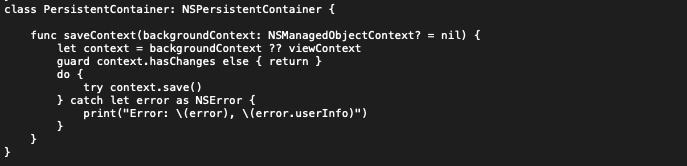
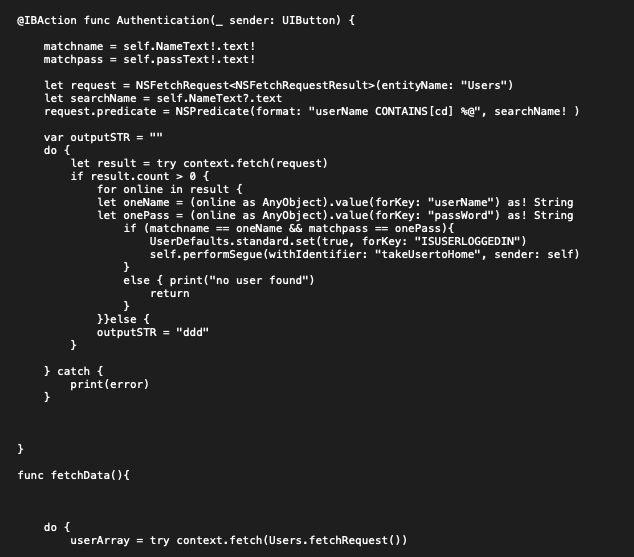
**UrFeeds** 

Shania Daley and Minyen Chiang

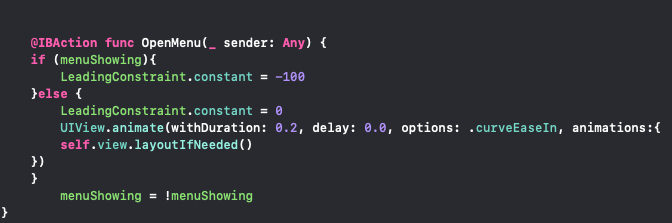
Our applications is much less complex than our original plans in our proposal but we tried hard to implement as many of the features as we can. First we created a login and register page for our users. We utilized **persistence and core data** to store the newly created user name and password to a local data source.





This method takes in the information from the UITextfield and commits a **fetch request** from the data stored to check if it matches information. Once the user’s credentials match in the system it automatically brings them to our home screen.

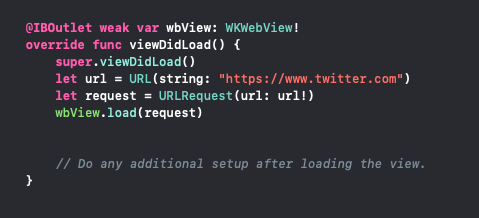
Our home screen is embedded inside a **UINavigationBar** and uses animation to create a seamless slide in and slide out menu. Open menu places leading constraints when the menu is out and when the menu is in and is connected to our navigation button. We added a delayed animation to make it look more smooth.

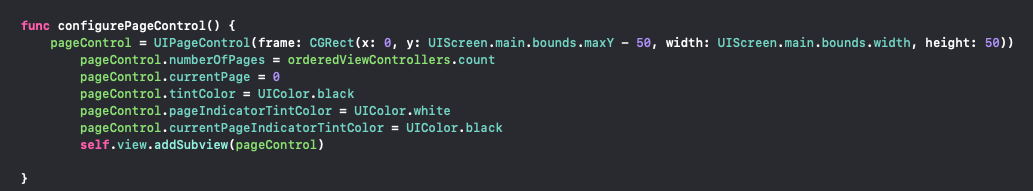


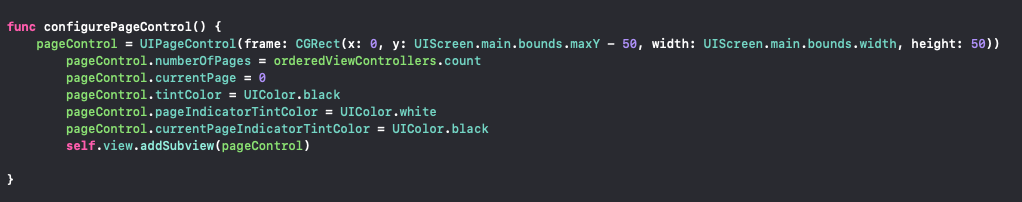
Our main menu also has a **collection view** on top of it to be mini buttons to go directly to the associated view controller. We have four views associated with each social media account. Each **collection cell has a UIImage and UILabel** attached to it and we used the dequeue method to generate different titles and labels for each.



We use a **WKWebKit** to display each page of social media accounts. We have a view controller for Twitter, Instagram, Facebook and to view News. The webkit allows for embedded web pages to be presented. We use the URLRequest method to fetch each https web page.



To add more animation we embedded each of the pages inside a **page view controller** to create a **swiping transition**. We adopted the UIPageViewDelegate and Data Source to conform the pages to page through. Inside the orderedViewController method, we create and array of view controllers that transition through. The newVC method instantiates the view controllers to prepare them to be paged through. The first pageView method handles view controllers before the current one in place. It takes the current index and updates it. This method and the after method sets the last index and first index up so the pages continuously slide and don’t stop. 

A **page control method** is added to add the bottom indicators to the view and turns black when you are on the the current page while the unselected pages are white.The control is added on top of the views using addSubview 

This is our first time experimenting with **local notifications**. For this method we added a timer icon to our app to allow the user to enter how long they want to be using the app and when the time is up a notification will appear from their Notification center. We had to import **UserNotifications framework** in order to get the Notification Center functions and delegate. In this method we attached out timer button as an outlet and added an **AlertController** to produce a pop up which prompts the user for the time in minutes. We then added two actions, Submit and Cancel. When submit is pressed the text within the Textfield is placed inside a local variable and converted to a type Double. Making sure the input is correct and not nil, we then created a notification design and added a title and body to it. We created a trigger which is the timer set and set the time interval to the input\*60 to get the time in minutes. We had to conform the Notification center to its delegate in order to show the notification while the app is running and in the forefront of the screen. Using all of those parameters we created a request object that creates the timer. This is a non-repeating notification so the user can keep setting new ones.

Location Service: Imported CoreLocation service to get users live location, updates new location when user moved 10000meters. 