

SMARTPHONES BASED WEB DEVELOPMENT
FINAL PROJECT:
THE BEAUTIFUL GAME - SOCCER APP

YASH KHOPKAR

001850102

Overview

An application that aims to provide users information on soccer news, fixtures, standings and teams from all over the world in one single app. The users must log in with their email and they would have access to all information in a single application. The admin can log in to keep a track of all the users that have registered in the app.

Motivation

Many a times one must visit different websites/apps to find out information on soccer. Like different websites/apps for latest news, game results, club, standings etc. My app aims to simplify this process by providing all the latest soccer information under one roof by aggregating all information in one single app.

Key Tools/Services used

☐ **Google Firebase**

1. FirebaseAuth

2. FirebaseDatabase

☐ **API Calls**









☐ **CoreData**

☐ **GCD**

☐ **Tab View Controller & Web Kit View**

1.1 Firebase Authentication

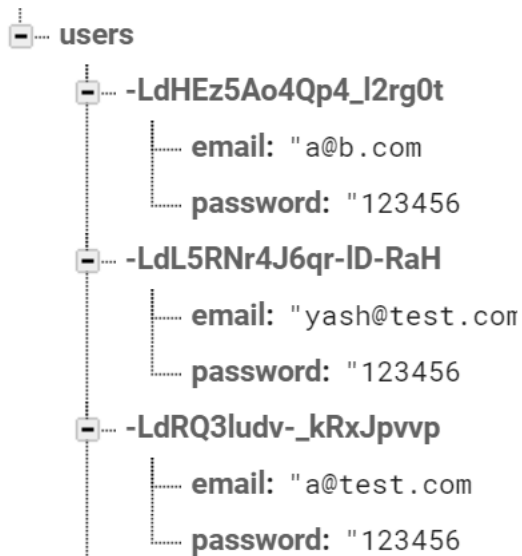
The firebase authentication comes in with the FirebaseAuth kit provided by firebase. This service contains all the user information and enables the administrators to enable logging into their application using social media platforms like Facebook, twitter etc. FirebaseAuth allows us to authenticate users based on which platform they are using to log in.

Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Enabled
 Play Games	Disabled
 Facebook	Disabled
 Twitter	Disabled
 GitHub	Disabled
 Anonymous	Enabled

1.2 Firebase Database

Google Firebase is being predominantly used as a backend server for storing username and password of the registered users. A real-time database is being used which can directly talk to the application and pass data back and forth to the same. Every time a user registers a new entry is inserted into the database. The admin can log in and view all the records inserted.

the-beautiful-game



2. API Calls

All information about news, fixtures, standings and clubs are fetched from API's. They are football.data.org and newsapi.org. Fetched information (mostly JSON) from the API is parsed and displayed in various views.

Below is an example:

```
func parseClubURL(theURL : String){
    let url = URL(string: theURL)
    var request = URLRequest(url: url!)
    request.addValue("b4d02fa1622f4ea3b3970dc1786952cb", forHTTPHeaderField: "X-Auth-Token")
    request.httpMethod = "GET"
    URLSession.shared.dataTask(with: request){ (data, response, error) in
        if(error != nil){
            print("ERROR!")
            //print(error as! String)

            DispatchQueue.main.asyncAfter(deadline: .now()){
            }
        }else{
            do{
                let parsedData = try JSONSerialization.jsonObject(with:data!) as! [String:Any]
                //print(parsedData)
                for(key, value) in parsedData{
                    if(key == "teams"){
                        //print("\(key) ----- \(value)")
                        let teamsArray : [[String:Any]] = value as! [[String : Any]]

                        for teamDict in teamsArray{
                            for(key, value) in teamDict{
                                //print("\(key) ----- \(value)")
                                if(key == "name"){
                                    self.clubNameArr.append(value as! String)
                                }
                                if(key == "shortName"){
                                    self.clubShortNameArr.append(value as! String)
                                }
                                if(key == "tla"){
                                    self.clubTLAArr.append(value as! String)
                                }
                                if(key == "address"){
                                    self.clubAddrArr.append(value as! String)
                                }
                                if(key == "venue"){
                                    self.clubVenueArr.append(value as! String)
                                }
                                if(key == "website"){
                                    self.clubWebsiteArr.append(value as! String)
                                }
                            }
                        }
                    }
                }
            }catch{
                print("Error parsing JSON")
            }
        }
    }
}
```

3. CoreData

CoreData is used to display to temporarily store data from the API and display in various views. When the app terminates all the data stored in CoreData is erased. When the app relaunches again fresh API calls are made to ensure that the app always contains the latest information.

```
281         var count = self.homeTeam.count - 1
282         while(count != 0){
283             let game = Game(context: PersistenceService.context)
284             game.id = UUID()
285             game.homeTeam = self.homeTeam[count]
286             game.awayTeam = self.awayTeam[count]
287             game.homeScore = Int16(self.homeScore[count])
288             game.awayScore = Int16(self.awayScore[count])
289             game.date = self.matchDate[count]
290             count = count - 1
291             //PersistenceService.saveContext()
292         }
293     }catch let error as NSError {
294         print(error)
295         DispatchQueue.main.asyncAfter(deadline: .now()){
296
297         }
298     }
299 }
300
301 }.resume()
302 }
```

4.GCD

The GCD or Grand Central Dispatcher allows us to incorporate multithreading in our applications. Here, however we need GCD to perform a few CPU intensive tasks asynchronously. For example resizing a very large image to proper size so that it can be displayed in the Image View.

```
--
57     DispatchQueue.global(qos: .background).async {
58         //print("This is run on the background queue")
59         if let data = try? Data(contentsOf: url ?? URL(string:"https://www.horizont.net/news/media/27/Luca-vo-Cranac-is-CE-un-Grnde-vo-Onefootball-268618.jpeg")!)
60         {
61             let image: UIImage = UIImage(data: data)!
62             let myThumb = image.resized(withPercentage: 0.050)
63             cell.newsImg.image = myThumb
64             //cell.newsImg.contentMode = .center
65             //print(data)
66         }
67
68         DispatchQueue.main.async {
69             //print("This is run on the main queue, after the previous code in outer block")
70         }
71     }
72
```

UIImageView.image must be used from n

5.Tab View Controller & Web Kit View

The Tab View Controller is used to display various views namely Games, News, Clubs and Tables. The Web Kit View is used to display embedded Web content i.e. team websites and news articles.

```
var targetNews : News? = nil
@IBOutlet weak var webView: WKWebView!

override func viewDidLoad() {
    super.viewDidLoad()

    // Do any additional setup after loading the view.

    let url = URL(string: (targetNews?.url)!!)
    webView.load(URLRequest(url: url))
    webView.allowsBackForwardNavigationGestures = true
}

override func loadView() {
    webView = WKWebView()
    webView.navigationDelegate = self
    view = webView
}
```

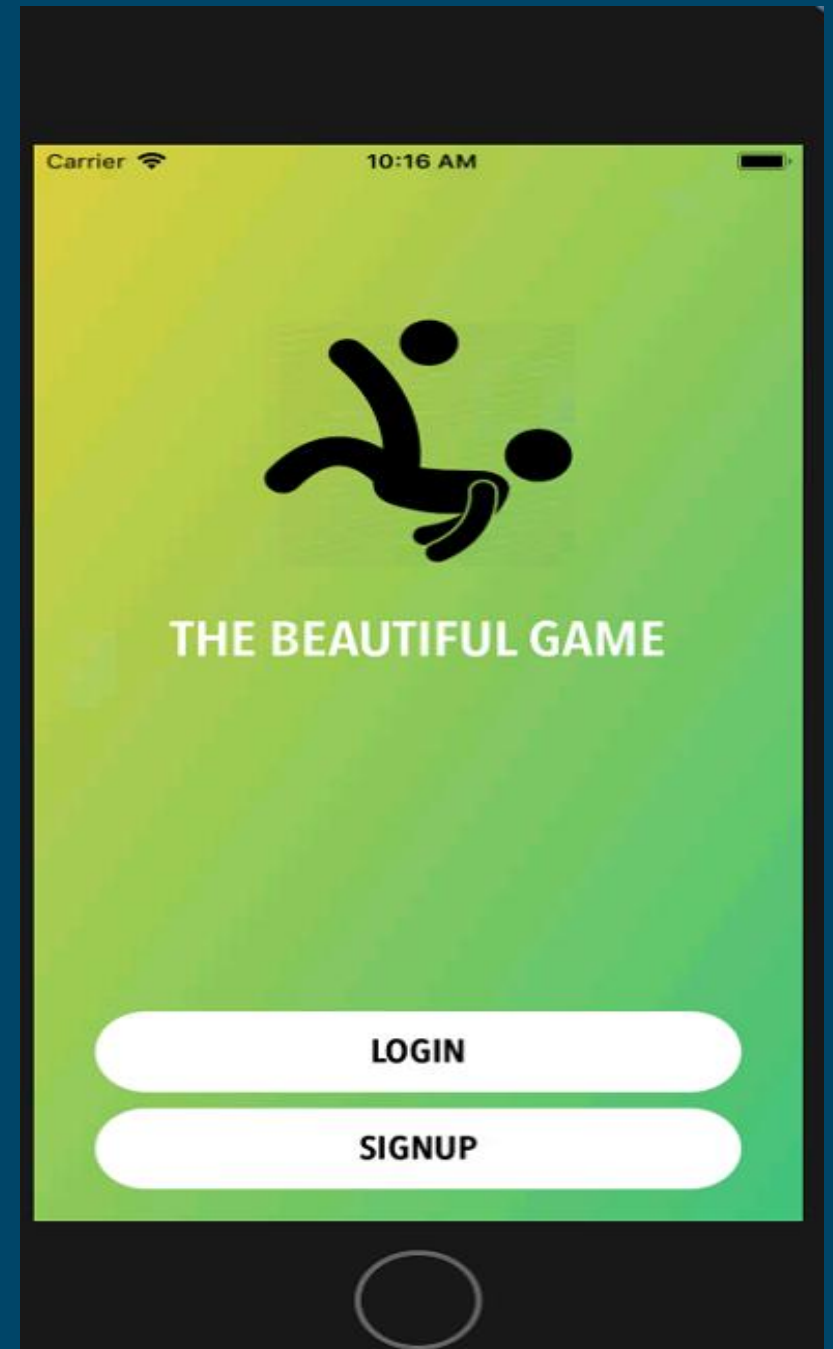

5.Future Prospects

Future prospects would include the following :

- ☐ Information on multiple leagues all over the world having multiple filters.
- ☐ Live Score Updates & Push Notifications featuring them.
- ☐ Deployment

The Beautiful Game : Soccer App

Yash Khopkar



Project Description & Motivation

My app aims to provide users with access to all news and fixtures happening in soccer all around the world.

Rather than visiting different websites to know about the latest news and fixtures, I intend to aggregate and display them all at once.

Features

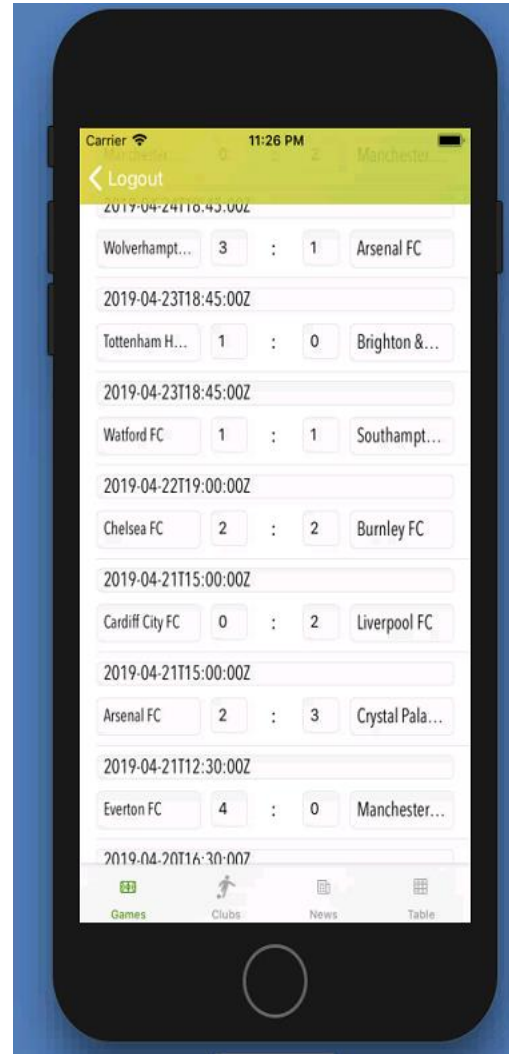
- Role Base Access
- API Calls
- Firebase Authentication
- Core Data Database
- Tab View Controller & Web Kit View

Role Based Access

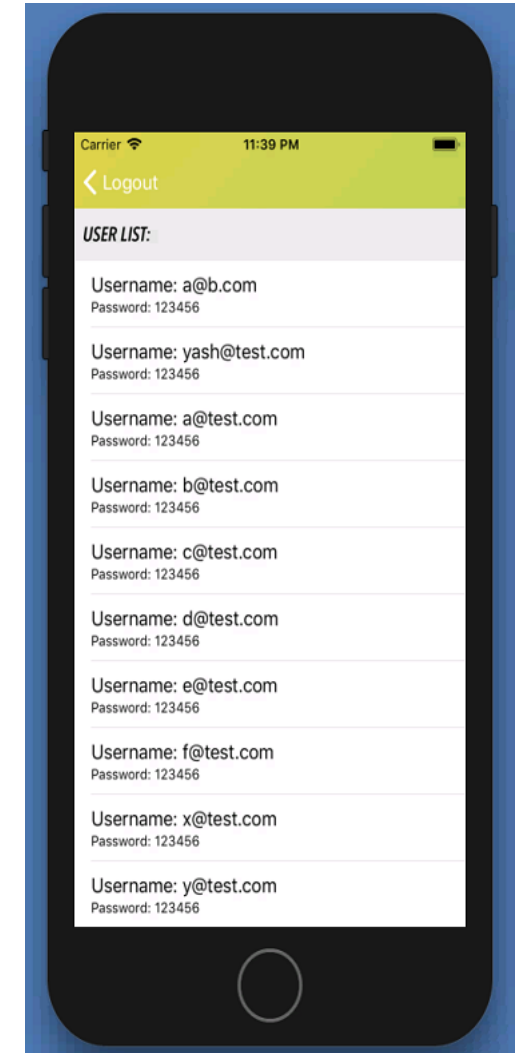
- Two Roles:
- User Role :
- Users can signup/login to view football matches, standings clubs and news.
- Admin Role
- Admin can log in to keep a track of users registered.

Landing Pages

User Role



Admin Role



API Calls

- All the data in the app is made available using calls to multiple API'S
- They are : football.data.org and newsapi.org

Example Code

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













    //Uncomment the below lines of code when running the app for the very first time.

    parseClubURL(theURL: "http://api.football-data.org/v2/competitions/PL/teams")
    parseMatchURL(theURL: "http://api.football-data.org/v2/competitions/PL/matches")
    parseTableURL(theURL: "http://api.football-data.org/v2/competitions/PL/standings")
    parseNewsURL(theURL: "https://newsapi.org/v2/everything?q=OneFootball&from=2019-03-26&sortBy=publishedAt&apiKey=532b70ffa6c94e9eb9acc809e26affc1")
    FirebaseApp.configure()
    return true
}
```


Firestore Auth

- Firestore Authentication is used to signup new users.
- It is also used to sign in existing users.

Example

<div><div> Search by email address, phone number or user UID</div><div><div>Add user</div><div> </div></div></div>				
Identifier	Providers	Created	Signed In ↓	User UID ↑
x@test.com		26 Apr 2019	26 Apr 2019	1HrT8fBbT9cvehvhvNvhfl4IWuu1
a@b.com		24 Apr 2019	27 Apr 2019	2brkwU8LAFS7Qj0qTsu649iZ2t62
y@test.com		26 Apr 2019	26 Apr 2019	BNxAfYtIhnY0tb9IPsGelo8M7oc2
yash@test.com		25 Apr 2019	25 Apr 2019	DNt777yUezPtIon1T2amkoFQXUj1
a@test.com		26 Apr 2019	26 Apr 2019	WLCFnn6m2vMCXG3UgJXKWYm...
c@test.com		26 Apr 2019	26 Apr 2019	Zb3oWD62TSYLnI5wZKINn7UkS9G2
b@test.com		26 Apr 2019	26 Apr 2019	cRt7kcBaa2Zd2Dfh0glfXZwCGfm1
e@test.com		26 Apr 2019	26 Apr 2019	cooTD9zgiHe2GX3rg3MCrbt6NSX2
f@test.com		26 Apr 2019	26 Apr 2019	m36EJg2dsW1p43wbtA8ecgbFcG3

CoreData

- The data from the API is fetched and stored using CoreData.
- This data is further used to populate various views.

Example

Club
▼ Attributes
address
crestURL
id
name
shortName
tla
venue
website
▼ Relationships

News
▼ Attributes
content
date
id
imgURL
title
url
▼ Relationships

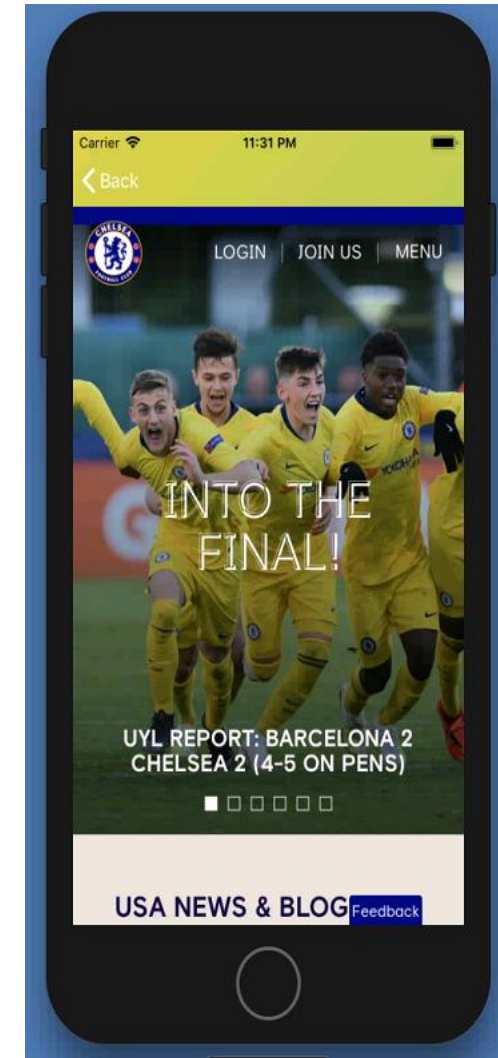
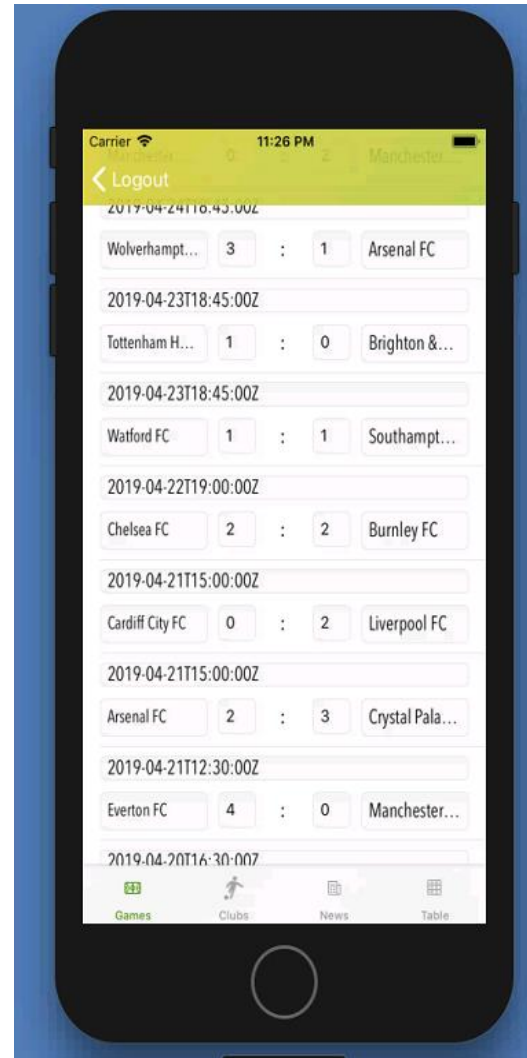
Game
▼ Attributes
awayScore
awayTeam
date
homeScore
homeTeam
id
▼ Relationships

Table
▼ Attributes
club
goalDifference
goalsAgainst
goalsFor
id
playedGames
points
position
▼ Relationships

Tab View Controller & Web Kit View

- I am using TabViewController to present various sections in the app namely games, clubs, news and standings.
- I am using Web Kit View to display club websites and news articles.

Example



Future Scope

- Multiple filters.
- Data on Multiple leagues around the world and World Cup!
- Live score and updates (Notifications!)
- Deployment on AWS??

Thank You

