COSC346 Assignment 2

Group: DJ-Rommie
Members: Daniela Lemow (2926484), Megan Seto (1227523)

Application Testing Setup

Once the application is running, click the import button. Navigate to the folder where you have cloned our project to, and find the file named “library-contents.json”. Choose this and import it. You will now have files to test our application with.

Research/UI Design:

- We looked at applications with similar functionality (Spotify, iTunes, Safari, Finder) to our application to see what conventions seemed to be universal. We chose to follow these conventions in order to make our application as intuitive for a new user as possible. For example, in many applications, bookmarks/playlists are shown in a sidebar on the left of the screen (as shown below). Because this is a recurring feature, we decided to implement this, and have a panel that will have “bookmarked” items for quick and easy access. ***Refer to Figure 1 and 2 below.***

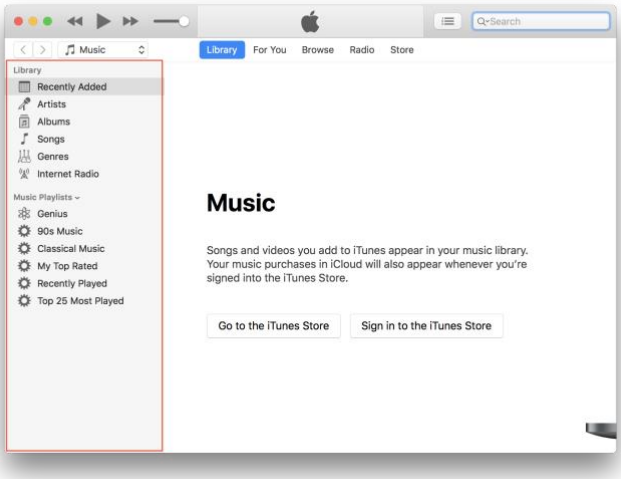


Figure 1: *iTunes UI*

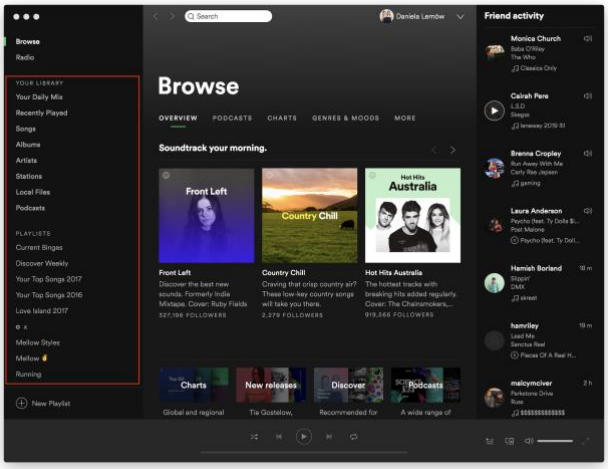


Figure 2: *Spotify UI*

- One thing we decided to think about was being forgiving to the user, especially in regards to the deletion of items. Anywhere where a delete button exists, we decided we would put an alert to double check that the user really did want to remove the item.

Implemented Features:

- ***"Users can navigate through a given media file, including moving forward and backward for video file, moving to the next image in the same folder, moving to the previous image, and jumping to a specific media file while showing a list or grid":*** We found this the hardest feature to decipher in terms of wording. We believe we have implemented it through our content view, which shows a list of the files in the library. They can be scrolled through using arrow keys, or any file can be jumped to by clicking on it.

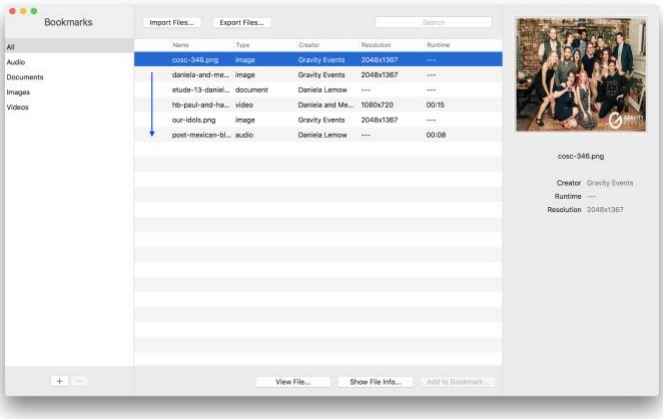


Figure 3: *The content view for navigating between files while showing a list*

- ***"Users can navigate between different media files within a set, providing both "local" navigation, such as "next media", "previous media":*** In the "FileViewer" window, users can navigate through the imported media using the "*Previous"* and "*Next"* buttons. Refer to ***Figure 4 below.***

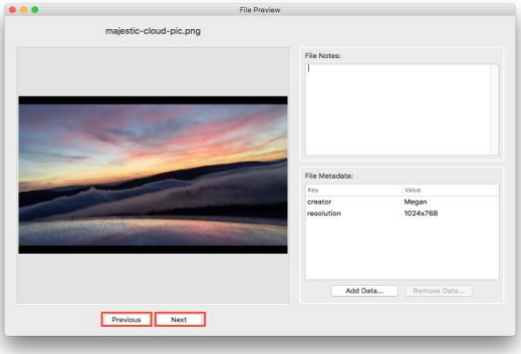


Figure 4: *The Next and Previous buttons used to navigate through a set of media files*

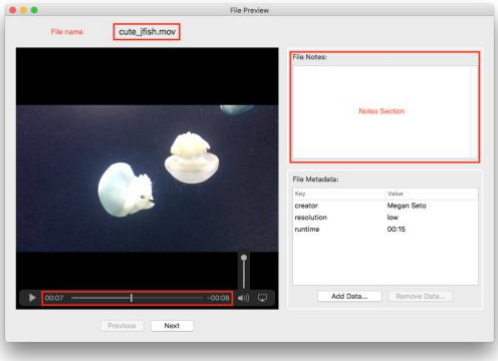


Figure 5: *Example of file identification, notes and current runtime*

- ***"An indication is provided of the current media file being viewed, and the current time for example for the video type with the specific notes":*** For each different file being viewed in the FileViewer, the name of the file being viewed is displayed at the top of the window. A box containing the associated notes for a file is shown in the upper right-hand corner. For playing videos, the run time and current time were inherent features therefore we did not think we needed to implement any additional feature. Refer to ***Figure 5 above.***

- ***"You application provides functions for zooming in, zooming out and zooming to fit the media document contents into the window/Application's controls should resize in a sensible manner when its containing window is resized" :*** When a piece of media is opened in projector view (by pressing the "View File..." button), the window can be resized, and this resizes the media content appropriately. This is how we interpreted this instruction.

- ***"The application's controls should resize in a sensible manner when its containing window is resized":*** The Main View resizes appropriately (we're pretty happy with the constraints we managed to place on it). Because the "Show File Info" view is only used for editing/viewing notes associated with the file and editing/viewing metadata, and to give a reminder of what the file is (it is visible on the left), we decided to make this fixed size. We think this is reasonable as this view's intended purpose is editing data, not viewing the file. When the "Projector view" is opened for the file, the window can be resized, and this resizes the media content appropriately.

- ***"Users can record brief textual notes that are related to a particular media file":*** There is a side panel on the File Preview where users are given the opportunity to record notes if they wish to. Furthermore, these notes are persistent. When the application is shut, and then opened again, the notes remain with the file.

- ***"Users can bookmark particular topics of media or bookmark a particular media file then later use these bookmarks to jump back to appropriate topic or type of media file":*** You can create a new "bookmark" and add imported files in to them. The app is set up so that bookmarks for the different file types (Audio, Document, Images, Videos) are automatically created and populated with the appropriate files based on their media type. Automatically generated bookmarks cannot be deleted but user-created bookmarks can be. You can add one or more files to a bookmark at a time. ***Refer to Figure 6 below.***

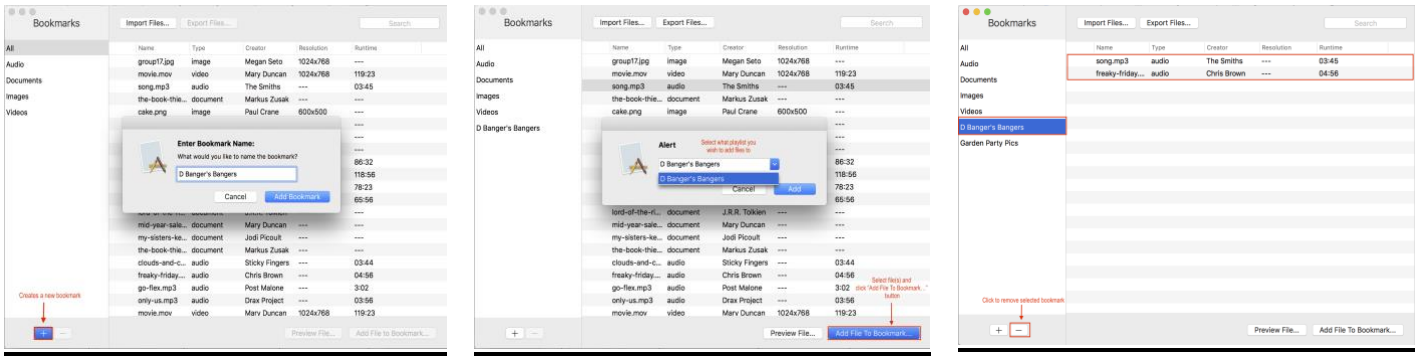
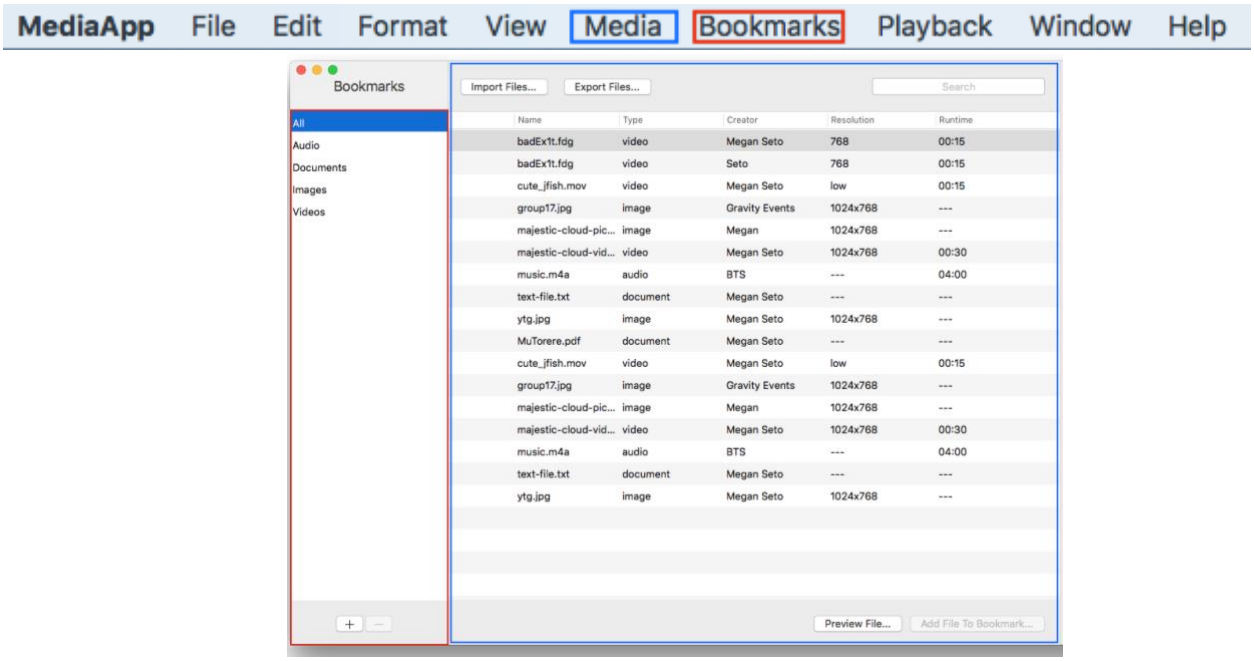


Figure 6: *Photo series shows how to create a bookmark and add files to bookmarks.*

- ***"Search within a set of media files related to a particular topic or particular type of media file":*** You are able to use the search bar in the top right hand corner of the application to search on file metadata. To filter by media type while searching, you must have the corresponding type bookmark selected from the left hand bookmark view.

- ***"A useful menu structure is implemented, that complements your other user interface controls":*** The menu bar is somewhat functional. The way we have set up our window is that 3 views are on the main window. In general, the drop down items under the "Media" menu items (such as import files, export files) will only be applicable if the user is currently clicked on the Content View (outlined blue), whereas the "Bookmark" menu items (add bookmark, remove bookmark) will only be available if the user is currently on the Bookmark View (outlined red). We found the menu really difficult and opted out of doing it a little bit. This is something we would look to improve in the future.



- ***"About" panel is customised to produce relevant information about the project:*** We included a Credits.rtf file in our project and entered information we thought was relevant to include in the about. This file populates the about panel for the application when it is run.

- ***The same import/export function functionality for assignment one should be a feature:*** Files can be imported and exported using the "*Import Files…"* and *"Export Files…"* buttons on the Content View. We have set it up in such a way that only files with a .json extension will be able to be imported. Additionally, users are able to select multiple files to import or export at once. See **Figure 7 and 8** ***below.***

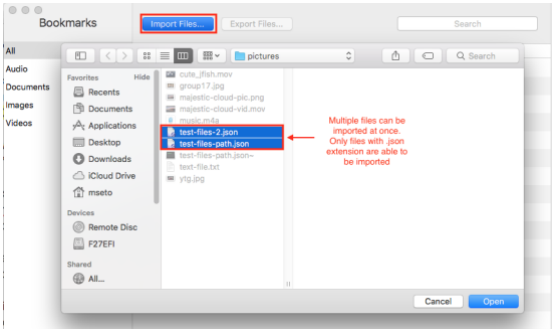


Figure 7: *Multiple files have been selected to be imported. Only .json files are available*

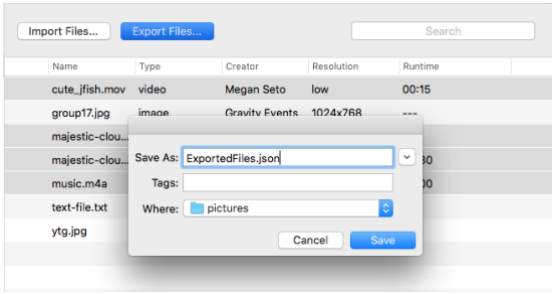


Figure 8: *Multiple files have been selected to be exported*

- ***Files with the extensions .txt, .mov, .png, .jpg, .pdf and .m4a can be opened and previewed:*** All of the mentioned file extensions are functional. Additionally, we also made it so that you can view pdf files. In the instance that the file format is not supported, the view is set to a default image, as pictured below. Even if the files format isn't supported, the files metadata is still available to be viewed.

Additional Features

- ***An alert appears when a json file fails to import*** - while some notifications can be annoying for users, we decided this alert was important. Importing files is an important feature of our app, so we thought it would be appropriate to alert the user when certain files failed to import and why, opposed to them wondering why it did not import or even worse, having them search for a file that does not even exist in app to begin with. An example of this error can be seen below in ***Figure 9***.

Figure 9: *Example Import Error alert*

- ***Media file preview "Image Preview" (right hand split)*** - We thought that preview split would be a useful feature to implement, this preview was mainly introduced with previewing images in mind. For images with auto-generated names (ie, IMG_5260.JPG) we thought that it would be helpful for users to have a preview of the photo to help them easily identify whether or not it was the image they were looking for. Documents that have the extensions *.txt* or *.pdf* can also be previewed. For media types that are not images or documents, the default "No Preview Available" will be shown in the image well. For videos, we intended to pull a screenshot from the video and set the image to the preview, but this was too difficult. ***Refer to Figure 10 and 11 below.***

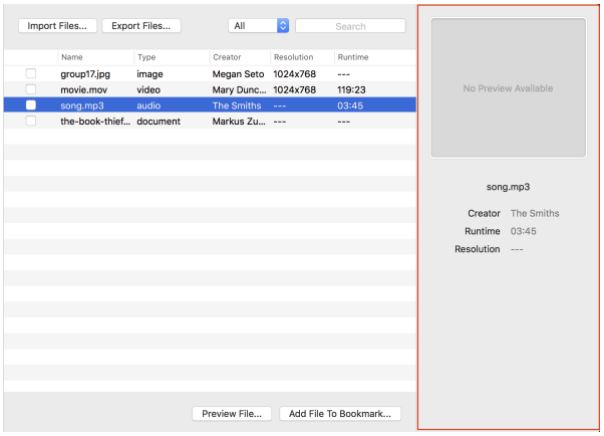


Figure 10: *The preview shows the files name and metadata. "No Preview Available" default shown, as it is an audio file and doesn't not have an associated image*

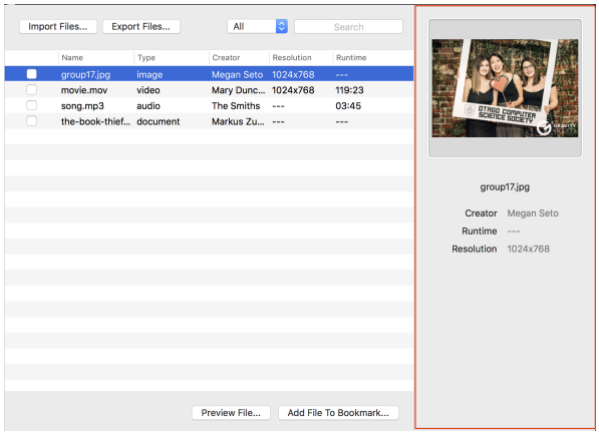


Figure 11: *Preview is showing the image and metadata associated with the selected file "group1.jpg".*

- ***Persistent Data Storage -*** When the application is shut, and then reopened, any files that have been imported to the library, and any notes that have been written on the files will remain. To do this, we read and wrote a file named "library.json" into/from the application user's "~/Library/Application Support/MediaApp" directory. If it is the user's first time running the application, the file will automatically be created, and if the data already exists the files will be read from and written over. We chose to store the files in

this location due to the “File System Basics” guide on the apple developer websites, which states:

“Put app-created support files in the `Library/Application Support/` *directory. In general, this directory includes files that the app uses to run but that should remain hidden from the user. This directory can also include data files, configuration files, templates and modified versions of resources loaded from the app bundle.”*

- ***Can Add and Remove Metadata to files (and edit):*** We are able to add and remove metadata to files. We believe this is a additional functionality because the assignment outline under the ***Essential Functionality*** subheading, the instructions just state that metadata should be able to be previewed, however we took it a step further and have enabled the users to be able to add new meta data or delete any added metadata. Users are not able to delete required metadata (such as *creator*). Unfortunately, we ran out of time to ensure that metadata being created with a key that already exists cannot be added to the file. This is something we would look to fix in the future.
- ***Keyboard Shortcuts on Content View:*** On the content view, we have implemented some keyboard controls. On the content view split (middle split), you are able to preview the current table row by pressing the ***enter/return****,* or open the projector view using the ***space bar*** key on the keyboard. We also made it so that users can navigate up and down the file table with the up and down arrows (note that because of this, the Image Preview/right hand view reloads the file data and preview with the correct information accordingly).

Object Oriented Design

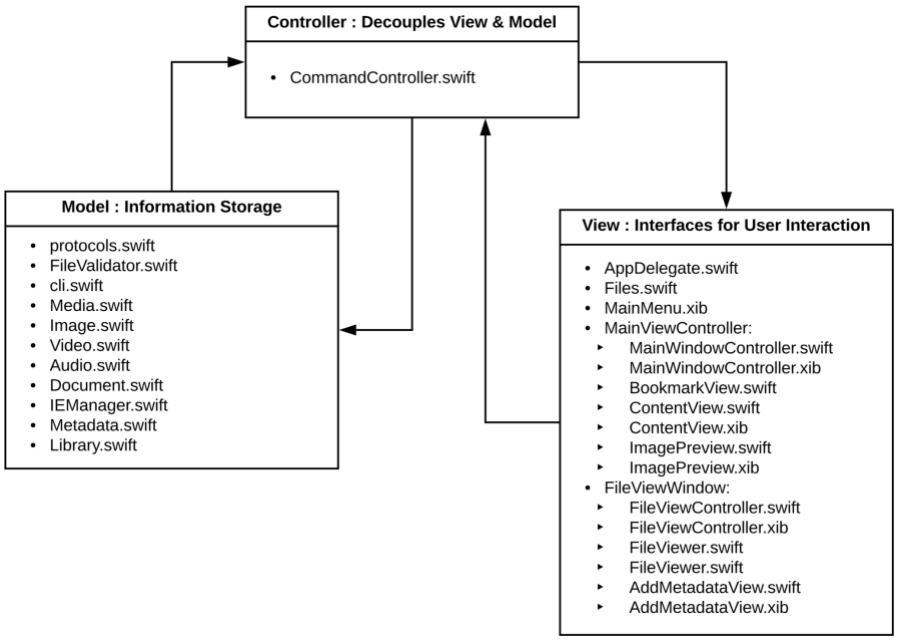


Figure 12: *The Structure of our Application*

For this assignment, we tried to closely follow the MVC (Model View Controller) design pattern, in order to decouple our views from the backend logic of the application. The diagram above shows where the files in our project belong in terms of the model, view, and controller. The Command Controller acts as an in between for the View and Model. Towards the end as we were finding some things difficult to implement, we sort of lost the structure of the model, but we followed it for the most part.

Because this was our very first time designing a Cocoa application, we had some trouble, especially concerning how to let different views communicate with each other. We decided to implement this by sharing instances of the views with other views that needed to know about them. This worked well, and made developing our application a bit easier, but results in the system being highly coupled. In the future, we would look at refactoring our application to fix this.

Testing

For your convenience, we stored our test files (actual images, audio, video, documents) within the project, to ensure that these were copied to the Bundle each time the application is run. These files (if they don't already exist in the directory we are writing to) are copied from the Bundle to the location "`~/Library/Application Support/MediaApp`". This ensured we could determine the path to these files, which allowed us to use these paths in our test JSON file, which will work on your computer. This was a little bit inconvenient to do and not very good practice, as large files are being copied to the Bundle every time the application is run. Not good, not good at all.

To test our project, we created User Acceptance Tests (UAT). If we had the ethical approval to carry these out, we would have users who had not interacted with the system before carry the tests out. The idea is you ask the user to follow a set of instructions using the application, without telling them *how* any of the instructions should be carried out. Whenever the user has problems following the instruction, a note will be written down explaining what they struggled with. If we have many people carry out our UATs and they all struggle with a particular instruction, we would look at redesigning the interface for that feature so it would be more intuitive for the user. Similarly, if many people find specific instructions simple, we can conclude that out UI for that specific instruction is designed well. We would have asked people to undertake the testing, but we didn't want any ethical breaches so we refrained from doing so. Below is an example UAT that we have written for our applications.

User Acceptance Testing Sheet:

Test Case Name: Learnability	
System: Media Manager Application	**Subsystem: N/A**
Executed By:	**Execution Date:**
Short Description: A user will be given instructions from a supervisor which they need to carry out.	

Pre-Conditions: The user can not be exposed to the system. Test case is for first time users so evaluate how easy the system is to learn.

Step	Instruction/Question	Expected Response	Pass /Fail	Comment
1	Import the file “test-data.json” to the library.	The user should click the “Import Files...” button, navigate to the file named “test-data.json”, and select “Open”.		
2	Filter the files by type “Audio”.	The user should click on the “Audio” bookmark.		
3	Filter the files by type “Documents”.	The user should click on the “Documents” bookmark.		
4	Filter the files by type “Images”.	The user should click on the “Images” bookmark.		
5	Filter the files by type	The user should click on the		

	“Videos”.	“Videos” bookmark.		
6	Return to the view that shows all the files in the library.	The user should click on the “All” bookmark.		
7	Add a new bookmark named “Test Bookmark”.	The user should click the ‘+’ symbol on the bookmarks panel. They should type “Test Bookmark” in the shown text field, and click the ‘Add Bookmark’ button.		
8	Add the file “goldilocks.txt” to the bookmark named “Test Bookmark”.	The user should select the “goldilocks.txt” file while either on the “All” bookmark or the “Documents” bookmark. They should then push the “Add File To Bookmark...” button. The bookmark “Test Bookmark” should be chosen from the drop down list, and the “Add Bookmark” button should be pushed.		
9	Add the files “strawberries.png” and “test-pdf.pdf” to the bookmark named “Test Bookmark” at the same time.	The user should select the “strawberries.png” file and the “test-pdf.pdf” file while on the “All” bookmark. They should then push the “Add Files To Bookmark...” button. The bookmark “Test Bookmark” should be chosen from the drop down list, and the “Add Bookmark” button should be pushed.		
10	Remove the bookmark named “Test Bookmark”	The user should select the ‘Test Bookmark’, click the ‘-’ symbol on the bookmarks panel, and when prompted click the ‘Delete Bookmark’ button.		
11	Return to the view that shows all the files in the library. Select the first file in the list, and preview it.	If the user is not already on the “All” bookmark, they should select it. They should then select the first file shown in the view (“bubbles.m4a”). The ‘Preview File...’ button should be clicked, or alternatively the first file in the view should be double clicked.		
12	Navigate to the next file in the preview view.	The user should click the “Next” button, and the file “fishes.MOV” should be displayed.		
13	Add a note to “fishes.MOV”	The user should click on the box		

	that says “Fish are friends.”	that is labelled “File Notes:” and should type “Fish are friends.”		
14	Add a piece of metadata to “fishes.MOV” with the key: “fish” and the value “are friends”.	The user should click the “Add Data” button. “Fish” should be typed in the text field labelled “Key” and “Are friends” should be typed in the text field labelled “Value”. The “Add Metadata” button should then be pressed.		
15	Edit the piece of metadata on “fishes.MOV” with the key “fish”. Change its value to “are made for eating”.	The user should select the piece of metadata with the key “fish”. They should then press the “Edit Data” button. In the alert, they should enter the value “are made for eating”, and then press the “Edit Metadata” button.		
16	Remove the piece of metadata on “fishes.mov” with the key “fish” and the value “are made for eating”.	The user should select the piece of metadata with the key “fish”. They should then press the “Remove Data” button. On the given alert, they should then press the “Remove” button.		
17	Navigate to the previous file in the preview view.	The user should click “Previous” and the file “bubbles.m4a” should be displayed.		
18	Navigate back to the main window.	The user should either make the main window active, or close the preview and make the main window active.		
19	Search the entire library for “Lana Del Rey”.	The user should select the bookmark “All”. They should then type “Lana Del Rey” in the search bar, and press enter. “Fishes.MOV” should appear in the content view.		
20	Search the Documents for “Channing Tatum”.	The user should select the bookmark “Documents”. They should then type “Channing Tatum” in the search bar, and press enter. “goldilocks.txt” should appear in the content view.		
21	Search the Audio for “Golden Studios”.	The user should select the bookmark “Audio”. They should then type “Golden Records” in the search bar, and press enter. “bubbles.m4a” should appear in the content view.		

22	Search the Images for “Sam Fleury”.	The user should select the bookmark “Images”. They should then type “Sam Fleury” in the search bar, and press enter. “Strawberries.png” should appear in the content view.		
23	Search the Videos for “Lana Del Rey”.	The user should select the bookmark “Videos”. They should then type “Lana Del Rey” in the search bar, and press enter. “Fishes.MOV” should appear in the content view.		
24	Close any open windows.	The user should click the red ‘x’ on the left hand corner of any open windows.		

Member Roles

Because pair programming worked well in the first assignment, we decided to follow a similar approach once again. The first thing we did was discuss how we wanted the user interface to look (as mentioned in the above *Research/UI Design* section) as this is what the assignment revolved around so it was important that there was a mutual consensus on the design and how certain actions/functions would be carried out - this understanding enabled us to work on different functionalities concurrently but still remain on the same page, a majority of the project was carried out under pair programming - but due to the large size of the assignment, this could not always be done as we needed to streamline the development in attempt to complete what was expected of us.

Future Extensions

- Decouple the code and implement better OO design
- Persistent bookmarks
- Implement zoom in and out
- Improve window/object resizing
- Menu bar working properly
- For audio files, implement a volume slider to allow the user to control the volume of the audio. Similarly, we would also want to implement a slider of sorts to allow the user to skip to and from specific parts of the audio, labels for the current time and full audio runtime as pictured below (Screenshot taken from itunes)

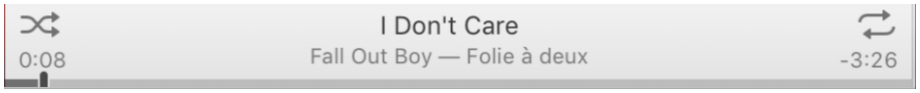


Figure 13: *Example of audio run time and time labels*

- Check if audio files have an image attached to them, if it does then set the image to the audio file preview, replacing the place holding headphones image.

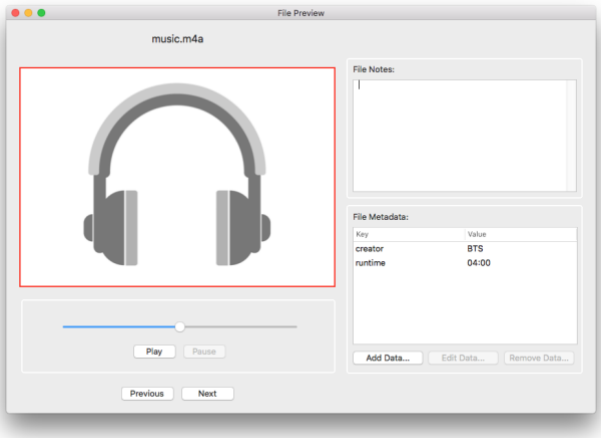


Figure 13: *Audio File Preview. Headphone icon outlined, in future could replace with audio image*

Final Cleanup

If you don't want our files sitting on your computer, navigate to the directory:

```
~/Library/Application Support/MediaApp
```

And remove all files (through terminal as the 'Library' folder is hidden to mac users).

Additional Resources

For this project, we had to take inspiration from some online resources and tutorials. These resources are listed below:

- ***macOS NSTableView Tutorial*** by ***Warren Burton.*** We followed a similar format in the tutorial to populate our tables with data
  Source: *https://www.raywenderlich.com/830-macos-nstableview-tutorial*
- ***Cocoa Architecture L1 - Controllers*** by account: ***Apple Programming.*** We followed this tutorial to get implement the split views
  Source: https://www.youtube.com/watch?v=Xy0-x-Zf1YE&t=590s
- ***Headphone Image used in Audio file preview:*** Created by Rawpixel.com - Freepik.com
  Source:https://www.freepik.com/free-vector/illustration-of-headphones-icon_2606091.htm#term=music%20symbol&page=1&position=35
- ***Function to check if a directory already exists:*** Brennan Stehling, on his GitHub: brennanMKE
  Source:
  https://gist.github.com/brennanMKE/a0a2ee6aa5a2e2e66297c580c4df0d66

- ***Function to get a String from a user using an alert box:*** Marc Fearby, Stack Overflow
  Source: https://stackoverflow.com/questions/28362472/is-there-a-simple-input-box-in-cocoa