**Nikolah Pearce** - 6377167, **git:** nikolahpearce | **Vivian Breda** - 1663266, **git:** vbreda
**Group name:** viviannikolah

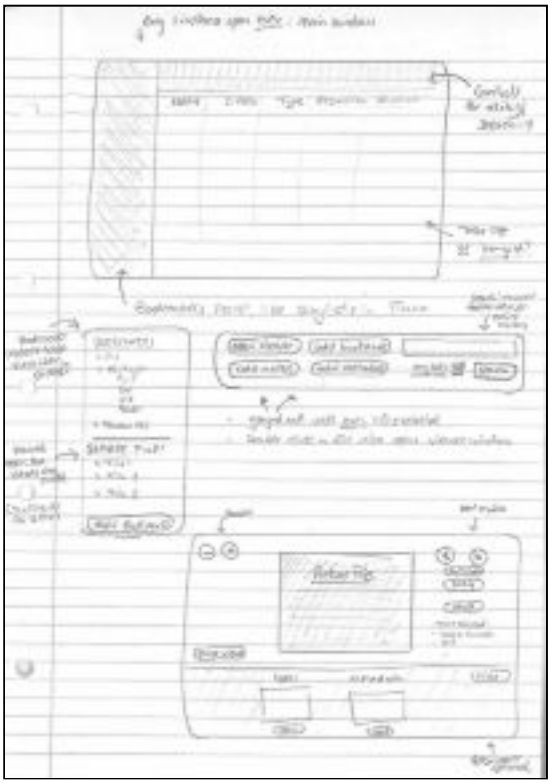
# 346 Assignment 2

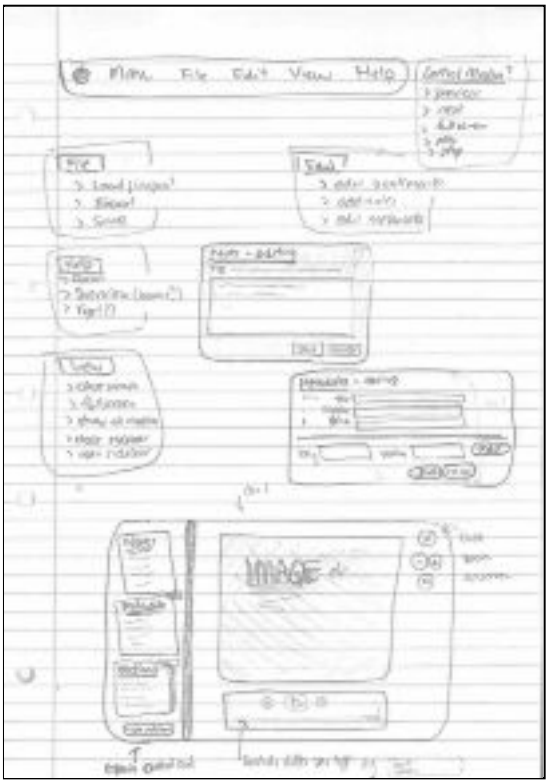
## Foreword:

Our jsonData.json file (which we were using for testing) is expecting media to be located in home > 346 > media e.g. the file paths are specified as ~/346/media/image1.png

## Design:

We designed our Media Library to be elegant, simple, and pleasing to look at/use. Our windows and views all stick to the same minimalistic theme. We started with understanding the minimum requirements, and sketching a UI that met those goals. Normally, we would have the goals of the user in mind when creating our design, but here we treated the minimum requirements instead as the user's 'goals'. Our initial designs looked like:





- One window with Bookmarks view left, Library table view right
- Another window (can open many) with Viewing Media in it.
    - Uses specific View Controller per type of file
- App Delegate file simply opens the main window and handles opening/closing
    - Main window opens the viewer windows

We tried to stick the MVC pattern as much as possible. We started by creating a model that was based on our previous main.swift file - handling the collections and the commands that act upon the collections. Our bookmarks View Controller and our library View Controller both

rely upon the model. Two delegates were created that would allow the the model to tell the view controller's when they each respectively needed to update. A tableDataDidChange() method is called within each view controller whenever the bookmarks are updated or the library runs a command that changes the 'results' to be shown e.g. import or search.

We added small but helpful features to assist the user, such as mapping the Previous and Next buttons to left arrow key and right arrow key respectively. You can also choose Command + S to save the notes when you are editing. Many of our menu items also have shortcuts attached to them.

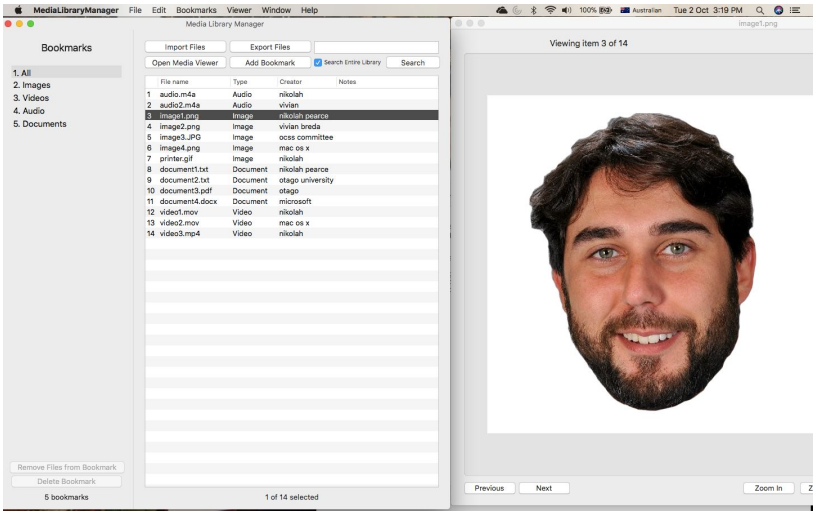
## Marking

### Minimum Functionality (60 marks):

1. Navigation within files done e.g. Video can play, pause and change time
2. Navigation between files and jumping to files implemented via bookmarks.
3. Specific information showed about file displayed, via Details (metadata) and Notes.
4. Media files zoom to fit the window when being viewed, e.g. if you expand the window, the media will zoom in, and if you make the window smaller it will zoom out.
5. Controls resize when the windows resize, but never resize too small so as to 'break'
6. Notes are implemented and can be recorded on each file.
7. Bookmarks can be made for groups of files or individual files.
8. Search within the whole library works, as well as within individual bookmarks.
9. Menu structure has items that complement the button layouts, however these are quite basic and we would improve them in future.
10. About panel shows and is customised to our project.
11. Import and export of files works.
12. All of the minimum required file extensions open and work, including some extras.
13. There is a small testing framework.

Note: the requirement "When presenting the media content it must be possible to at least view the notes, and use bookmarks, even if at that time notes cannot be edited and bookmarks cannot be changed" has in fact been fulfilled within our design. See the below screenshot, where media can be viewed and bookmarks still interacted with:

For example, there is nothing stopping a user from arranging their windows as below:




Though the bookmarks remain in the main window, they are completely still accessible and viewable when presenting media - including usable.

## Added Functionality (40 marks):

- Our application has its own icon. This is worth 3 marks.
- Five automatic bookmarks (topics) are created whenever a user imports files, and are kept up to date when they import more.
    - All, Images, Audio, Documents and Videos.
    - This is worth 5 extra marks.
- More than basic functionality has been added to the bookmarks, e.g.
    - Can delete bookmarks made by the user.
    - Can remove files from a bookmark made by the user.
    - Warns a user if they are to overwrite an existing Bookmark name, and gets their response before proceeding.
    - Prevents the user from deleting or removing files from the five automatic “permanent” bookmarks.
    - This is worth 4 marks per point above, so 16 marks total.
- Status labels appear at the bottom of each table to inform the user of helpful statistics about their library and bookmarks. These update whenever the table or selection does.
    - This is worth 3 extra marks.
- More than the minimum media file extensions are openable by the user
    - Documents: PDFs open.
    - Errors are thrown to inform the user the .docx throws cannot open.
    - Images: JPG and GIF open.
    - This is worth 5 marks.
- Fast search processing of the media files by topic and by type.
    - Fast search was implemented in assignment one, by using dictionaries to store metadata and searching on these instead of plain old linear search.

    - This is worth 5 marks
- When a user searches for a term that does not exist, the table will display 0 files to reflect that the term could not be found.
    - This is extra because as per the assignment one format, it simply threw and would not then update the table.
    - The user can then search for an empty string to return to “all” files.
    - This is worth 3 marks.
- When a user closes the application with the right-hand red cross:
    - The application prompts the user if they would like to save the state of their library.
    - The application also fully quits instead of annoyingly hanging around.
    - This is worth 3 marks each, so 6 marks.
- When a user opens the application for the second time, e.g. with some previously saved state:
    - The user is prompted whether they want to load the previous state or continue with a blank empty library.
    - If the user chooses to load, all files are restored from the previous state*.
    - Notes are stored in the state, so have been implemented as persistent**
    - This is worth 10 marks.

Total extra marks: 3 + 5 + 16 + 3 + 5 + 5 + 3 + 6 + 10 = 57. We understand that this number of marks is more than available to give out, so we argue that we should be awarded the maximum allowed bonus marks.

** the non-permanent bookmarks are excluded from saved state. None of these will be saved/restored.*

*** we utilised the already in place JSON encoding/decoding - simply adding a ‘notes’ field to our import/export struct and storing this in our File. As such, our testing JSON data has been adapted to match this format, e.g. {fullpath:" ", type:" ", notes:" ", metadata:{ creator:" "}}. If you would like to test on your own files, add another field for notes, directly before the metadata.*

## How It Works

When you open our Media Library Manage for the first time, the window will pop up with an empty table and will show no Bookmarks - the only available feature is the “Import Files” button. By clicking this button, a new window will show up, allowing the user to navigate through their existing folders and files so they can choose the JSON files that they would like to work with (note: they can do so by double clicking the file or pressing the “open” button on the bottom-right hand corner of the window).

At this stage, the Media Library Manager will now show all the files contained in the chosen JSON file, the Bookmarks’ navigation is updated with the 5 automatic bookmarks previously mentioned in the “added functionality” section, and all the buttons will become available to be used (some will still need files to be selected in order to be used though).

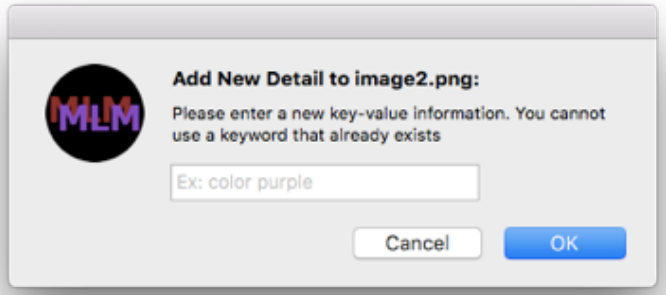
When the user opens a file (by either double-click, the “Open Media Viewer” button, or by selecting the file and Menu > Window > Open Viewer), a new viewer window will appear, where they can then adjust the window size as they wish (note: all the components, including the media viewer, will nicely follow the window resizing), add to/delete from the file’s “details”, add/edit notes, and move through the files. In this window, we chose to follow Apple’s patterns and use the “+” and “-” buttons to add to and delete from the “Details” panels, since that seems to be intuitive to macOS users.

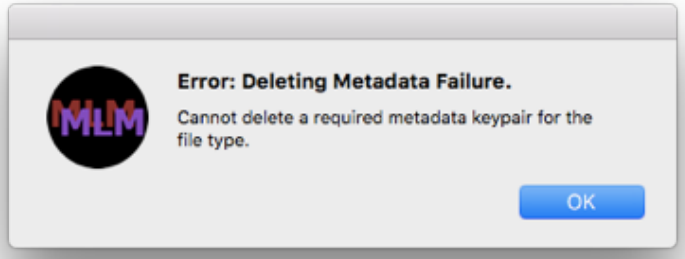
We would like to note that the user can also perform all of the above actions using the menu bar, under the “Media Viewer” option, and use keyboard shortcuts instead of the “next”, “previous” and “save notes” buttons (these are the common macOS shortcuts for said actions). In addition, multiple files can be opened, each on its own window, and the menu bar will work only with the one that is currently selected.

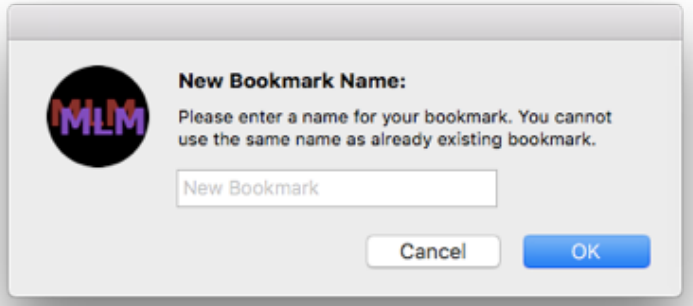
Back to the main Media Library Manager window, the user can then choose to import export their current library (with edited notes and metadata) to a new JSON file if they wish to. Alternatively, before quitting the application, they will also be prompted to choose whether they would like to save the current state of their files so they can continue working on them in the future. When they next open the application, they will also be prompted if they would like to reopen the last saved state, or simply start afresh. This prompt only appears once they have saved some state.

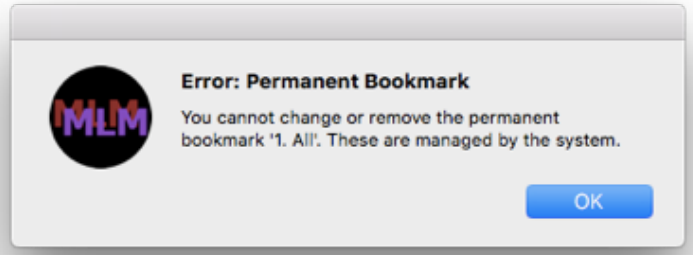
The following pictures are examples of the dialog boxes that we use to communicate with the user when things become complicated and/or we need some form of input in order to proceed with an action:

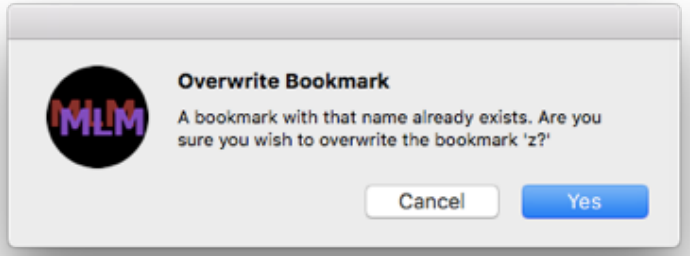
**Nikolah Pearce** - 6377167, **git:** nikolahpearce | **Vivian Breda** - 1663266, **git:** vbreda
**Group name:** viviannikolah

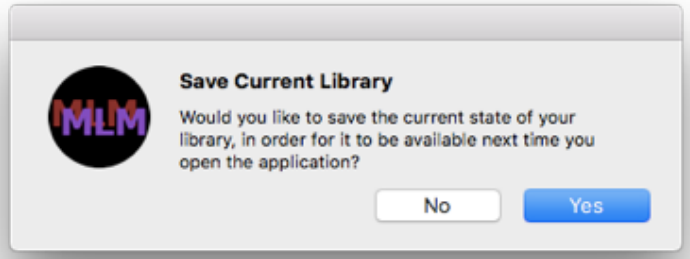


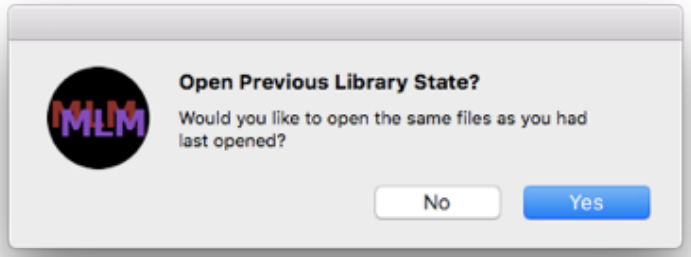



## Testing

We did manual testing throughout. Whenever a new feature such as a button or a textfield was added, we would use print statements to check it was connected and called when expected. To aid us in the testing, we commented out the code about the NSOpenPanel, allowing us to auto-load the same file each time, instead of having to select. Manual testing allowed us to test many different areas of the application, such as resizing and checking that visually things looked as expected, and clicking through the entire bookmarks to see each media displays properly. We added minor Unit Tests that test the Library Model can import and create/delete Bookmarks as expected, but would have made many many more should time not have been so constrained. Ideally, we would have even made some UI tests.

## Roles

Nikolah initially did the draft designs, and created the Main Window that displays the split view. Vivian worked to find resources that would teach us the functionality of implementing the designed features. Together we got our basic bookmarks, and table view working. Next, Nikolah worked on the finishing touches to our main window, while Vivian implemented the media viewer windows, and got the various media files displaying. Again, we together worked on the design of things, communicating ideas as they came and implementing things only once discussed as a team. We both worked on the report and the testing framework.

## References:

- 'AppleProgramming' on YouTube, specifically: *https://www.youtube.com/watch?v=Xy0-x-Zf1YE*
- Ray Wenderlich's blog, specifically: *https://www.raywenderlich.com/830-macos-nstableview-tutorial*
- Marc Fearby's question and answer on Stack Overflow, specifically: *https://stackoverflow.com/questions/28362472/is-there-a-simple-input-box-in-cocoa*
- 'The Swift Guy' on Youtube, specifically: https://www.youtube.com/watch?v=7iT9fueKCJM
- Antoine's question and answer on Stack Overflow, specifically: https://stackoverflow.com/questions/38016143/how-can-i-delete-derived-data-in-xcode-8/42760524
- Swiftrein.blogspot.com for the menu item instructions, specifically: https://swiftrien.blogspot.com/2015/04/adding-menu-items-and-their-actions.html