



CMPE 277: Smartphone App Development  
**Video Streaming – Project Report**

Submitted to

Prof. Chandrasekar Vuppala

By

Jon Guan  
Arun Malik  
Pravin Agrawal  
Matilda Bernard

01/02/2014

## Table of Contents

<b>1. Project Description .....</b>	<b>4</b>
1.1. Configuration 1: Web Stream and Save .....	4
1.2. Configuration 2: Computer Stream and Save.....	4
1.3. Configuration 3: Phone Stream and Save.....	4
1.4. Configuration 4: Local Phone Save .....	5
<b>2. Requirements .....</b>	<b>6</b>
2.1. System Requirements.....	6
2.2. Business Requirements .....	6
<b>3. Mobile UI Design Principles .....</b>	<b>7</b>
3.1. Storyboards.....	7
3.2. Wireframes .....	8
<b>4. High Level Architecture Design .....</b>	<b>9</b>
4.1. Configuration 1: Web Stream and Save .....	10
4.2. Configuration 2: Computer Stream and Save.....	10
4.3. Configuration 3: Phone Stream and Save.....	10
4.4. Configuration 4: Local Phone Save .....	12
<b>5. Component Level Design.....</b>	<b>13</b>
5.1. Phone .....	13
Inputs .....	13
Outputs .....	13
5.2. Computer .....	13
Inputs .....	13
Outputs .....	13
5.3. AWS Server.....	13
Inputs .....	13
Outputs .....	13
<b>6. Sequence of Workflow .....</b>	<b>14</b>
<b>7. Mobile and Cloud Technologies used.....</b>	<b>15</b>
7.1. Cloud Technologies .....	15
7.2. Mobile Technologies .....	15
7.3. Software and Hardware Requirements.....	16
<b>8. Interfaces: RESTful and Server Side Design.....</b>	<b>17</b>
8.1. RESTful .....	17
8.2. Server Side Design .....	18
<b>9. Client Side Design .....</b>	<b>19</b>
<b>10. Testing .....</b>	<b>20</b>
10.1. UI Testing:.....	20

<b>10.2. Smoke Testing:</b> .....	<b>20</b>
<b>10.3. Integration testing:</b> .....	<b>21</b>
<b>10.4. Operational Test .....</b>	<b>21</b>
<b>11. Automation .....</b>	<b>22</b>
<b>12. Design Patterns used .....</b>	<b>23</b>
<b>13. Profiling.....</b>	<b>24</b>
<b>13.1. Media Server - CPU and Network Usage Profiling:.....</b>	<b>24</b>
<b>13.2. Android Phone Profiling: .....</b>	<b>25</b>
<b>14. References .....</b>	<b>26</b>
<b>15. Screen Captures.....</b>	<b>27</b>

# **1. Project Description**

Our video streaming project was imagined after our desire to view lectures remotely, as well as to review past lectures missed. In the spirit of Apple iTunesU, Khan Academy, Harvard, MIT, and Berkeley Webcast, we wished to provide a database of lectures given by professors at SJSU. However, because of lack of funds within the California State University system causing a the dearth of infrastructure support for lecture recording and video streaming, our project aims to provide a convenient way to record and to broadcast lectures to all students attending SJSU.

Our video streaming system allows for several configurations, depending on the network and needs of the user. The configurations are as follows:

## **1.1. Configuration 1: Web Stream and Save**

This configuration enables users to record professors' lectures through the built in camera and microphone via the Android app and stream it in real time to an Amazon Web Service instance enabled with Adobe Flash Media Server. From there, the video stream is encoded and also saved for future playback. Users who wish to view the video stream simply accesses the video link on the AWS web server to stream the video file to their player of choice.

## **1.2. Configuration 2: Computer Stream and Save**

This configuration also allows users to record professors' lectures through their Android phone, but the encoder, server, and file storage resides on a personal computer sharing the same network. Video streaming is limited to those users who are able to access the ip address of the computer.

## **1.3. Configuration 3: Phone Stream and Save**

In this configuration, the camera, video encoder, and web server all resides on the phone itself. It must be emphasized that this configuration is very CPU intensive and thus, also power intensive. The Android phone should be plugged into a power source while recording and serving video.

#### **1.4. Configuration 4: Local Phone Save**

This configuration is used in the doomsday scenario where the user finds himself with only a phone and nothing else -- not even internet connectivity. This configuration saves lectures to local storage, and when internet connectivity resumes, the user is given the option to publish the video to the cloud for general consumption.

The various recording and streaming configurations provides to our video streaming system a very robust interface to provide webcast lectures for all CSU students.

Our app is available on the google play store :

<https://play.google.com/store/apps/details?id=com.cmpe277.mediastreaming>

## **2. Requirements**

### **2.1. System Requirements**

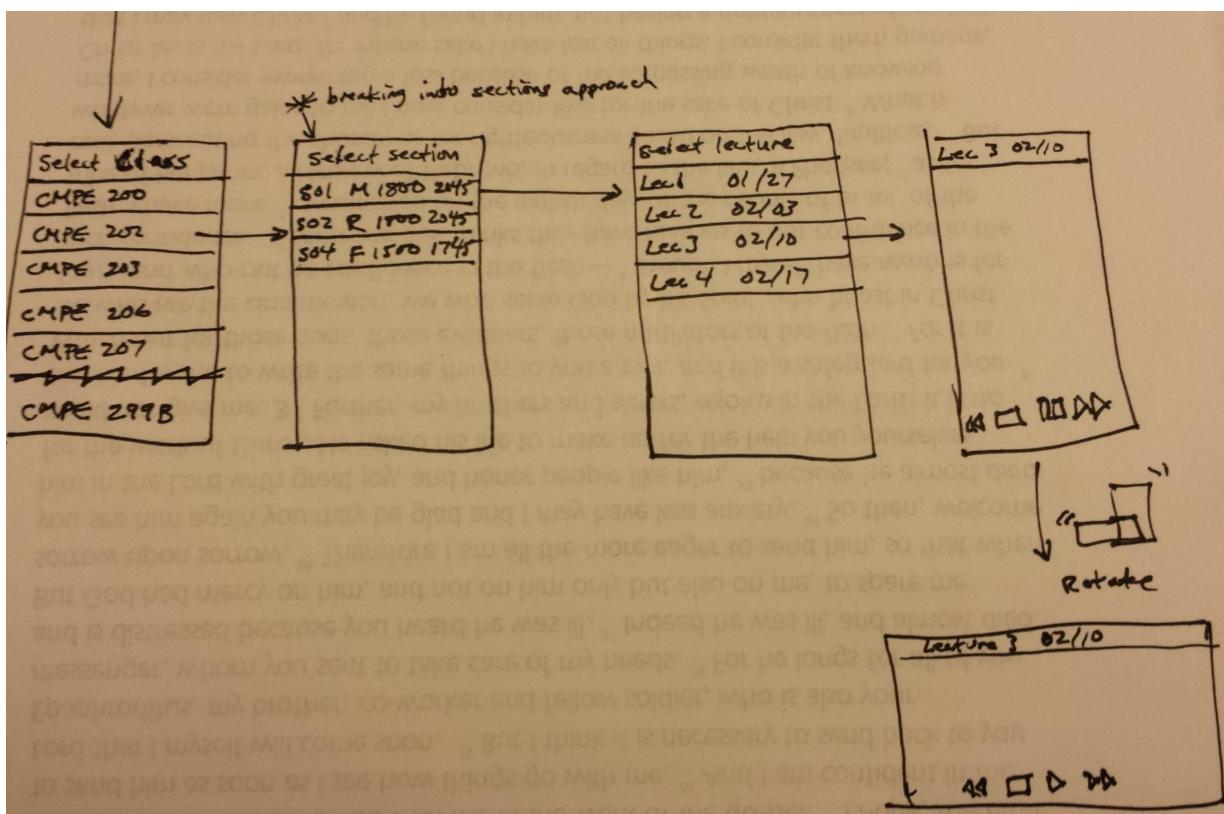
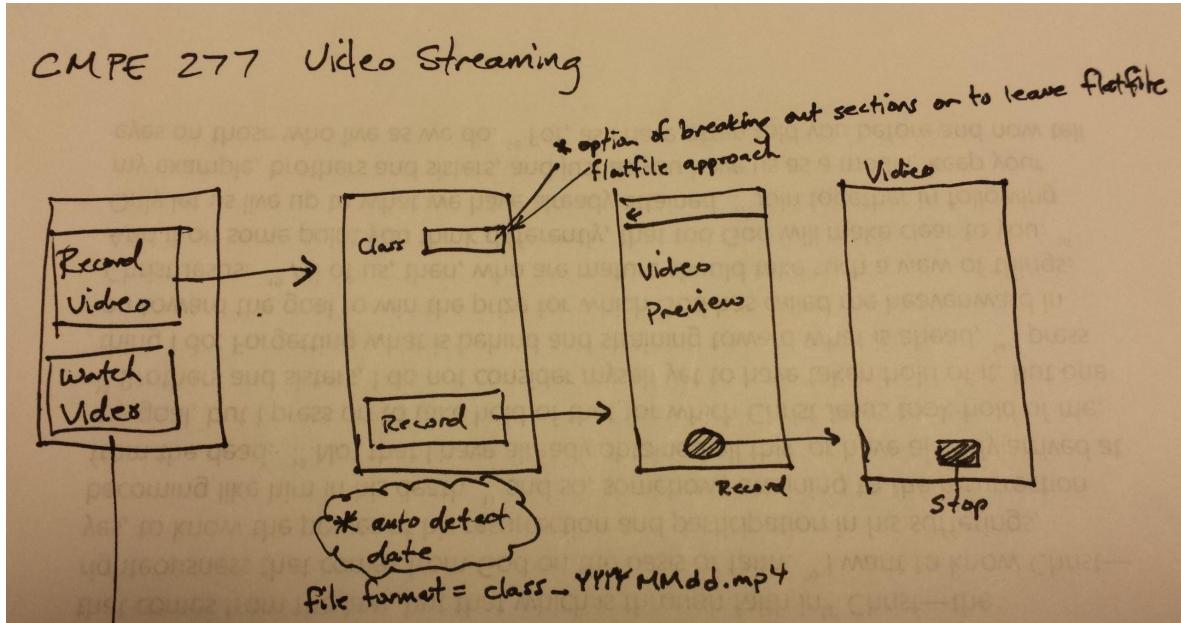
- Android phone, minimum os version 2.3 Gingerbread (API level 9)
  - Must have built-in camera
  - SDCard storage bay recommended, but not required.
- Amazon Web Service (AWS) instance (Configuration 1 and 4)
  - Adobe Flash Media Server (Configuration 1)
- VLC player on a Windows PC or Mac (Configuration 2)
  - Web server
- Internet connectivity during time of live stream to enable streaming capability.
- Internet connectivity for file upload if not present during time of live stream.
- SD card for saving video in case Internet is not available

### **2.2. Business Requirements**

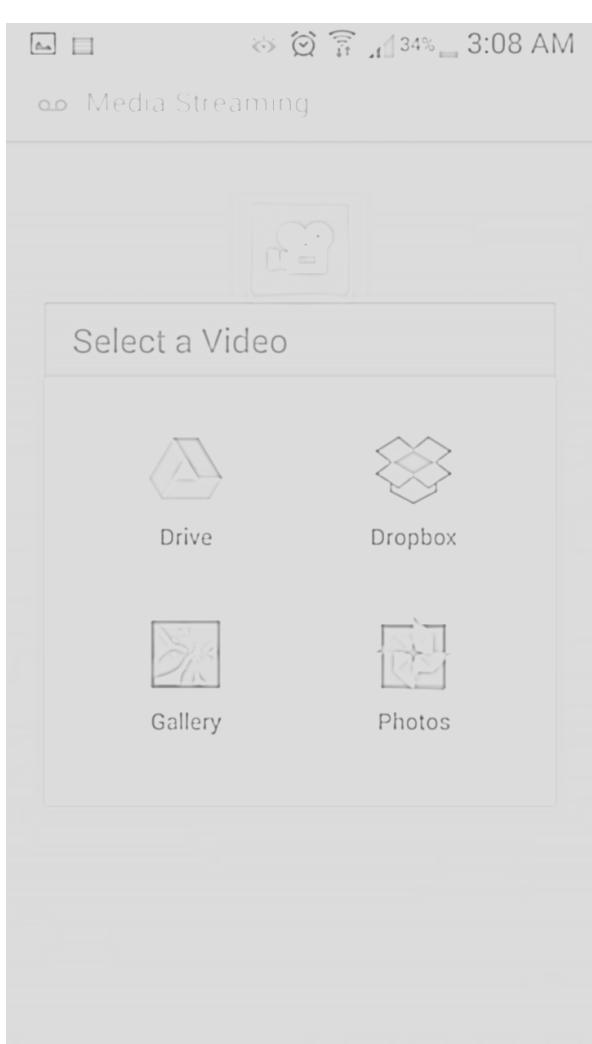
- App should provide single click streaming option
- The app should send the video to a cloud server where it is saved and can be accessed by users later on.
- There needs to be a provision for sending the video from the phone to the cloud server later on in case an internet connection is not available
- The UI of the app should be user friendly and intuitive
- App should provide the ability to view / browse the videos that have been stored on the cloud server

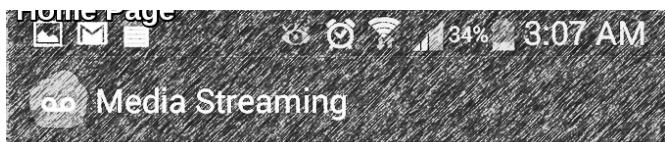
### 3. Mobile UI Design Principles

#### 3.1. Storyboards



### 3.2. Wireframes





Live Stream



Upload Video

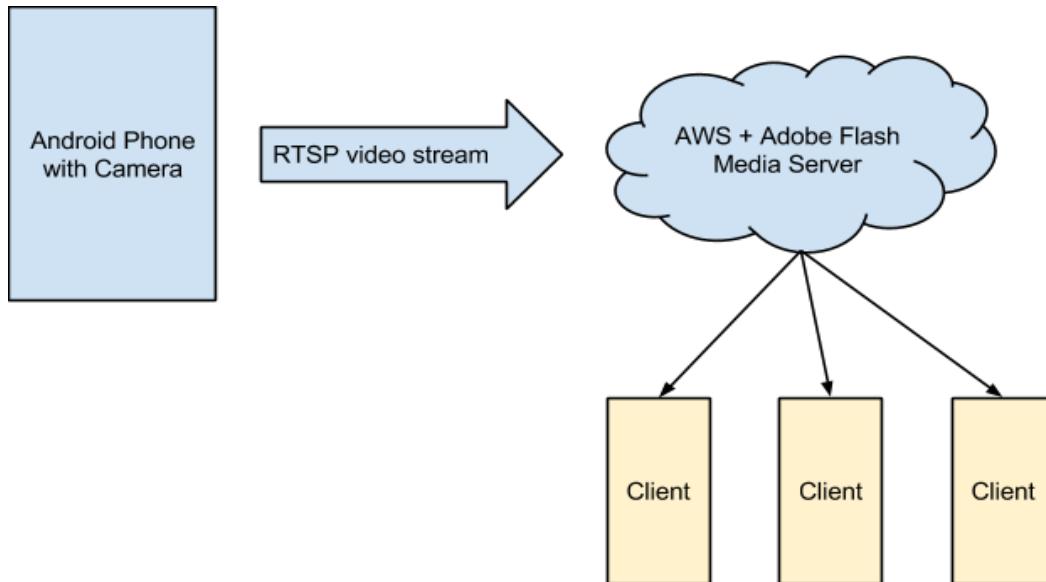


Play Video

## 4. High Level Architecture Design

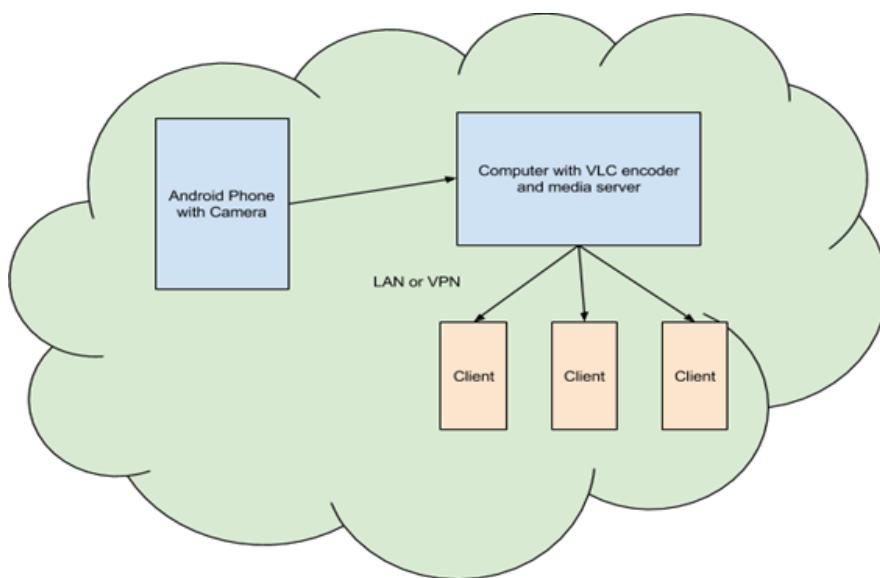
### 4.1. Configuration 1: Web Stream and Save

The Web Stream and Save configuration consists of the Android smartphone communicating via RTSP protocol to the Adobe Flash Media Server hosted on AWS.



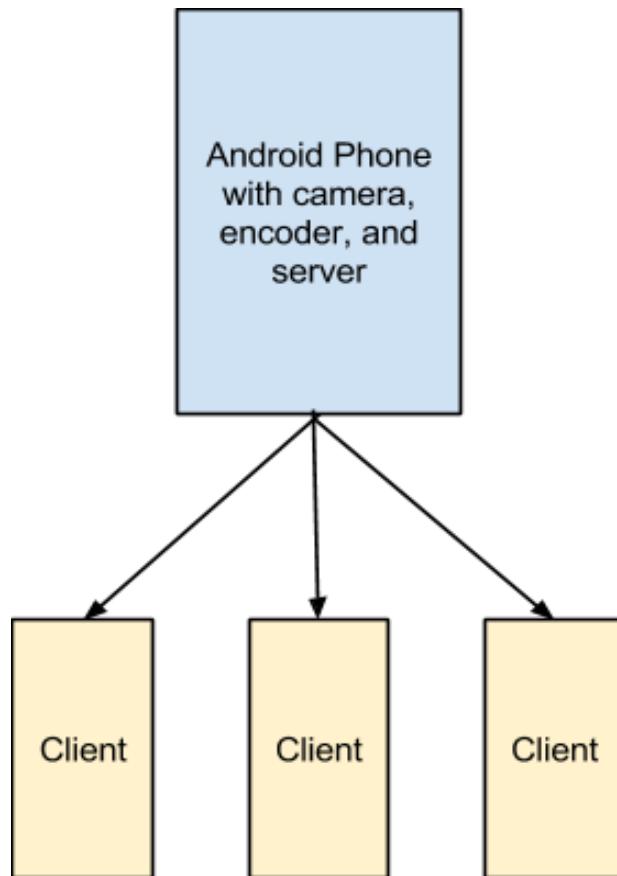
### 4.2. Configuration 2: Computer Stream and Save

The computer stream and save configuration consists of the Android smartphone streaming data to a connected computer. The computer then acts as an encoder and video server to stream video to all connected clients on the same network.



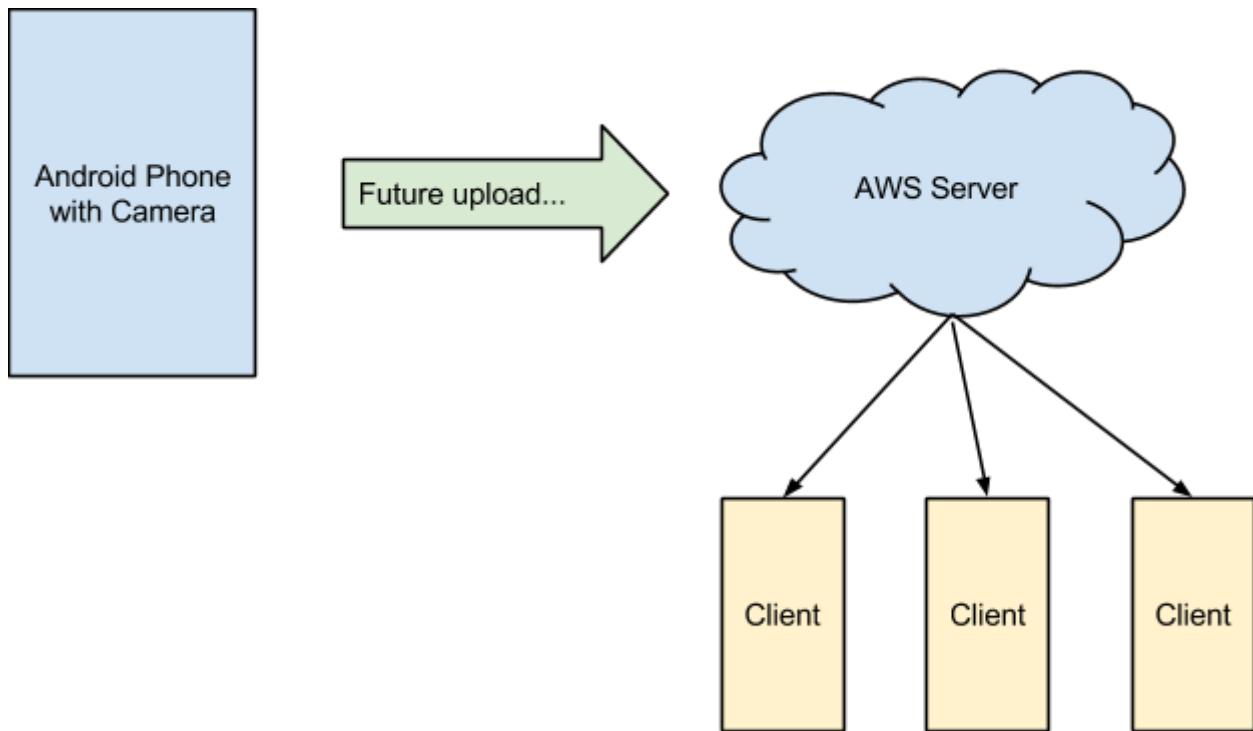
### **4.3. Configuration 3: Phone Stream and Save**

The phone stream and save configuration is identical to the computer stream and save configuration, except that the encoding and video serving is also done by the Android smartphone. Clients who are able to access the phone via its IP address will be able to consume video streams.



#### 4.4. Configuration 4: Local Phone Save

The local phone save configuration consists of saving the video to local storage and giving users the option of uploading the video to the AWS server when internet is available.



## 5. Component Level Design

Components of the system include the phone, the Windows PC or Mac computer, and the AWS server.

### 5.1. Phone

The Android phone enables the user to record video and to save it, stream it, and / or broadcast it. The app will arbitrate the system resources available and decide which configuration will best optimize the serving of the video stream.

#### Inputs

- Record
  - Class name, section number
  - Date / time
  - System resources (Internet connectivity, computer available, etc)
- Watch
  - Class name, section number, lecture number

#### Outputs

- Streaming video

### 5.2. Computer

The computer has the option to serve as a video encoder and a video stream server.

#### Inputs

- Phone connection
- Video stream

#### Outputs

- Video streams

### 5.3. AWS Server

The AWS server enables users from around the globe to view lectures and stream videos at their convenience.

#### Inputs

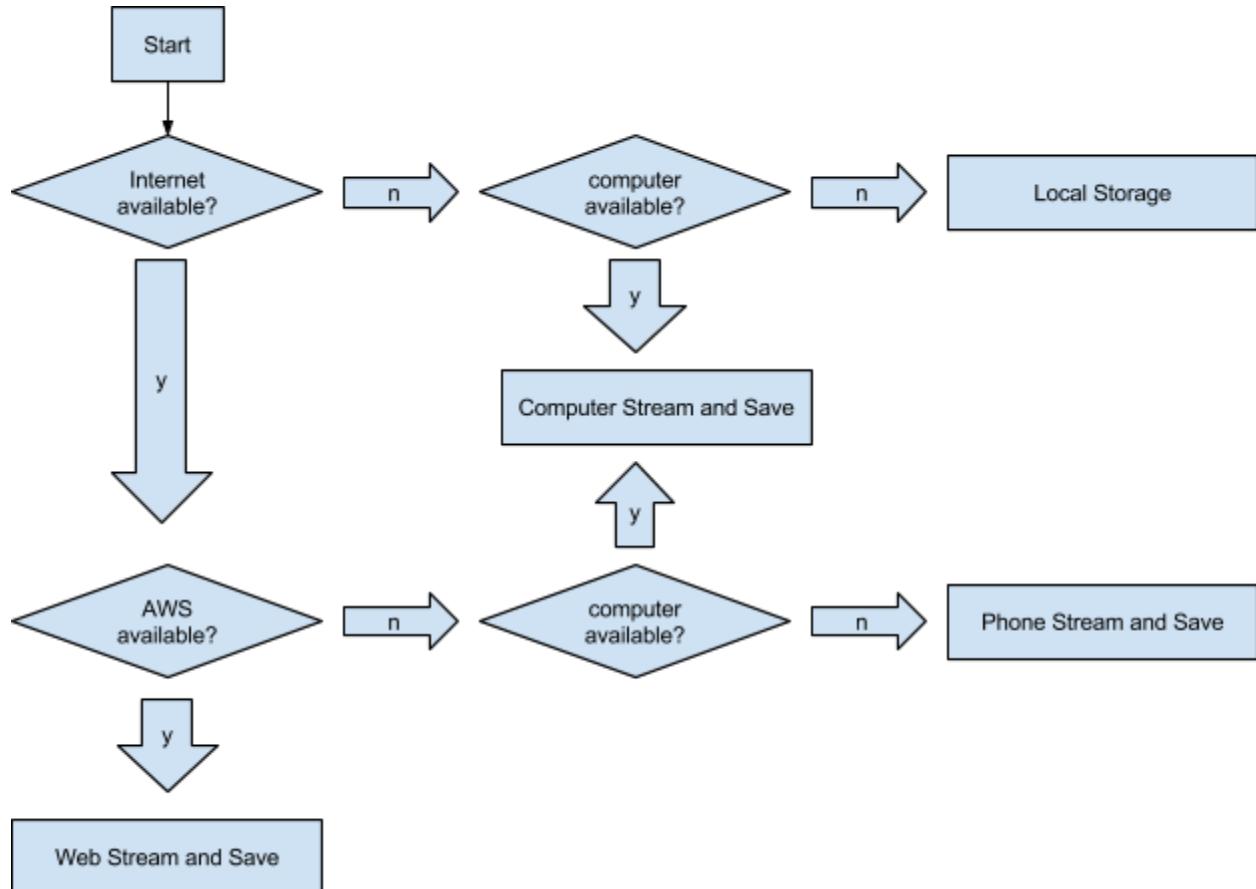
- Video stream from phone

#### Outputs

- Video streams

## 6. Sequence of Workflow

Detailed steps to setup the video streaming environment are written in the README document.  
A brief overview of the streaming workflow is as follows:



The watch streaming video workflow is relatively straight forward. Videos can be consumed through the phone app or through any other Internet enabled device. Simply navigate to the correct video location hosted on the AWS web server.

## 7. Mobile and Cloud Technologies used

### 7.1. Cloud Technologies

The backend of our project is hosted on an Amazon EC2 instance. This is a IaaS offered by Amazon along with an operating system of choice. We are also using Flash Media Server on it and using VLC live media streaming for our app.

- **Amazon EC2:** Amazon Elastic Cloud Compute is an IaaS offering which comes with an operating system of choice. It provides compute capacities with the flexibility of changing their sizes as per requirement. These instances can be managed programmatically using their SDK or directly from their management console. The advantage of using a server on the cloud is that in case there is a need to scale the project due to maybe high traffic, it can be done in a few minutes just by the simple click of a button. Another advantage if using EC2 is that it has a pretty good uptime, the server does not need to be managed and alarms can be used for maximum profitability.
- **Flash Media Server:** We are using flash media server on our Amazon EC2 instance which basically helps with media streaming and more importantly with saving the videos on our server so that users can view them later too. It supports HTTP streaming that is not only protected but also has dynamic packaging.
- **VLC Live Media Streaming:** VLC is a free and open source media player that also supports live streaming. It is a cross-platform framework that can play most codecs like FLV, MP4, MP3, WMV, etc. Apart from media streaming is also supports media conversion.

### 7.2. Mobile Technologies

- **Toast:** Toast messages have been used in our app to inform user when the Service is started, Service has disconnected or in case an error has been encountered.
- **Services:** The app uses IBinder to use bound service for streaming of video. The service does not run in the background endlessly and acts as the server in a client-server app.
- **Async Task:** In case there is no internet connection, the video is then stored on the SD card. The moment a connection is detected, a background task is spawned and the video is uploaded the cloud server.
- **Surface Holder:** This is used to be able to view the content that is being streamed on the phone too. The Surface View lets the camera object show the preview on it.

### **7.3. Software and Hardware Requirements**

- Operating System: Android
- Required RAM: 512 MB
- Minimum Android Version: Android 2.2
- Internet connection: Preferable
- Space used by App: 2 MB
- SD Card: Required

## 8. Interfaces: RESTful and Server Side Design

### 8.1. RESTful

Our app has been implemented using REST software architecture principles.

For uploading:

```
@POST
@Path("/upload")
@Consumes(MediaType.MULTIPART_FORM_DATA)
public Response uploadFile(
        @FormDataParam("file") InputStream uploadedInputStream,
        @FormDataParam("file") FormDataContentDisposition fileDetail) {

    String uploadedFileLocation = "/mnt/applications/vod/media/"
            + fileDetail.getFileName();

    // save it
    writeToFile(uploadedInputStream, uploadedFileLocation);

    String output = "File uploaded to : " + uploadedFileLocation;

    return Response.status(200).entity(output).build();
}
```

For saving:

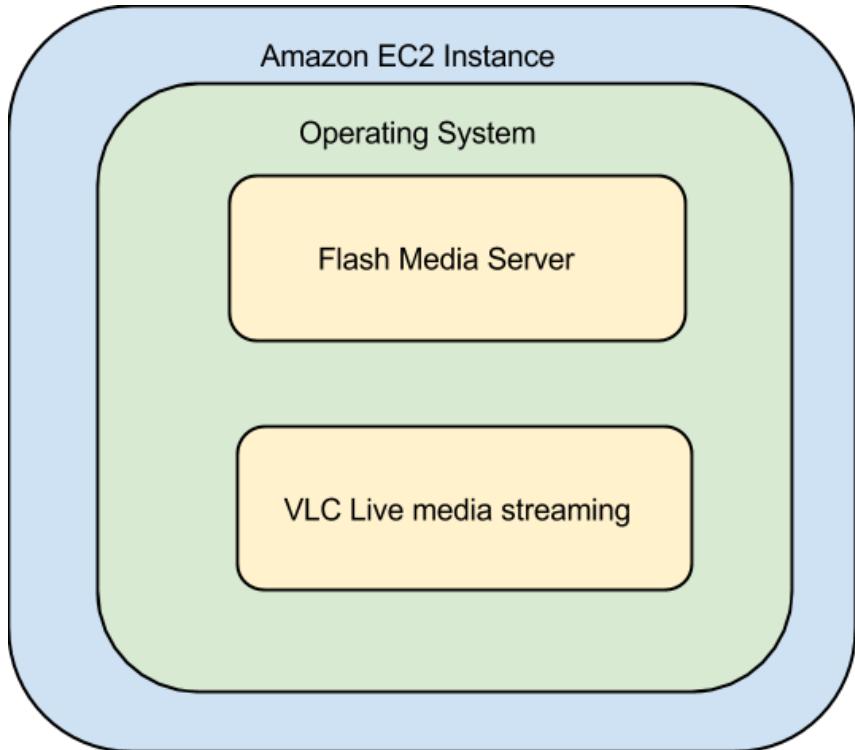
```
// save uploaded file to new location
private void writeToFile(InputStream uploadedInputStream,
                        String uploadedFileLocation) {

    try {
        OutputStream out = new FileOutputStream(new File(
                uploadedFileLocation));
        int read = 0;
        byte[] bytes = new byte[1024];

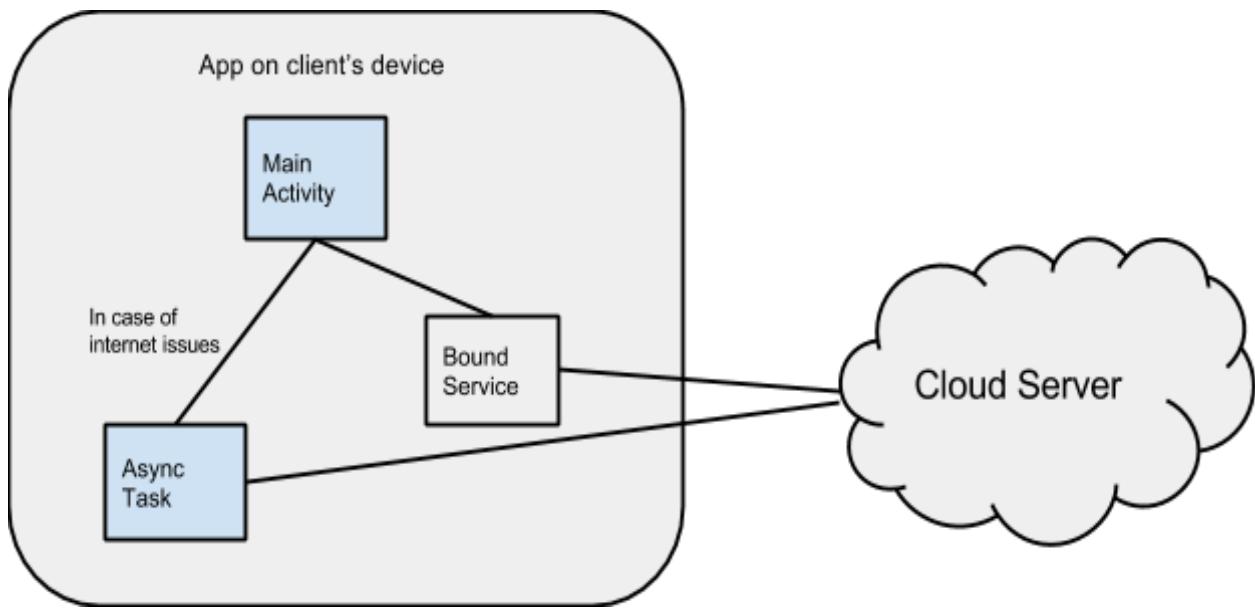
        out = new FileOutputStream(new File(uploadedFileLocation));
        while ((read = uploadedInputStream.read(bytes)) != -1) {
            out.write(bytes, 0, read);
        }
        out.flush();
        out.close();
    } catch (IOException e) {

        e.printStackTrace();
    }
}
```

## 8.2. Server Side Design



## 9. Client Side Design



## 10. Testing

The complete testing strategy for this android included:

1. UI Testing
2. Smoke Testing
3. Integration Testing
4. Operational Testing

### 10.1. UI Testing:

User interface testing is important to test the behavior and interaction of application with the user when it is running on a device. UI testing asserts the correctness of UI output in response to a user action on a device, such as touch, click, press, shake, menu bar selection and other UI controls.

UI testing is commonly done by run tests manually and verify that the app is behaving as expected. But there are automated testing involving creating programs that runs test cases to cover specific user scenarios, and then using the testing framework to run the test cases automatically and in a repeatable manner. For scope of this project all the UI testing was done manually.

Following Scenarios were tested under UI Testing

1. Navigation from one activity to another and back.
2. Touch and click behavior
3. Phone orientation change behavior
4. Multiple selects, long touch, hard button press

### 10.2. Smoke Testing:

**Smoke testing** is first line of testing that helps reveal simple failures severe enough to reject a prospective software release. A subset of test cases that cover the most important functionality of a component or system is selected and run, to ascertain if the most crucial functions of a program work correctly. For example, a smoke test may ask basic questions like "Does the app run?", "Does navigation works?", or "Does clicking the buttons do anything?". The purpose is to determine whether the application is so badly broken that further testing is unnecessary. Below are the list of smoke test scenario's for this app.

1. Check if app installs and loads main activity properly.
2. Check if all buttons work on the UI screen
3. Live stream works as expected.
4. Files are uploaded to media Server
5. User is able to browse and playback uploaded video files

### **10.3. Integration testing:**

Integration testing is conducted to test the integration between individual software modules when they are combined and tested as a group. This app mainly has three modules which requires extensive integration testing to verify the smooth working between the three modules. Below are the scenarios tested under integration testing

1. The media server API to upload recorded video works as expected with the recording module
2. Live streaming video from the app are viewable on media players.
3. Uploaded files to the server are available on the web server and can be play backed.

### **10.4. Operational Test**

Operational test validates the completeness and correctness of the android application. They are also called as functional testing or acceptance testing. They includes code analysis, installation testing, load testing and performance testing

## 11. Automation

There are a number of automated testing frameworks or tools for Android applications, including but not limited to Activity Instrumentation, MonkeyRunner, Robotium, or Robolectric.

The Android SDK comes with a testing tool called MonkeyRunner, providing an API and an execution environment for running tests written in Python. The tool has APIs for connecting to a device, installing/uninstalling apps, running apps, taking screenshots, comparing images to see if the screen contains what is supposed to contain after certain commands have been performed, and to run a test package against an application

We can also use the uiautomatorviewer tool to take a snapshot of the foreground UI screen on any Android device that is connected to your development machine. The uiautomatorviewer tool provides a convenient visual interface to inspect the layout hierarchy and view the properties of the individual UI components that are displayed on the test device. Using this information, you can later create uiautomator tests with selector objects that target specific UI components to test.

For scope of this application, Automated testing is part of the overall testing strategy.

## 12. Design Patterns used

In software engineering, a **design pattern** is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations.

Some of the patterns used in Android app:

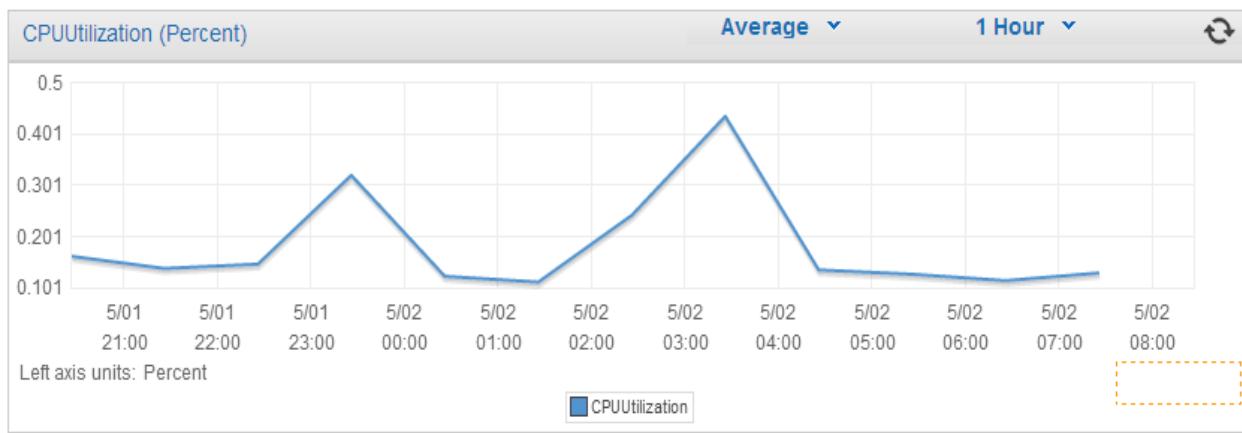
1. State Pattern
2. Command Pattern
3. Singleton pattern
4. Chain of responsibility pattern
5. MVC

## 13. Profiling

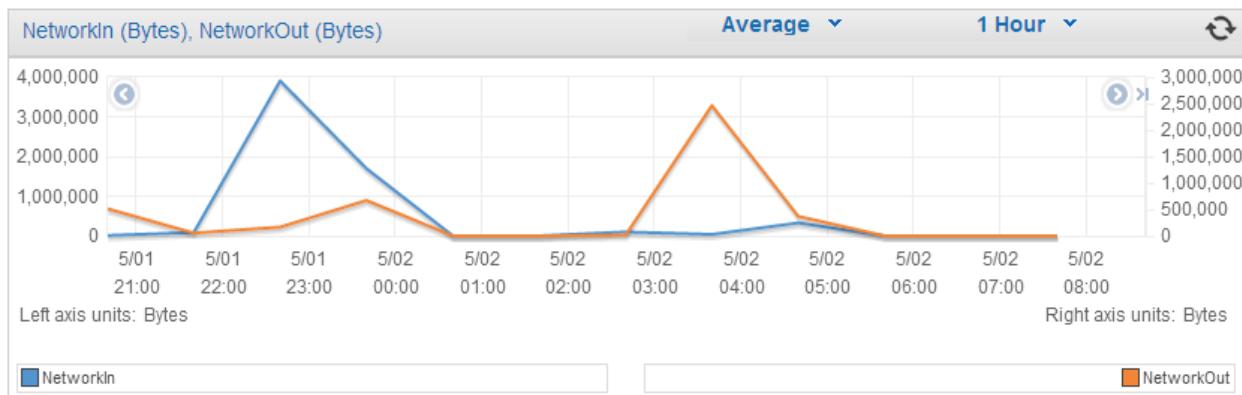
### 13.1. Media Server - CPU and Network Usage Profiling:

The CPU and memory usage of the Adobe media server running on AWS EC2 were monitored and profiled with single user load to observe the impact and processing power required to stream one video. The performance numbers increase linearly as the number of users connected to view the video stream of previously saved files. Below are the graphs obtained from the profiling:

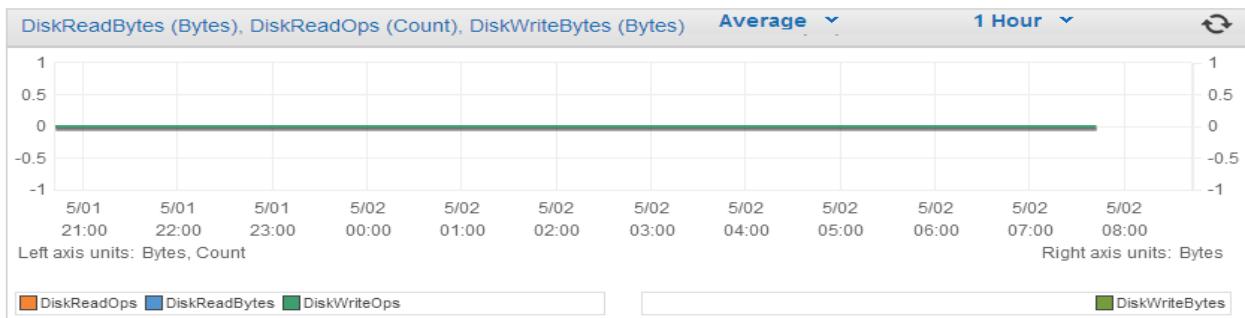
CPU Utilization:



Network Utilization:



Disk I/O:



### 13.2. Android Phone Profiling:

The profiling was carried out on Samsung Galaxy S3 and the app used camera, screen, and network to send the live video stream. Using these hardware components consumes battery life, cellular data and phones CPU

#### Network Usage:

- A video file of size 250 MB was uploaded to the server in 4:30 minutes at the rate of 0.92 MB/s on network with a bandwidth of 50 MBPS and upload speeds up-to 12 MBPS.
- Live stream video were played with none to few buffering freeze time and the overall stream quality observed was good.

#### Battery and CPU Usage

It proved difficult to accurately measured the statistics of android phone without rooting the device. Battery Monitor and System Tuner were used obtained the statistics.

The experiment was carried against 1 hour recoding and streaming of video using the app. About 45% overall battery usage caused was due to screen that displayed the video during the recording. About 26% of the battery consumption occurred to process and send the video. The CPU utilization was 60% average during the recording and processing.

#### Delay

There was delay observed between recording and playback on the VLC Player which was based on the quality of the video stream. A delay of around 2-4 seconds was observed between start of stream to the playback on VLC player

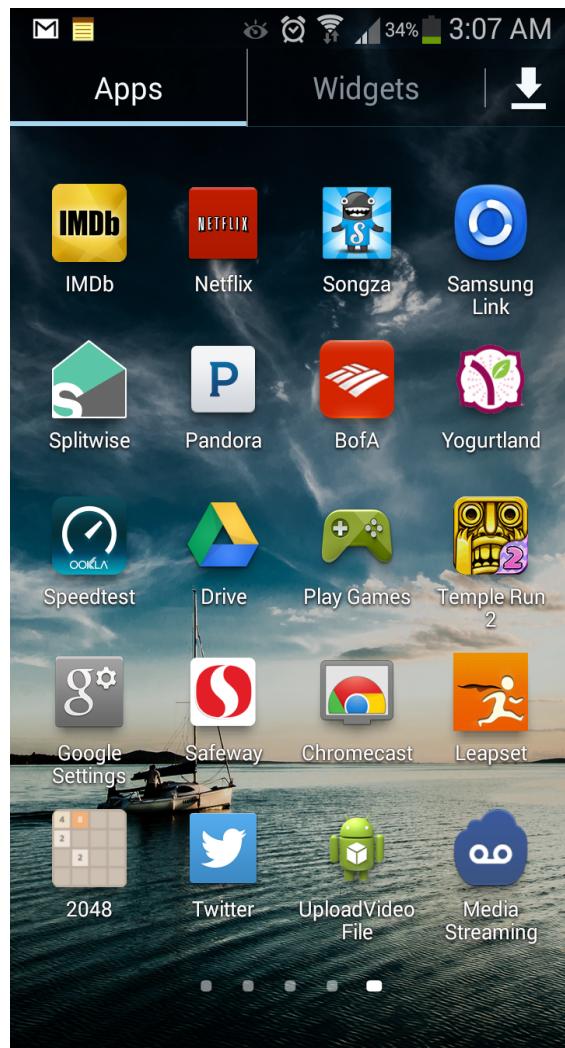
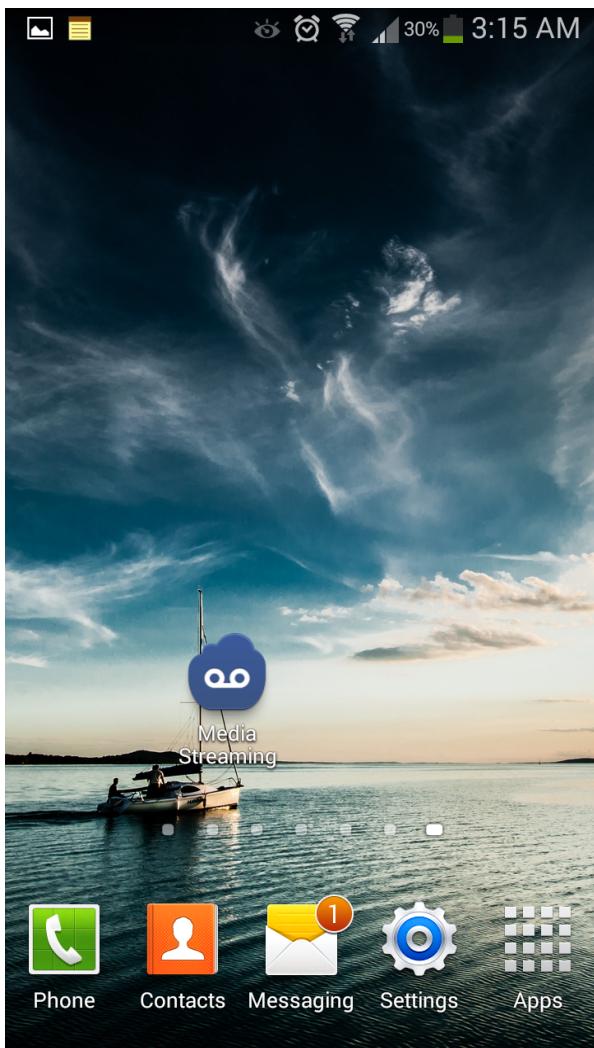
Delay in viewing the previously uploaded video from media server directly varied based on the size and format and quality of the video.

## 14. References

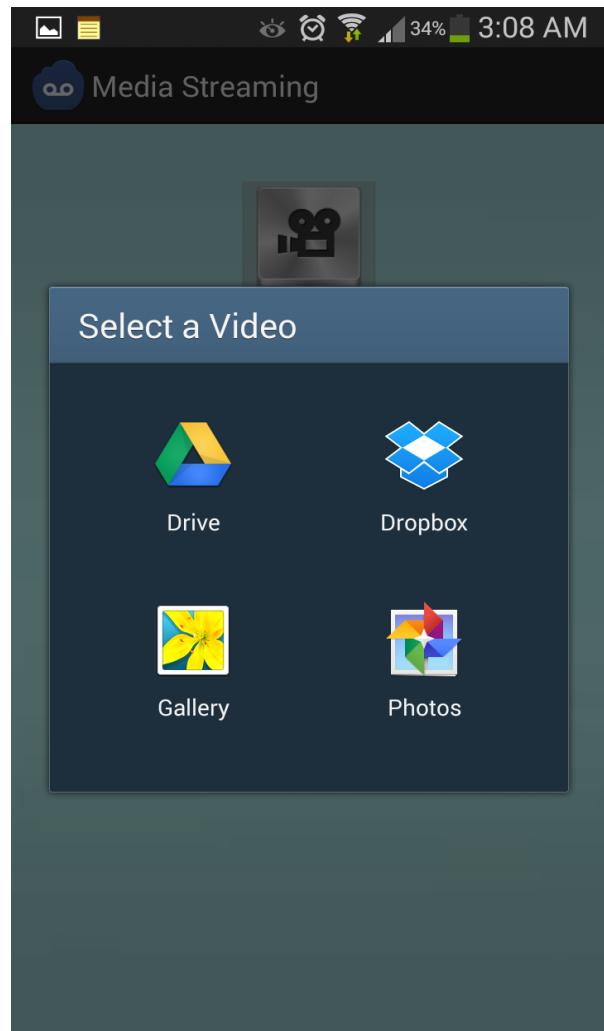
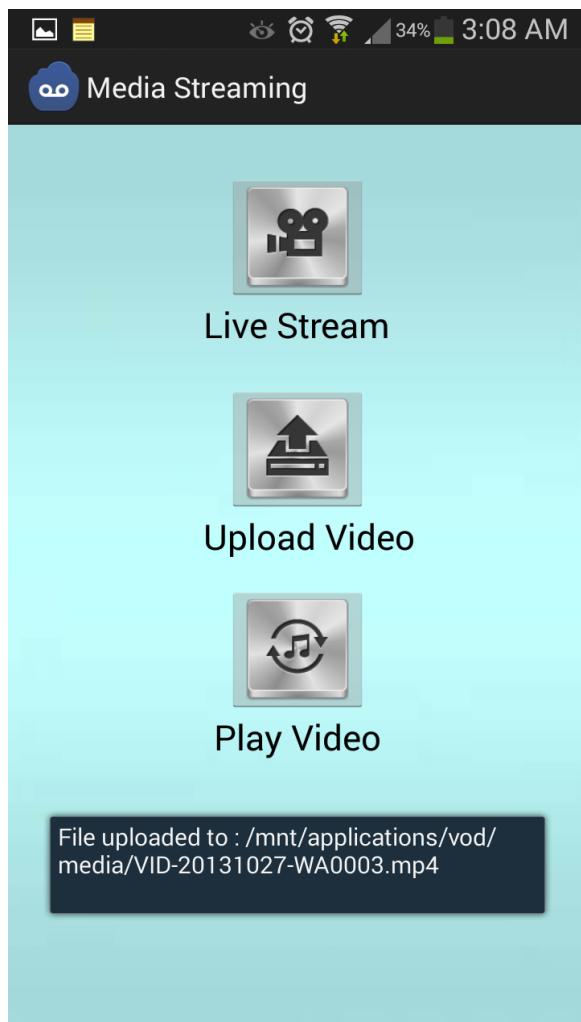
1. [http://developer.android.com/tools/testing/testing\\_ui.html](http://developer.android.com/tools/testing/testing_ui.html)
2. <http://www.packtpub.com/sites/default/files/3500-chapter-1-getting-started-with-testing.pdf>
3. <http://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=4190&context=etd>
4. [http://en.wikipedia.org/wiki/Integration\\_testing](http://en.wikipedia.org/wiki/Integration_testing)
5. [http://en.wikipedia.org/wiki/Smoke\\_testing\\_\(software\)](http://en.wikipedia.org/wiki/Smoke_testing_(software))
6. <https://developer.android.com/design/patterns/gestures.html>
7. [http://en.wikipedia.org/wiki/Software\\_design\\_pattern](http://en.wikipedia.org/wiki/Software_design_pattern)
8. [http://developer.android.com/tools/testing/testing\\_ui.html](http://developer.android.com/tools/testing/testing_ui.html)
9. <http://www.packtpub.com/sites/default/files/3500-chapter-1-getting-started-with-testing.pdf>
10. <https://code.google.com/p/spydroid-ipcamera/>
11. <http://themeforest.net/item/goodnex-premium-responsive-wordpress-theme/4948491>

The last two references are links to the server and encoder open source code that we referred for the video streaming in our app.

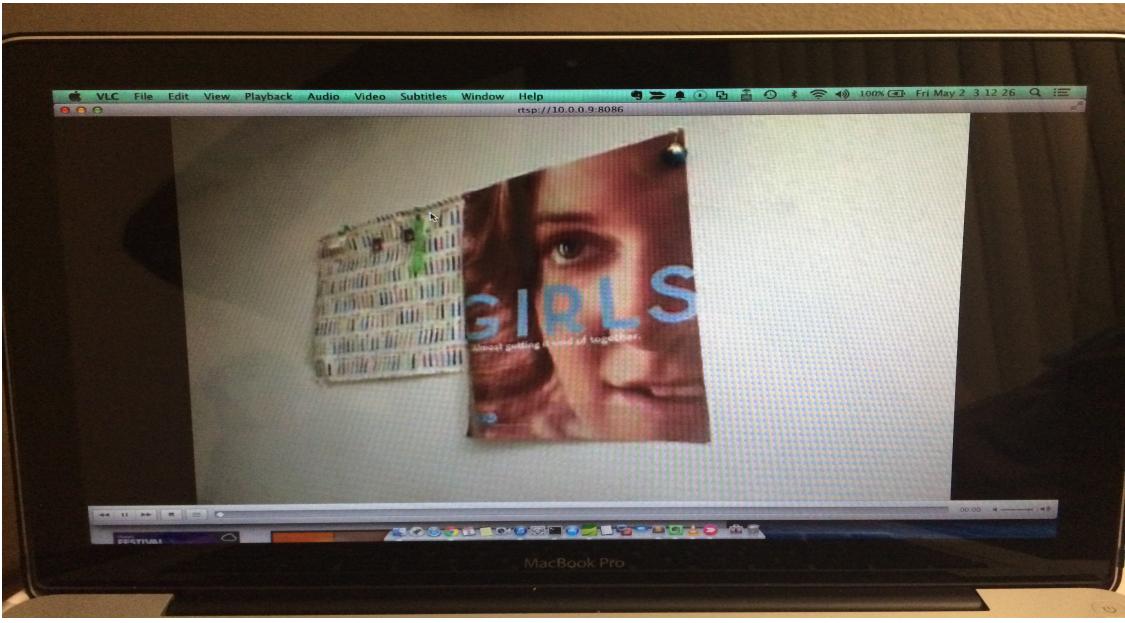
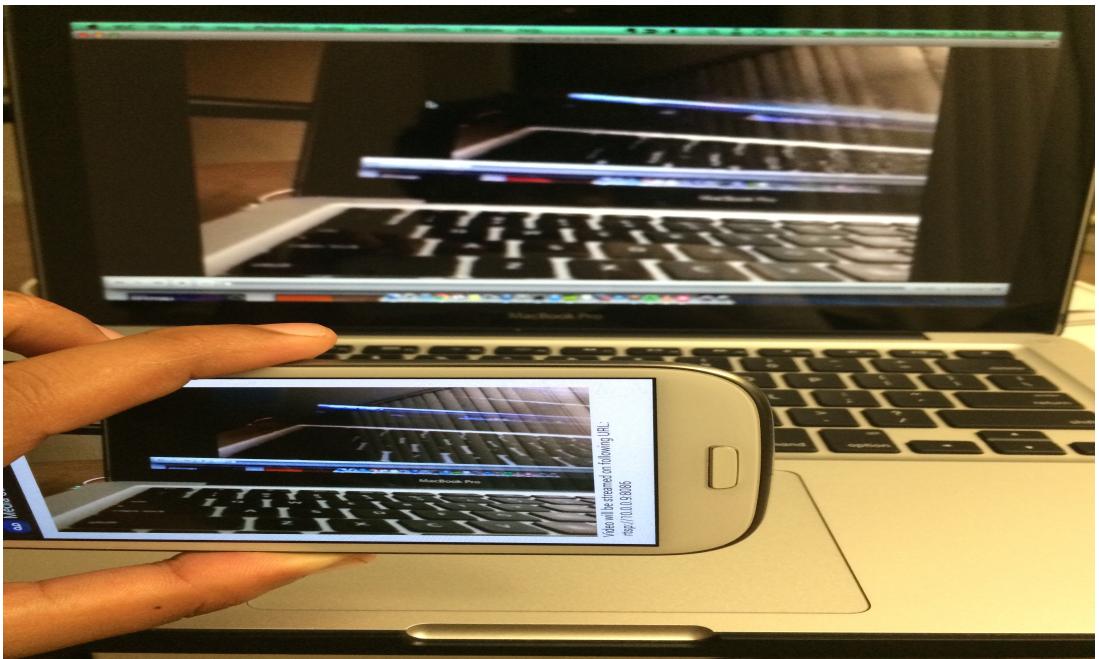
## 15. Screen Captures



App Icon



Screens of the App



Live Streaming on the cloud server

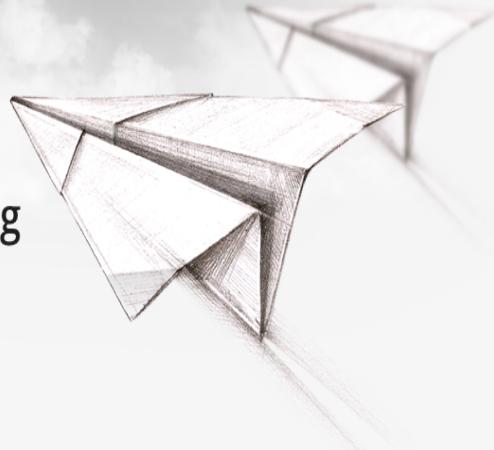
Website for browsing saved videos

The screenshot shows the homepage of the Media Streaming website. At the top, there is a navigation bar with links for HOME, ABOUT US, BROWSE VIDEOS, and CONTACT. To the right of the navigation bar are social media sharing icons for Twitter, Facebook, Google+, YouTube, and RSS feed. Below the navigation bar, the main content area features a large image of two tablets. The tablet on the left displays a grid of video thumbnails, while the tablet on the right shows a specific video player interface with a play button and other controls. To the left of the tablets, the text "No boundaries to class Media Stream" is displayed. Below this text are three green rectangular buttons with white text: "Learn Stuff, Watch Videos", "Share creativity online.", and "Even when you are travelling".

Media Streaming is an Android Application enables user to stream video online / offline and broadcast it to world.

The screenshot shows the homepage of the Media Streaming website. It features a similar layout to the first one, with a navigation bar at the top and social media sharing icons. The main content area includes a large image of a laptop. The laptop screen displays the Media Streaming logo and the text "ANDROID APP". To the right of the laptop, there are two green rectangular buttons: "Enjoy Your Classes on your MAC" and "Windows Or Phone". Below these buttons, the text "Live Stream on your PC" is prominently displayed.

Media Streaming is an Android Application enables user to stream video online / offline and broadcast it to world.



## Online Streaming

Faster Better Cheaper...

No it's free !!!

Media Streaming is an Android Application enables user to stream video online / offline and broadcast it to world.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'file:///Users/malik/Documents/Projects/cmpe277/cmpe277Project-OnlineMediaStreaming/website/contact.html'. The page content includes the 'Media Streaming' logo at the top, followed by a navigation bar with 'HOME', 'ABOUT US', 'BROWSE VIDEOS', 'CONTACT' (which is highlighted in green), and a search bar. Below the navigation is a section titled 'Contact' with a map of San Jose, California, showing various streets and landmarks. A red location pin is placed near the 'EAST CAMPUS' area. At the top right of the page, there is a phone icon followed by the text 'Call Us : +00 123456789' and a set of social media sharing icons.

Safari - Promotions - malik.mgm Media Streaming| Home Media Streaming| Home Goodnex Theme | Home

file:///Users/malik/Documents/Projects/cmpe277/cmpe277Project-OnlineMediaStreaming/website/browse.html

Apps SJU Tools Misc Study

Call Us : +00 123456789

Media Streaming

HOME ABOUT US BROWSE VIDEOS CONTACT

Directory Listing For /

Filename	Size	Last Modified
<a href="#">20140501_170750.mp4</a>	17303.8 kb	Fri, 02 May 2014 00:09:31 GMT
<a href="#">20140501_221350.mp4</a>	17891.6 kb	Fri, 02 May 2014 05:15:15 GMT
<a href="#">VID-20131005-WA0000.mp4</a>	2634.4 kb	Fri, 02 May 2014 09:39:30 GMT
<a href="#">VID-20131027-WA0003.mp4</a>	483.9 kb	Fri, 02 May 2014 10:09:01 GMT
<a href="#">VID-20131223-WA0071.mp4</a>	1770.5 kb	Fri, 02 May 2014 10:10:16 GMT
<a href="#">VID-20140301-WA0003.mp4</a>	9591.6 kb	Fri, 02 May 2014 00:05:00 GMT
<a href="#">Wildlife.wmv</a>	22627.7 kb	Fri, 02 May 2014 00:04:13 GMT
<a href="#">sample.flv</a>	2015.4 kb	Fri, 21 Feb 2014 11:12:55 GMT
<a href="#">sample1_1000kbps.f4v</a>	13411.6 kb	Fri, 21 Feb 2014 11:12:55 GMT
<a href="#">sample1_1500kbps.f4v</a>	19237.3 kb	Fri, 21 Feb 2014 11:12:55 GMT
<a href="#">sample1_150kbps.f4v</a>	2333.7 kb	Fri, 21 Feb 2014 11:12:55 GMT
<a href="#">sample1_500kbps.f4v</a>	6944.2 kb	Fri, 21 Feb 2014 11:12:55 GMT
<a href="#">sample1_700kbps.f4v</a>	9600.7 kb	Fri, 21 Feb 2014 11:12:55 GMT

## 16. Contributions

- Live stream: Pravin, Jon
- Upload video: Arun, Matilda
- Browse video: Jon
- REST API to upload video: Arun
- HTML Website: Matilda
- Configuring Adobe Media Server: Pravin
- Manual testing and profiling: Jon, Matilda
- Design document and report : Arun, Jon, Matilda, Pravin