

OrderIt

CSL343 GROUP 2 ANDROID PROJECT REPORT



Indian Institute of Technology Ropar

Table of Contents

Team Members	2
Overview	3
Auction Model	4
Database Schema.....	6
Server Scripts	7
Profile Module.....	7
Auction Module.....	7
Bidding Module	8
Profile Module	9
Login Screen	9
Home Screen	10
Navigation Drawer Fragments	11
1. Home Fragment	11
2. Profile Fragment.....	11
3. Feedbacks Fragment	12
4. Rate and Review Fragment	13
5. User Requests Fragment	14
6. Logout option	14
Broadcast Requests.....	15
Auction Module	16
Bidding Module.....	19
Post auction process	25
Networking components	27
External Libraries	27
Project Contributions.....	28
Future plans	30

Team Members

Name	Entry Number
Gaurav Kushwaha	2012CSB1012
Gaurav Mittal	2012CSB1013
Gurparteek Singh	2012CSB1014
Jaideep Singh	2012CSB1015
Jeevanjot Singh	2012CSB1016
Khan Uzair Suhail	2012CSB1017
Kunal Yadav	2012CSB1018
Maninderjit Singh	2012CSB1019
Naveen Kumar	2011CS1020
Mohit Garg	2012CSB1020
Paras Ahuja	2012EEB1068

Overview

The project is an Android application which provides a portal for the users to notify others about their upcoming excursions and start auctions, specifying the details such as date and time of their excursion, where they can accept orders from fellow customers who participate in these auctions by specifying their bids in the form of the extra fee they are willing to pay for their orders.

At the end of the auction, the service provider can select the certain number of top bids it is willing to cater to following which both the service provider and the customer whose order in the form of bid is confirmed are shown their identity using which they can contact each other for the fulfillment of the order at the extra fee decided by the system.

The app provides the added functionality to rate the users and provide feedback along with the facility to the user to view its own feedback that will allow future participants to make better decisions and also help in moderating the usage of the app.

Auction Model

The essence of the entire project is the complex Auction model which powers the Android app. The major challenge faced in the designing the model is to incorporate the effort required by the Service Provider. The effort can vary over a wide range considering the fact that the type of services offered or the different types of orders being catered to can be over an extremely huge spectrum and it won't be possible to enumerate each and every case simply in the app. Instead the Auction model is based on various heuristics and works fairly well in majority of test cases.

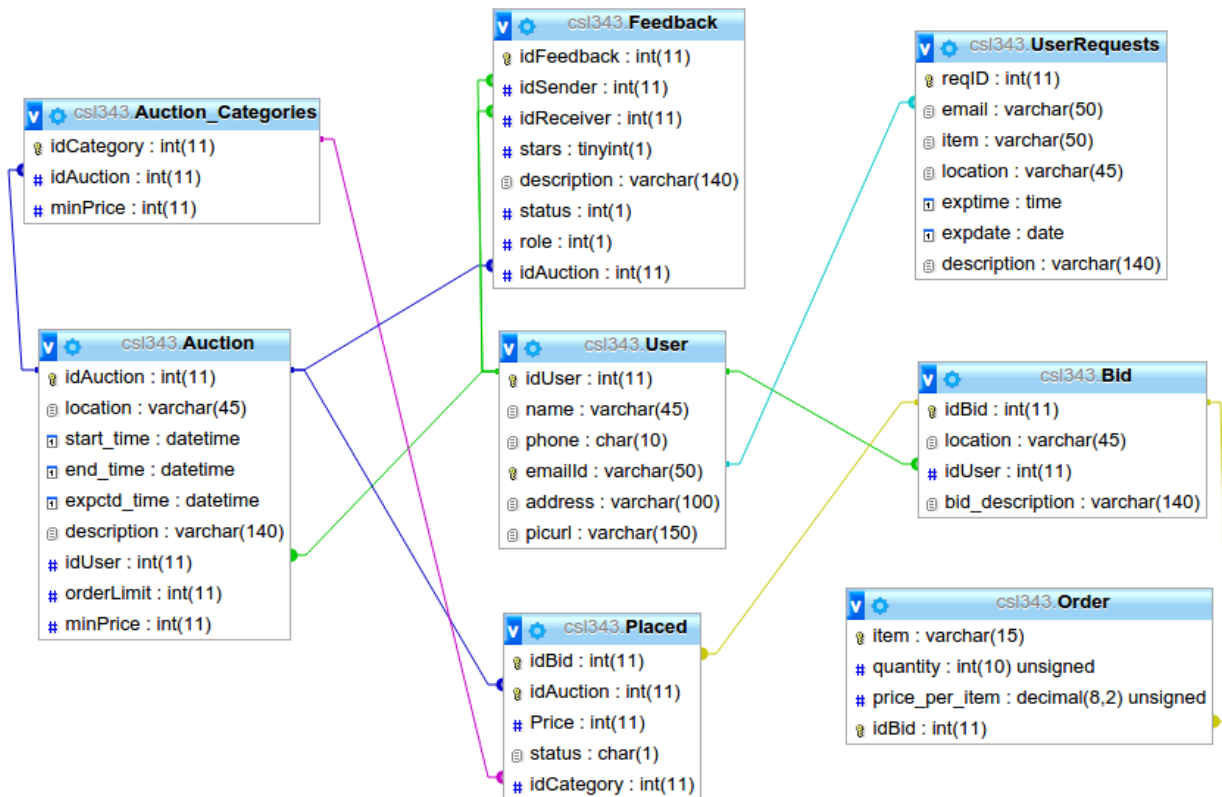
Following are the salient features of the Auction model that have been duly implemented and tested within the Android app:

- The service provider starts an auction stating a minimum fee it is going to charge on every bid. It also states the tentative order limit it is going to adhere to. However, as mentioned this order limit is tentative and the service provider can select any number of top n bids at the end of the auction.
- When a client wants to enter an auction, it clearly states what service it is asking for and sends a request to the service provider.
- The service provider accepts the request if it is found feasible and now the client enters the auction with the bid starting from the minimum fee.
- Once the service provider accepts to do the job at some minimum fee, the provider can't reject it during the auction or afterwards if that job wins. Otherwise it will reflect badly in the feedback by the customer. Also the provider can't accept any bid prior to the closing of the auction.
- Suppose the provider finds that the job is doable but the minimum fee being charged is quite less as compared to the effort required. So, the provider is given an option to place that job in a higher category. With that, it states a higher minimum fee (which can't be changed any time later). However, the option to create a new category is given once, so that the provider doesn't exploit it and some competition can be introduced.
- If the provider doesn't find the job doable at all, it can reject that straightaway.
- This way, all the incoming bids will be divided into at most two categories sorted according to the effort required and ranked according to the fee and now the auction can take place in a regular manner. If the provider tries to skew the bids in one or the other category, it might end up doing a job requiring a greater effort for a low fee, since

the provider is obliged to cater the top n jobs (not doing so will reflect poorly on the feedback)

- On the customer side, on being accepted, they will be shown their ranks in their respective category of auction, minimum bid that can be placed, their own current bid and the current number of jobs the provider willing to do. Accordingly, the customers can raise their bids to increase their chances of winning.
- The customer has the option to back out of the auction at any point during the auction but that will again reflect on the feedback by the provider.
- The top n number of jobs provider is willing to do, have to be kept adjustable, since even after categorizing the jobs according to the effort, it is possible that the total effort for the top bids is higher than the provider is willing to put. Also, the provider will only be shown the bids according to their ranks with only the description of the job associated with the bid visible. It is suspected that showing the actual fees for the bids can be misused by the service provider.

Database Schema



Server Scripts

Our database server is hosted at <http://netapp.byethost33.com/>

Below are the PHP scripts running on our server.

config.php - Specifies the various parameter needed to connect to the database server.

Profile Module

add_user.php - Adds a new user (email-ID, name, Google+ image link) in the database.

add_phone.php - Updates phone number of the user in the database after successful phone verification.

add_address.php - Updates the address of the user in the database.

get_phone.php - Fetches phone number of the user from the database.

get_address.php - Fetches address of the user from the database.

add_broadcast.php - Adds a new Broadcast Request in the database.

get_broadcast.php - Fetches Broadcast Request details from the database. Used for searching the requests as well.

service_rate.php - Fetches details from the database corresponding to pending feedbacks of both the customer and service provider. (Pending feedback details correspond to the auction details)

add_rate.php - Adds the rating and review corresponding to a pending feedback in the database. Used for both Service reviews and Customer reviews.

get_feedback.php – Retrieves feedback information (rating and reviews) from database where user was service provider or customer.

Auction Module

accept_bid.php - Accepts the bid request of a customers by updating the status of the bid in the 'Placed' table to 'A' and setting the idCategory to which the bid has been accepted.

auction.php - Allows to insert a new Auction into the 'Auction' table along with a default Auction Category with the minimum price specified by the Service.

confirm_bids.php - Confirms a bid for an Auction that has ended by changing the status of that bid to 'C'.

create_accept_bid.php - Accepts the bid request into a new category created by the Service Provider.

getAuction.php - Sends the details of the current Auction running for the specified user, its current status and all the bids associated with that auction.

reject_bid.php - Rejects the bid request from the Auction of the specified user by deleting the associated tuple from the database.

Bidding Module

addBid.php - File to add a new bid in auction.

getAllAuctions.php - File to get details of all running auctions, participating, not participating and bids placed in them.

getRank.php - Get Rank of a particular bid in an auction having same Category.

getRating.php - Get Rating of service providers.

placeBid.php - Place an already running bid into a new auction.

updatePrice.php - Update price of a bid.

Profile Module

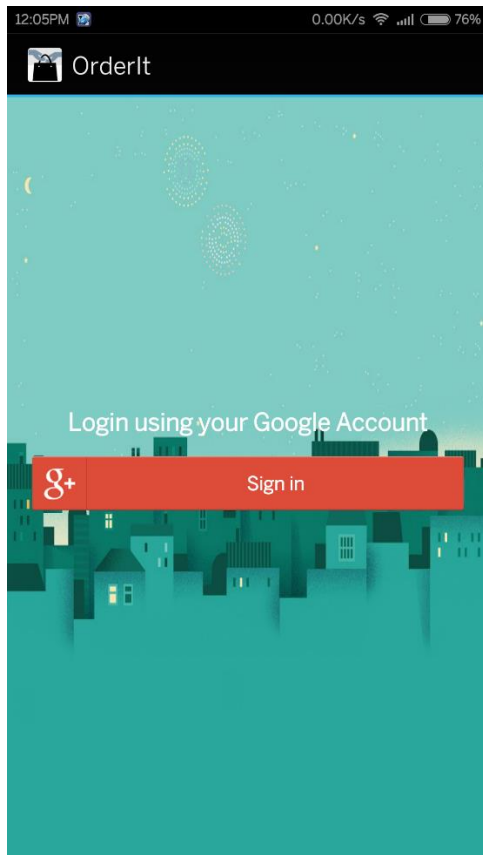
The profile module of the app consists of the Google plus login/logout functionality and all the sections under the User navigation drawer (slider that opens on swiping right at the home screen).

Login Screen

On opening the app for the first time, the user is presented with the Login screen that has “Sign in with Google Plus” option. On clicking the Google Sign-In button, the user is presented with the Google accounts, using one of which the user can log in.

Once the user logs in his email-id, name and Google Plus account profile-image are collected, and the user is added to our database. (See [add_user.php](#) for further details). All these details are stored in the SharedPreferences of the app, and used wherever required. As the details are stored locally, the user doesn’t need to log in again. So, the next time the user opens the app he doesn’t encounter the Login Screen, and taken directly to the Home screen. If he logs-out, then all the details are cleared, and the user will need to log in again on opening the app.

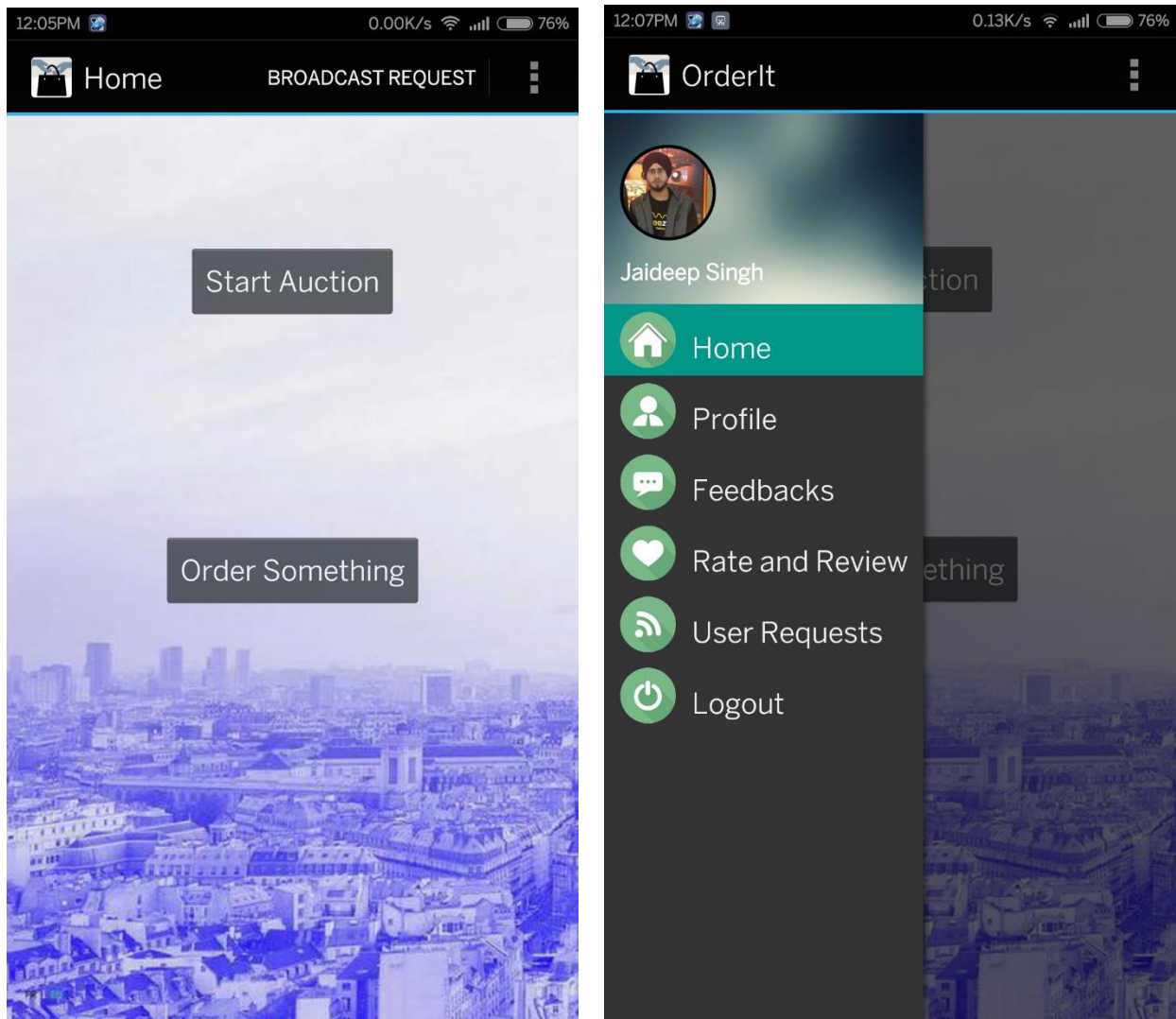
For this functionality, we have used Google OAuth 2.0 Authorization to access the Google + API.



Home Screen

After successful log-in the user is taken to the Home Screen, where there are 4 choices that the user can make. On the home screen the user can see 2 options, “Start Auction” (see [Auction Module](#)) and “Order Something” (see [Bidding Module](#)).

In the action bar, there is an option to “Broadcast Request” (see [Broadcast Requests](#)). And on swiping left to right on the Home Screen, the user can see the Navigation Drawer, which contains different fragments which are listed below:



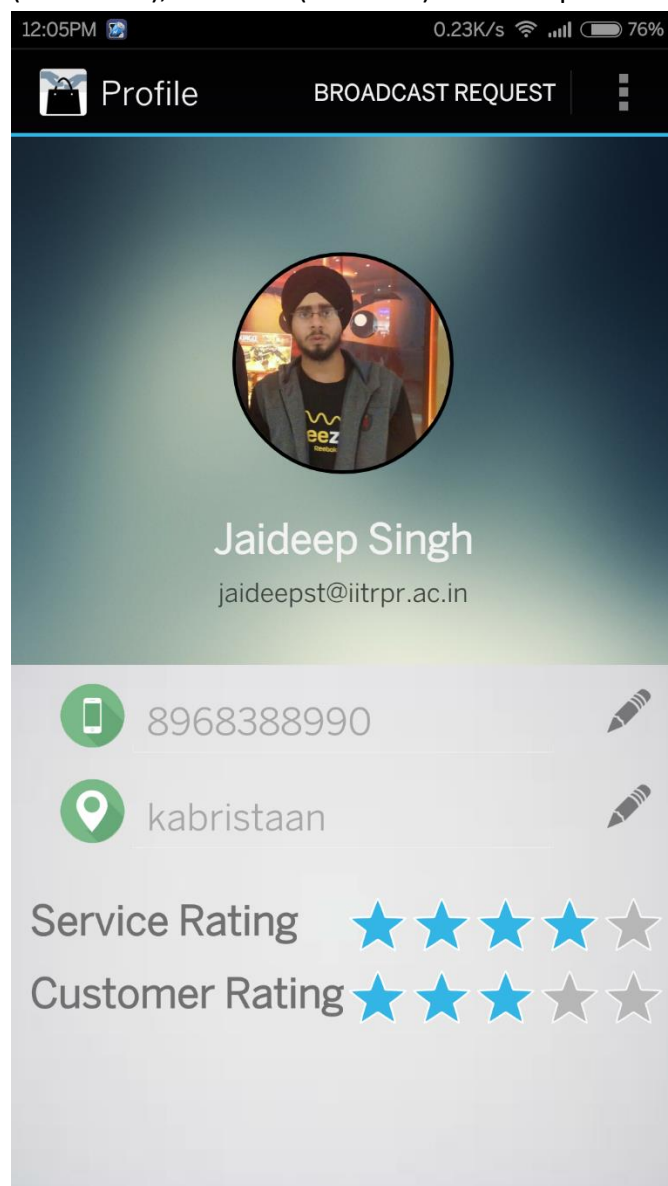
Navigation Drawer Fragments

1. Home Fragment

On clicking the “Home” option in the Navigation Drawer the default [Home Screen](#) is shown to the user. So, while navigating different sections of the Navigation Drawer, the user can anytime go back to the Home Screen on selecting this option.

2. Profile Fragment

On selecting “Profile” the user can see his/her profile details like name, email-id, phone number (if verified), address (if edited) and his profile image. And below the user can see his/her average Service and Customer Rating.

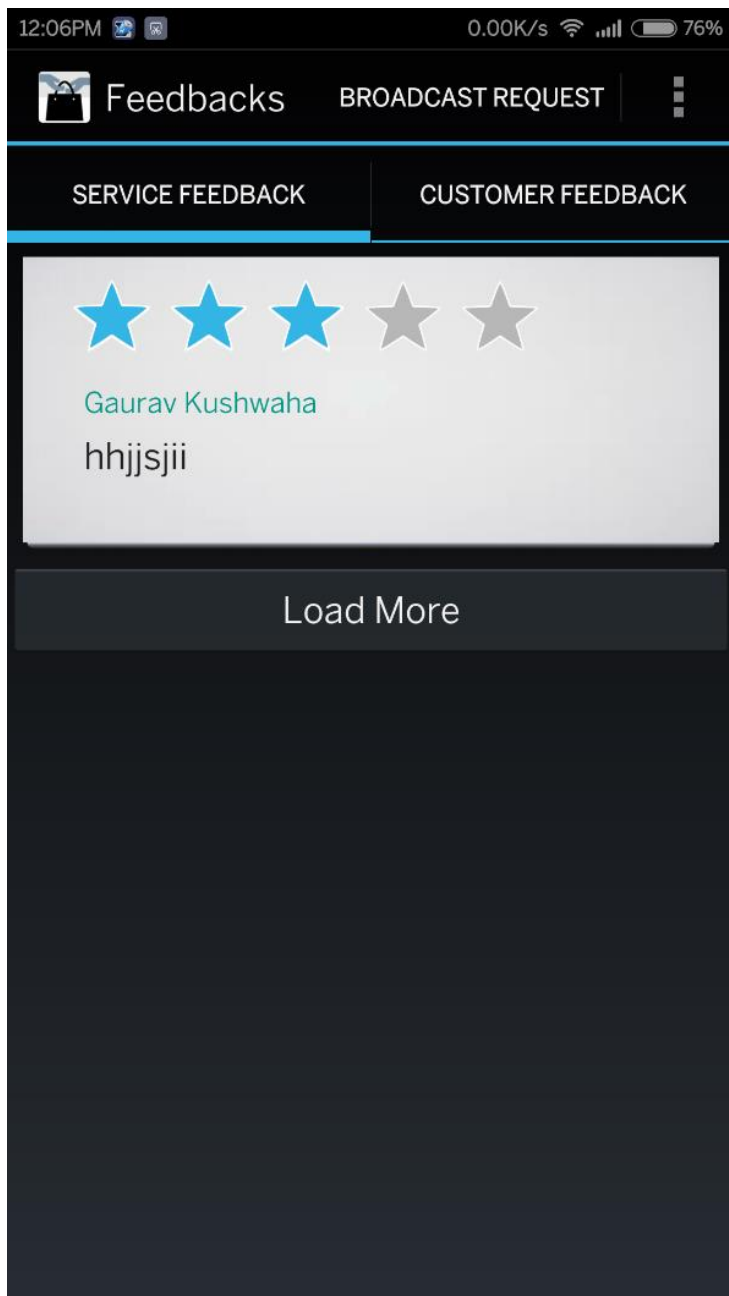


Next to phone number and address fields we have edit options. On clicking edit address icon, the user gets an input dialog, to enter the new address. On submitting the updated address, a post request is made to [add_address.php](#) on the server, which updates the user address in the database. The address is also updated in the default SharedPreferences of the app.

On clicking edit against phone number, an input dialog opens to enter the 10-digit phone number. On submitting the number, its verification is done as follows:

- The inputted number is sent to Cognalys API's servers
- If the user gets a missed call from the Cognalys server within 60 seconds, the number gets verified. Then the number is updated in the user's profile in our database using [add_phone.php](#) and also saved locally in the SharedPreferences.
- If the user doesn't get a call within 60 seconds, means the number that he had submitted was not of his/her device, so the verification fails.

3. Feedbacks Fragment



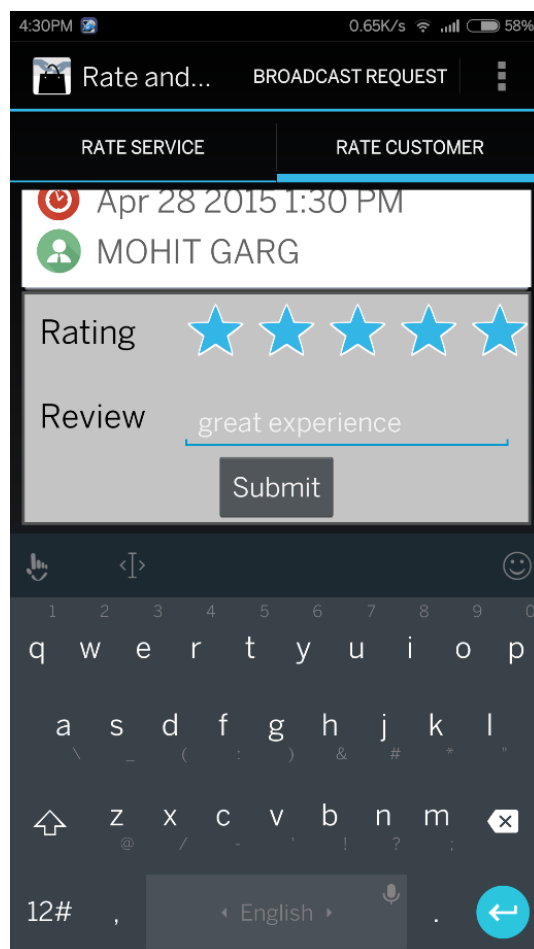
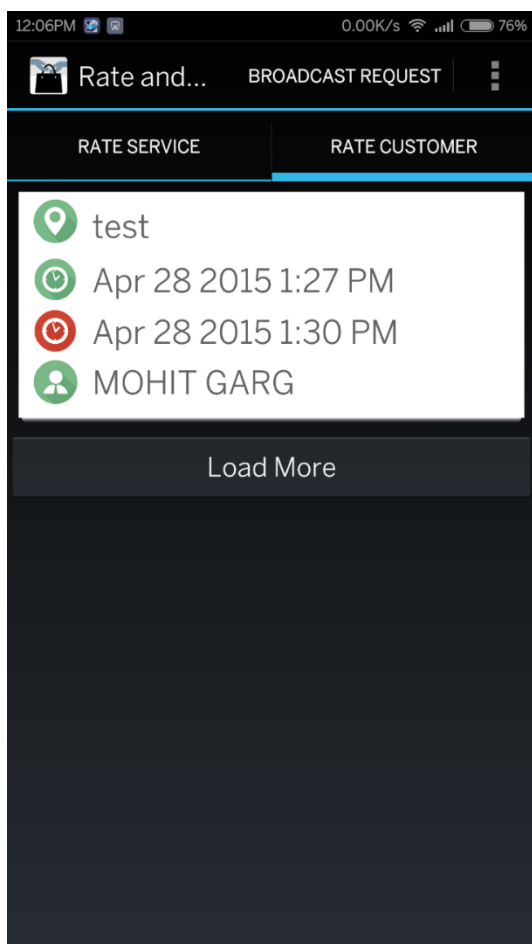
In the Feedbacks section, the user can see his/her rating and reviews both as a Customer, and as a Service Provider. There are 2 swipe able tabs in this fragment, one is the “Service Feedback” where the user can see feedbacks that he got from customers, for whom he had brought orders. And in the “Customer Feedback” tab the user can see the feedbacks from Service Providers that he/she got as a customer.

Initially 10 feedbacks are loaded from the server (using [get feedback.php](#)) in each tab. To load more feedbacks, there is “Load More” button at the bottom, which fetches the next 10 feedbacks.

4. Rate and Review Fragment

In the “Rate and Review” section the user can see the pending feedbacks that he/she has to rate and review. On every accepted bid after an auction ends and the expected delivery time is over, the Service Provider gets a pending feedback to rate the customer corresponding to that bid. And the Customer gets a pending feedback to rate and review the Service Provider of that auction.

So there are 2 swipe able tabs, “Rate Service” to rate the Service Providers and “Rate Customer” to rate the customers. Each of the tabs contains list of pending feedbacks which correspond to details of an auction. Then there is the “Load More” button similar to the “Feedbacks” fragment.



To rate a customer or a service provider, the user must click on the auction details card in the list, which expands the card showing rating bar and review text field. On clicking submit, the feedback is added to the database (using [add_rate.php](#)). The submitted feedback is shown to the other person in his/her feedbacks fragment. He/she also notified with a push notification.

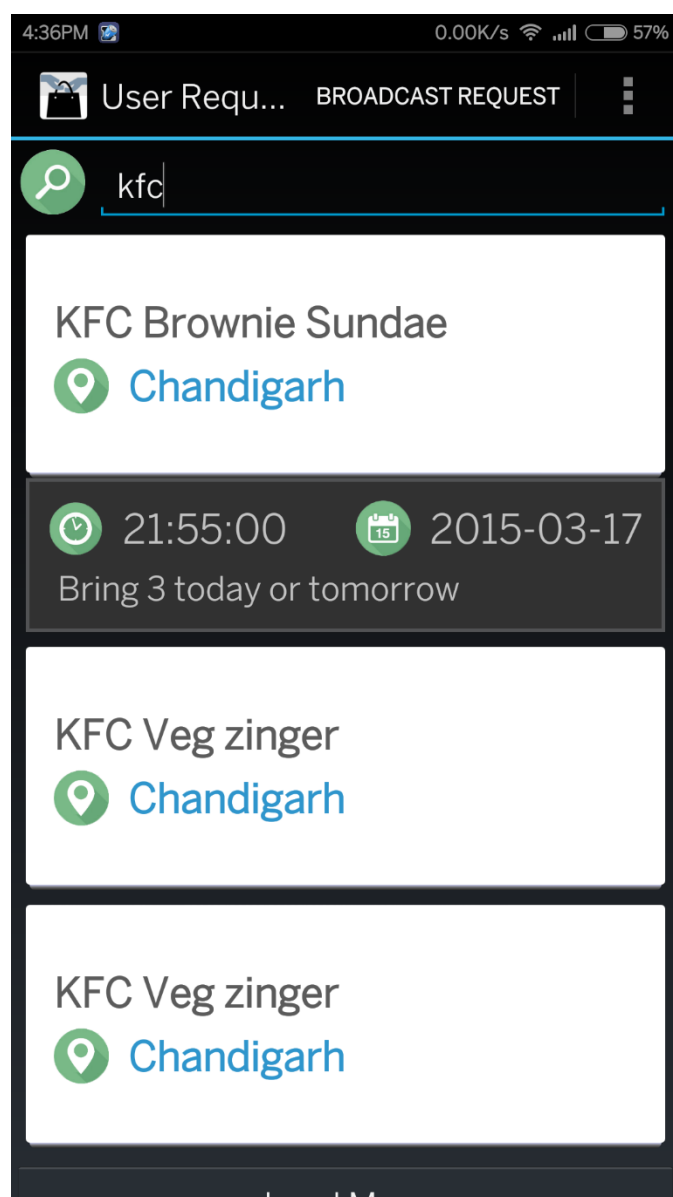
5. User Requests Fragment

In the “User Requests” section, the user can see the Broadcast requests made by other users using the “[Broadcast Request](#)” feature in the action bar. The item requests made through broadcast request feature are sent to all the users of the app using Push Notifications. Apart from those notifications, the requests appear in this “User Requests” section of the Navigation Drawer.

The user requests are shown as a list of expandable and swipe able cards (used CardsLib library for card layouts). Item name and Location are shown in the cards, and when a particular card is

clicked, it expands to show further details like the description, expected date and time.

On the top there is Search bar with Search-as-you-Type functionality. It updates the list with user requests as soon as the search text is entered. On erasing the search text, the 10 default recent requests are shown. Then at the bottom, there is a “Load More” button which fetches next 10 recent requests from server (using [get broadcast.php](#)).

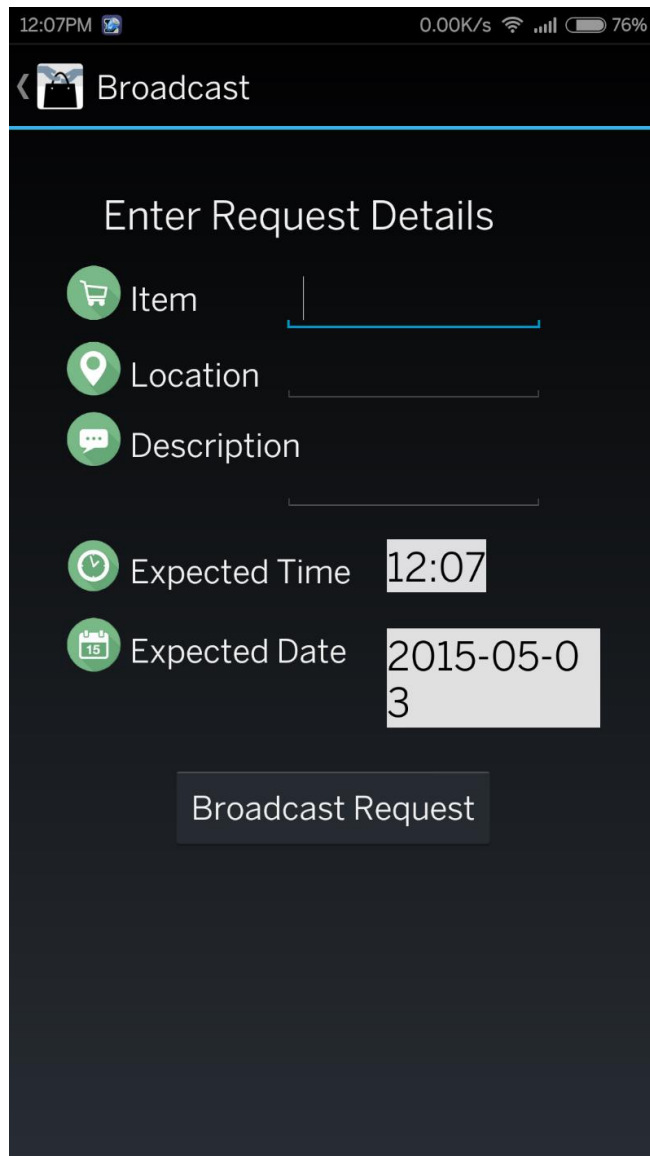


6. Logout option

On clicking “Logout” option, all the user details stored locally (name, email, Profile pic, phone number, address, UserId) are cleared and the Google OAuth 2.0 access token is revoked. The user is taken to the [Login Screen](#). As the access token is revoked, the user will need to log in again using his/her Google account from the Login Screen.

Broadcast Requests

Broadcast Request activity opens from the action bar. This activity is used for sending Push notifications to all the users of the app, regarding an item which the user wants. The activity is shown below:



The screenshot shows an Android application interface for creating a broadcast request. At the top, the status bar displays the time 12:07PM, network speed 0.00K/s, and battery level 76%. Below the status bar is a dark blue header with a back arrow and a shopping bag icon, followed by the title "Broadcast". The main content area has a dark blue background and is titled "Enter Request Details" in white. It contains five input fields, each with a green circular icon: "Item" (shopping bag icon), "Location" (location pin icon), "Description" (speech bubble icon), "Expected Time" (clock icon), and "Expected Date" (calendar icon). The "Expected Time" field is set to "12:07" and the "Expected Date" field is set to "2015-05-03". At the bottom, there is a dark blue button with the text "Broadcast Request" in white.

In the form, the user fill details such as item name, location from which the item needs to be brought, necessary description and expected date and time. Time and date are inputted using the default time-picker, date-picker of android. On submitting the request, all the users of the app get push notifications regarding the request. And the users can also see and search all the requests in the "[User Requests](#)" fragment.

Auction Module

This section of the Android app deals with the functionality to start Auction, specify its various parameter like date, time, end time and minimum price, and then, proceed to accepting bids for orders from various users followed by confirming certain number of top bids after the end of the app, resulting in the service provider to be shown the profile of the customers whom it can contact to and finally cater to their respective orders.

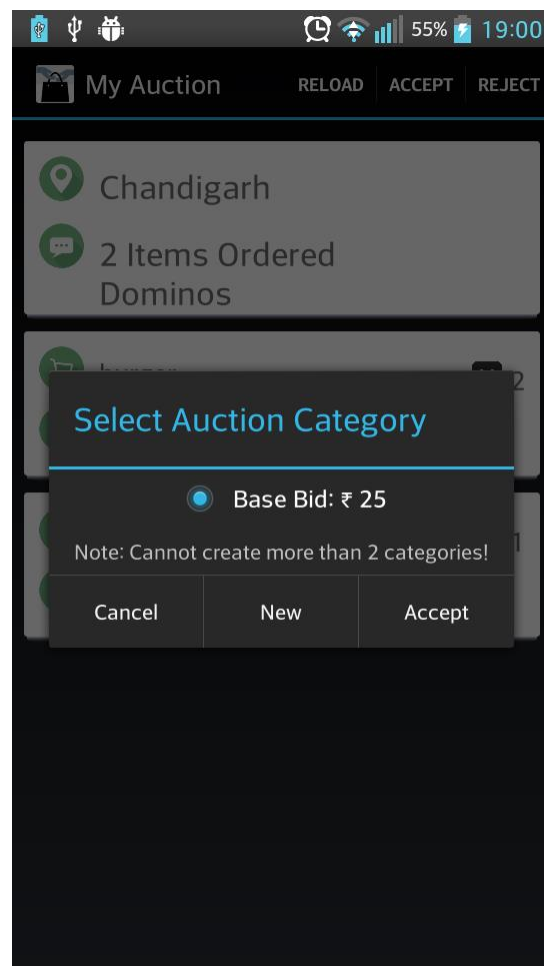
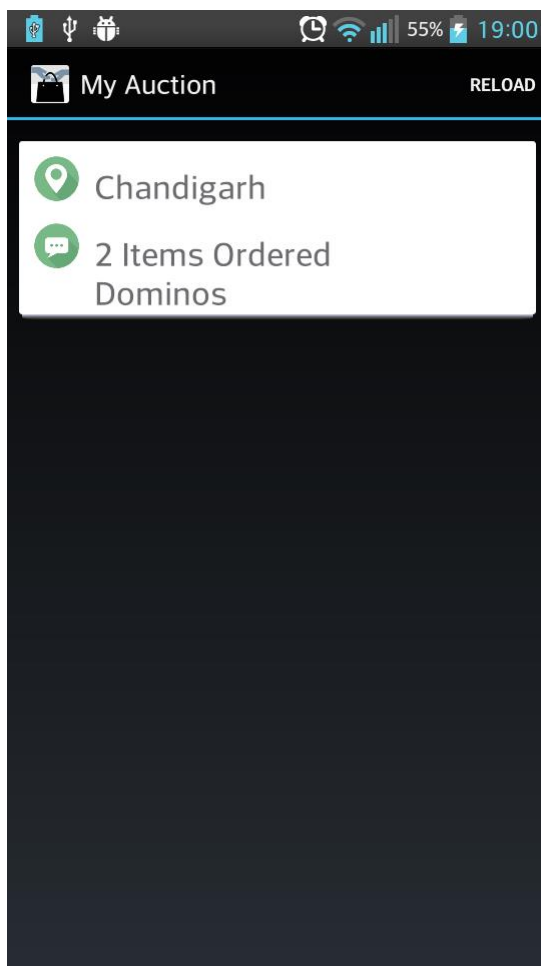
The following describes the major elements involved in the workflow of the Auction module:

1. When the user presses the 'Auction' button on the homepage of the app, it is taken to either a form or the auction dashboard depending upon whether the Auction has expired.

2. The Auction form is shown to the user when the Auction has expired, i.e., the expected time specified in any previously created Auction has been reached. This clearly suggests that a user can only create one Auction at a time.
3. The form contains various details to be filled like the location the user is visiting, an optional description to convey, the end date and time of the auction, the expected time

of completion of the service demanded, the minimum bid accepted by the auction and the tentative number of the orders that will be served. On successful filling up of these details based on various constraints on the ending and expected times, the auction can be started by the user.

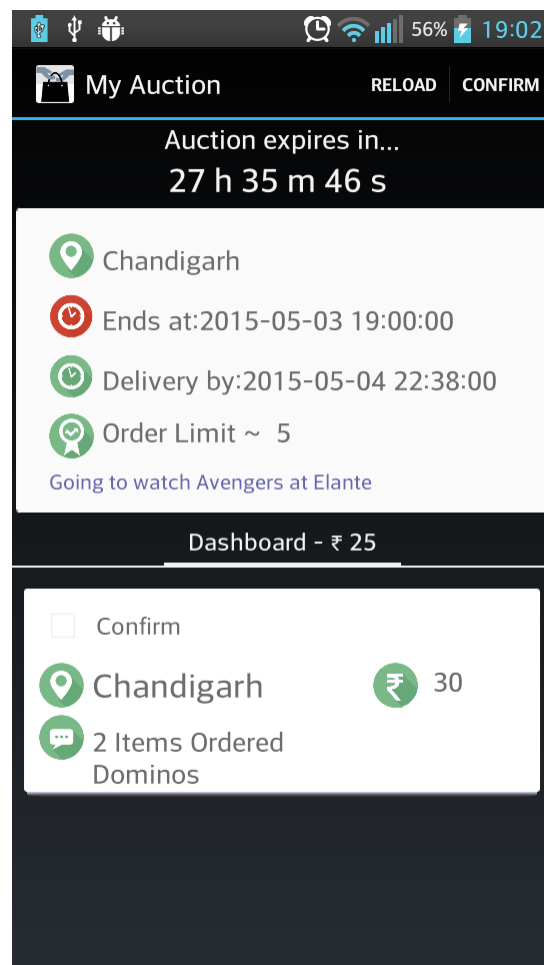
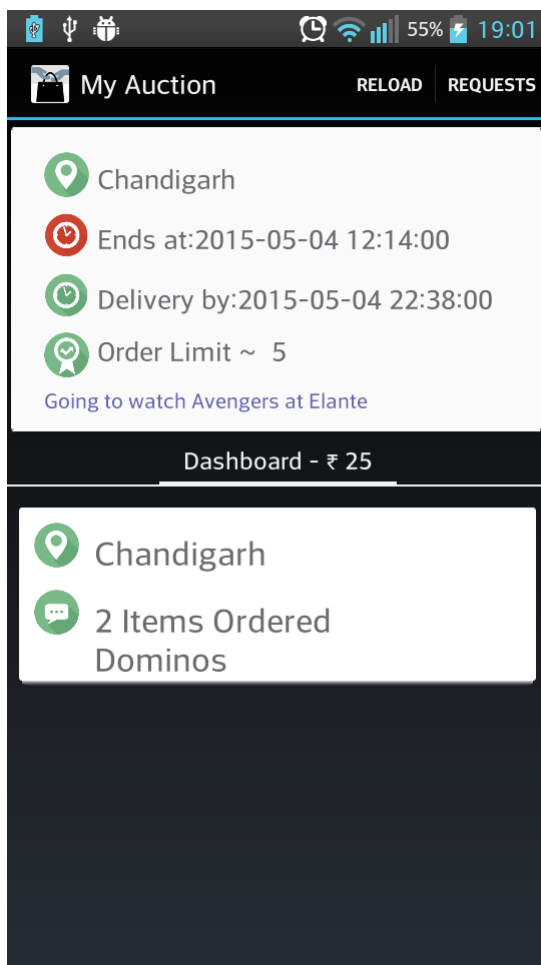
4. Once an auction is started or in case an auction was previously started (whose expected time is still sometime in future), the user is shown the Auction Dashboard. This is the place where the user can see all the bids associated with the auction.
5. The Auction Dashboard categorizes the bids into different tabs based on the starting minimum bids for each category. The user can click on any bid to view the details related to location and the order corresponding to that bid. New categories are added and are shown as new tabs, once they are created by the user (now the Service Provider) on acceptance of a bid request.



6. In the Action Bar of the Auction Dashboard, the user can see a Requests button. On clicking that button the user will be shown the list of pending bid requests by customers. Again by clicking on any bid request, the user can view its details pertaining to the order and location. Now, if the user finds the order or service requested by the bid to be doable, it can press the Accept button on the Action Bar to allow the bid in the Auction,

otherwise it can reject it straightaway. The user can take cues from the ratings and feedbacks of the customer associated with the bid to decide whether to accept the bid request or not.

7. Also, on accepting the bid, the user is shown the option to either accept the bid in anyone of the already present categories or if the user finds the bid requiring an effort very different from the rest of the bids accepted so far, it can create a new category specifying the minimum bid for the category. However, in order to make sure the user doesn't exploit this option, the current number of categories in any auction is limited to two. After the bid is accepted, it will be displayed in one of the tabs in the Auction Dashboard.



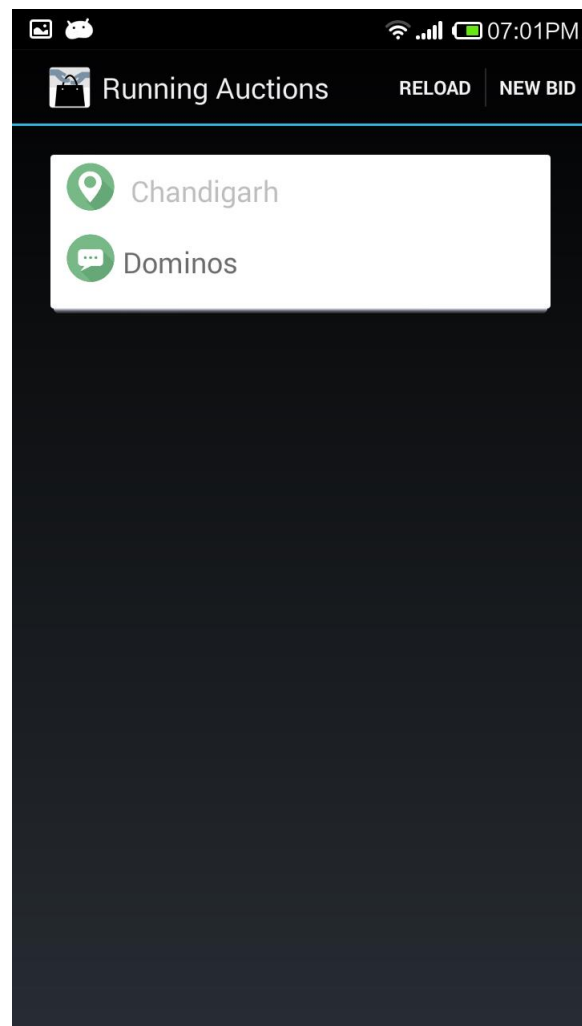
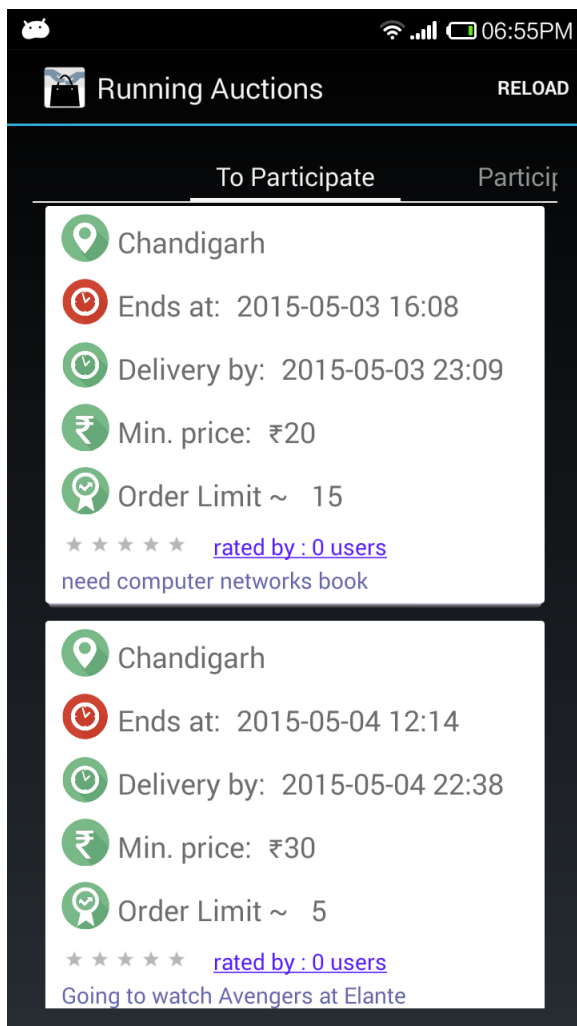
8. Until the Auction has ended, i.e., its end time has reached, the only function the user needs to perform is to accept the appropriate bids into the Auction.

Bidding Module

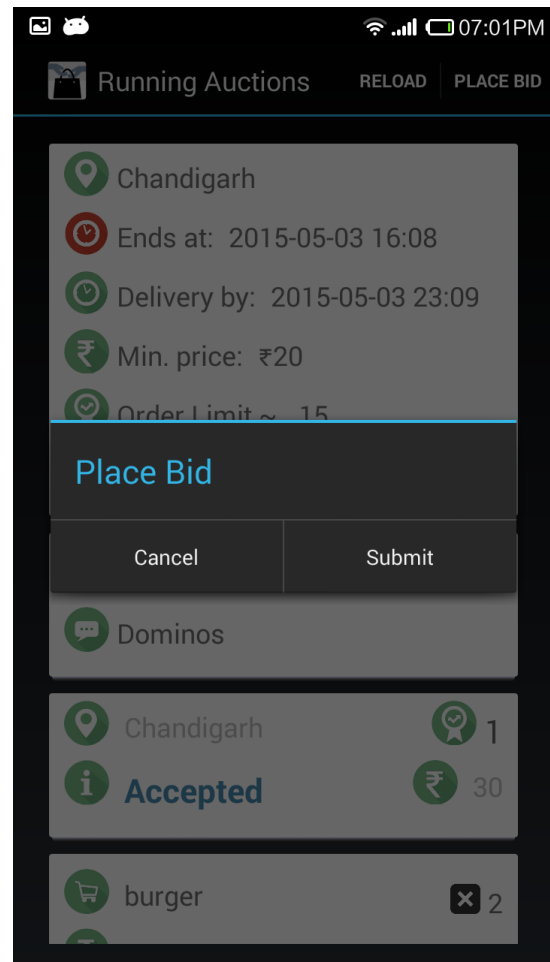
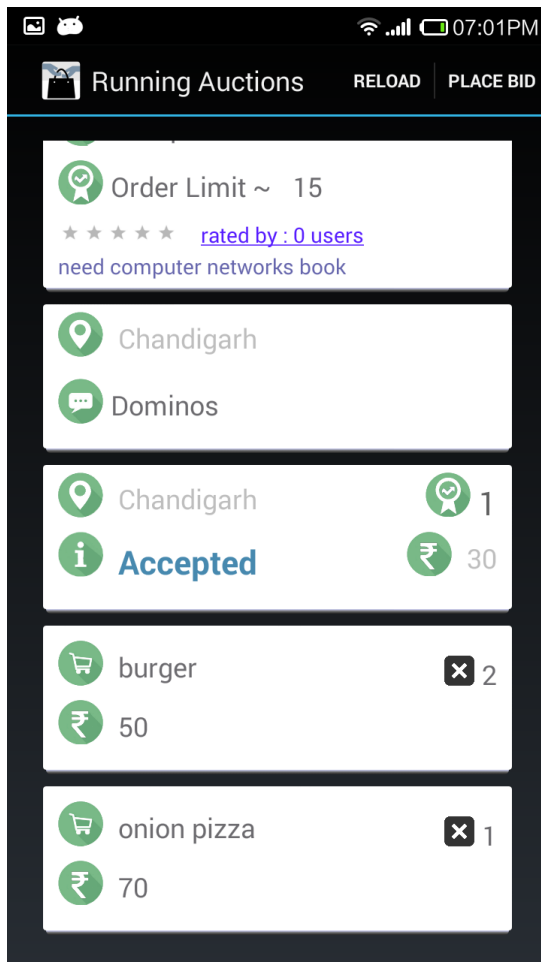
This section of the Android app deals with the functionality to place bid, specify its various parameter like place, order items, price and quantity. After placing the bid, once the bid gets accepted its price can be increase so its position in the ranking table is increased.

The following describes the major elements involved in the workflow of the Bidding module:

1. When the user presses the 'Order' Button, he is taken to a tab view, which contains 2 tabs.
2. First tabs consists of the auctions in which he is not taking part currently and he can place his/her bid to order.



- On clicking an auction in the first tab, the user moves to a new screen in which user is shown his currently running bids, so the user can select to place an already running bid to increase its chances of getting accepted in at least one auction, or the user can place a new bid.



- To place already running bid, user selects the bid the user wants to place. After selecting a new place will open up, consisting of all the details, Current Auction in which he will be placing the bid, bid details like location, description, auction details in which it is already placed, ranks and prices in that auction and finally order items in that bid. User can place this bid by selecting 'Place' button from the action bar. He will be asked to confirm and then it will be placed.

The image displays two side-by-side screenshots of a mobile application interface titled "Running Auctions". The interface is dark-themed with white text and green icons.

Left Screenshot (06:55PM):

- Header:** "Running Auctions" with a "RELOAD" button.
- Section:** "Enter Bid Details".
- Form Fields:**
 - Location:** Input field with a location pin icon.
 - Description:** Input field with a speech bubble icon.
 - Add orders:** Section with a shopping cart icon.
- Table:**

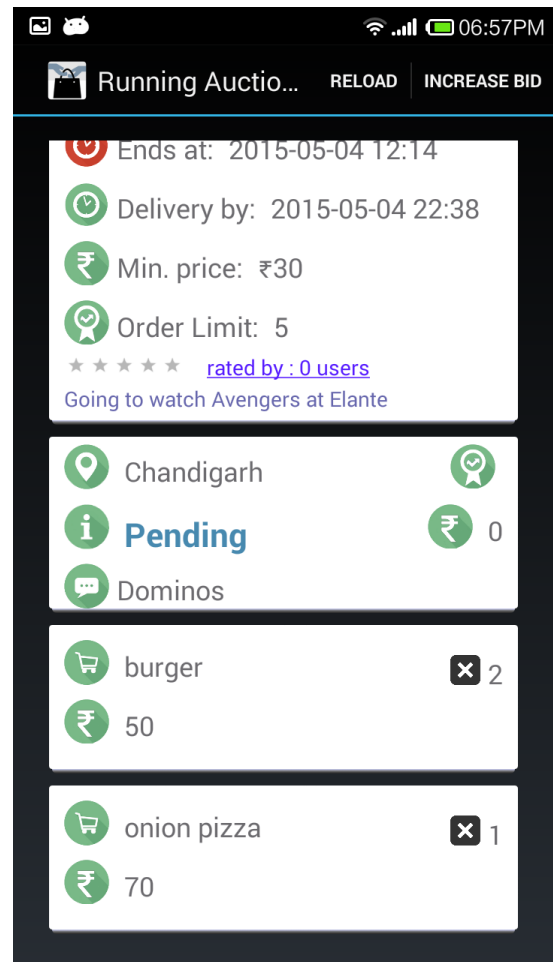
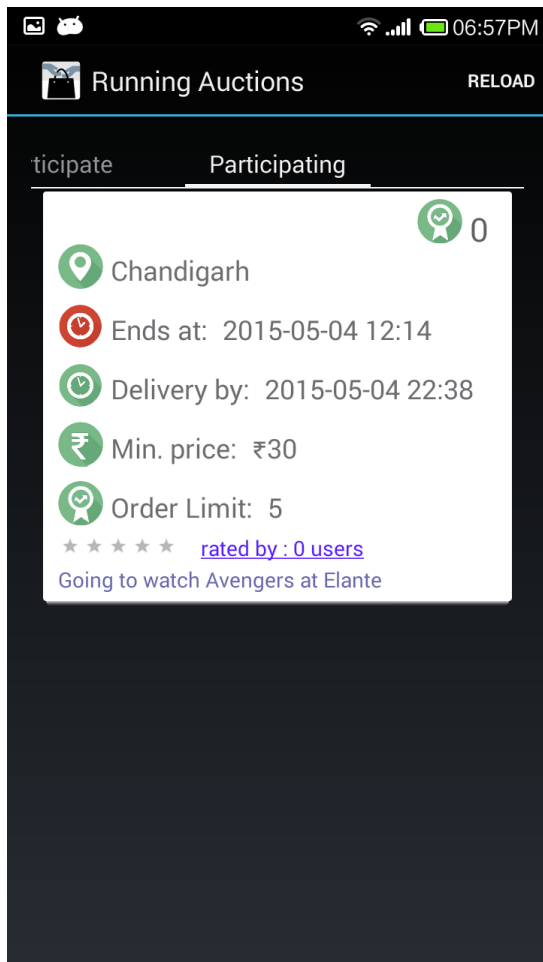
Item	Quantity
Price per item	
- Buttons:** "Add item" and "Place Bid".

Right Screenshot (06:56PM):

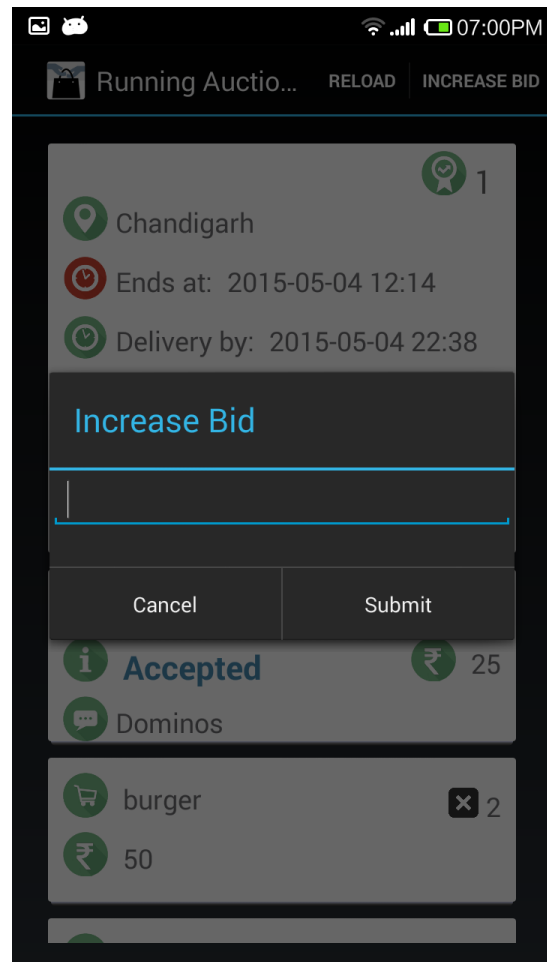
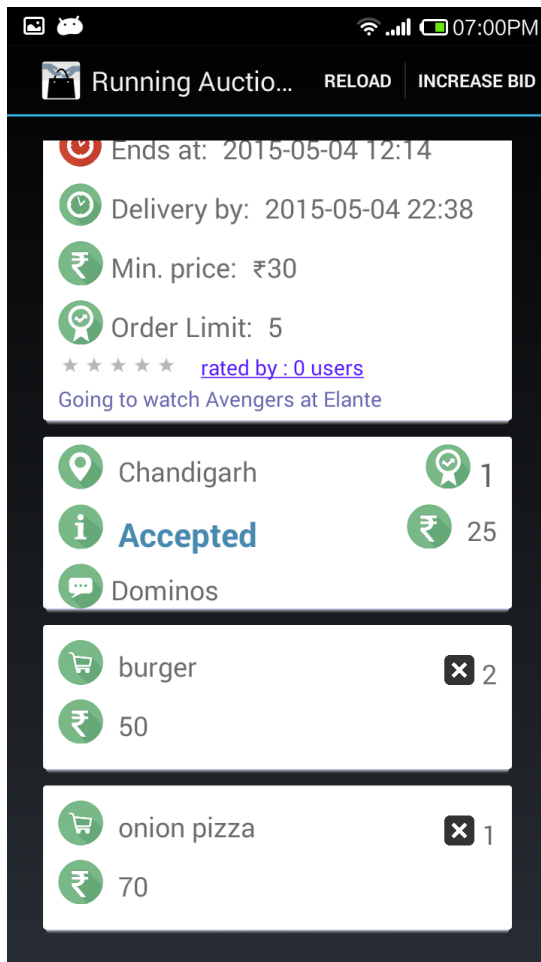
- Header:** "Running Auctions" with a "RELOAD" button.
- Section:** "Enter Bid Details".
- Form Fields:**
 - Location:** "Ghandigarh".
 - Description:** "Dominos".
 - Add orders:** Section with a shopping cart icon.
- Table:**

onion pizza	1
70	
burger	2
50	
- Buttons:** "Place Bid".

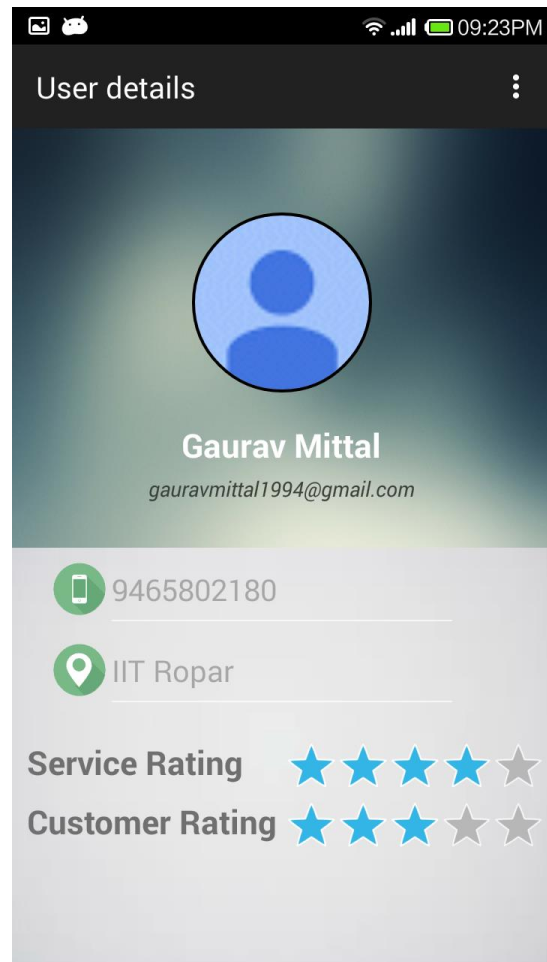
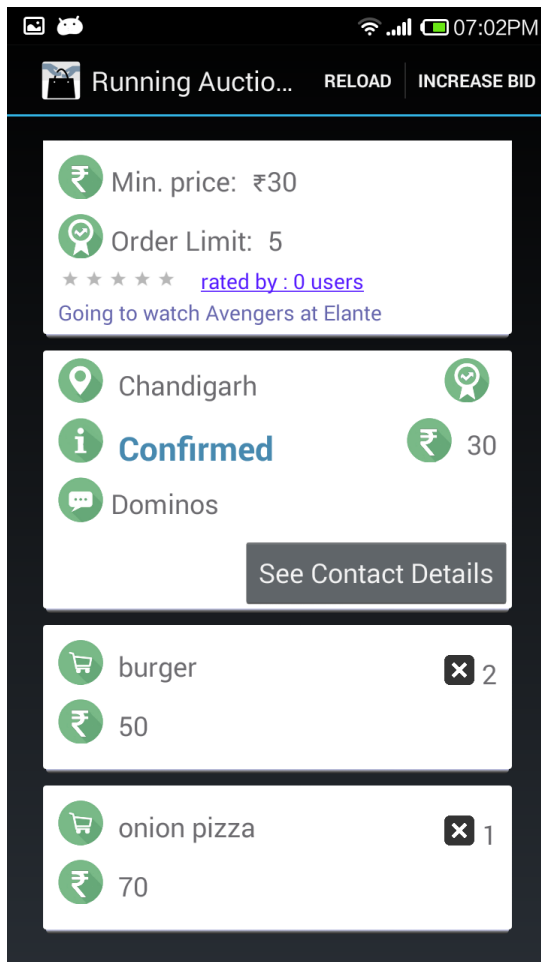
5. To place a new bid, user can select a button from action bar to place a 'New Bid'. After Clicking the user will be taken to a new page consisting of a form to fill all the bid details. After filling user can submit the form.



6. Second tab consists of the auctions in which he/she has already taken part and can increase the bid amount to increase the chances of getting select.
7. On clicking an action in the second tab, the user moves to a new screen consisting of details, Auction details like location, end time, delivery time, bid details like location, status, price and rank and then the order of the bid.



8. In this view the user can increase his/her bid amount if he/she wants to by clicking the 'Increase Bid' button in the action bar. Bid amount can be increased of only bids having accepted status. Then a popup will be shown asking for the new bid price. The user must increase the bid. After entering the new price, its new price is synced and its rank is correspondingly changed.



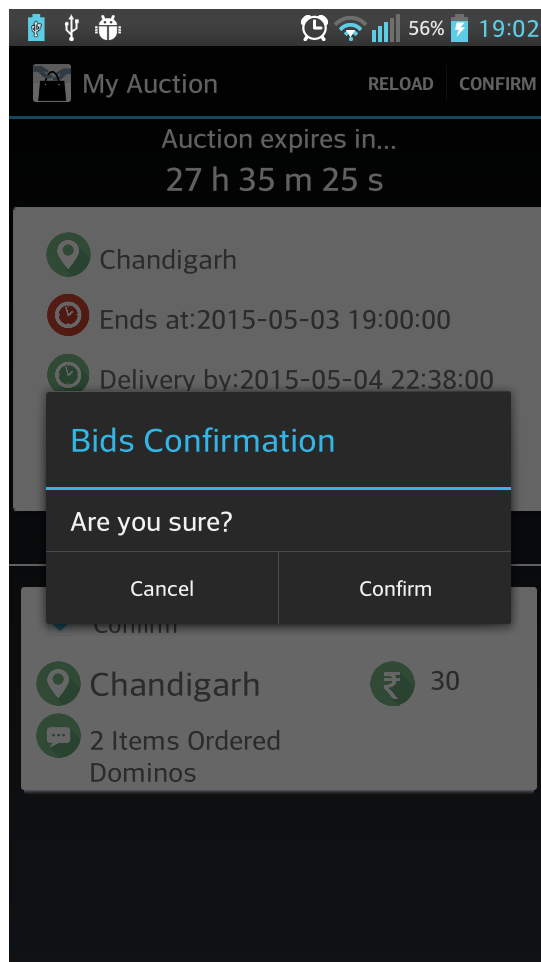
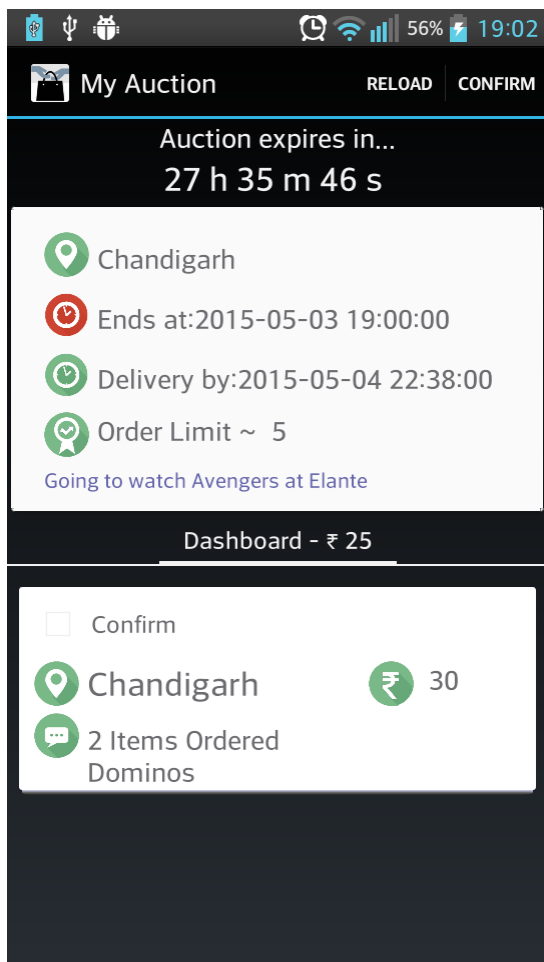
9. After the Bid of the user is confirmed he/she is shown the option to see the user (who started the auction) profile.

Post auction process

Once the end time of the Auction is reached, the Auction comes to an end. After the Auction has ended, no further bid requests can be entertained by the Service Provider and no further changes can be made to the amount of bid by the fellow customers participating in the Auction. The Auction completely expires once the expected delivery time of the Auction is reached, which is a minimum of 6 hours and a maximum of 1 day.

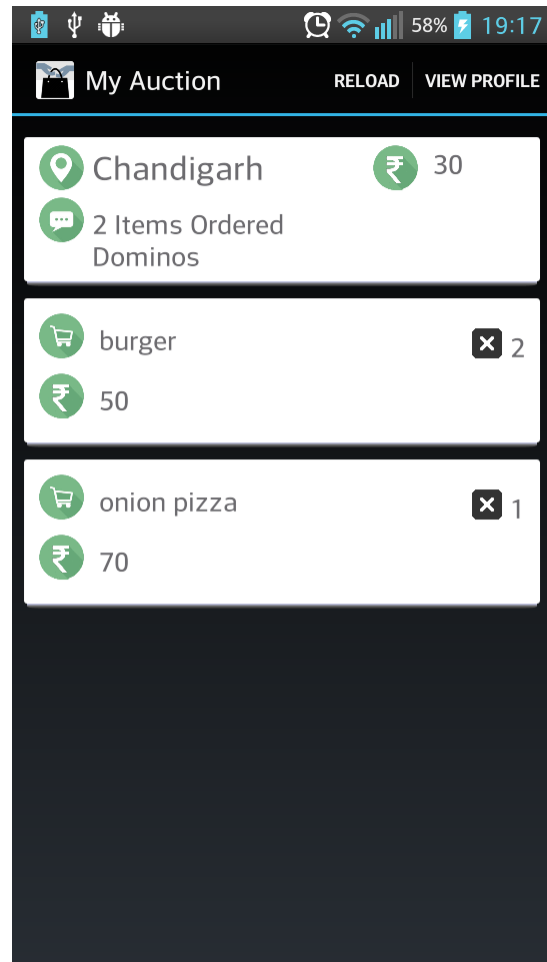
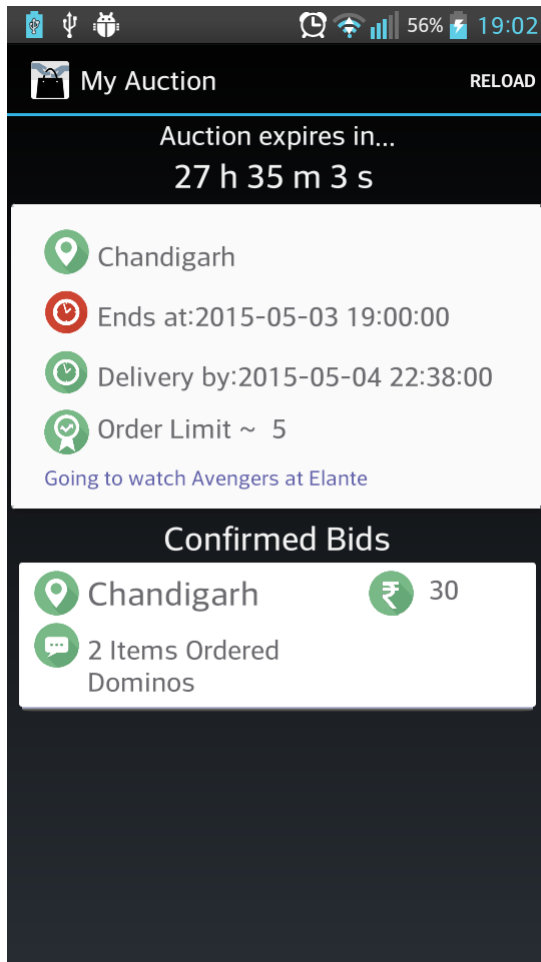
The following describes the major elements of the workflow involved during the Post Auction process:

1. After the Auction has ended, the Service Provider, on the Auction Dashboard, is shown the finally list of Accepted Bids under various categories sorted according to the rank decided by the bid amount of each bid. Here, the Service Provider is also shown the final bid amount of each bid apart from details like location and order.



2. The Service Provider has the option to select any number of top bids from any of the categories (clearly at least one overall). After selection of the bids, the Service Provider can confirm the bids to be serviced by hitting the confirm button.

3. Once the bid is confirmed, all the associated customers are notified about the confirmation.



4. Following the confirmation, both the Service Provider and the customers are shown each other's profile and contact details which can be used by them to contact each other to get the order fulfilled.
5. Also, upon the confirmation of the bid, tuples are added for pending feedbacks in the database corresponding to each bid for the Service Provider and the Customer involved.

Networking components

- Using [Parse API](#), we have used push notifications for notifying users when the user is not actively using the app. Push notifications are sent to all/specific users under the following events:
 - A new broadcast requests gets added. Broadcasted to all.
 - User gets a feedback from customer or service provider.
 - New bid is placed in a one's auction.

When these events occur, changes first occur in our database. Then our server (hosted on netapp.byethost33.com) sends a request to Parse's servers using its REST API. After which Parse sends push notifications to specific/all users based on the request.

- The Android app extensively uses the Internet to communicate with the server and the database for sending and receiving data related to the Auction, Bids, Feedbacks, User Profile, Ratings and other such items.
- To communicate with the server, Android API's AsyncTask with HTTP GET and POST have been utilized. There exists a java class ServerConnect.java which extends the AsyncTask. Under its doInBackground function, it creates an httpClient and pushes the parameter in the form of a GET request to the specified URL.
- Then, in the onPostExecute() method, through a callback method, the JSON object received is sent to the activity from which the execute method of this class object was called.
- The parameter are given to an object of this class in the form a JSON object and care has been taken to keep the data shared as low as possible in order to keep the network consumption low and ensure that the Android app works fine even in sparse network connectivity. For instance, in order to identify an auction, bid, feedback or anything similar, integer identifier have been used rather than attributes like email id which keep the data transmitted over the network to a minimum.

External Libraries

- [Parse API](#): Parse API has been used to send broadcast as well as targeted push notifications.
- [Cognalys API](#): Cognalys android library has been used for the Phone Number Verification process.
- [CardsLib](#): CardsLib library has been used for card based layouts. List of custom cards have been used throughout the app.
- [Gson](#): Gson has been used to parse the fetched json data from the server into java objects.

Project Contributions

Database Schema design

Gaurav Mittal

Google+ Login

Gaurav Kushwaha

Home Screen

Paras Ahuja, Jeevanjot Singh

Profile (Navigation Drawer) Module

App Backend

Gaurav Kushwaha, Jaideep Singh

Server Backend

Gaurav Kushwaha, Kunal Yadav

Layouts

Gaurav Kushwaha, Jaideep Singh, Khan Uzair Suhail

Broadcast Request

App Backend

Naveen Kumar, Kunal Yadav

Server Backend

Gaurav Kushwaha

Layout

Khan Uzair Suhail, Naveen Kumar

Auction Module

App Backend

Gaurav Mittal

Server Backend

Gaurav Mittal

Layouts

Maninderjit Singh, Guparteeek Singh

Bidding Module

App Backend

Mohit Garg

Server Backend

Mohit Garg

Layouts

Mohit Garg, Naveen Kumar

Post Auction

Gaurav Mittal, Mohit Garg, Jaideep Singh

Push Notifications

Gaurav Kushwaha, Mohit Garg

Future plans

- One of the major future optimization planned is to simplify the workflow of the app, which has become too complex to be very usable due to the complexity of the Auction model involved. This can be done possibly by reducing the number of clicks the user has to make to get a task done or by making the User Interface even more interactive, compact and convenient to use.
- Another optimization planned is to efficiently use the local storage capabilities of Android such as caching to reduce the network activity and make use of offline data as much as possible for stuff which is not requiring constant update.
- Further, currently the user has to press the reload button to refresh the page every time some relevant update occurs which might become cumbersome as the activity on both sides of the Auction increases. To deal with this, it is planned to introduce automatic loading of the page with every new notification or update to again reduce the clicks and keep the users from the hassles of every time reloading the page.
- The current Auction model is based on various assumptions and heuristics which means it is going to perform fairly well in the majority of cases but it might fail quite badly in the rest of the fewer number of cases. It might be possible to improve the model as more and more number of users use the app and provide their feedback on various aspects of the app.
- Another major extension to the app planned is to make use of location capabilities like GPS to make the app a general community app where in the user will be shown Auctions and bids only pertaining to its own locality which will be drastically increase the user base using the app.
- Currently, we have implemented phone number verification used missed calls through the tgAuth API. It is planned to replace this method by SMS/Voice verification which is a more widely used method for phone verification and this too will probably be implemented using the current updated version of tgAuth API.