DESIGN DETAILS

1. SERVER

To develop the server, I have used Perfect Server Side Swift. It's completely developed in swift. A RESTful Web Service has been developed to serve the data fetched from MySQL database.

2. IOS APP

It's iOS client has been completely developed in Swift also.

A. Tech and Dependencies used.

i. Autolayout, Size class: For UI and layout design
ii. ObjectMapper: For JSON parsing
iii. RESTful Web Services
iv. CocoaPods for dependency management.
v. Alamofire: For network layer
vi. Siesta framework: To consume Web Service
vii. XCTest: For Unit Testing
viii. Also there are some other minor plugins that have been used on this project.

3. ANDROID APP

Java 8 has been used as the main development language.

A. Tech and Dependencies used.

i. Activity and Fragments: For UI and layout design
ii. Gson: For JSON parsing
iii. RESTful Web Services
iv. Gradle for dependency management.
v. Retrofit2: For network layer
vi. Butter Knife: For binding
vii. Timber: For logging.
viii. Junit: For Unit Testing.
ix. Mockito: For Object mocking

x. Also, there are some other minor plugins that have been used on this project.

4. DESIGN PATTERN (FOR BOTH IOS AND ANDROID)

Following design patterns has been used for both iOS and Android.

A. Creational pattern

i. Singleton
ii. Abstract factory

B. Structural pattern

i. MVVM
ii. Decorator
iii. Adapter
iv. Façade
v. Bridge

C. Behavioral pattern

i. Observer
ii. Memento

5. CLASS DIAGRAM

A class diagram has been given below. Which is for Android App mainly. But it represents both iOS App and Android App. Almost 80-90% classes and its interactions/communications/design are identical for both iOS and Android. Naming conventions are also identical.

NB: Please enlarge this picture to see the details.

6. ARCHITECTURAL DESCRIPTION

Following you will find a short description on the app's architectural design. Both Android and iOS app retains same architectural pattern, class structure, communication flow, control flow and method signature. Both are almost 80-90% identical

A. Network Layer: "FamilyTreeAPIManager" class

As the name says, it's the API manager which manages all the network call, parsing the response and delivering it to the API consumer. Only this class talks to the network. It's a "Singleton" class.

B. Appearance

i. For iOS app: "AppearanceHelpers" group holds the classes, categories responsible for managing whole app wide appearance. And it applies the appearance on the AppDelegate.

ii. For Android app: Appearance values and keys has been managed into "res" folder as standard android resource management.

C. UI and Viewes

i. iOS: "Main.storyboard" file holds the main UI layout file. And "UI" source group contrails all of the sources and layouts responsible for UI.

1. *FTHomeViewController: Draws family tree*
2. *FTAddPersonTableViewController: Adds personal details and family members, relations.*
3. *FTPersonDetailsTableViewController: Shows personal details*

ii. Android: "activitys" and "holders" packages contains classes related with the view and UI. And XML layout files are into "res" folder.

1. *MainActivity: Draws family tree*
2. *AddDetailsActivity: Adds personal details and family members, relations.*
3. *ShowDetailsActivity: Shows personal details.*

iii. JSON Models: Following class contains JSON model classes on both iOS and Android App.

1. *NodeDetails*
2. *PersonDetails*
3. *Relations*
4. *RelationshipDetails*

iv. Utilities: Both "conf" and "Utils" folder/group contains App configuration and utilities files.

7. SEQUENCE DIAGRAM

Following is a sequence diagram for moth iOS and Android (individual part have been pointed out)