COEN 296
Introduction to Cloud Computing Term Project

Saas - Client server Android application with notification service in personal cloud storage System for Cloud Computing.

Submitted To
Dr. Prof. Ming Wang

Submitted By
Darshita Shah 1033286
Priya Bhatt 1065636
Chen Sun 1000983

Table Of Content :

Table of Figure:

2. Introduction

2.1 Objective

Our objective is to develop a client server system which includes a cloud storage system and provide clients with the following services:

- Cloud Storage System (Storage as a service)
- Optimized Push Notification service using GCM(Google Cloud Messaging)
- RESTful web service using HTTP protocol for client/server communication.

2.2 What is the problem?

The previous infrastructure was everything being local. With that approach there was high risk of loss of data plus data unavailability. Eventually that problem was overcome by centralized cloud storage. But the next problem was data syncing between client and central storage. Clients used to pull from central storage system, but that approach drains so much power from clients. They used the traditional request/response model where client raises a request and server upon receiving the request appropriately prepares a response and sends it back to the client. Internet is growing in leaps and bounds in regards to the volume of content. This makes it increasingly difficult to reach the relevant information correctly and swiftly.

2.3 Why this is a project related to class?

This project presents two services cloud storage as well as push notification service. Cloud storage is providing storage as a service to the client with security, reliability and availability by storing the client's data on cloud. The push notification service provides notifications via cloud on the client's system . The notifications are pushed by the cloud to the client's device notifying that they have some data to fetch. These aspects cover up the services provided by a cloud and are thus concerned with introduction to cloud computing class.

2.4 Why other approach is not good?

There were two approaches to provide data syncing between client and central storage server. The first approach was the request/response model where the client will request the data and server will send the appropriate response. But in case where one is concerned with real time information update as soon as it is created or wants to track changes in existing information in real time this approach failed.

The next approach was to provide publish/subscribe model. Technology which uses the concept of publish/subscribe model offers an appropriate way of exchanging just in time information. Push (also called server push) describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server and a client might subscribe to various information "channels". Whenever a new content is available on one of those channels, the server would push that information out to the user. This approach was good since it provided real time update. But as the number of applications increased on a device, maintaining connection with various servers lead to high drainage of battery life as well as decreased the throughput of the device.

2.5 Why this approach is better?

In this approach we are implementing notification service which not only notifies result but also notifies errors , warnings & also available services. We are also trying to implement all these features with words as well as symbolic so a person who either knows notification language or not can understand. Also we aim to implement this service using GCM server , in that case device just needs to maintain connection with google cloud messaging and thus saves the battery life and increases throughput.

2.6 Statement of the problem?

We are implementing Saas Software as Service - Push Notification service for Android Application by using GCM - Google Cloud Messaging. This notification service can be used in multiple applications at the same time by making little modifications.

2.7 Area or scope of investigation

We can enhance our cloud storage application by providing security and backup. We are aiming to provide notification for adding/deleting a file. In future we can also provide notifications for file sharing, editing etc. The third possible enhancement can be using both push as well as pull mechanism on the same device. Pull mechanisms can be used where the data does not need real time update. We can provide RESTful web service using HTTPs protocol instead of HTTP.

3. Theoretical bases and literature review

3.1 Definition of the Problem

Implement Saas Software as Service for cloud storage system and Push Notification service for Android Application by using GCM - Google Cloud Messaging.

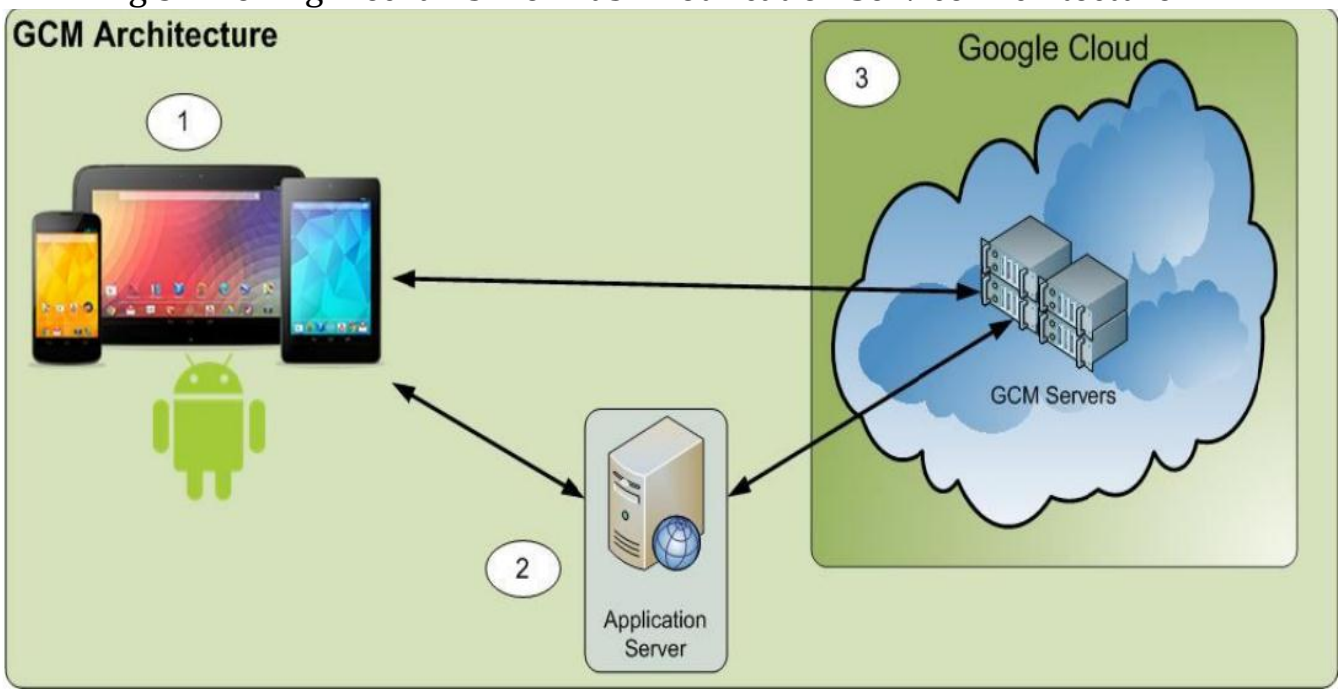
3.2 Theoretical background of the problem

Most mobile solutions have a client-server architecture, where mobile cloud storage application frequently download updated data from the server. In some situations, the server needs to notify the client on changes such as app update, advertisements etc. Availability is all time major issue in such criteria. For cloud storage system security is also major issue.

3.3 Related research to solve the problem

Our notification service model is based on two main mechanisms - pulling mechanism & pull notification.

Fig 3.1 Polling mechanism or Push Notification Service Architecture



As shown in fig 3.1 , This is achieved by using either Polling mechanism or Push Notification services.

A) Push Mechanism

Following 2 approaches can be leveraged to implement Push Notification Service:

Approach 1: We can use a proper push bearer, a network mechanism that delivers unsolicited messages to devices. SMS and Blackberry's proprietary push solution are examples of push bearer.

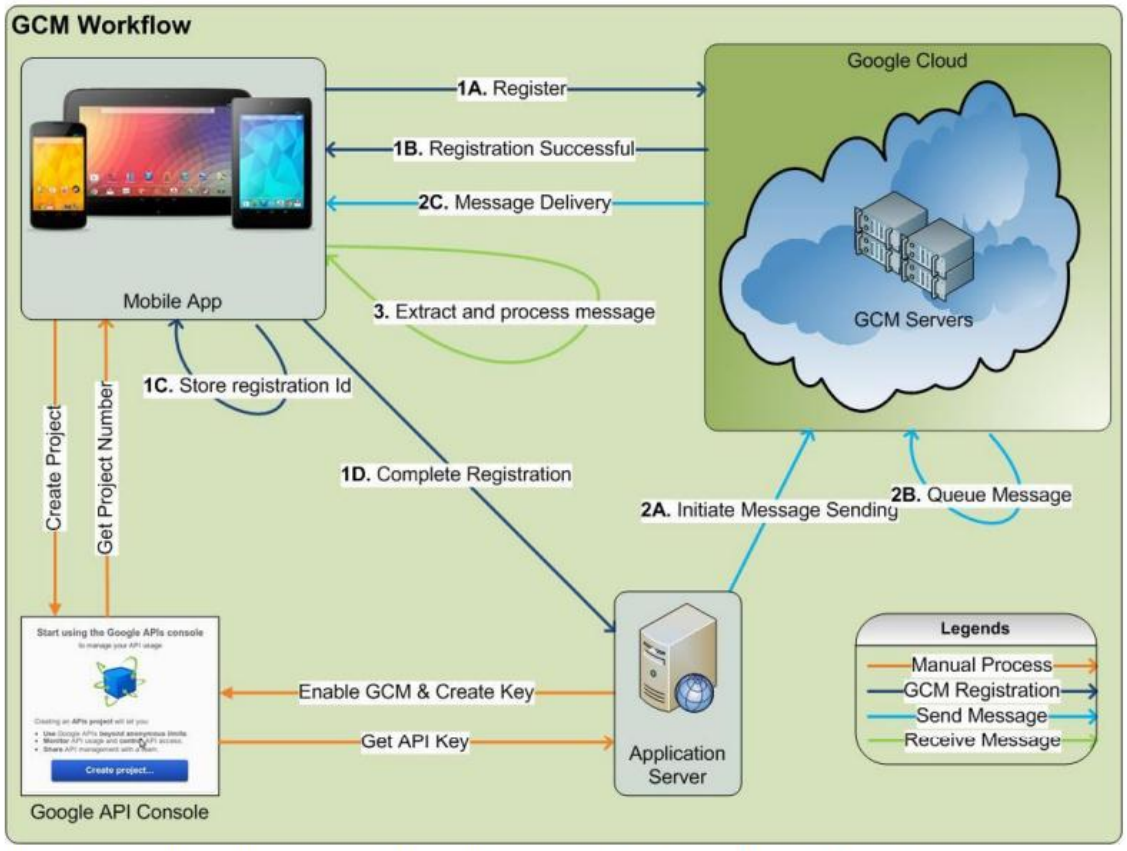
Approach 2: We can simulate push notification by opening TCP connection to the push server. As we know, TCP streams are full duplex (having pair of byte streams, one following each direction) and therefore the server can push data to device anytime, provided the stream is active.

B) GCM :

Another main research is on GCM - Google Cloud Messaging & how it works.

Fig 3.2 Graphical representation of GCM :

A representation of the GCM workflow is shown in the following diagram.



The diagram shows the entire GCM workflow, starting from GCM registration to actual message delivery.

C) RESTful Webservice:

Resource identification through URI – A RESTful Web service provides a set of resources which identify the targets of the interaction with its clients. These resources are identified by URIs.

D) Cloud storage application Server

Storage flows carry either store commands or retrieve commands. This permits storage flows to be divided in two groups by checking the amount of data downloaded and uploaded in each flow. By means of the data collected in test environment, documente the overhead of store and retrieve commands and derived a method for labeling the flows. Furthermore, identify a direct relationship between the number of TCP segments with the PSH flag set in storage flows and the number of transported chunks.

3.4 Advantage/Disadvantage of those research

3.4.1 Advantages :

- Cloud Storage presented an extensive characterization both in terms of the system workload as well as the typical usage.
- The usage of per-chunk acknowledgments in the client protocol combined with the typically small chunk sizes deeply limits the effective throughput of the service.
- Push Notification provides real time information update as soon as it is created.
- Provides richer set of meta data and travel over the data network.
- Improves battery life and consumes less CPU cycles.
- Same service for multiple applications with very little changes.

3.4.2 Disadvantages :

- For Cloud storage security is a major issue. We need more work on improving security of personal cloud storage.
- Push Notification is not feasible for the cases where data is not required immediately and one can achieve the desired output by using poll mechanism.

3.5 Our solution to solve this problem :

- Provide cloud storage service.
- Implement notification service using push mechanism.
- Develop an application to access the services.

3.6 Why this solution different from others :

Users of mobile devices may have applications installed that depend on remote services. Notification messages typically represent events of interest defined by the applications. Push notifications are SMS-like messages. Unlike their GSM counterparts, push notifications are addressed to an application not to a device or a phone number. They also typically provide richer set of metadata and travel over the data network. Push notifications are based on push technology as opposed to polling. Push is a style of communication where the request for a given transaction is initiated by the publisher. Underlying technology is a long-lived socket.

3.7 Why this solution is better

- Personal cloud is more reliable and available system.
- In push notification server only notifies clients when data changes in such case network bandwidth can efficiently be used.
- Such conditions also can improve battery life for wireless instruments.

4. Hypothesis

4.1 Positive hypothesis

Push notifications services have provided great convenience and flexibility for applications to receive light messages or notifications from application servers; GCM support android platform and third party application server in a high reliability and good performance. As a result our project should achieve goals as listed below:

- a) To provide a real-time updated application and storage which is good in availability, reliability, security.
- b) Support for the adding, deleting, and editing files via the cloud service with push notification.

4.2 Negative hypothesis

After our project comes to the end, we cannot guarantee:

- The availability of very huge number of clients' notification;
- The security while GCM was facing hacking and GCM has it's own bug.
- The reliability while the clients encountered network interruption or other technical problems.

5. Methodologies

5.1 How to generate/collect input data

Basically we generate or collect files in type which covers all the usual types as listed below:
JPEG, GIF, BMP, PNG, PDF, DOC, DOCX, MP3, APE.
Also generate or collect files with variable sizes.

5.2 How to solve the problem

5.2.1 Algorithm Design

The application server sends a message to GCM servers. Google enqueues and stores the message in case the device is offline. When the device is online, Google sends the message to the device. On the device, the system broadcasts the message to the specified Android application via Intent broadcast with proper permissions, so that only the targeted Android application gets the message. This wakes up the Android application and the application does not need to be running before to receive the message. Finally, the Android application processes the message.

5.2.2 Pseudo Code:

- Load web client for cloud storage
  Implement "doget" method to load web client.
- Check number of registered devices.

```
if (devices.isEmpty())
 {
 out.print(" No devices registered  ");
                    }
                    else {
 Send All Method to post Upload File…
            }
```

- File Upload
  Call for File Upload.

```
// Initialization
String fName=fileupload(req,resp);
// Find Content Part for uploading file
response.setContentType("text/html");
PrintWriter out = response.getWriter();
Collection<Part> parts = request1.getParts();
```

```
out.write(" Total parts : "+parts.size()+"");
                    // Fetch File Name & all the Content
String fName=null; for(Part part : parts)
{
fName = printPart(part, out);
part.write(fName);
}
```

- File Delete
- Find Registration ID for device

```
String registrationId = devices.get(0);
```

- Successful Upload Push Notification

```
// Generate Push Notification Message
Message message = new Message.Builder().addData("FileName", fName).build();
// Send to registered Device
Result result = sender.send(message, registrationId, 5);
```

- Successful Delete Push Notification

5.3 Language Used

5.3.1 HTML

HyperText Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed on a web browser.

5.3.2 Java Enterprise Edition (HTTP Servlet)

We get the Java EE 7 SDK here: http://www.oracle.com/technetwork/java/javaee/ downloads/. The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

In Java EE concepts, the “web tier” consists of components that handle the interaction between clients and the business tier. Its primary tasks are the following:

- Dynamically generate content in various formats for the client.
- Collect input from users of the client interface and return appropriate results from the components in the business tier.
- Control the flow of screens or pages on the client.

- Maintain the state of data for a user's session.
- Perform some basic logic and hold some data temporarily in JavaBeans components.

The servlets technology's purpose is providing Java programming language classes that dynamically process requests and construct responses, usually for HTML pages.

5.3.3 XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. Many application programming interfaces (APIs) have been developed to aid software developers with processing XML data, and several schema systems exist to aid in the definition of XML-based languages.

5.4 Tool Used

5.4.1 GCM Server APK :

Third-party Application Server: An application server that developers set up as part of implementing GCM in their applications. The Third-party Application server sends data to an Android application on the device via the GCM server.

5.4.2 Credentials:

The IDs and tokens that are used in different stages of GCM to ensure that all parties have been authenticated, and that message is going to the correct place.

- Sender ID: The sender ID is used in the registration process to identify an Android application that is permitted to send messages to the device.
- Application ID: The Android application that is registering to receive messages. The Android application is identified by the package name from the manifest. This ensures that the messages are targeted to the correct android application.

- Registration ID: An ID issued by the GCM servers to the Android application that allows it to receive messages. Once the Android application has the registration ID, it sends it to the third-party application server, which uses it to identify each device that has registered to receive messages for a given Android application. The registration ID is tied to a particular Android application running on a particular device.

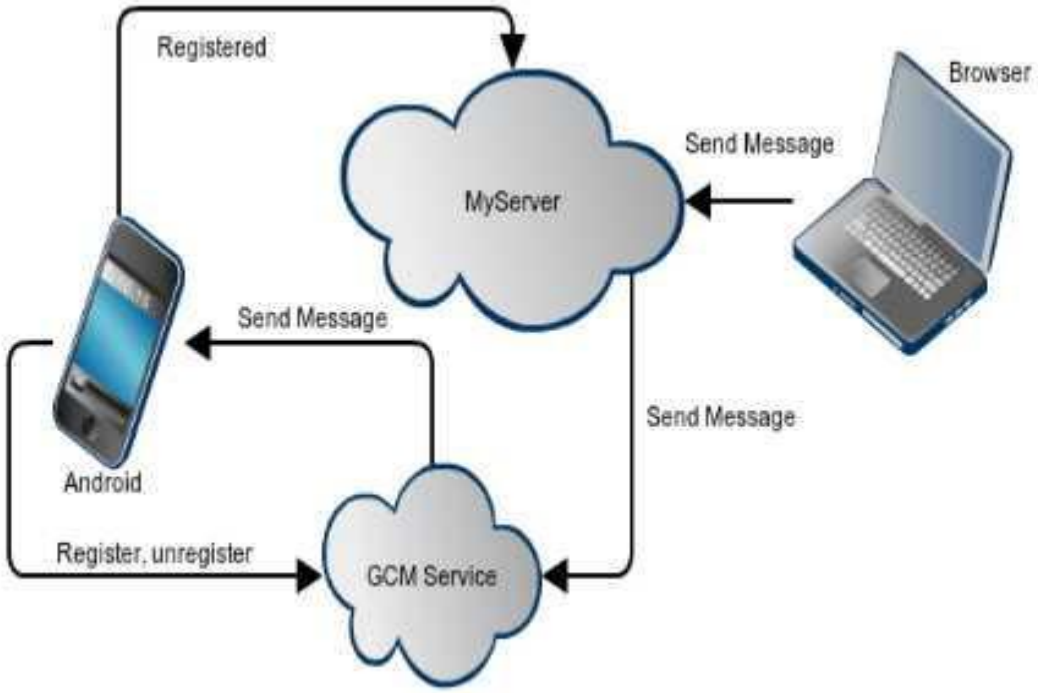


Fig 5.1 -The diagram which indicates the implementation components' relation and the result test topology

- Google User Account: For GCM to work, the mobile device must include at least one Google account if the device is running a version lower than Android 4.0.4.
- Sender Auth Token: An API key that is saved on the 3rd-party application server that gives the application server authorized access to Google services. The API key is included in the header of POST requests that send messages.

5.4.3 Apache Tomcat Server

Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in.

Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

A Server element represents the entire Catalina servlet container. Therefore, it must be the single outermost element in the config/server.xml configuration file. Its attributes represent the characteristics of the servlet container as a whole.

5.4.4 Android SDK

The Android SDK provides us the API libraries and developer tools necessary to build, test, and debug apps for Android. We can download ADT (Android Developer Tools) from here: http://developer.android.com/sdk/index.html

5.4.5 Eclipse

We use Eclipse IDE to build client side application. Eclipse is created by an Open Source community and is used in several different areas, e.g. as a development environment for Java or Android applications. The roots of Eclipses go back to 2001.

The Eclipse Open Source community has over 200 Open Source projects covering different aspects of software development.

5.4.6 Ant

We selected Apache Ant to build server side application. Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++

applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks. Ant is written in Java. Users of Ant can develop their own "antlibs" containing Ant tasks and types, and are offered a large number of ready-made commercial or open-source "antlibs". Ant is extremely flexible and does not impose coding conventions or directory layouts to the Java projects which adopt it as a build tool. Software development projects looking for a solution combining build tool and dependency management can use Ant in combination with Apache Ivy. The Apache Ant project is part of the Apache Software Foundation.

5.5 Implementation source

5.5.1 GCM Server

To develop Android applications that use GCM, we must have an application server. The two primary steps involved in writing a client Android application are:

- Creating a manifest that contains the permissions, the Android application needs to use GCM.
- To use GCM in an application implementation, it must include:
  - Code to start and stop the registration service.
  - Receivers for the *com.google.android.c2dm.intent.RECEIVE* and *com.google.android.c2dm.intent.REGISTRATION* intents.

Fig 5.2 - The Google APIs console

5.5.2 Apache Tomcat Server

The following links provide access to selected sources of online information, documentation, and software that is useful in developing web applications with Tomcat.

- http://jcp.org/aboutJava/communityprocess/mrel/jsr245/ - JavaServer Pages (JSP)
- http://jcp.org/aboutJava/communityprocess/mrel/jsr315/

5.5.3 Android SDK

Use http://developer.android.com/sdk/index.html link to download Android SDK to write HTTP servlet java code . Install whole SDK package , develop dynamic web project & implement functionality class.

5.6 How to generate output

Steps to generate the output:

1. Upload or delete the added files on the server.
2. The server should push notify the application regarding any upload/deletion of file.
3. Receiving push notification should be working correctly although the application hasn't running beforehand and wake up once the notification comes.
4. Receive push notification on the client device with the appropriate alert message.
5. View the updated storage media on application running on client device incase of any upload/delete of files.

5.7 How to proof correctness

By uploading multiple files, modifying them and deleting existing files, then tester check whether all these changes are reflected well in on test client device. If everything reflects correctly and exactly, the correctness is proved.

6 Implementation

6.1 Code:

Live working code for this application has been provided online.

6.2 Design Documentation

6.2.1 Implementation:

We have used methodology for cloud storage & push notification. Multiple components, tools & run time environment. Here we have work flow & methodology we implemented this application.

1. GCM Server Configuration

Google-provided servers take messages from the 3rd-party application server and sending them to the device. And it is google-provided servers which provide connection servers.

1.1.Authentication

A POST request to https://android.googleapis.com/gcm/send from the application server is for sending a message.

A message request = HTTP header + HTTP body.

HTTP header:

- Content-Type: application/json for JSON; application/x-www-form-urlencoded; charset =UTF-8 for plain text.
- Authorization: key=YOUR_API_KEY

1.2.Request Format

The source website which describes the detailed methodology has noted that, while deploying or implementing GCM HTTP server, the user should check for the firewalls. For our projects we just close the firewall application on the PC.

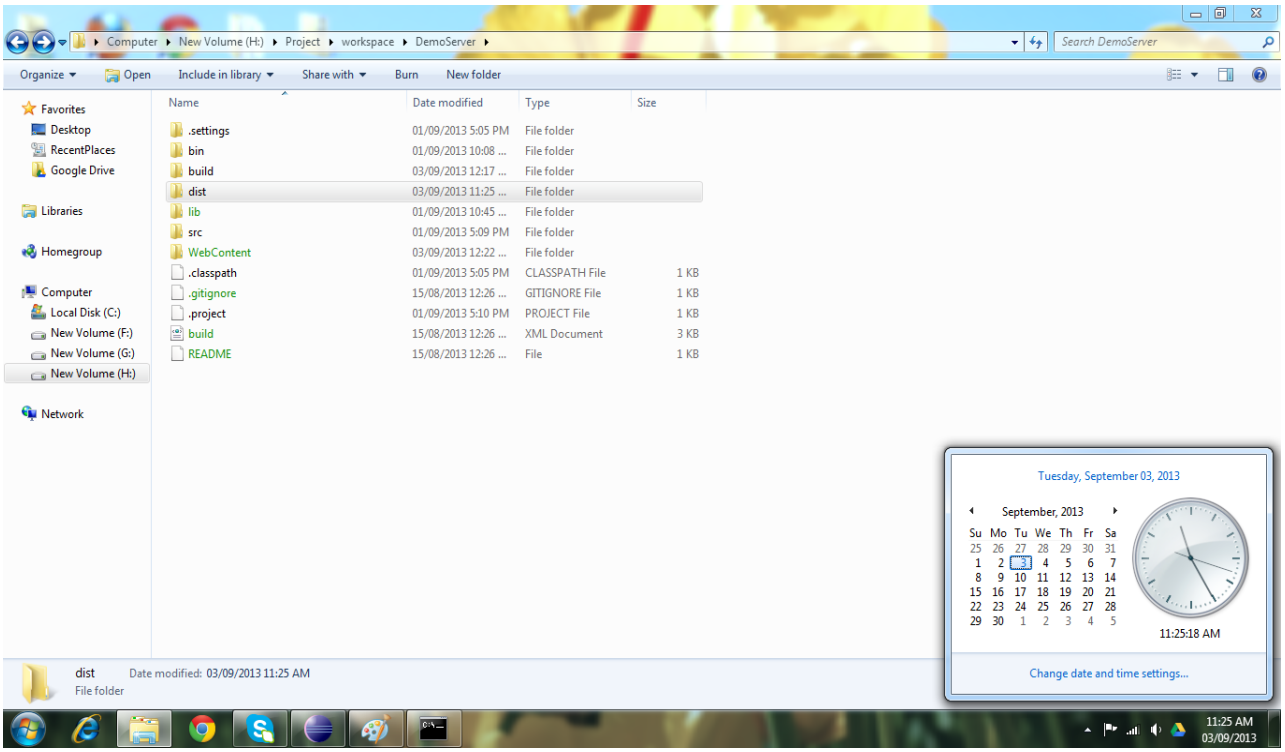
1.3.Response Format

There are two possible outcomes when trying to send message: the message is processed successfully or GCM server rejects the corresponding request. For the former, the http response has a 200 status and the body contains more information about the status of the message(including possible errors); But for the later one, the http response contains a non-200 status code. Following table is used as the indicator of response number:

Response	Description
200	Message was processed successfully. The response body will contain more details about the message status, but its format will depend whether the request was JSON or plain text.
400	Only applies for JSON requests. Indicates that the request could not be parsed as JSON, or it contained invalid fields (for instance, passing a string where a number was expected). The exact failure reason is described in the response and the problem should be addressed before the request can be retried
401	There was an error authenticating the sender account. Troubleshoot
5xx	Errors in the 500-599 range (such as 500 or 503) indicate that there wa an internal error in the GCM server while trying to process the request, or that the server is temporarily unavailable (for example, because of timeouts). Sender must retry later, honoring any Retry-Afterheader included in the response. Application servers must implement exponential back-off.

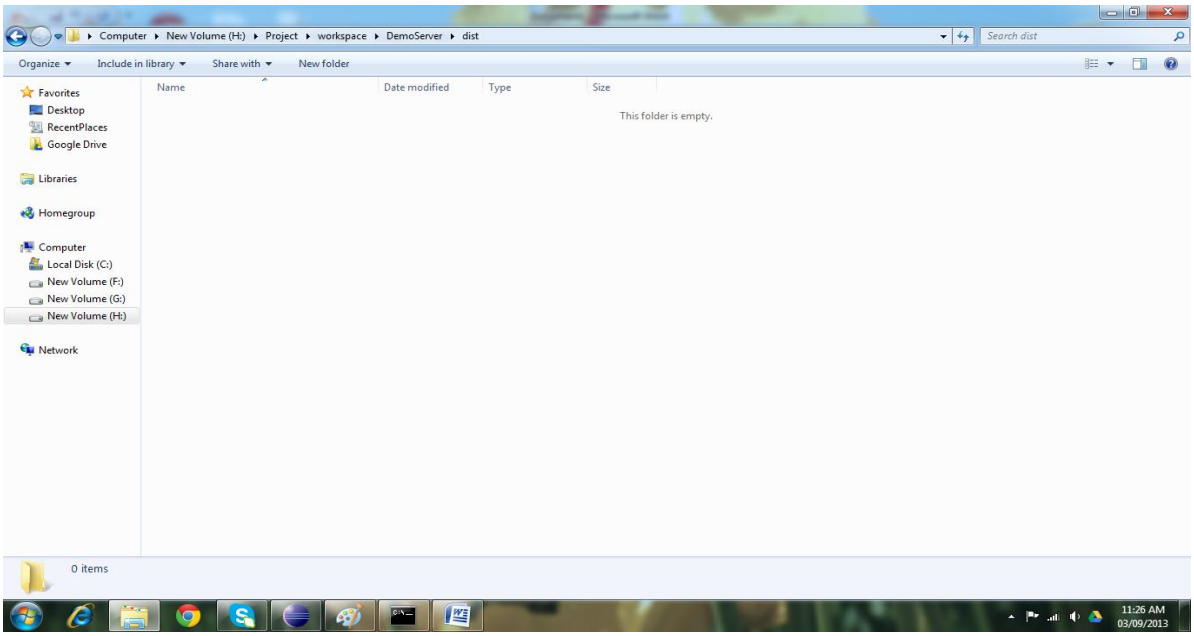
2. HttpServlet Level Implementation

2.1.Structure of HTTP Servlet



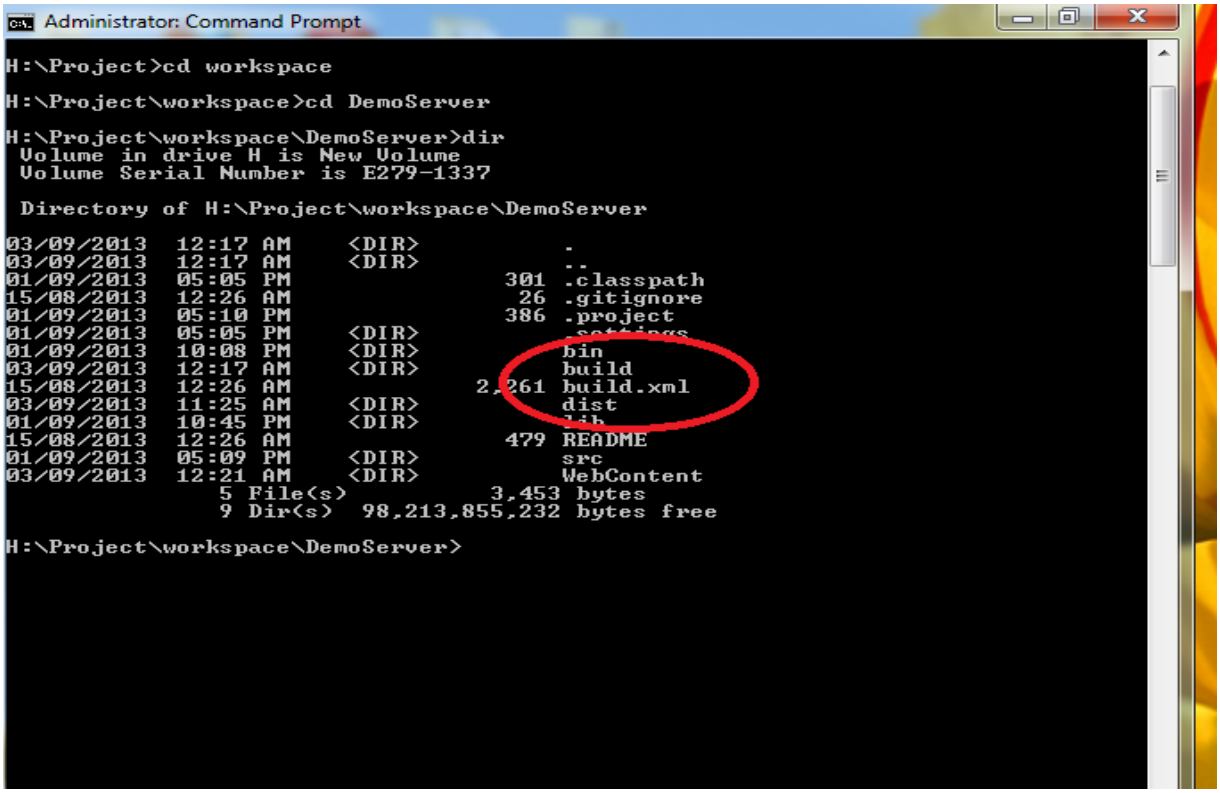
2.2. Generate War File

2.2.1. No War File – Empty Dist Folder



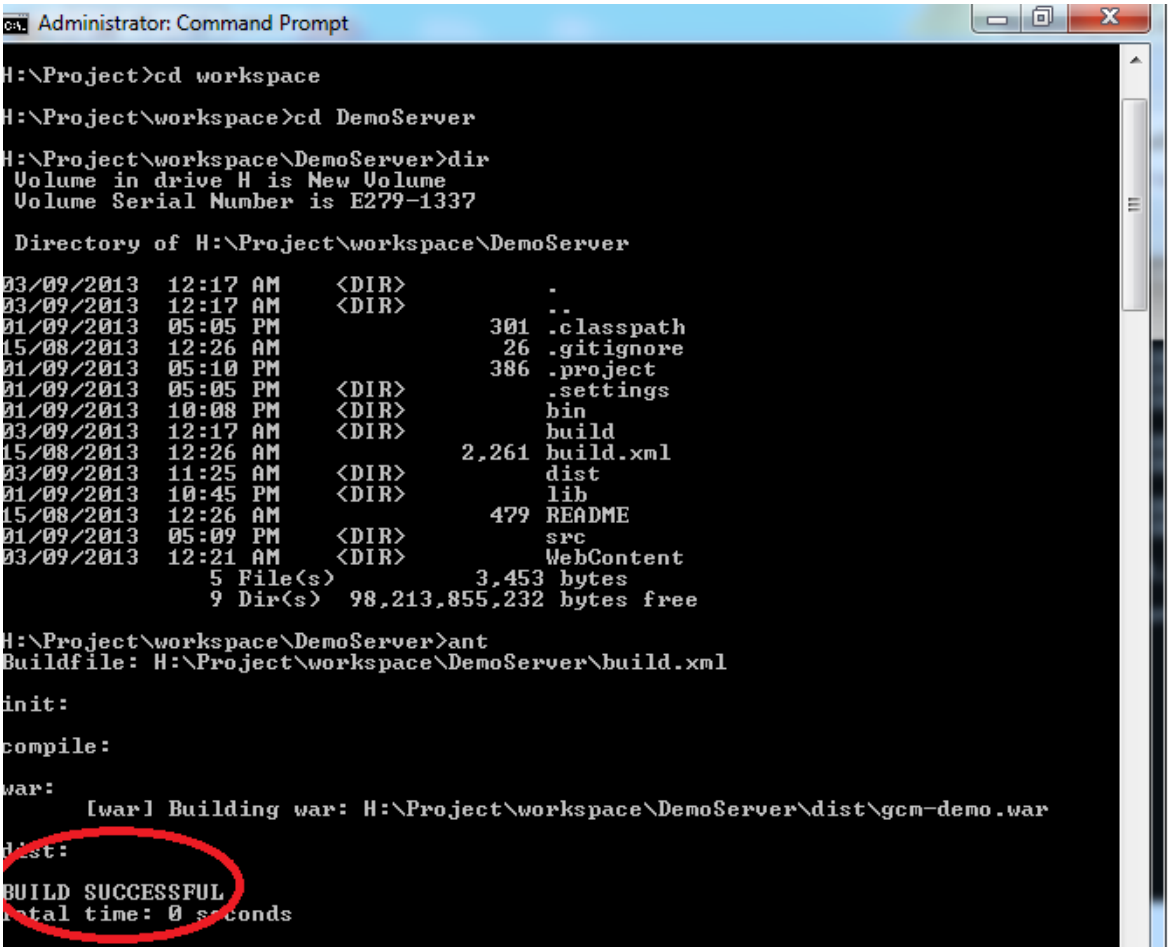
2.2.2. Generate WAR file by building application – using ant

- Ant command works only in the directory where build.xml is present or else gives error.

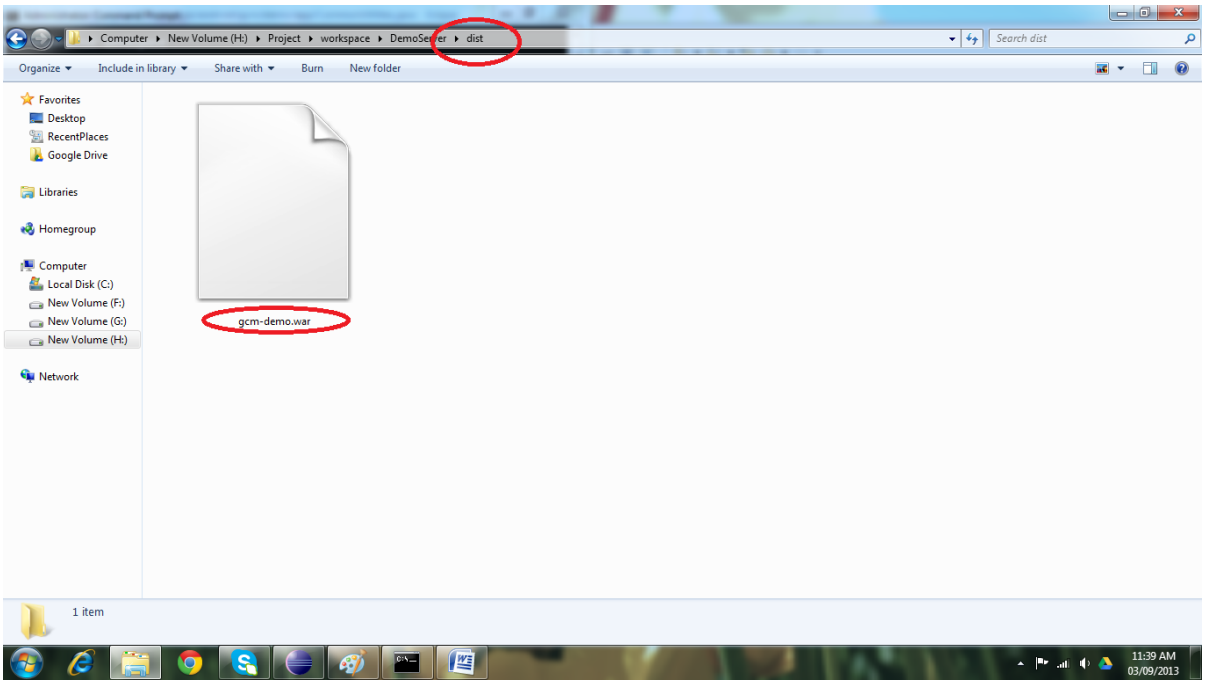


Now run ant Command :

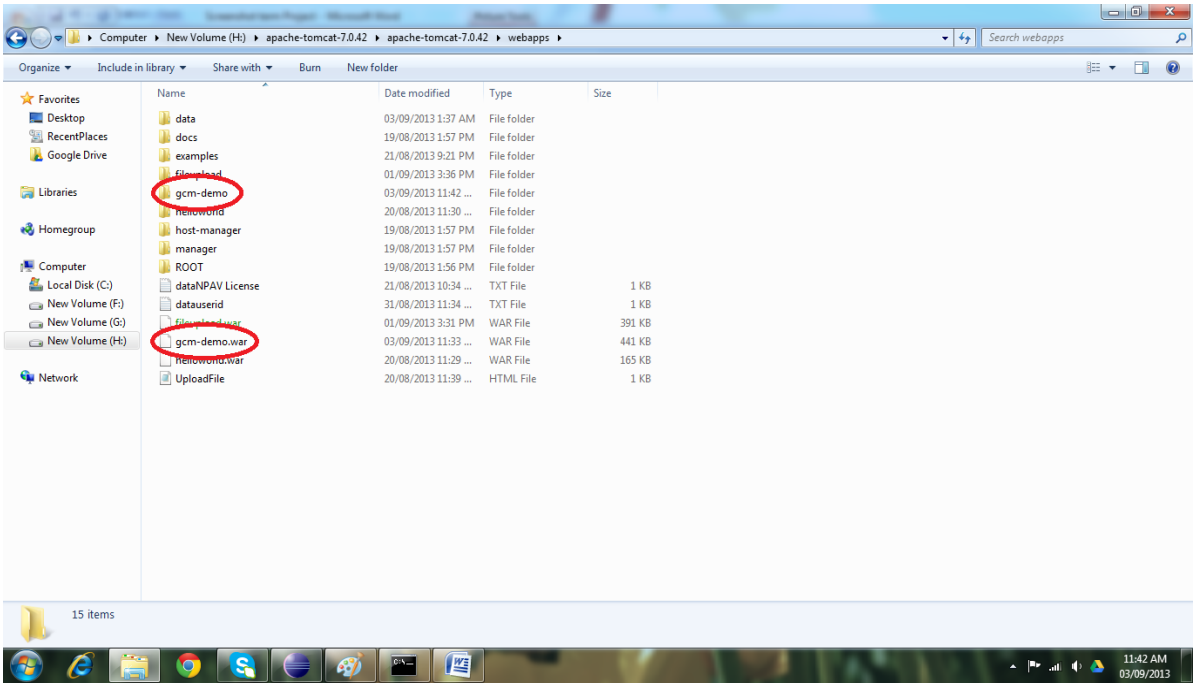
- H:\Project\workspace\DemoServer> ant where H:\Project\workspace\DemoServer> is the path where build.xml is present as shown above.



- The below screen shows the successful generation of war file.

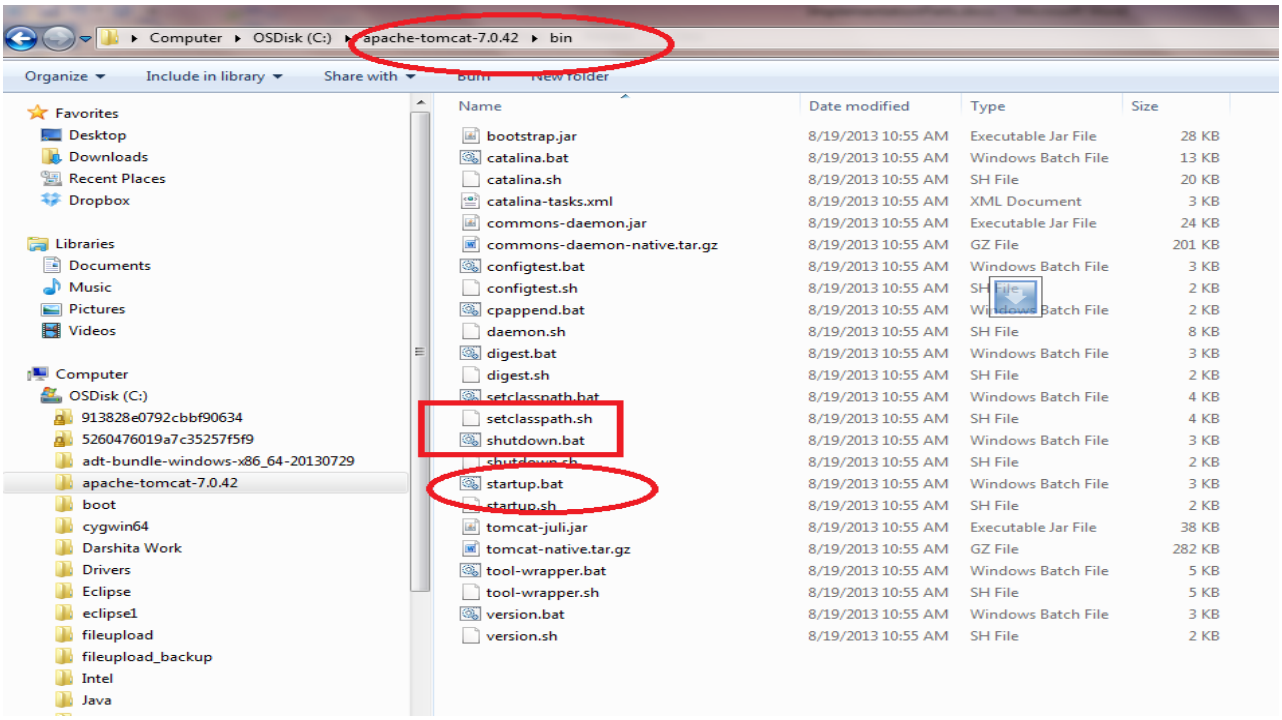


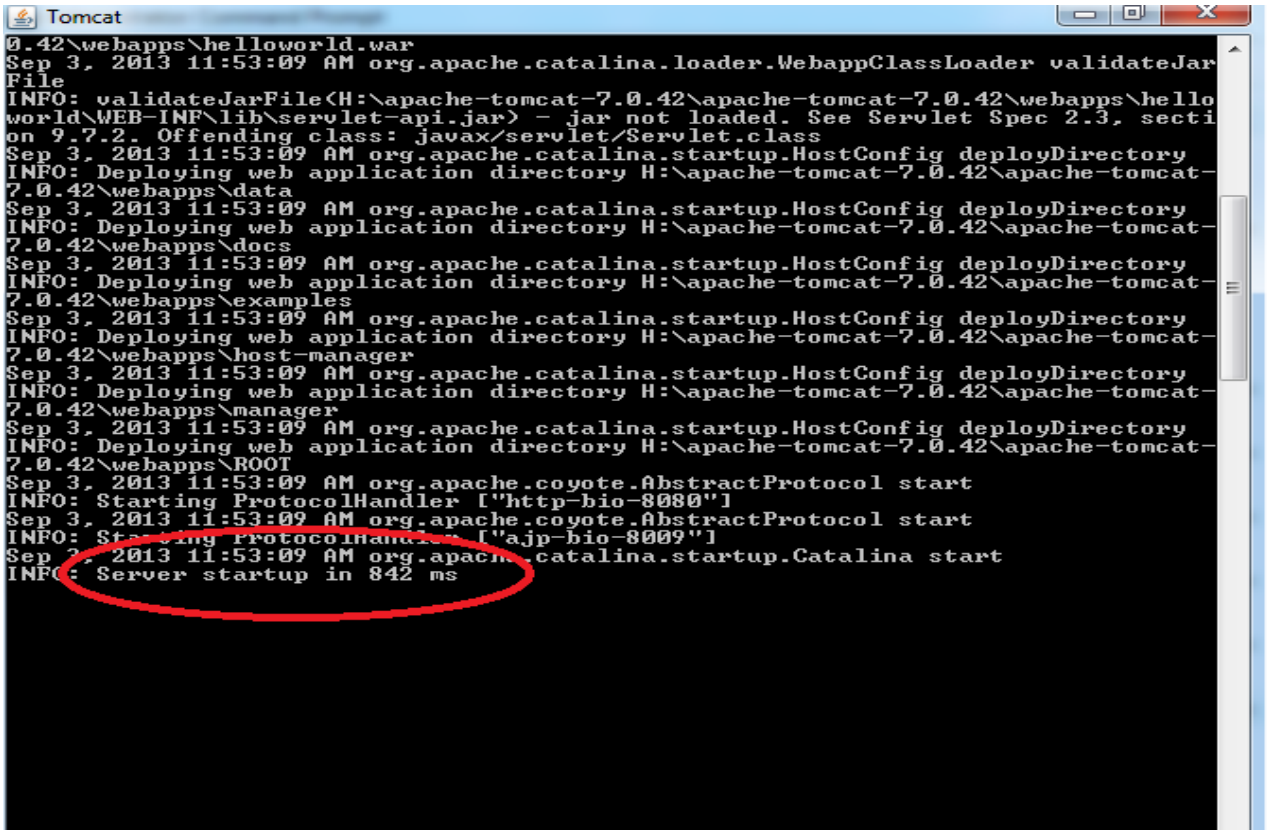
2.2.3. Copy the generated WAR file in the server's webapps folder and run the server using startup.bat present in the bin directory.

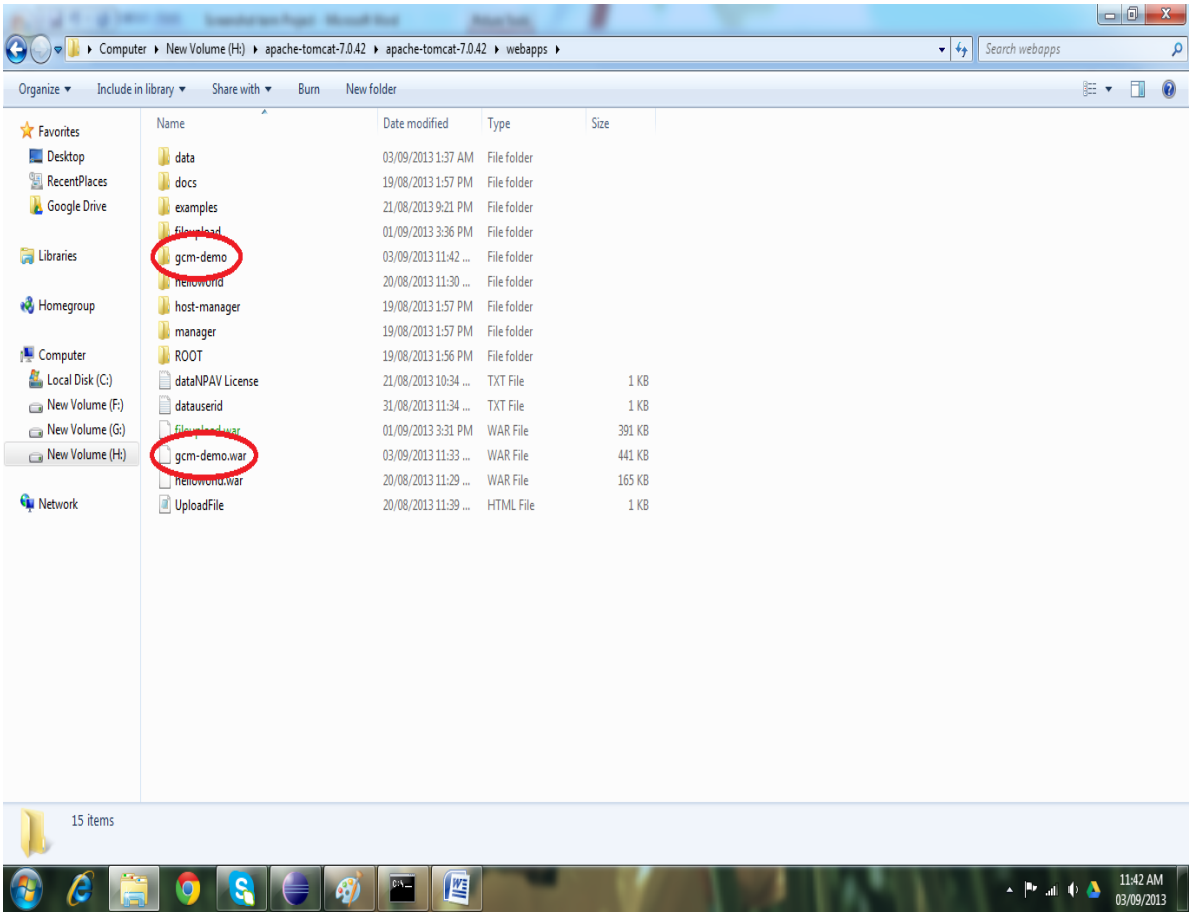


3. Apache Tomcat Server Implementation:

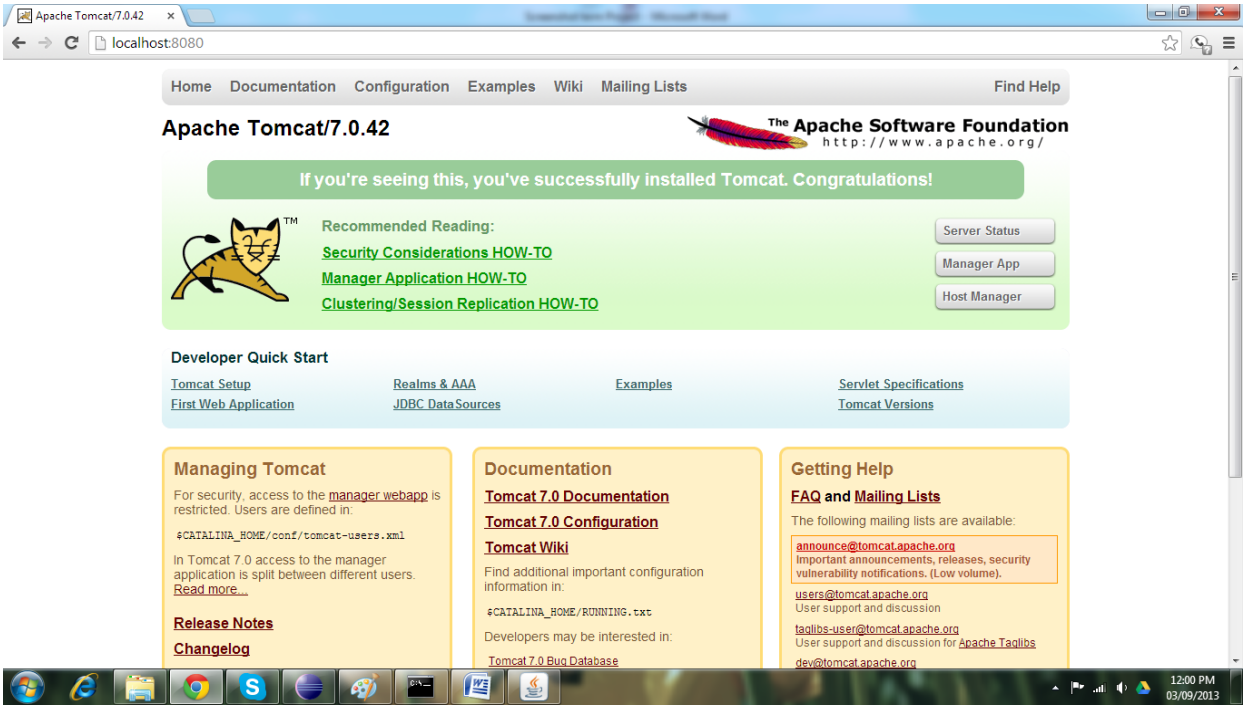
3.1.Run Tomcat Server



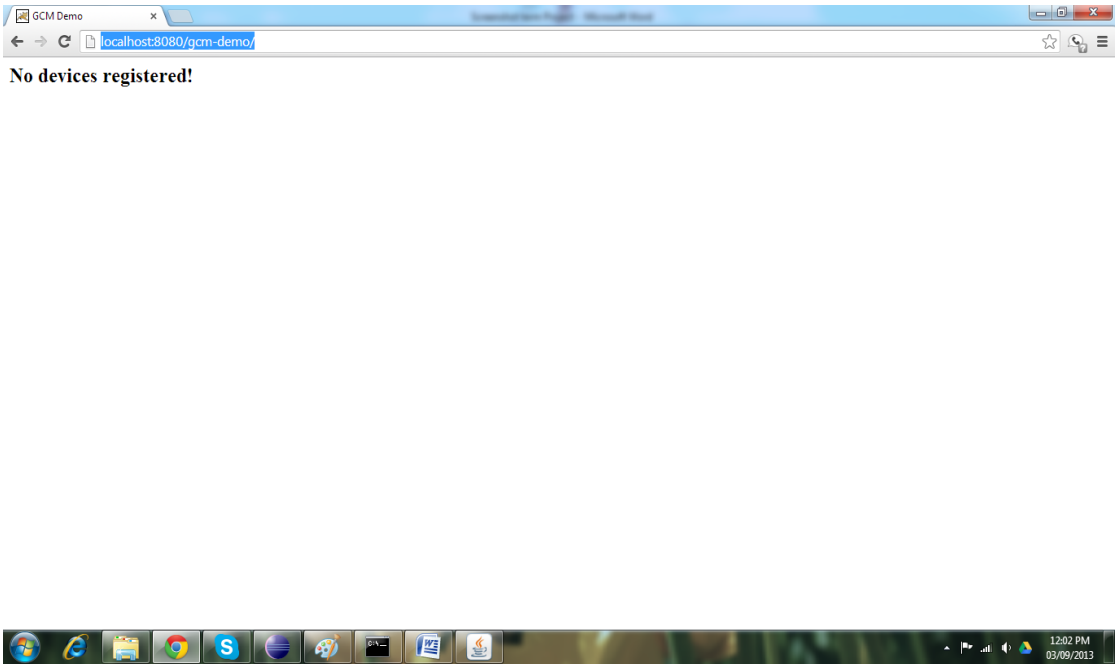



Computer ▸ New Volume (H:) ▸ apache-tomcat-7.0.42 ▸ apache-tomcat-7.0.42 ▸ webapps ▸
Search webapps
Organize
Include in library
Share with
Burn
New folder
Favorites
Desktop
RecentPlaces
Google Drive
Libraries
Homegroup
Computer
Local Disk (C:)
New Volume (F:)
New Volume (G:)
New Volume (H:)
Network
Name
Date modified
Type
Size
data 03/09/2013 1:37 AM File folder
docs 19/08/2013 1:57 PM File folder
examples 21/08/2013 9:21 PM File folder
gcm-demo 03/09/2013 11:42 ... File folder
host-manager 19/08/2013 1:57 PM File folder
manager 19/08/2013 1:57 PM File folder
ROOT 19/08/2013 1:56 PM File folder
dataNPAV License 21/08/2013 10:34 ... TXT File 1 KB
datauserid 31/08/2013 11:34 ... TXT File 1 KB
01/09/2013 3:31 PM WAR File 391 KB
gcm-demo.war 03/09/2013 11:33 ... WAR File 441 KB
20/08/2013 11:29 ... WAR File 165 KB
UploadFile 20/08/2013 11:39 ... HTML File 1 KB
15 items
11:42 AM
03/09/2013

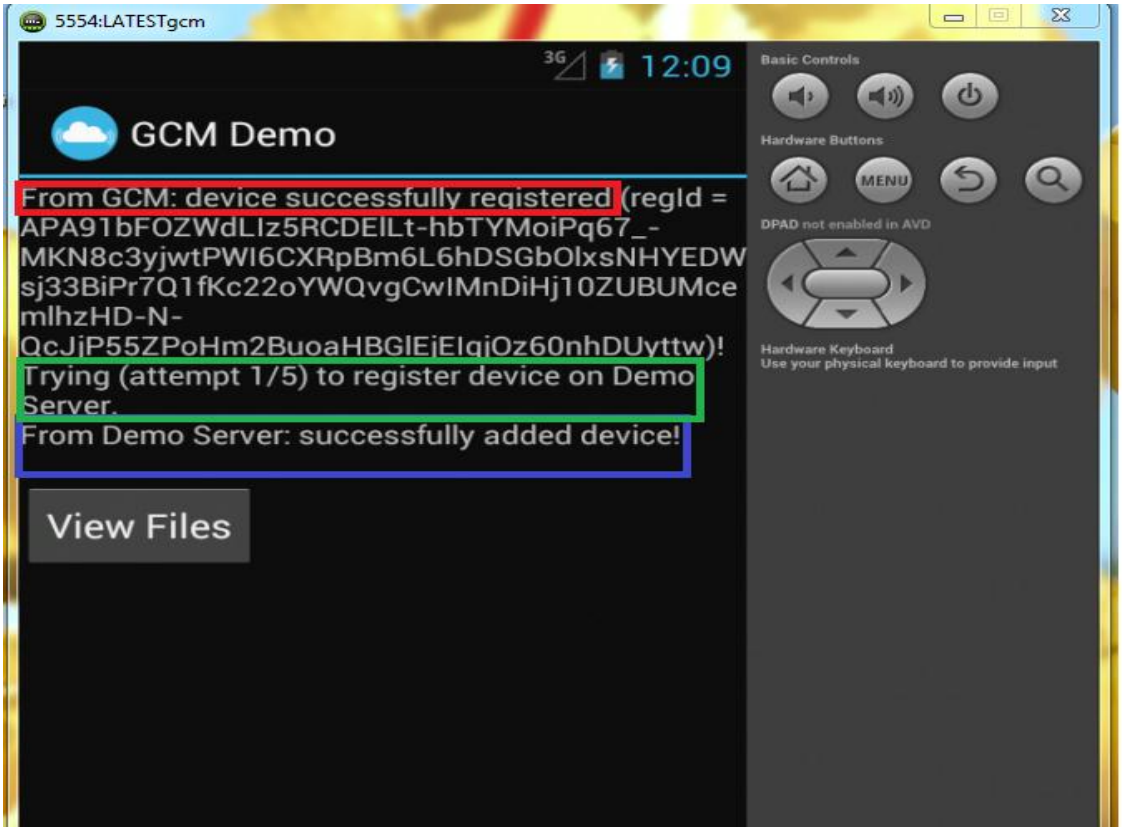
4. Web browser Implementation



- Run our gcm-demo application : Using the address http://localhost:8080/gcm-demo/
- Currently no devices our registered with our server so it prints: No devices registered!

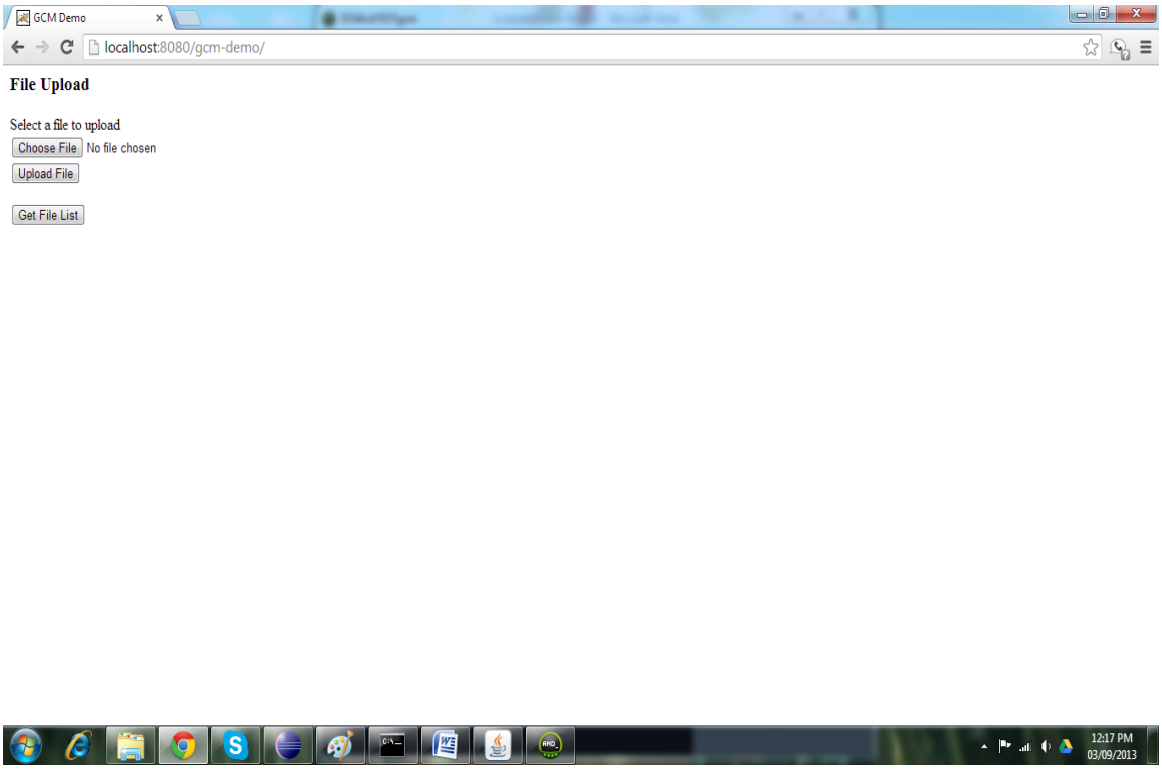


5. Client Device Registration
    - Install the application on the client's device. The application after running will connect to the application's server and will register for the push notification service.

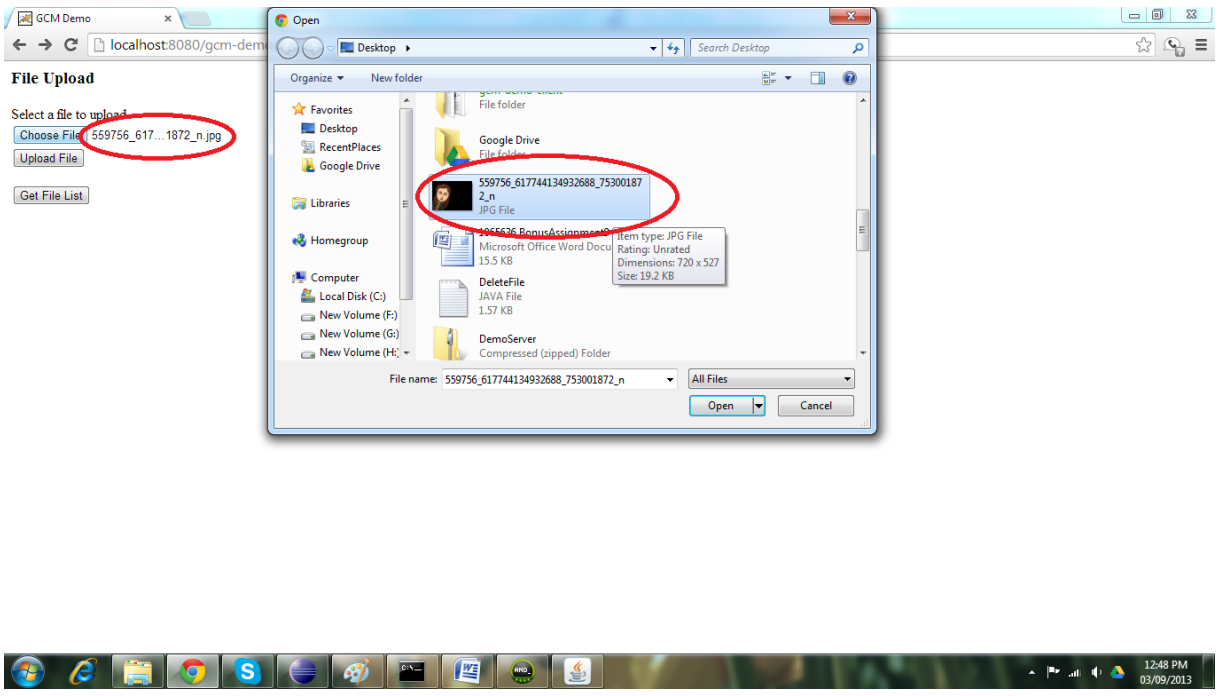


- After successful registration the user can upload file through the web browser.

6. Upload File Function

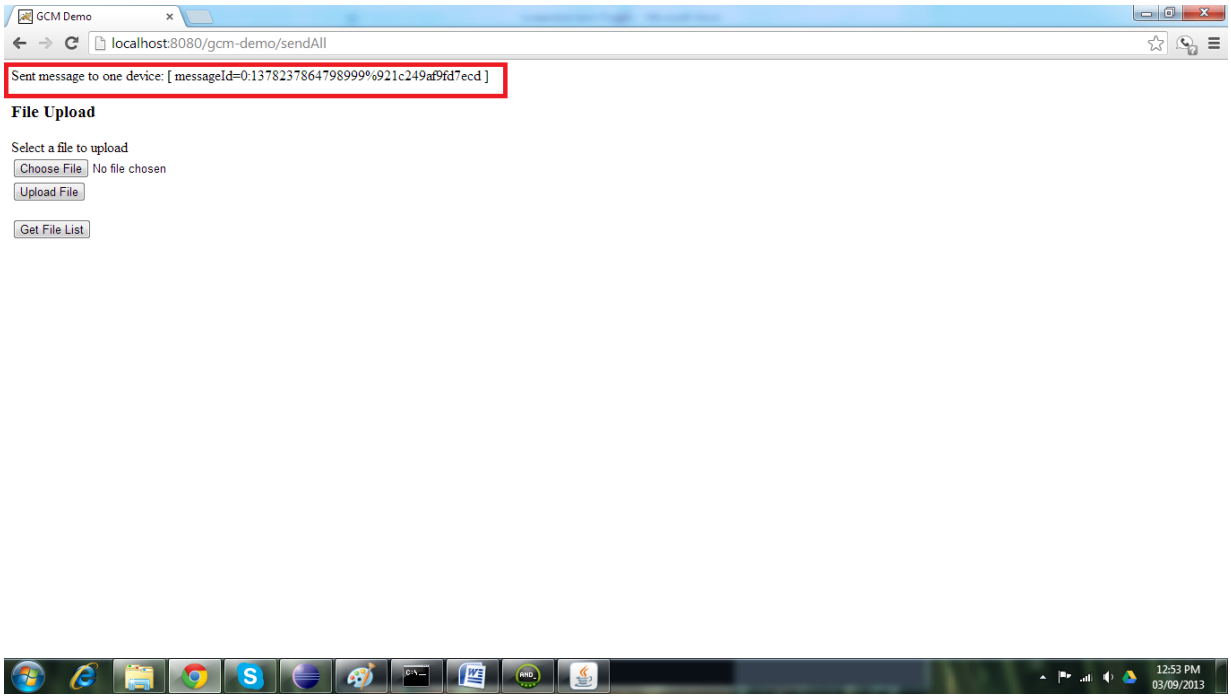


- Click on file upload and select the file to be uploaded.

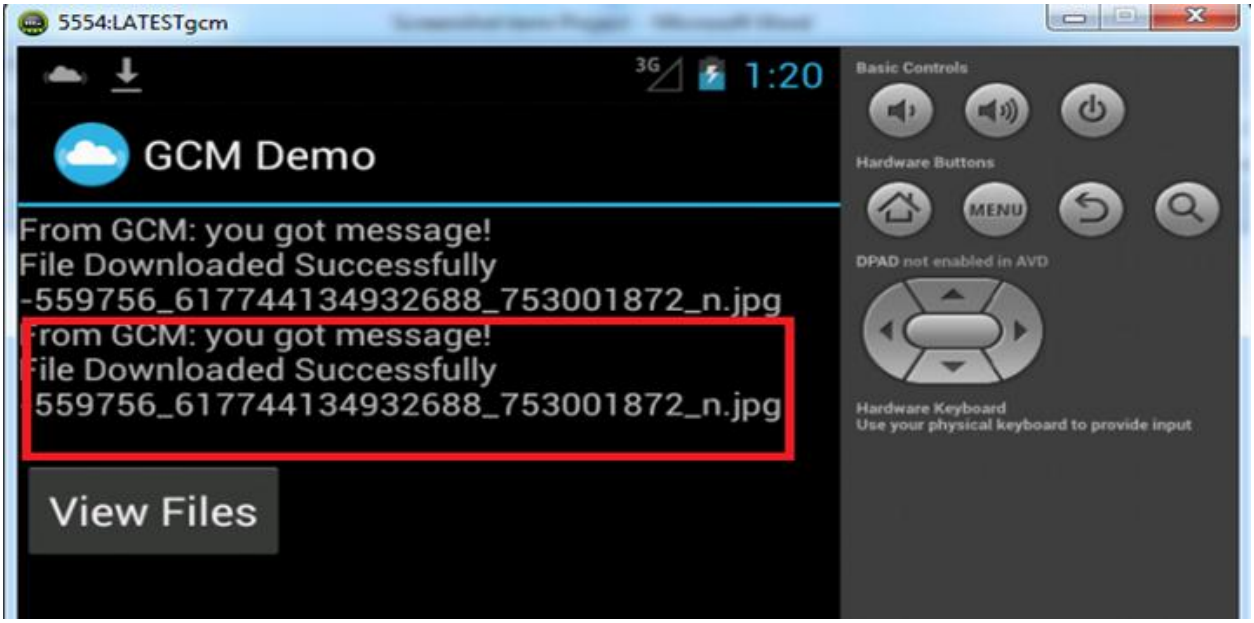


7. Upload file Push Notification

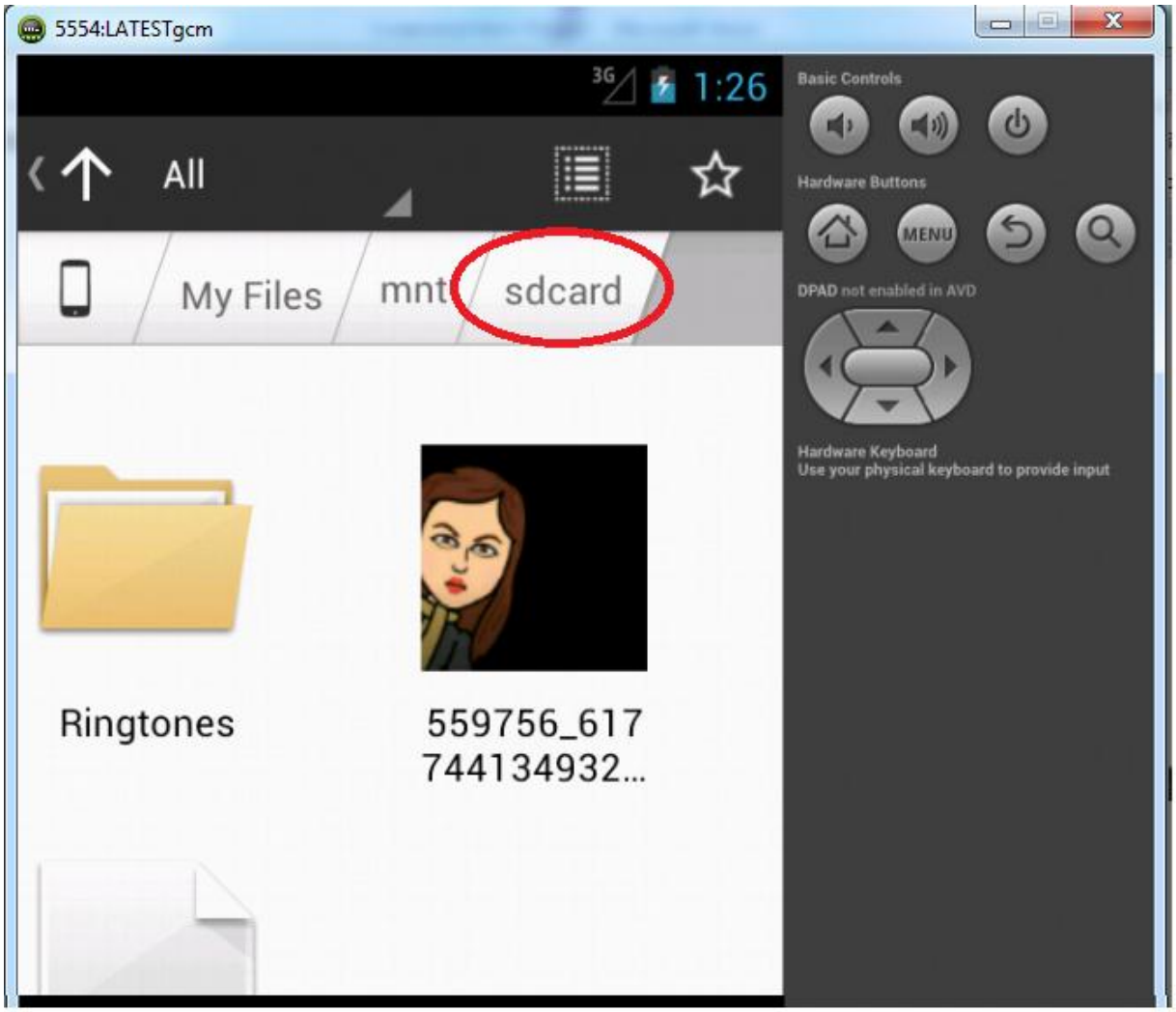
- It will send a “Push Notification to the client device” and display “Sent message to one device ” with the registration ID.



The client will receive push notification message and file will be downloaded automatically.



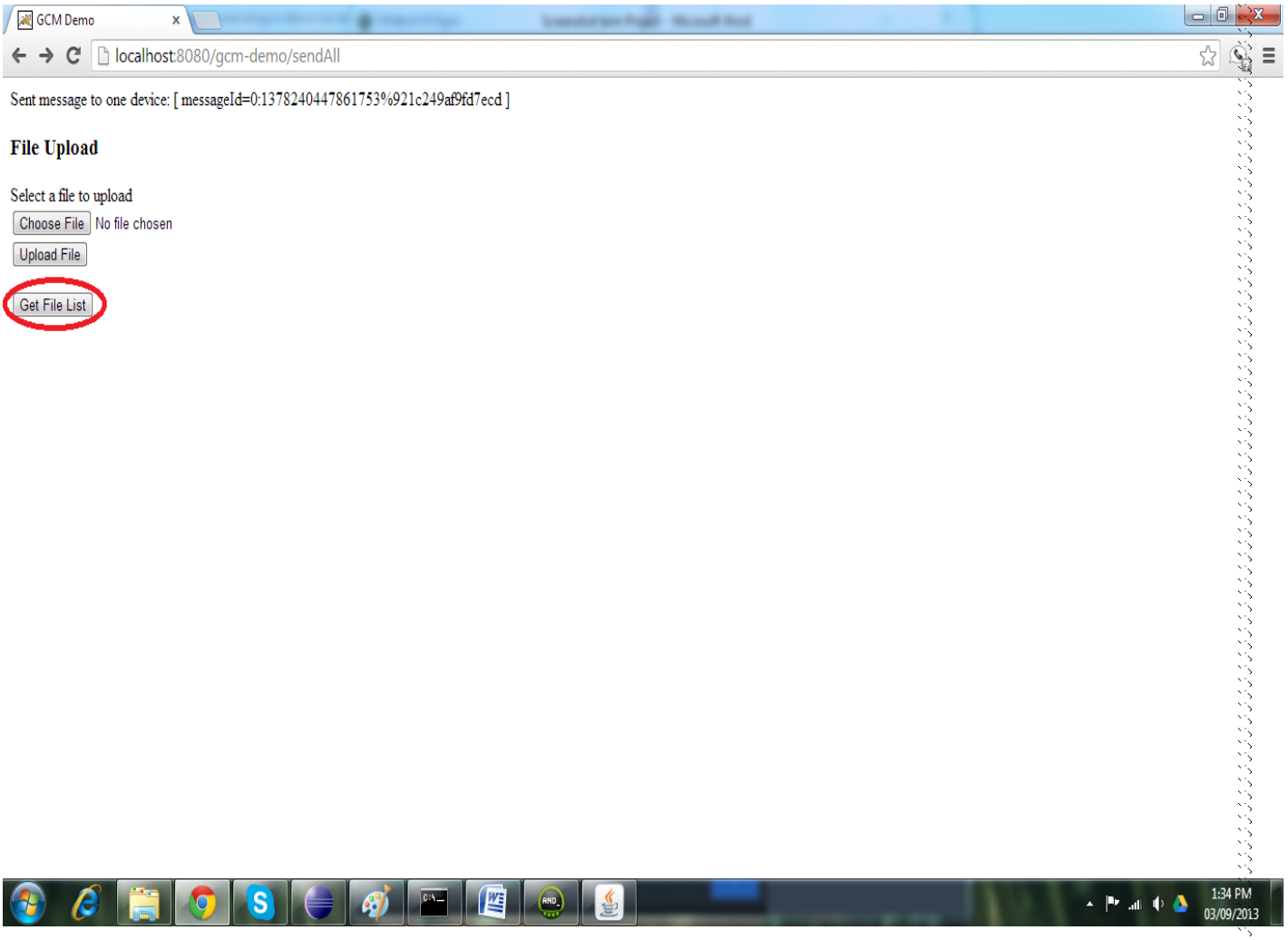
8. Download file from Client Device:
   The downloaded file will be stored in the SD card of client device.



- The file can be downloaded and deleted explicitly through web browser.

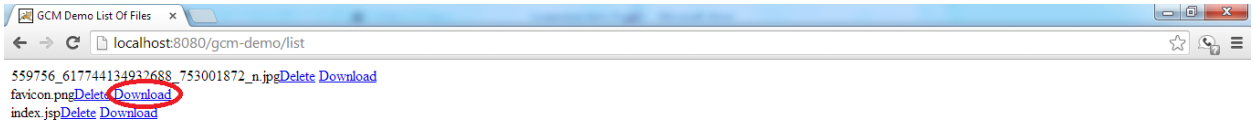
9. List of File

- Click on get file list.



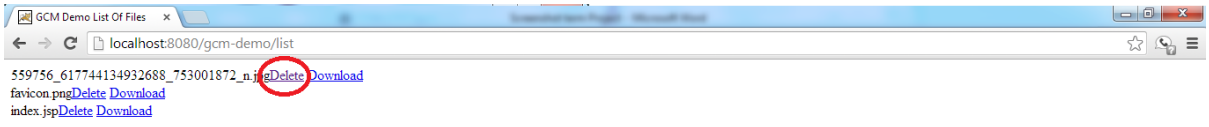
10. Download Uploaded files

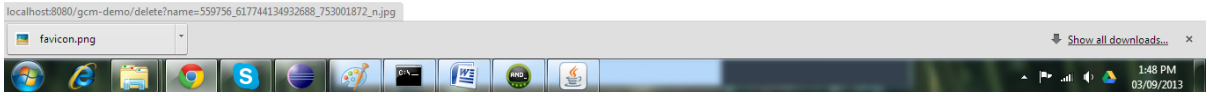
The user can download any file by clicking on download.



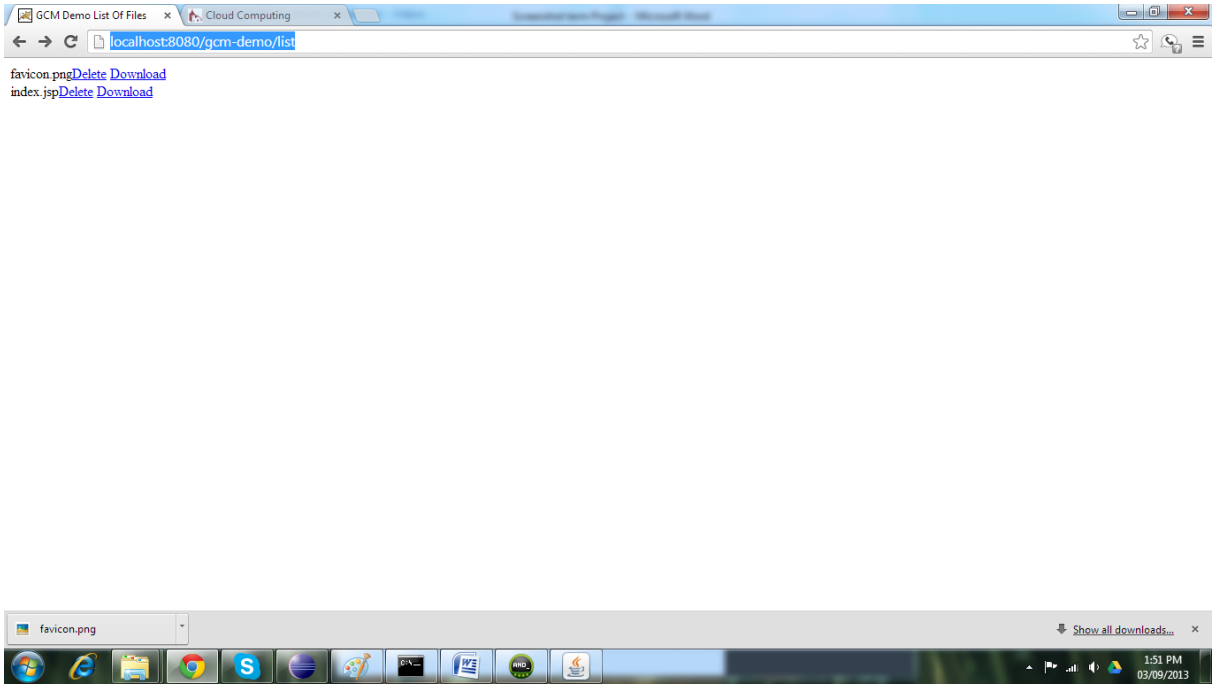
11. Delete Uploaded File

- The user can also delete any file by clicking on delete.





- Now go to the link again to view list of files http://localhost:8080/gcm-demo/list. The deleted file will not be shown any more.

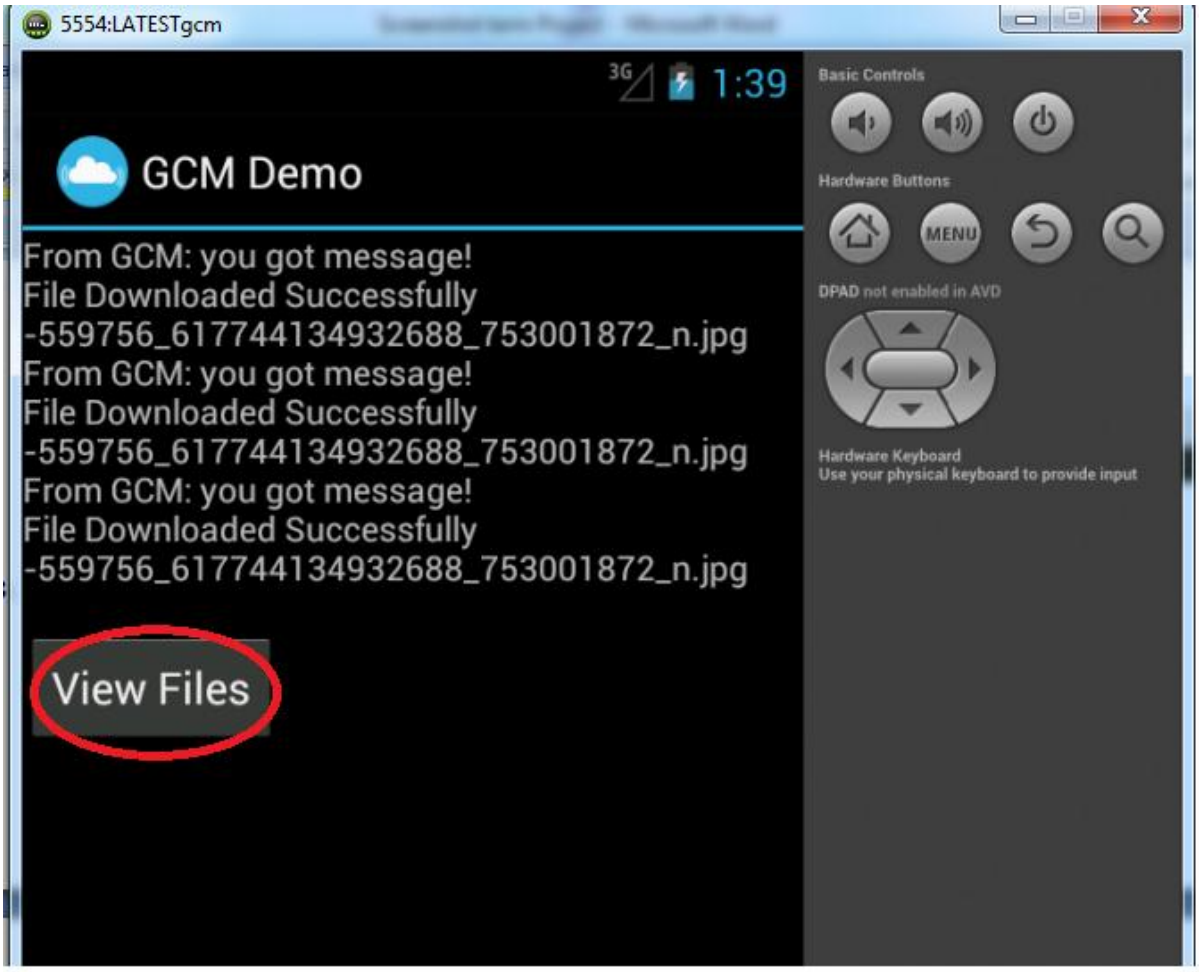


The file can also be downloaded and deleted explicitly through client device.

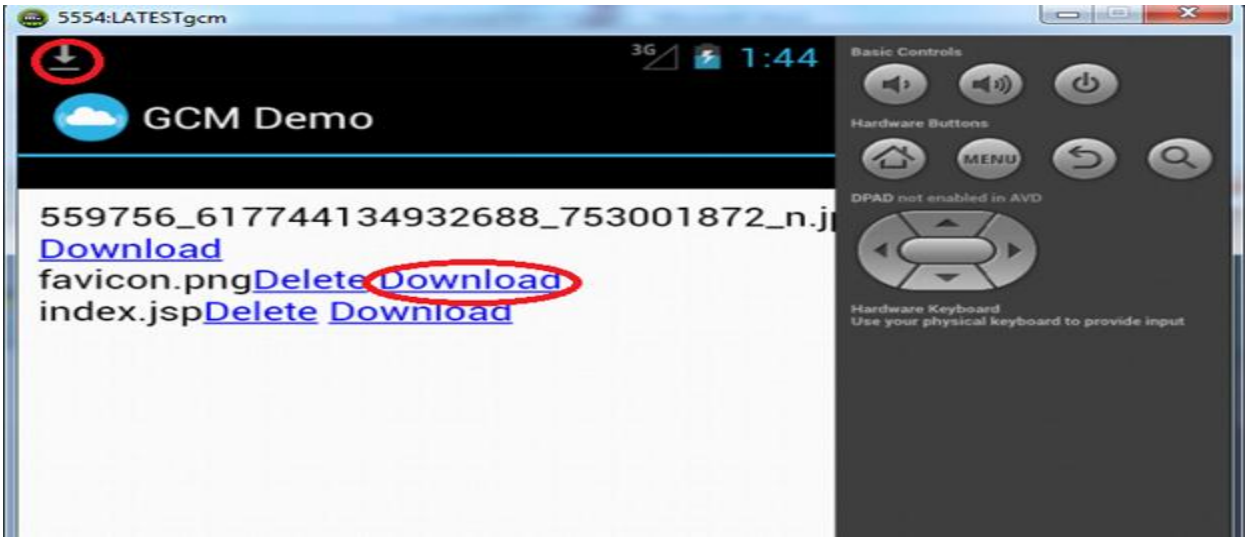
12. Client Device access :

12.1. View File :

12.1.1. Click on View Files to get the list of files uploaded on server.

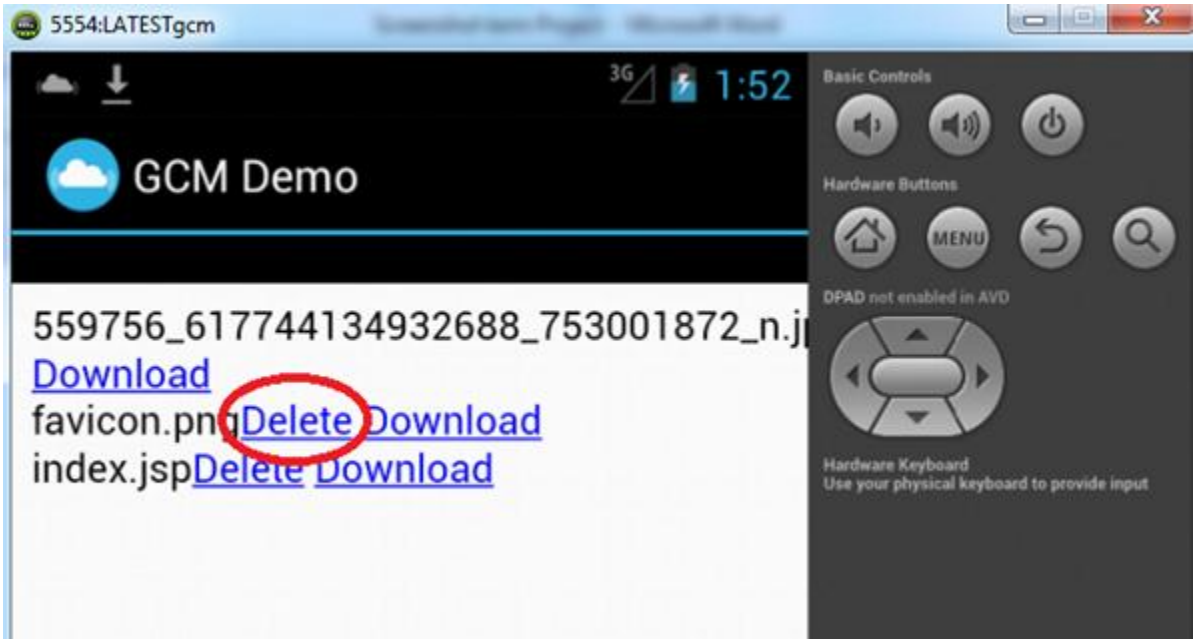


12.1.1.2 The user can download any file by clicking on download.

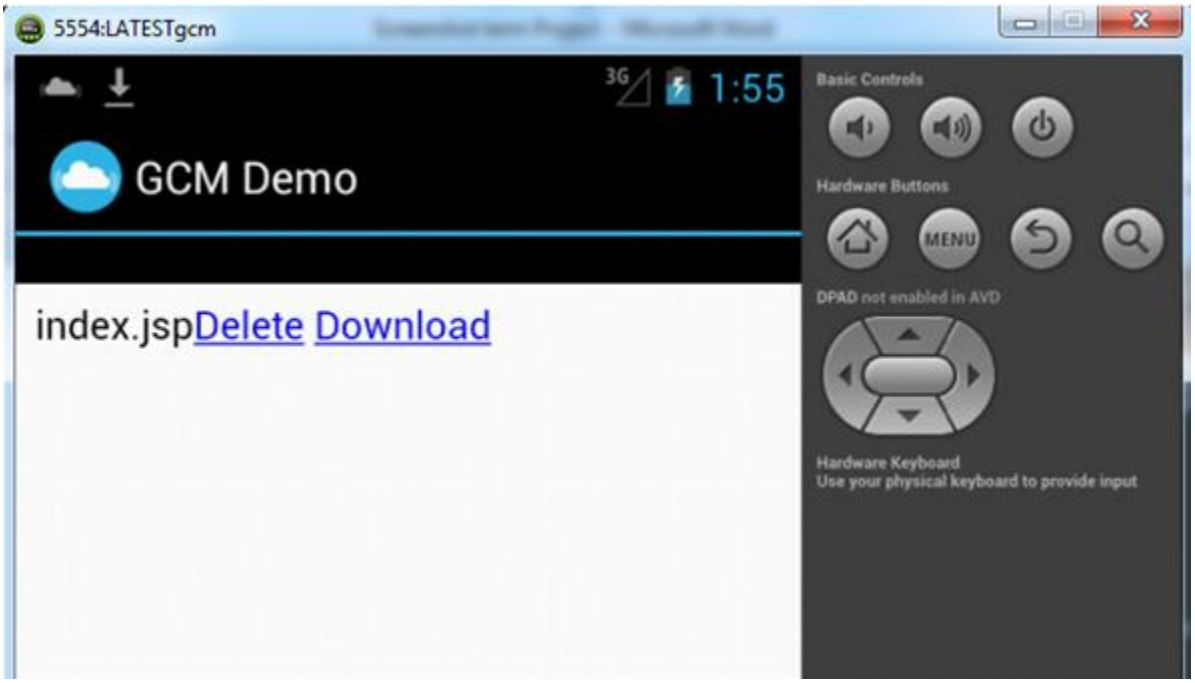


12.1.2. Client Device File Delete :

- The user can delete any file by clicking on delete.



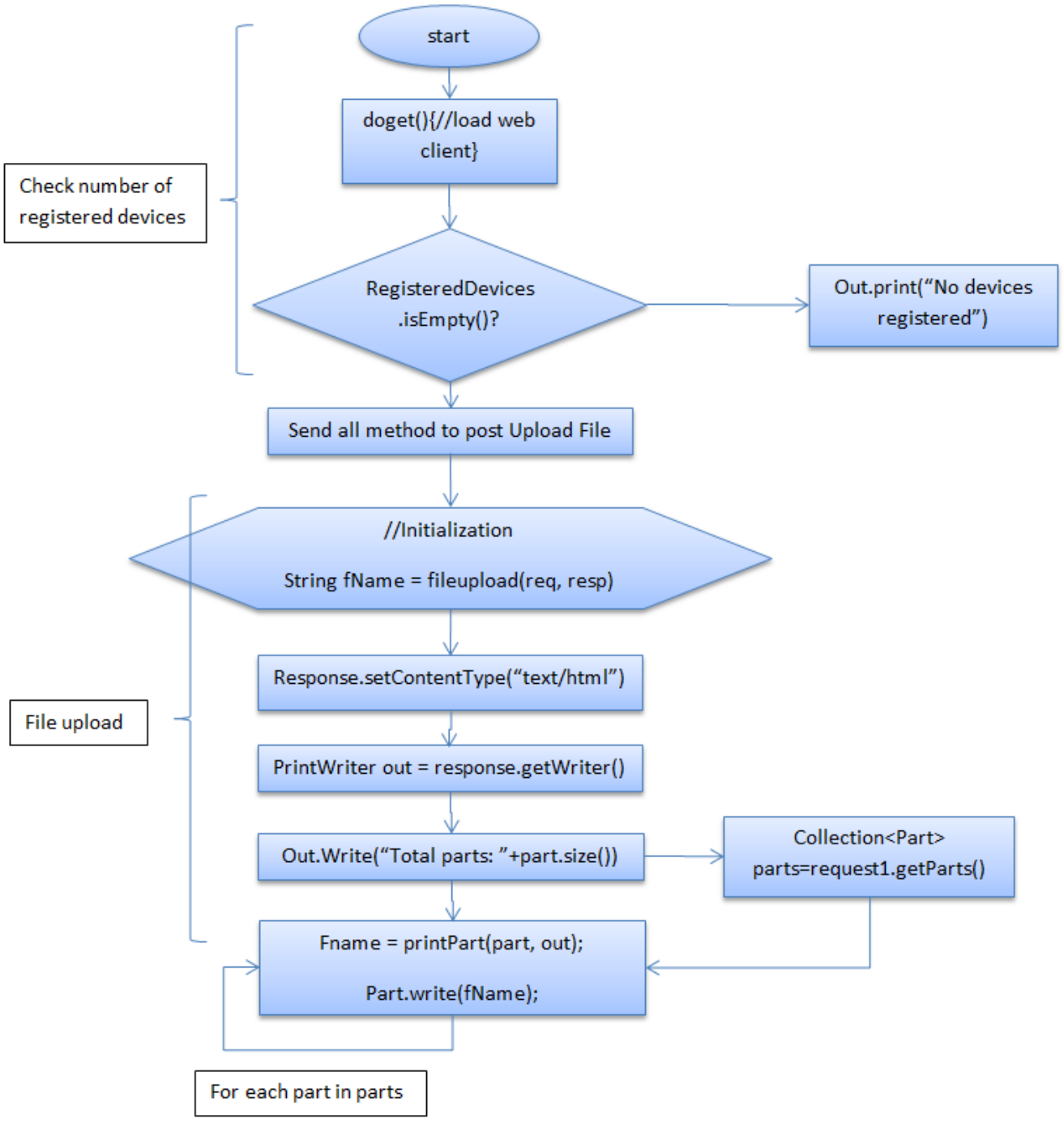
- Go to home page and click “View Files” button again. We can see the updated list of files.



12.2 Flow Chart

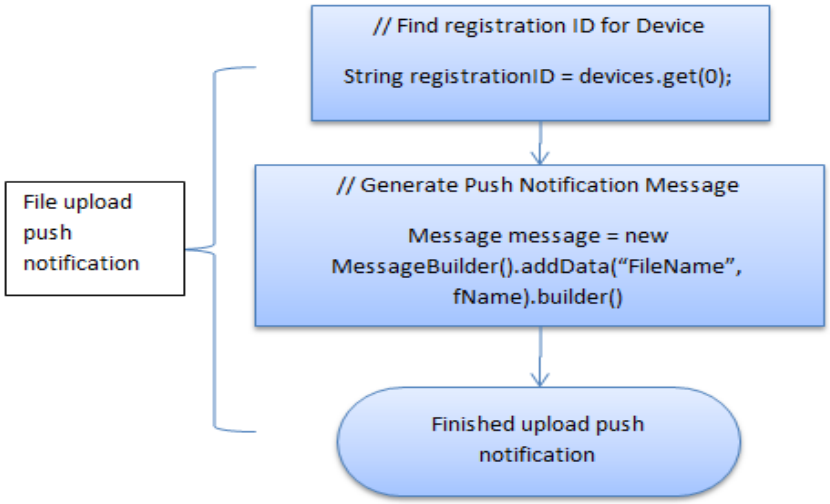
Below is the flow chart of the file upload procedure:

Fig 6.1



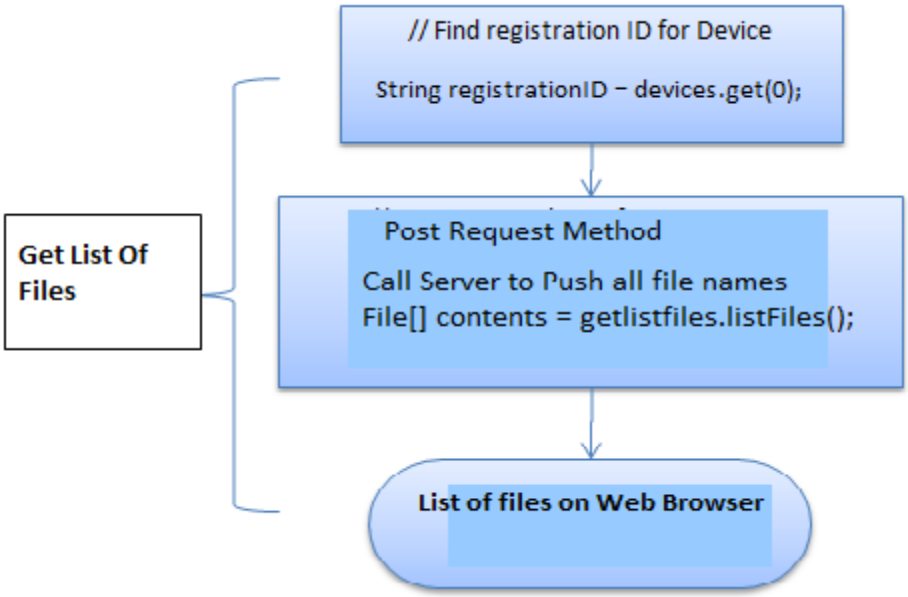
Below is the flow chart of the file upload push notification procedure:

Fig 6.2 Below is the flow chart for get list of uploaded files.



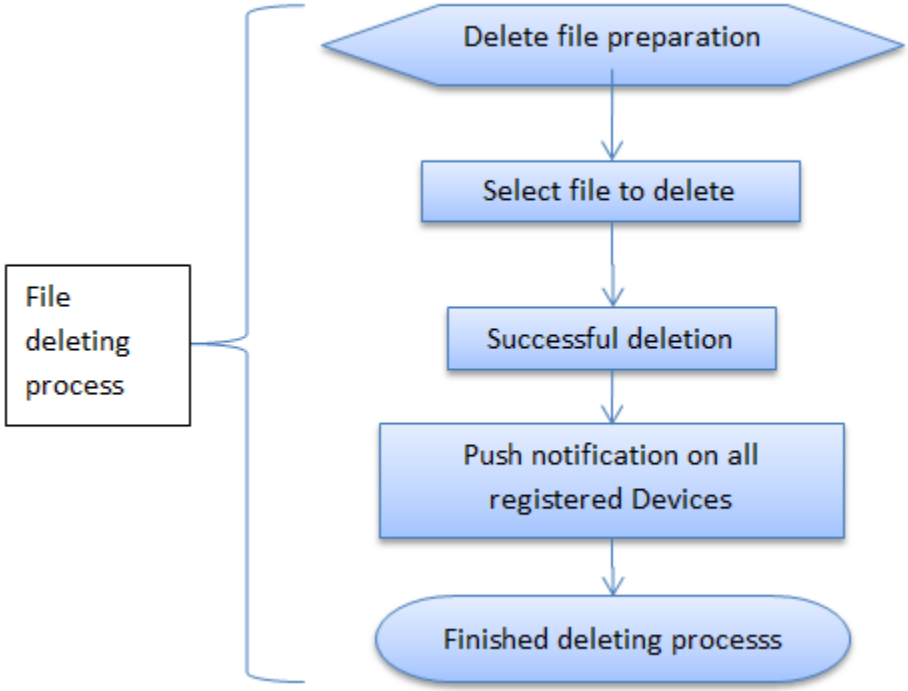
- Below flowchart shows how list of files are available to view on web browser as well as client end.

Fig 6.3



Below is the flow chart for file deletion operation:

Fig 6.4



Below is the flow chart for file download operation .

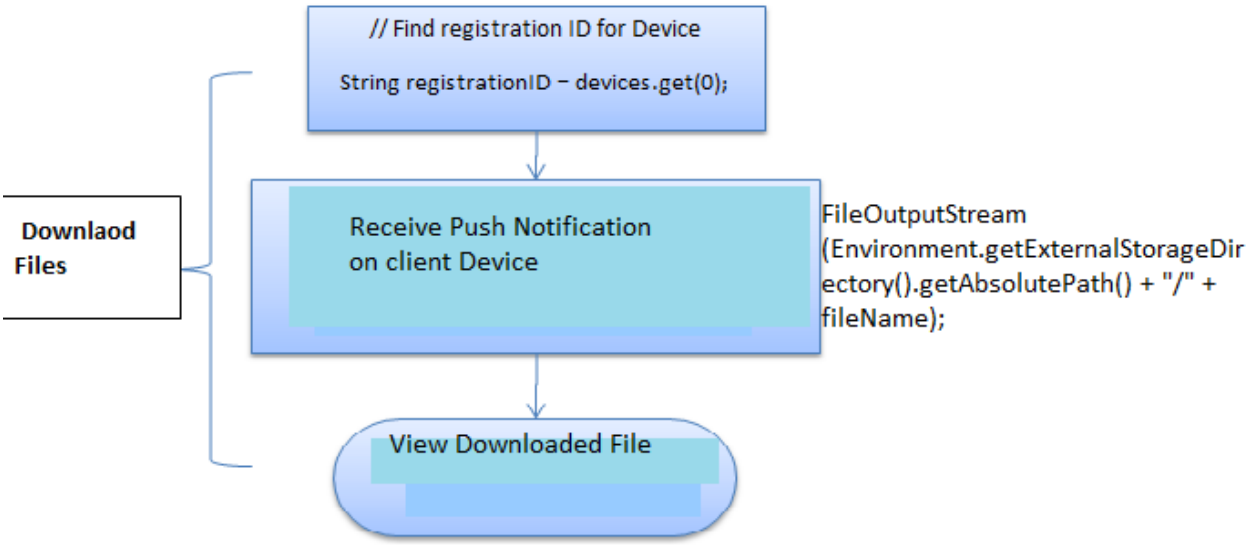


Fig. 6.5

7. data analysis and discussion

7.1 Output generation

- All types of files up to size 8MBcan be uploaded , downloaded & listed out using this application from all synchronized registered android devices.

7.2 Output analysis

- Our application we could deploy on different android devices.
- Application accessed same ID for synchronization purpose from different android devices.
- Application allows to upload all type of file systems & notifies by push notification on single or multiple devices at the same time .
- Application allows to Delete uploaded files & notifies by push notification on single or multiple devices at the same time .
- Application allows to see list of all type of uploaded file systems .
- Application allows downloading all type of uploaded files from all registered clients.
- Application allows up to 8 MB size of file.

7.3 Abnormal case explanation

- We could not handle file size more than 8 MB.
- We could not access directories or folders.
- Our application can perform one task at a time, multiple task is limitation for our application.
- We cannot select multiple files to upload , delete or download.
- All android devices we want to access required to be registered & be able to access server IP address.
- For now GUI is least concern portion for our application. This is demo version where our main concentration is on application functionality.

7.4 Discussion

- Cloud Storage is really wide & needs lot of enhancement in real world. Security , Availability & reliability are main component where all companies are working for cloud these days.
- We are implementing functionality here & trying to enhance our application in later version.

- For installation details , for service details & for basic requirement for this application we mentioned everything in section 5 & 6.

8. Conclusions and recommendations

8.1 Summary and Conclusions

This application helps a lot in small business centers where buying costlier software is big matter. This application allows to perform all transactional related tasks on files & sametime notifies all other users accessing same server && registered with it.

8.2 recommendations for future studies

- We will improve GUI part of our application.
- We will enhance application with more derived functionality like type of file list , update files without downloading.
- We will enhance application which allow more than 8MB size of data file.
- We will try to make this application to work on all devices – not mandatory android device.
- We will provide security & authentication area in later versions.

9. bibliography

1)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=1404cf11cde9a844&attid=0.1&disp=inline&realattid=f_hjz8jx2k0&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376349990620&sads=sipQIu-sqwR2pVHF8opdjNKN91k
2)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=1404cf11cde9a844&attid=0.2&disp=inline&realattid=f_hjz8jx3h1&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376350003629&sads=9nrJ0WSqxfjEPt0IPsfIRB17elA
3)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=14074aab01dc9c3c&attid=0.1&disp=inline&realattid=f_hka9dwuv0&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376347009980&sads=Carv9gbEItBSSG34_qkLQWAFWbc
4)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=14074a4f05c672fb&attid=0.1&disp=inline&realattid=f_hka95zlz0&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376346756332&sads=QJ9VYQHY8v3bvOIGMtDRA3VKLL4
5)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=14074907e3a2503b&attid=0.2&disp=inline&realattid=f_hka8cg661&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376345196283&sads=wXM-q1m-12kKLL8TdFxyNRyJiBA
6)
https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=25b13401c7&view=att&th=14074907e3a2503b&attid=0.1&disp=inline&realattid=f_hka8

5c4q1&safe=1&zw&saduie=AG9B_P8eOCHEYJaM-fnZcN36ws1M&sadet=1376345192850&sads=TwFIZiJx-k4GhBwj3XxXHPoSYXU

7) http://www.infosys.com/manufacturing/resource-center/Documents/oracle-push-notification.pdf

8) http://developer.android.com/google/gcm/http.html#auth

9) https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CEcQFjAA&url=http%3A%2F%2Fwww.ijert.org%2Fbrowse%2Fvolume-2-2013%2Fmay-2013-edition%3Fdownload%3D3597%253Agcm-service-driven-communication-with-an-android-application-in-cloud-computing%26start%3D230&ei=Jj8YUqH8CuWw2wWl_oCoDA&usg=AFQjCNEDBBl_NHQRt9B_8Yekhl2DKZKhXA&sig2=FYOHGinmJML9gfh2ZexVCA&bvm=bv.51156542,d.b2I&cad=rjt

10) http://ant.apache.org/

11) http://www.youtube.com/watch?v=jdSm0C919PM

12) http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServlet.html

10 Appendices

Figures:

program flowchart

- We have provided flowchart of this system in chapter 6 of this documentation.

program source code with documentation

- Source code is submitted online.

input/output listing

- Any type of files
- File size up to 8MB