D4 – Jai Lafferty 101610772

The repo containing my Distinction App can be found at:
https://github.com/CreDataDrivMobApp/task-d-JaiLaff

This task was riddled with challenges. As I was extending my 5P, 6P and 7C tasks into this, I was implementing a lot of features for the first time into this single app which felt at times as if it were being held together by hopes and dreams.

The main challenges that I faced however were:

- Parsing API Data
- Handling Asynchronous tasks responsibly

I will be discussing both challenges and how I overcame them in this report.

Parsing API Data – (Overcame by persistent debugging)

I only really had a problem parsing the returned iTunes Search API data. While it's a great data source the way the data was formatted certainly had room for improvement.

I was using the Ray Wenderlich Half Tunes tutorial as a base for this entire task as it featured the same API, and I was having trouble unpacking the dictionaries. I didn't realise at the time that I had to continuously create a new dictionary with a subset of the previous dictionary. I would have loved to access attributes 4 layers down in a single line but had to allow the following mess to happen.

```
do {
    response = try (JSONSerialization.jsonObject(with: data, options: []) as? [String:Any])!

    // Make sure we actually have results
    guard let array = response["results"] as? [Any] else {
        return
    }

    let targetIndex = findAlbumInCurrentArtist(id: collID)

    for trackDict in array {
        if let trackDictionary = trackDict as? [String:Any],
            let title = trackDictionary["trackName"] as? String,
            let trackNo = trackDictionary["trackNumber"] as? Int{

            if targetIndex != nil {
                currentArtist.albums[targetIndex!].tracks.append(memTrack(title: title, trackNo:
                    trackNo))
            } else {
                print("Could not find relevant album in data")
            }
```

1
2
3
4

Each rectangle represents a deeper level into the decoded JSON

I've since learnt that mirroring the JSON in equivalent structs allows the decoding of json straight into a datatype to be unpacked however in this case it was just a matter of repeatedly printing the current data, checking the type of the next subset and repeating until a single value was obtained rather than a dictionary of values.

Asynchronous Tasks – (Overcame by reading)

This was the big one for me. I had sparse async task knowledge other than from small time spend studying them in the Android unit I had taken last semester. I had to battle with methods being called before the previous method was actually done. For example, the details screen would *constantly* open before I had loaded the tracks into memory resulting in an empty track list. This was a bug from task 5P that followed up until about 4 days before the demo. I then spoke to Ayman who led me to the light at the end of the tunnel. Completion Handlers.

Now I knew what a closure roughly was, and I know that data task has an argument called handler but past that I had no idea what a completion handler was. And believe me I spent a long time reading articles and code snippets to attempt to deeply understand what was going on. (I also have never used callbacks in any other language so this was incredibly new to me) I then stumbled on Bob Lee's blog where an article entitled "*Completion Handlers in Swift with Bob"* laid out what a completion handler was using various emojis and very casual language. Bob states, "Based on my expansive vocabulary list, completion handlers stand for, 'Do stuff when things have been done.' " Then, my undergraduate version of a eureka moment occurred, and I heavy-handedly sprinkled completion handlers in every possible location within my app. This article further led me to understanding exactly how the dataTask was working (A closure as a completion handler?! Amazing.) and what was really going on instead of just looking at my program not having a clue how it was doing what it was doing. I then realised I could nest completion handlers within each other, and I was really having fun from then onwards.

```
func begin(callback: @escaping () -> Void) {
    fetchAlbumData(callback: callback)
}

// Gets data from the API
func fetchAlbumData(callback: @escaping () -> Void) {
    var dataTask: URLSessionDataTask?

    dataTask?.cancel()

    if var urlComponents = URLComponents(string: "https://itunes.apple.com/lookup") {
        urlComponents.query = "id=\(currentArtist.id)&entity=album"

        guard let url = urlComponents.url else {return} // failed

        //Actual API Call
        dataTask = session.dataTask(with: url, completionHandler: {(data, response, error) in

            if let error = error {
                print("error: " + error.localizedDescription)
            } else if let data = data, let response = response as? HTTPURLResponse, response.statusCode
                == 200 {
                // If we have a 200 status (OK) then continue
                self.parseAlbumData(data: data)
                callback()
            }
        })
    }
    dataTask?.resume()
}
```

Using nested completion handlers to complete any task with a signature of () -> Void after fetching and parsing album data

While this example is of the most basic form of completion handler, having the ability to pass values and even pass closures was mind blowing and is incredibly powerful.

Conclusion

If anything, this Distinction App has taught me 3 key things:

1. That just because someone else wrote some code and put it online (Looking at you iTunes Search API & Stack Overflow answers), it doesn't mean it's particularly great.
2. How to better find resources for understanding a topic than the first two or three articles from google.
3. Sometimes the best way to teach a topic is by using everyday language and common place examples.

This has definitely been one of those tasks that I recognise that I've grown as a developer upon reaching the end.

REFERENCES

Lee, B 2017, *Completion Handlers in Swift with Bob,* Bob The Developer Blog, Viewed 29 May 2019, <https://blog.bobthedeveloper.io/completion-handlers-in-swift-with-bob-6a2a1a854dc4>

Tam, A 2017, *URLSession Tutorial: Getting Started*, Ray Wenderlich Tutorials, Viewed 29 May 2019, <https://www.raywenderlich.com/567-urlsession-tutorial-getting-started>