Nate Perry and Nate Kierpiec
Source Code: https://github.com/nateperry/LegoHelper
About: http://thenateperry.com/projects/LegoHelper/

Lego Helper

Lego Helper allows its users to access easy to read instruction manuals for some of their favorite Lego sets. Our goal for this app was to do just that. A super easy interface allows for easy navigation between different themes and subthemes, helping user quickly find their next project.

Difficulties in the development process included many issues with API's that we were intending to use, not being experienced with certain types of view controllers, or silly errors that came from late night grinds. Initially we were going to use two different API's. One for the information about each theme and set, and the other to load in the instructions on how to build each set. Once into development, it was found that the second API did not actually provide instructions as there was no endpoint listed in their documentation. Thus the first API was for the instructions as well. While the instructions it shows are the exact pictures from the original models manual, it is listed in a PDF which is in a webview. This works but it does not give us the ability to track the users progress which we had hoped to do. We also wanted to implement a 'Favorites' type of bookmarking for quick access to each set. Due to our lack of skill working with storyboard and nested view, we were unable to complete this feature in time. Another issue we encountered with the API was that its documentation was not accurate. For example, when requesting a list of all sets within a theme, it claims that there are only two required parameters. When we made the request was made with only the required arguments, nothing was returned. The fix to this was to provide all of the optional arguments with null values. And lastly, while we have a valid API key, we are limited to only twenty sets. This is a restriction the API has for projects that do not have an API key. Although we have checked the validity of our key, we still seem to be restricted to the twenty set limit. Lastly, one issue that took us a while to fix was handling request completions. After our API calls were finished, we would post a notification. The notification handler would then load in the appropriate view. This seemingly simple action would take upwards of 30 seconds to complete until we found out that we could call dispatch_async(). This method would move our processes on to the main thread, allowing for the view to be loaded in a matter of one or two seconds. This technique was implemented across the rest of our project as similar issues would come up again.

The work on this project was almost 100% done with both us working side by side. Usually one of us would be in charge of typing while the other would be looking over the others shoulder and providing guidance. The time and effort put in was pretty even since we mostly worked exclusively when the other person was there.

Expected Grade: A

The reason why we deserve an A or greater for this project is because of how much work that we did that was and was not covered in class. Implementing collection views, attempting to use the new Webkit Webviews, using a chain of view controllers to create a navigation, and using various techniques to manipulate the current view are all things that tested our knowledge and skill in iOS development. Everything we created adheres to the MVC standard and all of our code is filled with comments and hints across all of the class files we used.

Resources We Used

Brickset API - http://brickset.com/tools/webservices/v2
Cubiculus API - http://www.cubiculus.com/api/ (not used in final)
Collection Views - http://www.raywenderlich.com/22324/beginning-uicollectionview-in-ios-6-part-12
Searching - http://www.raywenderlich.com/16873/how-to-add-search-into-a-table-view
WKWebView -
http://www.kinderas.com/technology/2014/6/7/getting-started-with-wkwebview-using-swift-in-ios-8

We also referred back to our class projects, such RIT Maps for the webviews and Blackjack for the popups and view nesting.