**Great Number Game**

* Uses Simple Alert Dialog (background color, title, message, button)
* From String to number

## Tipster

* Uses One IBAction for 10 buttons
* Formatting the decimal number into 2 digits after the decimal

Calculator

* Uses One IBAction for 19 Buttons
* Using Tags To track The Clicked Button

AddingToTheBeast (Super Good Test: Branch)

* Test sections in table
* Space between sections
* Add 3 rows to the sections
* Color each row
* Slide to delete from table
* Add to the table
* Tab on row to do something
* Corner the cell
* Set heigh to the row in the cell

Binary Counter

* Using table
* Uses Add Target To button
* Call button from another class and add target to local function
* Very weird solution
* Using Regular Protocols and Delegate in my second branch

MadLibs

* Using Model Segue
* Using Unwind Segue to pass data
* Passing Data Back From Second Control Viewer
* Passing 4 text field
* Not using protocols

PocketListUD

* Uses table view controller
* Not using view controller
* Uses Navigation Controller in all views
* Add to the list
* Edit items
* Slide to delete
* Using alert if entry empty
* Uses select on section
* Uses section and spaces between sections
* Uses one row per section
* Pass Data Using Protocols
* Pass data not using protocols, inside prepare function, using segue, throw UI Navigation Controller
* Branch (Bucket List Refactor)
* Uses One Segue for Add and Edit and compare sender
* Brach (CoreDataToBucketList(Add/Update))
* Using CoreData as Database
* Add new item to CoreData
* Edit Item In CoreData
* Using Details Button for Edit instead of row click
* Branch (CoreDataToBucketList(Delete))
* Using New version of Swift CoreData Handling in fetch and add
* New Way of Fetching the Data
* New Way of Adding to the CoreData
* Delete From CoreData

ToDoListApp

* Uses CoreData to fetch, save and delete
* Uses table view controller
* Not using view controller
* Uses Navigation Controller in one View
* Uses Show to move to other view
* Uses pop to back to previous view
* Uses TextField, TextView and Date Picker
* Uses Protocol To pass data
* Uses Custom Cell Class
* Uses CheckMark in Table Row when selected
* Uses Sections and spaces between sections
* Formatting the date to show only date without time
* Fix The Corner Radios for TextField, Buttons, TextView, Cells and Static Cell
* Set Border Width and Border Color to TextField and Cells
* Uses Text View and Inherit from UI Text View Delegate to Add Place Holder (hint)
* Uses Alert
* Printing The Database Path Folder

MyDogs

* Uses Image, Pick Image from phone albom, display image in image view
* Uses Collection View as Main View to display data instead of View Controller
* Uses on Click on Collection view cell to be edit
* Uses CoreData to fitch Data from database (Strings and Photos)
* Uses CoreData to save strings, and images as binary data
* Uses CoreData to delete and edit from database
* Uses protocol to pass the data for save, edit and delete data
* Passes Entire database to edit class to be edit or deleted (not passing elements)
* Uses show and modal segues
* Uses navigation for collection view and modal view
* Styling (Border Color, Border Width, Corner Ratio, Text Shadow offset, Text Shadow Color) my view elements (Button, Text Field, Button Title, Image View)
* Check Data Entry for new data before saving
* Check Data Changed before Editing
* Uses Alert
* Printing The Database Path Folder

NorthEastSouthWest

* Uses unwind Segue
* Custom button title (attributed title) and tint color

LocalNotifications

* Uses Notification (give Authentication)
* Check Authorization Every Time before Sett it
* Start New Notification
* Cancel Running Notification from Schedule
* Delete Notification from Device Center
* Display Notification even when the app open
* Make changes in the app when the notification show
* Uses CoreData as Database to Save and fetch data only
* Uses regular Picker View to pick time from list
* Uses Picker View Methods to set the data (Delegate and Data Source)
* Uses Custom Alert Function to display alert from with different
* Uses Switch View Function to Hide and Unhide Labels
* Uses Date to get the time and Add Seconds to Get Future Time
* Format the Date to get only Time without date with style
* Uses Attributed (Custom) String to style my string
* Uses Enum to handle the alert types
* Uses Struct to handle the data
* Uses semasphore to stop the app until the user respond to the alert (same as Delay)

TimeVisualizer

* Uses CoreData with 3 entity connect to each other (week = 7 days = 12 times)
* Fetching all three entity
* Calling data by filter the current week and day and save it in new array
* Uses tabelview and pickerview
* Using date with formatter to get day number only
* Save settled data when creating new week (7 days and 12 times for each day)
* Set heigh or the table view row using code and using function
* Uses 2 table view in one view (one on storyboard one created programmatically)
* Uses add button to show button popup menu
* Uses programmed tableview for popup menu from down with animation to go to other pages or start new week
* Programmed tableview created from 0
* Uses window to show as background when menu up (add observer (UITapGestureRecognizer) so when click = menu and window disappear with animation)
* Passes data to other class using prepare func
* Passes data to other class using delegates and protocols
* Passing indexPath to save (update) data
* By using 2 table all functions have (if) to know which table is called
* Custom tableview has different settled heigh for each row
* Uses alert inside did select raw function (tableview function)
* Uses pickerview to change the day
* Uses attributed string for the string in the row of the picker view
* Have Custom Cell Class created programmatically (not exist in the storyboard) and uses lazy variable
* Have class created from scratch, created manually (programmatically)
* Uses Charts (Pie and Radar) created manually in the manual created class
* Manual class have created button with addTarget to observe button clicking
* Uses one action function for 6 buttons using tag and switch with delegates
* Uses dismiss to close the window
* Uses Extension for cell class to use text field functions (textFieldDidEndEditing) to the data when user finish editing without need of save button and point the delegate to the sell
* Print the database path in the app delegate class

GETPeople

* Get data from API using String way
* Save data in String array
* Using table view to display data
* Using Next and Previous button to refresh data from other API pages

# GETFilms

* Same as GETPeople add to it also get films
* Uses API in sperate class Using My Weird Way without using static func

# MovieQuotesAssignment

* Uses background image
* Getting data from API using struct as variables database to simplify getting data
* Uses search to get the wanted movies
* Uses replace to replace the empty space in the search string to something that the API can reed
* Uses collection view
* Uses custom cell for the collection view
* Download images from internet and save them in dictionary
* Display images and movie name in the middle of the image on the collection view cells
* Uses default image for movies that misses images
* Uses attributed text in the label to add shadow to the text so the name can be seen
* When cell selected open new view created manually to show movies details
* Passes data to the created view
* Make changes to the cell layout using the collection view layout functions
* Change the spaces between the cell using collection view layout functions
* Uses search bar and its functions (cancel Button, result list button)
* Uses auto complete library
* Uses auto complete struct (model) as auto search database, Using the helping site
* Uses delegate class to passe the selected string from auto complete class
* Uses custom label and custom background image in the manual created details class

# MVCiniOS

* Same as GetFilms but uses the instructor’s way
* Uses navigation controller and tab bar controller to navigate between movies and people
* Uses separate model class to get data from API
* API model class uses static functions to be called directly
* Uses escaping in the function to uses this function as wanted in the called class
* Basically call the data from API in the background in the separate class and uses the called data in the main

# FinalSteps

* Same as MVCiniOS
* Add custom manual created class to display details
* Passes data to the created custom class
* The created custom class called (vanilla view controller)

BucketList

* Has branches, Bucket List IV branch is the last
* Same as FimalSteps but uses local API database
* Uses Get, add, Edit and delete from API (CRUD)

SportsAndPlayers

* Using CoreData
* Using 2 Entity with relation one to many (like array)
* When parent deleted all children deleted
* Passing parent to children class to filter the children data
* Filter the data using NSPredicate
* Add, Edit, Get and Delete from CoreData
* Uses alert with 3 text field inside
* Uses button inside the table view raw to get photo from library
* When photo get selected the button disappear
* Save the photo inside the CoreData and display it in the table view when exist
* Uses table view functions (swipe to delete, and set height to the raw)
* Uses image picker to pick an image from the library

IOSQuiz (Belt Exam)

* Uses Lot of animations