Practical 9b: Using Firebase for Data Persistency

In this lab, we will make use of Firebase, a database engine used in iOS application to handle data stored in the cloud.

Section 1a: Add All Necessary Frameworks (CocoaPods)

1. Download and unzip the 10b-MovieApp from Blackboard.
2. Open Terminal from Launchpad.
3. Type 'sudo gem install cocoapods' and press Enter.
4. Use 'cd' to change the directory to where you 10b-MovieApp project is.
5. In Terminal, type 'pod init' and press Enter.
6. Using Finder, double click on the podfile created in your project folder to open it in TextEdit
7. Add the following lines to the end of the podfile:

```
pod 'Firebase'
pod 'Firebase/Database'
```

   Save the podfile and close it.
8. Back in Terminal, type 'pod install' and press Enter. It will take a while to download and install the necessary files.
9. Open your **MovieApp.xcworkspace** in XCode. (Do not open the MovieApp.xcodeproj file!)
10. In XCode, assign your own unique Bundle ID to the MovieApp project.

NOTE: Step 8 may take a long while to complete due to rate limiting on Github. If too slow, consider using the instructions for Section 1b.

Section 1b: Add All Necessary Frameworks (Manual Installation)

NOTE: Use this method only if the instructions in Section 1a is too slow, or if you do not have admin rights to the Mac machine.

1. Download and unzip the 10b-MovieApp from Blackboard.
2. Download and unzip the Firebase Framework from https://dl.google.com/firebase/sdk/ios/3_6_0/Firebase.zip
3. Drag and drop the following files from the Firebase Framework to below your second-level MovieApp project. Ensure that you select "Copy Items When Needed" when you are prompted. You do not need to copy the files in the same folder structure of the files that you downloaded from Google.

   Analytics\FirebaseAnalytics.framework

   Analytics\FirebaseInstanceID.framework

   Analytics\GoogleInterchangeUtilities.framework

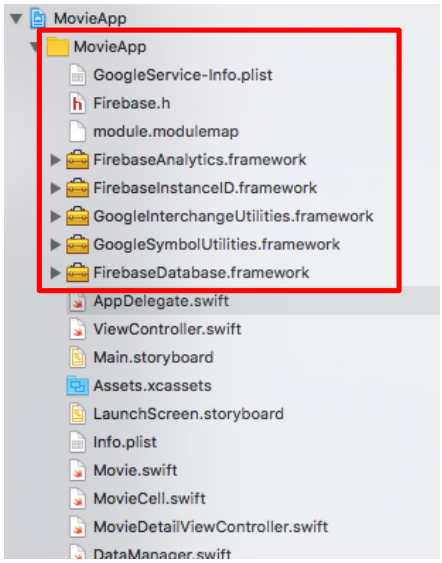
   Analytics\GoogleSymbolUtilities.framework

   Database\FirebaseDatabase.framework

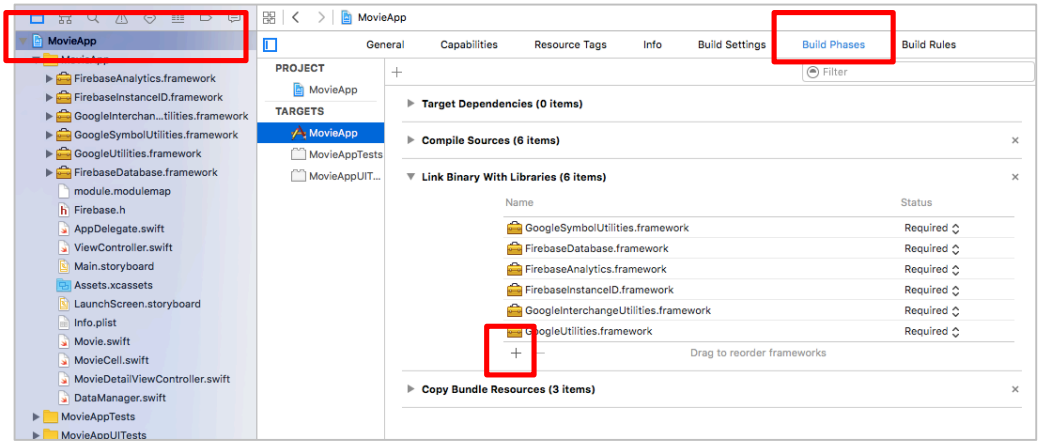
Module.modulemap

Firebase.h

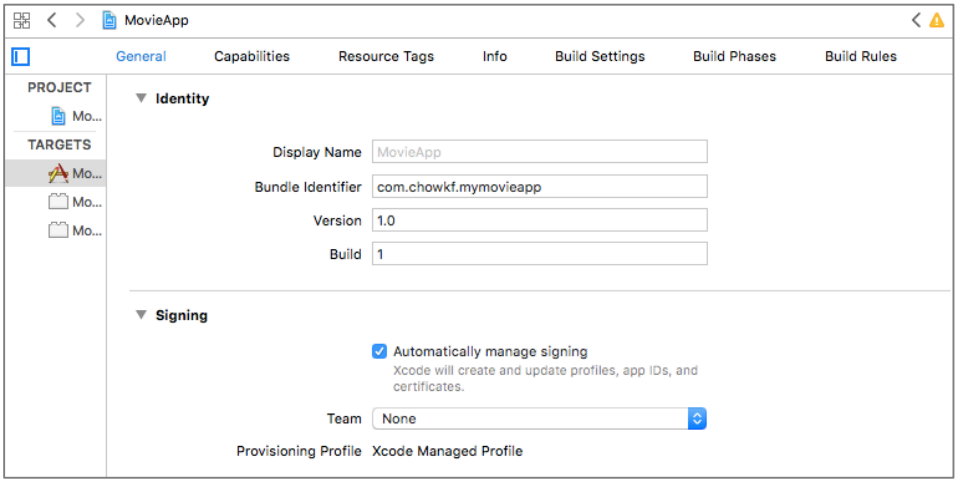
4. Your XCode project file structure should look like the following:



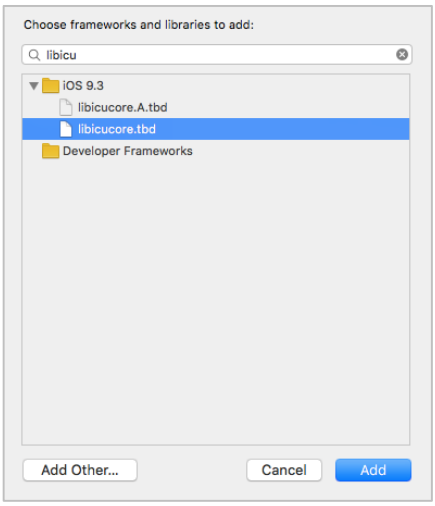
5. Click on your top-level MovieApp project, then click on Build Phases. Then click on the + button.



6. Click on General, enter a Bundle Identifier that consists of your own name and .mymovieapp. This Bundle ID will be used to set up your Firebase app later.

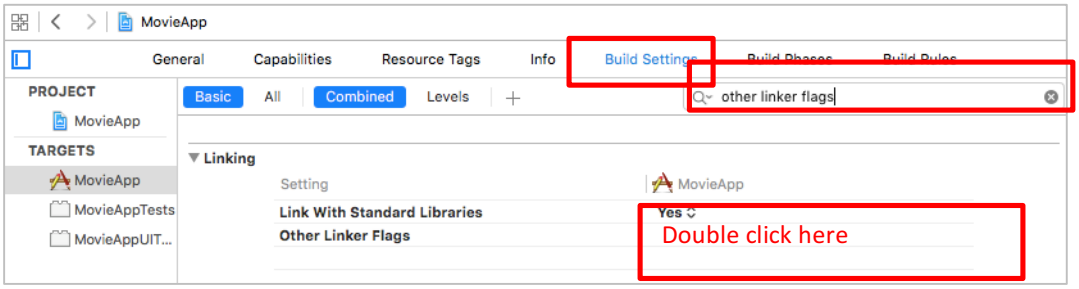


7. When the dialog box appears, search and select **libicucore.tbd**, then click Add.

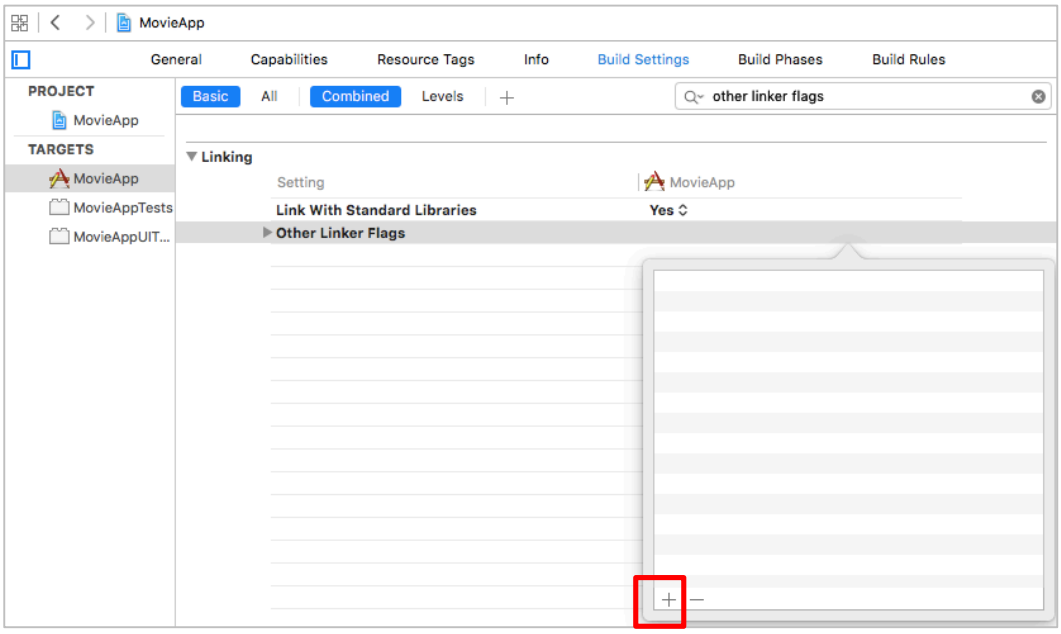


8. In the same way, add **AddressBook.framework**, **libc++.tbd** and **libsqlite3.tbd**.

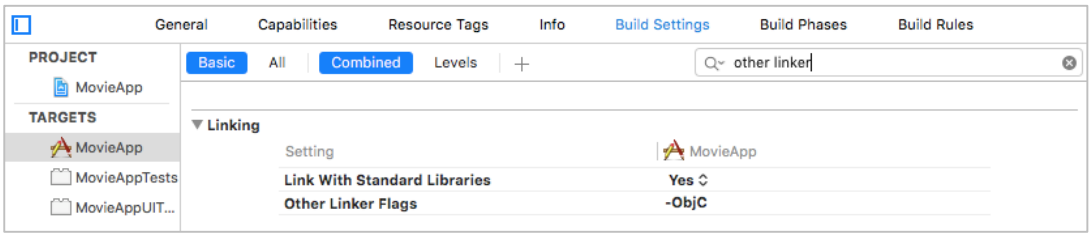
9. Now click Build Settings, search for "Other Linker Flags" and double click on the Other Linker Flags settings.



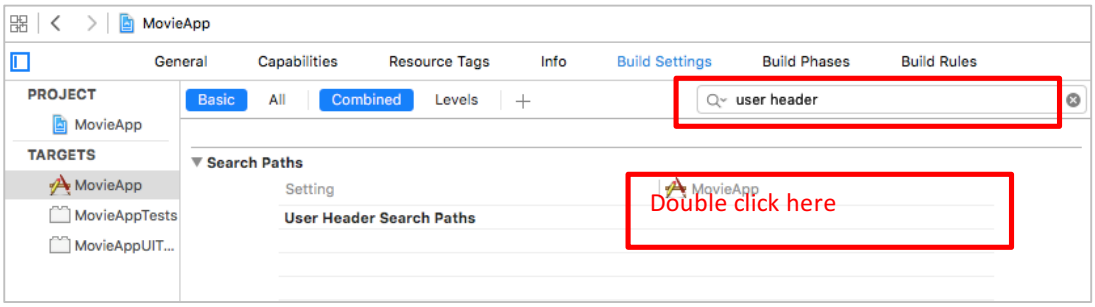
10. In the pop-over that appears, click on the + button and enter "-ObjC".



11. In the pop-over that appears, click on the + button and enter "-ObjC". Your settings should eventually look like that:

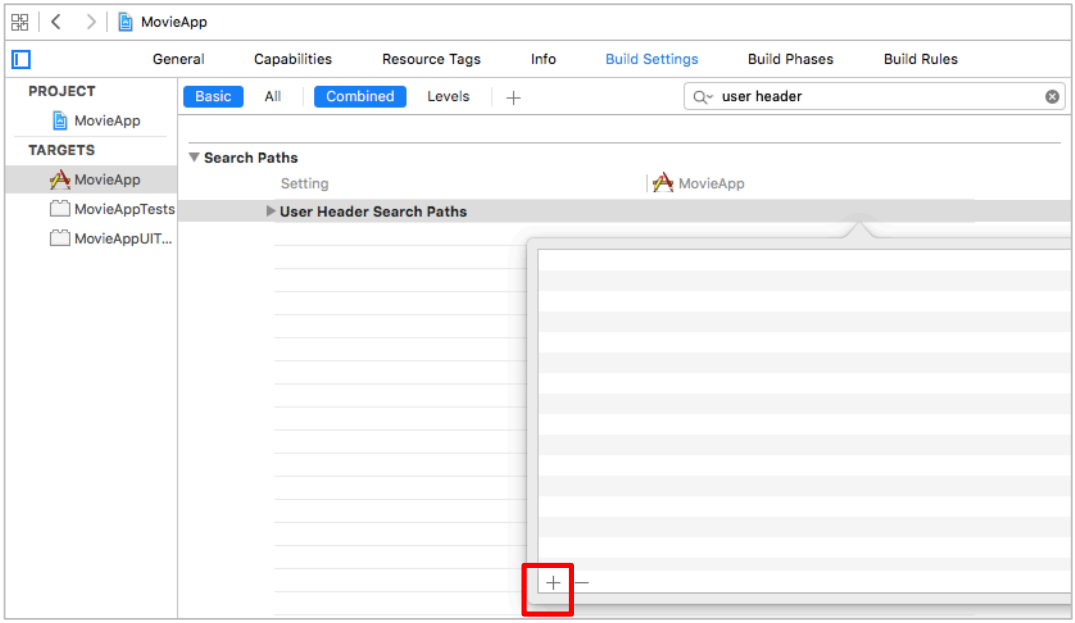


12. In the same Build Settings screen, search for "User Header"



13. Double-click on the User Header Search Paths option.

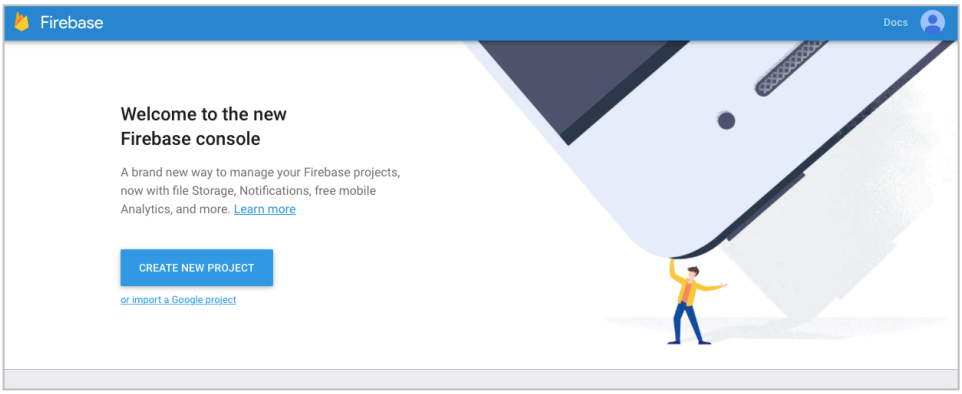
14. In the pop-over that appears, click on the + button and enter the path of the MovieApp project that contains the Firebase.h file.



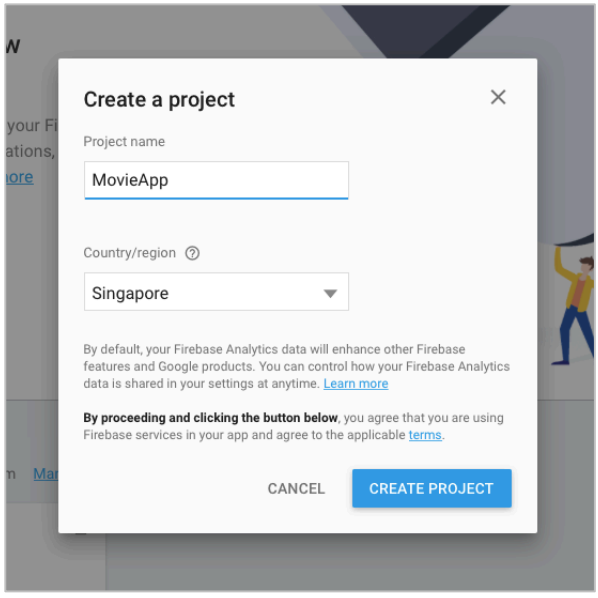
15. You should be able to compile the application but it will run with errors.

Section 2: Create a New Firebase Database

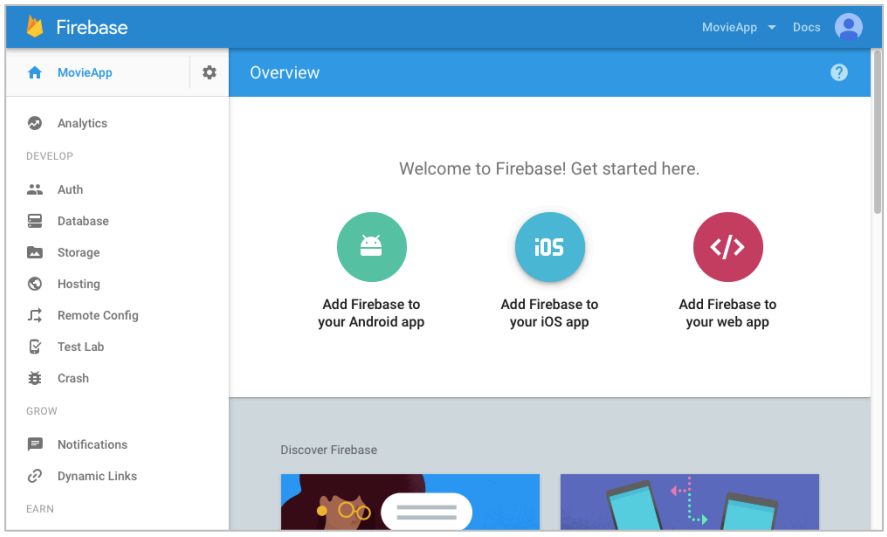
1. In this section, we will sign up for a new Firebase account and create a new Firebase database to be used by our application.
2. Open your browser and navigate to https://firebase.google.com
3. If this is your first time using Firebase, click the Sign Up with Google button and proceed with the instructions to complete your sign up.
4. Once complete, you should see the following screen:



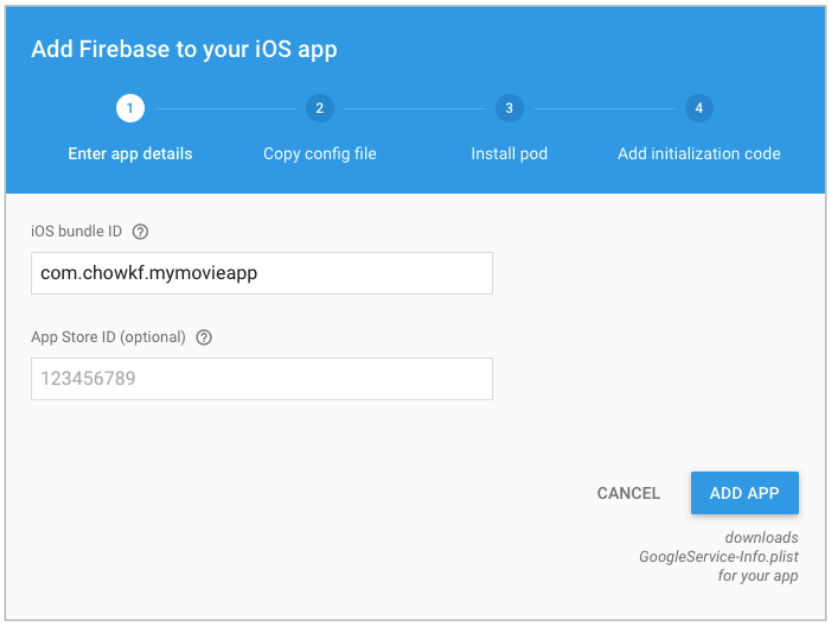
5. Although an existing app has been created for you, we will create a new one ourselves. Click on CREATE NEW PROJECT.



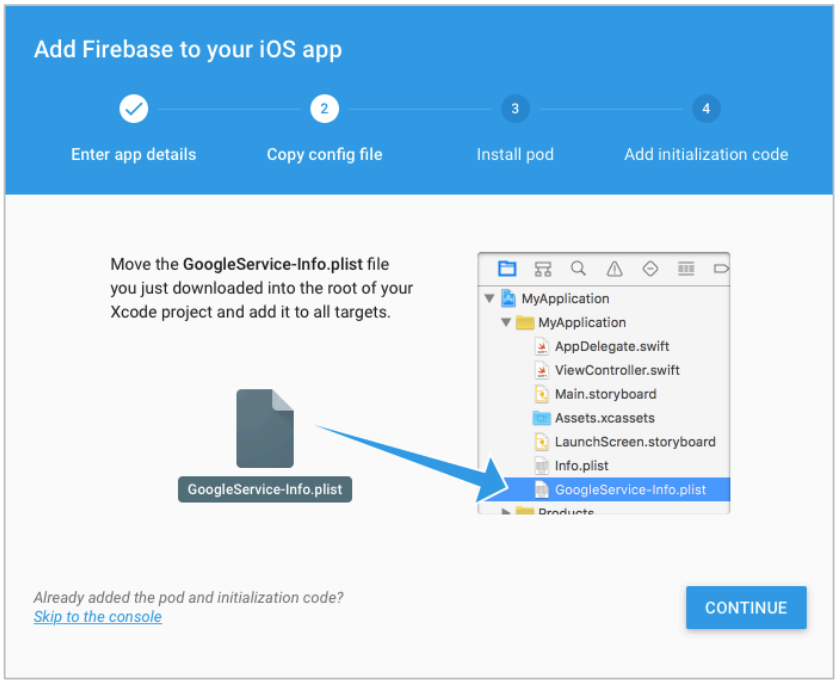
6. Enter a project name and select Singapore as your country. Cick CREATE PROJECT.
7. Once your project is created, you should see the following:



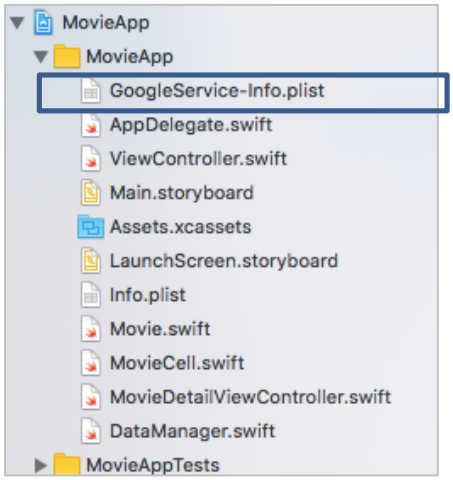
8. Click on the "Add Firebase to your iOS app" button, and the following screen will show up.



9. In the screen that shows up, enter your MovieApp's unique Bundle ID. Use the same one that you have entered into your Project's General Settings in the previous section. Then click **ADD APP**.
10. Your browser will show the following screen and automatically download a file GoogleService-Info.plist into your Downloads folder.

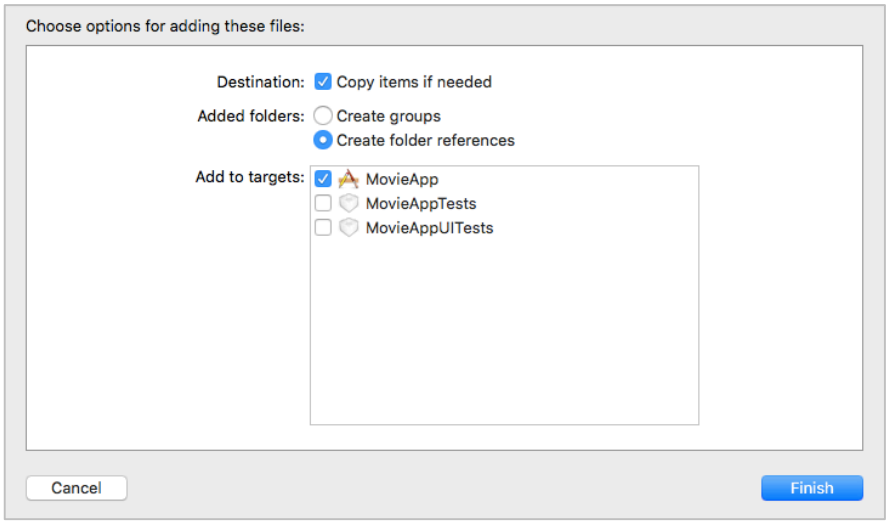


11. Go to your Finder, navigate to your Downloads folder and drag-and-drop the GoogleService-Info.plist file into your MovieApp project in XCode.



Drag-drop your GoogleService-Info.plist here

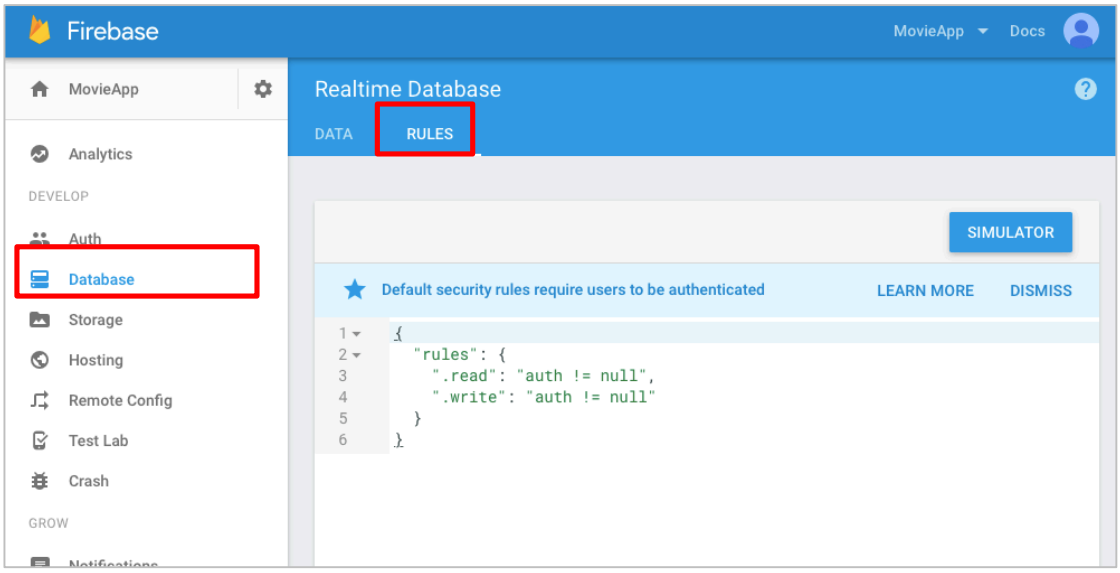
12. When prompted, ensure that you check Copy Items if Needed:



13. In XCode, open your AppDelegate.cs file and insert the following highlighted lines to initialize Firebase.

```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(application: UIApplication,
        didFinishLaunchingWithOptions launchOptions:
        [NSObject: AnyObject]?) -> Bool {

        FIRApp.configure()

        // Override point for customization after application launch.
        return true
    }

    ...
}
```

14. Back in your browser, click “CONTINUE” until your complete the dialog box. We have already done all the pod installations and the initialization of Firebase in our AppDelegate class.
15. In your browser, click on the Database on the left side panel, and click RULES at the tab.



16. Then edit the security rules on the right side of the screen to look like the following:

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

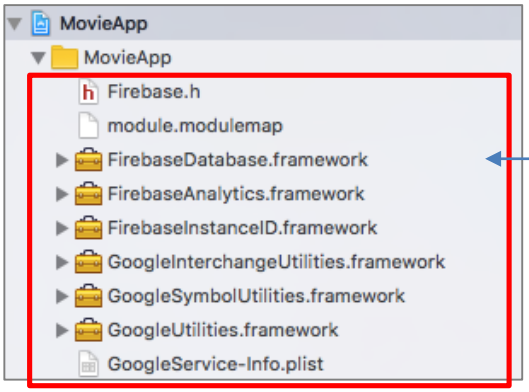
NOTE: The purpose of this is the allow all users to be able to read / write data to your movie database. In a real application, you may want to only allow users who have logged in to Firebase to be able to read / write data, but this will not be the scope of this module.

17. Click on the PUBLISH button that appears at the top of the rules you have just edited.

18. Try to run your application to ensure that you do not have any errors. If you encounter errors with the following message when the app starts up.

    - Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '+[NSData gtm_dataByGzippingData:]: unrecognized selector sent to class

You can remove all the .frameworks, Firebase.h, module.modulemap from your MovieApp folder and re-add them back in again. This time, include the **GoogleUtilities.framework**.



If you have issues running the app, delete these files and add them back in again.

Section 3: Implement DataManager

1. In this section, we will implement some of the CRUD operations in the **DataManager.swift** class. This class is responsible for all loading, updating and deleting of movies from the Sqlite database. It will do those by calling functions from the Firebase API.

2. In the **DataManager.swift**, we need to import Firebase:

```
import Foundation
import Firebase

class DataManager
{
   ...
```

3. In the **DataManager.swift**, we need to implement the **loadMovies** method.

```
class DataManager: NSObject {

    ...

    // Loads the full list of movies from Firebase
    // and converts it into a [Movie] array.
    //
    // Since this is asynchronous, we need an
    // onComplete closure, a piece of code that can
    // be triggered to run once the loading from
    // Firebase is complete.
    //
    static func loadMovies(onComplete: @escaping ([Movie]) -> Void)
    {
        // create an empty list.
        var movieList : [Movie] = []

        let ref = FIRDatabase.database().reference().child("movies/")

        // observeSingleEventOfType tells Firebase
        // to load the full list of Movies, and execute the
        // "with" closure once, when the download
        // is complete.
        //
        ref.observeSingleEvent(of: .value,
            with:
            { (snapshot) in

                // This is the "with" closure that
                // executes only when the retrieval
                // of data from Firebase is complete.
                // Meanwhile, before the download is complete,
                // the user can still interact with the user
                // interface.
                //
                for record in snapshot.children
                {
                    let r = record as! FIRDataSnapshot

                    print (r.key)

                    movieList.append(Movie(id: r.key,
                        name: r.childSnapshot(forPath: "name").value
as! String,
                        description: r.childSnapshot(forPath:
"desc").value as! String,
```

```
                            runTime: r.childSnapshot(forPath:
"runtime").value as! Int,
                            imagePath: r.childSnapshot(forPath:
"image").value as! String))

                }

                // Once the list of movies have been downloaded,
                // call a function (a closure) passed in through
                // the onComplete parameter. This allows the caller
                // to do any further processing or update the UI.
                //
                onComplete(movieList)
            })
    }
}
```

4. In the **DataManager.swift**, we need to implement the **insertOrReplaceMovie** method.

```
class DataManager: NSObject {

    ...

    // Inserts or replaces an existing movie
    // into Firebase.
    //
    static func insertOrReplaceMovie(_ movie: Movie)
    {

        let ref = FIRDatabase.database().reference()
            .child("movies/\(movie.movieID!)/")

        ref.setValue([
            "name" : movie.movieName!,
            "desc" : movie.movieDesc!,
            "runtime" : movie.runtime,
            "image" : movie.imagePath!])
    }

}
```

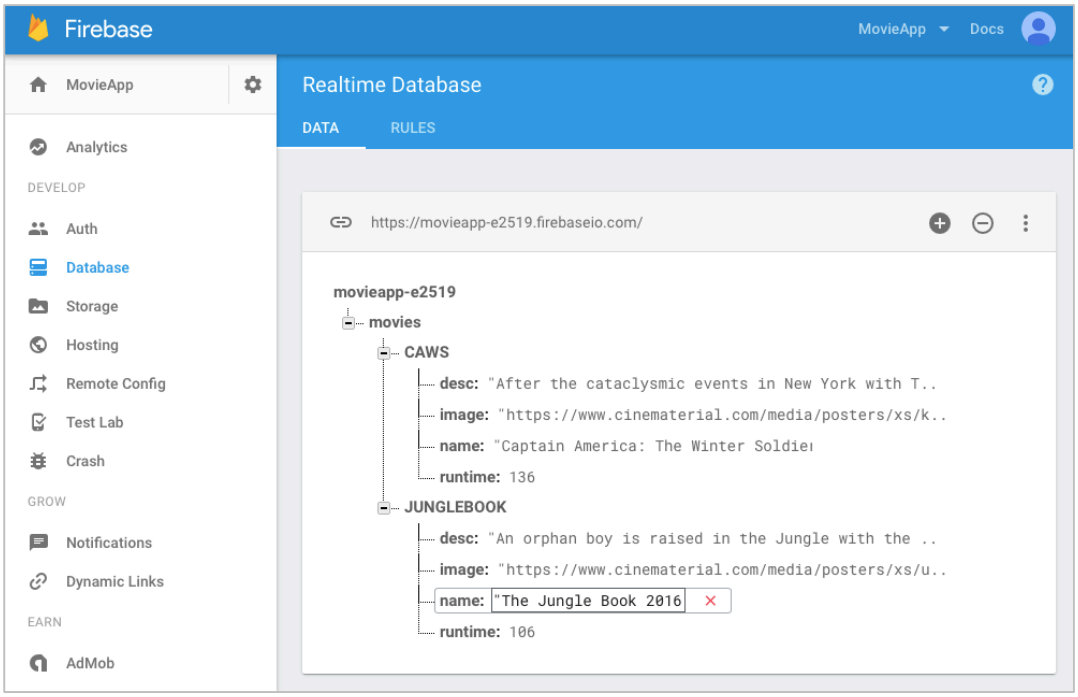
5. In the **DataManager.swift**, we need to implement the **deleteMovie** method.

```
class DataManager: NSObject {

    ...

    // Deletes a movie from the Firebase database.
    //
    static func deleteMovie(_ movie: Movie)
    {
        let ref = FIRDatabase.database().reference()
            .child("movies/\(movie.movieID!)/")

        ref.removeValue()
    }
}
```

6. In **ViewController.swift**, modify the **loadMovies()** function to call the DataManager.loadMovies function. This function is called when the ViewController's viewDidLoad is triggered, and when the user adds / updates a movie in the MovieDetailViewController.

```
class ViewController: UIViewController,
    UITableViewDelegate, UITableViewDataSource {
    ...


    // This function reloads the movies from the Firebase
    // server. Once complete, gets the table view to
    // refresh itself.
    //
    func loadMovies()
    {
        // This is a special way to call loadMovies.
        //
        // Even if loadMovies accepts a closure as a
        // parameter, I can pass that parameter after
        // the round brackets. In a way it is
        // easier to read.
        //
        DataManager.loadMovies ()
        {
            movieListFromFirebase in

            // This is a closure.
            //
            // This block of codes is executed when the
            // async loading from Firebase is complete.
            // What it is to reassigned the new list loaded
            // from Firebase.
            //
            self.movieList = movieListFromFirebase

            // Once done, call on the Table View to reload
            // all its contents
            //
            self.tableView.reloadData()

        }
    }
}
```

7. Run your application to add some movies.
8. See the list of movies appear in both your Table View and the Firebase database backend. Understand why the data is structured this way.


Firebase
MovieApp
Docs
MovieApp
Realtime Database
DATA
RULES
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications
Dynamic Links
EARN
AdMob
https://movieapp-e2519.firebaseio.com/
movieapp-e2519
movies
CAWS
desc: "After the cataclysmic events in New York with T..
image: "https://www.cinematerial.com/media/posters/xs/k..
name: "Captain America: The Winter Soldier
runtime: 136
JUNGLEBOOK
desc: "An orphan boy is raised in the Jungle with the ..
image: "https://www.cinematerial.com/media/posters/xs/u..
name: "The Jungle Book 2016
runtime: 106

Section 4: Loading Images From the Web

1. The current application doesn't load any images from the resource anymore, so the image part of the movies list is always empty. We are going to change the behavior so that your app will download movie images from the internet to display it.
2. In **ViewController.swift**, let's first create a function **loadAndDisplayImage**. This function is responsible for downloading the image from the internet and displaying it in your user interface.

```
class ViewController : UIViewController,
    UITableViewDelegate, UITableViewDataSource {


    ...

    // This function loads images from
    // the internet and displays it in an UIImageView
    // control
    //
    func loadAndDisplayImage(imageView: UIImageView, url: String)
    {
        // The following lines download data from the
        // internet. This data should represent an
        // image in the JPG, GIF, PNG or any acceptable
        // graphics format.
        //
        let nurl = URL(string: url)
        var imageBinary : Data?
        if nurl != nil
        {
            do
            {
                imageBinary = try Data(contentsOf: nurl!)
            }
            catch
            {
                return
            }
        }

        // After retrieving the data, we convert
        // it to an UIImage object.
        //
        var img : UIImage?
        if imageBinary != nil
        {
            img = UIImage(data: imageBinary!)
        }

        // These codes set the downloaded image
        // to the UIImageView control.
        //
        imageView.image = img

    }
}
```

3. In ViewController.swift, modify the tableView(cellForRowAt:) function to call the function you wrote above.

```
class ViewController : UIViewController,
    UITableViewDelegate, UITableViewDataSource {


    ...


    func tableView(_ tableView: UITableView,
        cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {

        // First we query the table view to see if there are
        // any MovieCells that are no longer visible
        // and can be reused for a new table cell
        // that needs to be displayed.
        //
        let cell = tableView.dequeueReusableCell(
            withIdentifier: "MovieCell", for: indexPath)
            as! MovieCell

        // Using the re-used cell, or the newly created
        // cell, we update the text label's text property.
        //
        let p = movieList[(indexPath as IndexPath).row]
        cell.nameLabel.text = p.movieName!
        cell.runTimeLabel.text = "\(p.runtime) mins"

        loadAndDisplayImage(imageView: cell.movieImageView,
            url: p.imagePath!)

        return cell
    }
}
```

4. Try to modify your movies to include images from the internet. You can search for movie images at www.cinematerial.com. You may use some of the following URLs for your movie images:

   Star Wars – The Force Awakens

   https://www.cinematerial.com/media/posters/xs/ls/lse20kd6.jpg

   Batman vs Superman

   https://www.cinematerial.com/media/posters/xs/xq/xqyl4hw0.jpg

   The Jungle Book 2016

   https://www.cinematerial.com/media/posters/xs/uy/uy8spv6t.jpg

   Deadpool

   https://www.cinematerial.com/media/posters/xs/mx/mxxovvpf.jpg

   The Revenant

https://www.cinematerial.com/media/posters/xs/ui/uian13eu.jpg

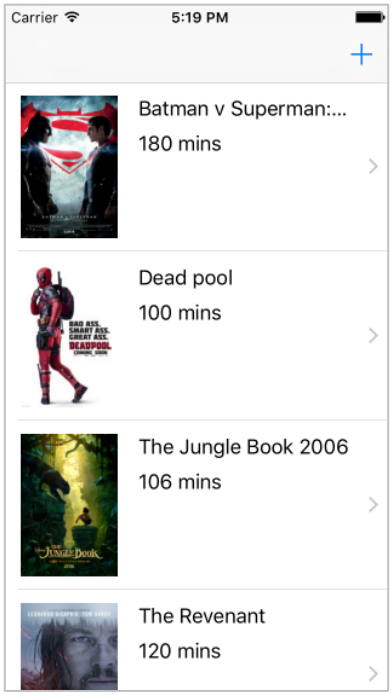
Mad Max: Fury Road

https://www.cinematerial.com/media/posters/xs/ts/tsumoawx.jpg

Zootopia

https://www.cinematerial.com/media/posters/xs/c6/c6pg81dz.jpg

5. Try adding many movies, either through the app's user interface, or through Firebase's user interface.



6. Restart you application and see the behaviour of the table view when you have many movies with images. You will notice that at some point, your app may become unresponsive. This is because the main thread is spending too much time waiting for the images from the internet to load completely.
7. To resolve that, add the following lines to perform the image download asynchronously using a background thread.

```
class ViewController : UIViewController,
    UITableViewDelegate, UITableViewDataSource {

    ...

    // This function loads images from
    // the internet and displays it in an UIImageView
    // control
```

```
    //
    func loadAndDisplayImage(imageView: UIImageView, url: String)
    {
        DispatchQueue.global(qos: .background).async
        {

            // The following lines download data from the
            // internet. This data should represent an
            // image in the JPG, GIF, PNG or any acceptable
            // graphics format.
            //
            let nurl = URL(string: url)
            var imageBinary : Data?
            if nurl != nil
            {
                do
                {
                    imageBinary = try Data(contentsOf: nurl!)
                }
                catch
                {
                    return
                }
            }

            DispatchQueue.main.async
            {

                // After retrieving the data, we convert
                // it to an UIImage object.
                //
                var img : UIImage?
                if imageBinary != nil
                {
                    img = UIImage(data: imageBinary!)
                }

                // These codes set the downloaded image
                // to the UIImageView control.
                //
                imageView.image = img

            }

        }

    }

}
```

8. Run your app again to experience the difference.

For more iOS Firebase apps in Swift 3:

https://www.raywenderlich.com/139322/firebase-tutorial-getting-started-2

Challenge

1. Try to manually add up to 7-10 movies, all with different images. Then scroll your movie list up and down. Notice anything wrong? Are you able to explain why this happens and resolve the issue?
2. Firebase uses

   ref.queryOrderByChild(…).queryStartingAtValue(…).queryEndingAtValue(…)

   to sort and filter records. Can you implement the following 4 buttons:

   All Movies

   A-H,

   I-P,

   Q-Z,

   so that when the user clicks on one of these buttons, the movie list is filtered accordingly by the first letter of the movie name?