SHOUTING PINWALL: A simple pinwall-app by team CENTRAL PERK

Ivo Maag
Dept. of Computer Science
Hanyang University
Seoul, Republic of Korea
maagivo1@students.zhaw.ch

Jing Yang
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
alumpof@hanyang.ac.kr

Daeyoung Jung
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
dyjungs@gmail.com

Eonwoo Yoo
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
dbdjsdn123@naver.com

***Abstract*—Online Bulletin Board for friends, bringing the idea of Analogue bulletin board to your mobile devices. This project is to create an Android app that uses Firebase as a backend. It displays a pin wall where everyone can post a text. Firebase will store the text in capital letters. The main purpose of this project is to learn how to use Firebase and how to run software on it, how to build an Android App with backend and how to plan, document and organize a project in a team. On SHOUTING PINWALL, users can post their messages in CAPS in form of virtual post-it memo (with a character limit), on a virtual bulletin board that can only be shared with his or her friends. Through this, friends can share their stories easily. This basic app has a lot of potential for further development in case we have leftover time. For example, encryption or different media types.**

***Index Terms*—SHOUTING PINWALL, Android, Firebase**

Table I
ROLE ASSIGNMENTS

Role	Name	Task description and etc.
User	Ivo	What do I need so the app is nice to use? Test the app and give feedback.
Customer	Jing	What do we need for our company to maximize productivity? Communicate with users as well as the development team. Responsible for cost and carries risk.
Software Developer	Daeyoung	Internal Worker. Tries to fulfill the costumers need in the code. Ideally with regular contact with Dev manager as well as costumer. Making incremental progress and shares it.
Development manager	Eonwoo	Has an overview over budget, costumer dev. progress etc. Is in direct contact with costumer as well as dev. He has a the big picture and should detect early when things go sideways.

I. INTRODUCTION

Social Media has already become a huge part of our lives and the ways we can connect to each other are so broad that we often feel things have gotten too complicated. So, we decided that we want to bring back simplicity and intuitiveness to the way we communicate. Related SW or Services: Twitter, Instagram, Facebook, etc.

II. REQUIREMENTS

A. Functional requirements

1) As a user, I want to be able to see what other users posted.
2) As a user, I want to be able post a text on the pinwall.
3) As a user, I want to see an error message when sending.
4) As a user, I want to see the same content, no matter which device I use.
5) As a user I want to be sure my posts are saved, even if I lost my phone.
6) As a user I want feedback after posting to ensure it was successful.
7) As a user, when the voting phase for ‘reset’ starts I want to receive notification asking if I agree to reset the board.
8) As a user, I want to receive a notification when the board is successfully cleared.
9) As a user, I want to choose the color of the memo I am going to post.
10) As a user, I want to delete the post I have uploaded when I want to.
11) As a user, I want to set a time limit on the post I upload. When the limit expires, the post is automatically deleted.
12) As a developer, I want to add further functionalities if necessary.
13) As a developer, I want tests to ensure the functionality is still provided after I changed the software.

B. General requirements

1) **Performance**: Posts should be saved and loaded within half a second after initializing the process.

2) **Scalability**: The app should be able to handle up to 100 users wile staying in the set performance threshold.

3) **Responsiveness**: The app should adapt to screen sizes from 4-7 inches.

4) **Usability**: The functionality should be self-explanatory and not require any instruction to use it.

5) **Reliability**: The app should confirm visually if a task was done successfully.

6) **Security**: No security measures are planned at this point. Encryption is due to further development.

7) **Availability**: The app should be available 90% of the time, since this application is not critical.

8) **Adoption to slow/no networks**: The app should display cached data if there is no connection.

III. Development environment

A. App development

1) **IDE**: Android Studio
   We decided to use Android Studio for development, because it's the official development tool from google and it's the only way to program native Android apps. Also all the official Documentation is made for Android Studio.

2) **Programming language**: Kotlin / Java
   Kotlin is the new preferred programming language from Google. Most of the official documentation is in Kotlin. The programming language feels more modern to us than java. It compiles to Java bitecode and can even be mixed with java code in case we will need that. It gives us the opportunity to get more experience on a programming language that will most likely be very relevant in the future.

3) **Development OS**: Ubuntu / Windows / MacOS
   Android Studio is officially supported for all three mentioned operating systems. We actually have all of those in use and work together cross-platform. So far this didn't cause any problems. We use MacOS 10.15, Windows 10 and Ubuntu 20.04.

B. Backend

1) **Firebase**:
   We see it as an opportunity to learn how to use a cloud environment. Our plan is to use Firebase for our backend. This way we don't have to worry about a physical hardware infrastructure and are still able to provide a server based service.

C. Collaboration

1) **Github**: Version Control and Collaboration of code. We use Sublime Merge as a Github desktop client.

2) **Kakao Talk**: Messenger to communicate among us.

3) **Overleaf**: It's a cloud based Latex-editor, so we can work together on the same document simultaneously.

D. Cost Estimation

Software

1) Android Studio: Free
2) Kakao Talk: Free
3) Overleaf student subscription: 8 USD / Month
4) Firebase: Free

Hardware

1) Our personal computers: Around 5'000'000 Won. But we need them anyways for school, so we do not calculate them in.

Working hours

1) Around five 8-hour working days per person. So around 20 working days in total. At a hypothetical 60'000 krw/hour total would be 9.6 Million Won.

Total The total would be 9.6 Million + 3 * 9k (Overleaf * 4 Months) = **9.627 Million Won** At this price we would probably not be very competitive in the market.

IV. Specifications

A. Mobile Application(Android)

1) Main Screen
   a) View Post
      i) By clicking the post on the list, it will move to another screen where it displays content of the post.
   b) Adding New Post
      i) "+" icon on the bottom of the screen will let the user access the post-editing page. The screen will switch to new post editing page so the user can write what he/she wants to upload.
   c) Refresh the post list
      i) By pulling the list down, the list refreshes and fetches new content (if there is any) from the server.
   d) Display Posts made by others and yours
      i) By starting the app, the main screen will display posts uploaded by others and yours in timely order. (Older posts are beneath and later ones are above)

By using RecyclerView on the layout, the posts will be loaded from the firebase database and displays them in chronological order.

2) Add Post Screen
   a) Editing title of the Post
      i) On the title text box, it will display hint text as 'add title' so user can tap(or click) the designated text box to edit the title of his/her post.
   b) Editing content of the Post
      i) On the content text box, it will display hint text as 'add content' so user can tap(or click) the designated text box to edit the title of this/her post.
   c) ADD Button
      i) Uploads the title and content to firebase database and closes the edit page. Automatically goes back to main page
   d) CANCEL Button
      i) Deletes every change you made on title and content and closes the edit page, goes back to Main screen.
   e) Choose Color of Memos
      i) By toggling button, the user can choose which color the post will be.
3) Alerts
   a) Post Upload Successful
      i) If the new post is successfully uploaded to the database, an alert message will appear for brief seconds saying "Upload Successful!"
   b) Post Upload Failed
      i) If the content of the post fails to reach the database, an alert message will appear for brief seconds saying "Failed to Upload!"
   c) Post Deleted
      i) If user successfully deletes his/her post, an alert message will appear for brief seconds saying "Post Deleted!"

B. Save Post

As a user I want to be sure my posts are saved. Not in the storage of my device but in online DB or server.

To achieve this we use cloud server services such as amazon cloud services. Whenever a user finishes editing and clicks the 'upload' button, it saves the written content to designated server, so it can be accessed later on by others as well.

C. Feedback after posting

As a user I want feedback after posting to ensure it was successful.

When the post is successfully uploaded, the app will display a message that it has been successfully uploaded. Or we simply update the view after storing it. This will give the user visual verification that it worked. If storing fails we will need to place an error message.

D. Choose Color of Memos

As a user, I want to choose the color of the memo I am going to post.

```
if User wants to choose the color of the
    memo
         print"What color of the memo do
             you prefer?"
         if color = xxx
                  print memo in xxx
     print"Do you want to change the color
         ?"
     if user click the "Yes" button{
         repeat print"What color of the
             memo do you prefer?"
                  if color = xxx
                  print memo in xxx
     }
     else
         exit the program
```

E. Delete Posts

As a user, I want to delete the post I have uploaded when I want to.

If we manage to implement this feature, we need to make every post identifiable and we need a http request for the backend to delete a post.

```
if User wants to delete the post
         DeleteThePost()
    if post successfully deleted{
                  print"Deleted."
         }
         else{
                  print"Error: Delete
                      failed."
         }
```

F. Add Further Functionalities

As a developer, I want to add further functionalities if necessary.

We try to achieve this with a modular build and by taking care about coding standards. Code fragments that are used in multiple screens should be exported into a separate class.

G. Tests

As a developer, I want tests to ensure the functionality is still provided after I changed the software.

This can be done by unit tests. However, based on experience testing the UI doesn't make a lot of sence, so we should

focus on functionality tests such as the following example that is thesting the validity of an e-mail address.

```
if software changed
        TesttheFunctionality()
        if the functionality doesn't
           provided
                print("Something wrong
                   with the functionality
                   . Rewrite the code!")
        else
                exit TesttheFunctionality
                   ()
```

H. Performance

Posts should be saved and loaded within half a second after initializing the process.

Cached content should be displayed first, so if the connection is slow it doesn't wait for connection until it displays something. To display the list itself in under half a second should not be a problem.

```
loadCachedData()
updateView()
loadFromNetwork()
updateView()
```

I. Scalability

The app should be able to handle up to 100users wile staying in the set performance threshold.

This should not be a problem on the frontend but maybe on the backend. Using AWS for our backend should solve this problem already since we can add resources if needed. If we get to the limit, we could assist by loading only the necessary data, for example only newly added post and not the whole list.

```
loadNewPosts(int postNr) {
    for post in posts {
        if post.nr >= postNr {
            sendPost()
        }
    }
}
```

J. Cross-device compatibility

The app should adapt to screen sizes from 4-7 inches.

This can be achieved with constraint-layouts in Android Studio which allows to place items in relation to each other.

K. Usability

The functionality should be self-explanatory and not require any instruction to use it.

This can be achieved by naming buttons in a way that makes sense and place them in a natural way. Following Android guidelines also helps, because users are also used to the gestures from other apps. Also getting user feedback can help to achieve this.

L. Reliability

The app should confirm visually if a task was done successfully

The best way to do this is probably not with pop-ups, but by displaying some kind of loading animation and updating the view after completion.

```
postContent() --> asynchronous
displayLoading()
callback --> updateView()
```

M. Security

No security measures are planned at this point.Encryption is due to further development.

We decided because security is not the simplest thing to implement and would be beyond the scope of this project. But in a real-life productive application, specially if it involves personal data, proper encryption is a must.

N. Avialability

The app should be available 90% of the time, since this application is not critical.

No implementation necessary. Uptime can be calculated as follows:

```
Time of running service / Total aimed
   operation time.
```

O. Adoption to slow/no networks

The app should display cached data if there is no connection.

Also minimize network traffic can be helpful. The following article explains well how to achieve this in an android app. https://developer.android.com/training/efficient-downloads/redundant_redundant

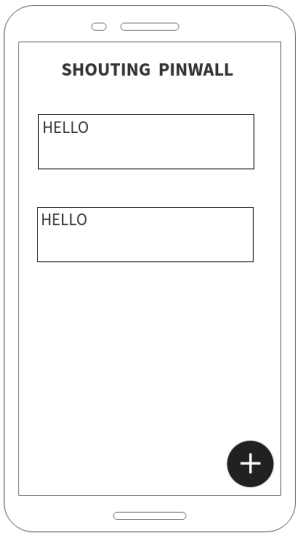
```
loadContent(){
    checkIfLoadingIsNecessary()
    if yes {
        loadFromNetwork()
    } else {
```

```
            loadCache()
        }
        updateView()
}
```

V. Screens

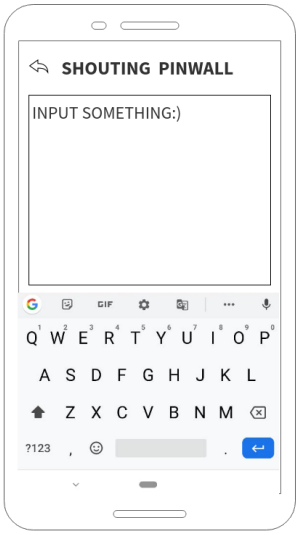
A. Home Screen

The Home Screen displays our app name on the top. Under it is a Listview with all the posts loaded from the backend. On the bottom is a plus sign that can be used to create a new post.



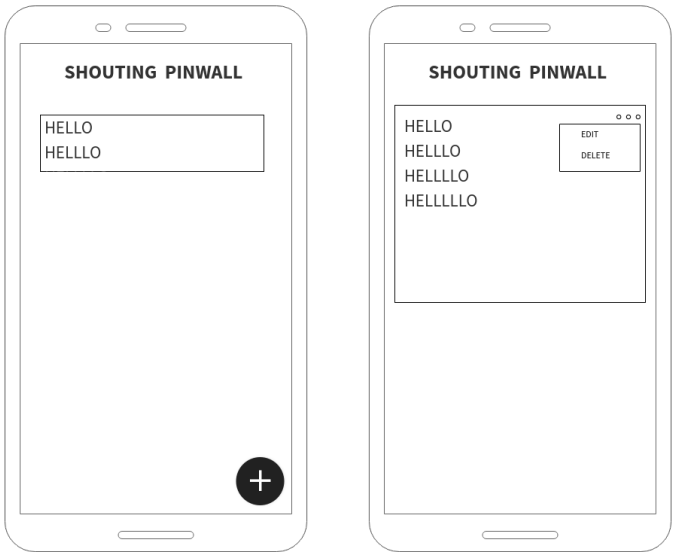
B. Add Post Screen

This screen shows a button to return to the home screen alongside the app name on the top. Under it is an input field to type a post. When clicking on it, the keyboard opens and with enter-button the post can be sent.



C. Add Post Screen

When clicking on a post on the home screen, this view opens. It shows the app name and the post in a bigger view. In the future here could be added more details such as pictures.



VI. Backend API

A. Configuration

API Key: 9xx5MAxxbO61HW2EJNFEN1iGidfAAlAucKPUt8M0