Review of Setting Up Firebase Project

```
//  AppDelegate.swift
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

   func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
       // Override point for customization after application launch.
       FirebaseApp.configure()
       return true
   }

   // MARK: UISceneSession Lifecycle

   func application(_ application: UIApplication, configurationForConnecting
connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) ->
UISceneConfiguration {
       // Called when a new scene session is being created.
       // Use this method to select a configuration to create the new scene with.
       return UISceneConfiguration(name: "Default Configuration", sessionRole:
connectingSceneSession.role)
   }

   func application(_ application: UIApplication, didDiscardSceneSessions
sceneSessions: Set<UISceneSession>) {
       // Called when the user discards a scene session.
```

```
        // If any sessions were discarded while the application was not running, this will be
called shortly after application:didFinishLaunchingWithOptions.
        // Use this method to release any resources that were specific to the discarded
scenes, as they will not return.
    }


}
```

ViewController.swift

- Remember to import the Firebase library and make a db variable

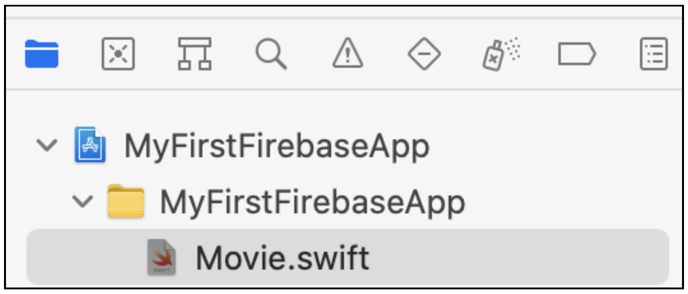
```
import UIKit
import FirebaseFirestore          // needed to access Firestore library functions
import FirebaseFirestoreSwift      // needed when representing document as objects

class ViewController: UIViewController {

   // create your firestore variable
   let db = Firestore.firestore()

   override func viewDidLoad() {
      super.viewDidLoad()
      // Do any additional setup after loading the view.
   }

   @IBAction func getAllPressed(_ sender: Any) {
    }
}
```

Objects and Firestore

IOS can be configured to send and receive data as *objects*

1/ Represent your firestore document as a class (struct) in your IO project

- Model a single document as a struct
- And then use that struct within your application



```
import Foundation
import FirebaseFirestoreSwift

struct Movie:Codable {
    // property to represent the document id
    @DocumentID var id:String
    // properties to represent the other fields in your document
    var title:String = ""
    var runningTime:Int = 0
    var genre:String = ""
}
```

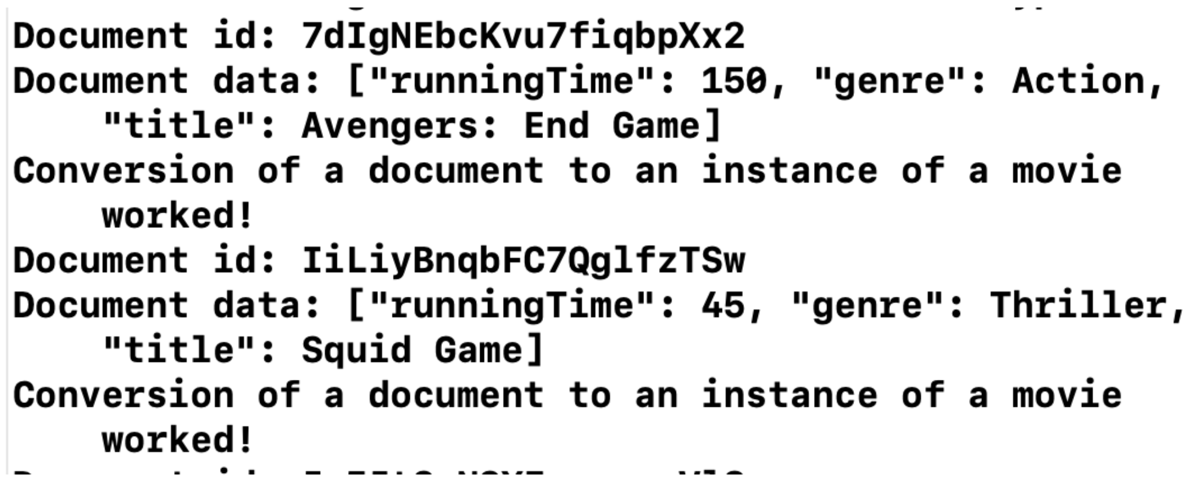
ViewController.swift

```
import UIKit
import FirebaseFirestore           // needed to access Firestore library funcitons
import FirebaseFirestoreSwift      // needed when representing document as objects

class ViewController: UIViewController {

    // create your firestore variable
    let db = Firestore.firestore()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func getAllPressed(_ sender: Any) {
        // when the button is pressed, retrieve all your documents form Firestore
        // AND represent them as a Movie struct (instead of just a generic dictionary like before)


        db.collection("movies").getDocuments {
            (query, error) in

            // error validation
            if let error = error {
                print("Error occured while retrieving documents")
                print(error)
                return
            }
            // otherwise everything went okay
            // loop through the documents and represent them as an object
            for document in query!.documents {
                print("Document id: \(document.documentID)")
                print("Document data: \(document.data())")

                // try to convert the "document.data()" into a instance of a Movie struct
                // In swift the do-catch is the equivalent of the try-catch in other programming
languages
                do {
                    let movieFromFS = try document.data(as: Movie.self)
                    print("Conversion of a document to an instance of a movie worked!")
                } catch {
                    print("Error converting document to an instance of type movie")
                }
            }
        }

    }
```

}

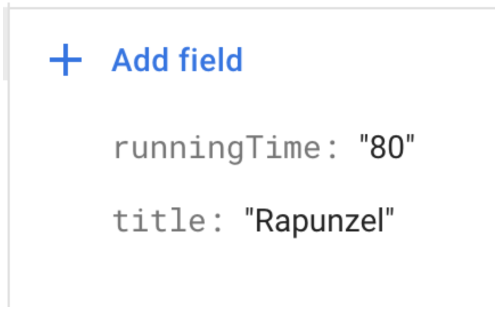
Expected result:

- After pressing the GET ALL button, the data should be retrieved
- The data should be converted to an object of type Movie

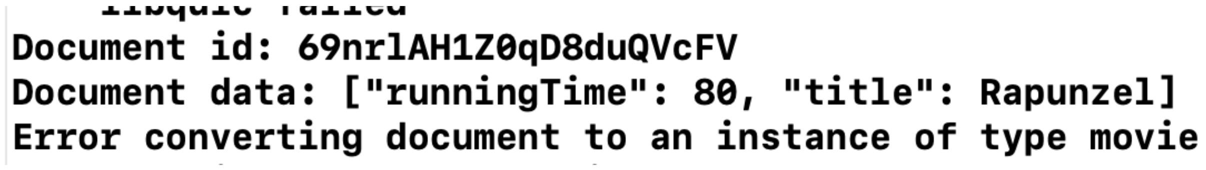


```
Document id: 7dIgNEbcKvu7fiqbpXx2
Document data: ["runningTime": 150, "genre": Action,
    "title": Avengers: End Game]
Conversion of a document to an instance of a movie
    worked!
Document id: IiLiyBnqbFC7QglfzTSw
Document data: ["runningTime": 45, "genre": Thriller,
    "title": Squid Game]
Conversion of a document to an instance of a movie
    worked!
```

If the document does not conform to the properties specified in the struct, then the conversion will fail

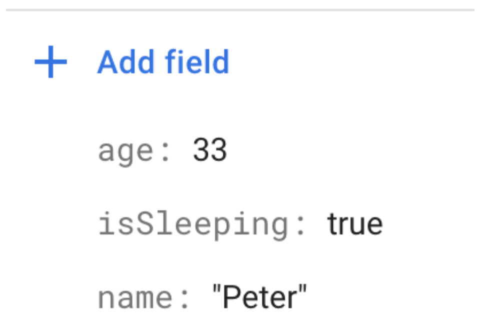


```
runningTime: "80"
title: "Rapunzel"
```

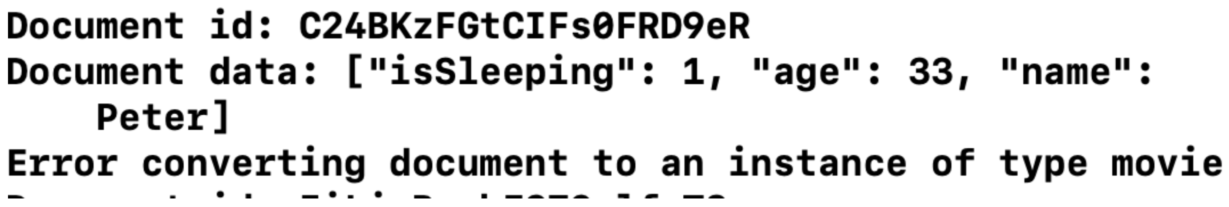


```
Document id: 69nrlAH1Z0qD8duQVcFV
Document data: ["runningTime": 80, "title": Rapunzel]
Error converting document to an instance of type movie
```

Another example of failure



```
age: 33
isSleeping: true
name: "Peter"
```



```
Document id: C24BKzFGtCIFs0FRD9eR
Document data: ["isSleeping": 1, "age": 33, "name":
    Peter]
Error converting document to an instance of type movie
```

Add a movie

Add outlets and action for the text boxes and SAVE ALL button
In the SAVE BUTTON action:

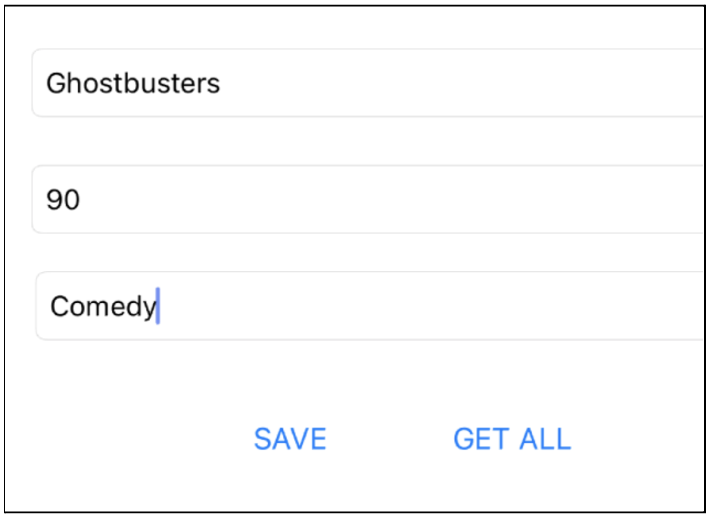
- Get the data from the text boxes
- Convert the input from the running time text box to match the data type of runningTime in the Movie *struct*
- Using the above information, create an instance of the *Movie* struct
- Using the addDocument(from:*Encodable)* function, save the Movie to firestore

ViewController.swift

```
import UIKit
import FirebaseFirestore          // needed to access Firestore library funcitons
import FirebaseFirestoreSwift      // needed when representing document as objects

class ViewController: UIViewController {

    // outlets

    @IBOutlet weak var lblTitle: UITextField!
    @IBOutlet weak var lblRunningTime: UITextField!
    @IBOutlet weak var lblGenre: UITextField!

    @IBAction func saveButtonPressed(_ sender: Any) {
        // 1. get data from the UI
        // You should use a guard-let / if-let
        let titleInput = lblTitle.text!
        let runningTimeInput = lblRunningTime.text!
        let genreInput = lblGenre.text!

        // 2. convert the running time to a number
        let runningTime = Int(runningTimeInput) ?? 0

        // 3. model the data as a Movie struct
        let movieToSave = Movie(title: titleInput, runningTime: runningTime, genre: genreInput)

        // 4. Send the movie struct to firestore
        do {
            try db.collection("movies").addDocument(from: movieToSave)
        }
        catch {
            print("Error saving document")
        }

        // 5. Done like dinner!
    print("Done!")

}

 // create your firestore variable
let db = Firestore.firestore()

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
}


@IBAction func getAllPressed(_ sender: Any) {
    // when the button is pressed, retrieve all your documents form Firestore
    // AND represent them as a Movie struct (instead of just a generic dictionary like before)


    db.collection("movies").getDocuments {
        (query, error) in

        // error validation
        if let error = error {
            print("Error occured while retrieving documents")
            print(error)
            return
        }
        // otherwise everything went okay
        // loop through the documents and represent them as an object
        for document in query!.documents {
            print("Document id: \(document.documentID)")
            print("Document data: \(document.data())")

            // try to convert the "document.data()" into a instance of a Movie struct
            // In swift the do-catch is the equivalent of the try-catch in other programming languages
            do {
                let movieFromFS = try document.data(as: Movie.self)
                print("Conversion of a document to an instance of a movie worked!")
                print(movieFromFS!.title)
                print(movieFromFS!.genre)
                print(movieFromFS!.runningTime)
            } catch {
                print("Error converting document to an instance of type movie")
            }
        }
    }

}
```

```
}
```

Expected result:

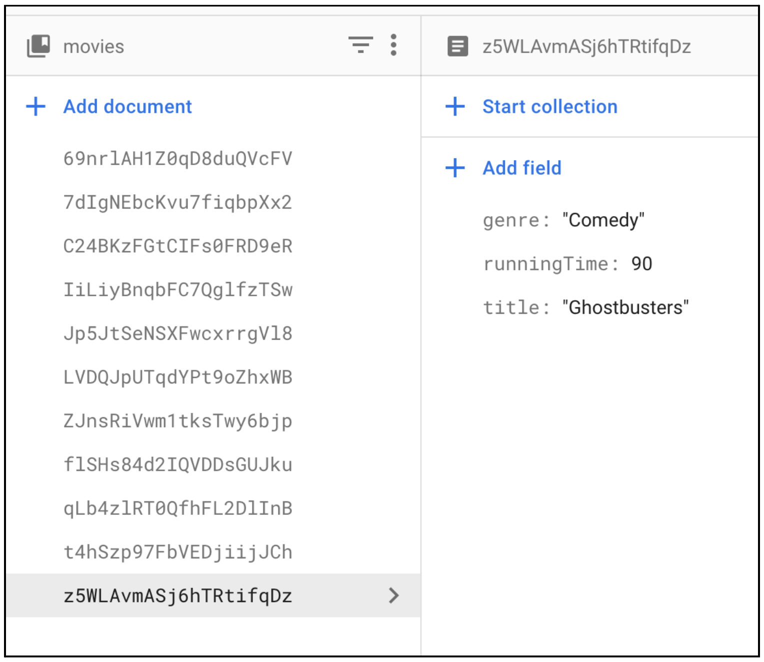


Ghostbusters

90

Comedy

In Firestore:



```
69nrlAH1Z0qD8duQVcFV
7dIgNEbcKvu7fiqbpXx2
C24BKzFGtCIFs0FRD9eR
IiLiyBnqbFC7QglfzTSw
Jp5JtSeNSXFwcxrrgVl8
LVDQJpUTqdYPt9oZhxWB
ZJnsRiVwm1tksTwy6bjp
flSHs84d2IQVDDsGUJku
qLb4zlRT0QfhFL2DlInB
t4hSzp97FbVEDjiijJCh
z5WLAvmASj6hTRtifqDz
```

z5WLAvmASj6hTRtifqDz

```
genre: "Comedy"
runningTime: 90
title: "Ghostbusters"
```

Update and Delete

Deleting an item

Nothing changes with the delete

1. Specify the document id
2. Call db.collection("movies").document(*id*).delete {}

Update an item

You must specify the ID of the item you want to update

.setData(from:*Encodable)*

```
@IBOutlet weak var txtDocumentId: UITextField!


 @IBAction func updatePressed(_ sender: Any) {
        // 1. get the id from the text box
        let id = txtDocumentId.text!
        // 2. Create the object you want to update the information with

        let titleInput = lblTitle.text!
        let runningTimeInput = lblRunningTime.text!
        let genreInput = lblGenre.text!


        // 2. convert the running time to a number
        let runningTime = Int(runningTimeInput) ?? 0

        // 3. Create the object to send
        let movieToUpdate = Movie(title: titleInput, runningTime: runningTime, genre:
genreInput)

        // 4. Send the object
        do {
            try db.collection("movies").document(id).setData(from: movieToUpdate)
            print("Movie updated")
        }
        catch {
            print("Error!")
        }

    }
```