**DStv iOS Practical Test**

This practical test is aimed at testing basic UI creation with some connectivity to a few services to retrieve some info to be displayed.

Various aspects of the individual taking this test will be evaluated. You’ll need to show your abilities in the following areas of iOS development:

* Ability to create simple UIViews and UIViewControllers
* Navigation between two screens using segues and UINavigationControllers
* Ability to detect and handle simple touch events
* Basic UI animation
* Ability to consume services to download/upload data from/to a remote server
* Ability to retrieve and deserialize JSON
* Ability to design and properly manage object instances in an iOS App
* Basic design patterns

**Task background**

Your task for this test is to develop the initial stages of a chat client. This chat client must consist of at least 3 (three) screens, each responsible for displaying different areas of our App. The App must be capable of performing the following tasks:

* Present a user with a login screen to enter their username and password
* Allow the user to login
* On successful login, present the user with their list of friends
* The user must then be able to view more details about each of their friends.

**Technical requirements**

Below, the app is broken down into its various parts detailing the technical requirements for each.

**Login screen**

On the login screen, the user must be given the option to enter their username and password. When the user taps the “Login” button, the users credentials must be posted as JSON in the body of an HTTP POST request. The key to be used for the username is “username”, and the key for the password must be “password”:

The username is **user** and the password **1234**

The server will respond with a JSON output that indicates a successful login, or a failure. In the event of a failure, handle this is an appropriate manner. If the login is successful, then the user must be taken to the next screen.

Login endpoint: **http://mobileexam.dstv.com/login**

**Friends screen**

When a user successfully logs in, transition the user to the “Friends Screen” using a UISegue (a custom segue will be advantageous). This screen must be presented inside a UINavigationController with “Friends” as the title. The App must then download the list of friends for the logged in user, using the unique GUID, and the users’ first name, that would have been received in the login successful JSON response.

The list of friends are to be retrieved using an HTTP GET request, with the unique ID passed as a URL matrix parameter, with the key ‘uniqueID’. Also send through the users’ first name with the key ‘name’.

Once the friends’ list is retrieved, the “Friends Screen” should be refreshed to show the list of friends that the logged in user has. For each friend, the following details must be displayed:

* An image for the user (to be loaded asynchronously)
* The friends alias name
* The last time the user was online.
* An “info” button on each friend allowing the user to tap on it to see more info about the user

The user can then scroll through their list of friends and select the “info” to view more details about each friend.

Endpoint: **http://mobileexam.dstv.com/friends**

**Details screen**

The details screen for a friend simply displays all fields (except the ID) that are received for each friend. The design is at your discretion. For this screen, a simple, unobtrusive animation should be incorporated when the screen is fully presented.

**Final thoughts**

The aim of this practical test is to see your understanding of Swift and/or Objective-C and Apple design methodologies using Xcode as the IDE of choice. In the past, Swift was introduced by Apple as their language of choice going forward for iOS and Mac based Apps. You are welcome to develop the following App using any of these languages. You will not be penalized in any way based on your choice of language.

As Xcode seamlessly allows for the mixing of Objective-C and C, you are encouraged to stick to Objective-C as far as possible, using C only if necessary.

Also, no dependencies on third party components, and additional functionality over and above the requirements of this document, will be advantageous.