OpenKart: Collaborative Shopping

Srujan Barai
NC State University
sjbarai@ncsu.edu

Daxkumar Amin
NC State University
dkamin@ncsu.edu

Aayushi Agrawal
NC State University
agrawa@ncsu.edu

Shenee Ashara
NC State University
spashara@ncsu.edu

ABSTRACT

With the advent of technology, most of the people wish to purchase goods online. However some people are reluctant to buy goods online. One of the major reason for this is that users have to pay shipping charges if their total shopping amount is not exceeding the free shipping amount. To solve this problem people tend to ask their friends to collaborate or buy unnecessary things just to exceed the free shipping amount. We are looking at an innovative approach to solve the above challenge by reducing the cumbersome task of finding people in nearby area and providing a single platform to resolve the above problem.

Keywords

Collaborative shopping, Android app, Notifications, Firebase, GeoFire, GeoLocation, NoSQL, SMS authentication, Realtime chat

1 Introduction

We tend to face many challenges while trying to adapt to a new city or country after we move. Especially when students travel to another country for further studies or research project, they have problems in commuting to grocery stores without cars or while buying things online and not be able to reach upto free shipping amount. These things adds to difficulties that they are trivially going to face. In the beginning everything is new including people and city and thus, it is difficult to find people to collaborate in buying things together or commuting together to grocery store.

We ourselves have faced similar problems when we moved to another country and have found solutions like forming a group on social media platforms like Facebook or WhatsApp, and communicating on these groups by asking fellow group members whether they are buying something from particular website or going to grocery store that weekend. This takes a lot of trouble and sometimes it happens that people don't check the group or aren't members of group and thus bear some loss by ordering something extra or paying the shipping amount. This solution is not well coordinated and also it takes a lot of time to gather total number of people necessary to reach the free shipping amount. Thus, taking an inspiration from this, we are providing a single platform in form of an android application that would solve all the above problems and help the users to collaborate easily to solve these issues. Here the users will be allowed to announce their purchase in form of creating a prospect order on our application. When someone is finding users to collaborate with, they will be able to set a radius and the app will display the ongoing prospect orders within their mentioned radius with amount due and approximate order date. Also additionally when users collaborate and wants to communicate with other collaborators in the same order they will be able to use the inbuilt chat interface that we provide to resolve any questions they have regarding the order. Thus, we are hoping that our application would provide an efficient platform to solve this day to day problem of the users.

2 SOFTWARE REQUIREMENT SPECIFICATION

2.1 STAKEHOLDERS

For our project there can be very few defined stakeholders and each one has different requirements. The following entities can be considered as stakeholders for our project and their requirements are also given below:

- **Developers and Management team**: Here the project is small scale and the team is thus small. We have developers who are also handling the management responsibilities. Their requirements would be like including implementation of all functionalities, reliability of third party entities, smooth execution of the application, scalability, optimized time and space complexity, easy maintenance and management, etc.
- **Users**: This would include customers and end users. As this application has only one type of end users their would not be a third party dealing directly with users. Users's requirements would be like: interactive user interface, notification alerts, accurate display of order information, a platform to communicate with other collaborators for that specific order, etc.
- **Database Admin**: The main requirement of database admin would be the structure in which data is stored i.e. how easy it is to search and query on that structure and scalability of database.
- **Security Handler:** The requirement of security handler would be authentication without any loop holes such that only verified users can access their respective accounts.

2.2 REQUIREMENTS

2.2.1 FUNCTIONAL

The functional requirements given in Table 1, describe behavior of our application from users point of view. Here, we have considered taking enumerated approach in which we have given priority to the functional requirements. This would make it easier for developers as they can implement the most weighted functionality first and go in decreasing order of weights for implementation. This would make sure that the behavior that is most important to the user is not missed out. This approach can be very useful especially when the implementation time is short and there are chances that all the functional requirements won't be implemented. The priorities are given on a scale of 1-5, higher the number,higher the priority.

Requirement	**P**	**Description**
Authentication	5	Only authorized users are allowed to access the account. Basically signup and login features can fulfill this requirement.
Platform compatibility	4	The application should be fully functional on all versions of Android that are in the market.
Create prospect order	5	This would be the main requirement. The user should be able to create his/her order to find other collaborators.
View order details	5	This is required for collaborators to decide what amount has the current order reached, who are the other collaborators, shipping address, anticipated date to place the order, etc.
Set radius limit of orders	5	Collaborators would like to see only orders started in the set proximity which can be done by setting radius around them.
Communicate with other collaborators	4	This is important because the collaborators may need to communicate with each others for information exchange like where and when to pick up the order once it arrives. There might be some collaborators who wouldn't like their personal information like email ID or phone number to be disclosed and hence it is required that the app itself provides a medium to chat
Notification alerts	3	When someone collaborates in a user's order then he/she should be notified about it.
Attractive and interactive User Interface	3	The user interface should be simple, elegant and easy to navigate through. It should be such that user's have to make minimum efforts.
Rating collaborators	2	A user would definitely like to know who he/she is collaborating with and rating would help them decide this.

Table 1: Functional Requirements

2.2.2 NON FUNCTIONAL

Non-functional requirements can be used to judge operation of the system and not its behavior. In Table 2, non functional requirements are described from developers point of view.

Requirement	**Description**
Usability	User's ease of use is important. This would include autofilling of OTP for authentication, once OTP is auto-filled application should redirect to it's main page and also the sequence of actions.
Reliability	This requirement would include accuracy of location, the reliability of server, no loss of data, if some hard limit crosses then the application should not crash rather it should show some error message, writing of try-catch clauses to avoid the above, etc.
Performance	Performance includes flow of different activities of the application, the latency period of receiving OTP, detecting it and autofilling it should be minimum, database should be scalable and should be able to handle multiple requests parallely.
Supportability	Developers would like the application to work on all the available platforms which would also increase number of users.
Security	Data stored in database should be secure and accessed by authorized people only. In case of some failure measures should be taken that there is no or minimum loss of data.

Table 2: Non Funtional Requirements

2.3 USE CASES

Use case diagram is a behavior diagram which describes a set of actions that a system can or should perform when collaborated with users i.e. actors. These set of actions are called use cases. They basically describe the functionality of application that the user can perform. Use cases will provide some result to either the user or to any stakeholder of the application. Thus, use case diagrams are used to associate actors or users to different set of actions that the application can perform.

Figure 1 is use case diagram for our application associating different functionalities with the user.

2.4 SOFTWARE SEQUENCE DIAGRAM

A software sequence diagram is used to describe interactions of objects in association with time. It depicts the objects and classes involved in the our application and the sequence of messages exchanged between these objects which are needed to carry out the functionality. The parallel vertical lines represent lifelines of objects and the horizontal arrows represent the messages exchanged in order of their occurence.

The sequence diagram of our application is as follows:

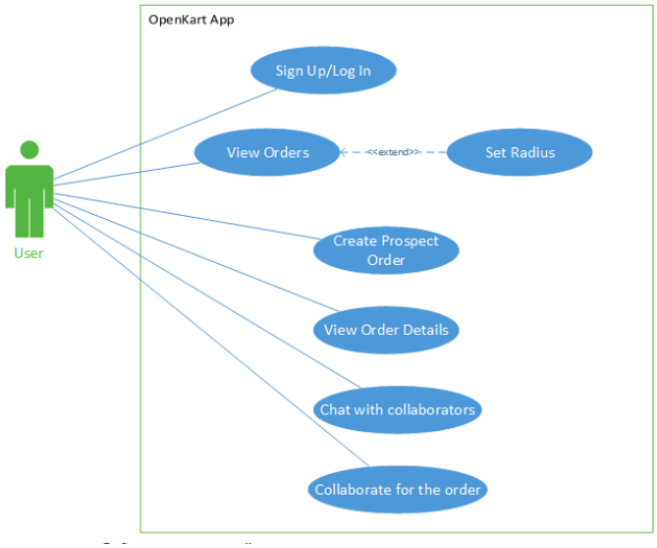


Figure 1: Use Case Diagram

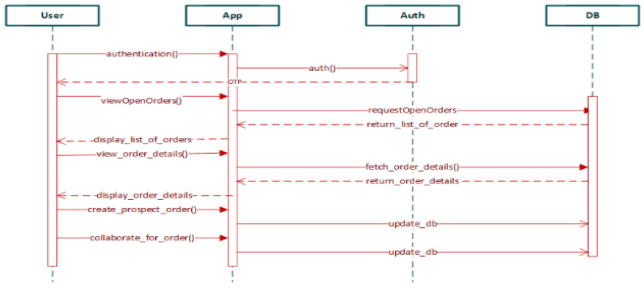


Figure 2: Sequence Diagram

3 METHODOLOGY

3.1 SOFTWARE DEVELOPMENT LIFECYCLE MODEL

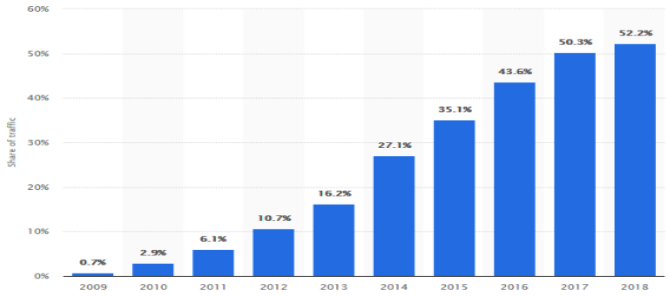


Figure 3: Source: Statista 2018

Agile development was used in developing this application. After reviewing various methods of development, agile was chosen because of the flexibility it offers, with the development phase being divided into multiple small sprints. This was very important because of some uncertainty involved in the development with new ideas coming in at every stage and short period of delivery time.

The technology stack was chosen considering the target audience and ease of usability. Various other factors such as prior knowledge and experience of team members was also taken into consideration. In not-so-patient world, people want access to things that they want from their pockets instantly. Web-traffic from mobile devices have surpassed traffic from Desktop, as depicted from Figure 3. In 2017 Q2, 61 percent of Google searches came from mobile devices.

Based on the above research, we decided to develop an android application. During the development, there was a need for continuous integration, so that during each sprint we can test all the new features added, in accordance with the whole app. For this purpose Github was a very helpful tool. We also managed the development by creating Issues and agile-collaboration board on Github as shown in Figure 4.

3.2 ARCHITECTURE

Any android application uses MVP (Model View Presenter) Architecture. The MVP pattern allows separating the presentation layer from the logic which can be quite used for an android application where in the look and feel of UI is as important as logic implemented. MVP helps making the application extensible and maintainable by well defined separate layers. And, MVP makes things easier for developers and it makes the views independent of the data source.

As shown in Figure 5 given below presenter communicates between model and view. This type of architecture makes it easier to create unit tests. Many times for an android application the view becomes really complex because developers have to take in consideration creating appealing and attractive UI for which MVP provides creating multiple presenters. This is the main reason why android apps uses MVP instead of MVC because in MVC, the controllers are behavior based and shares multiple views and view can directly communicate with model.

For our application view is implemented by Activities i.e. the normal Android views like card view for displaying list of prospect orders. This depends on structure of the application which will help in deciding the layout like fragment or relative. It will call method from presenter everytime there is an user interface action like button click. The model implements all our use cases.

3.3 TECHNICAL DETAILS

The main requirement for a database is scalability and reliability. Firebase provides both these features and thus we have used it to make our application efficient. It is a platform for realtime database, hosting, authentication both email and SMS, storage, cloud messaging, test lab, etc. Firebase is capable of handling 100,000 simultaneous connections at a time giving our application an added advantage.

We are using GeoFire to fetch location based results when the possible collaborator gives the radius to find list of prospect

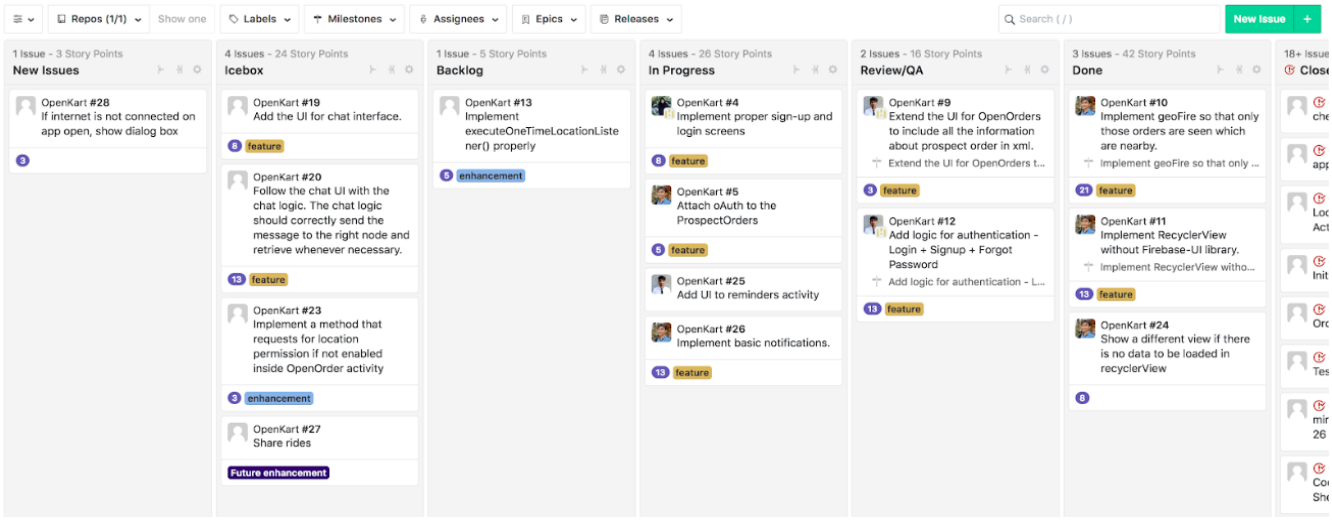


Figure 4: Github Board

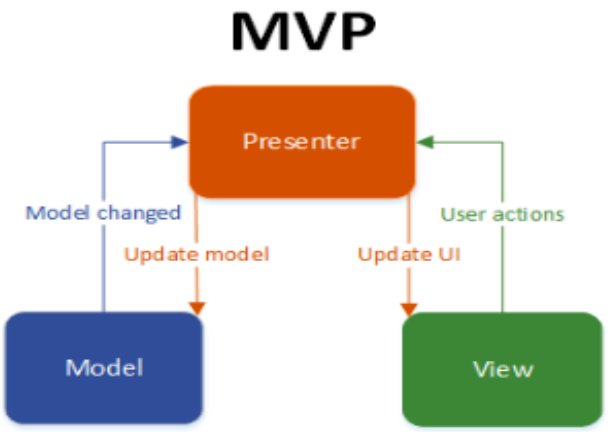


Figure 5: MVP

Libraries Used	Purpose
Firebase Auth	used for user authentication
Firebase Database	used as a remote data repository
Firebase Messaging	used for Push Notifications implementation
Firebase Geofire	allowing query results to be updated in real time as they change
Play services Google Maps	used for including google maps in application

Table 3: Libraries Used

orders. The main advantage of GeoFire is that it uses hashing which makes working of our application efficient. To get location of user we need to get the latitude and longitudes which we are doing with the help of GeoLocation. For collaborators to communicate with each other while maintaining their privacy i.e. not disclosing their phone numbers, we have implemented chat interface which allows realtime bi-directional communication. It is implemented using Firebase.

To create our Android application we are using Android Studio. We chose Android Studio because of many reasons like preview layouts on multiple screen configurations, deep code analysis i.e. it gives detailed explanation about an exception based on the annotation added, template based wizard to create common layouts, gradle based build support, etc.

Figure 6 shows the technical architecture of the main functionality i.e. searching prospect orders.

Here, the data is stored in database on cloud server. Controller is the entity which makes all decisions runtime like firing queries to database, loading sign when database says that it is the last result for the query, showing no results found when database doesn't return anything, etc. To find prospect orders in the given radius, the controller queries the database with input as focus i.e. the location and radius. On receiving this query the database will return the results. The main advantage is that controller works runtime and so if user changes the radius it will immediately take that and returns new results. Here, the database will use hashing to access the nodes. Hashing gives index to nodes which will result in fast execution of search and hence generation of results.

Now instead of finding all the prospect orders in the given radius and then returning the results to the controller, the database will keep pushing the results to controller as soon as it finds them. Thus, it parallely performs search and pushing results to controller which makes our application fast realtime and increases the efficiency.

Once the controller receives the results it generates an object key and passes it to the adapter. The function of

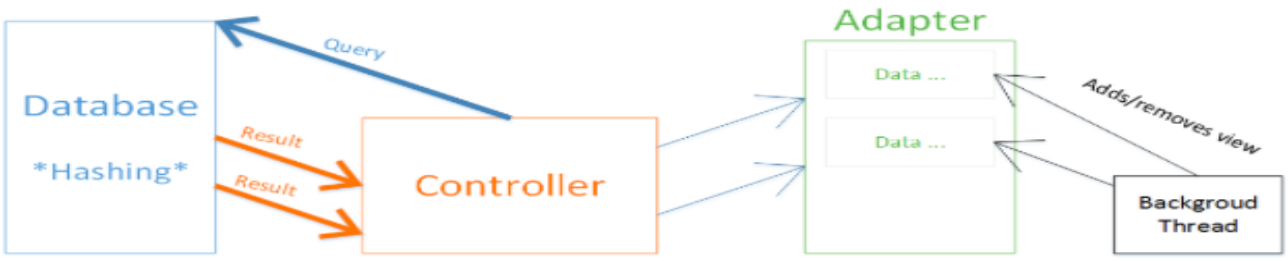


Figure 6: Technical Architecture

adapter is to add or remove the data (related to the object key) in or from Android view(the activity which displays the prospect orders found after search) respectively. Here the background thread is running for the adapter which is used to populate the data in Android view.

3.4 SERVERLESS BACKEND NoSQL DATABASE

We have used "Serverless architecture" (Firebase) with NoSql database for the project. NoSql database means the data is stored in JSON (Javascript Object Notation) format rather than the usual Tabular format. This architecture and database have their own advantages and disadvantages but we decided to use it for this project because of the following reasons:

- It supports serialization of Data to POJO and vice versa.
  - A direct conversion of Data objects to data to-be stored on server is required at a lot of places in our project. Direct serialization makes this conversion to JSON format very easy and efficient.
  - Complex Data Structures like Lists and Hashes can be stored directly in the database without conversion to other formats.
- Makes Authentication simpler
  - We planned to include SMS authentication to make authentication simpler. Firebase backend provides SMS protocols which makes SMS Authentication simpler.
  - We ask user only for their phone number and rest of the authentication (Even the decision of Sign up / Login) is automatic. Also, if user has logged in from a device once, next login (even after app reinstalls) does not even require OTPs (One Time Password). We do this by using Android's core APIs and Server's real-time responses.
- Real-Time bidirectional database
  - We need real-time database for two reasons. Whenever user changes the radius of his interest, even though the request is queried on the server, the results are filtered and responded in real time. This happens because of the real-time nature of NoSql database.
  - We have also included chat facility for users to chat with members of each prospect order. Chat facilities require real-time bidirectional data-transfer.
- Support of GeoFire.
  - We use GeoFire which uses the technique of Indexing and Hashing on Location values to give results of "Near-by locations". It works efficiently even when the data is in thousands. It requires NoSql database to work.
  - It does not wait for all the data to be parsed, rather it starts returning results as soon as it parses individual items. This pipelining is possible bacause of NoSql nature of database.

4 USER INTERFACE SPECIFICATION

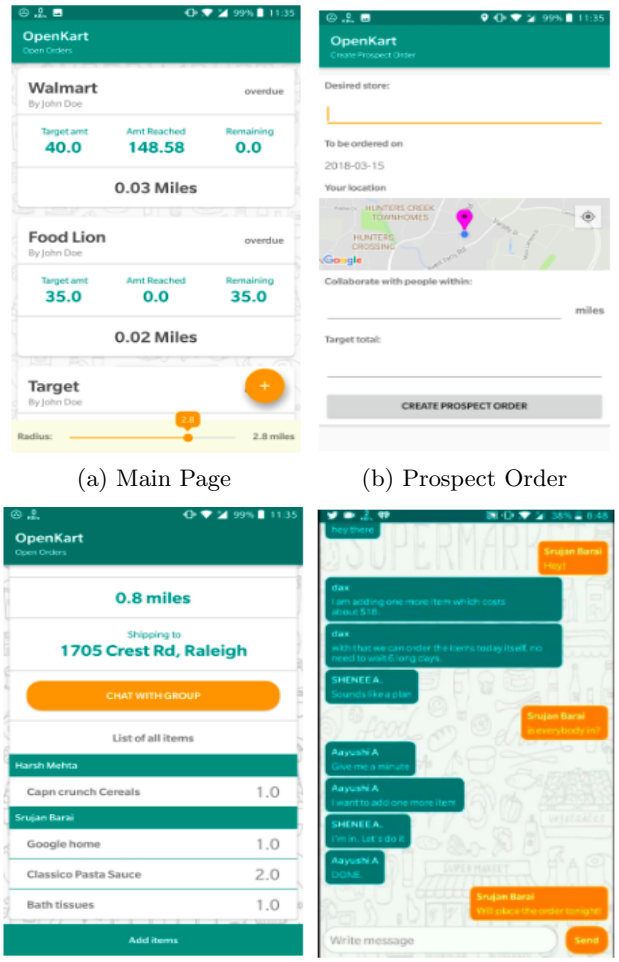
4.1 PRELIMINARY DESIGN

Figure 7(a) shows the main page when you open the app. There are various cards for the orders that are open around the user's location. Here the card shows basic details of the orders such as store name, target amount, amount reached and the distance from user's location to the location where order was created. It also shows the intended date of the order. If that date is today, it shows "today" and if the date has passed it shows 'overdue'.

The page is being implemented in Recyclerview. The major advantage of recycler view is that it populate the view at the run time, hence very less memory is used and dynamic, more efficient and robust view hierarchy is created. Hence, new cards containing the order details will load only as you scroll down the page and hence taking very less time. User can adjust the radius from the bar at the bottom of the screen and the page will get updated accordingly in real-time.

Figure 7(b) shows the Create Prospect Order page. Here the user can enter the details of the order has he/she wants to post on the application. It takes the current location of the user and points it on the map. The latitude and longitude of the location is stored in the database, which is used to show the orders based on distance.

When user clicks on the card of a particular order, it renders a page where more details of that order are given. Here a user can see the list of items and other users that have collaborated within that particular order. From this page, a user can also switch to the chat interface, where he/she can send messages to the users in collaboration and also view their messages.



(a) Main Page (b) Prospect Order

(c) Particular Order (d) Chat View

Figure 7: Screenshots

4.2 USER EFFORT ESTIMATION

User effort estimation is the technique by which we can calculate the ease of use from user's point of view. It takes in consideration two parameters: count of clicks and/or keystrokes that are needed to complete a task and the background process that takes place for them like entry or manipulation or extraction of data from database. This can be done for few selected scenarios of an application and can be calculated by taking above mentioned parameters.

For any application the goal should be to minimize user effort as it will be more convenient for users because the work done by them decreases in comparison with what application does. Here, we taken several important scenarios for our application as shown in Table 4.

5 TRADE OFF MATRIX

Usage Scenarios	Number of clicks and/or strokes	Entry/Manipulation/ Extraction of data from database
SignUp	2 (to submit phone number for OTP and name for profile)	2 (entry in database for phone number and name)
Login	1 (to submit phone number for OTP)	0 (Only checking database for given phone number)
Creating prospect order	3 (for click on + sign, then click on create prospect order and lastly one click for submitting prospect order details)	3 (latitude, longitude and locationID are added in GeoFire node in database) + 8 (Entry of various details like order data, minimum amount, etc in 4 hierarchical nodes of Prospect order node in database)
Searching some prospect order to collaborate with	1 (slider in UI to adjust the radius)	7*number of nearby prospect orders found (It will extract orders according to the radius and get details from 7 nodes like order date, name of store, amount due, total amount, etc)
Collaborating with some prospect order	2 (one click for add item button and then one click to submit the form for your order)	6 (4 entries made in hierarchical way in Collaborators node for link of item added, item name, price and quantity)

Table 4: User Effort Estimation

-	1	2	3	4	5	6
OpenKart	++	+++	++	+	++	++
Whatsapp Group	++	+	-	+	-	-
Facebook Post	+	-	–	+	-	-
Face to Face Communication	++	–	++	–	-	–

(1) Security - Proper validation of user and non-disclosure of personal information like phone number or email ID
(2) Adaptability - Users adaptability of using the service
(3) Coordinability - Ease of coordination among collaborators
(4) Availability - Scope of getting prospect orders
(5) Time Saving - Time saved in finding collaborators
(6) Convenience - Level of convenience in finding collaborators

6 TESTING

It is important to test every feature in an application so that when a user crosses a boundary condition then the application instead of crashing, an error message is displayed and further instructions are provided. To test any feature first we need to study its behavior and decide pass-fail criteria

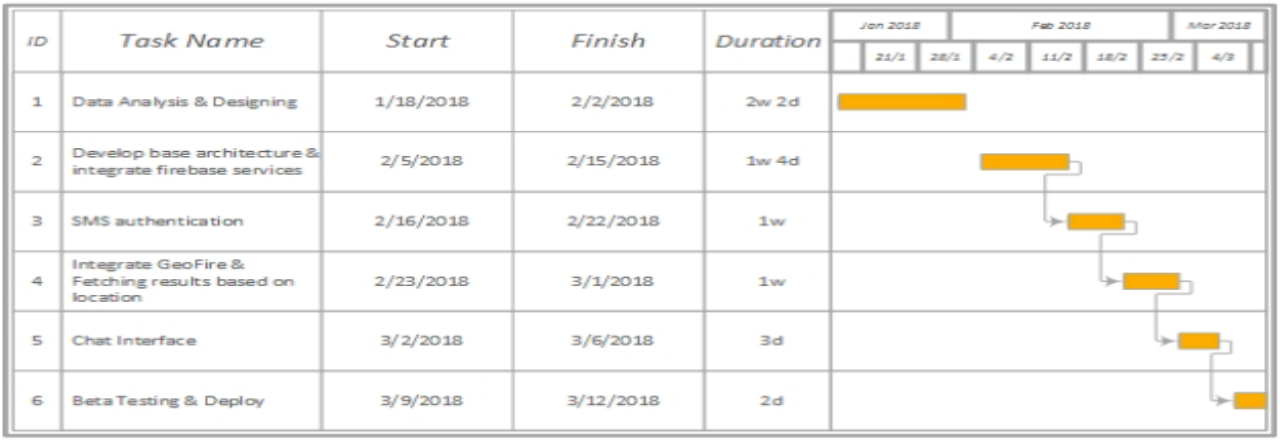


Figure 8: Gantt Chart

according to which tests should be written with boundary conditions. Following are few features that are tested for our application:

- Runtime working of GeoFire:
  To test GeoFire's working we set two boundary conditions i.e. when radius is zero, no prospect orders should be shown and when radius is set to its maximum limit, all the orders present in database should be shown to the user.

- Accurate display of prospect orders according to the radius set by user:
  GeoFire requires the radius to be in kilometers while the application deals with distance in miles. Thus, we have tested the conversion wherever and whenever required, to ensure smooth working of GeoFire while connecting distance given in radius with

- GeoFire to find nearby prospect orders.
  List of prospect orders:
  Here, if any entry in the database gets corrupted due to some reasons we made sure that it doesn't affect other entries and functioning of application remains the same.

7 PLAN OF WORK

Gantt chart, Figure 8 shows the list of the main functionalities that we implemented for our application in accordance with timeline. It was used by our team members to look at regularly and allocate time to sub-tasks accordingly. It gave a fair idea as to what is going on in the development.

8 USER EVALUATION

To check the acceptance of of our application among users we created a google form to take survey of our fellow classmates. 19 students filled our survey which included a link to APK file of our application and also youtube link to the video which would help users navigate through the application.

The survey questions were aimed at knowing how users felt about the application and whether they would be interested in using it in future. Following section elaborates our survey questions with reason why that question was a part of the survey and also the statistics of the response that we got.

The survey included sections for email ID and name of the user but as mentioned by professor we didn't make them required as to ensure the right to anonymity of the users.

1. How useful do you think this app could be?
This question was intended to get the usefulness of the application which in turn would tell us that whether people would like to use it in future or not. This is important because user base is the most important for an application. There's no meaning of an application that people are not going to use in future.
Figure 9 shows the result for this question:

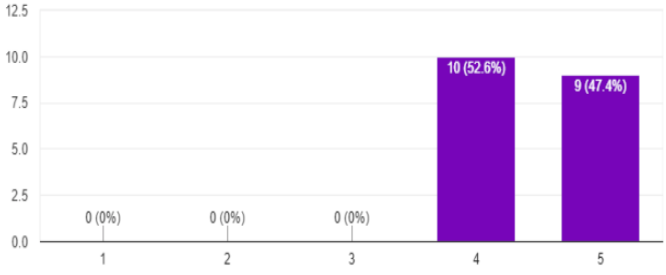


Figure 9: Usefulness of App

As we can observe all the users rated 4 and 5 for this question which indicates that this application would serve its purpose and be useful in future.

2. Are you able to navigate easily through the app?
It is very important that user can easily navigate through the application. No user would like to use an app in which they have to remember the flow of actions or arenâĂŹt able to find the functionality that the application is meant for. Our

app has simple navigation, for example, if the user wants to create a prospect order he/she will have to click on a plus sign at right bottom corner. This would be really easy for users to understand as plus indicates adding something new.

Figure 10 shows the result for this question:

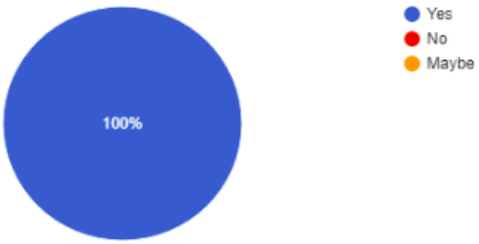


Figure 10: Easy Navigation

As we can see all our users found it really easy to navigate through the app without any assistance and we got 100 percent result for this question.

3. Rate the user interface of the app.
For any any application look and feel of user interface is the most important thing. The first thing that the user is going to observe is the UI of an application. If the UI is appealing and attractive the user would automatically be excited to try the new app. Users are always eager to use applications with cool UI. Also user interface should not only be appealing but should be simple as well. A complex UI with lots of things on one page wouldn't look good, rather a UI with clearly defined objects in various page would be preferred.
Figure 11 shows the result for this question:

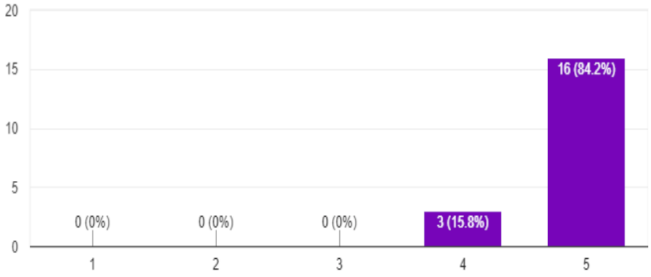


Figure 11: User Interface Rating

We received ratings of 4 and 5 from every user that filled our survey which indicates that users are comfortable using our application.

4. Did our app fulfill your expectations?
This question was intended to know user satisfaction. We wanted to know whether we fulfill all the expectations that the user thought on reading the description of the application. Further we have included questions in which they can also tell us whether they want something different or want to add some new feature. Figure 12 shows the result for this question:

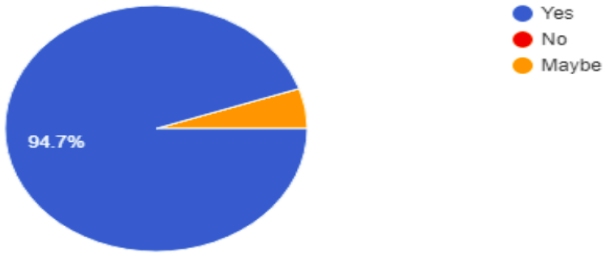


Figure 12: Fulfillness of Expectation

94.7 percent users were satisfied with the functionalities that we provided which indicates that our application fulfills most of the expectations if the user.

5. What feature did you like the most?
We wanted to know what feature of our application was the one which users liked. This information can be used to may be enhance that feature in future versions of the applications or also in some new application that we make in future.
The answers that we received are as follows:

- Showing order status and already integrated room for future improvements.keep up the good work. Best of luck.
- The ability to go to the item in the app directly by clicking the link.
- The chat features within the app.
- For authentication, we receive otp in sms and it gets autofilled at the same time, very quick.
- The radius feature which automatically filter the plans which is helpful to get the nearby order plans. Also, i can create my own plan and share it with someone who wants to join my plan.
- Setting the radius and viewing who ordered what in the package.
- UI is stunning, Taking the radius into consideration is a really good idea.
- The results of search prospect orders are really quick.
- Ease of use and idea, map redirection
- Setting of the radius and collaborating with other users to finish the target amount.
- The UI is really good especially sliding for radius and adding link of item while collaborating which will facilitate the person who is going to place order.
- Feature of group chat enabling all the collaborators to chat together
- Collaborating with other users to finish the target amount and also showing setting up the radius
- Group chat between collaborators
- It has built in maps, locates people close by, Very useful in real time for online ordering, fetches details from Walmart link directly, UI is beautiful
- collaborating with other shopper is actually great idea. Also adding distance metric it very useful for users.
- The UI is very elegant. The radius bar which they have included to filter the items is amazing Clicking on the item list redirects you to the app. I was very impressed with the phone number verification. Many big companies still aren't even using that. Chat was

also a great idea! Radius is a must have feature I probably would have forgotten about. Great job!

6. What can we do to make our app better?
This question was intended so that users can give their opinion regarding our application. In a case where user want something different or wants to add a new feature, he/she can inform us in this section. We would try and include it in next version there by meeting user expectations and giving them a satisfiable experience of using our application.
The responses that we received are as follows:

- Can remove already closed orders or reduce priority so that other orders can also meet their target.
  - Justification: We have already included a 'Status' flag in Prospect Order model. This can be used to make sure that prospect orders won't show in list once their status changes to 'Ordered' or 'Delivered'.
- Directly parse the amount from the link
  - Justification: We did plan this and successfully implemented using JSOUP. We did not include in deployed version as parses works for specific website and it is impossible to write a generic parses when people can order from ANY ONLINE WEBSITE.
- Can make it available across multiple platforms such as ios
  - Justification: Noted.
- You can add a feature to add custom notification which takes in the miles and get the notification if someone adds plan in that radius.
  - Justification: We have mentioned that in our "Future work". The team who takes our project can plan to work on that.
- Add transaction management
  - Justification: We aren't sure where we can do transaction management exactly. We can though, collaborate with splitwise and make the order export to splitwise or similar apps.
- Notifying the main person who is placing the order when some collaborator joins.
  - Justification: Considered under Notifications.
- Providing notification when group limit surpasses the needed limit
  - Justification: Considered under Notifications.
- The team has included all the features including the chat which is impressive. I think the app is perfect for the scope which is defined. The developers might need to take care of edge test cases
  - Justification: Thank you. We have tried to check all test cases as far as we could, we will keep this in mind and do it for all future iterations as well.

9 FUTURE WORK

For future work there are many new features that can be added to the applications. In the current implementation there are some immediate scope of improvements. There can be some improvements done to the chat interface. Keeping the already implemented features preserved, future work would focus on including a carpooling feature into the present application. Push notifications can be implemented. This will notify users whenever there are any changes in the order that they have collaborated in.

Another feature in which notifications can be sent to users is when they set a reminder. There can be facility where a user can set a radius limit and store name in reminder and if any suitable prospect order is created within their proximity then alert notification should be sent to that user so that he/she can collaborate.

Also, collaboration can be made with transaction management companies such as PayPal or Venmo to handle the payment and sharing of cost among various collaborators. This would make payment and cost sharing easy and automatic within the application.

10 CONCLUSION

Our initial research on international students group gave us a revelation on the problems faced by them during grocery shopping. The survey results showed that more than 80 percent of the survey takers faced a situations where they paid shipping charges and were interested in an app that could solve this problem. The purpose of creating such an application was to facilitate online shopping for users and to provide a trusted platform where they can easily collaborate saving their time and money. The user evaluation results also agrees with our motive and also that we were able to meet the expectations of the users.

References

[1] Jiao, Junfeng, Anne Moudon, and Adam Drewnowski. "Grocery shopping: how individuals and built environments influence choice of travel mode." Transportation Research Record: Journal of the Transportation Research Board 2230 (2011): 85-95

[2] Sultan, Muhammad Umar, and Md Uddin. "Consumers' attitude towards online shopping: Factors influencing Gotland consumers to shop online." (2011)

[3] Michael Yoseph Ricky Mobile Food Ordering Application using Android OS Platform. Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 2014

[4] Shanthi, R., and Kannaiah Desti. "Consumers' perception on online shopping." Journal of Marketing and Consumer Research 13 (2015): 14-21.

[5] Delafrooz, Narges, Laily Hj Paim, and Ali Khatibi. "Students' online shopping behavior: An empirical study." Journal of American Science 6.1 (2010):137-147

[6] Potel, Mike. "MVP: Model-View-Presenter the Taligent

programming model for C++ and Java." Taligent Inc (1996): 20.

[7] Landon Cox Firebase Realtime Database Duke University, 2017

[8] Nagra, Gagandeep, Gopal, R. "An Study of Factors Affecting on Online Shopping Behavior of Consumers." Internatinal Journal of Scientific and Research Publications, Volume 3, Issue 6, June 2013

[9] Jianye Liu, Jiankun Yu Research on Development of Android Applications 2012

[10] Firebase, firebase.google.com/docs

[11] Wikipedia, *Software development life cycle*

[12] YoYo Library, *https://github.com/maxogden/yo-yo*

[13] GeoFire, *https://github.com/firebase/geofire*

CHIT NUMBERS

- VBM
- STT
- FOR
- ZCZ
- IMP
- IJK
- ZCK
- XHH
- EYQ
- PLH
- MXK
- UWU
- SGK
- AGQ
- HBE
- NFG
- KCN
- WHI
- WXS