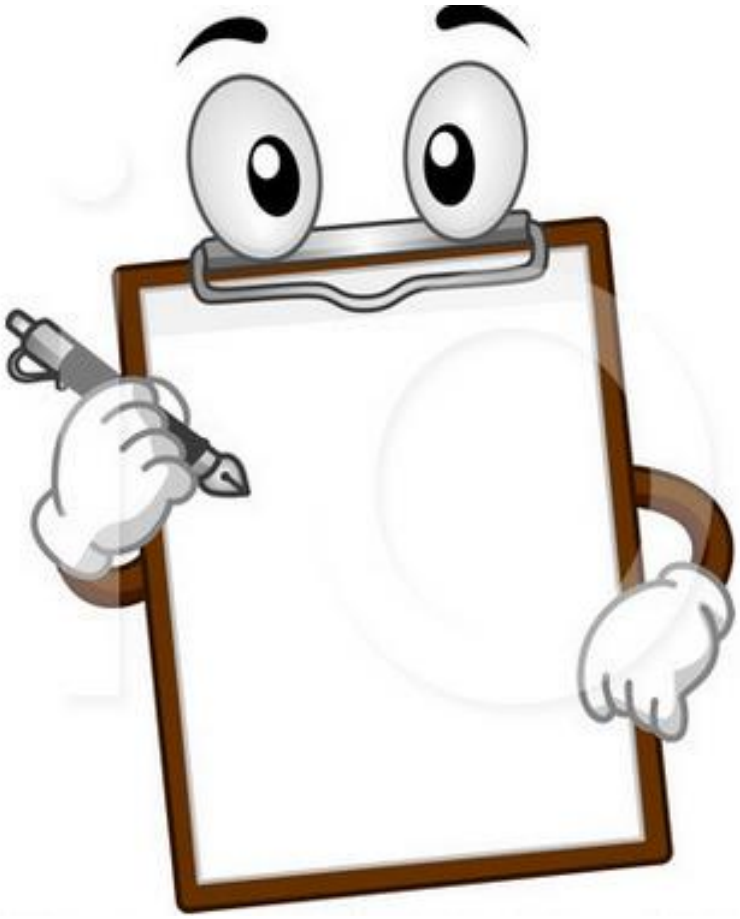
NYU POLYTECHNIC SCHOOL OF ENGINEERING

CS 9033: Mobile Application Development



Design Document

ClipBoard++

Members	N ID	Emails
Pavan Kumar Atri	N 16652853	pab424@nyu.edu
Uttara Chavan	N 12745967	uc252@nu.edu
Rohit Shridhar	N 19496213	rsb408@nyu.edu

REVISION HISTORY

Date	Version	Description
10/18/2014	1.0	Initial Design Document
12/15/2014	2.0	Final Design Document

TABLE OF CONTENTS

1. Project Overview

1.1 Purpose

There is a need to keep a history of all copied items(Clipboard History). Currently, the last copied text overwrites the previous one, and if we need to access the last one, we may have to go back to the application or place where we copied that from and copy again. Instead if there is a place where we can view the list of all copied clips and get the one we want that would be a lot convenient.
Faster Text transfer between devices- This is the age where we use multiple smart devices in our daily lives, and often we may want to transfer something that we typed in PC to mobile device or vice versa. For example, say we want to type out a huge message and broadcast it to a *Whatsapp* group. Typing a large message on a mobile device is very inconvenient. You can type it out on your PC and transfer it to your mobile device. And the text transfer again needs to be done either through an email application or you can use slightly better options like *Pushbullet* or *Evernote*. But imagine if all you need to do is copy it in your PC and paste it on your mobile device. That would make your life a lot easier.
Clipboard++ exactly addresses these two daily problems faced by any smart phone user.

1.2 Introduction

Clipboard++ will be a good clipboard manager that automatically saves everything you copy. Access your collected clippings later and organize them in lists. Copy, paste, view, edit and share their contents. Store repetitive pieces of text in Clipboard++and copy them whenever you need to. Take control of copy and paste with Clipboard++.
Some of its features are:

- **Automatic & seamless clipboard history and extension**. All copied text is collected and saved for later use. Don't worry about copying over anything important.

- **Easy clipping organization and editing.** Copy a clipping back to the Clipboard++ with a single tap. Clips are neatly organized into three categories- Today, Yesterday and Last Week.

- **Quick and easy access.** Open Clipboard++ through the notification tray for quick access to your collection.

- **Note:** Simply send text of any kind to other devices.
- **Link:** Sends a URL with its title, along with any notes you may want to include. When you then click it on your phone, it opens it in your default browser.

2. CURRENT SOLUTION AND RELATED WORK

Many a times we want to transfer text from one application to some other application or even from mobile application to our computer. For example every time we want to copy some text from mobile web browser to the messaging app, we have to go to the webpage where text exists find those particular lines and copy them. Then open the desired application and paste the contents in application. We have to do this process every time even if the contents are same. In case, we want to transfer text from mobile application to computer or some other devices we have to explicitly use email services for this purpose. Wouldn't it be great if we can get all these functionalities within a single app?

Several apps are developed to meet some of these requirements but they do not include all these features. Let's take a look at each of the existing apps, features they lag and how Clipboard++ can be used as a solution.

Clipper:
Clipper takes control of copy/paste and automatically saves everything user copies. These collective clippings can be accessed later and organized in lists. Though clipper provides cross application functionalities on mobile device, it cannot be used on multiple devices synchronously.

Evernote:
Evernote is used to store the clip articles, texts, and images directly on the cloud which can later be accessed anytime on any device.

But user has to push the contents to the application explicitly.

GoogleKeep:

This app can be used to quickly capture anything on user's mind. Provides facility of adding pictures, voice memo etc. Application even reminds you of your note at right location. Although this application can be used on multiple devices synchronously, it cannot be used to copy the contents from one mobile application to other. It also puts the character limit on the notes user makes and no categorization options are available.

PushBullet:

Similar to Evernote, PushBullet can be used to transfer the contents from one device to other. But it is better than Evernote in a sense that it does its work without user's active interaction to the pushbullet.

As a downside PushBullet cannot be used to copy contents from one mobile application to other.

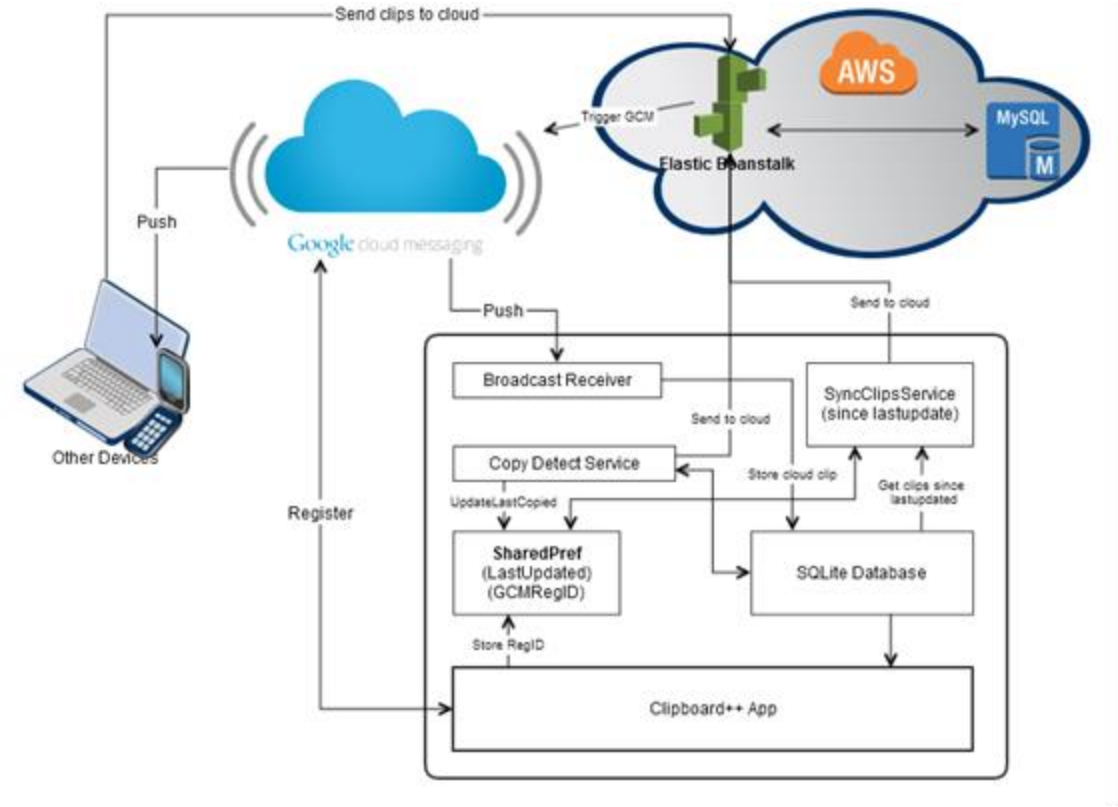
How ClipBoard++ is better:

Clipboard++ is better application to use as it aims to provide the clip access across multiple devices including desktops, to push the clip contents without user's active involvement, to provide copy storage with unlimited space and categorization along with cross device availability.

3. SOLUTION

3.1 SOFTWARE ARCHITECTURE

Clipboard++ is a mobile app that helps to maintain clipboard history (of text) and also to sync the copied items across devices. The app uses AWS resources and Google Cloud Messaging Service in achieving its purpose. User will have to install the device specific Clipboard++ app for each of his devices. Detailed explanation of the architecture is as below:



Installation: On installation, the app contacts Google Cloud Messaging (GCM) and registers the device. The GCM component gives a unique registration ID upon successful registration, and this ID is stored as a SharedPreference key in the local device and also an entry will be made in the MySQL DB on cloud in the device table. This entry will have the device ID (combination of Android ID and IMEI number), email ID(from the android account), GCM sent registration ID and the device description. This process happens only once per installation.

Copy Detect Service: is a background service that has a listener setup on the Android's Clipboard Service, so whenever text is copied in the mobile device, the Clipboard service notifies the listener and gives the clip that is copied. The service then

stores the clip into device's SQLite database and also makes a call to the web service running in AWS Elastic Beanstalk. For the details on the web service calls refer to the Web Service section. The web service then stores the clip in the MySQL database (AWS RDS), and also with the help of device ID gets all the registered devices against that user and notifies all the other devices using GCM service. The GCM sends a Push Notification to all the devices, which will also have the clip content.

The Broadcast Receiver on the mobile device receives the push notification and stores the clip into its SQLite database.

SyncClipsService: This is an intent service that updates the cloud DB with all the clips that were copied when the device did not have the network connectivity. This is listening to the network access state, which will be notified every time there is a change in network connection. When the network connection is back on, this service gets all the clips from the device's SQLite database, that were copied after the *LastUpdateDateTIme* SharedPreference key.

Shared Preferences: Two keys are being used that are stored in different shared preference files:

LastUpdatedDateTime: This will be updated with datetime, everytime a clip is successfully sent to cloud. This key will be used by *ClipSyncService*.

RegistrationID: Contains the GCM registration ID

SQLite Database: This component holds all the clip contents (Details in Database section).

Clipboard++ User interface will fetch the clip contents from the local database every time the activity is resumed or created.

3.2 USE CASES

3.2.1 Access the Application from the Notification Tray

Use Case	Access the application from notification tray
Description	This process helps user to get quick access t the application which is continuously running in the background
Preconditions	1. User has installed application
Normal Flow	1. Once user has installed application, to open the application he has to slide down the notification tray 2. Tap the Clipboard++ application icon appears in notification tray 3. Application is even accessible by clicking the application icon on main menu.
Post Conditions	Homepage of the ClipBoard++ application opens

3.2.2 Main Screen

3.2.2.1 Home Page

Use Case	Home page
Description	On Homepage user can see all his clippings listed.
Preconditions	1. User has installed application
Normal Flow	1. User opens the application using notification tray quick access or going through the main menu. 2. Make sure he is under ‘Clipboard’ tab on the screen.
Post Conditions	Application main screen displays the clippings in the order of most recently clipped to the old ones.

3.2.2.2 Navigate Categories

Use Case	Navigate categories
Description	Using this feature user can view the clippings listed by categories like • Frequently used • Recently used • Used in last today/yesterday/Last Week • Latest First/Oldest First
Preconditions	1. User has installed application 2. Copied some text, so the clippings exist in the app
Normal Flow	1. User opens the application 2. Clicks on the ‘Timeline’ tab 3. On this page he will click the category from which he wants to view the clippings
Post Conditions	On clicking the category, appropriate list will open up.

3.2.2.3 Search the Clip

Use Case	Search the clip
Description	User can search for clipping containing a specific word
Preconditions	1. User has installed application 2. Some copied clippings exist in the application
Normal Flow	1. User opens the application 2. In either of the tabs he clicks on the search menu provided at the top and type in word to search from clippings
Post Conditions	Under the selected tab, all the clippings containing the search will appear on the page.

3.2.3 Copy Text

Use Case	Copy text
Description	User can copy the text in any application which will then be available in Clipboard++ for later use.
Preconditions	1. User has installed application
Normal Flow	1. User copies text from any application.
Post Conditions	The copied text gets listed in the Clipboard++ and then it is pushed to the cloud which can later be accessed by other applications and even other devices.

3.2.4 Paste Text

Use Case	Paste text
Description	User can view the already listed clippings in Clipboard++ and paste them to desired application with a single click.
Preconditions	1. User has installed application
Normal Flow	1. User opens the application selects the text he wants to copy over to other application or device. 2. Then opens the desired application and simply long press to paste the content he already copied.
Post Conditions	Copied text appear in the destination application.

3.3 MVC FRAMEWORK

3.3.1 Models

ClipBoard++ project does not contain any models for clips or any other components. We decided to go for this design as project contains just one MainActivity and doesn't require to send the clips from one activity to other via intent. Creating an object for clips would have simply created overhead for accessing the clips.

3.3.2 Views

3.3.2.1 ActivityMain

Links to the MainActivity that contains the below three fragments:

3.3.2.2 Clips

Maintains the listview for the clips loaded from android internal Sqlite database. This View also gives search bar for searching the clips containing input word.

3.3.2.3 Frequently_Used

Maintains the listview for the most frequent 10 clips user has copied from clipboard++ to other application on the same device.

3.3.2.4 History

This view maintains the Expandable listview to display timeline according to the clips copied by user. Timeline gives user facility to use clips copied 'Today', 'Yesterday' and over last seven days i.e. 'LastWeek'.

3.3.2.5 List_group

This view is simply created to maintain the three timeline groups separate and to give them customized styling.

3.3.2.6 List_item

This view is used to maintain the list of clips maintained under each section and style them.

3.3.3 Activities

3.3.3.1 MainActivity

This will holds following three fragments
Activity uses TasksPagerAdapter to implement swappable tabs used for fragments. Also starts the DetectCopyEventService and contains the GCM registration functionality which will run once per installation in an AsyncTask and stores the received registration ID in a SharedPreference key.

1. **Clips** –
   This fragment maintains all the clips copied by the user on various devices connected to the clipboard++ through same user ID.
   All the clips present here are searchable and can be deleted on long press on each item in list.
   To load, select, search and delete the clips methods defined in ClipBoardDatabaseHelper are used.

2. **ClipHistory**-
   This fragment holds the clip data arranged according to the date of clip copying across the devices.
   To load, select, and delete the clips methods defined in ClipBoardDatabaseHelper are used.

3. **Frequent**-
   Ten most frequent clips are displayed under this tab with most frequent clip present at the top.
   To load, select, and delete the clips methods defined in ClipBoardDatabaseHelper are used.

3.3.4 Other Helping Classes

3.3.4.1 GCMBroadcastReceiver

This class continuously listens to the messages sent by the GCM and sends the package received to the GCMMessageHandler for further processing. It also launches the WakeFulService.

3.3.4.2 GCMMessageHandler

This class unbundles the package sent by GCMBroadcastReceiver and copies to the clipboard to paste it in other application. This helps user to use the clip copied on any other devices to other applications on clip receiving device without explicitly copying it again.

3.3.4.3 ClipDataBaseHelper

Class is responsible for create the internal Sqlite database. All the database maintenance functions are defined in this class such as:
Insert the clips – insertClip
Fetch all clips – getAllClips
Check if clip already exists to avoid duplicates – checkIfClipExists
Get clips in a date range provided – getAllClipsFromDate
Get clips on particular date – getDayClips
Delete the clip – deleteClip

Get frequent clips – getFrequentClips
Get all the clips searched containing particular word - getAllSearched

3.3.4.4 ExpandableListAdapter

This class inflates the expandable parent group view and child list view for each group.

3.3.4.5 TaskPagerAdapter

Each fragment class clips, ClipHistory, Frequent fragments are linked to the tabs using this adapter.

3.3.4.6 ClipboardSynchronizeService

This service will be called by NetWorkChangeReciever Boradcast Receiver, that is when the network availability is back on, this service is called to synchronize clips with that in the cloud. It will fetch the value from the SharedPreference key LastUpdatedDateTime, that will have the *datetime*, the last time a clip was successfully sent to the cloud. So this service fetches all the clips copied after that *datetime*, from the SQLite and sends to the cloud so that other devices could fetch it.

3.3.4.7 DetectCopyEventService

This is a background service that will be listening to the Android Clipboard service, and gets notified whenever a new clip is placed onto it. It then fetches the clip, inserts into the device's database(SQLite) and also makes a call to the web service to insert into the database on the cloud. This service is started by the MainActivity's onCreate method, and it will be running in the background.

Methods included:

OnBind(Intent intent) - This method returns an IBinder object that defines the programming interface that clients can use to interact with the service. If you only need to interact with the service while your activity is visible, you should bind during onStart() and unbind during onStop(). If you want your activity to receive responses even while it is stopped in the background, then you can bind during onCreate() and unbind during onDestroy().

onPrimaryClipChanged() - This method picks the user copied text and places into cloud and notifies the user, if copied. Suppose if the item copied already present in cloud notifies the user as it is present in cloud.

getDeviceID() - Device id is concatenation of android id and IMEI number

updateLastUpdSP() - Sets the time at which the text is sent to cloud

getUsername() - This method gets username from the device’s google account.

ClipboardAWSService: This is an AsyncTask that makes a call to web service in Beanstalk, to insert the copied clip into the database in cloud. The web service

command used here is "insertclip". The AsyncTask does not implement pre or post execute methods.

3.3.4.8 ClipBoardPlusPlusUtil

This class has a utility method to convert datetime value to its equivalent string format.

3.3.4.9 NetworkChangeReceiver

This module tells about the type of connectivity the device has been connected it could be either Wi-Fi or mobile network.

Methods included:

Public void onReceive(Context context, Intent intent) - This method lets the user know that the app is connected to internet.

Public static boolean getConnectivityStatus(Context context) - This method shows the internet status and type of connectivity it could be either Wi-Fi or mobile network.

3.3.5 Database Design

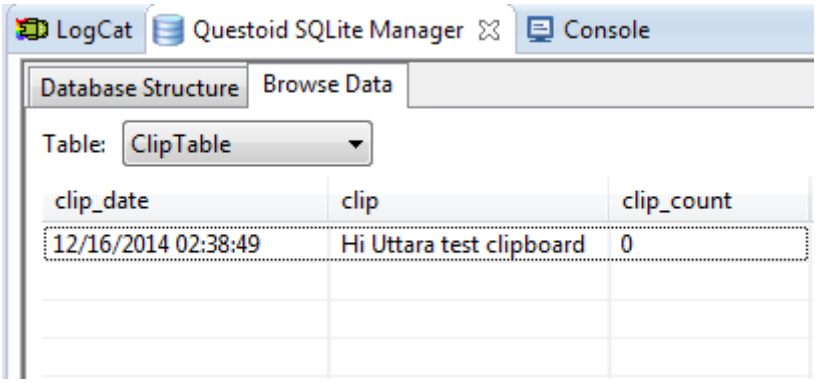
3.3.5.1 Internal Database Storage

Android Sqlite Database –

Copied clips are stored internally in Clip_internal table. TimeOfCopy is the primary key for the table and maintains the timestamp of the clip copied. As though the clip content is same time of copy across the devices can't be same. Clip_content maintains the clip content copied from device itself and across various devices. Count field maintains record of how many times a clip in database is copied for use. Every time a clip is used count for it is incremented in database.

Shared Preferences –

- Last Updated: This shared preference is used to keep record of the last time clips copied on device are updated to cloud for the use by other devices registered with the same user ID.
- Device ID: When device installs the application clipboard++, and gets registered to the GCM server, the device ID generated by GCM server is stored in this shared preference. This ID is then later used to send the clip to the server running on AWS where device ID mapping is maintained I database. Server then determines to what all device IDs this newly updated clip should be sent via GCM.



3.3.5.2 AWS Database Storage

RDS –

RDS maintains two tables for synchronization between various devices for the same user.

Device: maps device IDs to the users and description of devices. Device_ID is the primary key and it is combination of Android ID and IMEI number of device.
Reg_id stores GCM generated key for the device.
User_name is the user ID with which application is downloaded. Description field keeps information about the device type.

Clips: This table has all the clips copied across all the devices with the same user_id. TimeOfCopy and Device_ID are the primary key for table. Clip_content stores actual clip content.

clipcontent		copydatetime	deviceid
Poster board presentation at AT and T	37B	2014-12-14 03:42:17	a7b707d009fbcb4d:864587020875945
And where is everyone?	22B	2014-12-14 03:42:41	a7b707d009fbcb4d:864587020875945
Hello at and t	14B	2014-12-14 03:42:46	a7b707d009fbcb4d:864587020875945
Feela good to be here	21B	2014-12-14 03:42:52	a7b707d009fbcb4d:864587020875945
Hey I changed place.. To the other side	39B	2014-12-14 03:42:58	a7b707d009fbcb4d:864587020875945
message from pc to clipboard	30B	2014-12-14 03:46:44	simulated-desktop-pc-2

3.3.5.3 AWS Web Service

ClipboardAWSService is a webservice hosted on AWS Elastic Beanstalk. This service basically helps in performing CREATE and READ operations on the MySQL database also on the AWS. The request structure and the Web service URL is as below:

http://clipboardplusaws-env-e33d5e3cep.elasticbeanstalk.com/clipboardplusplusws

METHOD: GET

PARAMETERS:

Parameter Name	Description
command	The function done by the respective command
deviceid	Generated device ID (concatenation of Android ID and IMEI number)
username	The email ID which will be the google account associated with the user's Android device
clipcontent	The clip text that is copied from the android device

The Web service has the below classes:

ClipboardPlusPlusDAL (Data Access Layer): This is the data access layer that contains all the functions required to interact with the database.

ClipboardplusplusWSServlet: This is the main servlet that receives the GET commands from the app, gets the required parameters and passes onto the Data Access Layer.

POST2GCM: This contains the functionality to send push notifications to the devices using GCM. It creates a Content object (explained below), which will hold the device registration ID and the clip content.

Content: This is a data type that implements Serializable, and it has two attributes registrationId, which is the device registration ID to which the notification is to be sent, and the data, that will hold the clip content itself in a key value pair.

The following are the different commands performed by the Web Service:

- ***insertclip***: This command inserts the given clip into the cloud database, against the given device ID.

http://clipboardplusaws-env-e33d5e3cep.elasticbeanstalk.com/clipboardplusplusws?command=insertclip&deviceid=a7b707d009fbcb4d:864587020875945&username=atripavan@gmail.com&clipcontent=say%20hi%20to%20clipboard

- ***createdevice***: This command creates an entry into the Device table, completing the registration process of a device. The command will have the deviceId, registrationId, username and device description as shown below in the sample request. This will happen when the app is installed in the device

```
http://clipboardplusaws-env-
e33d5e3cep.elasticbeanstalk.com/clipboardplusplusws?command=cr
eatedevice&deviceid=a7b707d009fbcb4d:864587020875945&regid=APA
91bEk_4y3e8ZoEnkLkmclTFbDOnahz_XUUhmZCAthL1AISj1AGNb6IQ0zo0eY7
v18AH9V1V_fNj9jomVXOlowQcDTKMYJtqE8JvFMOy9Cl2ttSyaSkAhvKEvKxE2
lTBgderTkIER3dhpkR3riln1L3wx6EXTjPqj-W4KcvFJUy02IbBjG5-
8&username=atripavan@gmail.com&description=android%20device
```

- ***insertallclips*:** This command will have multiple clip entries in its request. The command is executed when the app regains the network connectivity and the clips copied while the network was off is collected and sent to the web service to store in cloud. Clips are concatenated into a single string using “|@|” delimiter. The data access layer splits the string and inserts into the clip table.

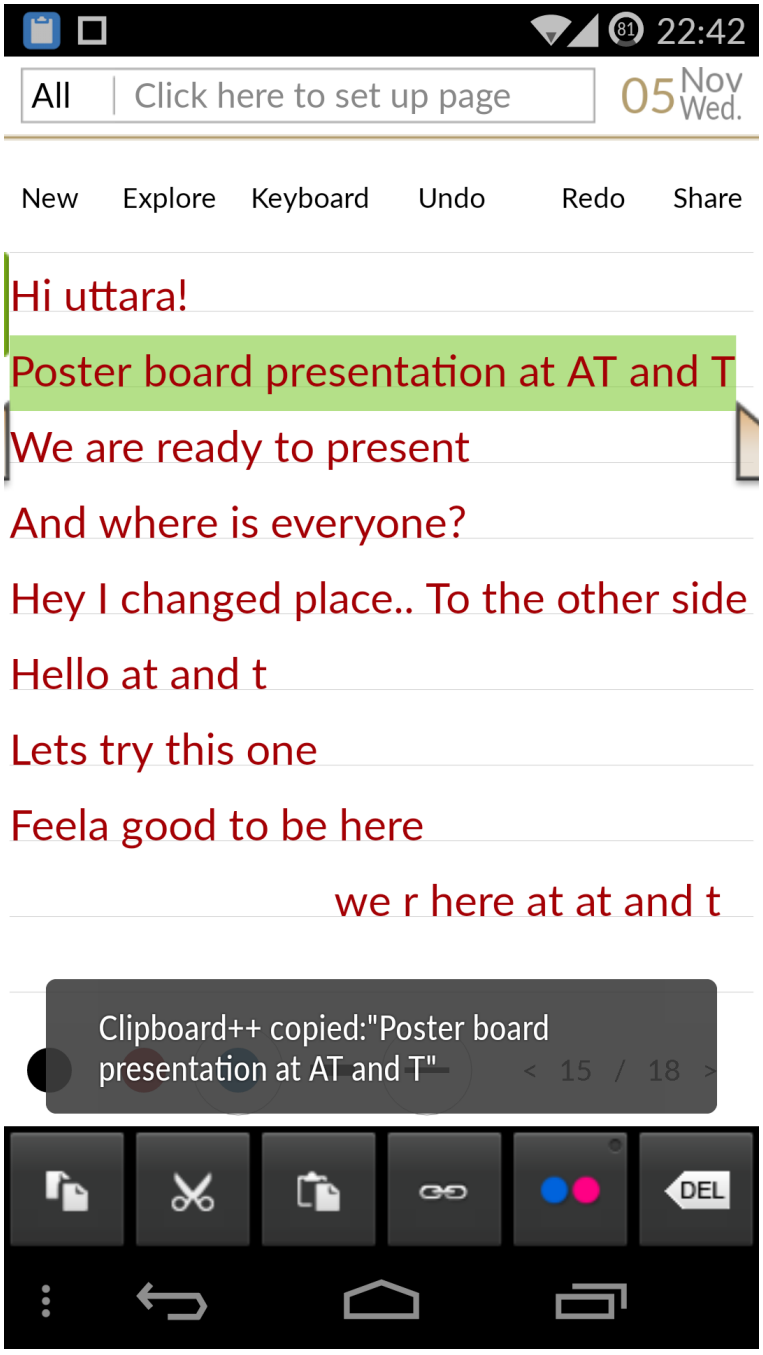
```
http://clipboardplusaws-env-
e33d5e3cep.elasticbeanstalk.com/clipboardplusplusws?command=in
sertallclips&deviceid=a7b707d009fbcb4d:864587020875945&usernam
e=atripavan@gmail.com&allclips=how%20r%20u%20doing?|@|i%20am%2
0doing%20good%20how%20r%20u
```

- ***getclips*:** This command is currently only used by the Desktop Simulator to fetch all the clips under a particular device ID. This command is used only for the simulation purpose.

```
http://clipboardplusaws-env-
e33d5e3cep.elasticbeanstalk.com/clipboardplusplusws?command=ge
tclips&deviceid=a7b707d009fbcb4d:864587020875945
```

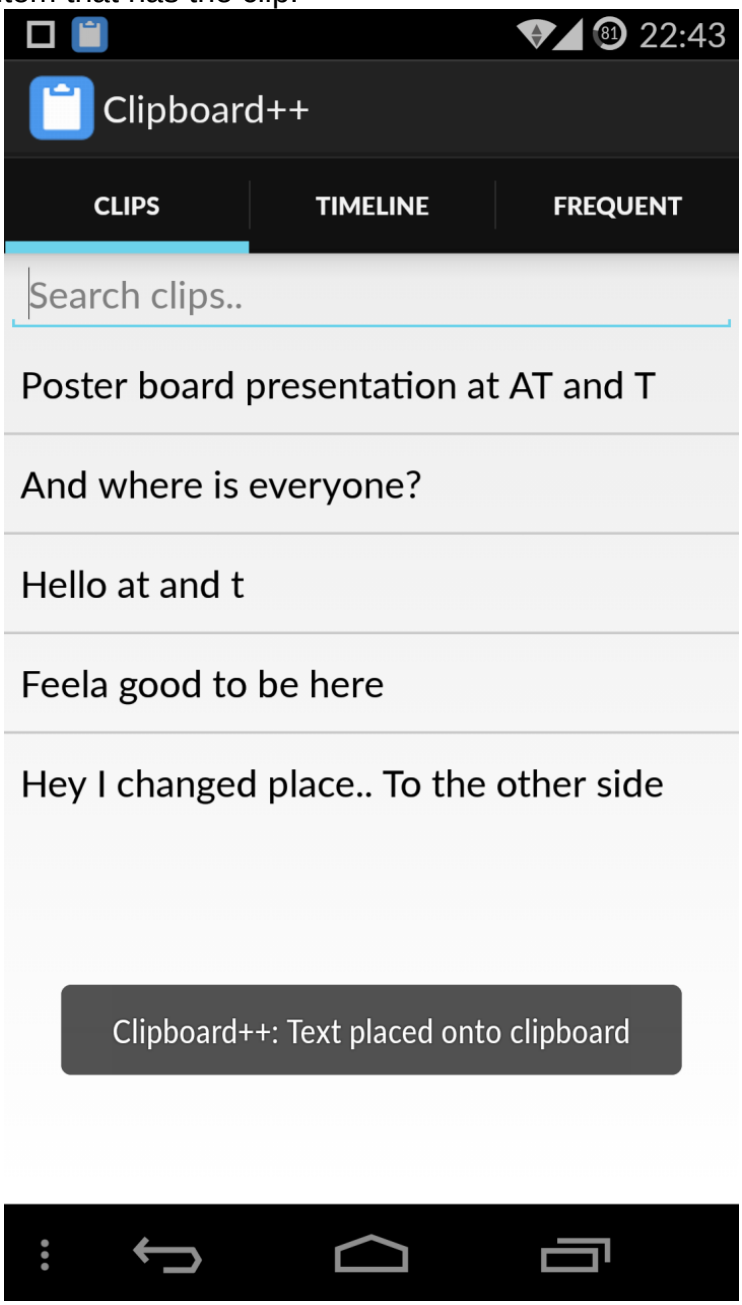
3.4 USER INTERFACE DRAWINGS

The snapshot shows the text “Poster board presentation at AT and T” being copied, which is captured by the Clipboard++ app indicated by the toast message.



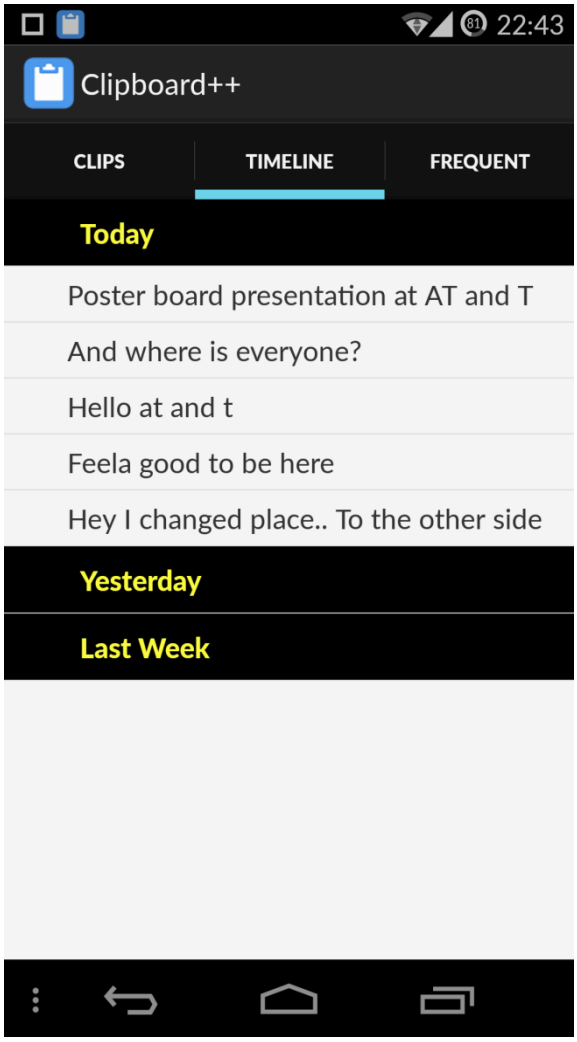
3.4.1 Clips Tab:

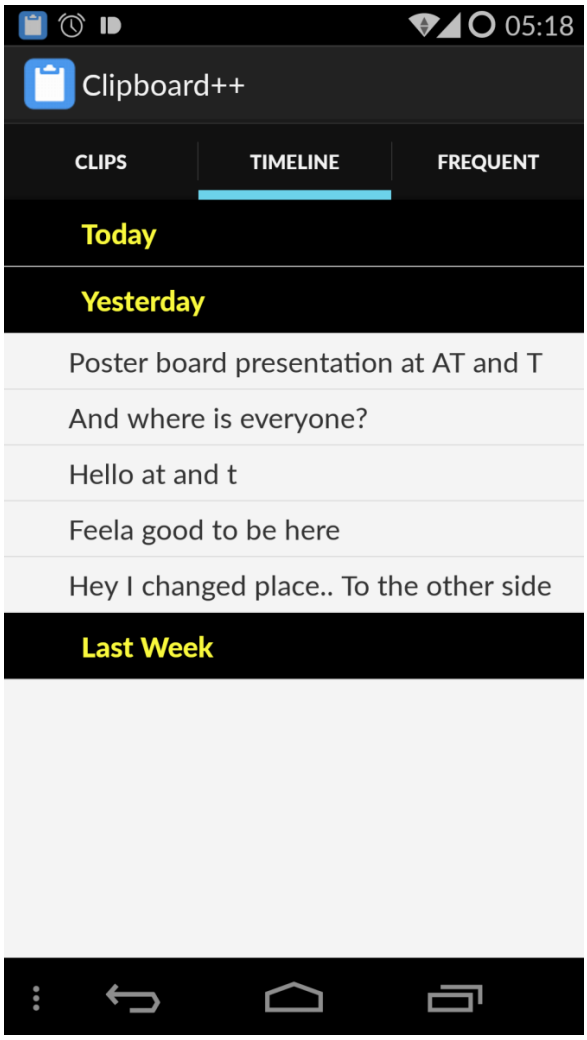
The below snapshot shows the Clips Tab displaying all the clips which have been copied. It also includes search feature, wherein the user can enter keyword to filter out the clips in the list. Also shows that the clip is being placed onto the clipboard on touch of the list item that has the clip.



3.4.2 Clips History Tab:

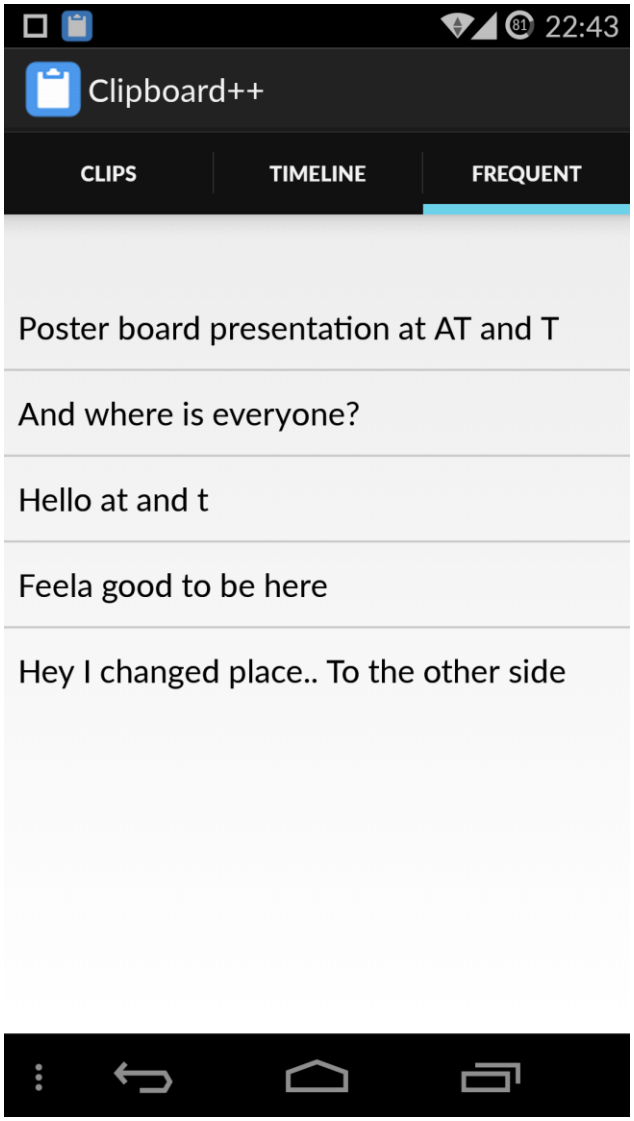
TIMELINE Tab showing “Todays” and “Yesterday’s” clips (Please notice the change in the clock).





3.4.3 Frequent Clips:

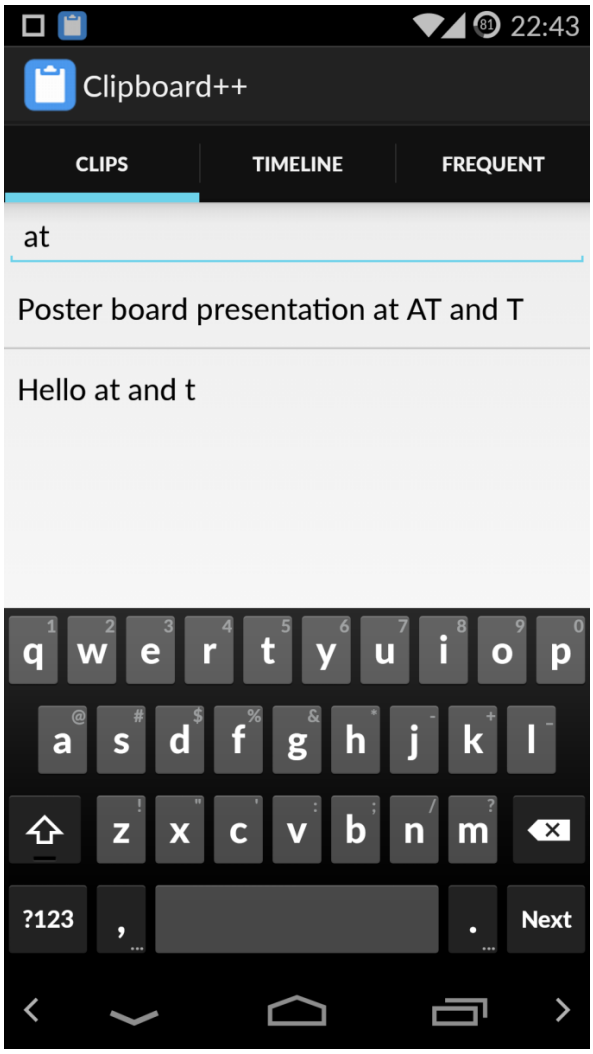
The below snapshot shows the Frequent tab that lists the clips in the order of the number of times they were copied, with the highest being on top.



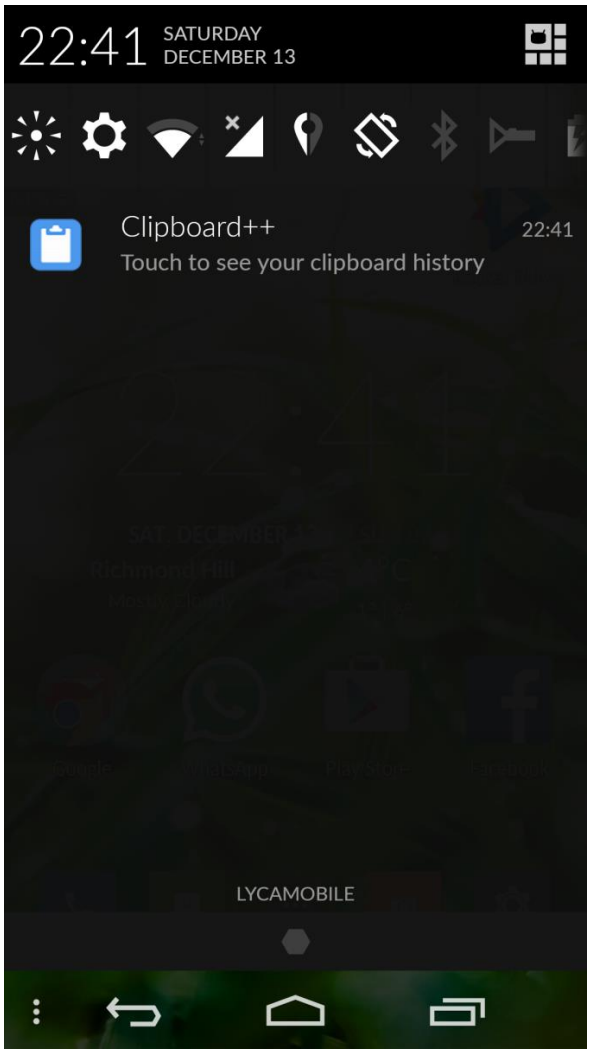
3.4.4 Other Features:

The first snapshot shows the clips that were filtered using the *Search* feature. The other snapshot shows the Permanent Notification that will enable user to access the Clipboard++ UI from anywhere on the phone. All the user needs to do is to drag the notification tray down, and touch the notification.

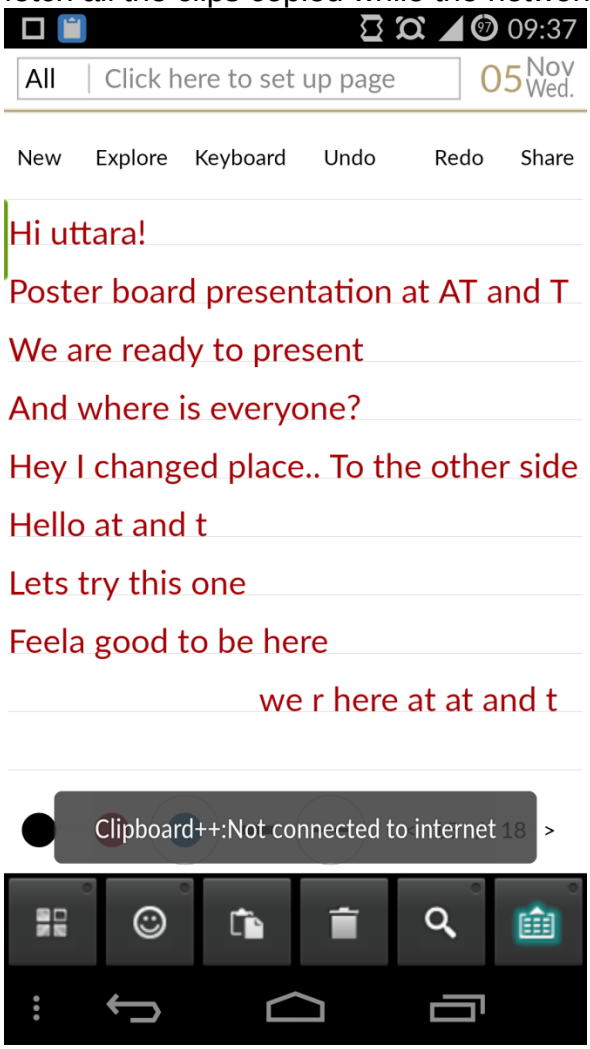
Search

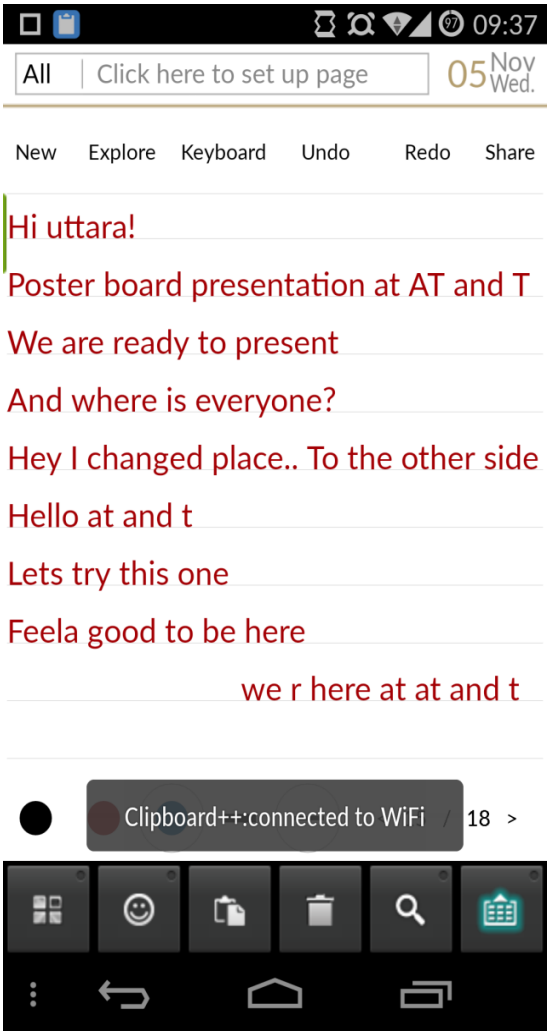


Permanent Notification

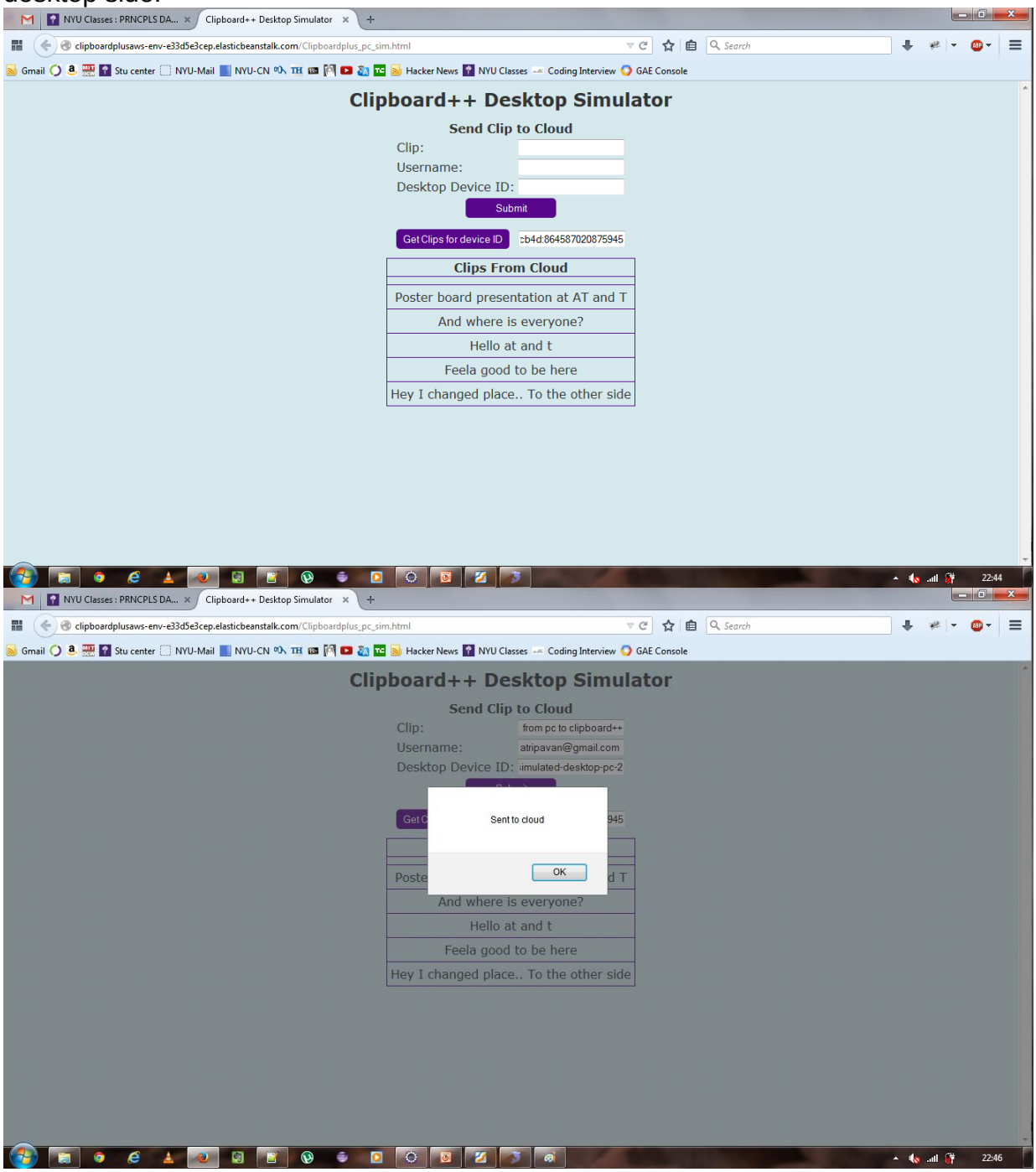


Network Connectivity:
Below snapshots show that the app is getting updates whenever there is a change in the Network status. When the network is back ON, the ClipboardSyncService will fetch all the clips copied while the network was off and sends them to the cloud.





Desktop Application Simulation:
Below snapshots show the desktop simulator, that was used to test the app. This simulates the Clipboard++ desktop app. You can see the clips fetched from the cloud and also a clip sent to the cloud, which simulates the *Copy* action on the desktop side.



4. Project member Breakdown

Uttara Chavan

- Design Android Internal Sqlite database to support insert-update from cloud database, delete clips and fetch for display in timeline.
- Implemented GCM and Broadcast receiver.
- Create Fragment UI with swipe functionality. Created clips timeline feature.
- Get android User Name, Device ID and Android ID.

Rohith Shridhar

- Implemented DetectCopyEvent.
- To support the application when device has no network access, implemented Network Synchronizations Service.
- Implemented permanent Notification Bar for the application.
- Incorporated Search functionality for clips.

Pavan Atri

- Created Webservices which interact with cloud services to maintain synchronization between devices.
- Setup database on cloud.
- Integrated various functionalities developed independently by team members into one single project.

5. Milestones

Date	Milestone
10/13/2014	Initial Design Document
10/27/2014	Setup GCM and Implement Broadcast Receiver
11/04/2014	Design Datastore and Models
11/22/2014	Implement Web Service
11/30/2014	Implement Background Services
11/26/2014	Poster presentation at AT&T
12/07/2014	Clipboard++ simulator, fragment and search feature implementation
12/08/2014	Integration of different modules and testing
12/09/2014	Final Demo

6. AT&T Suggestions

6.1 Suggestions Received

Feedback 1: *If the user purchases a new device and registers with Clipboard++, he should be able to have access to the clips he has copied earlier from the other devices.*
Implementation: Initially we had planned to delete the clips from the cloud when the other devices download the clip. This delete feature is removed and the clips are retained in the cloud so that the new devices that user registers will have access to these clips.
Feedback 2: *For sensitive data like password there should be security feature implemented.*
Implementation: Implementing this feature is a challenge. To keep the user's data secure on the cloud, the data (clip content) should be encrypted using AES key generation. And this key needs to be generated by the user. But the user should have access to the same key on the other devices to decrypt the clips the app receives from the cloud. To make the user to use the same key across devices is a challenge and needs to be thought of

6.2 Future Scope

- Develop Desktop Application:
As of now we have simulated the counterpart of the ClipBoard++ which runs on desktop. Developing a full fledged application to share clips across devices will be the first in future scope of the application. It involves some challenges like implementing a feature which will receive clips pushed by GCM.

- Develop iOS application:
Develop a clipboard++ application which will run on iOS.

6.3 Summary

What we have currently in the app store are these apps:
Pushbullet- Has a universal copy/paste solution where you can copy in one device and paste in the other, but does not maintain a clipboard history
Evernote- Not so convenient as you have to open the app and paste the content and then acces it in the same way from the other device
Clipper- Maintains clipboard history but does not help in transferring text across devices.

Clipboard++ scores over the above on two main features- Clipboard History and Easy text transfer. We can notice that none of the above apps provide both the features, and that the text transfer in Clipboard++ is way too convenient compared to the others. All it takes is copy on one device and paste in the other. And you also get a neatly organized Clipboard History, put into 'Today', 'Yesterday' and 'Last Week' Categories, along with access to the ones you use very frequently and a search feature on the list of clips.