes

When viewing books the Moderator has his own tab in the right sidebar.



1. **"Edit book information" button**: When one selects this, a screen similar to the uploading screen will show up with the current information. The user will then be able to change all of the key information which defines the book. The buttons "Save changes" and "Cancel" will be located at the bottom of the screen. "Save changes "will save the changes made and "Cancel" will do nothing and return the user to the screen he was previously viewing.



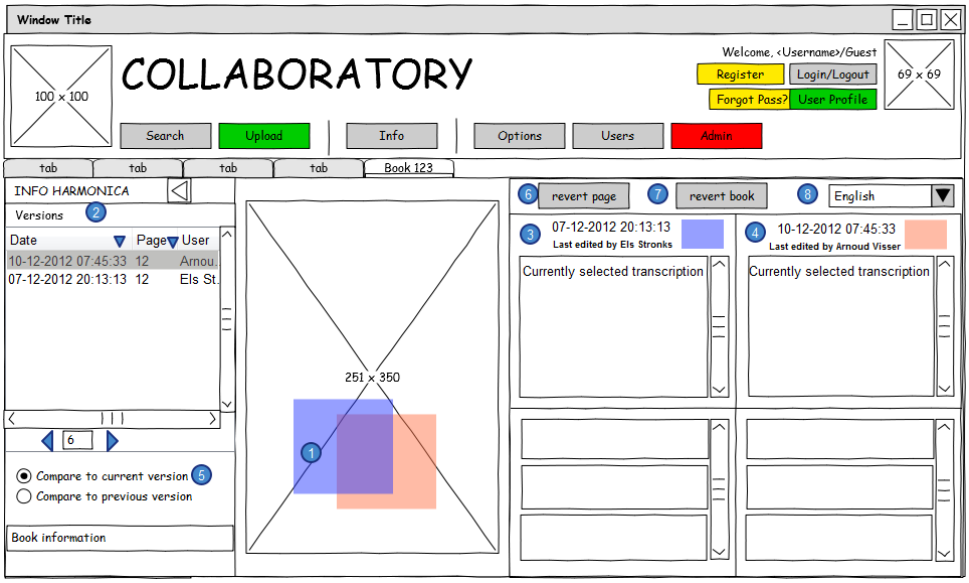
2. **"Delete entire binding" button**: When one selects this a prompt witht he following text: "This will remove the entire binding from the collaboratory and there will be no way to retrieve it after it is deleted. Is that okay?". When a user clicks "OK" the book will be deleted. When he selects "Cancel", the prompt disappears and nothing happens. Users editing a page from the deleted binding will receive a prompt telling them the binding has been deleted and they will be returned to view mode.

3. **See version history:** Will redirect the moderator to the version history screen (See 2.4 Viewing old versions).

4. **Locked message:** This tekst is normally not shown. While another Moderator is viewing the version history or editing the book information, buttons 2 and 3 become greyed out and the tekst "<insert Moderator's username> is already looking at this book's version history" appears.

2.4. Viewing old versions

The screen is similar to viewing annotations (FD - Annotations), but you're viewing two sets of annotations at the same time: one corresponding to an older version and the current one. Or an older version and the last version before it (the 'previous version'). When the version history is opened (See 2.3 Correcting Mistakes 3), the default shown is the currect verion compared to the version before it.



1. When you mouse over a polygon the polygon becomes blue if it belongs to the old version and red if it belongs to the current or previous version. (this if is not exclusive)

2. The accordion (incorrectly named Harmonica in the image) contains a list of versions, with the timestamp from when the version was created, the changed page and the user who created the version. In this list we can group versions by page and filter them by date. For this list we use the ext js list: <insert link(the sencha server is down atm)>. 20 versions are shown in the list, to see the other versions the user can use the arrows underneath the list, or enter a page number and press enter. When a user clicks on a version, that version is shown to the user in the current tab (it replaces the currently shown version). The page changes to the page the version refers to. The selected version becomes gray in the list and is shown at 3. At 4 the current or previous version is shown, depending on the selected radiobutton at 5.

3. The transcriptions from the old version selected at 2. Clicking on one selects it and lights up the corresponding polygon in blue.

4. The transcriptions from the current or previous version. Clicking on one selects it and lights up the corresponding polygon in red.

5. Here the user can select whether he wants to compare the version selected at 5 with the current or the previous version. When the user changes this button the version shown at 4

changes to the correct version.

6. The revert page button: reverts the transcriptions and polygons on the page the Moderator is viewing to the version selected at 2. This creates a new version which becomes the current version. Users editing the page will receive a prompt telling them the version has been reverted and they will be returned to view mode.

7. The revert book button: reverts the transcriptions and polygons in the book the Moderator is viewing to the version selected at 2. For each changed page this creates a new version which becomes the current version. Users editing a changed page will receive a prompt telling them the version has been reverted and they will be returned to view mode.

8. The language dropdown box: here the Moderator can choose in which language the transcriptions in 3 and 4 are shown. This contains the same languages as the language dropdown box in FD - Annotations.

1.14 FD - Notes + Snippets

1.14.1 1. Concepts

This page describes the 'notes' functionality, which allows users to keep personal notes.

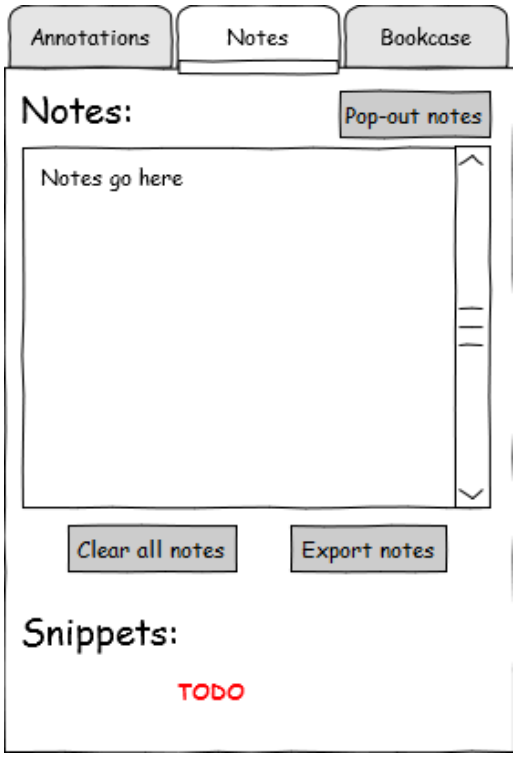
1.14.2 2. Notes

The following describes the notes functionality.

2.1. Notes sidebar tab

Notes tab:

Figure 1:



List of screen objects for Figure 1:

1. The notes tab includes a textfield with vertical scrollbar (disabled while not needed), a button to clear all notes (to remove the need to select all text and remove it manually), and a button to export the notes. To accomodate people who prefer the sidebar out of the way, a button to pop-out the notepad to a separate window can also be implemented (see Figure 3).

2. The notes tab can be found in the right-hand sidebar wherever applicable. This includes:

   (a) Viewer (for notes on books)

   (b) Bookcase (for notes on what books to check next, for instance)

   (c) Search? (might be useful to make notes on what queries get better search results, etc.)

   (d) Possibly other screens.

3. The most important aspect of the notes feature is that the text is synchronized amongst all open tabs. Or more clearly: there is only one instance of the notepad, which is shown and can be edited in every tab. An edit on the notepad in one tab will be automatically saved and shown in all other tabs where it is open.

4. The notepad is simple plain text, with linebreaks and wrap-around enabled.

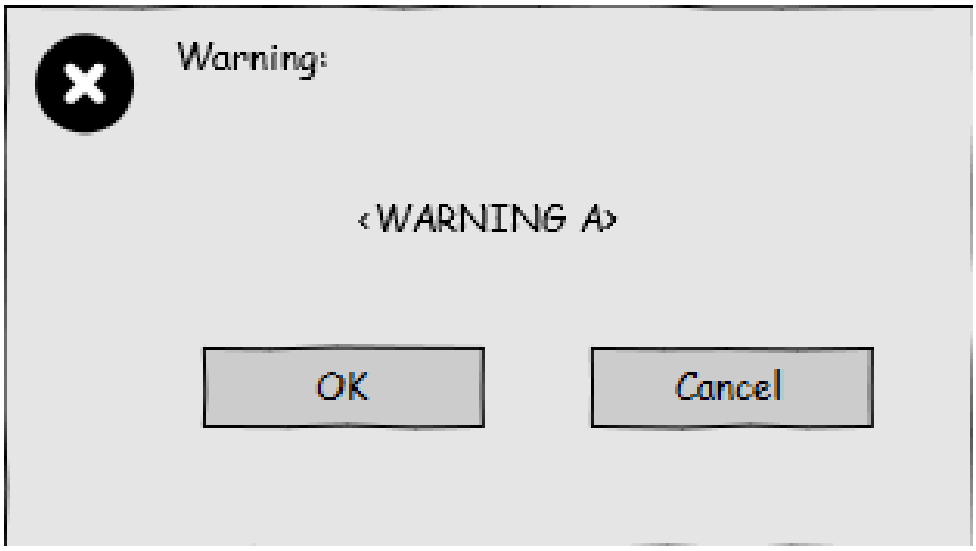
   (a) Resizing the sidebar will allow for longer lines, so the wrap-around must not be set to a strict line-length, but rather the actual size of the notepad in the sidebar.

5. Pressing the 'Clear all notes' button will bring up a warning message before actually clearing all notes: see Figure 2.

6. Notes should be saved / synchronized automatically, and not only when closing a tab or by pressing a button (hence no 'save notes' button), in case the user leaves the website by closing the browser (or because of a crash).

'Clear all notes' warning message:

Figure 2:



List of screen objects for Figure 2:

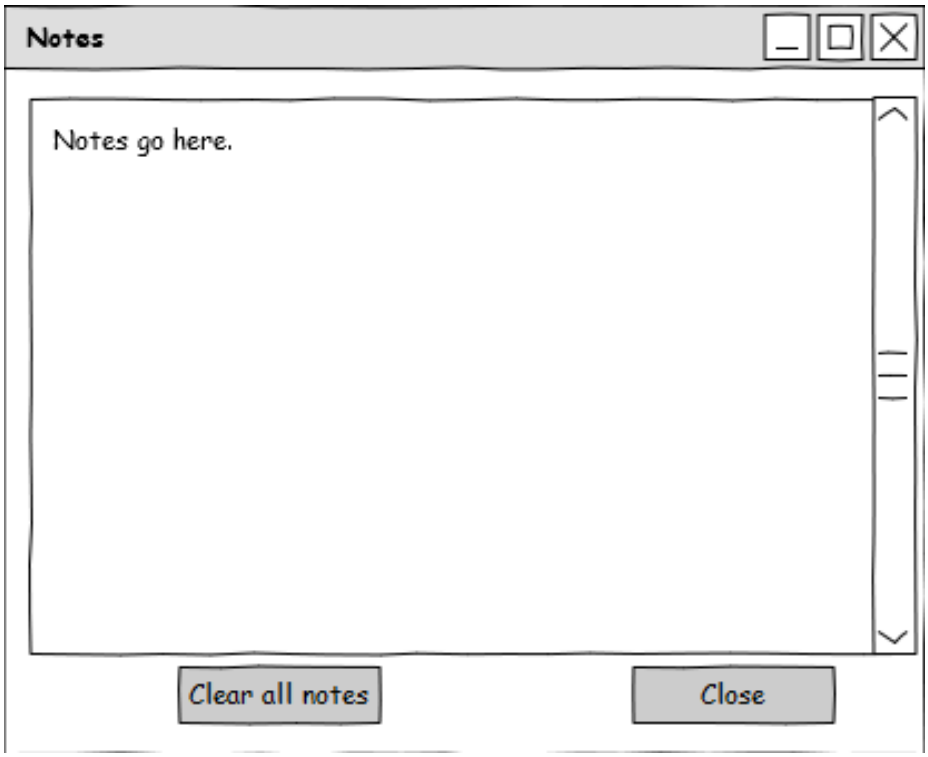
1. <WARNING A> in full (mockup software wouldn't wrap the text):

    (a) **Clearing all notes will remove the contents of your notepad. Be aware that this will apply to all your pages, as the notes are synchronized, and can not be undone. Do you want to continue? (OK / Cancel)**

2. The "OK" button clears the notepad, while the "Cancel" button leaves the notepad intact.

Notes pop-out window

Figure 3:



List of screen objects for Figure 3:

1. After pressing the pop-out button, a separate (freely movable and resizable) window pops up, showing the same notepad, and the 'clear all notes' button found in the sidebar.

2. Functionality of the notepad itself is identical to the sidebar version of the notepad.

3. The 'Clear all notes' button shows the same warning message (Figure 2) as it does in the sidebar.

4. Both the default 'X' button in the top right corner and the 'Close' button close the pop-out window.

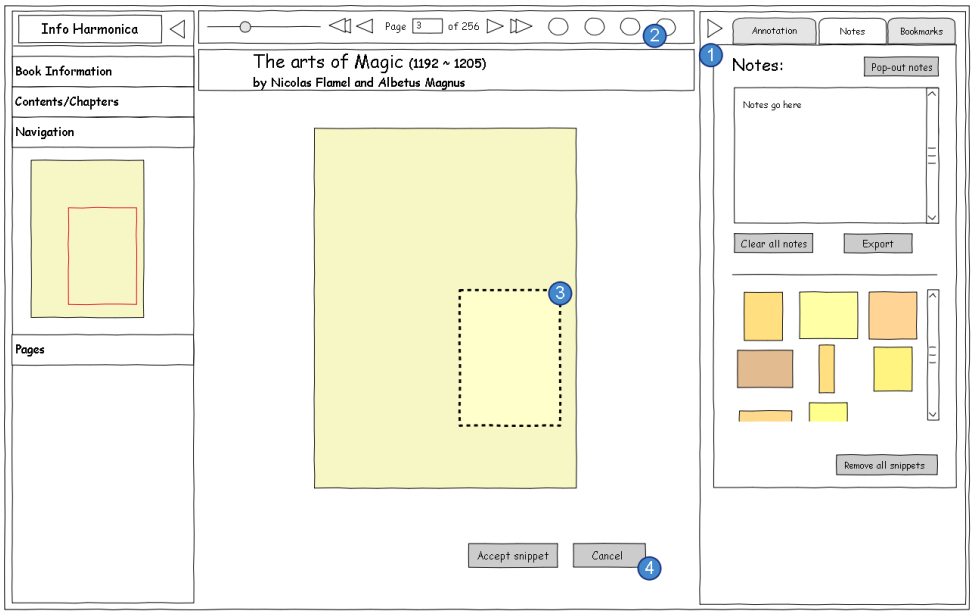
1.14.3 3. Snippets

In the same tab as the notes there is the snippets section.

3.1. Taking snippets

Here is a view of the Viewer with the notes tab:

Figure 4:



List of screen objects for Figure 4:

1. The notes tab in which notes and snippets are stored, viewed and edited.
2. The snippet tool button. When clicked, the snippet tool will be selected. The notes tab does not have to be opened to use the snippet tool, snippets can be taken at any time.
3. A snippet being taken from the displayed page. The selected area lights up and is enclosed by a dark dashed line. It will be called a snippet rectangle.
4. The snippet accept and cancel buttons. These buttons only appear when a snippet rectangle is placed.

When the snippet tool is selected, the user can draw snippet rectangles on top of the scan.
The tool will be unselected by:

- Pressing the snippet tool button again.
- Clicking with the right mouse button
- Clicking outside the Viewer
- Using an other Viewer function
- Moving to another tab

A snippet rectangle can be draw by clicking somewhere on the page and dragging the mouse towards another point on the page. When the mouse is released on the other point, the rectangle will be formed. These two points are opposite corners of the rectangle.

The page itself can be at any zoom and rotation, this will be taken into account with the snippet. If the snippet is viewed again later, it will be displayed in the same rotation. The zoom of the snippet depends on the available space for the snippet. Also, the snippet will always be an upright rectangle, the rotation does not rotate the border of the snippet.

When the user has drawn a rectangle on the page, the snippet accept and cancel buttons will appear in the Viewer. The reasoning can be found in the research page. The user now has several options:

- Click the accept button

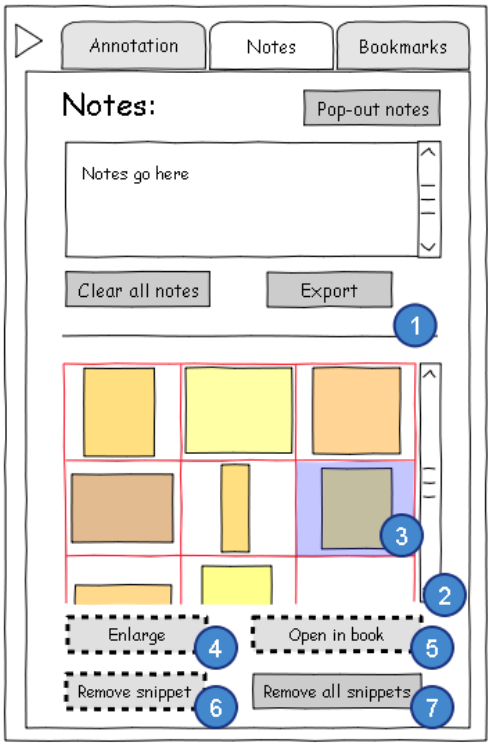
  - The selected area of the page is turned into a snippet
  - The snippet tool is unselected
- Click the cancel button
  - The selected area is not turned into a snippet
  - The snippet tool is unselected
- Unselect the snippet tool in any other way
  - The selected area is not turned into a snippet
- Draw a new rectangle
  - The former snippet rectangle is removed
  - The new snippet rectangle is displayed
  - The snippet accept and cancel buttons are still displayed
  - The user can again preform an action from this list.

If a snippet is accepted, it is stored in the snippet list. See the section 'The notes tab with snippets'.

3.2. The notes tab with snippets

The notes tab with the snippets section:

Figure 5:



List of screen objects for Figure 5:

1. A draggable border between the notes textfield and the list of snippets. This way, the user can decide for himself how large the notes field should be. There is of course a minimum size to the notes section: there should be adequate room for the buttons and 4 lines of text in the textfield.

2. The snippets list, a list of thumbnails displaying all the collected snippets. When the list gets to long to display in full, a scrollbar is added to the side.

3. A selected snippet

4. The 'enlarge' button will enlarge the snippet, see Figure 6

5. The 'open in book' button brings the user to the page the snippet was taken from. The page is displayed in the same zoom and rotation as the snippet and has its view set to the part where the snippet was taken from.

   (a) If the book the page is from has no instance running, the page is opened in a new viewer tab

   (b) If the book has an instance running, the user is taken to the book's tab and given a popup asking if he wants to turn to the page of the snippet.

6. The 'remove snippet' button is only available if a snippet has been selected. When pressed, it will remove the snippet without further question.

7. The 'remove all snippets' button empties the whole list of snippets. The user is given a popup asking if he is sure about deleting all snippets. If he clicks cancel, nothing happens. If he clicks yes, all snippets are deleted from the list. This button is only usable if there are snippets in the snippet list.

The thumbnails in the snippets list are matched into a grid, as displayed in Figure 5 in red. They are centered into their own rectangle. The grid is not visible for the user, giving the illusion of the snippets freely laying in the tab. The thumbnail should have the same proportions as the actual snippet, so a very thin snippet can have wide empty sides when displayed in the list. To make the thumbnails stick out against the background, a 1 pixel wide black line is added around them.

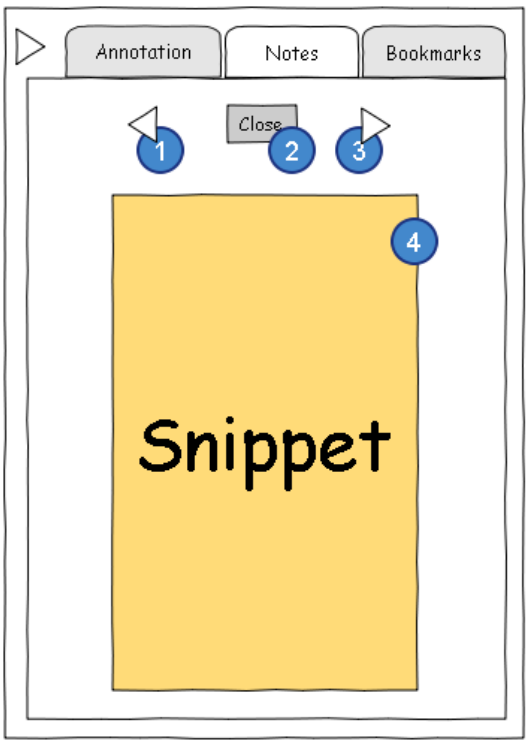
The buttons with the dashed border are invisible while no snippet is selected. This is seen in Figure 4. By left-clicking the snippet once, it is selected. The gridpiece it is on will turn transparent blue, as seen at point 2 in Figure 5. The buttons 'enlarge', 'open in book' and 'remove snippet' are made visible. The snippet will remain selected while the notes tab is open to the user. More concrete, when the viewer tab is closed or switched, or another tab is selected in the working area, the snippet is unselected. When a snippet is enlarged, it will also be unselected.

Furthermore, double clicking a snippet enlarges it as well.

3.3. Enlarged snippet

The thumbnail list is an easy way of displaying all the collected snippets, but not such a good way to view the snippets. When the user double-left-clicks on snippet, or clicks the 'enlarge' button, it should be enlarged within the notes tab.

Figure 6:



List of screen objects for Figure 6:

1. The previous snippet button. When clicked, the previous snippet on the list will be displayed. This function is only available if there is a previous snippet.

2. The close button returns the notes tab to its normal view, as displayed in Figure 5. The setting of the draggable border should be the same as it was when the enlarged snippet mode was entered.

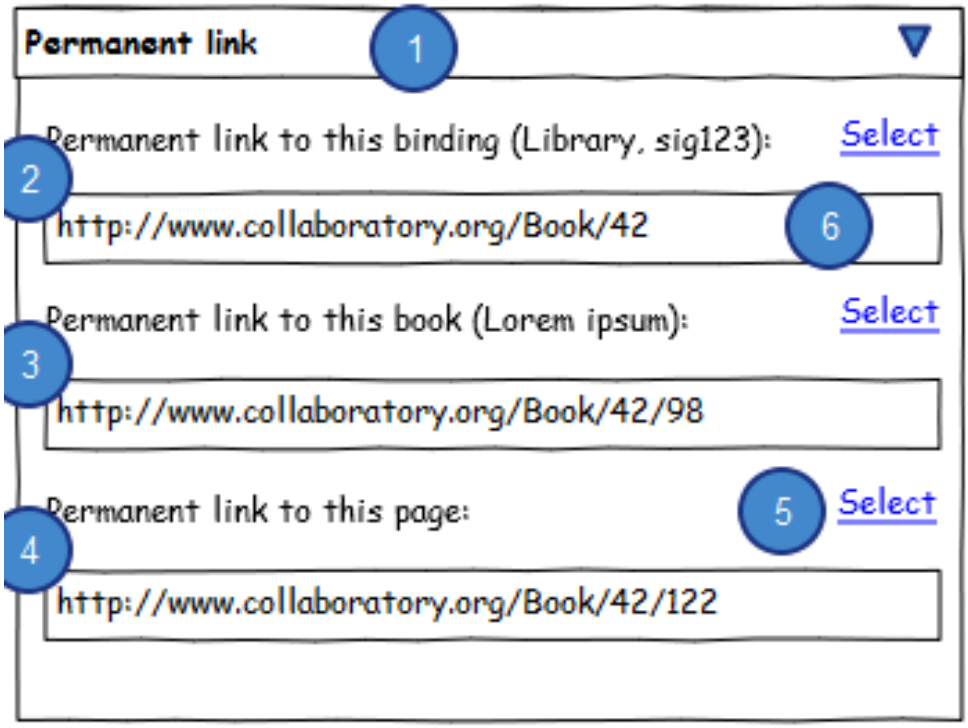
3. The next snippet button. When clicked, the next snippet in the list will be displayed. This function is only available if there is a next snippet on the list.

4. The enlarged snippet. It is matched to the size of the notes tab and centered; there will be no scrolling bar or zoom function. If it is a very large snippet, details may be hard to view. If it is a wide snippet, there will be empty space below and above the snippet. When the size of the notes tab is edited, the snippet zoom should be updated accordingly.

1.15 FD - References

1.15.1 1. References

This panel is integrated in the accordion at the left part of the layout. It will only be visible in the viewer tab.



List of controls:

1. Title of the panel in the accordion. The little arrow will show/hide the panel's content.
2. Controls to link to the current binding. In the text between the (and) the Library and Signature of the Bining are show, comma seperated.
3. Controls to link to the current book. In the text between the (and) the title of the book is show.
4. Controls to link to the current page.
5. Select link: Clicking will select the URL in the corresponding textbox.
6. Textbox with URL: is read only so cannot be modified, but it can be selected and copied from.

What the URL exactly is does not matter. The point is that we should be able to link to a binding, book or page.The URL to a page should open that page in the viewer.The same for the book except that it opens the very first page of the book.The same for binding except the viewer opens at page 1 of all the scans. So not the first page of the first book.

1.16 FD - Upload

Please see the Reseach about uploading for more background information on this design.

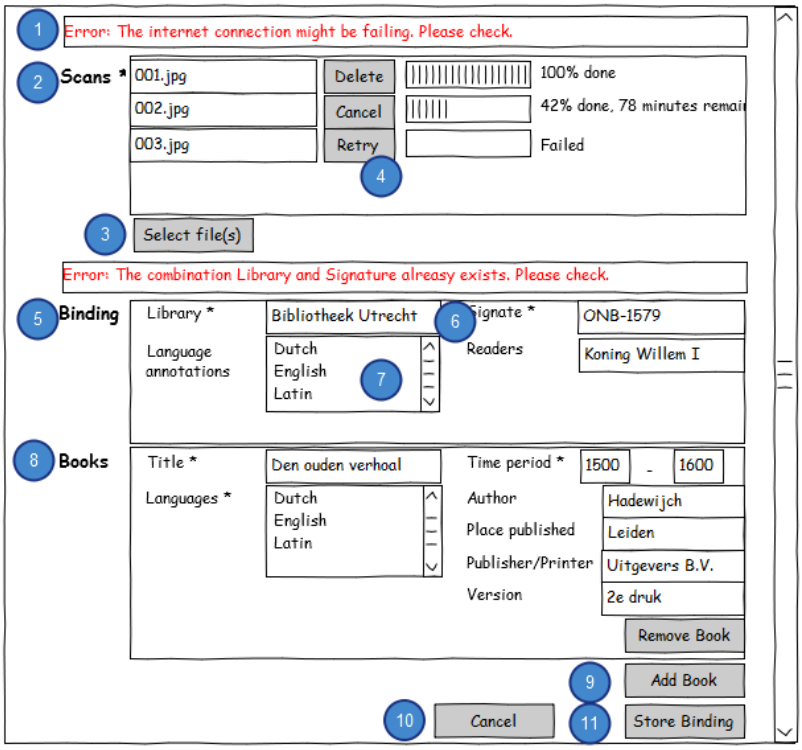
1.16.1 1. Step 1: Instructions

When startig with the upload process, the very first step is the instructions. It shows a simple screen with text instructing how to upload images and insert information. There will also be text to explain the requirements of the images, for example quality of the images.

Furthermore the screen contains a button at the page saying "Go to upload screen". Clicking will proceed to step 2 of the process.

1.16.2 2. Step 2: Upload

This step is the process of the uploading itself. I. e. the process of putting the information and the images into the collaboratory.



1. This box only appears when an error is encountered. It contains the nature of the error and a suggestion to solve it.

2. A line of controls to upload an image file. From left to right: a textbox containing the selected file name, a button ("Choose", "Cancel", "Retry" or "Delete"), a progress bar showing the upload progress, a label showing the percentage done and estimated time remaining. There is one line of controls for each image file that is selected, uploading of finished uploading. If a file upload failed it should say that in its status. The User should be able to select that file again to start uploading the same file.

3. A button "Select file(s)". This should use a box to select one or more image files. When the selection is done, these files are added to the list of 2.

4. A button with four options. First, "Choose" when no file is selected, it enables the user to select a file for upload. Second, "Cancel" when the file is uploading. It should stop the file from uploading and remove it from the list of controls on 3. Third, "Delete" when a file is finished uploading, it enables the User to delete the file. The file is removed from the system and from the list of controls in 3. Fourth, "Retry", when a file failed to upload the User can click this button to retry uploading. If the User chooses not to retry the upload this will not throw an error and just ignore the file.

5. A group of controls for entering information about a Binding. Fields marked with an asterisk (*) are mandatory. Other fields are optional.

6. The combination of the fields Library and Signatue should be unique in the system. When the User has entered this information the system should check the uniqueness of the combination. If it is not unique, the User should be notified.

7. A list of languages registered elsewhere in the system. The User is able to select multiple languages.

8. A group of controls for entering information about a Book. Fields marked with an asterisk (*) are mandatory, other fields are optional. The listbox Languages is identical to 7. When the User enters the first year, the second field should get that same value. Except when it already has a value. This is because in case of an exact year both field have the same value, else the User can change the second field later on to specify a period in case the year is not exactly known. The button "Delete Book" deletes the book and removes it from the list.

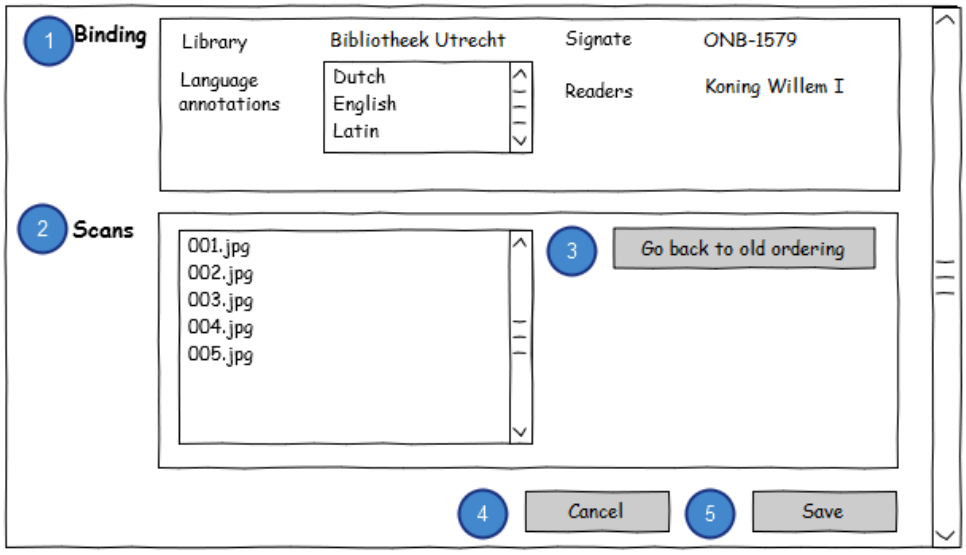
9. This button adds another group of controls like 8. The content is exactly the same. By clicking this button the system knows this is another book within the binding being entered.

10. This button will cancel the entire upload process and will delete all information about. So normally the user is able to resume the upload process at any time from any location, but when the user clicked Cancel this is not possible any more because all information is deleted.

11. This button is clicked when all available information is provided and all files have been uploaded. The system will store all information and notify the user about the result.

1.16.3 3. Step 3: Reorder

This step enables the user to change the order of the image files he uploaded. Most of the times the alphabetically sorted list is correct, but there are rare cases which will have to be corrected in the collaboratory.



1. A group of controls showing information about a Binding. No fields are editable.

2. A list of all uploaded image files. By using shift and ctrl the user is able to optionally select multiple files and then drag and drop it. This way the order of the scans can be changed.

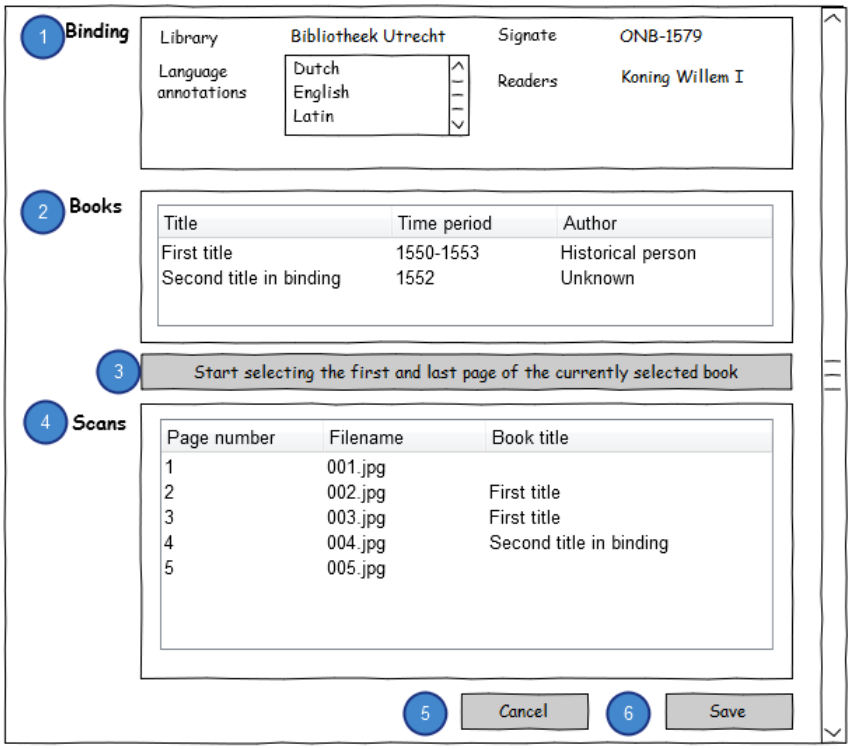
3. The button "Go back to old ordering" will reset the order of the image files to the original order (sorted alphabetically).

4. This button will cancel the entire upload process and will delete all information about. So normally the user is able to resume the upload process at any time from any location, but when the user clicked Cancel this is not possible any more because all information is deleted.

5. The button "Save" will save the order of the image files and continue to the next step of the upload process.

1.16.4 4. Step 4: Select books

This step is the last step in the upload process. Here the user will select which images belong to which book title. Sometimes there are more books within one binding and in almost every book there are pages that do not belong to the book itself. For example blank pages in the front, in the back or in between books. These will have no book title and thus belong to the binding rather than a book.



1. A group of controls showing information about a Binding. No fields are editable.
2. A list box showing information about all Books. No fields are editable.
3. The button "Start selecting the first and last page of the currently selected book" will do just that. After selecting a book in the list box of 2 and then clicking this button will enable the user to double click an image file in the list box of 4 twice. The lowest page number will be the first page of the selected book, the highest page number will be the last page of the selected book. So it is possible to double click the same scan twice or double click two different image files. In the latter case all image files whose page numbers are in between of the first and last will also belong to the selected book.
4. A list of all uploaded image files shown in the order saved in the previous upload step. There is just some more information, at least the page number (always starting from 1), the file name and the title of the book it belongs to.
5. This button will cancel the entire upload process and will delete all information about. So normally the user is able to resume the upload process at any time from any location, but when the user clicked Cancel this is not possible any more because all information is deleted.
6. When this "Save" button is clicked the upload process is complete. All information is saved, and as soon as the tile-pyramid-builder is complete the uploaded images and information will be visible to all users in the collaboratory.

1.17 FD - User management

1.17.1 1. Concepts

This page describes the way user management will work. When things are unclear or inconsistent, please contact someone from functional design (or someone who used to belong to functional design).

1.17.2 2. The hierarchy of user roles

There will be four user roles with the following rights:

2.1. Administrator

An administrator will be able to do the following:

- Everything a Moderator can do
- View all users
- Accept/refuse registration requests
- Ban/unban users
- Change user roles
- Change the settings about automatically accept a registration
- Change important settings which impact the whole collaboratory

2.2. Moderator

A moderator will be able to do the following:

- Everything a User can do
- View and revert changes made by users
- See the version history
- Change key book and binding information

2.3. User (registered and accepted and logged-in user)

A user will be able to do the following:

- Everything a Guest can do
- View basic info about registered users
- Add and Edit annotations
- Upload books

- Manage his/her personal bookshelf
- Keep his own notebook and snippets and export that
- Export books

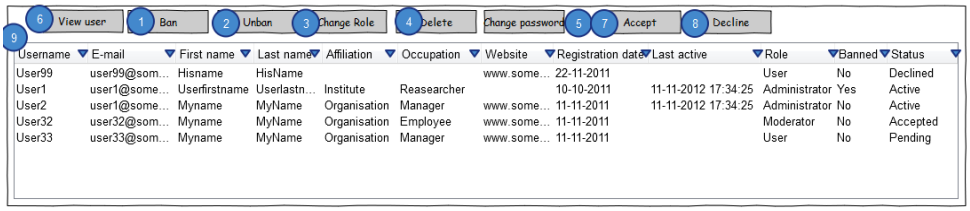
2.4. Guest (unregistered, site visitor)

A guest will be able to do the following:

- View books
- Search books

1.17.3 3. Administrator rights

3.1. View users

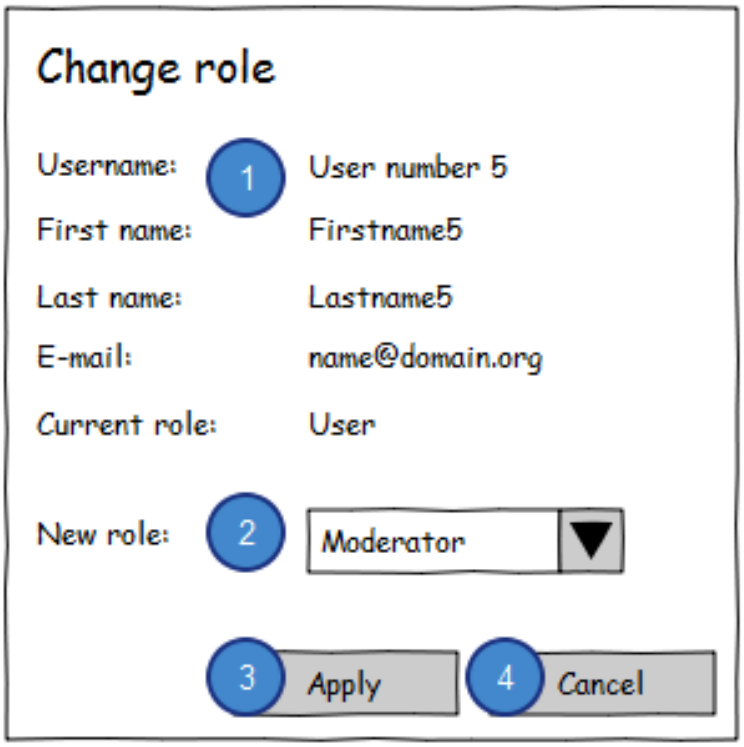


List of controls:

1. Ban button: Clicking will ban the selected user. This means the user cannot login anymore. It is only enabled if the user is not banned yet.
2. Unban button: Clicking will unban the selected user. This means the user can login again. It is only enabled if the user is banned.
3. Change role button: Opens a small window to change the role of the selected user. Only enabled on other users, i.e. a user cannot change its own role.
4. Delete button: Delete a user from the system. This will delete the user, but not its contents. In the contents all information related to the user should be kept intact. This button is only enabled when the selected user is banned. I.e. only a banned user can be deleted.
5. Change password button: Works exactly the same as the Forgot Password button a user can click now. Only difference is that it changes the password for the selected user, and not the current user.
6. View user button: Clicking opens a new tab in which the profile information of the user is shown.
7. Accept button: Clicking will accept the user's registration request. An e-mail will be sent to the user about him being accepted, he can now login. Only enabled when a user's registration is not processed yet.
8. Decline button:Clicking will decline the user's registration request. An e-mail will be sent to the user about him being declined. Only enabled when a user's registration is not processed yet.

9. List of users: these are all the fields of a user. The only thing an administrator should not be able to see is the password, or information related to login. Clicking a record only selects it, nothing else happens. Clicking the arrow next to a header allows the user to sort ascending or descending on that field. Default it is sorted on Registered, showing No first.

3.2. Change role



List of controls:

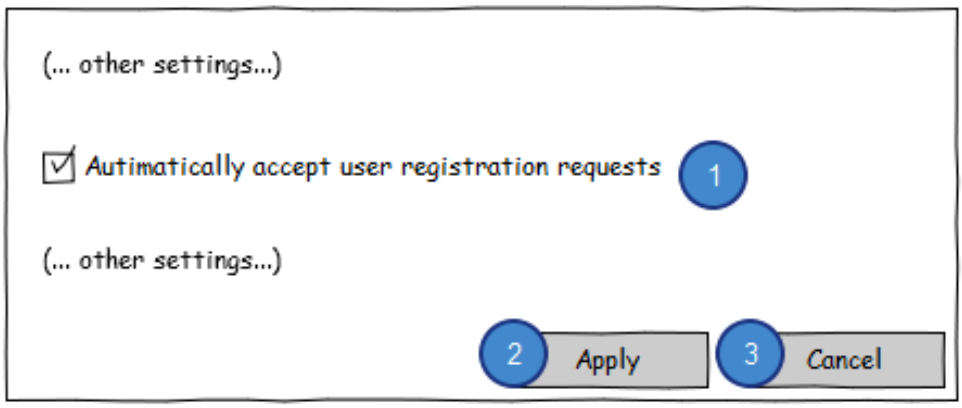
1. Labels show the information (username, first name, last name, e-mail, current role about the selected user.
2. A dropbown combobox to select the new role for this user. The options always are "Administrator", "Moderator", "User". The default selection is the current user role.
3. Apply button: Clicking changes the user's role to the newly selected role (if changed). Next this screen closes and returns to the view users screen, which will update its data.
4. Cancel buttton: Clicking will cancel all changes, close this screen and return to the view users screen.

3.3. View user

This can be just as it is implemented now. It is very straight forward, just show all fields except password.

3.4. Registration setting

In the global layout, in the bar with buttons a button "Admin" will open the screen with settings. There a setting will be added (or it can be a simple pop-up box window):



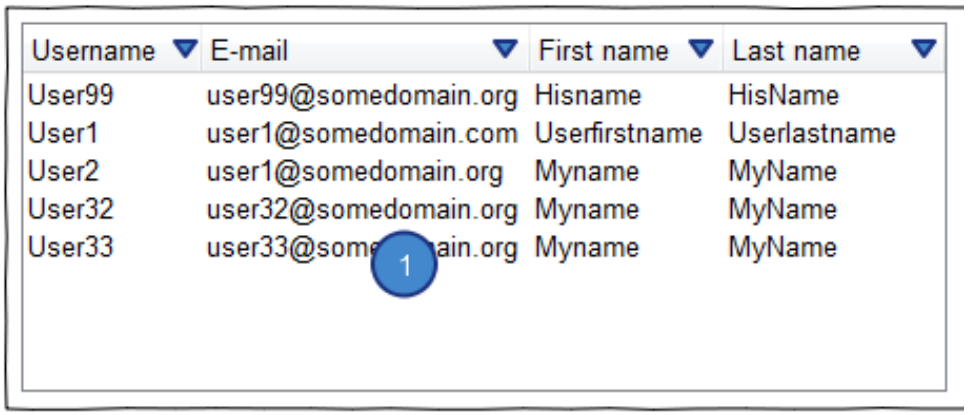
List of controls:

1. Checkbox: Enabling will automatically approve each registration. This means the Administrators will no longer have to accept or decline registration requests. Disabling will not automatically accept a registration and will require an Administrator to accept or decline the registration. In the registration a mail is sent to the user confirming its registration when this field is uchecked and an Administrator clicks Accept, or a mail is immediately sent when this field is checked.
2. Apply button: Clicking will apply the changes and close the screen or tab.
3. Cancel button: Clicking will cancel the changes and close the screen or tab.

1.17.4 4. User rights

4.1. View users

Normal registered users and moderators should also be able to view other users, but just not all information.



List of controls:

1. List of users: just the username, e-mail, first name and last name of all accepted and not banned users.

The sorting works the same as the list in Administrator and there is no other functionality here. The profiles cannot be viewed in a separate window, there are no buttons to do anything. Just viewing and sorting the columns.

1.17.5 5. Other rights

All other rights are integrated into other Functional Designs.

For Moderator in the FD - Managing user generated content

For User in the FD - Notes + Snippets, FD - Bookcase and bookmarks, FD - Uploading, FD - Exporting

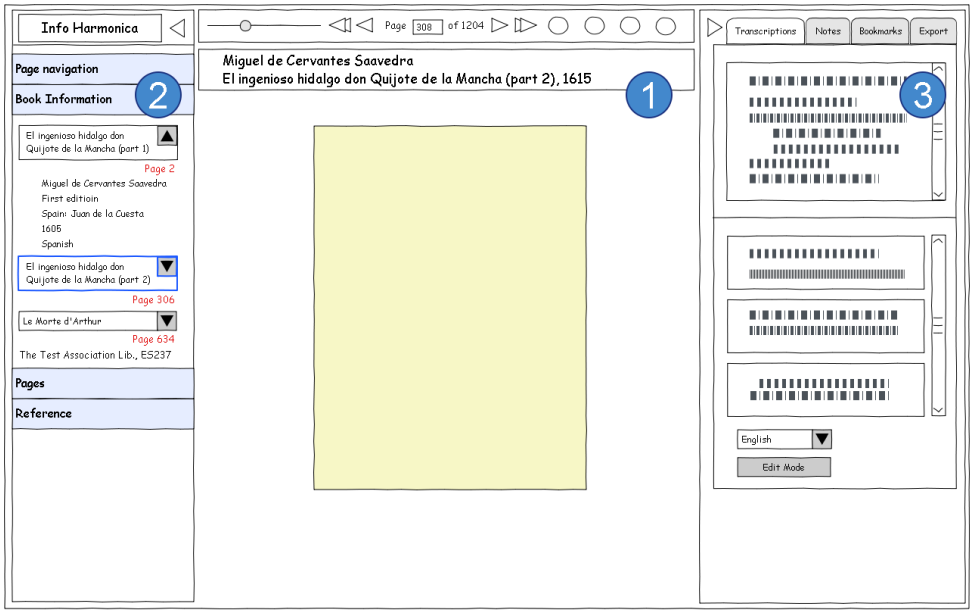
For Guest in the Functional Designs that apply globally (search, viewer, etc.)

1.18 FD - Viewer

1.18.1 Introduction

The Viewer is the core of the application. Most other features give access to or use the Viewer in one way or another. This results in the Viewer being a bundle of functions all packed into an assortment of buttons, tabs and keys. This document attempts to present all this in a structured and clear form. First we will split the Viewer screen in several sections and then go into depth on the functionalities of each of those.

Figure 1:



1. The Viewer Core - Everything that has to do with the Viewer itself.
2. Information Tabs - Functionalities aimed at suppling the user with information and support on the object being worked with.
3. Working Tabs - Here the user can work and preform more complicated tasks.

The Viewer-tab header displays the first 50 characters of the title of the book currently open in that tab.

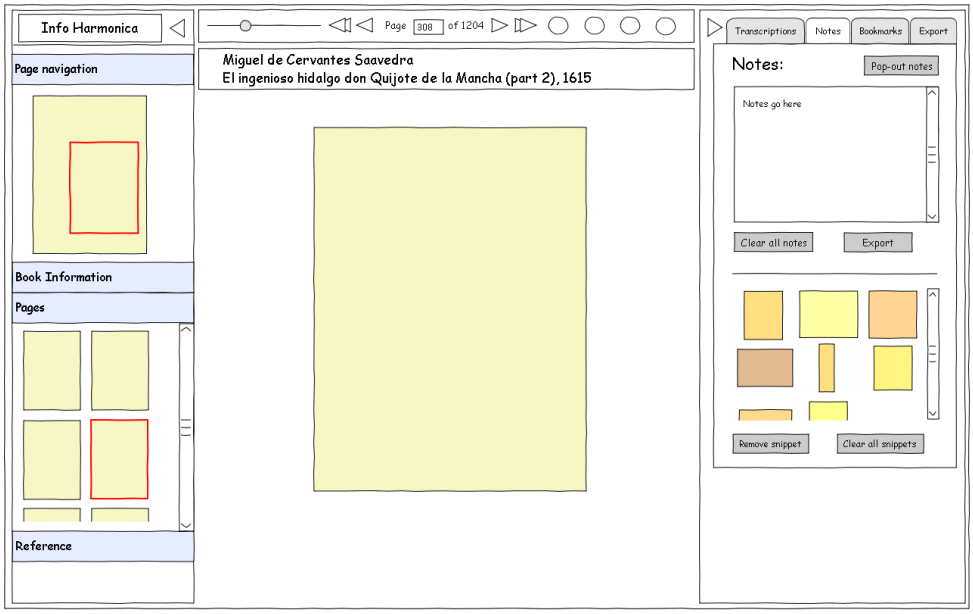
1.19 Information Tabs

The information tabs are on the left side of the side of the screen. It is a vertical stack of tabs, where multiple tabs can be open at once, as seen in figure 1. Available space should be divided

between the open tabs, taking into consideration the minimum needed space for a tab and some tabs having a limited need for space. For instance, the Page navigation needs enough space to show the thumbnail of a page, no more, no less.

If the user has more tabs open then can be displayed on the screen, a scroll bar must be added. Of course, one of the goals of the accordion is to eliminate the need for a list long scroll bar. However, our harmonica gives the user the tools to choose the information which is displayed on the screen, instead of letting the system regulate it.

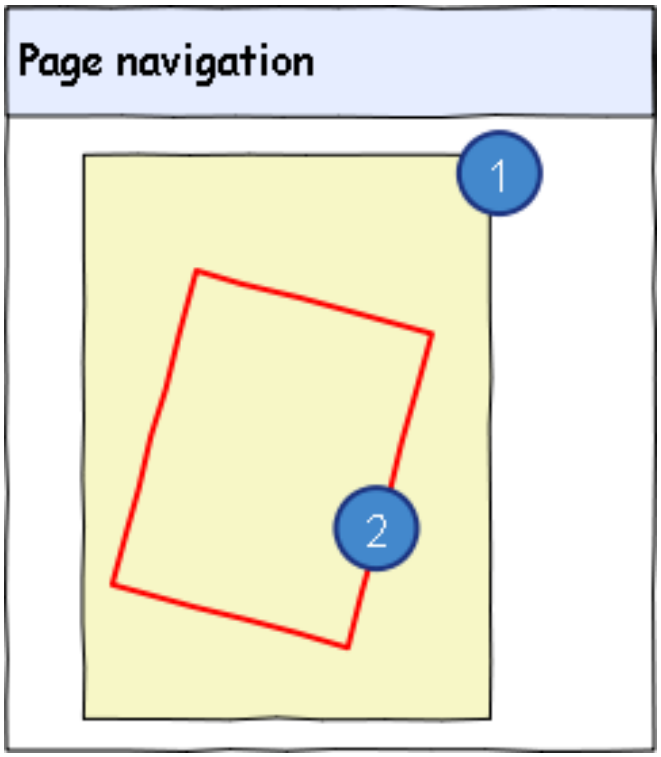
Figure 1:



Behind every one of the tabs of the harmonica lies a different feature with its own layout and functions. The following section gives a detailed description of each of these.

Page navigation The page navigation shows the user what part of the page they are viewing. This is especially useful when they are zoomed in and can not see the page layout in the Viewer Core.

Figure 2:



1. A thumbnail of the page currently being viewed
2. A frame highlighting the part of the page that the user can see. If the entire page is visible, the frame follows the border of the thumbnail. If the user has zoomed in, the part of the page that is within the user's view is framed. If the page is rotated, the frame is equally rotated to match with the view of the user.

The thumbnail is always displayed in the left-hand corner; its position does not change with widening or narrowing of the tab.

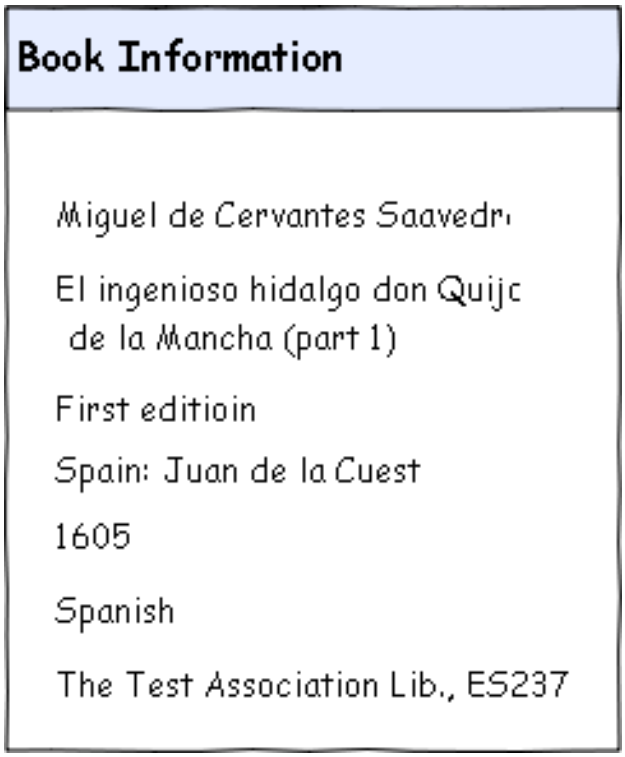
Book information The book information shows details about the book and its contents. If there is only one book in the Binding, the information is displayed in the MLA referencing style, which is commonly used in the fields of arts and philosophy:

- author(s) or editor(s)
- the complete title
- edition, if indicated
- place of publication
- the name of the publisher
- date of publication
- medium of publication

An example: Nabokov, Vladimir. *Lolita*. New York: Putnam, 1955. Print.

We will not name the medium, as this is not attached to a Binding or Book.

Figure 3:



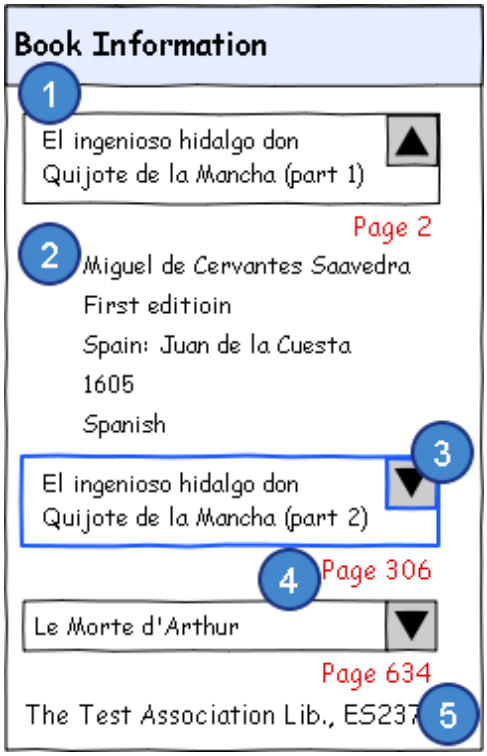
Note: below the standard MLA format there is also the library name and signature.

When a attribute has more text then can be displayed in the harmonica's width, it continues on the next line, as can been seen with the title in Figure 3. When the harmonica is made wider, the attributes are not concatenated.

Note, in the future, the to be displayed information may be extended. At the moment the only information from the Binding that is used is the library and signature.

If there are several books, the information is stored in drop-down boxes.

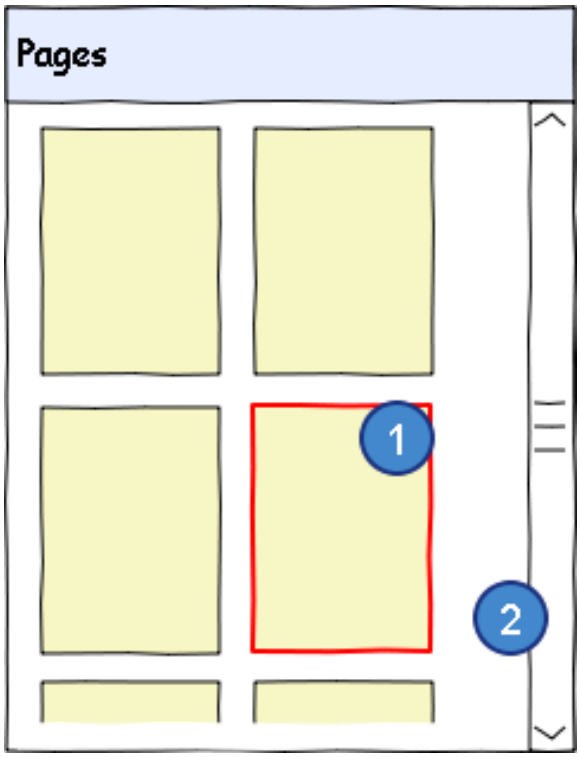
Figure 4:



1. Drop-down box with title, in dropped down state.
2. The details concerning this book. Notice how, in MLA, the author(s) comes first, followed by the title, and how here these have been switched. The title of the book in the drop-down box is clearer than the author(s). Also note that only information concerning this particular book, not the Binding, is displayed in the dropped down box.
3. To indicate which book is currently being viewed, the border of the box containing that book's title is turned a different color. The final color depends on the graphical design, but has been set at blue for the moment.
4. Below the title's box, there is the page number on which this book starts. If the user clicks on it, the page with this number is loaded in the Viewer.
5. Below the list of titles contained in the Binding, is the Binding information. The library and signature.

Pages This tab offers a visual way of navigating the entire Binding. It contains thumbnails of all pages and can be used to jump to a different page.

Figure 5:



1. The page that is currently being viewed by the user has a red border around it to make it recognizeble for the user.
2. When the list gets to long to display within the tab, a scrollbar is added to the side.

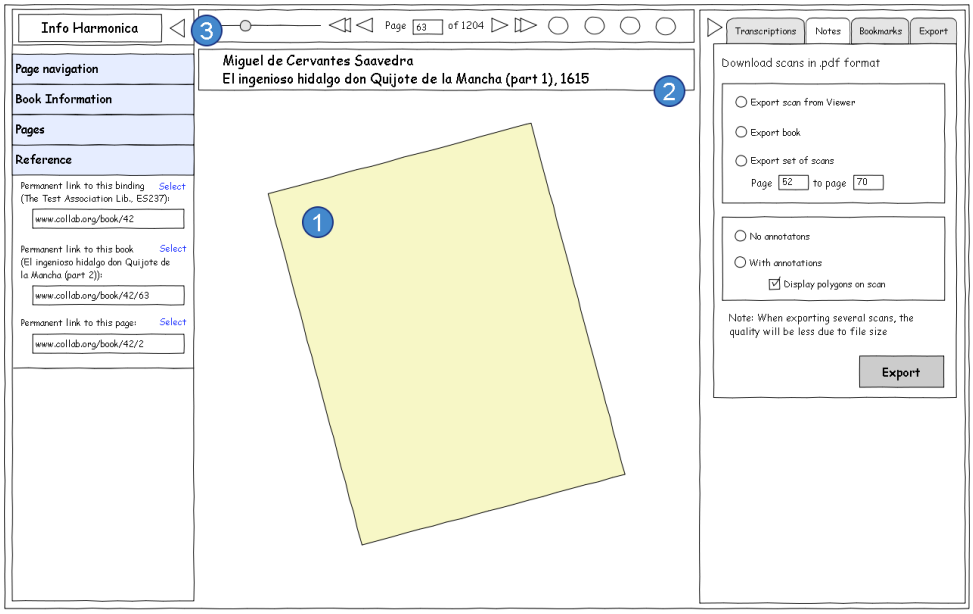
It may seem appealing to make the thumbnails small, to put a lot in the little available room. However, they should be distinguishable from each other. Details are not important, but the ability to see if there is a picture on the page, seems like a reasonable demand.When the harmonica is made wide enough to allow for another column of pages, it is added. This way, by making the harmonica wider, the user can get more overview of the book.

Reference The reference tab supplies direct links to the Binding, the book and the page being viewed. FD - References gives the specifications of this tab.

1.20 The Viewer Core

The Viewer core is the centre of the Viewer tab. It is the working area where the page is being displayed and can be manipulated. It also contains an information bar and a buttons bar.

Figure 1:



1. The page being viewed
2. Information bar - Gives general information about the book
3. Buttons bar - Contains buttons and other interfaces for activating most of the features

Aside from the buttons bar the user can also interact with the application through the keyboard and mouse.

Terms Throughout the definitions of the Viewer Core functions, certain terms are used to explain the state of the system.

Zoom level

The intensity of the zoom. A very high zoom level means the page is very much zoomed in on, a low zoom level means the page is very much zoomed out from.

Rotation

The degree at which the page is rotated. Zero rotation has the page standing upright. An example of a rotated page can be seen in Figure 1.

View

The part of the page that is visible to the user. The view is constrained by the borders of the Viewer Core. The view is changed by zooming, rotating and moving over the page.

Focus point

The focus point is the centre of the view.

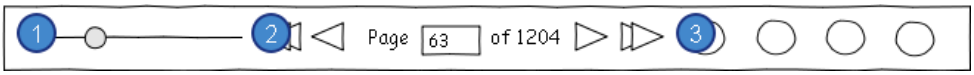
Standard view

The standard view is the zoom level and rotation that the page is displayed in when it is first loaded. The whole page is visible for the user, but the exact zoom level depends on the available space in the Viewer Core. The focus point is the centre of the page.

Information bar The information bar at the top of the page indicates what book is being viewed at the moment. The author(s), title and publishing year are displayed on the bar. It differs from the book information tab in that it only gives general information, instead of all details.

Buttons bar The buttons bar can be grouped into three kinds of input:

Figure 2:



1. Scale bar - used to adjust the zoom of the page
2. Page navigation buttons - used to go to other pages within the Binding.
3. Other buttons

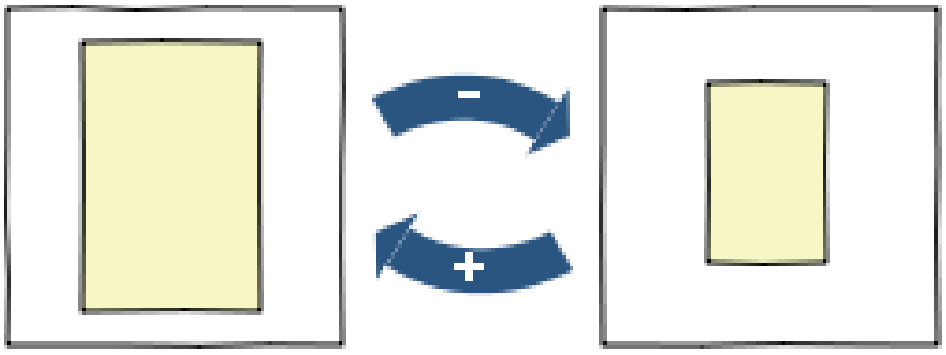
Each of these parts contains a set of functions.

Zoom in and out with slider When the page is first loaded, it sits in a standard zoom, the whole page is visible on the screen. To get a better view of the page and parts the user is interested in, the user can zoom in and out. One way to activate this is through the zoom slider, point 1 in Figure 2.

The user can operate the slider by clicking and holding the thumb and moving it to the right to zoom in and the left to zoom out. As the slider is altered, the image is constantly updated to the new zoom level.

When zooming in and out with the slider, the focus point does not change.

Figure 3:

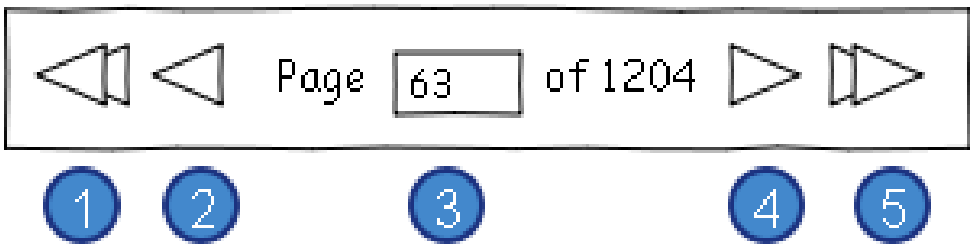


Page navigation buttons A Binding contains Books and loose pages. These loose pages are pages that don't belong to a single book, like blanco sheets at the end of a binding for notes. All of these combined give a virtual representation of the real world book.

The scans are arranged by the location their page has in the actual book. So the next scan in line, would be the following page, and the previous scan, the previous page. If the last page of a book is reached, the first page of the following book is displayed. By doing this, the virtual Binding is read in the same sequence as the real book would.

The page navigation buttons can be used to move through the Binding. Every time a new page is loaded into the Viewer, it is displayed in the standard view.

Figure 4:



1. First page button

Loads the first page in the Binding into the Viewer.

This function is unavailable if the first page is already being displayed in the Viewer; the button is then greyed out and unresponsive.

2. Previous page button

The previous page is loaded into the Viewer. Previous here does not mean the previously viewed page, but the page that sits before the currently viewed page. So if the Binding contains pages 1 to 10 and 99 to 156 of an actual book, which would translate to 68 virtual pages in the system, the previous page of book page 99, page 11 in the system, is page 10.

For this function to work there has to be a previous page. Non-existant scans cannot be shown. If there is no previous page, the button should be greyed out and unresponsive.

3. 'Jump to page' textfield

This field displays the page number of the currently being viewed page. It can also be used to jump to a certain page. The number of the new page should be entered into the textfield followed by pressing the enter key. If the entered page number is within the Binding, the new page is loaded into the Viewer. If the supplied number is out of bounds, or the page is not contained within the Viewer, no new page should be loaded and the current page number should again be displayed within the textfield.

When the user navigates to another page, in whichever way, the new page number should be displayed in the textfield.

4. Next page button

The next page is loaded into the Viewer.

Again, this function is only available if there is a next page; when on the last page of the Binding the button is greyed out.

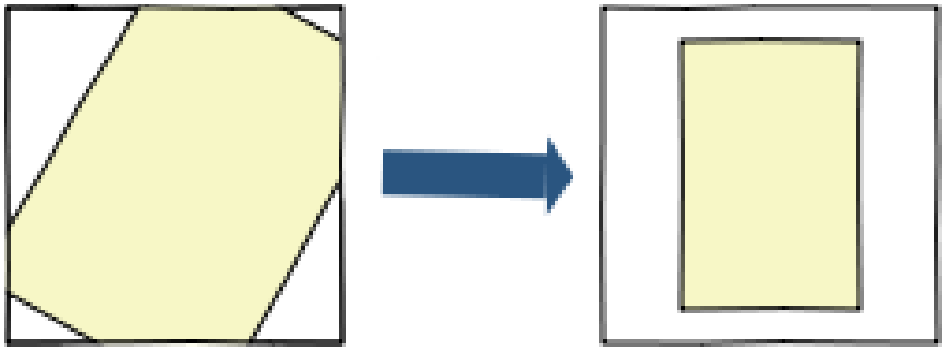
5. Last page button

The last page of the Binding is loaded into the Viewer.

The button is greyed out when the last page is already being displayed in the Viewer.

Standard view button The 'standard view button' resets the view to standard view. The zoom level, rotation and focus point are updated and the image is updated.

Figure 5:



Rotate button For quick rotating, and rotation when using a tabloit, there are the rotate buttons:

- Left rotate button

Rotates the page 45 degrees around the focus point to the left.

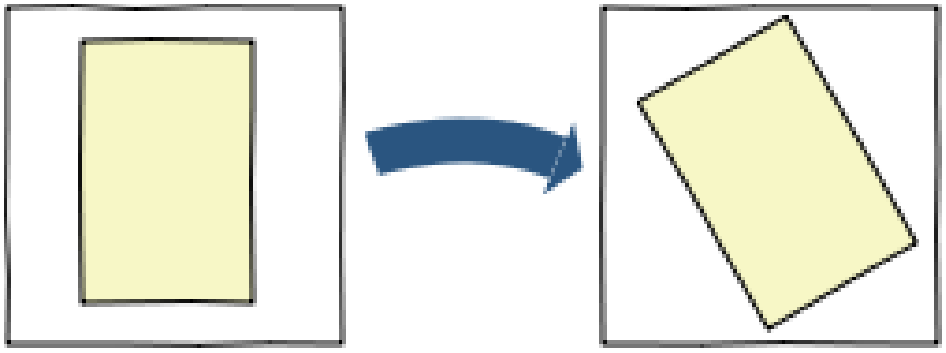
- Right rotate button

The same as 'left rotate button', only to the right.

The reason it rotates a set degrees in a set distance, is to keep it clear and easy. To much functionality can get confusing and results in not using the function. Two buttons is already a little confounding, but it would be greatly annoying to go one rotation to far, and then have to do a whole procedure to turn 45 degrees back.

When rotating, the zoom level and focus point remain unchanged.

Figure 6:



Transcriptions There are also functions for creating and editing transcription markers. They are clarified in FD Annotations.

Keyboard and mouse controls There is also a number of functions that can be applied with the mouse and keyboard.

Dragging The view of the page can be dragged, moving parts of the page in the drag direction into the view and opposite parts outside of the view. When the view reaches a border of the page, the image can not be dragged further into that direction.

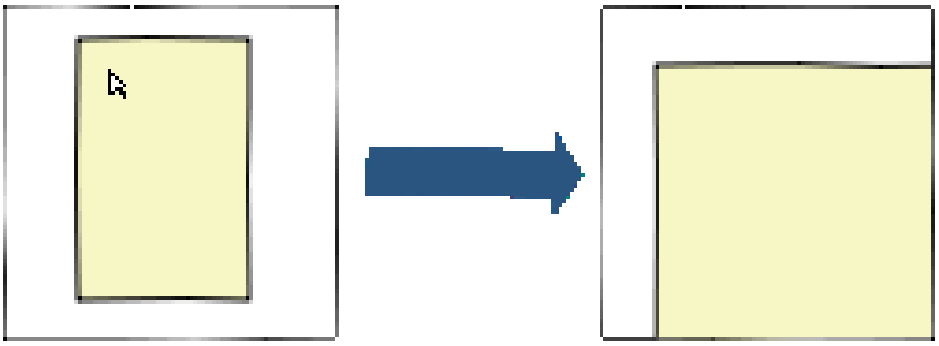
Dragging is done by clicking and holding a point on the page in the view and then dragging the mouse into the direction the view should move. The point that was 'grabbed' by the mouse, remains underneath the mouse while dragging, moving the page along with it.

Zooming There are two ways to zoom with the mouse; scrolling and double clicking.

For zooming with the scroll wheel, the mouse can be placed anywhere in the Viewer Core, except for the Information and Buttons bars, and scrolled with. Scrolling down will decrease the zoom level and scrolling up will increase the zoom level.

Double clicking can only be used for zooming in. When the user double clicks within the view, the zoom level is increased and the focus point is changed to the point the user double clicked on. See Figure 7.

Figure 7:



After each zoom level change, the image is updated.

Rotating Crude rotation can be executed with the rotate buttons, but fine rotation with the mouse is also possible.

For this, the rotator tool should be selected by pressing and holding the spacebar. The mouse icon will change into a turning arrow, something like shown in Figure 8, while the rotator tool is selected. With the space bar pressed, a point in the view is grabbed and dragged. By dragging to the left, the page is rotated to the left, and by dragging to the right, the page is rotated to the right. The further the mouse is dragged, the further the rotation. During the rotation process, the image is updated every time the rotation is changed.

Figure 8:

If the mouse is released or the space bar lifted, the rotation ends. If the space bar is released, thus automatically unselecting the rotator tool and selecting the dragging hand, the page is dragged when the mouse is moved.

Keyboard Aside from the spacebar, there are a few other functions activated by keys:

- Numpad 8

Moves the view up. This is like dragging, in that the page rotation and zoom level do not change and that the view can not be moved beyond the page border.

By tapping the key, the view is moved up a bit. When the key is held down, the view will move up continually the speed slightly accelerating at the start of the moving. When the key is released, or the page border is reached, the moving stops.

- Numpad 2

Moves the view down, in the same way as numpad 8 does for up.

- Numpad 4

Moves the view to the left.

- Numpad 6

Moves the view to the right.

Some of the functions with a button on the buttons bar, also have a key on the keyboard for quick use.

- Home - sets the page back to standard view. Does the same as 'standard view button'
- Numpad '+' - zooms in. Does the same as the slider moving right.
- Numpad '-' - zooms out. Does the same as the slider moving left.

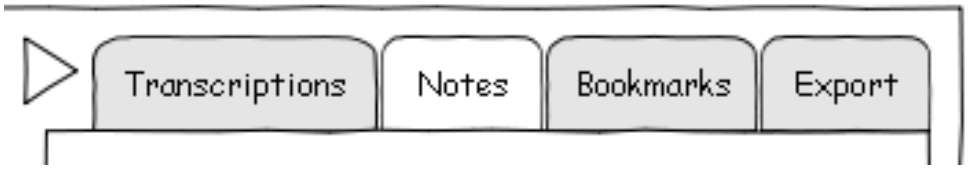
1.21 Working Tabs

1.21.1 Working Tabs

The working tabs are located to the right-hand side of the window. The user can request the system to preform more distinct functions and manage private and public content.

There are four tabs:

Figure 1:



- Transcriptions - Read transcriptions and enter the edit mode where transcriptions can be added to annotations
- Notes - Manage personal notes and snippets
- Bookmarks - Manage personal bookcase
- and Export - Make a .pdf of (part of) the book being viewed

Each has their own functional design, reachable through the above links.

Chapter 2

Research

2.1 Research

2.1.1 Index

2.2 General overview of our research

2.2.1 1. Requirements

Requirement	Why
The collaboratory needs to have a slick, attractive and user friendly user interface.	Our target audience consists of a select group of conservative people who are not very fond of computers. An interface as described above might convince them to use the collaboratory.
The collaboratory is usable using a web browser.	We want the collaboratory to be accessible from all over the world. And we want the system to be as accessible as possible, so using a web browser lets the user be free in his choose of platform and applications.
The collaboratory enables users to upload digital scans of books and writings.	Most of the books and writings will be scanned. These high resolution images must be uploaded to the collaboratory to enable other users to view them.
The collaboratory enables users to add and modify XML (TEI) to the digital scans.	This TEI XML is used for exchanging the books with all related information, so the used must be able to add, modify and export the XML data. This XML can also be used for advanced searching within the books, but this is more of a technical advantage.
The collaboratory and other required software run on a server that will be able to handle the amount expected amount of visitors and digital scans.	All the software will have a dedicated server. Obviously it should be able to handle the amount of users, and the amount of data it will contain.

The whole system is scalable.	If the number of users or the amount of data exceeds the capacities of the server, it should be able to modify the hardware without modifying the software to handle the number of users or the amount data.
The collaboratory enables users to view books and writings (digital scans and XML representation).	Perhaps the most important requirement. Users should be able to view the data stored in the collaboratory to conduct research.
The viewer enables user to zoom in on scans.	Some of the annotations are quite small, or hard to read. By zooming in it becomes much easier to read the text and to conduct research.
The collaboratory enables users to add notes to the writings.	These notes may contain personal comment, of comments regarding a research. Goal of these notes is to start a discussion to gain more knowledge about the writings themselves.
The collaboratory enables users to view all notes on writings.	In order to start discussions, it is obvious other user must be able to view all the notes.
The collaboratory's user interface is catching, clear, consistent and user-friendly.	We want the collaboratory to attract the user to work with it. Of course it should contain the right functionality, but we need that attractive touch to catch the user into using the collaboratory.
The collaboratory's interface replicates the historical sense.	Because the books and writings are quite old, and the target audience are generally not that fond of futuristic appearances of the collaboratory, so we need to close the gap between their perception and the technology.
The collaboratory enables meta-data to be stored with the annotations and notes.	Sometimes there is information that is not known for books or writings in general, but the collaboratory must enable to user to register all kinds of information.
The collaboratory enables the user to search books, and to search within books.	Using a broad range of optional filters, users must be able to quickly find exactly what they need.
The collaboratory contains means to investigate the evolution of text and writings.	Somehow relationships and cross-links must be registered to be able to investigate the readers and the evolution of the ideas of that time.
The collaboratory enables users to manually convert the scans to (XML) text.	In order to be able to search within books we need the XML text. So the collaboratory should enable the user to manually insert the text that is written.
The collaboratory enables administrators to manage the system and other users.	The system has to be maintained by administrators, so they need to be able to monitor and modify data and users.
The collaboratory enables users to view two books or writings at the same time.	Sometimes there are two copies of the same book, but the annotations will be different. To investigate this difference the two copies must be viewed at the same screen. This is just an example, users should be able to view any two books next to each other.

The collaboratory enables users to discuss the annotations and notes.	This could be like a strongly integrated forum where users can contribute to other people's research and exchange ideas about the books, annotations and notes.
The collaboratory enables users to print the scans. <J>	Printed scans are useful for handing out to students or for the User's own research
The collaboratory enables users to export the scans to pdf. <J>	When one is working offline one can still read the documents.

2.2.2 2. The project

2.1 Our target audience

The main target audience is a group of about 500 researches world-wide who investigate the evolution of annotations and ideas in the period 1490-1700. In that time printing was invented, so suddenly people had an all new way of accessing an enormous amount of information. Much like the introduction of the world-wide-web past century. Besides this group of researchers, we expect students and other booklovers to use the collaboratory. We expect the researchers to intensively use the collaboratory to add and view books, writings, notes and researches. The other groups will be more likely to view all the data stored in the collaboratory.

2.2 Goals of the project

- Create a digital library of annotated books which everyone can see and contribute to.
- Preserve information about the physical aspects and the contents of these books while the original texts deteriorate.

2.3 Background information

When the printing press was invented in 1490, people were overloaded with information because of the many, many books. While trying to cope with this information they turned to annotating their books. Annotating became a widely spread custom and people were even taught in schools how to effectively annotate their books. One could simply not be called a scientist if one did not annotate his books. Those annotations could vary from just underlining some words, to writing comments at the blank spaces. So all these annotations are important to understand how people coped with this new information.

2.4 Examples of research goals

- How did information spread?
- How did people interpret texts?

2.2.3 3 Acquired information

3.1 Users

We probably want the following roles (further research might be necessary):

The following permissions are the default permissions. We might want give or take away permissions in the future.
An administrator can:

- Manage users
- Do everything a moderator can do

A moderator/editor can:

- Accept new uploaded books
- Moderate notes/annotations
- Approve of changes in notes/annotations
- Approve a user's registration request
- Do everything an administrator can

A user can:

- Correct annotations. The corrections need to be approved of.

An anonymous user can:

- Register
- View books and notes/annotations

The system might keep track of the following information about a user:

- Name
- Address
- Password
- Email address
- Research area/interests
- Website (which contains the user's publications)
- Affiliation (University/Research institute)

Notes: Too much mandatory information might dissuade users from using the system. Too little information, on the other hand, will make checking the credibility of a user's contributions difficult.

3.2 The books

The documents which will be uploaded are usually books and sometimes just a few pages. A book consists of pages, optionally the cover of the book. The scans will be numbered. This numbering may differ from the page numbers and might not even use numbers.

The following information can be attached to a book:

- Author
- Title
- Year
- Owner

It is possible that some of the information is unknown.

Flexible fields might be an option.

3.3 Uploading books

When an error occurs during uploading a, we have a number of options at our disposal:

- Do nothing and let the user upload the book again
- Allow users to continue unfinished uploads
- A continue button in the uploading screen. User can only continue while the browser is still open

3.4 The viewer

Panning:

The customer would like to be able to pan an image.

Zooming in:

The customer wants to have clear icons for zooming in and out. Scrolling might also be an option for zooming. The ability to see on which part of the page you are zoomed in, might also come in handy.

Viewing pages:

Viewing 2 pages next to each needs to be an option. (The pages may be from the same book or from different books).

Navigating through a book:

One needs to be able to flip one or two pages at the same time. There must also be an option to get an overview of all the pages in a book. A sidebar or an underbar with an overview can also come in handy.

Multiple books:

The ability to quickly switch between different books would be nice.

Searching for books:

One needs to be able to sort and search books using the data under the header books. The search functionality needs to be very extensive and contain Boolean operators such as AND, OR, NOT, etc.

Note: Our customer has indicated that he would like the books to be presented in a bookshelf.

(cf. IPad book store). A normal list presentation should also be possible.

Annotations/Notes

Note: The customer might want a wikipedia like system containing the possibility to evaluate contributions and the ability to keep track of changes in the collaborator.

3.5 Other

The customer has shown interest in the ability to print certain parts of books.

It might be interesting if we could design an annotation detector in order to quickly find which pages are annotated. We could also let users mark annotated pages themselves. Navigating between annotated pages might be useful.

2.2.4 4. Practical information

4.1 List of books/articles concerning annotated books

Used Books Marking Readers in Renaissance England, William H. Sherman

“Studied for action”: How Gabriel Harvey read his Livy, Lisa Jardine and Anthony Grafton

Jumping through the computer screen, Anthony Grafton

"Reading Strategies for Coping with Information Overload, ca. 1550-1700," *Journal of the History of Ideas* 64 (2003), pp. 11–28. Ann Blair

2.3 Research about books

2.3.1 1. Requirements

1. A book needs to have enough information in order to be identified.

2. It needs to be possible to add, edit or remove the attributes of a book. This must not cause a conflict with requirement 1.

2.3.2 2. Research

2.1 Background information

Books that are added to the system, should come with metadata. E.g. title or author. This information will be used to identify the books in the system. It’s main function will be to allow users to identify the books.

This would be a very easy task when you look at today’s books. One can easily add title, author, publication year, etc. For medievil books however, this information might not be available that easily. The exact year of publication is often unknown and the author’s name might not be written in the book itself. Hell, the book might even be a collection of parchment unearthed in a cave in the middle of nowhere. What would you list as a publication date in that case?

We will need to come up with a flexible metadatasystem inorder to cope with the shortage of information. The metadata should not become a hindrance when one wants to add a book, but it should provide enough information to identify the book. We could make entering the title mandatory for instance, but entering the author’s name optional.

2.2 The definition of a book

A book consists of a cover, a binding, in which multiple books are contained, which we will call writings from now on. These writings usually have no connection at all. The only reason they are bound together is because they were bought at the same time. When catalogueing these writings they should be described separately. The binding is the element which encompasses the writings.

For definitions see:

- Binding
  - Book

2.3 Answers to questions asked to Arnoud and Els:

Kan het ook wel eens nuttig zijn om meerdere titels aan één boek te hangen? (Bijvoorbeeld de engelse en de latijnse titel. Soms heeft een boek wel eens meerdere titels)

ik geloof niet dat dit snel kan voorkomen; een dubbeltalige titel zou in principe in hetzelfde titelveld moeten worden geregistreerd, zolang het om een beschrijving van dezelfde titelpagina gaat.

Dezelfde vraag bij auteurs. Komt het voor dat men met meerdere personen aan één boek schreef? Gebruikte men vaak pseudoniemen? Kortom hangen er meerdere auteursnamen aan één boek?

je hebt gelijk: dit is wel mogelijk, bv. als er meedere auteurs aan een werk bijdroegen, of als je naast de auteur een redacteur (of editor) hebt. Het zou dus fijn zijn als er meerdere auteursnamen aan 1 titel kunnen worden toegevoegd. Het voert waarschijnlijk te ver deze sub-categorieren van co-auteurs danwel editores uitsplitsen in apart gelabelde velden – om allerlei bibliografische voetangels en klemmen te vermijden.

2.4 Research about editing and viewing annotations

2.4.1 1. Requirements

Requirement	Why
It should be clear which transcription belongs to which annotation	Otherwise a User has to figure it out himself by reading the annotations, which partly defeats the purpose of the transcriptions.
It must be possible to create a transcription for every annotation. A User who is viewing the scan containing the annotation must be able to see this transcription.	To prevent the Users from having to decipher the annotations every time they read them. For the notations can impossible to decipher without the proper knowledge. It also saves the Users a lot of time.
The transcriptions should not obstruct a User while that User is viewing a scan.	The transcriptions will distract the User from the original scan. And if they block parts of the original scan then those parts of the scan can't be accessed by the User.
It must be possible to enter and show a translation to English for each transcription.	Not all Users can read all languages. Most Users can read English.

It should be possible to support multiple translations later on	Not all Users can read English.

2.4.2 2. Research

De onderstaande afbeeldingen komen uit William H. Sherman. Used Books: Marking Readers in Renaissance England. Philadelphia: University of Pennsylvania Philadelphia, 2008.

Types of annotation:

- Text. Most text is rectangular and horizontal or vertical. But text can be in all sorts of shapes. A text can be written around something for example.
- Symbols. Most annotators use all kinds of symbols. The picture below is an example of a hand symbol. Arnoud has indicated a database for symbols might be desireable. So Users can easily document and access the meaning behind each symbol.
- Other kind of annotations exist. The picture below shows a computation. It might be difficult to create a transcription of such an annotation.

Possible problems:

- Annotations can overlap. If a User selects annotations by clicking on them it is unclear which annotation will be selected when the User clicks their overlapping part.
- Annotations vary in size. Solutions which work well for large notations might not be effective for small notations. An example of a very small annotation:
- A whole page can be annotated. If transcriptions should be created for all annotations on such a page we will have a lot of transcription text.
- It can be difficult to differentiate between different annotations on the same page. Often the annotations are not written in neat blocks of text. In the picture below the annotator has added to the index.

2.4.3 3. Possible solutions

3.1 Footnotes/Pins

This approach will emulate the idea of footnotes. The User will be able to place a pin (linked to a transcription) on the scan. When the User wants to know which transcription belongs to which annotation he can click on the annotation which will cause the pin to appear in the scan. The User will then need to examine the area around the pin until he is able to identify the corresponding area.

An enormous advantage of this approach is that it's very user-friendly, but it certainly has a lot of issues:

Identifying which transcription belongs to a certain annotation causes a certain number of problems. A lot of annotations might be centered around a certain point and we do not want the screen to become too cluttered with pins. When the screen is cluttered, it will be more difficult to select the right pin, see which pin belongs to which annotation and it will make navigating the screen annoying.

In order to keep editing simple, we will need to keep the size of each pin fixed.The size of the pins may have some consequences. When they are too big, a lot of annotations will be completely covered by pins. When they are too small, selecting pins will become a chore. Whichever size we will choose, the possibility of pins covering annotations will remain.

A few suggestions to alleviate problems:

- Make the pins transparent. Navigating will become less annoying because the User is able to see the text underneath the pins.
- Number the pins (Or use colour coding) and make it easier to change between overlapping pins (e.g. by clicking arrows). The User will be able to find the correct pin by changing until he sees the transcription he wants to see.
- The ability to turn pins on and off.
- Change the size of the pins when zooming in and out.
- A couple of different pin sizes.
- Only show pins in the vicinity of the mouse pointer.

3.2 Encircling

The idea of encircling is as follows: One draws a line around the area where the annotation is located and the inside of the area will be linked to the transcription the User chooses. The concept is exactly the same as the lasso tool used in a lot of graphics editing programs. By clicking on the transcription, the corresponding area will light up and vice versa. The advantage of this approach is that it will support nearly every kind of annotation.

The border selection could occur in a number of ways:

- Rectangles: The only kind of borders the User will be able to draw are rectangles.
- Polygons: The User will be able to create a polygon by determining the locations of the vertices of the polygon.
- Free hand: The User can draw the border freely. This may not sound user-friendly at first, but it might be a nice feature for people who work with iPads and graphical tablets.

Encircling also has a lot of disadvantages:

Working with rectangles is too limited. Although there certainly will be lots of annotations which can be surrounded by a rectangle, exceptions will occur and quite often.

Working with polygons and freehand might discourage Users from using the tool because it can cause a lot of minimanaging. Users might also try to use the tools in unpredictable ways e.g. Using a lot of points to encircle annotations in order to make the border more "round" or crossing the edge of the already existing border.

Some annotations might completely surround other annotations. In this case the encircling can not correctly capture the area.

Annotations can overlap. In this case we will need to choose one of the two areas. This is a minor issue because Users are not prone to click on overlapping annotations, but it may cause some frustrations nonetheless.

Some variations:

- Allow linking multiple regions to one annotation. This will partially solve the "Surrounding Annotation problem".

- Make it possible to add and remove parts of the selected area. This means that we can also make holes in the selected areas, solving the "Surrounding Annotation problem", but it might be confusing for Users.

Coupling regions to transcriptions brings along a couple of other issues and these will be addressed in a later section.

Another addition would be to offer the User multiple ways to select the area. The most obvious one is a simple rectangle. But a polygon or even a free-hand line will be better in some situations.

3.3 Marking

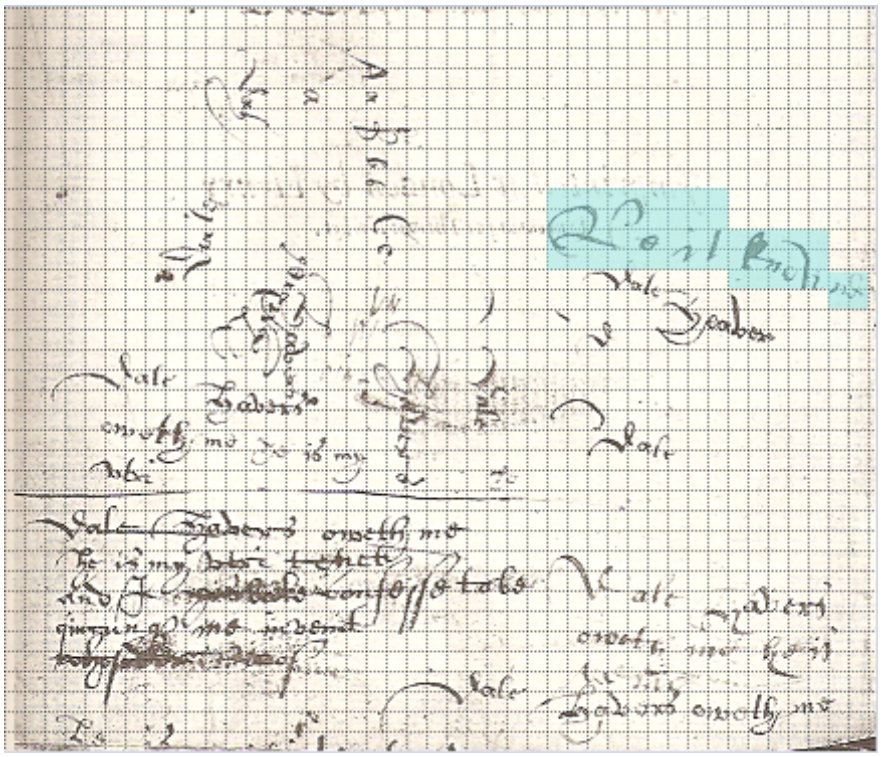
Based on the concept of marking important parts of text in a book. The User will be able to use some kind of marker to paint the area in which the annotation is located. The painted area will henceforth be linked to the chosen transcription.

The simplest form of marking is identical to a simplified painting tool, which is common in a lot of graphics editing programs. An advantage of marking is that the User will have a lot of control: He can exactly determine which area belongs to which annotation. A disadvantage is, that it is not easy to use at all. Most Users will be driven away because they can not use the tool correctly.

But there are ways to simplify marking: One possibility is that when the User has marked the area he wants, we use a closing to determine the area which will respond to the transcription. This way a User does not have to mind small holes in his marking. This concept emulates how humans determine which part of a text is marked with a marker. A disadvantage is the loss of control. A User will get frustrated because he can not control which parts of the text will be marked.

Another possibility is to put a grid on top of the image and let Users "mark" the various tiles of the grid. This option is rather user-friendly, but not very precise. In order to increase precision, the User should be able to change the size of the grid. This solution is probably the most attractive form of marking.

Building upon the ideas of grid based marking, we could give the User more options to edit the grid. Moving points and the ability to freely add new tiles are possible extra features.



This grid is rather fine and will be able to capture all details. Usually a coarser grid will suffice.

Another possibility is that the User uses the grid discussed above and when showing (instead of editing) the system just shows the figuration of the selected area. This will look like a kind of free-hand line, but it will be accurate. The only issues we have here is what if the area selected contains 'holes', or consists of multiple seperate areas?

3.4 Issues with area based viewing

Encircling and marking are both area based. The transcriptions are linked to an area the User has chosen. We need to be able to determine which transcription belongs to an area. The obvious solution would be to let the area light up during a mouseover. But this might be annoying for the User and might also create some problems for iPads and other tablets.

We probably do not want to see these areas, while we are panning and zooming.

A mode in which we can see all of the areas can also come in handy.

3.5 Miscellaneous

Voronoi diagrams could also be useful in our design. As of yet we have not developed a technique which sucessfully incorporates Voronoi diagrams. The idea remains interesting nonetheless.

We could add a preview of the location to the transcriptions.

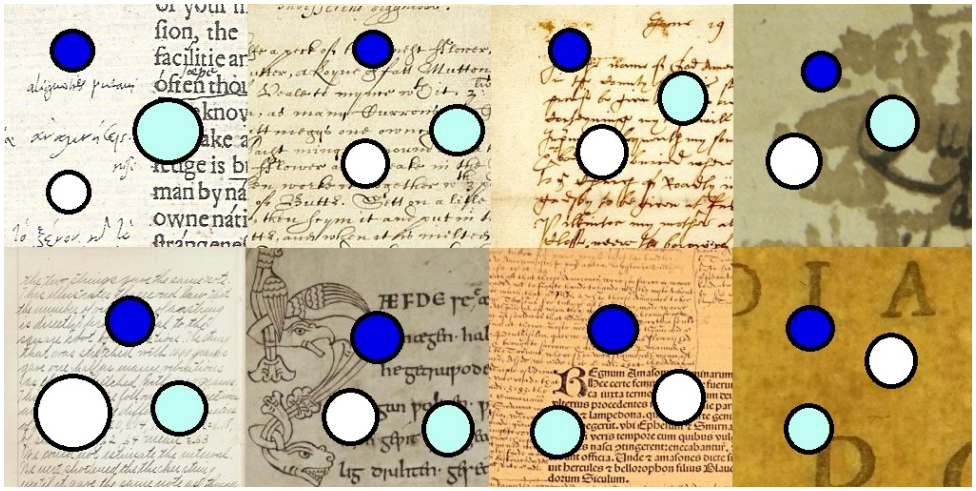
2.4.4 4. Our choice

After discussing our research with our clients and our technical staff, we concluded that our clients preferred the marking principle, but that this would take too much time to implement. Because of this, our clients chose the encircling principle based on polygons.

4.1 The colour of the vertices

In order to make working with polygons pleasant for the user, it is important to choose the colour of the vertices and the edges. One could argue that this is not a task for functional design, but because the choice of the colours has a huge impact on the functionality, we decided to make a temporary decision.

In order to be visually pleasing, the vertices and edges need to contrast with the background. The scans that will be studied are old books and the most prominent colour in these scans can vary from white or light brownish, to darkbrown, yellow or even orange. Finding a nice contrasting colour can be a bit annoying because brown consists of almost equal amounts of red, green and blue. In order to get a nice contrast, two conditions need to be met: The difference in brightness and the difference in r, g and b values both need to be big enough. Because red and green contribute the most to brightness, the most contrasting colours will be black, white and all taints of blue. Black covers most cases because the background will usually be light. Combining a black colour with a lighter interior will help cover the other cases. A light blueish colour seems to cover most of the cases.



2.5 Research about exporting

2.5.1 1. Requirements

Requirement	Why
Export the scan currently being viewed without transcriptions	Make a .pdf file of the scan currently being examined in the Viewer. The transcriptions belonging to this page wil not be added to the export. This is useful for researching the page in a different program, when without connection to the internet, and for printing it.

Export the scan currently being viewed with the transcriptions belonging to that page	Make a .pdf file of the scan currently being examined in the Viewer. The transcriptions belonging to this page should be added to the export. This is useful for the same reason as when exported without the transcriptions, only this time the user will have extra information about the page together with the scan. This way, the user can take not only the book itself, but also the research from the site and work with it in some other environment.
Export a Binding without its transcriptions	For the same reason as exporting a scan, only now the full book is exported. This is extremely useful when it comes to medieval books, where each book is considered unique. If you have a "Book X" from 1550 in your local library and one in some far away place, and you are researching the one in the far away place, you can not use the copy at your local library for quotes and research. It may greatly differ from the "Book X" at the far away place.Normally researches travel to these distant libraries, but this naturally makes things difficult. With online electronic copies they can now research the books on their computer, from wherever they are.The goal of the export is to facilitate not only (prepared) offline research, but to give the user the ability to print the pages if they so wish. This way, a book is no longer confined to a location; anyone can make a copy of it anywhere.
Export a Binding with its transcriptions	By adding the transcriptions made by users to the exported book, researching it should become easier. The transcriptions could contain translations, connections and points of further thought. The goal is to add information to the text aiding the researchers in their work.
Export part of a Binding	Often times it is not a single page that is interesting, but a set of pages. However, a Binding can easily compromise hundreds of pages. If you are only interested in a set of 20 pages, it is a lot of work to export every page individually. Hence, the option to export pages x to y with or without transcription.
Display information about the source of the export	It has to be clear from what book the scans come. Also the page numbers of the scans should be noted with them.The page numbers in the system do not always correspond to the page numbers in the actual book. However, by noting the system's page number with the exported scan, at least the user can find it back in the electronic online copy.

Add online location and date of export in .pdf	Because our system is an online source, it is very open to change. Pages can be swapped, books could be replaced with better scans, or maybe the whole thing is simply pulled of the system. If the exported product is used for referencing, the internet address and date have to be provided.If the export is with transcriptions, the date is even more important, as they change over time.
When exporting with transcriptions, select the language of the transcriptions	Transcriptions can be made in the original language, English, and possibly later in then added languages. The user should be able to select if he wants original or translated transcriptions.

2.5.2 2. Research

Here we will list all the information we collected

2.1 What is asked

The ability to print and export to .pdf scans, books and transcriptions. The usefulness has already been hinted at in the requirements. The idea is to let users decide how they want to work with the material provided by the system, instead of having the system design govern their methods. Also the exporting and printing can be applied for other activities, like education and recreation. Print outs of books can be made for students to use with assignments. Ultimately, users should be able to apply the contents of the system for their needs.

However, their may be a few problems. Does the content on the system, automatically belong to the system? Can we legally spread scans made and uploaded by third parties? And even if we can spread the scans, what about the transcriptions made by the users of the system. Are we entitled to add their idea's and work to the scans without their consent? It is likely that, without any legal steps, the answer to most of these questions would be no. However, if we create a Terms of Use that provides us with the right to reproduce all content on the system, then we are allowed to distribute it, be by monitor or paper. This Terms of Use is a part of the user registration; the exporting and printing will simply use the same rights as their digital sources.

A second problem, which is primarily a problem in design, is how to combine the scans with the transcriptions. It would be preferable to have it look natural and pleasant to use. However, under no circumstances can our text and polygons be added to the scan in a way that would suggest it being a part of the original work. For research, it is vital that the scan matches the book it was made from. If we, for instance, add the polygons that the transcription is linked to, it has to be clear that this is an edited scan of the book.

Potential solutions would be to add the transcriptions on a second page or add them to the side of the scan. The downside of adding them to the side, is that to fit on a A4 the scale of the scan may have to be edited. Also, if there are is a lot of text, they may need to continue on the next page anyways. The upside of adding them to the side, instead of on the next page, is that you can view scan and transcriptions on a single piece of paper or screen.

2.6 Research about Help

2.6.1 1. Requirements

Requirement	Why
Complete	Many users will be inexperienced with computers in general, which means we can not expect them to know anything. This means that if we need them to know anything at all, we must have an explanation ready for them.
Compact	Inexperienced users such as those we are expecting will be scared away if they, when seeking help, walk into a big wall of text. The shorter the answers, the more likely it is the users are willing to read them.
Clear	Both completeness and compactness can have negative effects on the clarity of help functionality: completeness means there will be a lot of information, while compactness can go too far and remove important information. This means that the goal of clarity needs to be kept in mind at all times: when designing the help interface (sorting the information, showing the information) and when writing the actual text.

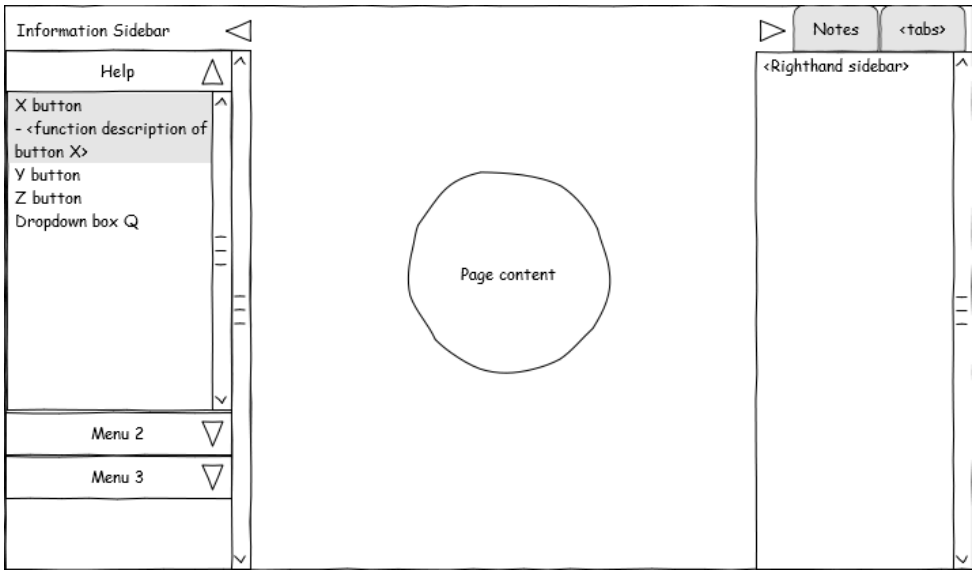
2.6.2 2. Research

2.1 Interface

Multiple options have been suggested for the interface. It comes down to two basic forms: putting the help functionality in the sidebar, or in its own page (or tab, similar to viewer and search results etc.).

Sidebar option:

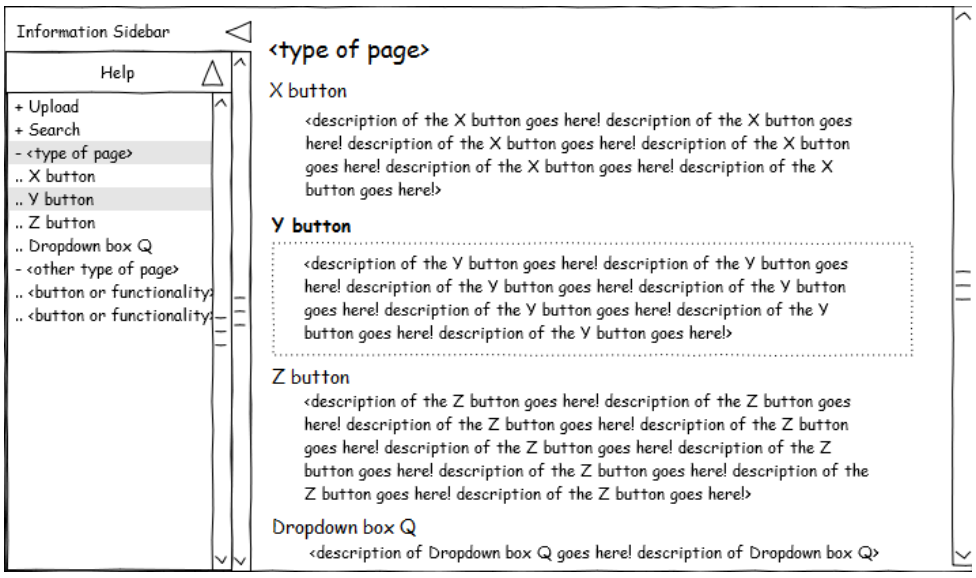
The following image shows (roughly) the option of help functionality in the sidebar:



- The options in the help listing are the names of buttons, or options, or other help subjects.
- Clicking on the list items will show the function description, or other help information regarding the subject.

Separate page option:

The following image shows (roughly) the option of help functionality in a separate help page:



- The Information sidebar shows a collapsable menu of subjects, ordered by page type (upload, search (results), bookcase etc.) or special functionality (notes, exporting) if the functionality is big enough to fill its own page.
- Clicking on the name of a page will show the page.
- Clicking on a specific subject will show the page it is on, and scroll down to + mark the specific subject (above image shows this as bold title, and a dotted box around the text).
- Links to other help subjects can be included in the text, working similar to clicking on the subjects in the list on the left (open page, scroll down to subject, mark it).

  - This way, the help can reference similar or related functionality on other pages, and users can read it by simply clicking the link. They will not have to go to that page and open its own help functionality.

2.7 Research about managing user generated content

2.7.1 1. Research

We are still unsure whether the collaboratory should mainly be a place where people can work together, a 'collaboratory', or mainly a publishing platform.

Potential problems:

- Spam. Some people might put unwanted content on the collaboratory. Some accounts could be spambots which use large scale spamming to affect not just a few pages but the whole collaboratory.
- While people are working on something, others might start to work on the same thing.
- When someone has published his work on the collaboratory, it might be altered and the original publication will be lost.
- Someone might delete or alter content which should remain as it is. If there is no way to regain the original content then it will be lost, which is something we wish to avoid.
- If people are forced to put one 'correct' piece of information on some part of the collaboratory, they might not agree on what the 'correct' information. This is not necessarily a bad thing, but it might create unwanted situations where people are changing the information back and forth all the time.

2.7.2 2. Possible solutions

2.1 Techniques

Here we will mention a few techniques which we might use in our solution.

Version control: By storing every change made on annotations, we can allow people to restore a page or a book to a state it once had. We can allow all users to reset changes or allow only admins and moderators to do that. Allowing users to reset changes may decrease the user-friendliness of the system, but allowing only admins or moderators increases the burden of maintaining the system. Another concern might be that the servers may require additional diskspace in order to store all of the previous versions. This is probably a minor concern, as the amount of additional data is not very large. If necessary we could allow admins to delete all previous versions up to a certain point. An advantage of this technique is that we can defend against spambots, ill-intended users and accidental changes or deletions at the same time. Disadvantages are that it is more difficult to manage, it increases the required diskspace and some users may abuse the system.

Locking: With locking we mean that we will disallow users to make changes to a page or a book. We can allow one user, a group of users or everyone to make changes to the book and lock it for everyone who is not allowed to do so. The only disadvantage of locking a book is actually equal to the advantage: People who might change the content and could improve it or make unwanted changes are not allowed to do so anymore. We could lock permanently (until a moderator or admin changes this) or a set period of time.

- Forced publication system: When a user has uploaded a book, he can choose to make the book accesible to everyone or keep it to himself. When he has chosen to keep it to himself he will be the only one (except for admins and moderators) who is able to edit it during a set period of time (e.g. 6 months). After this period the book will be published. The user can ask a moderator/admin to extend this period of time if he wants to. During this period he can give the right to edit the book to other users or take it away.

Rating: We could store multiple copies of an annotation/page/book and allow users to give a rating to them. The most popular one will be the one you will see first. However, this does not work with a small userbase like the one we expect for the collaboratory. Most researchers would find it degenerating and therefore we do not think it is a feasible option.

Drafts: By letting a user save his own version (draft) of a book, he can work on it while other users can not see what they have done. Once he has finished, a user can publish his work. The problem with this approach is what to do with changes already made to the book on the collaboratory. (You could throw them away, keep multiple versions etc.)**Registration:** Although not really a technique, the way people register in order to get an account may impact the way content is edited. Possible ways of registration are:

- Open registration: Anyone can register. The advantage of this approach is that people can easily add things to the collaboratory and therefore will be more inclined to do so.
- Closed registration: Only those who are approved by an administrator/moderator can register. With a closed registration you can depend on the User's goodwill a lot more then you could with an open registration. Most people who have something worthwile to add will be dedicated researchers who won't mind taking a little extra trouble to register. They would also feel more secure when they know not everyone can edit their work.

  - Registration on request: People who want to register first need to get approved of by an administrator/moderator.
  - Registration with invitation: Only people who are invited may register. It saves the administrators/moderators some trouble. But it might create extra trouble when people who have not been invited want to join.

2.2 Solutions

Wikipedia system By using open registration, version control (one change at a time) for everyone and locking, we will essentially mimic wikipedia's approach of regulating content. Users can, and will misbehave and content can be changed while someone is working on it. Users will try to regulate the content themselves by using version control, while admins and moderators can use banning, locking and version control (reversing multiple changes at the same time) to further regulate the content. This may scare away people who do not want other people to change their contributions, and it may not work well with a small userbase. On the other hand, by allowing multiple users to change things without irritating regulations, the people who wish to contribute will do this more often Because the collaboratory is not a likely target for spambots, the collaboratory will probably have a small userbase and the fact that it will probably be rare for two people to work on the same book (without working together) the problems which can occur will probably not occur often.

Research centre We could combine a closed registration with invitation and version control for administrators/moderators. Because registration is based on invitation, (most) Users will be trusted people who won't misbehave. Multiple users trying to work on the same thing also won't be a big

problem since Users can agree on such things by themselves. In the case something nasty does happen or a mistake is made, it can be made undone by an administrator/moderator using version control.

Forced publication system Combining the forced publication system and open registration with version control and locking methods might be a feasible way to combine the wikipedia system and users who want to work alone or in small groups. A disadvantage is that it might slow down the publication process**.**

Other combinations of techniques

2.8 Research about notes and snippets

2.8.1 1. Requirements

Because this research includes two features, there are two sets of requirements.

1.1. Requirements for notes

Requirement	Why
The notes are viewable in several parts of the application • – Viewer – Bookcase – Search – Possible other screens	The user has to be able to take notes about objects, like scans of pages and information on books, which can be viewed in these different parts of the system.
Add or remove text from the notes	The user has to be able to manage the contents of his own notes. Not only should he be able to always write something down quickly, but it should also be possible to remove this note when it is no longer needed.
Copy and paste from and to the notes	It should be possible to copy parts of information from and into the notes. This way, the user can quickly add more information to his notes, or take his notes out to place them somewhere else when needed.
The notes should always contain the same contents where ever the user is in the application	Basically, there is only one instance of notes being displayed in several parts of the application. This way, the user can add information about a certain object (book, page), being viewed, and take that information to another object.

The notes should be persistent throughout several sessions with the application	The user should be able to add information to the notes, log off, and find the same notes when he logs in again at a later time. This way, the notes can be used to help the user remember what he was working on last time he used the application.
There should be a way to export the notes	There has to be a way to take the notes out of the system. For instance, to print them or to add them to other programs or uses.

1.2. Requirements for snippets

Requirement	**Why**
Select a part of the scan and store it as a snippet	The user can make a copy of certain details he is interested in. A snippet is very much like a note, only it is taking a note directly from a scan. When a snippet is taken, it is added to the list of snippets in the notes tab.The reasons for making a copy of a page detail are to further investigate and compare it to other pages and books.
Remove a snippet	Of course, it should also be possible to remove a snippet from the list of snippets.
View a snippet	The snippet should not only be viewable, but also be displayed at a resolution where the contents of the snippet are meaningful.
The snippets are viewable in several parts of the application, the same as the notes	Because one use of snippets is to compare details of books and pages with each other,
Throughout the application, the list of snippets has the same contents.	If the list of snippets is updated in one tab, it should display the same contents when visiting another tab.
The snippets should be persistent throughout several sessions with the application	When the user logs off, the snippets should be stored for the next time the user uses the system.
There should be a way to export the snippets from the system	The user could want to use or investigate in a different way the copied page details. As such, it should be possible to convert the snippets into pictures or something that can be printed.

2.8.2 2. Research

This research page contains the requirements and research to two functions. The two are so closely linked that they are both discussed in the same research and functional design. However, as they differ in functionality, their requirements and reasoning for implementation are separated.

2.1. Notes

Notes is a simple field where the user can add and remove text. Only the user can access his own notes, others can not see or reach it. It is like their personal notepad.

The purpose is to supply the user with a means of taking and storing notes without having to resort to some other program, like Microsoft Word, or medium, like a real notepad.

It can be used to store observations about a certain book, page, search result or research, and take them to another book, page, search result or research. It can also hold a to-do list or any other sort of information the user wants to remember. If the notes were to be stored from one session into the next, the user could also add some remarks to himself about where he left off last time.

2.2. Snippets

A snippet is a copy of a part of a page. For instance, a copy of a certain character or drawing. It can be used to compare that detail to other details on other pages or in other books. For instance, to compare handwritings to see if certain annotations have the same annotator.

The snippets are stored in the notes tab and are synchronised throughout the system. That way, if a user adds a snippet, he is sure to find it in the snippet list when he accesses it from another tab. It should also be possible to export the snippets, as a way of taking pictures and visual notes from the scans. The snippets and notes could be linked together for export.

The snippets are displayed as thumbnails in the notes tab. When the user clicks the thumbnail of a snippet, that snipped is enlarged within the notes tab, allowing for closer inspection. This choice was made to make it easier to implement and use.

The snippets could be made persistent throughout several uses of the application; the snippet list could be stored for the next time the user logs in. With notes, this has an obvious advantage for storing information about the last session for the next. However, piles of pictures taken in a former session, their original intent long forgotten and ignored, could be more annoying then useful. Should the snippet list always be wiped at the end of a session? Or could we offer the user a choice when he wants to leave the application? If giving a popup upon closing the browser is to annoying, it could be some kind of options setting.

Problems that can arise from enlarging snippets

- – What to do if the snippet, at the zoom it was originally made, is much larger then the notes tab? Should a user be allowed to zoom in on a snippet?
  - – Even if it was technically feasible to do so, would these viewer functions not confuse the user? It is probably clearer to offer no support and enforce the user to make a snippet the desired size, or resize the notes tab, then zoom and drag within the snippet.

2.3. Sharing a tab

Certain problems arise when placing two features in the same tab

- – How should the space of the tab be divided between the two?
  - – When a snippet is enlarged, should the notes textfield still be visible?
  - – When exporting, should it be standard to export both snippets and notes?

2.4. The snippet tool

To take a snippet from a page in the viewer, the snippet tool has been added to the viewer. It was not added to the notes tab on purpose, allowing the user to quickly take snippets at any time.

When the user selects the snippet tool, he can draw a rectangle on top of the scan, where everything caught within the rectangle should be turned into a snippet.

The special thing about this tool, is that the confirmation buttons for accepting or declining the snippet, are displayed inside the viewer. The reasoning behind this is that the tool should be easy to use and not interrupt the user. If the users draws a snippet, but makes a mistake, it can quickly become annoying if he gets a system freezing popup every time he lets go of the mouse. It would be even worse to simply add every selected area as a snippet straight away.

By presenting the buttons within the viewer, the user can accept or decline while keeping an eye on the selected area. He can also redraw his snippet without any hindrance.

2.9 Research about select, search and sort

2.9.1 1. Requirements

Search - Requirements	**Why**
Ability to search by the Book attributes, as listed in the definition of Book. For clarity, (currently) these are: **Title**, **Author**, **Time period**, **Place published**, **Publisher** / **Printer**, **Version**, **Language**. <R>	Requested, and rather obvious.
Ability to search by the Binding attributes. These count for all the Books contained within the Binding. <R>	Addon to Book-specific search functionality. As the original reader is specified per Binding, and not per Book, searching by original reader (amongst other things) would otherwise not be possible.
Multiple field search (combination of multiple search criteria). <R>	Requested. Should not be hard to implement, will severely increase usefulness of search functionality.
Search for input X only in selected attribute type (see definitions of Book and Binding). <R>	To prevent large lists of false positives. One obvious example: if looking for books with a city name in the Title, books published in this city without the name in the title should not show up as results as this might be a very long list.
Sort - Requirements	
Sorting functionality should allow sorting by at least one, possibly multiple attribute(s) (derived from the definition of Book and Binding) simultaneously. <R>	To make it easier to find and order titles.
Select - Requirements	
(Optional?) Visualisation of some sort (cover, first page, selected scan of page, or similar). <R>	Requested. Useful to give a quick idea of the layout, size and type of book. If the exact title of a book is forgotten, a visual clue might help with finding what is being searched for.

Clear, not overly detailed results in listing. <R>	To prevent from making the system look like a database or Excel sheet, and keep it user-friendly. Example presented by Arnoud is the iBooks library, however this is not detailed enough for our needs as it is limited to a visual representation, which in our system might be optional.

2.9.2 2. Research

2.1 Background information

The users of the system will be literary researchers, most of them not very familiar with using complicated software. The goal is therefore to make the interface clear and easy to use, without requiring a lot of familiarity with computers. Before reading a document with the viewer, users will likely search for it, sort the results, and select the book(s) they are interested in. This page describes these three steps.

Examples given and requests made by the client are directing us to a "less is more" approach. The idea here is that if there are less buttons and fields for the user to choose from, there will be less confusion. On the other hand, we need to at least implement features that will be required to use the system for basic tasks, such as a robust search option that does not return too many false positives, and a search results listing with enough information so that it allows for selecting the correct book. The aim is to find the balance between too limited (useless) and too complex (confusing).

2.2 Search

Arnoud has indicated that it should (at least) be possible to search for the information in the Book (and Binding with the contained books as results) attributes, such as Title, Author and Provenance. Other search functionality, such as searching through comments made by the users of the system, or the upload or latest editing date, might be added later.

As there may be cases where the search query for one attribute would result in many false positives due to being common as value for another attribute, it should be possible to select the attribute that is to be searched. This means the user should be able to specify what to search for (such as Title or Author), and what it should match with (the book's actual title, or the author's name). For ease of use, the default could be set to "Search: any" (meaning: no limitation to what attribute is being searched). To further limit results, it should be possible to combine queries. An example would be searching for a book with a title containing "foo", written by "bar", in the year "1669". As many users are not very experienced with computers, both the attribute to be searched for and combining queries should be made visual (for example, a dropdown box for attribute, and each query on a new row) instead of using keywords such as "AND", or special characters.

Further research still needs to be done on the more detailed behaviour requirements of search queries. However, we should at least try to give satisfying results without being limited to exact matches. This includes partial matches (a search for 'dam' should also return 'Jan van Dam') and working around small typing errors ('Amserdam' returns results with 'Amsterdam' etc.). Other default behaviour we might want to copy is having spaces between words count as "OR". Results that match both search terms are listed higher than results only matching one, and in case more than 2 terms are separated by spaces, more matching terms results in an increasingly high spot on the results list. In case a more experienced user is looking for an exact match (and thus might want to not see 'near-match' results), it should be possibe to work around the above-mentioned behaviour. Many

search engines allow this by surrounding a query by quotation marks. Example: ' "amserdam" ' should only return exact matches, so 'amsterdam' is not returned as a result. This also eliminates the "OR" functionality of spaces within quotation marks.

2.3 Sort

We are still working on clearly defined required and requested sorting functionality.

Sorting by the Book and Binding attributes is currently seen as requirement. Preferably multiple at the same time in specified order, such as alphabetically listing all authors, then per author list the books by year of publication.

Search results (or 'all books', not filtering anything away) should probably have some relevance score. Then one sorting option could be 'sort by relevance', making exact matches end up higher than partial matches. If applicable, results that match more parts of an OR query (strings separated by spaces) would also have a higher relevance.

2.4 Select

Selecting a book depends on the visualisation in which it can be selected. Currently, this document is about this visualisation and selecting a book. It simplifies what happens after selecting a book to 'open in viewer', although more options ('open in new viewer' for a comparison, or moderating options) might be added later.

Examples given by Arnoud for what he imagines the system should look like include iBooks and Google Books. A short comparison:

- iBooks has nice looking visuals and shows many books in a small area, but does not have the level of detail required for this project. Another problem is that it works with front covers of books, which are not mandatory for our book uploads and therefore can not be used as identification.
- Google Books shows a lot of information, but has a rather spacious layout which means that long lists of search results will take too much scrolling to get through.

What we are aiming for is a mix between the clear, compact listing of iBooks, combined with the information supplied by Google Books. In the visualisation, at least the Title and Author should be present (information), but not all attributes of Book and Binding can be included (clarity). The exact attributes that need to be included in the visualisation are not yet set in stone, and will probably change depending on experiences during testing.

2.5 Answers to questions asked to Arnoud and Els:

Hoe zinvol is het om boeken te kunnen sorteren op het tijdstip dat het boek aan het collab is toegevoegd? (bijv. om te kijken of er een nieuw exemplaar van een boek geupload is) Of om te kunnen sorteren op het tijdstip dat het boek voor het laatst bewerkt is? (bijv. als er een transcriptie is toegevoegd) (Er van uitgaande dat er waarschijnlijk wel functies komen om jezelf te "abonneren" op een boek, waardoor je automatisch op de hoogte van wijzigingen wordt gesteld als je op een boek geabonneerd bent)

*Arnoud: Leuk idee! Het zou zeker een leuk extra feature zijn om te kunnen zien wat net nieuw is toegevoegd, maar ik weet niet zeker of dit een zoekterm is waarop mensen uiteindelijk vaak zullen zoeken (omdat het in beginsel een heel breed veld betreft met potentieel allerlei verschillende onderwerpen). Wellicht iets voor een aparte melding *nieuw*. Wat betreft je melding van bijwerkingen*

van een bepaald exemplaar: ja, dat lijkt me erg goed! Als je aan een bepaald boek werkt, of hierin geinteresseerd bent, dan lijkt het me erg nuttig te weten dat er ook door anderen naar gekeken en aan geschreven wordt.

2.10 Research about the viewer

2.10.1 1. Requirements

Requirement	Why
It has to be clear which part of the book is being viewed • Give the User feedback on location in book • Give the User feedback on location in page	The User has to know what he or she is currently viewing. Book, in this context, means a Binding with all its Books.
Support functionalities for navigating through the book • View pages in the book • Move to the next page • Move to the previous page - in this case previous does not mean the previously viewed page, but the page before this page in the book. • Move to a selected page	The User has to be able to select the page he or she wants to view.
Supports functionalities for navigating a page • Select a new focus point on the page for the displayed image in the viewer to centre around. The image is the part of the page(s) that is being displayed.	The User has to be able to select the part of the page he or she wants to view.
Supports functionalities for viewing a page • Zoom in • Zoom out • Rotate the image	The main purpose of the viewer, view and examine pages. To help with examining pages the ability to zoom in on details, zoom out and rotate the image are crucial. The image is the part of the page that is displayed in the viewer.

Support functionalities to view the pages in a book form, 2 pages, side by side • A way to switch between page-mode and book-mode • When viewing in the book-mode, all navigation methods should work as they do in the normal page-mode • When in book-mode, there should be the added possiblity to 'turn a page', displaying two new pages	The site has to give to the user the feeling he is reading books like was done 400 years ago. In those days, the research material was printed.
View information about the book	Information on the book has to be easy to find and process for the User. This information includes information about the Binding, about the current Book (the one within the Binding), and information added by Users.
View information about the current page	The User has to easily open and process gathered information about the book. This includes translations of the page, comments made about the page and its annotations and all other page related data stored by the system.

2.10.2 2. Research

2.1 Information about the Viewer

The Viewer is used to view and add information to pages from books. Books here means both the Binding and the Books contained in it. The Viewer contains both functionalities associated with a regular viewer, like zoom in and zoom out, and functionalities concerned with navigation in the book, like next page and jump to page.

Making old books with annotations accessible to people worldwide is the main purpose of the site. The viewer and the books are open to be viewed by anyone, without having to log in first.

Because the site has to give a virtual reality of ancient book-reading, it has to be possible to view the pages as like they are part of an actual book. I.e. we can see two pages side by side, as if we were looking at an actual book, and we are able to 'turn' a page. Arnoud expressed a desire to be able to view both sides of the same page at the same time. Moving just one page in the book-mode should therefore also be possible.

2.11 Research about uploading

This page contains most of the relevant research about uploading. The FD - Uploading is a follow-up on this document.

2.11.1 1. Requirements

Requirement	Why
The User is able to upload multiple images at the same time.	Because these images can be quite large, it is more convenient for the user to select a range of images, and let those be uploaded. Otherwise, the User would have to select the images one by one.
The progress of the upload action is shown.	The images can be quite large, so the User should be able to monitor the progress.
The upload action may be cancelled.	When the User wants to abort the action for some reason, he should be able to and possibly select other images for uploading right away.
Users are able to provide extra information about the images when uploading them.	When creating a new Binding, the User should be able to provide extra information about the Binding and/or Book. Otherwise, the User would have to wait for the uploading to finish, and than enter the information. Best would be if this process could be completely asynchronously. I.e. the User starts uploading some images, and while uploading enters all known information about the Binding or Book. Some information is mandatory, see the Binding page and the Book page for more information.
Users are able to correct their mistakes.	When some error occurs or the User made a mistake, the User should be able to correct this problem and continue the process.
Moderators and/or Administrators are able to approve and/or edit an uploaded document.	When the information uploaded is incorrect or the quality of the scans is insufficient or the User has made some other mistake.
Users are able to upload TEI XML.	Some Binding or Book may come with a TEI XML document, so the User is able to upload the XML.
Users are able to upload books without scans and/or XML.	This situation is very common and not desirable. But, in case there are no scans the User should be able to upload and insert other available information.

2.11.2 2. Research

2.1 Background information

A lot of books are bound together with other books into a Binding. This has several reasons. One reason is that printing books and binding books was a different trade. Quite often, people bought a couple of books, went to the binder and the books were bound together. So it can occur that the different books in one binding do not have a lot in common. On the other hand, the choice of books can be very important. Sometimes two or more books of rival authors were bound together

to investigate the differences between them and to have two contrasting ideas next to each other.

Another important thing about these bindings (or convolute) is the information that should be stored at this binding, instead of a Book. For example, in a Library each binding has a unique identifier for that library. So this has to be stored at the binding. Sometimes we also know who used to own the book because their names are written or printed inside the cover. And most important of all, the researchers are interested in the unique copies (binding) and not just the books inside a binding. So a book in the collaboratory should be the binding with all its information and all of the books. Second step is to register the different books inside this binding, to be able to index or search them. Another reason why the unique copy of the binding is interesting is the relation between the annotations in the different books within this binding. Especially when these annotations (in different books within the binding) are written by the same reader.

At the Book we also need to register some information that cannot be registered with the binding. You can think of properties like author, printer, year of publishing, language and so on.

To conclude, we need to make the uploading very flexible. There may be more than one book within a binding, and there may be scans and/or XML available. Another thing is that the available information about the binding or the book discussed above may be very limited. There exist books of which the author is unknown, for example. And bindings of books may not be in a library with a unique identifier, but in a private collection without any unique identifier.

2.2 Uploading

Because of all the reasons discussed above the design should incorporate a lot of freedom in what information the user provides. The administrator is of course able to intervene later on, when the books have been uploaded. We also need to keep the technical aspects in mind. The scans may require a lot of storage, so even today's internet bandwidth may become a limiting factor. So we need to provide the user with useful information like time remaining. Besides these aspects, the idea is basically to enable the user to upload a book (providing some infomation about it) and immediately make the book available to the administrator and any other visitor of the collaboratory.

Part II

Technical

Chapter 3

Coding standards

3.1 Coding standards

3.1.1 Indenting, line length and tags

Use an indent of 4 spaces, with no tabs. This helps to avoid problems with diffs, patches, SVN history and annotations.

Keep the lines under a 100 characters. Indent the next line with one indentation level (4 spaces) if you separate your expression over multiple lines. If a function call has a lot of arguments, this might sometimes be more clear:

```
$this->someFunctionCall(
    $firstArgument,
    $var + 2 * $someLength / $someFactor,
    SomeClass::SOME_CONSTANT
);
```

Use the full start tag at the beginning of the code: <?php. Do **not** use an end tag, as there might be some newlines after the end tag, and this will cause the **header** API call to fail, as PHP will output these newlines to the client.

3.1.2 Comments

Though a pain to write, they are absolutely vital to keeping our code readable. The following rules describe what you should comment and where. But remember: while comments are very important, the best code is self-documenting. Giving sensible names to types and variables is much better than using obscure names and then trying to explain them through comments.

When writing your comments, write for your audience: the next contributor who will need to understand your code. Be generous — the next one may be you!

Write proper sentences. Always use periods.

Comment types

Only use double slash comments (//) inside methods. This allows for blocks of code to be commented out using /* */ for debugging purposes! Always place comments on a line of itself, so never behind a statement. Do not put a space after a comment.

To mark sections use:

```
////////////////////////////////////////////////////////////////////////////////
// SOAP Service invocation
////////////////////////////////////////////////////////////////////////////////
```

For comments over 5 or less lines use:

```
// This is the first line of a comment. This
// may be a sentence that is divided over
// multiple lines.
```

For comments longer than 5 lines, use:

```
/*
 * This is the first line of a comment. This
 * may be a sentence that is divided over
 * multiple lines.
 *
 * More text could be over here, or on the
 * last line.
*/
```

Comment style

Write comments in 3rd person, use:

```
// Determine the total amount of all invoices
method () {}
// This method deallocates all resources used by the application
metthod () {}
```

Instead of:

```
// We determine the total amount of all invoices.
method () {}
// I deallocate all resources used by the application in this method.
method () {}
```

Write functional comments, not technical comments (do not refer to variable names). Example :

```
// Determine if the number of overdue invoices exceeds the allowed maximum.
if ($numOverdueInvoices > AMT_MAX_UNPAID_INVOICES)
{
}
```

And not:

```
// Determine if numOverdueInvoice is greater than AMT_MAX_UNPAID_INVOICES.
if ($numOverdueInvoices > AMT_MAX_UNPAID_INVOICES)
{
}
```

Javadoc markup

All Javadoc styling attributes can and should be used.

3.1.3 Naming conventions

Classes

Classes should be given descriptive names. Avoid using abbreviations where possible. Class names should always begin with an uppercase letter. Examples of good class names are:

Log	Employee	InvoicePaymentLine

Class fields and methods

Class variables (a.k.a properties) and methods should be named using the "lowerCamelCase" style (also referred to as "studly caps", "bumpy case" or "camel caps"). Some examples:

$counter	connect()	getData()	buildSomeWidget()

Name methods which determine something 'determine....()'

Always **try** to name member or local variables after their class. For example

```
$employee = new Employee();
```

For certain quantities a standard variable name prefixes should be used.

Prefix	**Example**	**Quantity type**
num	$numAttendees	A number of something
amt	$amtTotalLedgerEntries	An amount (currency)
date	$dateContractStart	A date
max	$numMaxAttendees	A maximum (if any other prefix is applicable append 'Max')
min	$minAltitude	A minimum (if any other prefix is applicable append 'Min') min

Constants

Constants should always be all-uppercase, with underscores to separate words. Prefix constant names with the uppercased name of the class/package they are used in. Some examples:

DB_DATASOURCENAME	SERVICES_AMAZON_S3_LICENSEKEY

3.1.4 Operators

Use spaces around **every** operator. That includes: =, +=, ==, , ||. Use one space after each comma. Use a lot of brackets to indicate precedence, as a lot of languages have different operator precedence rules. Especially use brackets around the short if: (a ? b : c). Brackets can also be used to group expressions, but only if it adds value to the readability.

3.1.5 Control structures

Curly brackets

Place curly brackets ({ and }) on a different line for every control structure. This is called the BSD (Allman) style. Add a space between the control structure keyword and the opening bracket. The while of a do..while structure hangs onto the ending curly bracket. An example:

```
if ($something == $somethingElse)
{
    $this->doSomething();
}
else
{
    $this->doSomethingElse();
}

do
{
    $this->doSomething();
} while ($something == $somethingElse);
```

http://pear.php.net/manual/en/standards.control.php

3.1.6 Function calls

Always use lowercase characters for API function calls. PHP has a case insensitive matching of functions names, but this will change in PHP 6. So to be forward compatible, use the same case as the definition of the method.

http://pear.php.net/manual/en/standards.funcalls.php

3.1.7 Class definitions

Use the BSD style for curly brackets.

http://pear.php.net/manual/en/standards.classdef.php

Function definitions

Use the BSD style for curly brackets. Always add a publicity modifier to a function (public, private or protected). Use __construct as the constructor name, not the deprecated class name. An example:

```
<?php

require_once 'framework/model.php';

/**
 * User model.
 */
class UserModel extends Model
{
    const RANK_NONE      = 0;
    const RANK_MODERATOR = 1;
    const RANK_ADMIN     = 2;

    private $username;

    public function __construct($id)
    {
        // Initialize with id
    }

    public function getRank()
    {
        return RANK_NONE;
    }
}
```

http://pear.php.net/manual/en/standards.funcdef.php

Field definitions

Always add a publicity modifier to a field (private or protected). Do **not** use the var keyword.
A class field is always private or protected, use getter and setter methods to change a member.

3.1.8 Arrays

If arrays are short, use a single line to declare them. If the array has a lot of entries, separate the entries over multiple lines. Indent each line with one extra indentation. Try to line up the keys of the array. Some examples:

```
$someVar = array(1, 2, 3);
$another = array('name' => 'Value', 'another' => 'Another value');
$foo = array(
    'a',
    'b',
    'c',
    'd',
    'e'
);
$var = array(
    'ab'   => 1,
    'b'    => 2,
```

```
    'cde'  => 3,
    'd'    => 4,
    'efgh' => 5
);
```

http://pear.php.net/manual/en/standards.arrays.php

3.1.9 Header comment block

```
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; version 3 of the License.

 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * Copyright: Mathijs Baaijens, Iris Bekker, Renze Droog,
 * Maarten van Duren, Jeroen Hanselman, Bert Massop, Robin van der Ploeg,
 * Tom Tervoort, Gerben van Veenendaal, Tom Wennink.
 */
```

Chapter 4

Configuration properties

4.1 Configuration properties

Property name	Description
Database	
database-dsn	Database data source name.
database-username	Database username.
database-password	Database password.
User	
password-salt	Salt for password hashing, should be unique.
Session	
session-prefix	Session prefix, should also be unique.
Logging	
debug-mode	Whether to send stack traces and complete errormessages to the client.
logging-level	How many messages are logged:0=none, 1=error, 2=warn, 3=info, 4=debug.
Frontend	
minify-resources	Whether to minify stylesheets and Javascript.
Paths	
upload-path	Path where upload files are initially placed.
Pyramid Builder	
builder-path	Path of the pyramid builder executable.
tile-output-path	Path of the directory where the tiles will be stored.
tile-quality	Value between 0 and 100 indicating the JPEG qualityof the output files.
Thumbnails	
thumbnail-path	Path where thumbnails will be stored.
thumbnail-width	Width of thumbnails.
thumbnail-height	Height of thumbnails.

Chapter 5

Permissions table

5.1 Permissions table

This page specifies which rows should be available in the Permissions table. Each row denotes the name of an action and how high the rank of a user should be in order for them to be able to execute them:

Action Name	Minimal Rank	Description
view-pages	Guest	Use the viewer to view books
search-books	Guest	Search for books
add-annotations	Regular User	Add annotations to pages
edit-annotations	Regular User	Edit existing annotations
upload-bindings	Regular User	Upload scans/books/bindings
manage-bookshelf	Regular User	**(Deprecated)** View and manage an personal bookshelf
manage-notebook	Regular User	View and manage a personal notebook and snippets
export-books	Regular User	Download scans of books in PDF format
view-users-part	Regular User	View a list of all registered users, not necessarily with all information
view-history	Moderator	**(Deprecated)** View the version history of changes made by users
revert-changes	Moderator	**(Deprecated)** Revert changes made by users
change-book-info	Moderator	Change the information of a book or binding
view-users-complete	Administrator	View a list of all registered users, with all information
accept-registrations	Administrator	Accept or decline the registration of new users
ban-users	Administrator	Ban or unban users
change-user-roles	Administrator	Change the role of a user
delete-users	Administrator	Delete users

change-global-settings	Administrator	Change global settings that affect the collaboratory

As specified in the database model, the database rank values associated with certain roles are as follows:

Role	**Rank Value**
Guest	0
Regular User	10
Moderator	40
Administrator	50

Chapter 6

Settings table

6.1 Settings table

In the database, there is a table called "Settings" which can be used to store various options which should be easy to modify by the system administrator and would probably change more often than the contents of backend ini-files, see also the database model.

If a setting is visible to the client, its value can be directly requested by the client.

The following settings are currently required to be specified:

Setting Name	Description	Visible to client
project-title	The title of the project, now "Annotated Books Online".	Yes
activation-mail-subject	The subject of activation e-mails to newly registered users.	No
activation-mail-message	The message of the activation e-mail send to new users. Should contain the string [LINK] somewhere to indicate the place the activation link should be inserted. [USERNAME], [FIRSTNAME], [LASTNAME] and [PROJECTNAME] can also be used. The latter indicates the name of the project.	No
forgotpass-mail-subject	The subject of the mail send to users who forgot their password.	No

forgotpass-mail-message	A mail message similar to that when activating. Should contain[LINK] and may also use [PROJECTNAME], [USERNAME], [FIRSTNAME] and [LASTNAME].	No
welcome-page	The text displayed on the welcome page, in HTML.	Yes
info-page	The text displayed on the info page, in HTML.	Yes
upload-insructions	The text displayed before uploading, in HTML	Yes
terms-of-use	The terms of use you can view when registering.	Yes
auto-user-acceptance	Set to "0" when users should be manually activated by administrators.If set to "1", users can activate themselves.	Yes
user-declined-mail-subject	The subject of mails send to registered users who are declined activation.	No
user-declined-mail-message	The message of a user declined message. Works the same asforgotpass-mail-message.	No
deleted-user-id	The row ID of a special user used to represent all deleted users.When a user is deleted all foreign keys pointing to that user will bereferred to this.	No
mail-from-address	The address used as a FROM-address whensending an e-mail.	No

Chapter 7

Technical documentation

7.1 Client-side

7.2 Views

Sub-pages of this page describe individual views. If multiple small files in the frontend/views directory describe related elements they can be put on a single page.

7.3 Search Panel

7.3.1 Summary

The search panel allows the user to search in an intuitive way. It can handle various kinds of search sorters and selectors.

7.3.2 Result fetching

When a search query has been entered, the request is sent to the server as an object with the following members:

- *selectors*: an array of selectors, which are objects with the following members:
  - *type*: the type to search on, i.e. 'title'
  - *value*: the value to search on, i.e. 'romeo and juliet'
- *sorters*: an array of sorters, which are objects with the following members:
  - *property*: the type to sort on, i.e. 'title'
  - *direction*: either 'ASC' or 'DESC'
- *limit*: the maximum number of results
- *page*: the current page number

The expected respone contains a *total* member, containing the total number of search results (disregarding the limit), and an array of *records* that are objects with named property values.

7.3.3 See also

- The book controller.

Source

- frontend/views/searchpanel.js

7.4 Controllers

The following sections provide a description of the various controllers in the system. For general information on controllers, please refer to the framework information.

7.5 Book controller

7.5.1 Summary

The book controller handles loading and searching books.

7.5.2 Actions

Load

The default load action, extended by a default sorting on the *firstPage* attribute in ascending order.

FirstLastPages

Sets the *firstPage* and *lastPage* of the requested book instances.

Search

Searches on several different properties of books, incuding the full transcription text. The input and output of this action is specified in the search panel view description.

All fields, except for the year of publication, allow for full text search. When searching using the 'Any' selector (which includes transcriptions), highlights are also generated, displaying a part of the document that matched most parts of the query.

7.5.3 Full text search

To enable full text search, the (full text) query is split in exact and non-exact queries. Exact queries are required to match literally, while words in non-exact queries may match in any order and may differ a little from the words in the document. This fuzzyness is reached by leveraging the Postgres full text abilities, simplifiying words. For example, 'knights' matches 'knight', as the 's' is redundant. Fulltext queries are strictly 'AND', therefore requiring all given words to be in the matching document.

7.5.4 See also

- The search panel view.

Source

- backend/controllers/bookcontroller.php

7.6 PDF controller

7.6.1 Summary

The PDF controller handles requests for exported scans and/or bindings, utilizing the PDF utility to actually generate the file.

7.6.2 Actions

Generate

Generates the PDF for the given range of scans, and writes them to the cache. Afterwards, it notifies the client that it is done, giving a token that can be used to download the file. The client can then proceed to download.

Exports that are unlikely to change (i.e. requests for a specific binding or page without transcriptions) are only generated if they are no longer in the cache. All other exports are generated on request.

Download

Sends the file contents of the of the generated PDF to the client from cache. The client provides a token, previously generated by the Generate action, that functions as a key for the cached PDF file.

7.6.3 See also

- The PDF export utility

Source

- backend/controllers/pdfcontroller.php

7.7 User Controller

7.7.1 Summary

This controller handles all actions that have an effect on the current or other user(s). It is used to create new users, edit their profiles or obtain a complete user list.

7.7.2 Actions

Load

Loads a list of all registered users in the system: the 'records'-field in the result contains an array of array of associative arrays, each of the latter corresponding to the user model on the client. The amount of info send depends on the rank of the currently logged in user: regular users only receive usernames, e-mail addresses and full names while administrators receive all personal data specified by the users themselves, along with information like whether a user is banned or what their activation stage (see database) is. A regular user does obtain all personal information about itself, but no administrative records such as activation stage.

Save

Saves the changes made to the current user. Used by the 'edit profile panel' and can change all properties a user can specify during registration (aside from username). Can only be done by the currently logged in user to itself.

Create

Creates a new user and is called after submitting the registration form. A PendingUser entry is also created, containing a unique confirmation code usable to activate a users' account. If the 'auto-user-acceptance' setting is turned on, this user will start in the 'accepted' status (see database documentation) and a confirmation e-mail will be send immediately. Otherwise, the user starts as 'pending' and needs to be manually accepted or declined by an administrator.

UsernameExists

Indicates whether there exists a user with a certain username.

EmailExists

Indicates whether there exists a user with a certain e-mail address.

DeleteUser

Deletes a user. Requires the logged in user to be an admin. Database tables referring to the user to be deleted but that are not deleted along with it (such as 'Annotations') will have their foreign key referred to one special dummy user (called '<deleted user>'). Because of this, '<deleted user>' will appear in the place of a transcription's author when that author was deleted.

BanUser

Bans a user. Only allowed for moderators and admins. Banned users simply can't log in and will instead see a message that they are banned. User will remain banned permanently until they are either 'unbanned' or deleted,

UnbanUser

Removes the ban on a specific user. Only for moderators and admins.

PasswordForgotten

Called when a guest clicks the 'password forgotten' button and inputs an e-mail address to which an e-mail will be send containing an URL through which they can modify their password (during this process, the 'passwordRestoreToken' field of the user is set). When a user has received such an e-mail but, instead of following the URL, logs in with their old password, it is assumed the user remembered and its 'passwordRestoreToken' will be unset.

ChangeRole

ChangeForgottenPassword

Called after the URL in the 'password restoration e-mail' has been followed. Changes the password of the user identified by the 'passwordRestoreToken'.

7.7.3 See also

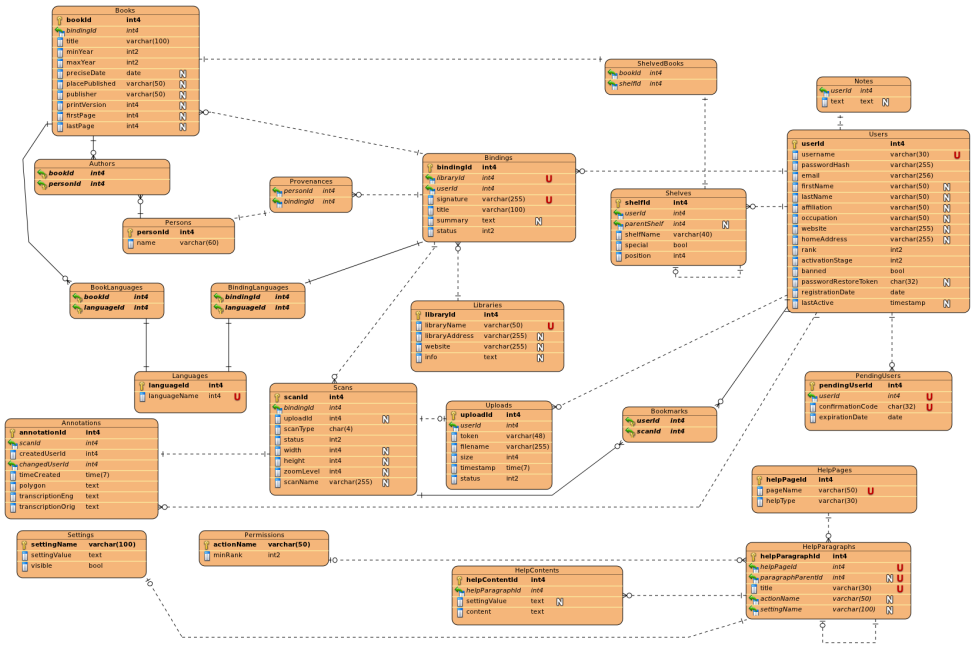
Source

- backend/controllers/usercontroller.php

7.8 Database

7.8.1 Overview

This model is an overview of all the database tables used, their columns, and their relations. Below is an alphabetical list of tables along with a description of their semantics and columns.



7.8.2 Generic columns

The following columns are present in each table, and are intended solely for administrative purposes. Each may be NULL, and the content of these tables should not affect the application.

Column	Description
createdOn	The date of when a row was inserted.
createdBy	The name of the user that was logged in while thisrow was inserted.
changedOn	The date on which a row was last modified.
changedBy	The name of the user that was logged in while thisrow was last modified.

7.8.3 Tables

Annotations

The Annotations table describes the location of an annotation marked by a user and contains the transcriptions thereof.

Column	Description
annotationId	Identifier.
scanId	The scan that contains this annotation.
createdUserId	The user that marked the annotation.
changedUserId	The last user that made a modification to either to location of the annotation or a transcription.
timeCreated	Timestamp of when this annotation was created.
polygon	A binary string of coordinates, each coordinate pair contains two 32-bit little endian IEEE 754 floating point numbers,respectively representing an x and y coordinate. Each pair descriptes the location of a vertex, and connecting thevertexes in order, you'll get the polygon.
transcriptionEng	A transcription of the annotation. In English.
transcriptionOrig	A transcription of the annotation. In its original language.

Authors

A connection between a person and a book, indicating said person was an author of the book.

Column	Description
personId	The person that is the author.
bookId	The book (co-)written by this author.

BindingLanguages

Note: the name of this table is somewhat deceptive. It probably should have been called BindingAnnotationLanguages.

Represents a many-to-many relationship between bindings and languages, and indicates in what language annotations made inside this binding were written.

Column	Description
bindingId	The binding.
languageId	One of the languages the annotations in the binding are written in.

Bindings

A binding represents a physical collection of books that were bound together. This table holds the information specified in the Upload-screen and is referenced by the books and scans contained within.

Column	Description
bindingId	Identifier.
libraryId	The library that currently stores this binding.
userId	The user that has uploaded this binding.
signature	Called **Shelfmark** in the upload screen. The combination of library and signature should uniquely identify a binding.
title	The title of the binding.
summary	**Unused**.
status	See below.

Binding status The status column of a binding indicates how far this binding is in the upload process. It can have one of the following values:

Status	Meaning
0	Starting state. Binding pages have been uploaded.
1	Reordering of the pages in the binding is done.
2	Selecting of books within this binding is done. Therefore they can now be found with Search.

3	The binding has been deleted. In order to prevent accidental loss of a lot of work the binding's contents are still physicallyavailable in the database, it has just become invisible. If necessary, an administrator will have to manually undo a deletion(set status back to 2) or wipe the contents of this binding (delete queries).

BookLanguages

Represents a many-to-many relationship between books and languages, indicating in what languages a book has been originally written. This does not denote the languages of the annotations inside the book.

Column	Description
bookId	The book.
languageId	One of the languages the book is written in.

Bookmarks

This table was intended for a feature that had been cut due to time constraints.

Books

Represents a book: a collection of pages held within a binding. This table contains the book information originally specified in the upload screen.

Column	Description
bookId	Identifier.
bindingId	The binding that contains this book.
title	The title of the book.
minYear	If *minYear* and *maxYear* are equal, this is the year the book is published in. If not, it means the exact year is unknown butprobably lies in between these two years.
maxYear	
preciseDate	**Unused**
placePublished	The location of the publisher or printer of the book. (optional)
publisher	The name of the publisher or printer of the book. (optional)
printVersion	The version or edition of this book. (optional)
firstPage	**Deprecated**
lastPage	**Deprecated**

HelpContents

Contains the HTML displayed under a certain paragraph in the help view. Sometimes, a single paragraph can have different kinds of content, depending on the value of a certain setting.

Column	Description
helpContentId	Identifier.
helpParagraphId	The help paragraph this entry stores the content of.
settingValue	If this is not NULL, this content should only be displayed if the setting with the *settingName* of the associated *HelpParagraphs* rowhas this value.
content	The actual content of the paragraph, in HTML.

HelpPages

A single help page. Its actual content is maintained by the *HelpParagraphs* and *HelpContents* tables.

Column	Description
helpPageId	Identifier
pageName	The name of this page.
helpType	When the help button is clicked while a certain tab is open, this identifer is used to determine which page should be opened first. The identifieris a string used in the client-side to identify views.

HelpParagraphs

A paragraph on a help page. Denotes the title of the paragraph and should be referred to by *HelpContents.* A paragraph is displayed in the help index. A paragraph can contain 'subparagraphs'. Paragraphs (that are not subparagraphs) on a page will always be ordered alphabetically when displayed.

Column	Description
helpParagraphId	Identifier.
helpPageId	The help page this paragraph should be displayed on.
paragraphParentId	If not NULL, this indicates the 'superparagraph' this one is contained in.
title	The title of the paragraph. This should not also be contained in the content but will be rendered automatically with the correct header size.

actionName	If not NULL, this paragraph will only be displayed when the user viewing the page it's on has permission to execute the action with this name.
settingName	If not NULL, this indicates that the content of this page depends on the value of the setting with this name. In order to determine the content,the *HelpContents* entry is picked with the correct *settingValue*.

Languages

Simple table that holds the name of a language. All entries in this table are selectable options for the language fields in the upload view.

Column	Description
languageId	Identifier.
languageName	The name of the language.

Libraries

Holds the name of a library. Eventually, nothing has been implemented that uses the other information fields.

Column	Description
libraryId	Identifier
libraryName	The name of the library, should be unique.
libraryAddress	**Unused**
website	**Unused**
info	**Unused**

Notes

For each user, a *Notes* entry is present that holds a text field containing anything a user wants to scribble in his/her personal notebook. Due to functionality that has now been scrapped, this is a separate table, but now it could just as well have been an extra column of *Users*.

Column	Description
userId	The user to which these notes belong.
text	The content of the notes-field for this user.

PendingUsers

After a user has registered, but is not yet able to login because he/she is not yet accepted or still has to click the activation link in an e-mail message, there will be an entry in this table for that user. Once a user is capable of logging in, the corresponding *PendingUsers* entry has already been removed.

Column	Description
pendingUserId	Identifier.
userId	The user this *PendingUser* represents. There can only be one *PendingUser* per *User*.
confirmationCode	The code that will be included in the link used to activate the user's account. Should be unique.
expirationDate	The date after which this *PendingUser* expires and the corresponding *Users* entry should be deleted if the user did not activatehis or her account. This deletion, however, has not yet been implemented and this column will be set but further **ignored.**

Permissions

This table is used in order to restrict what users with a certain *rank* or 'role' are allowed to do. For instance: an administrator can delete a user but a moderator can't, a registered user can add annotations but a guest can't. See this page for a description of all the actions and permissions.

Column	Description
actionName	A string identifier representing the name of a restricted *action*. When checking whether a user is allowed to do something, an action name is provided todescribe the thing the user wants to do.
minRank	The minimal *rank* a user should have in order to execute this action. See the description of the *Users* table for more details.

Persons

Holds the name of a person. Was originally intended to store more information, but this was eventually not implemented.

Column	Description
personId	Identifier.
name	The full name of the person.

Provenances

Called **Readers** in the eventual implementation. Denotes people that have previously owned or read a binding before it ended up in a library. The table is a simple many-to-many relationship between persons and bindings.

Column	Description
personId	A reader or owner of a binding.
bindingId	The aforementioned binding.

Scans

Denotes properties of scans of pages from books.

Column	Description
scanId	Identifier.
bindingId	The binding to which this scan belongs.
uploadId	An associated *Uploads* object. This field will be set temporarily, before the tile builder is done with this scan, as part of an optimisation.
scanType	Either '**jpeg**' or '**tiff**' and denotes the type of the full scan image that has been uploaded.
status	See below.
width	The width of the scan. Will be set after the tile building process because it can be easily determined then.
height	The height of the scan. Will be set after the tile building process because it can be easily determined then.
zoomLevel	The maximal depth of the tiles that have been build out of this scan. In other words: the number of zooming layers.
scanName	The path of the file that holds the original scan.

Scan Status The *status* column can have one of the following values.

Status	Meaning
1	Scan is awaiting to be processed.
2	Scan is still being processed.
3	Processing the scan has failed.
4	The scan has been removed.
5	Scan is fully available.

The *width*, *height* and *zoomLevel* of a scan with status 5 should not be NULL.

Settings

A number of settings are specified in the database. These simply have a name and a value. For a list of the settings and their meaning, see Settings and configurations.

Column	Description
settingName	The name and identifier of a setting.
settingValue	The value of this setting.
visible	If true, the value of this setting can be directly requested by the client. See Settings and configurations.

ShelvedBooks

Intended for functionality that has now been scrapped. **Unused.**

Shelves

Also **Unused.**

Uploads

Denotes a file that has been uploaded (currently it is only used for scans, since those are the only uploadable files).

Column	Description
uploadId	Identifier.
userId	The user that has done this upload.
token	A token used to identify this upload.
filename	The name of the uploaded file.
size	The size of the uploaded file.
timestamp	The point in time when this upload was finished.
status	0 if the upload succeeded, 1 if it failed.

Users

Holds users that have registered themselves to the application. Contains account data and personal information.

Column	Description
userId	Identifier.

username	The username this user registered with. When users are queried, usernames are compared case-insensitively.
passwordHash	The salted hash of this users' password. The hashing algorithm used is Blowfish, with 1024 passes.
email	The e-mail address of the user.
firstName	The real first name of the user.
lastName	The last name of the user.
affiliation	The users' affiliation (optional).
occupation	(optional)
website	(optional)
homeAddress	(optional)
rank	See below.
activationStage	See below.
banned	Whether this user has been banned by a moderator. If so, they can't log in.
passwordRestoreToken	When a user has clicked the *Password Forgotten* button, this column will be used to store a token that will be included in anURL and mailed to the user. With this URL, they can specify a new password.
registrationDate	The date on which this user has registered.
lastActive	Timestamp of the last moment this user was active.

User rank Different kinds of user ranks (called **User Roles** in the application) should have a different rank value. The higher the value, the higher the rank. A user has permission to perform a certain activity if and only if *Users.rank* $>=$ *Permissions.rank* for a specific user and *Permissions.actionName.*

At the moment the following ranks should be taken into account:

Rank	**Value (as in database)**
Guest	0
Regular User	10
Moderator	40
Administrator	50

See the Permissions table for an overview of what rank a user should be to execute a certain action.

Activation stage The 'Activation stage' of a user determines the moment in the activation process of a user. It can have the following values:

Status	**Meaning**

0	User has registered and is awaiting manual acceptance.
1	User is accepted by the administrator or system, but still needs to activate by clicking the activation link.
2	User is denied.
3	User is successfully activated.

After registration, user enter stage 0 or 1, depending on whether the setting *auto-user-acceptance* is turned on. Only users in stage 3 are allowed to log in.

7.9 External tools and dependencies

7.9.1 Programming Languages

The project is written using three different programming languages, each for a different component:

- The **client-side** is completely written in **Javascript**. No HTML has been written (aside from a single HTML-file that loads the Javascript files).
- The **server-side** is written in **PHP 5.3**. The project is not compatible with older versions of PHP, because we rely on new functionality.
- The so-called **Tile Pyramid Builder** can be seen as a seperate tool and is written in standard **C++**.

Javascript dependencies

All Javascript dependencies are included with the project. The client-side mainly depends on **ExtJS 4**. Futhermore: **JQuery**, **Modernizr** and **SWFUpload** are used.

PHP extensions

Aside from what is delivered with PHP 5.3 by default, the following PHP extensions are used:

- **Imagick** (ImageMagick) is used for creating image thumbnails.
- For database communication **PDO** is used, along with **PDO's PostgreSQL driver**.
- The **mbstring** extension is used for handling UTF-... strings when exporting.
- **Alternative PHP Cache** (APC) is used for caching.

C++ libraries and/or headers

An executable, with its required libraries statically linked in, is delivered with the project. In order to recompile the pyramid builder, the following dependencies are required:

- For reading and writing images:

  - **libjpeg** by the Independent JPEG Group or preferably **libjpeg-turbo**, a 2-4x faster libjpeg implementation maintained by D. R. Commander for the VirtualGL project.
  - **libtiff** from Silicon Graphics.

- For parsing command line options **getopt** is used.
- The C99 header **inttypes.h** should be available.

7.9.2 Database Management System

For database interaction **PostgreSQL** is used. PostgreSQL has to be version 8.3 or higher, the test environments run on version 9.0.

7.9.3 Required tools

- In order to send e-mails, the **sendmail** tool is required and PHP should be configured to know where it is installed.

7.9.4 Browser requirements

The client-side has been tested to work in Internet Explorer 7 or higher and recent versions of Firefox, Google Chrome and Safari. Some functionality might be disabled in older browsers. Additionally, Flash is required for uploading scans.

7.9.5 Server infrastructure

The project should be portable to different server architectures. It is developed under CentOS 5.7 with a Apache version 2.2.3.

7.10 Framework and utilities

Short description of the framework.

7.11 Controller

7.11.1 Summary

Short summary of how our controllers work, possibly with an image or diagram.

7.11.2 Creating a controller

Instructions on how to create a controller (deriving from abstract Controller-class and creating single-argument methods starting with 'action', then use static methods as getString and getInteger to extract data).

7.11.3 Communication

Description of how the received JSON packages are decoded and handled. Contains a link to the front end documentation page describing the Request Manager.

7.11.4 See also

- Controller implementations.

Source

- backend/framework/controller/controller.php

7.12 Database and Query

7.12.1 Summary

In order to communicate with the database we use a custom abstraction layer that is part of our framework. This layer uses the PDO-library and is geared towards PostgreSQL. Throughout the application, all database communication happens through this layer. So in order to port the project to a different Database Management System (after the database has been ported itself) only this layer needs to be modified.

7.12.2 Server class

This is a singleton class wrapping over a PDO-object. It initializes the PDO-object and offers an interface for handling transactions and executing parametrized queries.

The class supports 'nested transactions', meaning a transaction can contain another transaction, behave as expected.

In order to start a transaction, usage of the doTransaction method is recommended: this executes a callback as the transaction and will automatically roll back if an exception has been thrown.

The execute function executes a query: it receives a string containing an SQL statement (which should not depend on user input!) that may contain 'parameter markers' as used by PDO's prepare function. The values of these parameters can then be specified, along with types thereof. These types are strings representing a kind of data, they correspond to both a PHP type and a SQL type in the following way:

Type	PHP	PostgreSQL
bool	bool	bit/boolean
int	int	integer
base64	string (containing binary data)	text (containing said data encoded in base64)
timestamp	int (Unix timestamp)	timestamp without time zone
string	string	some string type (varchar, char, text)

istring	string	same as string, but will be compared case insensitively when queried

The execute function uses a prepared statement for each sequence of queries that look the same (same parametrized SQL string, different arguments).

7.12.3 Query class

The Query class is a so-called 'query builder' that offers an interface to build SQL-queries. Most simple and some more complex operations are supported. In the project, this builder utility is used for every query.

Examples

Delete a user with a certain ID:

```
Query::delete('Users')->where('userId = :id')->execute(array('id' => $userId);
```

Change the status of all scans with STATUS_PROCESSING to STATUS_ERROR:

```
Query::update('Scans', array('status' => Scan::STATUS_ERROR))
      ->where('status = :status')
      ->execute(array('status' => Scan::STATUS_PROCESSING));
```

A complex query:

```
$query =
    Query::select('u.userId')->
           from('Users u')->
           count('id', 'grantTotal')->
           aggregate('MAX', 'maximum')->
           where('username = :username', 'passwordHash = :hash')->
           whereOr('username = :username', 'passwordHash = :hash')->
           join('OtherTable o', array('o.id = u.id', 'o.name = u.name'), 'LEFT')->
           limit(0, 1)->
           orderBy('u.username', 'desc')->
           groupBy('u.name');

$rowSet = $query->execute(array(
    ':username' => $username,
    ':hash' => self::secureHash($password)
));
```

7.12.4 See also

Source

- backend/framework/database/database.php

- backend/framework/database/query.php
- backend/framework/database/resultset.php

7.13 PDF exporter

7.13.1 Summary

The PDF exporter is a utility class that can save scans or bindings and their annotations and/or transcriptions to a PDF version 1.7 compliant file as specified by ISO 32000:1-2008.

It can export, including or excluding transcriptions and/or annotations:

- A single scan with accompanying book and binding information;
- A range of scans in a binding, accompanied with information about binding and book(s);
- An entire binding, accompanied with information about binding and book(s).

The main focus of this implementation is speed instead of functionality: only very simple text and image formatting is supported.

7.13.2 More detailed description

Usage

The only public method of the PDF exporter class is the constructor. As the name says, it immediately 'constructs' a PDF document. However, as the resulting document might not fit in the memory assigned to PHP, the output is directly written to a CacheEntry that is supplied as an argument.

Domain specific assumptions

The exporter makes some assumptions about its input and behaviour of other functions.

- Tiles are rectangular, but may be clipped in one or two directions on the right and lower side.
- External Entity functions that provide lists of Entities do so in a logical order: it expects the scans to be ordered by page number, for example.
- Input text is UTF-8.

Fonts and text

Fonts are stored in the fonts directory in a format that is borrowed from the TCPFD software library by Nicola Asuni. They can be loaded by the addFont() command. The only kind of fonts that are supported are TrueType Unicode fonts, as the annotations may contain several UTF-8 characters. Internally, all text is converted to UTF-16BE and output this way, except for link URIs as those are encoded as a PDF text string.

It is highly recommended to use the default FullUnicodeSans font for text thats may be in any language: this font includes all characters in the UCS-2 character set, and will therfore most likely never show empty characters. As fonts tend to be big (especially the fonts having a lot of characters), only the characters in the font that are actually used are writen. This might consume a bit more memory when processing, though.

Compression

Wherever necessary, FLATE (gz) compression is used to minimize the resulting file size. The images are already compressed by a lossy JPEG compression, so we do not compress those again (that would be a waste of time).

Memory management

As the output files might become very large, depending on the number of scans, output is directly written to a CacheEntry. Writing to a file very often can be expensive, so therefore the exporter allocates an output buffer with a maximum size of PHP's memory limit divided by four. When the buffer is full or the document is ready, this buffer is automatically flushed to the CacheEntry. Only objects that are created as non-final by newObject() are forcibly kept in memory, as they may be updated after being created.

Coordinates

All coordinates and sizes are (unless transformed by an affine transformation) specified in points, equalling 1/72 inch. The coordinate (0,0) is located in the lower left corner, making the upper right corner (pageWidth, pageHeight). By default, the page size is set to A4, equalling 595 by 842 points. Several helper functions exist to transform the coordinate system, such as affine(), translate(), scale(), rotate().

7.13.3 See also

- The PDF controller

Source

- backend/util/pdf.php

7.14 Quick guide to changing settings or help pages

This page contains a quick reference that can be used to inspect or change a setting specified in the database (such as the contents of the welcome page, the name of the project and the terms of service) or the contents of a help page. It describes how to access the database and how to format queries that can be used to achieve reading or changing (but not adding) such a setting. For a more detailed description of how these tables actually work, please refer to the database documentation.

7.14.1 Changing a setting

Changing a setting is simple: simply modify the 'settingValue' associated with the 'settingName' you are looking for. The meaning of settings with a specific name is described here.

The easiest way to inspect or change settings is by using a tool such as pgAdmin to inspect the settings table. Otherwise, you can use a query. An example of such a query that changes the content of the welcome page

```
UPDATE "Settings" SET "settingValue" = '<h2>Welcome</h2>
```

```
<p>Hi!</p>' WHERE "settingName" = 'welcome-page';
```

Obviously, the value of a certain setting (in this case the general 'info' page) can be retrieved like this:

```
SELECT "settingValue" FROM "Settings" WHERE "settingName" = 'info-page';
```

7.14.2 Changing the content of help page

Changing the content of a help page is a little trickier, since there are three different tables involved that maintain the help structure and may indicate certain content is only visible when a certain setting has a certain value.

The content of help page is stored per page (HelpPages) and per paragraph (HelpParagraphs). The text per paragraph is HTML and is stored in the HelpContents table. In order to find the correct HelpContents entry for the part you want to modify you will first need to know the title of the page where the paragraph is on. For instance, when you want to select the text under '<paragraph>' in '<page>', you will need to use the following, relatively complex, query:

```
SELECT "content" FROM "HelpContents" WHERE "helpParagraphId" IN (
  SELECT "helpParagraphId" FROM "HelpParagraphs"
  WHERE "title" = '<paragraph>'
  AND "helpPageId" IN (
    SELECT "helpPageId" FROM "HelpPages" WHERE "pageName" = '<page>'));
```

If the text under 'Result options' depends on a certain setting, there will be multiple results. This is currently only used for the 'Finalizing your registration' paragraph on the 'Register' page though, so it does not need to be taken into account for all other pages.

When changing the content of a paragraph, an UPDATE query can be used that changes the "content" column of "HelpContents" with the same WHERE-clause as the selection query above.

Some paragraphs are contained within another paragraph. If you would want to select '<subpar>', which is a subparagraph of '<superpar>', from page '<page>', use the following extended query:

```
SELECT "content" FROM "HelpContents" WHERE "helpParagraphId" IN (
  SELECT "helpParagraphId" FROM "HelpParagraphs"
  WHERE "title" = '<subpar>'
  AND "helpPageId" IN (
    SELECT "helpPageId" FROM "HelpPages"
    WHERE "pageName" = '<page>')
    AND "paragraphParentId" IN (
      SELECT "helpParagraphId" FROM "HelpParagraphs"
      WHERE "title" = '<superpar>'));
```

7.14.3 Finalizing your registration

There are two versions of 'Finalizing your registration' paragraph on the 'Register' page: one for when the auto-user-acceptance setting is turned on and one for when it is turned off. Let '<auto-accept>' be '1' for the version where the setting is turned on or '0' for the version where it is turned off and the following query can be used to select one of the two versions:

```
SELECT "content" FROM "HelpContents" WHERE "helpParagraphId" IN (
  SELECT "helpParagraphId" FROM "HelpParagraphs"
  WHERE "title" = 'Finalizing your registration'
  AND "helpPageId" IN (
    SELECT "helpPageId" FROM "HelpPages" WHERE "pageName" = 'Register'))
    AND "settingValue" = '<auto-accept>';
```

7.15 Settings and configurations

7.15.1 Configuration properties

The following settings are specified through the file config.ini. These settings are likely to change while migrating to a different server, but should probably remain pretty constant otherwise:

Property name	**Description**
Database	
database-dsn	Database data source name.
database-username	Database username.
database-password	Database password.
User	
password-salt	Salt for password hashing, should be unique.
Session	
session-prefix	Session prefix, should also be unique.
Logging	
debug-mode	Whether to send stack traces and complete error messages to the client.
logging-level	How many messages are logged:0=none, 1=error, 2=warn, 3=info, 4=debug.
Frontend	
minify-resources	Whether to minify stylesheets and Javascript.
Paths	
upload-path	Path where upload files are initially placed.
Pyramid Builder	
builder-path	Path of the pyramid builder executable.
tile-output-path	Path of the directory where the tiles will be stored.
tile-quality	Value between 0 and 100 indicating the JPEG quality of the output files.We recommend setting this to 80.
Thumbnails	
thumbnail-path	Path where thumbnails will be stored.
thumbnail-width	Width of thumbnails.
thumbnail-height	Height of thumbnails.

7.15.2 Settings table

In the database, there is a table called *Settings* which can be used to store various options which should be easy to modify by the system administrator and would probably change more often than the contents of backend ini-files, see also the Database documentation and the Quick guide to modify these settings.

If a setting is visible to the client, its value can be directly requested by the client-side. This option can be toggled with the *visible* column of the *Settings* table.

The following settings are currently specified:

Setting Name	Description	Visible to client
project-title	The title of the project, now "Annotated Books Online".	Yes
mail-from-address	The address used as a FROM-address whensending an e-mail.	No
activation-mail-subject	The subject of activation e-mails to newly registered users.	No
activation-mail-message	The message of the activation e-mail send to new users. Should contain the string [LINK] somewhere to indicate the place the activation link should be inserted. [USERNAME], [FIRSTNAME], [LASTNAME] and [PROJECTNAME] can also be used. The latter indicates the name of the project.	No
forgotpass-mail-subject	The subject of the mail send to users who forgot their password.	No
forgotpass-mail-message	A mail message similar to that when activating. Should contain[LINK] and may also use [PROJECTNAME], [USERNAME], [FIRSTNAME] and [LASTNAME].	No
welcome-page	The text displayed on the welcome page, in HTML.	Yes
info-page	The text displayed on the info page, in HTML.	Yes
upload-insructions	The text displayed before uploading, in HTML	Yes
terms-of-use	The terms of use you can view when registering.	Yes

auto-user-acceptance	Set to "0" when users should be manually activated by administrators.If set to "1", users can activate themselves.	Yes
user-declined-mail-subject	The subject of mails send to registered users who are declined activation.	No
user-declined-mail-message	The message of a user declined message. Works the same asforgotpass-mail-message.	No
deleted-user-id	The row ID of a special user used to represent all deleted users.When a user is deleted all foreign keys pointing to that user will bereferred to this.**This should not be changed.**	No

7.16 Tile Pyramid Builder

7.16.1 Summary

The tile pyramid builder is a C++-application that takes a (huge) JPEG or TIFF image as input and starts by cutting it in pieces (tiles) of 256x256 pixels (other tile dimensions can also be specified). Subsequently, quartets of neigbouring tiles will be combined and scaled to new tiles of 256x256 pixels. This continues until a 'pyramid' has been build with, at the top, one tile showing the scaled whole image. At each level, it is possbile to 'zoom in' on one of four tiles. At the highest 'zoom level' you will find tiles with the resolution of the original image.

The Tile Pyramid Builder is intended as a command-line tool that will be called by a PHP-script that is processing uploaded scans. The resulting pyramid of tiles will be used by the viewer. The reason that we used a custom application for this purpose rather than an existing image manipulation program is that we have to deal with very large images that would take up a lot of memory if they were loaded entirely, and most existing tools and libraries (like Imageshack) don't handle this well. We instead read the input images by scanline and throw away (or write to disk) any data we no longer need, minimizing memory usage.

The PHP script executing the builder will be runned every few seconds by a cronjob. When it sees that a new scan is uploaded, it will execute the builder with the scan file as input. If the builder terminates succesfully, the tiles will have been stored in the file system. The tiles can now be used by the viewer, which can determine which tiles it needs by looking at the filenames.

7.16.2 Tool usage

The compiled builder tool can be called through the command line. The following options can be specified:

Option	Default value	Description
-i <type	'auto'	The type of the input image, can be either 'jpeg', 'tiff' or auto. In the latter case the file type is derived from its extension.
-p <path>	'.'	The path in which the tiles to be created will be stored.
-f <string>	'tile_%z_%x_%y.%e'	A format describing the filenames output tiles will be given. %z will be replaced by the tile's 'zoom level' and %x and %y by its coordinates.
-q <num>	'60'	The JPEG quality of the tiles that are created. Should be a value between 0 (worst) and 100 (best).
-w <num>	'256'	The width of the created tiles. Should be a power of two.
-h <num>	'256'	The height of the created tiles. Should be a power of two.
-u	off	Whether to pad tiles on the edges that end up smaller than the specified dimensions.
-c	'#000'	If padding is turned on, tiles will be padded with pixels of this color.
–help		Display a description of these options, similar to this one.

If the builder fails for some reason, it will print an error message to stderr and terminate with a nonzero return status.

7.16.3 The tile building process

The builder starts by reading an amount of scanlines from the input image that is equal to the height of the tiles to be created. This immediately creates the upper row of tiles on the highest zoom level, so those will be written to disk as compressed JPEG's.

Then the following row of highest level tiles will then be read from the input image. After writing these to disk, pairs of four tiles can be combined and scaled. Since the dimensions of tiles are powers of two: scaling can easily be accomplished (and without any artefacts but a decrease in 'sharpness') by taking the average colors of each four pixels.

The builder continues reading chunks of scanlines, immediately storing and combining tiles if possible. Whenever one tile on zoom level x is absorbed by a tile on level x - 1, its raw image data can be deallocated from memory since the level x tile has already been saved on disk and is no longer necessary. This way, memory usage is minimized. Note that the wider the input image is, the more memory is required, while the height of the image has almost no influence on the maximum amount of memory used at any time.