Design Rationale

FIT3077 Assignment 2
Team Name - Optimize Prime
Members - Sadeeptha Bandara

A number of design decisions as well as using design patterns have been made by our team for this project .
In order to consume the REST API, we use RetroFit which is an HTTP client used for Android Applications. Retrofit allowed us to consume the web service using Java interfaces. The HTTP client used has a huge impact on how the application is structured as it is required to obtain data from it.

The rationale behind using Retrofit was that we could access all the functionalities of the HTTP Client through dedicated interfaces for each endpoint, making the application extensible and decoupling the Application business logic from the HTTP client requests.

The design patterns we have used as per this include,

1. Singleton
   We access the API using a singleton instance of the Retrofit Client and a Utility class APIUtills. The rationale behind using the singleton pattern was that we could access the web service throughout the application with a single shared instance. This limited cluttering the codebase with repeated calls.

   Using the singleton pattern gave strict control over the retrofit client.

2. Facade pattern
   We use the Facade pattern when using Retrofit, by using a Utility class called APIUtils. This class provides a single point of access to the core functionality required to consume the web service.
   The class APIUtils provide access to each endpoint of each resource type through the interface,
   It helps in filtering out the specific functionality we need.

Since, we are developing an Android Application and have used RecyclerViews, in order to populate the RecyclerViews, we have used RecyclerView.Adapter classes. These classes implement the Adapter pattern in our codebase and are used for displaying lists in our application. This allowed to separate the logic of displaying the list from the business logic of the application.

In developing the application, we have used a number of model classes that are used for passing data to and from the web service. This includes classes for User, Qualification, Competency,Bid and such. These classes follow the Single Responsibility Principle and contain data relevant to its purpose only. Further, related functionality is abstracted and modularized and use. A good example of this is the use of the BidInfoForm class, which is used to abstract the commonalities of the various form pages that we use in our application.

These are some of the main notable design decisions and the rationale behind them.