

Polling and Quiz Application

B.Tech Project Report

Submitted by
Akshay Malav(B16CS003)
Chinmay Garg (B16CS041)

Under the supervision of
Dr. Sumit Kalra



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Department of Computer Science
Indian Institute of Technology Jodhpur
April 2019

Table of Contents

1. Introduction	3
2. Problem Formulation	4
3. Overview of Methodology	5
3.1 Online mode	6
3.2 Offline mode	7
4. Detailed Methodology	8
4.1 Online Mode	8
4.2 Offline Mode	8
5. Results	12
6. Conclusion	14
7. References	14

1. Introduction

Today in the era of digitalisation, everything is coming online , whether it is shopping, buying groceries, etc. People do not have to go anywhere but they can be virtually present everywhere, everything is available at the tip of their fingers. One such use case is polling. Everyone in daily life sometime require to know opinion of a group of people or ask some general questions. In this case having a application which has a common forum across people helps a lot. One can ask a question on that forum and members can easily vote their opinions and the result can be seen.

Taking this use case we have made an android application which can do the above mentioned task. The application provides a lot of hands-on features to the user. User, first has to login with it's account details or it can first register. Then there are two modes of operation Offline and Online. After choosing the mode of operation, user can use the services provided by that mode. Both Offline and Online modes have different services and are used in different case scenarios, we will see those scenarios later in the report.

The report contains the details about the app, it's mode of operations, features, etc. The methodology of the processes are also explained in detail. All the references and conclusions are made at the end of the report.

2. Problem Formulation

As explained in the Introduction our problem statement for the B.tech project was to make a **Polling** and **Quizzing** android application which can be used to conduct a general purpose poll where people can vote and to conduct a quick quiz in classroom. There are two modes of operation of the application **Offline** and **Online**.

Offline mode can be used to conduct a quiz and a poll where only people who are in the range of the device can participate, for that we had to find what APIs or third party services that can be integrated in order to achieve the offline functionality. We also had to ensure that no internet connection is required for the device and poll or quiz can be shared with the whole classroom and not with just few people.

Online mode can be used to conduct a poll where people have to join a group, of which they want to be a part of. Now they can receive the ongoing polls in that group. For this we had to ensure that there is a database which is fast and fetches the details of the poll feed in a real-time fashion. We used Google Firebase as our database and integrated it with our application.

3. Overview of Methodology

Below are the component diagrams to explain the working methodology of the application:

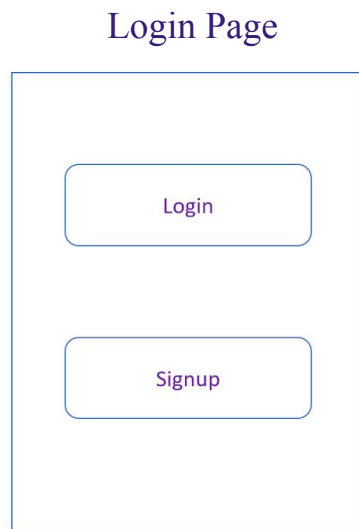


Figure - 1

Figure-1 is the login page of the application where user can register or log directly into an existing account.

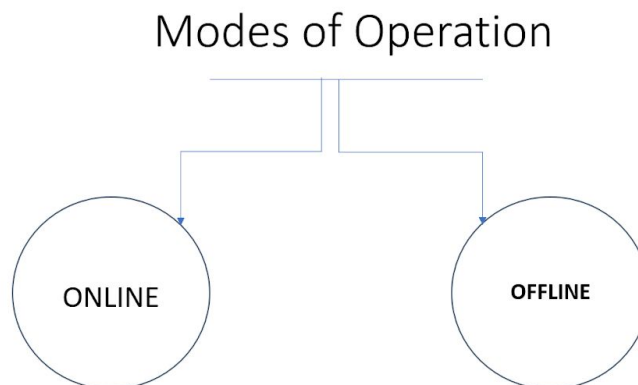


Figure - 2

After Logging in the user will have two options either to go offline or online mode.

3.1 Online mode

Database Involved



Figure - 3

Figure - 3 depicts the device interacting with the Firebase. Devices in the online mode will be continuously interacting with the Firebase any changes there will be reflected in the devices.

3.2 Offline mode

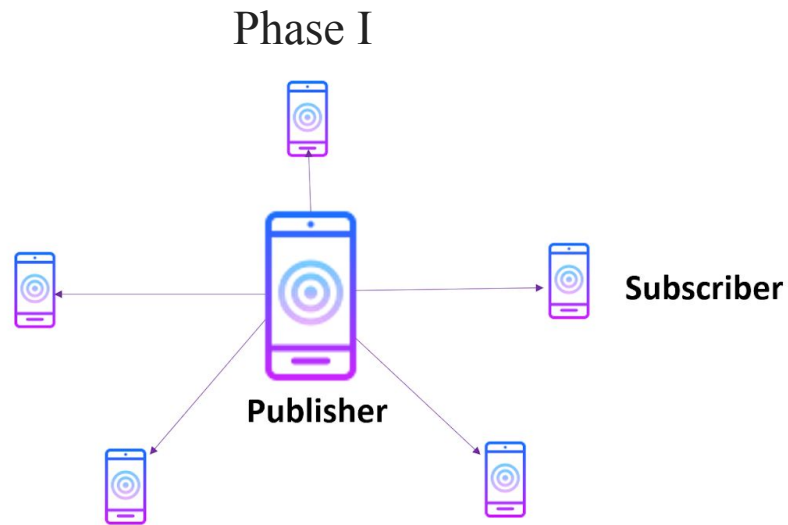


Figure -4

Figure - 4 explains the working model of offline mode, in first phase the publisher will publish the poll or quiz to the nearby devices which will subscribe to the publisher.

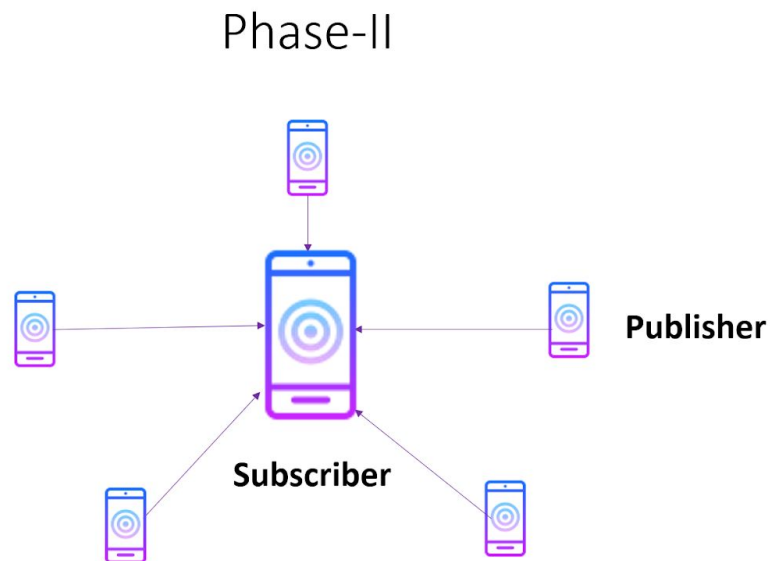


Figure - 5

Figure - 5 explains the second phase of the working model , here after publishing the question , the publisher will have to now subscribe to all the publishers who are sending the answers.

4. Detailed Methodology

4.1 Online mode

In online mode we have used Firebase as our database which stores the information about the user and groups.

The poll feed tab in the application contains poll in the order of their expiry time that is active poll first and expired afterwards. Clicking on the + Floating Action button the user will be redirected to new page where he/she can create a new poll and send it to different groups.

The Group tab contains the list of groups that the user is part of, here user can create new groups, can delete them and even join existing groups.

The third tab contains the information about the members of the group.

To keep the all the lists dynamic we have used **RecyclerView**. Because whenever there are updates in the database the lists have to be updated. So the recyclerview will be listing all the items which has to be listed and make changes if possible.

4.2 Offline mode

For transferring our polls/quizzes and their replies between the devices in a **fully offline** P2P manner, we have used the **Nearby Connections API 15.0.1**. It provides connections between devices that are high-bandwidth, low-latency, and fully encrypted to enable fast, secure data transfers.

Nearby Connections uses WiFi, Bluetooth LE & Classic Bluetooth under the hood to discover and establish connections to nearby devices. It abstracts away the inherent complexity of these radios by leveraging the strengths of each, while circumventing their respective weaknesses. Aside from the obvious advantage of

sidestepping the pain of dealing with the vagaries of these radios across different OS versions and devices, this abstraction enables seamlessly upgrading the bandwidth of a connection by switching between the radios as and when it makes sense, as well as getting invisible over-the-air updates to use new radio technology as it becomes available - with no change whatsoever in the application code.

Initially, we had used the **older version** of this API, which required setting up and connecting to the application's **Google API Client**, obviously through the internet. The Google API Client was required so that message published by one app does not ends up in some another app, which is of course unrelated. But this issue of internet connectivity was solved in the 2017 rollout for this API. So, now our offline mode is 'genuinely' offline.

Now let's dive into the technicalities of the present version that we have implemented:

- Usage of the API falls into two phases: pre-connection, and post-connection.
 - In the pre-connection phase, **Advertisers** advertise themselves, while **Discoverers** discover nearby Advertisers and send connection requests. A connection request from a Discoverer to an Advertiser initiates a symmetric authentication flow that results in both sides independently accepting (or rejecting) the connection request.
 - After a connection request is accepted by both sides, the connection is considered to be established and the devices enter the post-connection phase, during which both sides can exchange data.
- We are **automatically accepting** the connection requests on both the sides. So, the only requirement for the connections to be established is that they must have same Service IDs. Generally speaking, every app has a unique Service ID, so that Advertiser of one app does not get connected to

Discoverer of another app, in case two different apps are using the Nearby API at the same time at the same place.

But, our app has **2 different Service IDs**, one for Polls and another for Quizzes. So, a person discovering for polls would not receive a quiz.

- Nearby Connections supports different Strategies for advertising and discovery. We have used 2 different strategies for 2 different scenarios. Let's understand it with an example of teacher sharing a poll with his students.

- **P2P_STAR**

First the teacher advertises for a poll and the students act as discoverers. So, the star connection is established with teacher as a hub and the students as spokes.

P2P_STAR is a peer-to-peer strategy that supports a 1-to-N, or star-shaped, connection topology. In other words, this enables connecting devices within radio range (~100m) in a star shape, where each device can, at any given time, play the role of either a hub (where it can accept incoming connections from N other devices), or a spoke (where it can initiate an outgoing connection to a single hub), but not both.

- **P2P_CLUSTER**

Now after the poll is shared, we destroyed all the previous connections, and established new connections with students acting as advertisers and the teacher as the discoverer.

P2P_CLUSTER is a peer-to-peer strategy that supports an M-to-N, or cluster-shaped, connection topology. In other words, this enables connecting amorphous clusters of devices within radio range (~100m), where each device can both initiate outgoing connections to M other devices and accept incoming connections from N other devices.

- Now, this is a mistake that we did, as there is no need for destroying the previous connections and re-establishing new connections. It happened because, previously, we did not have knowledge that these connections are fully-duplex. So, in short, only P2P_STAR is required.
- Once connections are established between devices, you can exchange data by sending and receiving Payload objects. We are using the **Byte Payloads**. Byte payloads are the simplest type of payloads. They are suitable for sending simple data like messages or metadata up to a maximum size of byte array of 32k. BYTES payloads are sent as a single chunk, so there is no need to wait for the SUCCESS update (although it will still be delivered, thus making it a reliable data transfer, like in TCP).

5. Results

All the operations were done successfully. Any bugs found were removed during the debugging and test running process.

Both online and the offline modes are working in a streamlined fashion with no problems. All the data transfers were almost instant.

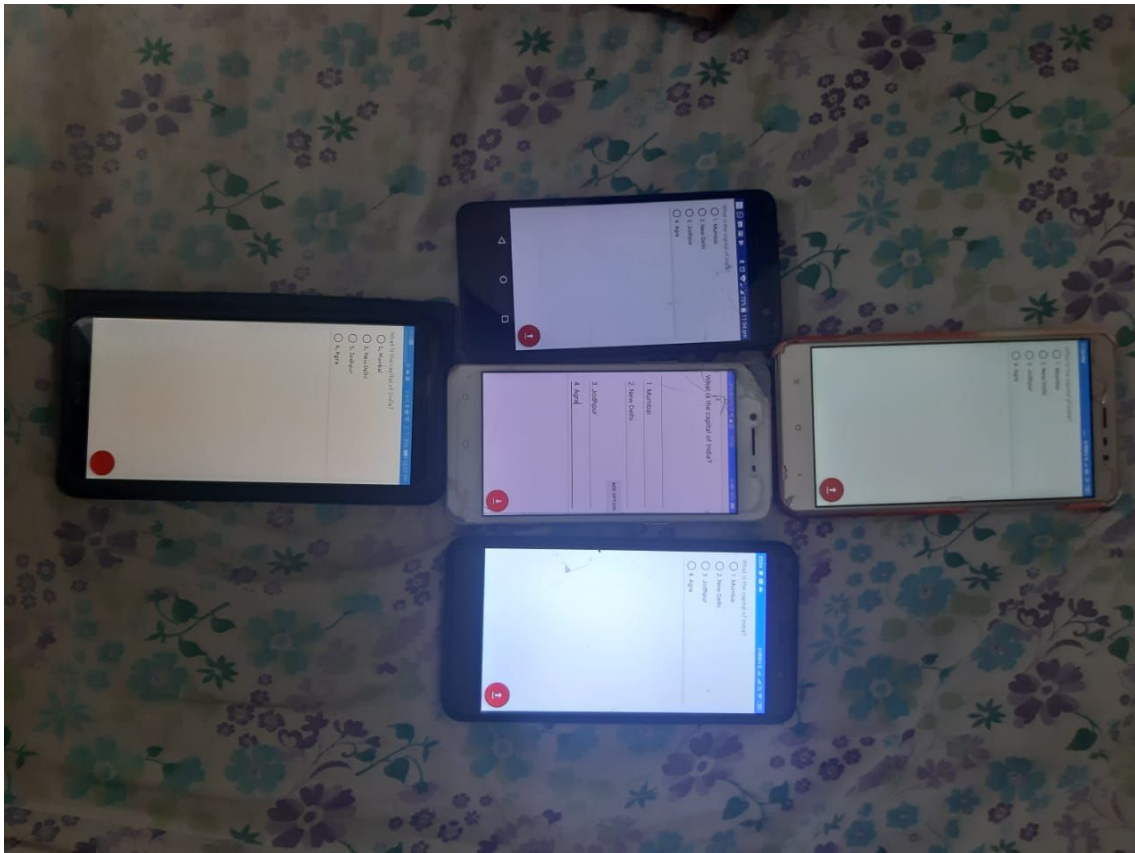
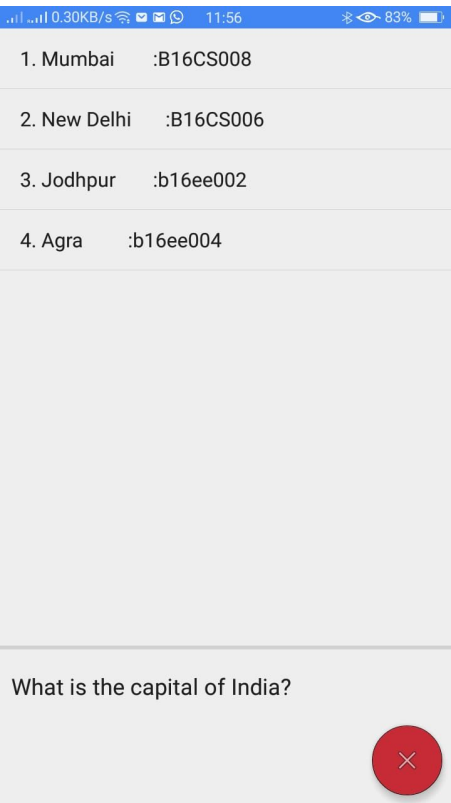
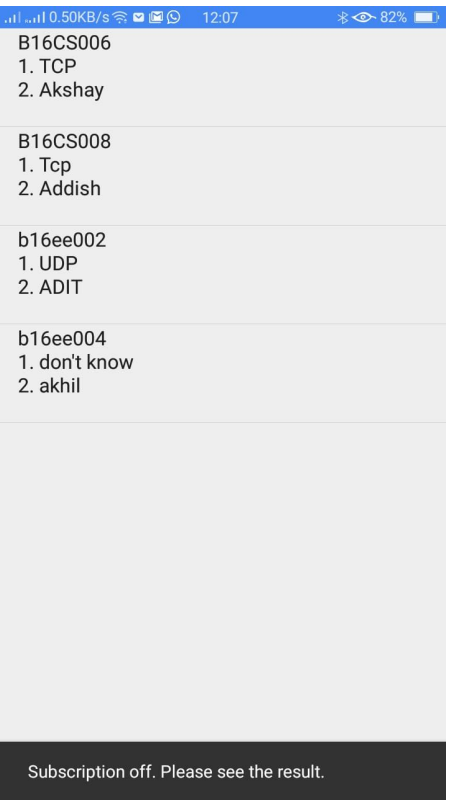


Fig 6. Nearby devices receiving the poll.

 <p>1. Mumbai :B16CS008</p> <p>2. New Delhi :B16CS006</p> <p>3. Jodhpur :b16ee002</p> <p>4. Agra :b16ee004</p> <p>What is the capital of India?</p>	 <p>B16CS006</p> <p>1. TCP</p> <p>2. Akshay</p> <p>B16CS008</p> <p>1. Tc</p> <p>2. Addish</p> <p>b16ee002</p> <p>1. UDP</p> <p>2. ADIT</p> <p>b16ee004</p> <p>1. don't know</p> <p>2. akhil</p> <p>Subscription off. Please see the result.</p>
<p>Fig 7. Asker receiving the replies for his poll.</p>	<p>Fig 8. Asker receiving the replies for his quiz.</p>

6. Conclusions

Firebase should be handled carefully in the application, handling it carelessly can result in app crash or uneven updates in the database. There should not be any [.] in the name of the key while storing data in the Firebase.

Nearby connections API has two versions, in the first version the devices need a low speed internet connection in order to remain connected with the API Client but in the second version that we have used this dependency is removed and there is no internet connection needed anymore.

7. References

1. Android Developers. “Nearby Connections API” [Nov 28, 2018]. Retrieved in Apr, 2019. <https://developers.google.com/nearby/connections/overview>
2. Android Developers. “Connect to Firebase”. Retrieved in Mar, 2019. <https://developer.android.com/studio/write/firebase>
3. Android Developers. “Create a List with RecyclerView”. Retrieved in Feb, 2019. <https://developer.android.com/guide/topics/ui/layout/recyclerview>
4. East David. “What is firebase and how to use it in Android?”. [Nov 15, 2015]. Retrieved in Mar, 2019 <https://stackoverflow.com/questions/33720614/what-is-firebase-and-how-to-use-it-in-android>