Additional specifications

Consider a password.

Consider auto-save options. It should auto-save every 5, 10, or 30 minutes.

URLs should be made relative to the saved location. If a location is specified for the first time, you should painstakingly go through each URL in all project data and make it relative if it’s not empty or null.

After construction: printing support, sharing (export/import all levels), and autofill chemical formulas and species group if possible; auto-suggest names (both from stored local database), undo/redo maybe, and finally a CHM help manual.

Importing will be done with *load*, except non-database files are added to the database instead. Exporting collection exports the template it uses with it. Exporting group exports the collection with the template it uses, but none of the rest of the entries not included. Exporting entry exports the collection with its template and only one entry.

The search box filters as you type with each character for the current group. If you start searching in a collection directly, it chooses the ‘all’ group.

A columns widget is needed to display two columns with arrows between them so you can move content between.

~~An image rotation widget is needed to rotate cyclically on hovering and let you drag to go left/right every 50ms.~~

~~An images display widget is needed to show an image that changes every second.~~

~~An image viewer widget is needed to view a single image in full scale. Arrows appear on the sides to let you flip through images in a series one-by-one, if it’s loaded with more than one image.~~

When you make default tags for Dad, include these as text: primary mineral species, secondary mineral species, primary chemical formula, species group, origin / location, gps location, notes. Includes as text, text: market value.

Display an error message if content cannot be displayed or retrieved and hide empty fields when viewing. The error message should allow you to locate and link to the content again.

Use arrowkeys to navigate tree. They auto-expand and auto-fold relevant sections. Press delete to delete them (with confirmation dialog). Double-click in the treeview to rename something. Each database, collection, grouping, and entry must have visible characters to accept a rename (or to be created). ~~When created, a GUID is generated and checked against others for uniqueness. Making a new collection, grouping, or entry updates the treeview and auto-selects it.~~

If you become Neo, have an internet forum to post all of this stuff. Enjoy server costs.

~~Have it so the initial page is a logo of the application~~ with links to recent databases, if any, displayed directly. ~~When a project is opened, switch to the normal view. When it's closed, go back.~~

~~The .mdat file format stores a database and everything in it. Everything will be loaded in memory except images, which will have urls recorded and accessed as needed. This file format is split into different types of chunks, which are encoded in a byte:~~

~~0 = template~~

~~1 = template field~~

~~2 = template column~~

~~3 = collection~~

~~4 = grouping~~

~~5 = grouping entry reference~~

~~6 = entry~~

~~7 = field~~

~~The header of each chunk must be uniform so the system can read them first without having to know their types.~~

~~ulong next length in bytes~~

~~byte Chunk type~~

~~ulong GUID~~

~~File format .mdat structure:~~

~~- int next string in chars~~

~~- string application version number (major.minor)~~

~~- int next string in chars~~

~~- string database Name~~

~~- bool Use a background image or not (no stretch or tile)~~

~~- uint next string in chars~~

~~- string The image url~~

~~Any number of templates.~~

~~Template format:~~

~~- HEADER~~

~~- int next string in chars~~

~~- string Template Name~~

~~- bool Center main images on top. Ignores column order if true and centers at top.~~

~~- bool Use two columns or not.~~

~~- byte The number of extra images to display.~~

~~- byte Where to position extra images. 0 = right, 1 = under, 2 = left, 3 = above.~~

~~- uint next string in chars~~

~~- string Font name~~

~~- byte r Font title color~~

~~- byte g Font title color~~

~~- byte b Font title color~~

~~- byte r Font content color~~

~~- byte g Font content color~~

~~- byte b Font content color~~

~~Should be followed by one or two template column order objects:~~

~~- HEADER~~

~~- bool Whether this list is for the first column.~~

~~- ulong reference to template GUID~~

~~Should be followed by any number of template fields:~~

~~- HEADER~~

~~- int next string in chars~~

~~- string Field Name~~

~~- ulong reference to template column GUID~~

~~- byte Type of data. entry name; entry images; text; image; webpage; map; text, text; text, text, text.~~

~~- bool Whether the field is visible or not.~~

~~Any number of collections, followed by template ID and any no. of groupings, followed by any no. of entries.~~

~~Collection format:~~

~~- HEADER~~

~~- int next string in chars~~

~~- string Collection Name~~

~~- ulong template GUID This GUID is a reference to an existing template ID.~~

~~Grouping format:~~

~~- HEADER~~

~~- int next string in chars~~

~~- string Grouping Name~~

~~- ulong reference to collection GUID~~

~~-int Number of Conditions~~

~~- byte condition type~~

~~- string name1~~

~~- string name2~~

~~- ulong field GUID~~

~~Should be followed by any number of entry references:~~

~~- HEADER~~

~~- ulong reference to containing grouping GUID~~

~~- ulong reference to an entry GUID~~

~~Entry format:~~

~~- HEADER~~

~~- int next string in chars~~

~~- string Entry Name~~

~~- ulong reference to containing collection GUID~~

~~Should be followed by any number of data fields:~~

~~- HEADER~~

~~- ulong reference to containing entry GUID~~

~~- ulong reference to template field GUID~~

~~- RAW DATA (as a string)~~

~~When loading the data, it reads each section and constructs a representation of the hierarchy. All positions in the file are stored for each chunk. All data is read except RAW DATA and used to construct the Collection and Grouping objects in full.~~ When a collection is opened, the first three entries in each group are identified and a rotating image display is constructed for each one. When a grouping is opened, the rotating image display from the active collection is destroyed. The necessary entries are identified in the hierarchy and a rotating image display is constructed for each one. Thumbnail images are displayed in the treeview if image view is on. When an entry is opened, it’s identified in the hierarchy and all of its data is loaded and displayed according to the template. When another grouping or collection is opened, existing thumbnail / image display widgets are destroyed. However, if it’s a grouping being opened, overlapping data is preserved. If you switch from text to image view, thumbnails are constructed if they don’t exist.

Entry data is read by jumping to the location specified for the filestream in each hierarchy object, then reading the raw data and interpreting it based on the template’s specified field type for the data. For text, this means converting bytes to Unicode (2-byte) strings. For images or video, this means attempting to load from a local file url. For urls, this means constructing a web widget and searching for a cached copy of the url (and otherwise trying to load one). All loading should be asynchronous and in alphabetical order. For entry names, a string is made. For entry images, the frames are collected into an array and given to a rotating image widget.

After all data blocks of the current entry are specified, the next entry is found, if any. If not, the file has reached its end.

Example .mdat:

template 1...

template 2...

collection minerals (template #1)

grouping1, 2, 3, etc.

entry1, 2, 3, etc.

collection notes (template #2)

no groupings

entry1, 2, 3, etc.

collection localities (template #1)

grouping1, 2, 3, etc.

entry1

Example .mcol:

template 1...

collection minerals (template #1)

grouping1, 2, 3, etc.

entry1, 2, 3, etc.

Example .mgrp:

template 1...

collection minerals (template #1)

entry1, 2, 3, etc. Only the group’s minerals are stored.

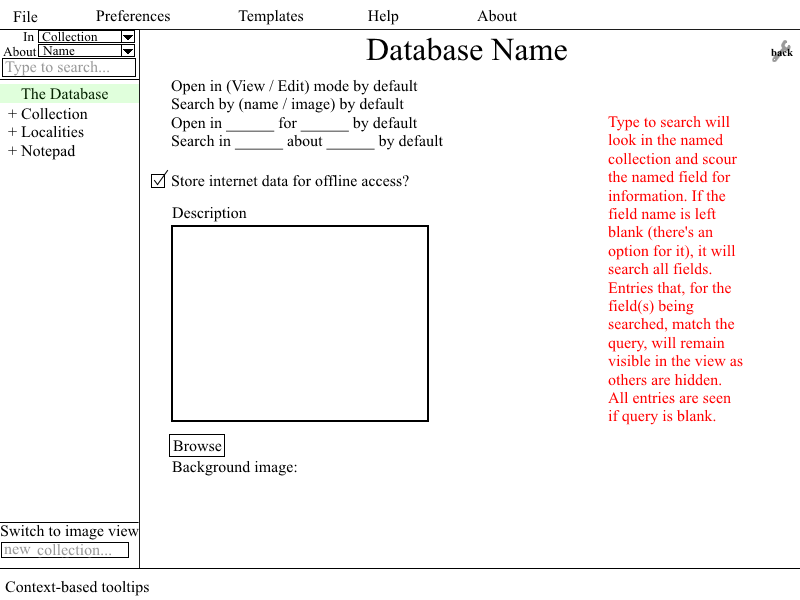
Example .ment:

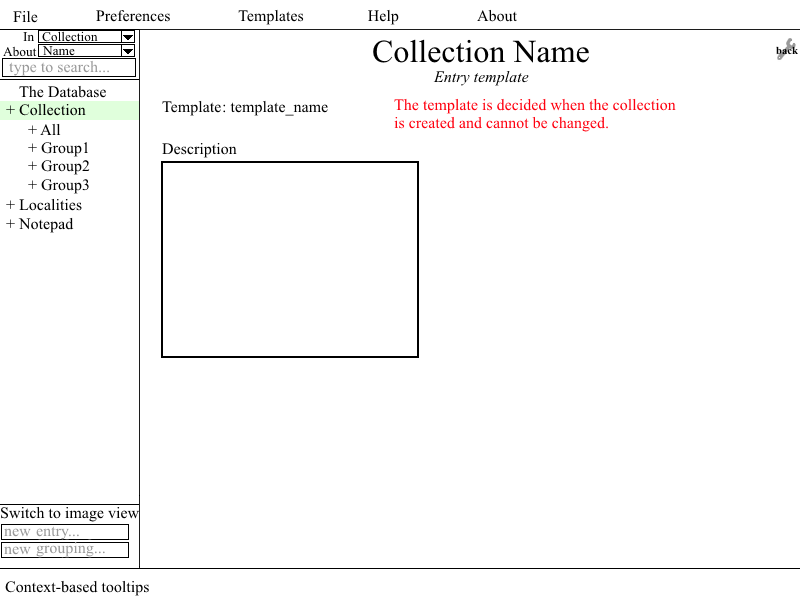
template 1...

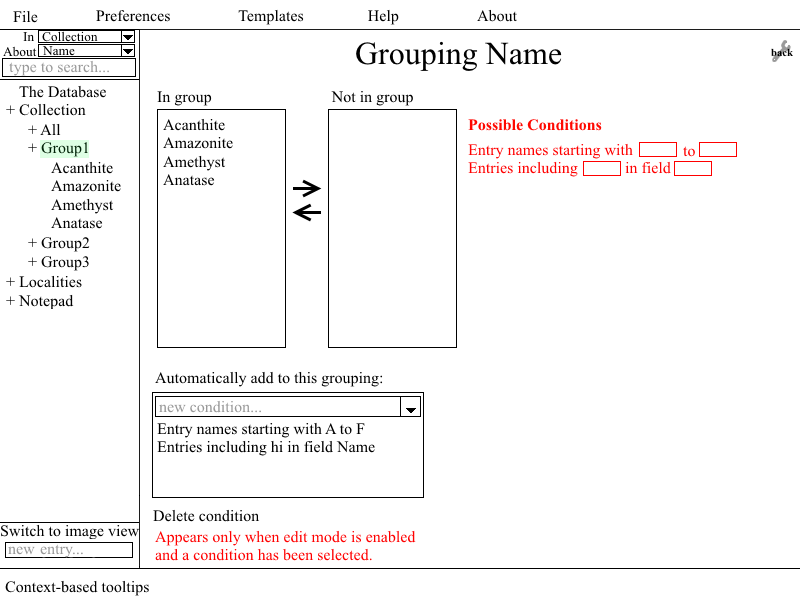
collection minerals (template #1)

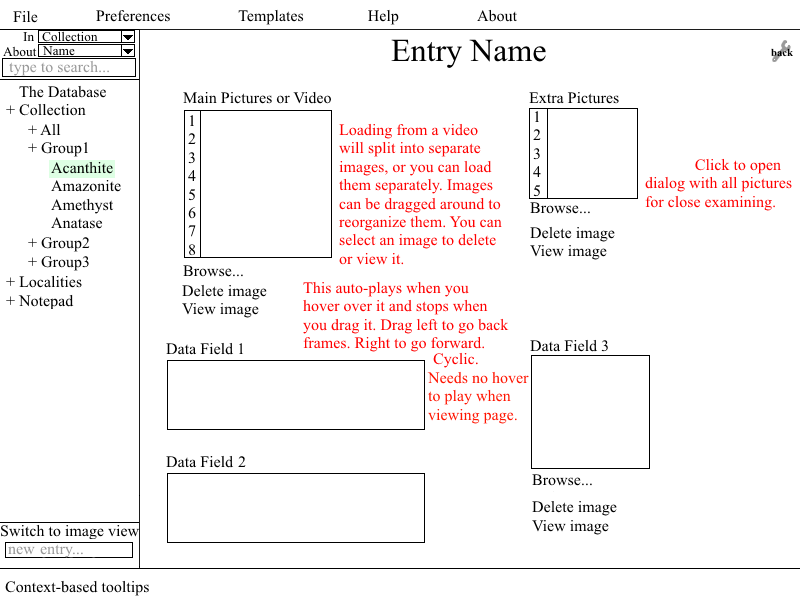
entry1

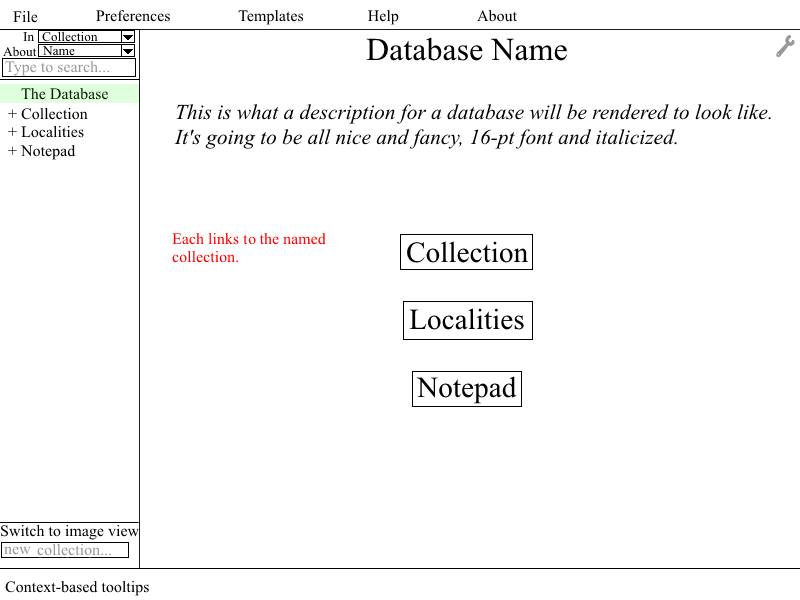
GUI Design

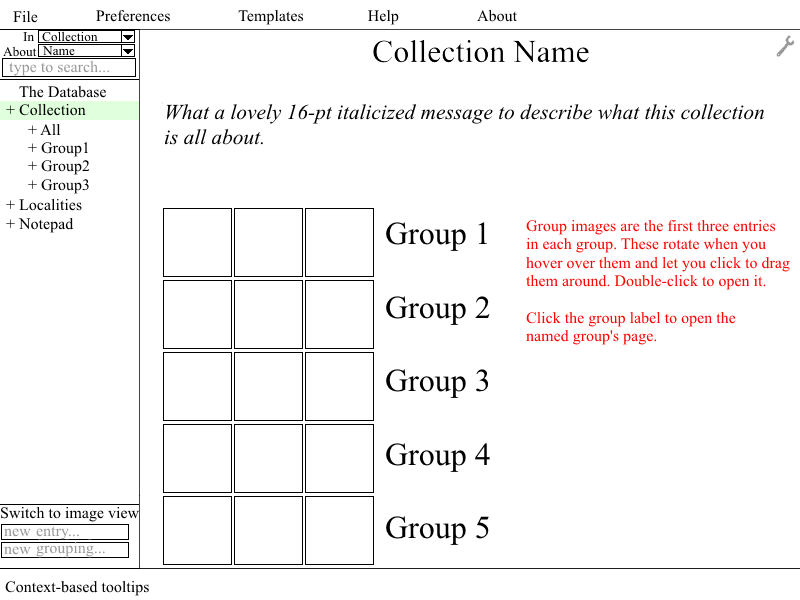


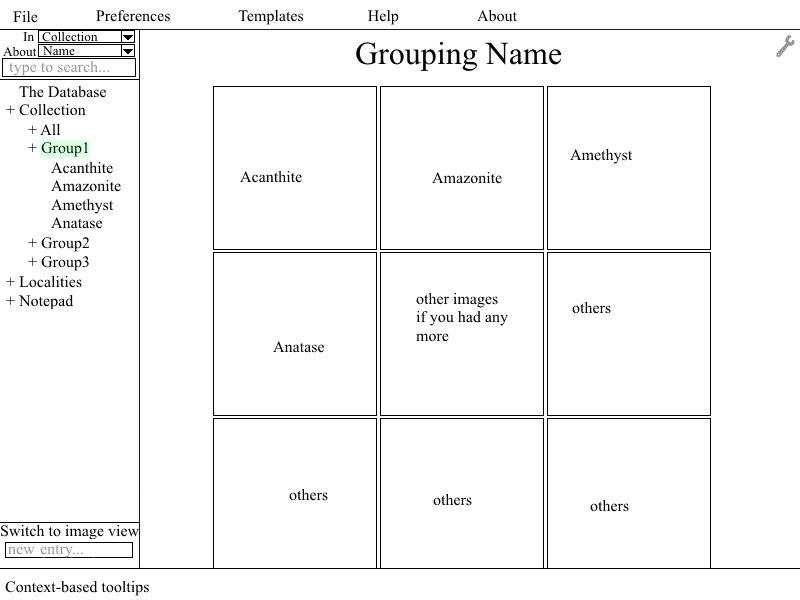


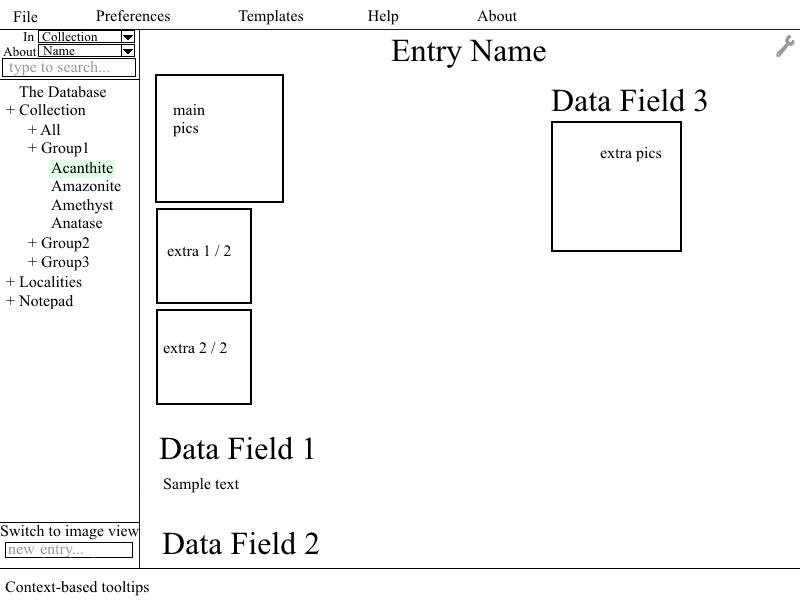




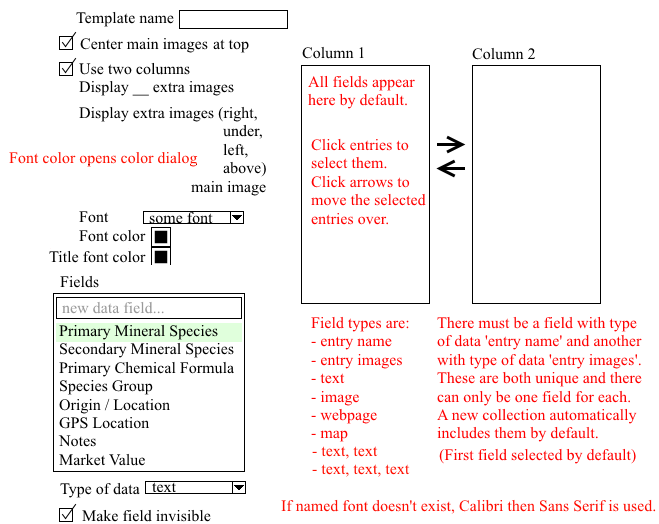




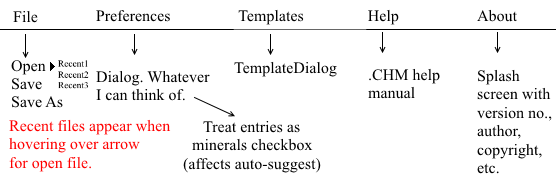




The template dialog follows.



The menubar follows.



The hierarchy pane follows for image view being enabled.

