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.

Link to our Repository: Github Repository

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!"
   d) Delete Unsuccessful
      i) If the app fails to delete the post in whatever reason or error, an alert message will appear for brief seconds saying "Delete Unsuccessful."

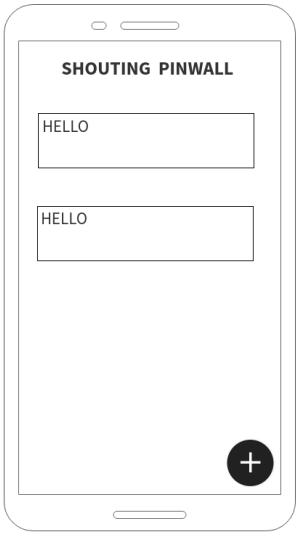
*B. **Server***

1) Mobile Application
   a) Application connects and loads post data from database on start up. (onCreate phase)
   b) Based on 'timestamp' data, the application displays posts on the main screen in chronological order.
   c) In case when app fails to fetch data from server, it generates message saying "Error getting documents: *Exception*"
2) Firebase Server
   a) In the server's database, the data of the uploaded post will be saved as document under 'post' collection.
   b) The post collection saves uploaded post data package by tagging randomly generated ID.
   c) Inside the tagged document, it includes data of 'text', 'timestamp' and 'title'. The title and content data from the user is saved under these categories including timestamp data at the moment of upload.

V. UI/UX

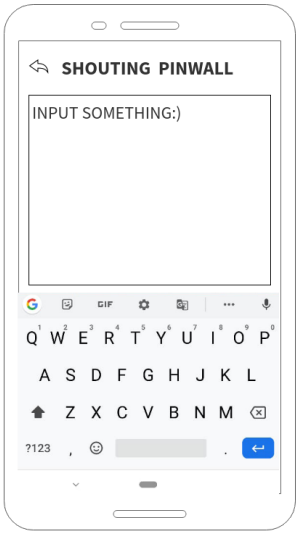
A. Main Screen

The Main Screen displays our app name on the top. Under it is a listview with all the posts loaded from the server. On the bottom right is a plus button that triggers intent to new post editing screen on click.



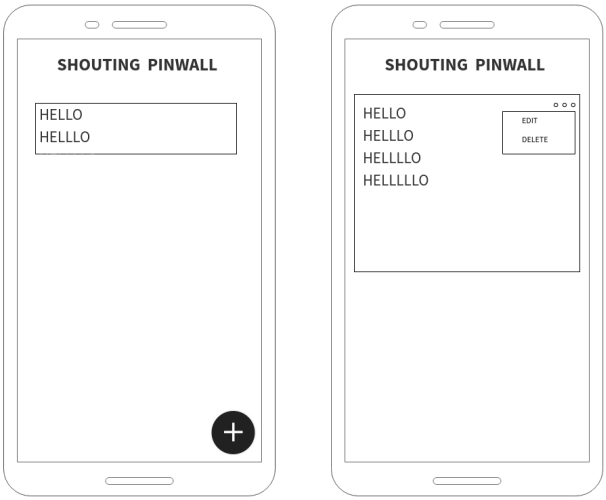
B. Add Post Screen

This screen is for editing new post and upload it. On the top left it has text box for the post's title, and beneath it there is also a text box for the content. On the bottom, there are 'CANCEL' and 'ADD' button. 'CANCEL' button discards everything written above and brings back to Main screen. 'ADD' button completes post editing and uploads post data to the server.



C. View Post Screen

When clicking on a post on the Main screen, this screen opens. It shows the title at the top and content below in a bigger view. In the future here could be added more details such as pictures.



VI. Architecture Design

A. Overall System Architecture

SHOUTING-PINWALL is set together with two modules. The frontend is an Android App and provides access for users and communicates with the back-end through the Firebase API from Google including encryption of the data. It's responsability is to make sure the user has easy and reliable access to the application and also handling the data.

The back-end stores the data it gets from the application. Firebase is a part of the Google Cloud plattform, comparable to amazons AWS. Android and Firebase are both developed by google, are very well compatible and even provide integrated encryption. We tried to get a back-end with amazon AWS to work for quite a while, but it seems to be impossible with a student account and it's restrictions and we had to give up. The Google cloud service also allows integration of analytics, functions, IAM, etc., which would be possible exteionsions for our project.

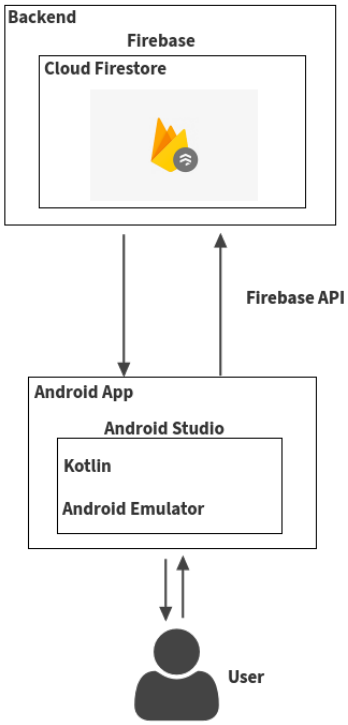


Figure VI-A.1. Overall System Architecture

B. Directory Organization

Table II
DIRECTORY ORGANIZATION

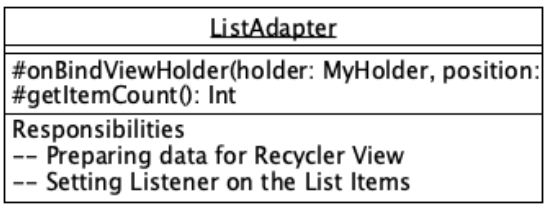
Directory	File Name	Module Name
Pinwall/app/main	AddPostActivity.kt ListAdapater.kt MainActivity.kt Post.kt PostComperator.kt	Android App
Pinwall/app/res/drawable	add.png men.png women.png	Android App
Pinwall/app/res/layout	activity_add_post.xml activity_main.xml list_item.xml	Android App
Pinwall/app/res/mipmap-hdpi	ic_launcher.png ic_launcher_round.png	Android App
Pinwall/app/res/mipmap-mdpi	ic_launcher.png ic_launcher_round.png	Android App
Pinwall/app/res/mipmap-xhdpi	ic_launcher.png ic_launcher_round.png	Android App
Pinwall/app/res/mipmap-xxhdpi	ic_launcher.png ic_launcher_round.png	Android App
Pinwall/app/res/mipmap-xxxhdi	ic_launcher.png ic_launcher_round.png	Android App
Pinwall/app/res/values	colors.xml strings.xml themes.xml	Android App
Pinwall/app/res/values-night	themes.xml	Android App
Pinwall/doc	SHOUTING PINWALL.tex	Documentation

VII. CLASSES

A. AddPostActivity.kt

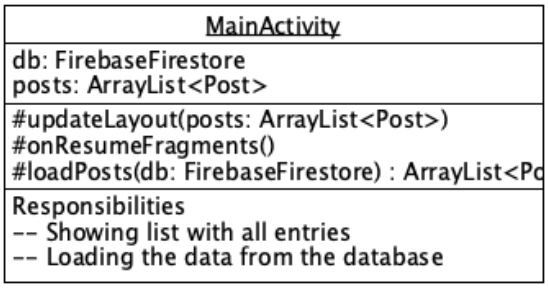
1) The var 'db' is responsible for handling database and provides function such as storing and loading data from firebase server.
2) addBttn in this class triggers action of uploading title and text data to database 'db.'

B. ListAdapter.kt



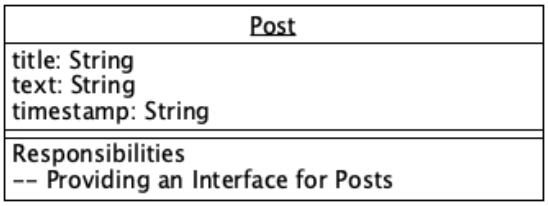
1) This class includes controls how list of posts are displayed on main screen, by using recyclerview. It fetches 'title' and 'text' data and displays as "listitem.xml" design.

C. MainActivity.kt



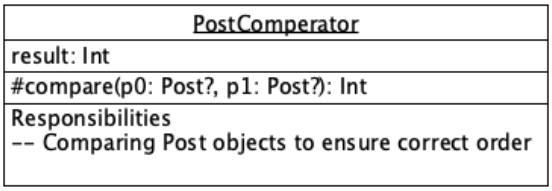
1) posts
   a) this is array of data from Post.kt which includes 'title', 'text' and 'timestamp' data.
2) updateLayout()
   a) This function loads latest data from the server on starting the app.
3) loadPosts()
   a) This function is responsible for loading post data from the server.
4) onResumeFragments()
   a) This function allows the app to automatically refresh after returning from other intent such as 'AddPostActivity'

D. Post.kt



1) title
   a) Contains String data of what user wrote on 'aptitleBox' textbox.
2) text
   a) Contains String data of what user wrote on 'aptextBox' textbox.
3) timestamp
   a) Contains String data of timestamp when user uploads his/her post on the server.

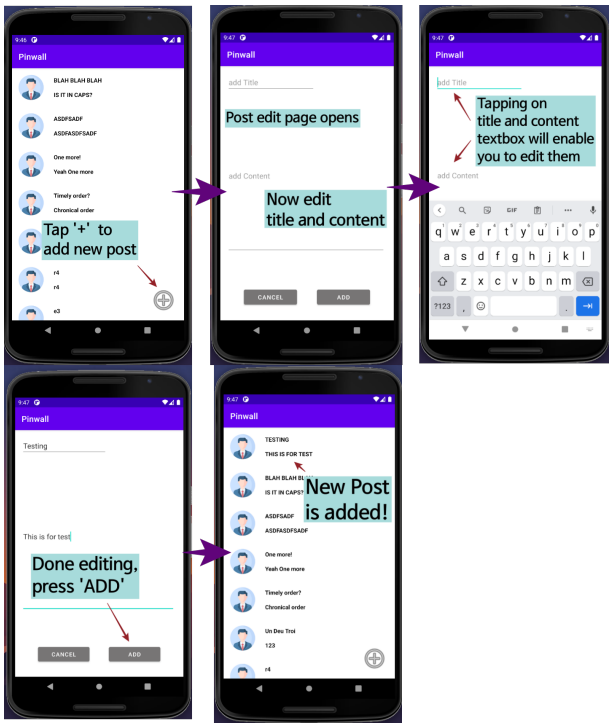
E. PostComperator.kt



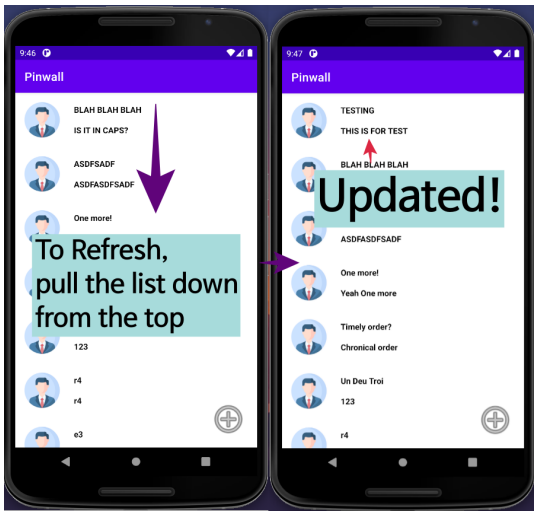
1) By default Posts are sorted by the ID, which is a randomly generated value by firebase. As a result the posts are sorted in a random order. To fix this and order the items by the time added, we added a timestamp to the posts and created this comperator.

VIII. Use Cases

A. Adding Post



B. Updating Post list



C. Discard New post