Name : Lee Kang Wei Date : 16th Oct 2020

General.

1. What version(s) of iOS that you have worked with?

iOS Version	Years of Experience
≤ iOS 11	-
iOS 12	3 (months)
iOS 13	1

2. What are the language(s) for iOS that you have worked with?

Language	Release/Version	Years of Experience
Swift	4.2 - 5.2	1.5
Objective-C		-
Others (Please specify):		-

3. What are some of the IDEs and your choice of IDE when developing iOS applications?

4. What are some of the source control systems that you have used?

Name	Skill Level (Basic/Intermediate/Advance)
GIT	Intermediate
SVN	-
Others (Please specify):	-

5. What are some of the iOS build tools that you have used?

Name	Skill Level (Basic/Intermediate/Advance)
Xcode	Advance
Xcode Server	Basic
Others (Please Specify):	

6. What are some of the unit test frameworks/tools that you have used?

Name	Skill Level (Basic/Intermediate/Advance)
XCtest	Basic
Appium	-
Calabash	-
Others (Please Specify):	

7. What kind of database systems (all the versions) that you have worked with?

Name	Release/Version	Skill Level (Basic/Intermediate/Advance)
SQLite (Mobile)		-
Realm (Mobile)		-
MS SQL Server		-
Oracle RDBMS		Basic
MySQL		Basic
Others (Please specify):		

8. What are some of the API calling framework that you have used?

Name	Release/Version	Skill Level (Basic/Intermediate/Advance)
Alamofire		Intermediate
AFNetworking		Basic
Moya		Basic
Others (Please specify):		

9. What kind of continuous integration systems (all the versions) that you have worked with?

Name	Release/Version	Skill Level (Basic/Intermediate/Advance)
Jenkins		-
Fastlane		-
Bamboo		-
Others (Please specify):		-

10. Aside from native application development, do you have experience in the following development frameworks?

Name	Release/Version	Skill Level (Basic/Intermediate/Advance)
PhoneGap (Apache Cordova)		-
React Native		-
NativeScript		-
Ionic		-
Xamarin		-
Flutter		-
Android (Java/Kotlin)	Java	Basic
Others (Please specify):		

Hands-On Test – Create a Recipe App

Create a **Recipe App** with the following criteria: -

- Use Swift **(preferred)** or Objective-C as the programming language
- Create an xml file with recipe types data (recipetypes.xml), use that to populate the recipe types into a UIPickerView control
- Create a listing page to list out all recipes (filterable by recipe types from recipetypes.xml)
- Pre-populate your own sample recipes data complying with recipetypes.xml
- Create an Add Recipe page based on available recipe type with picture, ingredients and steps and update the existing list
- Create a Recipe Detail page that display the recipe's image along with the ingredients and steps. This page should include update (all displaying items should be editable) and delete feature
- Use at least one type of persistence method available in iOS to store data
- Upload the project into any public Git hosting services and ensure that your project is buildable

Your App should:

- Adhere to Apple's HCI principles for UI design
- Fit into any screen size and orientation while adhering to safe area
- Display recipe data in a properly formatted way
- Persist recipe data across app restart
- Able to perform normal operation without crashing

Please demonstrate the following:

- Adherence to Object Oriented Programming principles, and good programming practices
- Adherence to Swift/Objective-C programming naming and format convention.
- Proper use of the app lifecycle methods
- Use 3rd party libraries with CocoaPods to aid your development such as

* You are given 3 hours to build the app. Zip the iOS project files and send back to me via email along with the project hosting URL.

Bonus points if you can complete the additional requirements below:

- Reactive Programming: Exhibit Reactive Programming with RxSwift. Show how you properly achieve component/value binding with minimal usage of delegates.
- Architecture pattern: To build a clean and easy maintain code, by applying one of the architecture patterns like MVVM. Additional time given: 1.5 hour.
- Authentication: Login and Logout feature with authentication, encryption and session persistency until logout. Additional time given: 1 hour.
- Networking: Adding API layer to fetch data from online or self-hosted sources. Examples of implementation are authentication, self-hosted recipe type, etc. Additional time given: 1 hour.
- Dependency Injection: Implement dependency injection to modularity with Swinject library. Please show that you are able to achieve reusability, maintainability and scalability using dependencies architecture. Additional time given: 1.5 hour.

- Unit Test: Adding unit test and UI test to detect changes that can break the code design and reduce defects. Additional time given: 1.5 hours