## Mobile client implementation

The Android-based development and system operating environment are as follows:

Operating System: Windows 10

IDE: Android Studio

Android Simulator: Android emulator

Development Language: Java

Construction System: Gradle

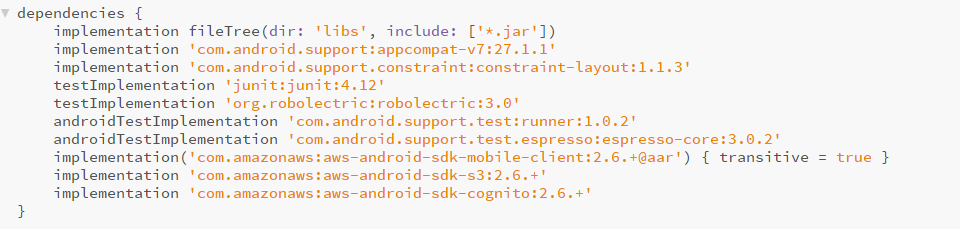
As designed in the planning step, there are 4 milestones in the implementation step. The first step is that finish the basic functions, upload and download which is ensure that the client is connected to the S3.

Before developing, Android project should finish configuring to make sure the app can connect to the S3. Because of the bucket, deadlinefighters, which is used as the server storage is the common use for mobile client and desktop client, so it does not use the mobile hub this tool to configure AWS services and integrate it into the app.

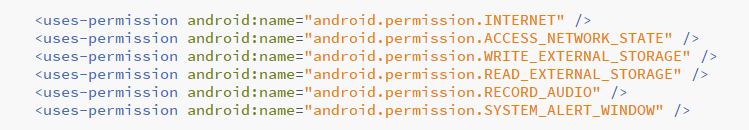
To use the aws SDK in coding, the API it has to choose over or equal to 25 (Android 7.0). The Higher the Version of Android means that it has higher safety for the mobile and meanwhile it increases the difficulty of developing with the increasing permission. On the other hand, using high versions of Android API for development means that you can save some development time without considering the compatibility of low versions.

The configure is basically set on the file app/build.gradle, AndroidManifest.xml, and awsconfiguration.json which is added externally.

app/build.gradle: Add implementation in the dependencies so that it can use aws android SDK to make the coding easier.



AndroidManifest.xml: In this file it adds uses permission so that the app can access the device storage and connect to the internet and other functions as well.



awsconfiguration.json: This file makes this mobile app integrate into the bucket, deadlinefighters.



AWS provides multiple choice of APIs which can help the developers to make application conjunction with Amazon S3. The first one is the low-level API, provided in Amazon.S3 which almost cover S3 APIs completely. Another is TransferUtility, which provided in Amazon.S3.Transfer which is much easier for upload and download.

First, the team uses a low-level API (putObject function and getObject function) because it has the same pattern as other services it uses. It is easier for multiple developers to read the whole code. After building the frame of the whole application the function, this app change to use TransferUtility for upload and download.

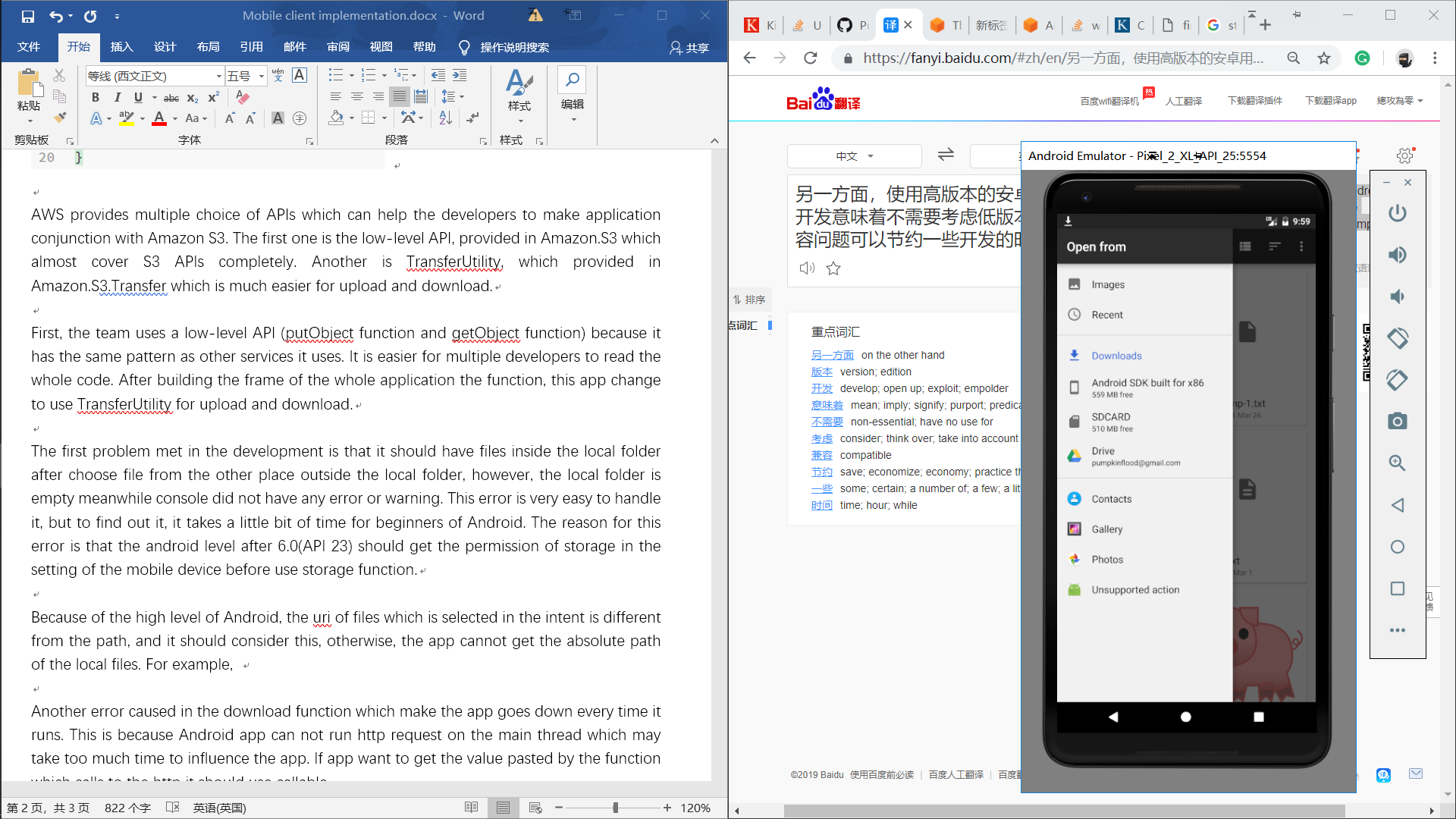
The first problem met in the development is that it should have files inside the local folder after choose file from the other place outside the local folder, however, the local folder is empty meanwhile console did not have any error or warning. This error is very easy to handle it, but to find out it, it takes a little bit of time for beginners of Android. The reason for this error is that the android level after 6.0(API 23) should get the permission of storage in the setting of the mobile device before use storage function.

Because of the high level of Android, function getdata() of files which is selected in the intent is different from the path, and it should consider this, otherwise, the app cannot get the absolute path of the local files. For example, in the list shown in the figure, the file URI in image and download is different, even if it's actually the same file in the same folder.

For the file whose actual file path is /storage/emulated/0/Download/22236-pig-icon.png, it is displayed after select the file in the images is

content://com.android.providers.media.documents/document/image%3A78, and after selected the file in the download is

content://com.android.providers.downloads.documents/document/12



The error occurred in the download function which makes the app goes down every time it runs is that Android app cannot run HTTP request on the main thread which may take too much time to influence the app. This problem only needs to new a thread and then run the request inside it. If the app wants to get the value passed by the function which calls to the HTTP it should use create a new class which inherit the class callable and call it.

After the first step, development goes to the second part which can get the list of files in the server and add delete and rename function, for the edit function it can upload or download what it is different between the server file and the local file. However, for these ideas, some of them are not achieved yet. What has to be mentioned is the edit function, this is used to check if the local file and server file is different by using some identification information like eTag, which is automatic generation when a file upload to the server, other than file name. Based on the original idea, this function should be used to reduce data communication on the network. If there is a local file which has the same eTag, size and last modified as the server file, it means that this file did not change and does not need synchronization. If there is a local file which has the same eTag, with different size and last modified, it means that this file changed and the app will use rsync algorithm to update the delta part of the file. Unfortunately, due to lack of time and personnel, this function cannot be achieved for the time being. As for rename function, it achieved in the old version of the app, but considering the actual use on the mobile device, it seems not necessary for this app and it remove this function in the latest version. S3 does not provide rename, but it can be implemented by using copyObject() to copy a file in the bucket and then use deleteObject() to delete the previous file. It is regretful that no testing has been done to see if this function will change its identification information before remove it.

Because of the edit function is not finished the sync part have some problem. In the case of using the fileObserver class, half of the solution is achieved. It can detect the file change in the local file folder which means that the if there is a file create or modified in this folder it can upload to the server directly. If a file is deleted from this folder it will delete on the server as well.

As for UI of the Android application, the team haven't paid much attention to it. Sometimes it skips frames because it runs too much on the main thread, but because it does not cause major problems in actual use, it is not included in the solution plan.

**To solve the sync problem, it simply use the upload all files and download all files as the first plan. When the app download multiple files it seems good when the number of files is not big, but it lose the files when it download lots of files. So the rsync is necessary for file sync.**

Use AWS SDK to develop which means that s3 is not only used as server storage and it also plays a role as the server. This means that the conflict makes by multiple operations, for example, if two users upload two files which have the same name, will simply cover the old version without compare may cause the unexpecting loss of data.

In this case, the server should make separately, and the s3 is only working as server storage. Because of the change of the server, it should no longer use the s3 SDK for it connect the local storage to s3 bucket directly without going through the server which it should go through.

The app removes all code which is implemented from AWS, and change to use JSON to send HTTP request to get the URL of the operation. And then it goes to the operation functions to upload or download files.

## Mobile Test

First, it should test whether every function is work well, concerned about the actual use in a mobile device these type of files is paid the utmost attention to, text, document, images, audios and videos.

This is a easy test case for the common file extension after change the server.

|  |  |  |  |
| --- | --- | --- | --- |
| File type | extension | upload | download |
| text | txt | No error |  |
| document | doc |  |  |
| pdf | No error |  |
| ppt |  |  |
| image | png | It shows the file in the server, and the according to the size it has upload succeed.  But it does not display the actual picture as it should be. |  |
| gif | Same problem. |  |
| jpg | Same problem. |  |
| audio | Mp3 |  |  |
| videos | Mp4 |  |  |

The delete function is run successfully. Because there is some issue of the list server files, it cannot test the download function now. The problem has found before is that if it downloads a picture, it have a certain chance of not getting the whole picture means that data loss while it is transporting.

Evaluation:

Up to now the mobile app is still cannot run as a finished product. There are 2 factors cause it. First is because no clear design was made before the development. It is a waste of time to decide to replace the existing model only after it is found that it can not solve the conflict. Second, the lack of workers on developing the mobile client makes the error cannot have found on time. To keep up with the progress of the desktop client group, the function after it developed have no fully tested and then it goes to the next step. Some error occurred when developing the next function and should be solved before exactly.

Code reference:

[https://grokonez.com/android/uploaddownload-files-images-amazon-s3-android#5\_Connect\_to\_Amazon\_S3](https://grokonez.com/android/uploaddownload-files-images-amazon-s3-android" \l "5_Connect_to_Amazon_S3)

<https://androidexample.com/Upload_File_To_Server_-_Android_Example/index.php?view=article_discription&aid=83>

https://stackoverflow.com/a/26501296