**[Milestone 2](https://bblearn.griffith.edu.au/webapps/assignment/uploadAssignment?content_id=_2157476_1&course_id=_48251_1&assign_group_id=&mode=view)**

**A Photo Collection Viewer**

***Xcode Project "MyPhotos"***

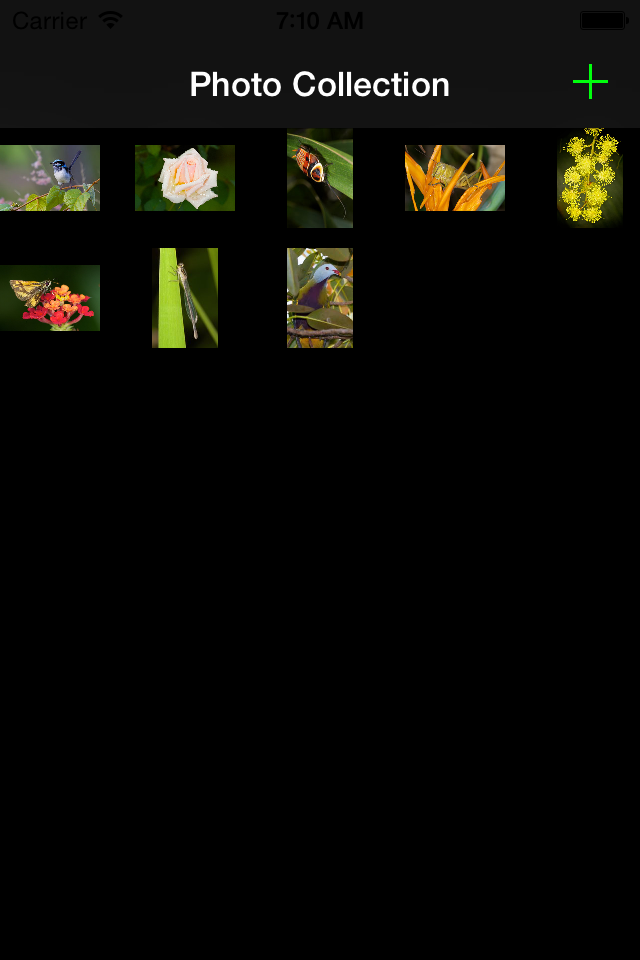
Create a new Xcode project consisting of an iOS Single View Application called **MyPhotos**. When you set up the project, make sure the Company Identifier is set to **au.edu.griffith.ict.prog** and Devices is set to **Universal** After clicking **Next**, select the **Desktop** folder to create your project in and make sure you select **Create local git repository for this projec**t, before clicking **Create**.

***Git Repository***

*Commit every step* described below.  Make sure that there is *at least one commit for each of*: the*empty project* (with just a view, no table view controller), the *master and detail views*, the*unit tests without the model* (test first!), and, of course, the working unit test for the *model* (i.e. with a full implementation of the model).  Ideally, you want a lot more commits: every time your program compiles, make sure you commit!  Also make sure you have a meaningful description for every commit that outlines the changes you made since the last commit (another reason to commit often as this makes it less likely to forget the details of all your changes!).

***Master and Detail Views***

Edit your main story board to have a **master** and a **detail** view.  On the master view, the user should see a collection of all the photos.  There should be a [+] button that takes the user to the detail view to **add** a new photo.  Tapping on an existing entry in the table should also take the user to the detail view to display the web page and **edit the Title, Tags, and URL** of the existing entry.  Here is an image of what your master and detail view user interfaces should look like:



Make sure you link up the necessary actions and outlets!

***Testing***

Create **unit tests** for your model making sure that you follow a tests-first approach (you need to **commit your tests to git before the model is implemented to demonstrate this**)!  Make sure you at lest test the **setters** and **getters** of all the properties as well as **adding** and **inserting** elements *at each end* as well as the *middle* of the array for both *empty* and *non-empty* arrays.

Create **user interface tests** for your view!  Make sure you at least test **adding** elements with the **+** button on your master view and **inserting** text *in each text field* as well as **editing** an existing photo.  Ensure that all tests that transition from the master view controller to the detail view controller also transition back and that changes are visible on both views!

***Model: Array of Photos***

Create a **Photo** class that contains the title (string), tags (array of strings), and URL (string or NSURL) of a photo.  Use an *array of photos* for your photos collection.  Save your changes and run your program to test that it still compiles.  All the tests should work now!  Don't forget to commit your changes!

***Controller: Putting it all together***

[*Read the****Collection View Programming Guide****for iOS*](https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/CollectionViewPGforIOS/Introduction/Introduction.html)*in the Developer Documentation!*  Link up all your GUI elements and add the necessary code to your controller to make the buttons and all other user interface elements work. Selecting a name on the master view should take you to the detail view to edit the corresponding entry.  Pressing the [+] button on the master view should take you to the detail view with empty fields to add a new entry.  On the detail view, pressing **Back** (or **Photo Collection**) button should save the current changes.  After pressing the return key, the web image corresponding to the URL in the text field should be loaded and displayed on the detail view.  Pressing the back button (may be called *Photo Collection* on the detail view), the user should be returned to the master view.  Also on the detail view, make sure the text fields are editable so the user can enter these data through the keyboard.

***General Style***

Use best practices when writing your code.  E.g., use subclassing, delegation, and other object-oriented design patterns as appropriate.  Make sure all your source files (.swift files) have a comment header that contains your name as the author and your student ID.  Put [swiftdoc](https://developer.apple.com/library/ios/documentation/Xcode/Reference/xcode_markup_formatting_ref/index.html) style comments in front of your classes, properties, and methods (including test methods), describing what each of these do!  For methods, make sure you document parameters and return types as well as pre- and post-conditions!  Make sure your code is well formatted and readable, and uses consistent indentation!

***Submission***

Zip up the folder that contains your project (e.g. if your project is called MyPhotos, you should find a *MyPhotos* folder on the Desktop). On the Mac, you can easily create a ZIP file of a folder by holding the Control key, clicking on the folder you want to zip up, and then select Compress "MyPhotos" to compress).  Submit the ZIP file to BBlearn (click on the link that reads **Milestone 2** at the top of this milestone page, then attach your ZIP file, and submit).