|  |
| --- |
| **Software Engineering**  **Term Project final Report**  **(Doodle)** |



|  |  |
| --- | --- |
| **Due Date: 2019. 06.04** | **Professor: Ok Ran Jung** |
| **Lecture: Software Engineering** | **Student: 권수환, 이석재, 이승수** |

**Contents**

1. **Introduction -**

**1.1 Purpose of the System**

**1.2 Scope of the System**

**1.3 Objectives of the Project**

1. **Current Systems(Related works)**
2. **Proposed System**

**3.1 Overview**

**3.2 Requirements**

**Requirement elicitation**

**Requirement specification**

**Requirement validation**

**Requirement change**

**3.3 System Modeling**

**a)Structural model**

**b)Behavioral model**

**1)Data-driven model**

**2)Event-driven model**

**3.4 Architectural Design**

**a)Layered Architecture**

**b)Client-Server**

**c)MVC model**

1. **Implementation**
2. **Testing and Evaluation**

**Development testing**

**Used SE tools & testing tools**

**Release testing**

**Testing considering requirements, user scenario, performance**

**User testing**

**Alpha&Beta testing, acceptance testing**

1. **Conclusion**
2. **Glossary**

**1.Introduction**

**1.1 Purpose of the System**

We thought many interesting ideas on the road, when we are going somewhere. But it is unfortunate that we easily forgot it and it’s hard to reminding. So our purpose of application is to elaborate the striking ideas with another people to record our sudden idea and makes it better.

**1.2 Scope of the System**

We thought there is the way to express and sharing idea is to drawing and writing. Of course there are recording video too, but it’s too heavy. So we thought drawing function will be proper way. The system will include drawing function, login process to certificate users, room management, related database and networking. Our core part will be networking in real-time and relayed drawing part.

* 1. **Objectives of the Project**

Our system have no particular target objectives but common people will be users. But our application have networking and ‘like’ system, so it can make a side effect similar to social network service too. And considering smartphone distribution rate for seniors, our main targeted users will be ‘common peoples who want to share their bright idea with another people’ and young users will be a lot.

**2.Current Systems(Related works)**

We investigated some related applications about relaying or sharing idea on GooglePlay store. We thought two relayed applications, ‘Relay Novel2’ and ‘Pinterest’.

Pinterest is a famous application for designers, which is to get design ideas. It provide just the display the list of images related to keyword. It can be used to get inspiration of new design from examples. Pinterest have similarity with us at helping discovering new Idea.[[1]](#footnote-1)

Relay Novel 2 is an application to relaying the novel with other people. It’s main idea can be similar to our’s but there is difference that the application only focused on writing novel in text. [[2]](#footnote-2)

Our system will be combination of two application’s characteristics and add drawing and social networking functions. We will accept Pinterest’s GridLayout display for high visibility and expand basic concept of Relay Novel2.

**3.Proposed System**

**3.1 Overview**

process of software development goes, the cost to change is go higher. In particular, this change is very difficult in the Android structure where each screen and function is divided into objective. We have been able to understand the overall structure and the necessary functions to communicate with the structure, the requirements of the app, the system model, and so on by following procedure.

**<Requirement elicitation >**

Following requirement engineering process, firstly, we use **interviewing** for requirement elicitation with students who made similar application at last year Mobile Programming lecture. They made real-time drawing tool last year. For requirement discovery, we asked some questions for what they think most challenging part of our project and how does user and people’s reaction at their application. So we got that we have to think a lot for real-time connection and visualizing drawing tool. And they adviced us that we have to consider a lot more than we thought because many users are more sensitive to UI design than we think.

As a result, we can summarize interview’s point to **real-time networking** and **user-friendly UI**.

**<Requirement specification>**

**User requirements**

* Want to write their ideas with others
* Want others will evaluate my ideas
* Want to create a project with the users I want.
* Want to be able to express my thoughts in various ways, such as writing and painting.
* Want to be able to show you my ideas and a list of the projects I have participated in.
* Want keep my ideas private so that only a limited number of people can see and participate

**3. System requirements**

**Functional requirement**

* Login Function
* Make pictures and notes in relay
* Save note and picture in DB to share
* Evaluate saved pictures or notes
* Edit friend lists
* Keep personal informations in private.

**Non-functional requirement**

* User’s real-time display connection delay have to be less than 2 seconds.
* Keep processing communication even if some participants is lost
* Intuitive for immediate using tools without reading the instructions.
* Keep simple UI interfaces to fit in small mobile phone display.
* Should have portability for executed in multiple heterogeneous device environments.
* Login process: sign up should be quickly. -use google sign up
* Main Menu: show most voted ideas on the top of list.
* Painting Tools: fast reaction, support various painting tool settings.

**<Requirement validation>**

Requirements validation is a process which is concerned with demonstrating that the requirements define the system that the customer really wants. **Requirements error costs are high** so validation is very important. We used requirements validation techniques called requirements reviews.

**Checking point**

**Validity**

The customer has two functions (paint tool, smooth network communication). And you can create a system that meets these requirements.

**Consistency**

Each function will be implemented independently, so there will be no conflicts in functionality. However, network state conflicts can occur.

**Completeness**

Functionality required by the customer Two of the painting tools required a minimum of eight colors and three or more writing instruments. All of these requirements can be implemented in the system.

**Realism**

Rather than creating a new API or library, most functionality can be implemented through Android studio, Java, and built-in libraries.

**Verifiability**

All requirements can be verified.

And we made the following conclusions about the four elements of the review check.

**Verifiability**

Our program is an Android application. Therefore, you can test using various testing tools built into the Android studio.

**Comprehensibility**

We periodically check with our customers for requirements.

**Traceability**

We store meetings and records of customers and requirements on a date-by-date basis. We manage version-specific and improvement points by version.

**Adaptability**

From the perspective of functional implementation of an Android app, you can add features that are relatively flexible if you do not require a completely different application.

**<Requirement change>**

The planning tool in the planning phase for the initial app requirements was not able to enter other users when multiple users were typing. With a similar feeling to the turn game, when user A was typing, user B was unable to use the drawing function until user A completed the input. Also, it is a structure that user A sends a signal that the input is finished and reflects the screen of other users while refreshed, not sharing the input screen. This is incomplete while satisfying the function requested by the customer.

So the customer wanted to be able to simultaneously input multiple users simultaneously in real time, and to be able to simultaneously launch the image changes during the input on other users' screens.

The planning tool in the planning phase for the initial app requirements was not able to enter other users when multiple users were typing. With a similar feeling to the turn game, when user A was typing, user B was unable to use the drawing function until user A completed the input. Also, it is a structure that user A sends a signal that the input is finished and reflects the screen of other users while refreshed, not sharing the input screen. This is incomplete while satisfying the function requested by the customer.

So the customer wanted to be able to simultaneously input multiple users simultaneously in real time, and to be able to simultaneously launch the image changes during the input on other users' screens.

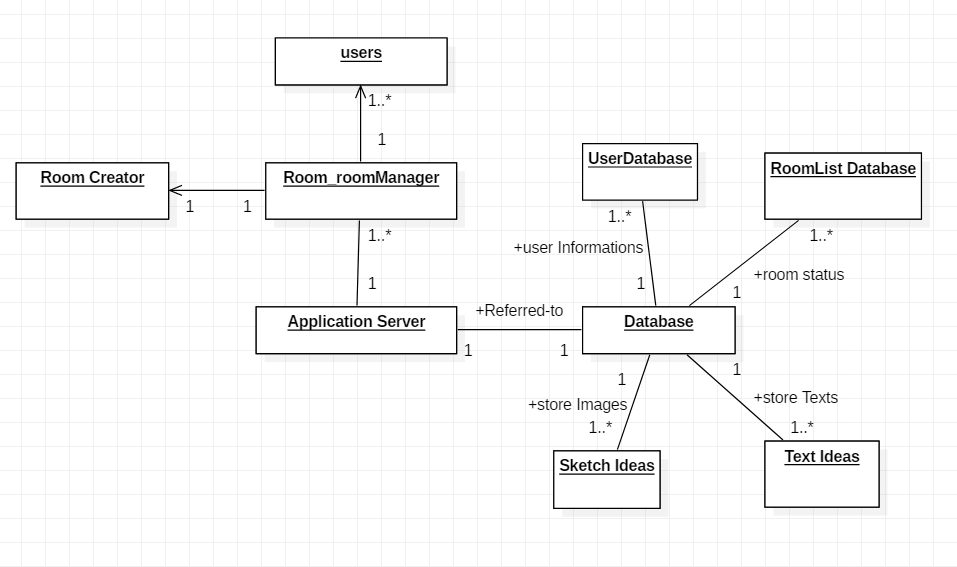
We did a preliminary investigation to see if the customer's requirements were technically feasible. In addition, the real-time screen refresh was interlocked with the FireBase DB to update the DB for a short time, and the data was transmitted to all users, satisfying the requirements of the repeated user as long as the network allowed.

* 1. **System Modeling**

1. **Structural Model**

We selected Structural model to express our objects’ structure and relationship.

**<class and object relation in Application with class diagram>**



1. **Behavioral Model**

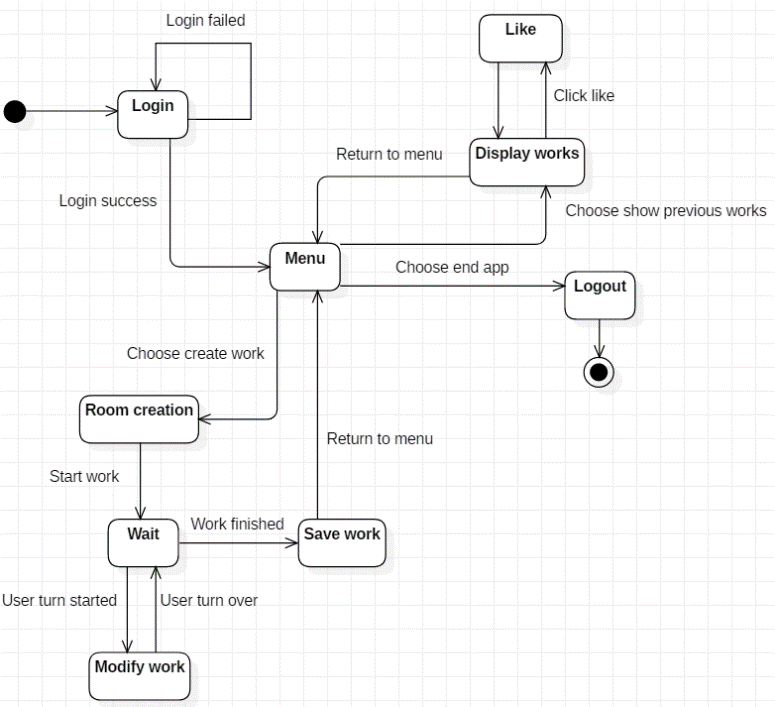
Our real-time networking based on Firebase DB, so it has a lot of transaction in application. So we decided behavioral model is best to express data flow.

**<data-driven modeling>**

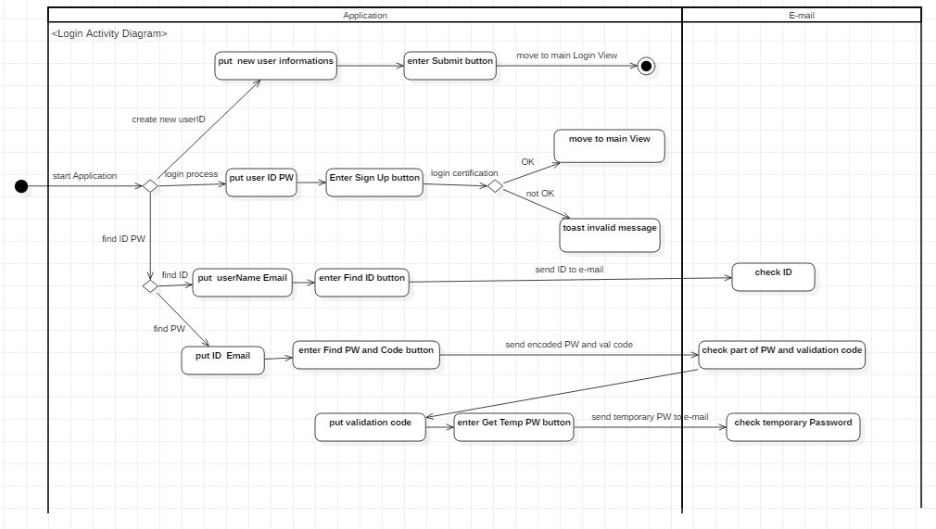
We use activity diagram and sequence diagram to express major processes on application.

**<Total process in activity diagram>**

This activity diagram express our application’s main functions approximately.

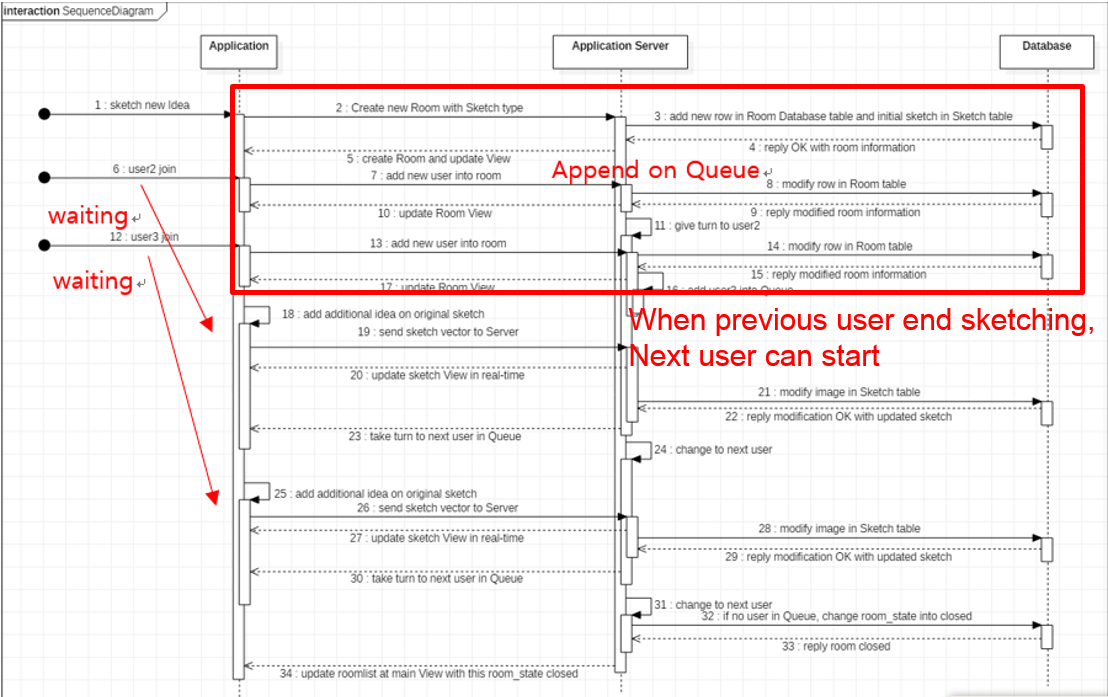


**<Login process in activity diagram>**



At login process, user have 3 choices, sign in/sign up/find ID/PW. When sign in and sign up, user just have to do it all in application but find ID/PW, user have to check e-mail together.

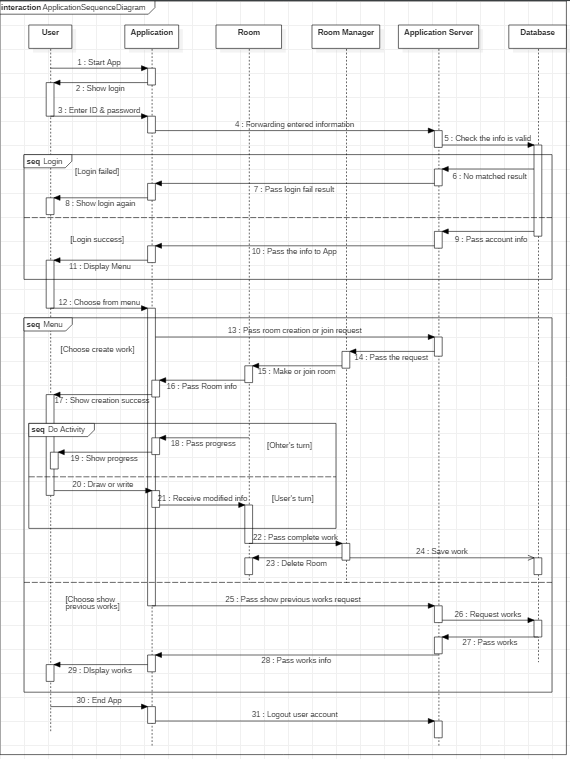
**<Sketchroom process in sequence diagram>**



This sequence diagram express the data movement between user-application-server-database in simple idea room case which using image data. Idea initiator (user 1) initiate new idea and then user2,3 join to room and modify the idea with some additions. Then modified idea stored to database and if room is empty, room is closed.

**<Event-driven model>**

**<Event driven model of total Application in sequence model>**



This sequence diagram express our application’s main functions in detail with event-driven perspective. The diagram will be the simple life cycle of our user’s application usage.

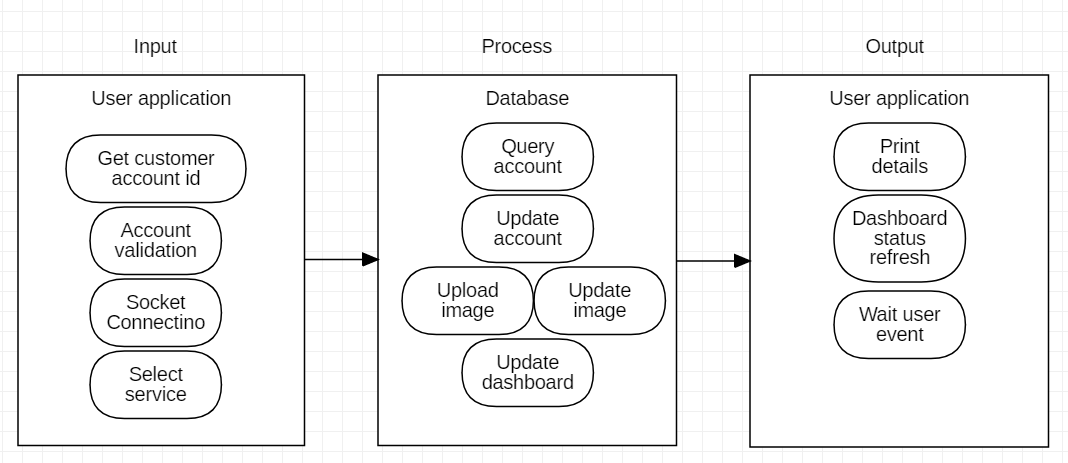
**3.4 Architectural Design**

For architectural design, we consider of application type and architecture pattern .Our application is **transaction processing application** type, which occurs a lot of transactions between users.

Architectural Pattern can be abstracted in two ways**.**

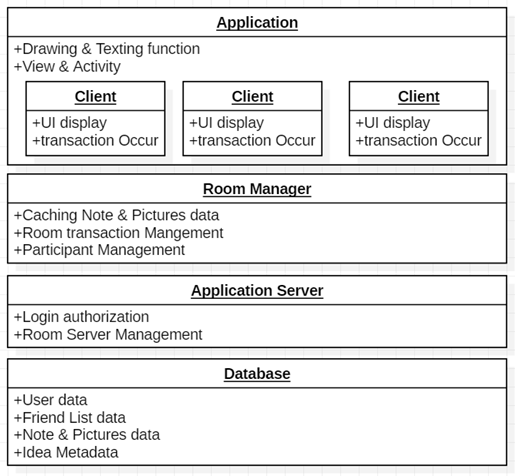
**At large view**, application can be **client-server architecture pattern** designed for real-time connection.

**a. client-server architecture**



This client-server architecture model express interactions between users’ text and image data update. at each user’s event input-process-output cycle is end, user’s View and Database will be updated.

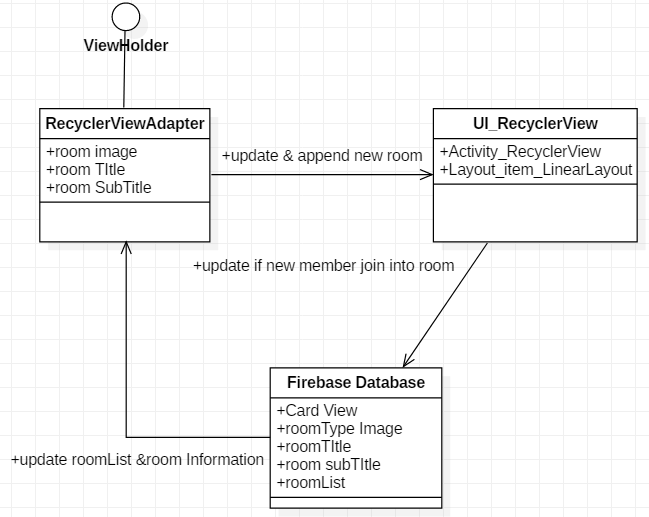
**At small view**, we accept **layered architecture** for encapsulating user’s information and idea datas and use **Model-View-Controller pattern** to build appendable list at MainMenu .

**b. layered architecture**

Client login with application and Application Server authorize user login data from database. After login, User will create or join into the room and show their idea. List of Rooms will be managed by Application Server and each room will be managed by each Room Manager. For accessibility, each Room Managers will cache the text or picture about the idea. After idea modification, Database’s idea metadata and pictures will be updated.

**c. MVC pattern**

. At dynamically allocating room items in room list view, RecyclerView(controller) get data from Firebase DB(model) and update it.



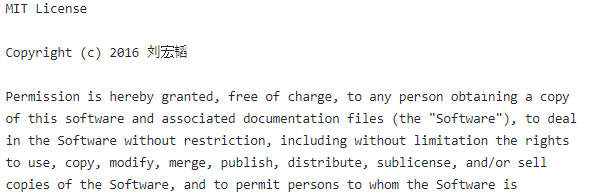
**4.Implementation**

We use Android Studio and developing tool and Firebase database using Java and Android.

At implementing Login stage, we had considered to build or re-use open source code. But it’s hard to find out almost similar LGPL or BSD licensed open source example, so we thought reuse cost of time will be bigger than build it. So decided to partly use open source and build by self rest of them.

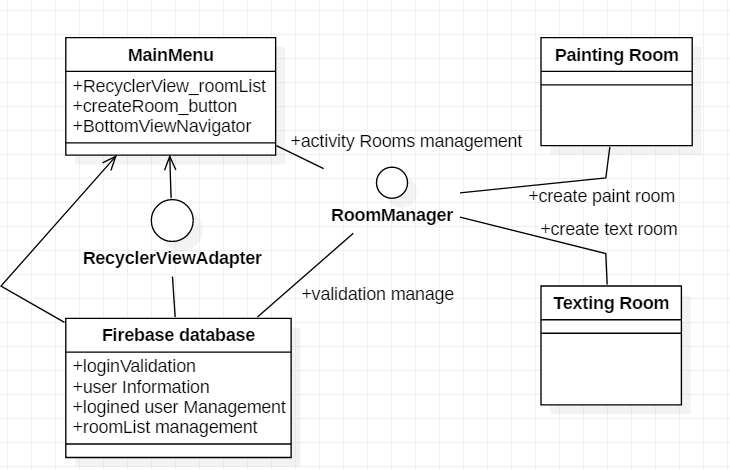
We reuse open source login component in **abstract level**. We got similar painting tool open-source project, which can draw on android map. We reuse at **component level** to building it, references some parts at just drawing with android. At building other components, we build by own with members. [[3]](#footnote-3)[[4]](#footnote-4)

login opensource is at Apache License and painting open source is at MIT License. They both allowed us to modify and distribution, so we accepted them.

** <login opensource license> <painting opensource license>**



At designing our MainMenu, firstly we thought users will want to quick login for immediate launching their idea so we make quick login function to go to MainMenu. Then we design simple mainmenu’s architecture and objects. Firebase support much more varieties of functions than we thought so we can do less efforts at managements. At first we identified many views and controllers but we can reduce many of them into as below.

**<abstract system architecture of roomlist management with Firebase at MainMenu >**

While building application, 권수환 student design back-end related jobs like managing Firebase queries and connecting applications to database and 이승수 student design front-end side UI at application. Because two people works together, we need to match our application’s sync together so we integrate our system using repeatedly pull/request at GitHub. It additionally help us to remaining our applications in variety of versions.[[5]](#footnote-5)

After building demo version of application, we have to check application will works at target user’s device too. So we perform alpha testing at many android devices and we found at galaxy S3 and A7, display isn’t fit well. So we modify layout’s size and alignment differences for versions for those devices. We will elaborate more about alpha-beta testing at following testing section.

Then we release our modified first demo version of application at **GooglePlay** market as **‘Doodle’** name. We have some tasks to do and we will develop below additional functions for software evolutions in repeating evolution step.

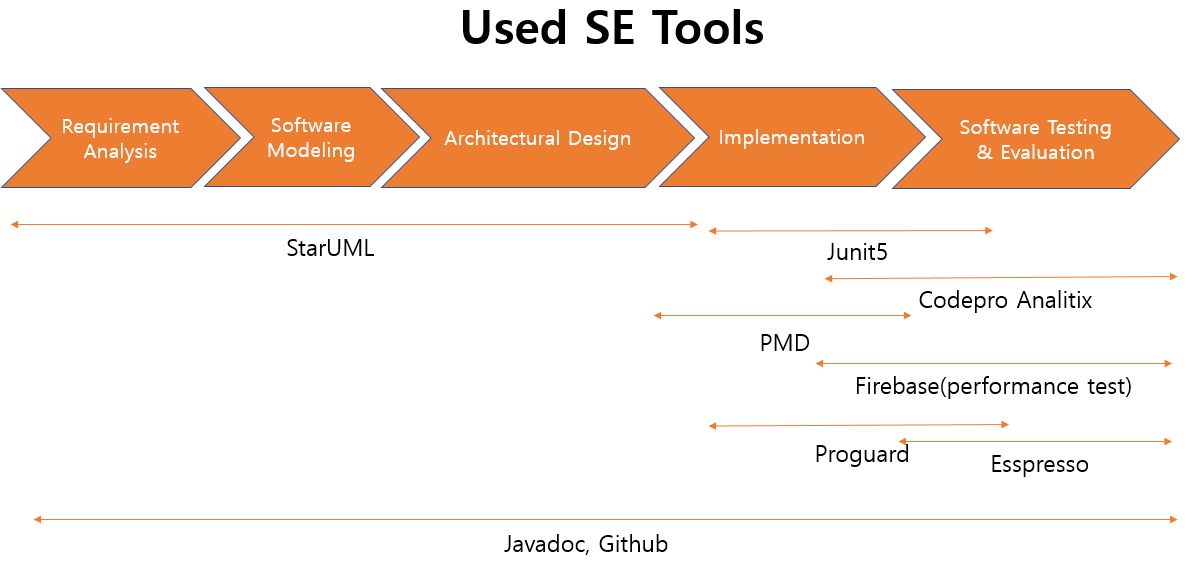
**Todo list for us to do in evolution step**

* **Add social networking function**
* **Setting room title and type function**
* **Public/private Authority for Room and Idea**
* **Trying to minimize transactions at application**

**5.Testing and Evaluation**

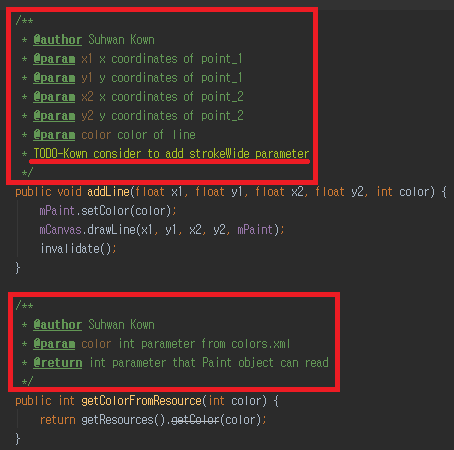
**5.1. flow chart of used SE tools**

A variety of SE tools can be used for each stage of software development and testing. We used the following program for each step. we wrote about tools other than the starUML that used and explained in the modeling and structure.



**5.2. Javadocs**

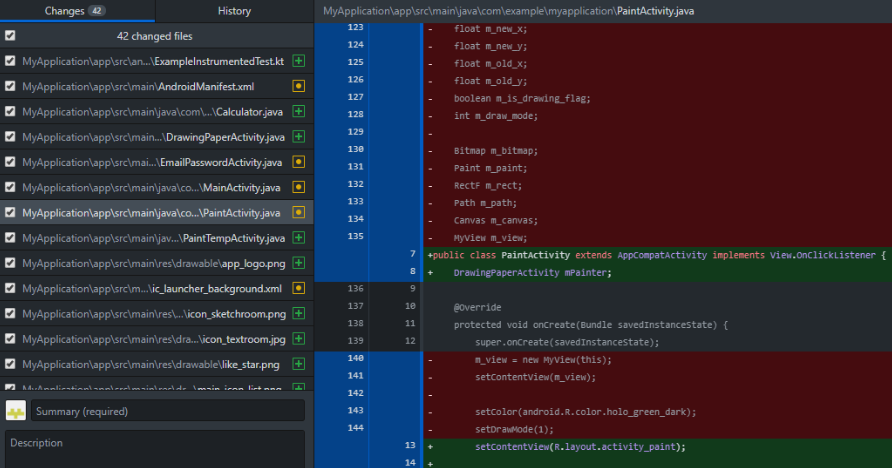
Javadocs is in the default Android studio that annotates and shows the organic connections and changes of each method and component, and communicates with other developers by helping them know what is needed more. As shown in the picture below, we made a distinction between the developer and the version of the function, the role of the parameter and the contents of the return value, the remain work through the TODO list so that the necessary contents can be efficiently shared.



**<Javadoc documentation at unit code>**

**5.3. GitHub**

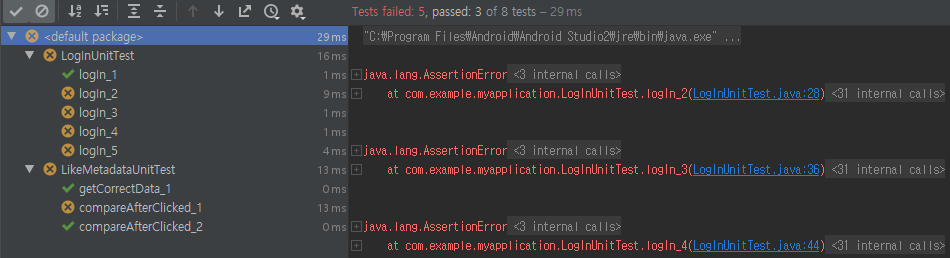
Since it is difficult to understand all of the development process such as layout, Java activity, and DB, it is sometimes difficult to develop compatibility and even duplicate content during development can happened. GitHub enabled us to see the changes that occurred during each of these developments, as well as the starUML and Javadocs.



**<version control and system integration functions at GitHub desktop>**

**5.4. Junit**

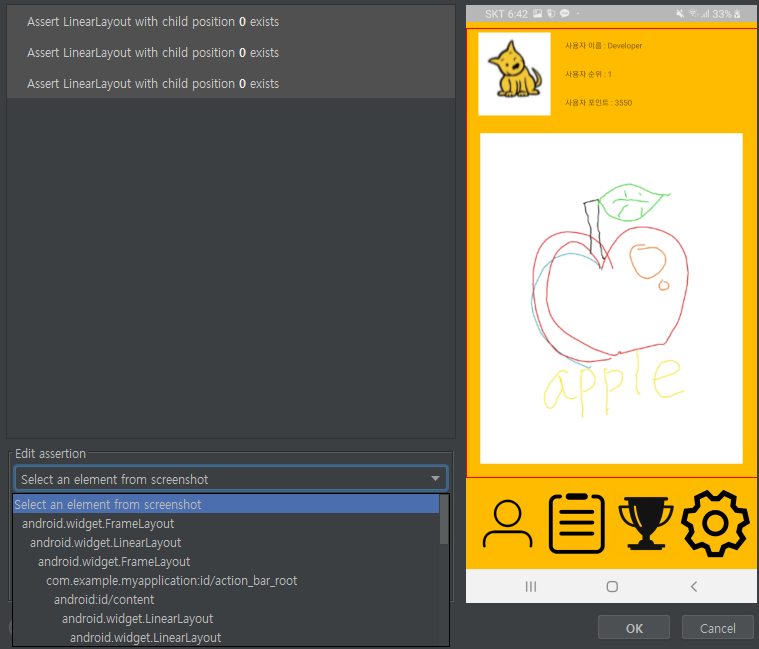
Junit is also included as part of the Android studio and is a tool that **automatically tests each unit**, method, without building an application. We use Junit which provides high time efficiency to check the validity of characters such as emails and passwords to be used when try logging in. We also use to check metadata of the created idea such as title or author. We also used it to confirm the validity of the changing metadata like the recommendation number.



**<Junit’s automated unit testing function>**

**5.5. Espresso**

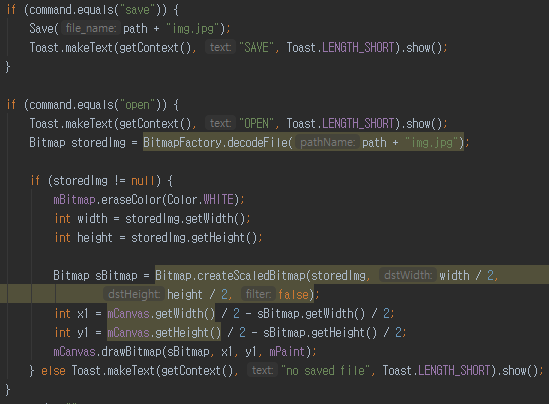
As a basic tool in the Android studio, It has similar role with Ranorex recorder that used to validate interfaces that ultimately interact with the user, such as capturing or recording each screen and analyzing the layout that makes up the screen. After the software was developed, it was used to check the final appearance.



**<Esspresso UI testing tool>**

**5.6. Proguard**

In order to modify basic functions, we developed a centralized method that puts all functions into one method. However, after that, we divided them into functions for efficient development and maintenance. In this process, a number of dead codes have occurred. Proguard has made it easier to find and delete unused code in this situation. When building a file to apk, it can used on obfuscated so that it can not be imported through decompilation.



**<deadcode searching with Proguard>**

At **Release testing step**, testing our application by checking requirements, following user scenario, and testing performance.

First, we check if we missed any **key requirements of our application**.

We have many requirements, so we selected 3 major functional, non-functional requirements from the list precede.

**Functional Requirements**

* Login Function
* Make pictures and notes in relay
* Save note and picture in DB to share

**Non-functional requirements**

* User’s real-time display connection delay have to be less than 2 seconds.
* Login process: sign up should be quickly. -use google sign up
* Main Menu: show most voted ideas on the top of list.

First, we will consider of user requirements from interview to be user-friendly UI and low-delay real-time networking. UI is subjective element to check, so we will accept and fix our UI by user’s feedback. And we expected networking delay maximum to 2 seconds but Firebase supported us well, so delay can be almost none.

Considering functional requirements, we evaluate all of three requirements are well-satisfied than we expected.

At non-functional requirements, user’s delay is much faster than 2 seconds and sign up process is pretty simple at login, but we have to add room list at MainMenu to be sorted by ranking to fulfill third requirement.

Second, we make some **user scenarios** and evaluate isn’t our UI and components fit well with the scenario. We tested sort of **performance** testing together by setting many users in one drawing room and to draw complicated image.

We pretend user scenario to draw detailed tree image with 10 users. We start from user’s sign up and sign in. Sign up process well using e-mail certification and sign up have no problem but we find out we didn’t make log-out button yet so we add it to todo list. Next we press ’create room button’ and make painting room with 10 users. Room generation performs well but problem occurs while drawing detailed image. Firebase only support 10GB memory for free users but detailed image consume big amount of memory, so it cause memory shortage problem when image goes more complex. So we add it to todo list too.

todo list is changed after adding new issues on release testing step is below.

**Updated Todo list for us to do in evolution step**

* **Add social networking function**
* **Setting room title and type function**
* **Public/private Authority for Room and Idea**
* **To minimize transactions at application(solve memory shortage problem of Firebase)**
* **Sort rooms of roomList in MainMenu by ranking**

At **user testing step**, we will testing by alpha/beta/acceptance testing steps.

**5.7. User testing**

**5.7.1 Alpha testing**



Alpha testing is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams. Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for beta test.

Alpha testing can also be done by the potential users or customers of the application. But still, this is a form of in-house acceptance testing.

Alpha testing was done in the following situations: We tested the functionality and network communication status that we implemented using three different Android smartphones**.**

****

**5.7.2. Beta testing**

Beta testing is a testing stage followed by the internal full alpha test cycle. This is the final testing phase where the companies release the software to few external user groups outside the company test teams or employees. This initial software version is known as the beta version. Most companies gather user feedback in this release.

Beta testing was done in the following order: We gave APK file to three external users and asked for testing. Functional testing, and network communication testing, as well as testing the program in a variety of different ways. As a result, we have discovered an issue that even if you connect to another network with a different frequency band on the same router, the communication will occur.



**5.7.3. Acceptance testing**

Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

Acceptance tests were conducted in the following order. A total of 6 people participated in the testing. Six people tested different Android smartphones with different models and versions. The APK file was modified to reflect the results of the alpha test and the beta test. User test results revealed two problems. The first is that the application does not install and start up normally in versions below marshmallow. The second is that when more than five people use the picture function at the same time, the frame drop occurs intermittently.

**6.Conclusion**

While applying SE process into our Term-Project, We realize that Software Engineering treats all aspects of design, development, testing, release and maintenance. We did number of term projects, but we can experience to devide the project into parts same as experts does in real field.  We think it’s really good chance to experience systematic development process.

 We experienced some difficulties at co-working with 3 members, so we realized that Software Engineering methods at real field for working at big size development team will be more necessary. Communications and colaborations between departments will be more important and need to be more specified. We hope this project’s experience will be helpful to do more complete software development.

**7. Glossary**

**Firebase** - mobile and web application development platform serviced by Firebase Inc. They provide multiple services and analysis APIs for managing database.

**layout** - it’s basic visible display units at android application.

**APK** - Android Application Package. it’s package file for android software and middle-ware’s release.

**method(unit)** - methods insides a class. Junit test is executed by unit scale.

**RecyclerView**  - it is kind of view supported by android in XML. it’s extended version of ListView for listing items, which each of them are single views.

**Room** - basic communication unit of our application clients communicates with other clients.

**TODO list** - it’s documentation style at javadocs to marking and arrange tasks we have to do.

**Ranorex** - UI level testing tools. it support problems at UI level and similar to Espresso testing tool in Android Studio.

**deadcode** - it’s non-used or not important code lines at program. we can eliminate them by using tools like Proguard or Codepro Analytix.

1. Pinterest: <https://play.google.com/store/apps/details?id=com.pinterest&hl=ko> [↑](#footnote-ref-1)
2. Relay novel 2: <https://play.google.com/store/apps/details?id=com.gofd.relaystory2&hl=ko> [↑](#footnote-ref-2)
3. Android login opensource url: <https://github.com/CodelightStudios/Android-Smart-Login> [↑](#footnote-ref-3)
4. Painting tool opensource url: <https://github.com/LiuHongtao/PaintView> [↑](#footnote-ref-4)
5. Doodle application GitHub link: <https://github.com/ClimbGoldy/Mobile_Project/wiki> [↑](#footnote-ref-5)