COSC 346 Assignment Two

Rhianne Price / 1953770

My GUI is made up of three key parts:

- A view in which single pages of a PDF file are displayed.
- A tool bar underneath the PDF View for recording notes specific to the current PDF as well as control buttons that apply to the current PDF.
- A sidebar on the left for displaying and navigating between multiple open documents and bookmarks.

I have put the four main elements in resizeable split views so that the user can customise the GUI. As extra functionality I have implemented rich text formatting for the notes, and document search within the displayed PDF file.

I an outline list to navigate between different lectures and different pages because it provides a nice visual summary of the open lectures and the slides within them. The user can quickly find the desired lecture from the list and jump straight to the slide they want, or tab through the slides in the lecture. The user can expand or collapse the slide numbers of each lecture, so they don't have to see all the slides available of all the lectures they have open at any one time. This keeps the interface clear of items that the user isn't focussing on. This list gives a visual indication of the current lecture and current slide by keeping the current slide highlighted even when other parts of the GUI have focus or other controls are used to change the slides. This gives the user a sense of where they are relative to other slides and documents at all times. A more visual approach would have been to also include thumbnails with the page number, so that the user needn't know the number of the page they are looking for. I decided against this because thumbnails would clog up the list and be impractical for longer documents.

The controls for managing bookmarks are in a similar list. Bookmarks can easily added, removed and navigated between. I have made the distinction between selecting a bookmark and going to the bookmarked slide because the user can select a bookmark to rename or delete. Selecting a bookmark does not jump to the bookmarked page because this would be annoying if the user wanted to rename or delete a bookmark but not go to the bookmarked page.

I have grouped the list of bookmarks with the list of open lectures because both lists and their associated add/remove controls are relevant no matter which page is being displayed. All the information and controls relevant to the current lecture is to the right of the sidebar. This gives the user a clear sense of which information and controls are relevant to the current document and which information and controls are relevant to all open documents (the sidebar).

I have put a search bar at the top of the current lecture view because I feel like that's where they normally go. Users are used to adding text at the underneath something because we write top to bottom, so I have put the text box for recording notes at the bottom of my GUI, underneath the PDF View. The controls for rich text formatting are hidden in the menus, as they are seldom used clutter. Since the zoom controls are specific to the lecture slide currently in view, I have put them next to the notes. Technically the previous and next buttons aren't specific to the current view because they can be used to navigate between lectures, but I wanted most of the buttons to be in the same place so that the user only ever has to go to one place if they know they are looking for a button. Disabling these navigation buttons at the start or end of the lecture set give another visual cue to where the user is. I have left all these buttons unlabelled because too much text is overwhelming, and I feel like a lecturer should be a frequent user and doesn't need the labels.

In terms of object oriented design, my implementation makes extensive (but inevitable) use of inheritance through the Cocoa Framework. Looking back, I should have given the sidebar and main view their own view controllers, which would enforce better encapsulation and abstraction among view elements. I realised this a bit late. Refactoring the `DocumentWindowController` class would have been a large task. Even with separate view controllers there would be tight coupling between the the sidebar and the main view, because the lecture and bookmark lists need to interact with the PDF View for most things. If I had thought about it earlier I would have still split things up a bit better instead of just throwing everything into `DocumentWindowController`. I have managed to make a split between model and controller logic, which at least enforces some level of abstraction and encapsulation.

Extra keyboard shortcuts

⌘O	Open a PDF. You can open multiple files at once.
⌘N	Start a new lecture set in a new window.
⌘→	Go to next item in lecture set, whether it be in the current lecture or the next lecture.
⌘←	Go to previous item in lecture set, whether it be in the current lecture or the previous lecture.
⇧⌘→	Go to the next lecture.
⇧⌘←	Go to the previous lecture
⌘P	Present the current lecture set. Opens a new window with a single page PDF view that mirrors the page in the parent window.
⌘+	Zoom in.
⌘-	Zoom out.
⌘0	Zoom to fit.
⌘D	Add a bookmark at the current page.
⌘S	Search in the current document (⌘F is taken by find within the notes box).

Alternatively, users can also navigate through a document using the scroll wheel/trackpad. Users can access all text formatting controls for the notes in the “Format” menu, and all text editing controls from the “Edit” menu.

Note: Moving from Swift 4 to Swift 3 started giving me a missing layout constraint in a Cocoa class I can't access or add constraints to (`NSVibrantSplitDividerView`). This doesn’t seem to actually affect usage of the GUI.