EVote – The revolution of Vote

Tianming Zhu

Follow this and additional works at: https://digitalcommons.harrisburgu.edu/csms_student-coursework

Part of the Computer Sciences Commons

Recommended Citation


This Dissertation is brought to you for free and open access by the Computer and Information Sciences, Graduate (CSMS) at Digital Commons at Harrisburg University. It has been accepted for inclusion in Other Student Works by an authorized administrator of Digital Commons at Harrisburg University. For more information, please contact library@harrisburgu.edu.
EVote – The revolution of Vote

by

Tianming Zhu

Applied Project report submitted to the Faculty of the Graduate School of the

in fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

Supervised by: Abrar Qureshi, Ph.D.

Spring 2020
Abstract: This report focuses on independently developed applications that are primarily used to support group or group solutions. The tools used to develop this application include Android development tools, and Google Firebase development tools are also involved in the development process.

Keywords: Android Studio, Android SDK, Java, Google Firebase

I. INTRODUCTION

Today, decision-making always plays an important role in companies, research institutions, jobs and technologies. Voting is the most effective way to make the results fair and convincing, especially in intense discussions that cannot be compromised.

To solve this problem, you need to develop a voting application. There are many ways to vote, and traditional methods (such as writing in a poll or voting using a "hand-held" method) have disadvantages:

1. Voting on paper: Similar to voting on a black box. Handwriting by individual
makes it difficult to recognize the recorder. In addition, we do not consider the environment.

2. Raise your hand. People may be ashamed to announce real ideas. Meanwhile, if other players raise their hands, they can lead to mistakes and fool you.

With these possibilities in mind, I would like to develop fair, accurate and environmentally friendly applications.
This is the birth of EVote, a mobile voting system. The application has two voting scenarios.

The first scenario is that each individual in the group has its own mobile application and each has access to the Internet.

In this case, you need All Mighty, a server that plays a role that knows
everything but doesn't tell anyone. Each one told him his / her companion, and he would record what he said, reporting results only when everyone gave their opinion from their mobile phone.

The second scenario is based on the situation where not all voters have their own mobile devices or people have restricted access to the Internet.

In this case, use only one mobile phone for the entire voting process. The mobile phone itself will play the role of almighty. Everyone voices their opinion on the phone and shows results only after everyone has finished voting.

II. TASKS AND SOLUTIONS

A. UI & UX:
This section focuses on the application flow and user experience.

The screen saver first appears when the user clicks the icon on the mobile phone's home screen. After the application loads successfully, the user is presented with a landing page asking if they want to participate in online or local polls:

- Voting Online
- Voting Local
The following screen will be shown for online voting:

- Create Session
- Join Session

fig 6. Online voting landing page

The app will ask you number of people and number of selections when you’re creating a session:

# of Voters:

# of Options:

fig 7. Online host page
After a while the app will generate a session key for you:

Your session Id is:

1234567

The host waits for guests to make a selection, and the screen updates as soon as all guests have completed the poll:

Winner is A:

- 60%
- 10%
- 10%
- 10%
- 10%

New Session
On the other hand, in the case of the guest screen, it is necessary to join the session based on the session ID:

![Session Id](image)

next

![Input session id](image)

after session id done the following screen shown up:

Please make your vote:

- [ ]
- [ ]
- [ ]
- [ ]
- [ ]
- [ ]

next

![Make selection](image)
The app will be waiting for all the voters to make their decision:

Please wait for other voter:

- [ ]
- [ ]
- [ ]
- [ ]
- [ ]
- [ ]

next

fig 12. Select and wait

After everyone has finished voting, results will be displayed as organizer results:

Winner is A:

- [ ] 60%
- [ ] 10%
- [ ] 10%
- [ ] 10%
- [ ] 10%
- [ ] 10%

New Session
This diagram shows the entire user flow:

As you can see in the diagram above, the idea is very clear without any instructions to the user. Users simply do what they want to do, following the on-screen options.

B. Technology dive in:
To understand the above user story, you need a logic diagram.
This diagram shows the logic of the code throughout the application. Think of each square component as one screen. When designing screen objects, some objects are similar and can be reused. On the other hand, some screens are similar to the previous screen changes. In this case, there is no need to create a separate screen for each of the above squares. After each vote, the application
can allow the user to resume another voting session, process and delete objects and data, and improve service throughout the system.

Based on the logic above, we combine the whole activity and the fragment into two main Android components to represent the whole application.

An activity is a screen that is displayed to the user as a major component of Android and is an interaction between the user and the screen for any operation. There is an important callback method for setting up the user interface with an activity called onCreate (). This method is used to initialize and configure all presentation elements that will be displayed to the user. However, you can also configure the user interface with onStart () and onResume (). The main difference
between these two methods of `onCreate()` is that `onCreate()` runs the process before displaying the screen to the user. That is, the changes are visible to the user, regardless of the view initialization at `onStart()` and `onResume()`.

The difference between `onStart()` and `onResume()` is that the user can manipulate the screen, such as entering text or pressing buttons. Conversely, there are always methods that do exactly what the above methods need, calling `onPause()`, `onStop()`, and `onDestroy()`.

Actions are screens, but they are always unique. The inconvenience of this problem is that you need to create your own action for each of the above screens. This increases the capacity of the code and results in poor maintenance. In this case, a more flexible component called Fragment is displayed in the application. Figure 16 clearly shows the relationship between activities and fragments. Fragments serve as a fabric that can be modified, added, or deleted from the host.
The snippet also has similar callback methods for actions such as `onCreate()`, `onStart()`, and `onResume()`. However, invoking the life cycle of each method also depends on the activity of the node. For example, Fragment's `onCreateView()` method is always called after calling `onCreate()` on the posted activity.

After all screens are installed, there are communication issues between them. Android has another component, called an intent, specifically designed to solve such problems. Figure 17 illustrates the capabilities of Intent. The intent can point the screen to another screen. In addition, you can also transfer the necessary data from the previous screen to the next screen.
Once you deploy components from Android, you need to deploy another powerful tool-Firebase. Firebase is a mobile and web application development platform that provides developers with a wide range of tools and services to help them develop high-quality applications, grow their user base, and generate more revenue.

There are several features such as push notification, crash report, analysis, A / B test, etc. However, the main feature used and introduced is the real-time database.

Real-time database is a very powerful tool that is a fully managed SQL document database for developing web and mobile applications. Designed to easily store and sync application data from around the world. Key features include, but are not limited to:

1. Powerful documents and collections
2. SDK and offline data access for iOS, Android, Web
3. Real-time data synchronization
5. SDK is Node, Python, Go, Java Server.

With eVote, the primary use of the real-time database is to create rooms for voters to participate in, count the number of votes for each option, and provide the results after the vote.

III. DEVELOPMENT AND QA
Because the application is based on Android. All design and implementation is done from Android Studio.

fig 21. Android Studio interface

Android Studio is a powerful Android application development tool. This provided an SDK and runtime for programming Java code. Currently, Android development mainly uses two languages, Java and Kotlin. For eVote, the application will be 100% Java.

Using Android Studio is more than just developing and running code. It also provides tools for debugging and performance analysis.
The UI design will be fulfilled in XML file.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#ffffff">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/frame_created"
        android:visibility="gone"/>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/layout_options"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/edt_options"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="How many options you need?" />
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/layout_persons"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp">

```

fig 22. XML file for layout design

When the code is ready, Android Studio has a built-in simulator, called an emulator, that lets developers check the behavior of their applications.
fig 23. Emulator User Interface
If an error or incorrect design is detected in the user interface, Android Studio has a layout inspector to help identify the problem.

For Firebase Real-time Database, the dashboard can help developer check what is going on with the data. Developer can also manipulate data from dashboard.
After some work on the process, development is in progress. The last thing needed for testing. There is a list of test cases covering small group voting. Each of them got results but is pending.

<table>
<thead>
<tr>
<th>n</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>p/ns</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>2</td>
<td>0/1/2</td>
<td>0/1/2</td>
<td>0/1/2</td>
<td>0/1/2</td>
<td>0/1/2</td>
</tr>
<tr>
<td>3</td>
<td>0/1/2/3</td>
<td>0/1/2/3</td>
<td>0/1/2/3</td>
<td>0/1/2/3</td>
<td>0/1/2/3</td>
</tr>
<tr>
<td>4</td>
<td>0/1/2/3/4</td>
<td>0/1/2/3/4</td>
<td>0/1/2/3/4</td>
<td>0/1/2/3/4</td>
<td>0/1/2/3/4</td>
</tr>
<tr>
<td>5</td>
<td>0/1/2/3/4/5</td>
<td>0/1/2/3/4/5</td>
<td>0/1/2/3/4/5</td>
<td>0/1/2/3/4/5</td>
<td>0/1/2/3/4/5</td>
</tr>
</tbody>
</table>

**IV. EXPERIMENT RESULT**

Actual testing is required to ensure that the application is working properly. Since there is no test equipment, use the emulator to get test results.

Everything was done based on the test design above. However, we are discussing too many cases and cannot introduce each one. Use the following script as an example.

First, create a room with 6 people and 4 options. Then, as expected, the fire base returns the key to the room:
fig 25. create room

fig 26. return room key
fig 27. make option 1

fig 28. make option 2
As it is shown in screen for the selections, I made option 1 for screen 1, option 2 for screen 2, option 0 for screen 3, option 2 for screen 4, option 3 for screen 5, option 2 for screen 6.

fig 29. Selection result 1

fig 30. Selection result 2
The results are obtained as desired. If everyone decides with one option 0, one option one, three options two, and one option three, the whole screen will be updated with the result.

V. CONCLUSION AND EXPECTATION

The eVote application is a useful and practical voting application, especially in today's world where smartphones are an irreplaceable personal property. The question of whether the concept of eVote is widely accepted is a problem, but it can certainly solve many voting problems.

The next step in this application is to introduce an iOS version for iPhone users. On the other hand, some ads may be added for small benefits. Need to enhance other voting features.
References:


[6]. K.W. Tracy, Mobile Application Development Experiences on Apple’s iOS and Android OS, IEEE Potentials, 31(4):30-34

[7]. Kocakoyun. S, Developing of Android Mobile Application Using Java and Eclipse: An Application, DOI: 10.17932

