**Live Video Streaming (Android)**

**Under the guidance of**

**Shrinivas Dudhani.**

**Index**

# Table of Contents

1. Problem definition

2. Project Scope

3. Architecture

4. Coding standards

5. Software Development Life Cycle (SDLC)

6. Waterfall Model

7. Feasibility Study

* 1. Technical Feasibility
  2. Economical Feasibility

7.3 Operational Feasibility

8. Data Flow Diagram

8.1 Level 0

8.2 Level 1.1

8.3 Level 1.2

8.4 Level 1.3

8.5 Level 1.4

8.6 Level 2.1

8.7 Level 2.

9. UML Diagrams

9.1 Use Case diagram

9.2 Class diagram

9.3 Sequence diagram

9.4 State diagram

9.5 Activity diagram

9.6 Deployment diagram

10. Flowchart

11. Screen Shot

12. Quality Assurance

13. Software and Hardware Requirements

14. Team Members

15. References

16. Project log

# Table of Diagrams

3.1. Architecture Diagram

6.1. Waterfall model

DFD Diagrams

8.1: DFD Level 0

8.2: DFD Level 1.1

8.3: DFD Level 1.2

8.4: DFD Level 1.3

8.5: DFD Level 1.4

8.6: DFD Level 2.1

8.7: DFD Level 2.2

UML Diagrams

9.1.1: Use case 1

9.1.2: Use case 2

9.1.3: Use case 3

9.2.1: Class diagram

9.3.1: Sequence diagram 1

9.3.2: Sequence diagram 2

9.3.3: Sequence diagram 3

9.4.1: State diagram

9.5.1: Activity diagram

9.6.1: Deployment diagram

Flowcharts

10.1: Client Side Activity

10.2: About us clicked

10.3: Start button clicked

**Live Video Streaming**

**(Android)**

Stream live video using a mobile camera to a distant/remote computer and view the same on the web browser.

**1. Problem Definition:**

Current solutions, first streams the video to the server and save the same. It is then broadcasted to the client’s browser. This type of streaming is not live (i.e. stored and then broadcasted).

**2. Project Scope:**

This project allows a real-time video streaming service from an Android mobile device’s camera to a server. The real-time video can then be viewed from a web browser on the client’s computer. The project builds on open source code and open protocols to implement a set of software components that successfully stream live video using NanoHTTPD. It is an open-source, small-footprint web server that is suitable for embedded applications, written in the Java 1.1 programming language.

Users will have the ability to broadcast news and events live, using only an Android based mobile devices and an internet connection via the cellular network or Wi-Fi.

All these phases are cascaded to each other so that the next phase is started as and when a defined set of goals are achieved for first phase and it is signed off. Hence the peculiar name. All the methods and processes undertaken in the model are more visible.

**3. Architecture:**

The following figure presents architecture overview of the project. When Application is on in Android phone ,It shows the start button ,when we click on the start button, the Application starts capturing frames, which is in the format of YUV, then it converts into the RGB format and finally stored in the JPEG format.

The stored frames are forwarded to the web browser (client) by using the NanoHTTPD web server.

**Web**

**Browser**

**Android Mobile Camera**

**Frames**

**NanoHTTPD**

**Web Server**

**Fig 3.1: Architecture Diagram**

**Android Mobile Device:**

Android is a recently developed operating system designed for mobile devices. It was developed by Google and uses a Linux based kernel, Java compatible libraries along with the just-in-time compiler for development in the Java programming language. It supports many hardware components. Common hardware consists of cameras, a Wi-Fi communications chip, Bluetooth sender and receiver and a color touch screen. The Android Application Program Interface (API) contains many functions and classes to control the cellular devices. This functionality is all available in a single device with at least a day worth battery life.

In our project we use platform Android 2.2.

**NanoHTTPD:**

An open source, simple, tiny (1 java file) and nicely embeddable HTTP server in Java.

NanoHTTPD is an open-source, small-footprint web server that is suitable for embedded applications, written in the Java 1.1 programming language. The source code consists of a single *.java* file. It can be used as a library component in developing other software (such as measurement, science and databaseapplications) or as a standalone ad-hoc style HTTP daemon for serving files. Due to independence from Java features beyond JDK 1.1, NanoHTTPD is suited for embedded application development, and has been used to build, for example, Android software.

The original version, released in 2003, only included simple HTTP 1.0 features, but the software has been since forked and extended to support some more advanced techniques such as HTML5 video streaming or HTTP uploading through multipart extensions.

**Ways to use:**

* Run as a standalone app (serves files from current directory and shows requests)
* Subclass serve() and embed to your own program Call serveFile() from serve() with your own base directory
* To test file uploading, try browsing file-upload-test.htm through NanoHTTPD, upload something and watch the console output

**Features and Limitations of NanoHTTPD:**

* Only one Java file
* Java 1.1 compatible
* Released as open source, Modified BSD license
* No fixed config files, logging, authorization etc. (Implement by yourself if you need them.)
* Supports parameter parsing of GET and POST methods (+ rudimentary PUT support in 1.25)
* Parameter names must be unique. (Adding support to multiple instance of a parameter is not difficult, but would make the interface a bit more cumbersome to use.)
* Supports both dynamic content and file serving
* Supports file upload (since version 1.2, 2010)
* Never caches anything
* Doesn't limit bandwidth, request time or simultaneous connections
* Default code serves files and shows all HTTP parameters and headers
* File server supports directory listing, index.html and index.htm
* File server supports partial content (streaming)
* File server supports E Tags
* File server does the 301 redirection trick for directories.
* File server supports simple skipping for files (continue download)
* File server serves also very long files without memory overhead
* Contains a built-in list of most common mime types
* All header names are converted lowercase so they don't vary between browsers/clients

**4. Coding Standards:**

General coding standards pertain to how the developer writes code.

General Software Coding Standards and Guidelines:

1. Inline Comments

Inline comments are used to explain the functioning of the task.

1. Naming Convention

a. Classes

Names of classes must start with Capital Letter. They should follow Upper Camel Case.

b. Methods

Methods should be in Lower Camel Case; that is, with the first letter lowercase and the first letters of subsequent words in uppercase.

c. Variables

Variable names should be short yet meaningful. They should follow Lower Camel Case.

d. Constants

Names should be in uppercase.

**5. Software Development Life Cycle:**

A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the [development of a software product](http://en.wikipedia.org/wiki/Software_development). There are several [models](http://en.wikipedia.org/wiki/Software_development_process#Software_development_models) for such structures, each describing approaches to a variety of [tasks or activities](http://en.wikipedia.org/wiki/Phases_of_the_software_development_cycle) that take place during the development process.

**6. Waterfall Model:**

This project follows 'The Waterfall Model' for the SDLC. In this approach, the whole process of software development is divided into separate phases. These phases are:

1. Requirement specifications phase
2. Software design
3. Implementation
4. Testing and maintenance

All these phases are cascaded to each other so that the next phase is started as and when a defined set of goals are achieved for first phase and it is signed off. Hence the peculiar name. All the methods and processes undertaken in the model are more visible.



**Fig 6.1: Waterfall model**

**Reasons to follow this Approach:**

The project requirements are very well known, clear and are fixed. There are no ambiguous requirements. Similarly, the technology is going to be used is also known. As every phase is fixed, there is no need of revisiting.

**7. Feasibility Study:**

The final step of initial investigation of system is the feasibility study. It is carried out to

check the workability of the candidate system. It’s impact on the organizational ability to

meet user needs and effective use of the system. Feasibility is the testing of the proposed

system according to its workability.

A feasibility study is carried out to select the best system that meets all performance

requirements. The result of feasibility study is a formal proposal. This is simply a report-

a formal document detailing the nature and scope of the proposed solution. The proposal

summarizes what is known and what is going to be done.

There are three key consideration involved in the feasibility analysis:

• Technical Feasibility.

• Economical Feasibility.

• Operational Feasibility.

**Technical Feasibility:**

The technical feasibility study compares the level of technology available in the  software development firm and the level of technology required for the development of   the product. Here the level of technology consists of the programming   language, the hardware resources, software recourses.

**Our technical feasibility parameters involves:**

* Do we have the right technical inputs / clarity of this project?
* Do we have the right technical skills to take up this project?
* Do we have the right tools to take up this project?
* Is there any training required / technical mentoring?
* Do we have the right understanding of the concept and deliverable that the

customer desires to have?

* Did we analyze the right way to do this project?

An Androidmobile device combined with its camera and internet capabilities is used to stream real-time video to a web page.

**NanoHTTPD Web Server:**

NanoHTTPD is an open-source, small-footprint web server that is suitable for embedded applications, written in the Java 1.1 programming language.

**Features:**

* Only one Java file
* Java 1.1 compatible
* Released as open source, Modified BSD license
* No fixed config files, logging, authorization etc. (Implement by yourself if you need them.)
* Supports parameter parsing of GET and POST methods (+ rudimentary PUT support in 1.25)
* Supports both dynamic content and file serving
* Supports file upload (since version 1.2, 2010)
* Doesn't limit bandwidth, request time or simultaneous connections.

**Android Development Tools (ADT)** :It is a plug-in for the Eclipse IDE that is designed to give you a powerful, integrated environment to build Android applications. Developing in Eclipse with ADT is required and is the better way to get started. ADT extends the capabilities of Eclipse viz: to let you quickly set up new Android projects, create an application UI.

**Eclipse (Helios):** Its free a software tool, **open implementation.** It’s release also features new [Linux tools](http://www.eclipse.org/linuxtools/) and JavaScript tools.

**SVN (Sub versioning):** **Apache Subversion** (often abbreviated **SVN**, after the command name *svn*) is a [software versioning](http://en.wikipedia.org/wiki/Software_versioning) and [revision control](http://en.wikipedia.org/wiki/Revision_control) system distributed under an [open source license](http://en.wikipedia.org/wiki/Open_source_license). Developers use Subversion to maintain current and historical versions of files such as [source code](http://en.wikipedia.org/wiki/Source_code), web pages, and documentation. It’s goal is to be a mostly-compatible successor to the widely used [Concurrent Versions System](http://en.wikipedia.org/wiki/Concurrent_Versions_System) (CVS).Subversion is a full-featured source control technology and an integral part of Collab Net. Subversion is a client-server system. You must have a Subversion client on your local machine to access the Subversion server for projects hosted on the site. We use TortoiseSVN1.7.7 for our project.

**Economical Feasibility:**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system or project benefit of the proposed system. More commonly known as [cost/benefit analysis](http://en.wikipedia.org/wiki/Cost-benefit_analysis), the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs.

Return on investment is also important. [Analysis](http://www.businessdictionary.com/definition/analysis.html) of a [project's](http://www.businessdictionary.com/definition/project.html) [costs](http://www.businessdictionary.com/definition/costs.html) and [revenues](http://www.businessdictionary.com/definition/revenue.html) in an effort to determine whether or not it is logical and possible to complete. The objective of the economic feasibility is to develop a financial model of the business venture. The product of this step is a complete integration of the technical product information and the market study into one or more break-even financial models.

The estimated cost usually considers the whole cost of ownership which includes:

1. Ongoing support

2. Maintenance cost

3. Acquisition cost

In this project, financial requirements are very low, required things are:

**1. Eclipse (Helios):**

It is a free software tool.

**2**. **Android Development Tools (ADT):**

It helps us to develop mobile applications on the Android platform. It is a plug-in for the Eclipse IDE that is designed to give you a powerful, integrated environment to build android applications. It is also a free tool.

**3. Android based mobile** device with a built in camera.

**4**. **NanoHTTPD web server**.

**Operational Feasibility:**

Operational feasibility is a measure of how well a proposed system solves the problems, takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The willingness and ability of management, employees, customers and suppliers to operate and support a proposed system. Operational feasibility depends on several vital issues. For example, consider the following questions:  
  
1. Does management support the project?

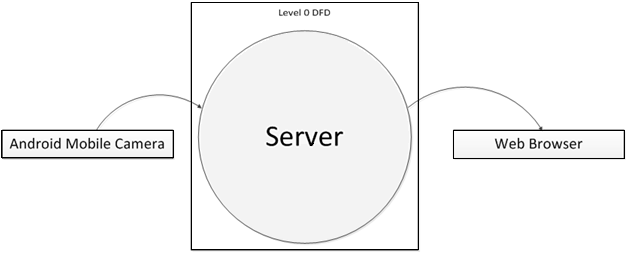
2. Do customers support the project?

3. Do users see the need for change?

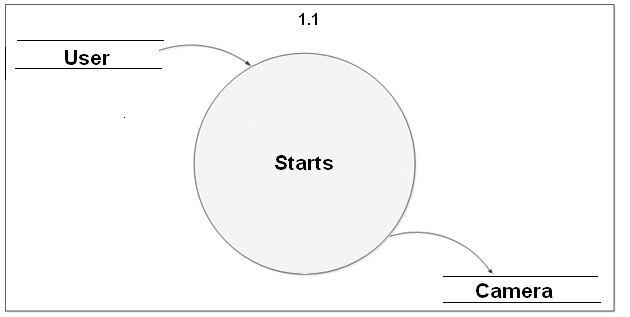
Operational feasibility is mainly concerned with issues like whether the system will be used if it is developed and implemented. A system that has operational feasibility is the one that will be used effectively after it has been developed. The proposed system works effectively on different Android mobile devices. Analyzes the inside operations on how a deemed process will work, be implemented, and dealing with change resistance and acceptance.

**8. Data Flow Diagrams (D.F.D.):**

**Level 0:**

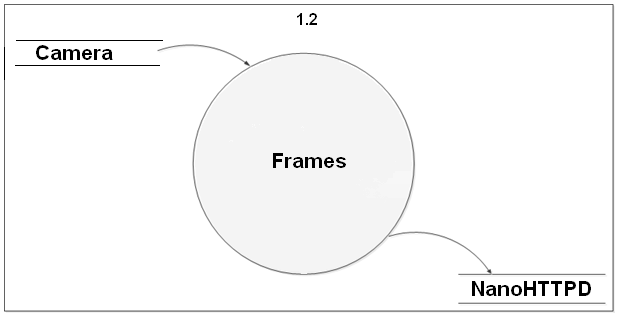


**Fig 8.1: DFD Level 0**

**Level 1.1:**

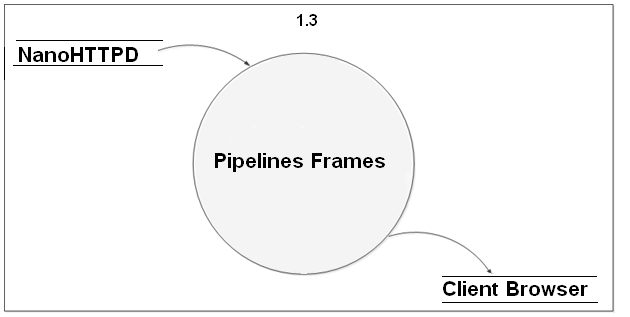
**Fig 8.2: DFD Level 1.1**

**Level 1.2:**

****

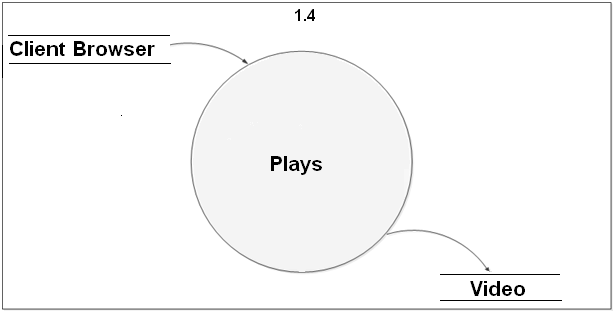
**Fig 8.3: DFD Level 1.2**

**Level 1.3:**

****

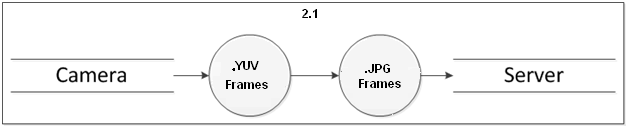
**Fig 8.4: DFD Level 1.3**

**Level 1.4:**

****

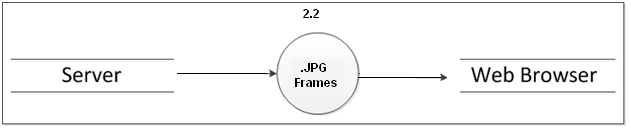
**Fig 8.5: DFD Level 1.4**

**Level 2.1:**

****

**Fig 8.6: DFD Level 2.1**

**Level 2.2:**

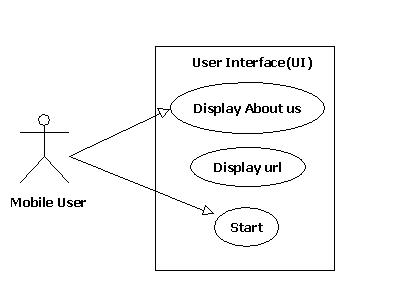
****

**Fig 8.7: DFD Level 2.2**

# 9. UML Diagrams:

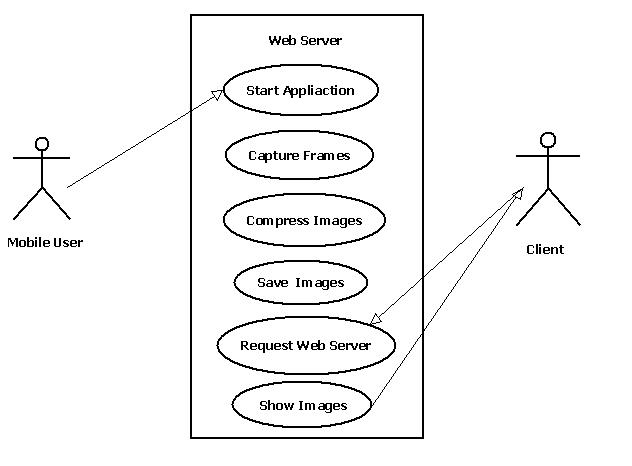
**9.1 Use Case diagram:**

* **Main Application start:**

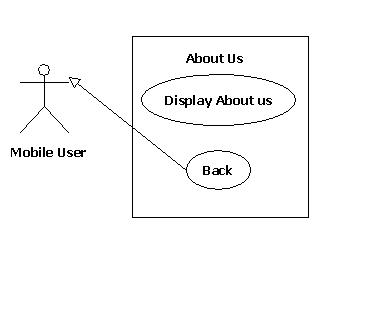
****

**Fig 9.1.1: Use case 1**

* **When User click on start:**

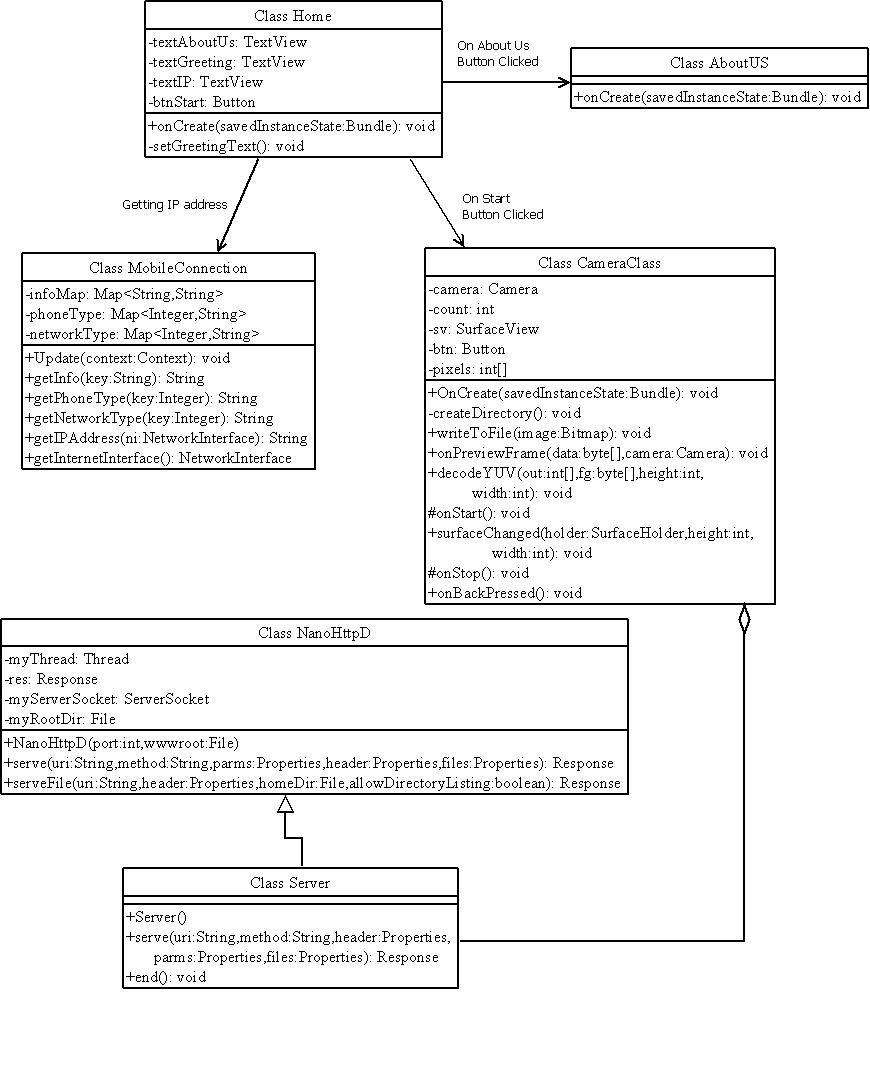
****

**Fig 9.1.2: Use Case 2**

**When user click on About us:**

**Fig 9.1.3: Use case 3**

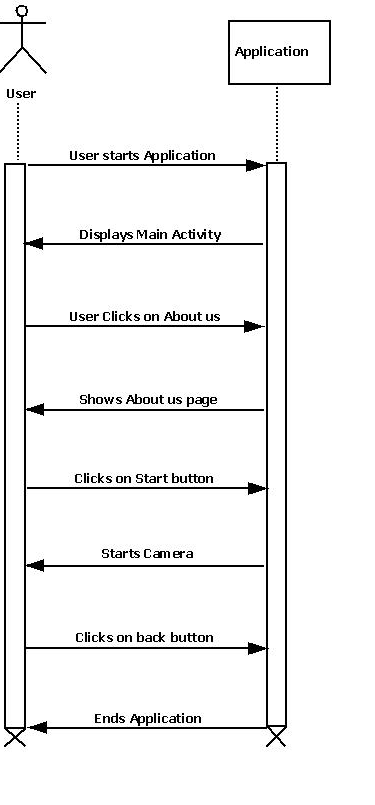
**9.2 Class diagram:**



**Fig 9.2.1: Class diagram**

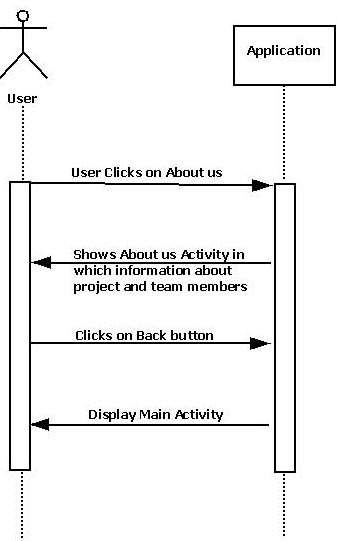
**9.3 Sequence diagram:**

* **When user starts Application:**



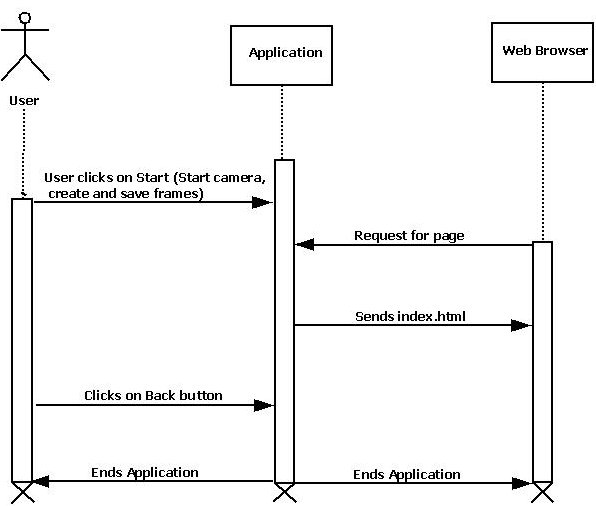
**Fig 9.3.1: Sequence diagram 1**

* **When User Clicks on About us page:**

****

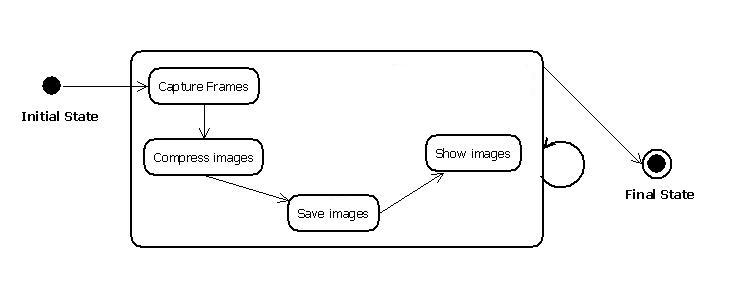
**Fig 9.3.2: Sequence diagram 2**

* **When user clicks on start:**

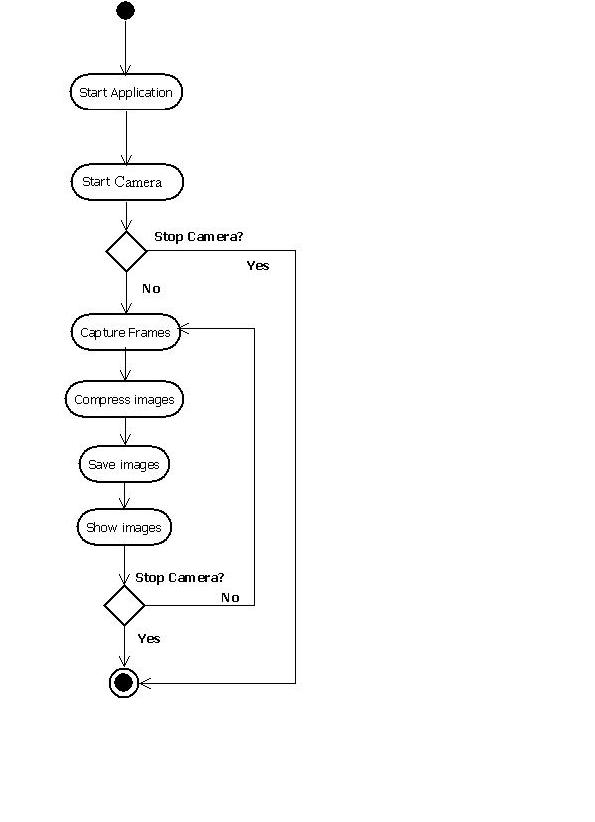
****

**Fig 9.3.3: Sequence diagram 3**

**9.4 State diagram:**

****

**Fig 9.4.1: State diagram**

**9.5 Activity diagram:**

**Fig 9.5.1: Activity diagram**

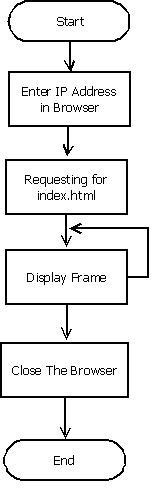
**9.6 Deployment diagram:**

# 

**Fig 9.6.1: Deployment diagram**

# 10. Flowcharts:

* **Client side activity:**

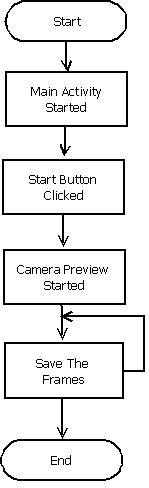


**Fig 10.1: Client side Activity**

# About us clicked

# Fig 10.2: About us clicked

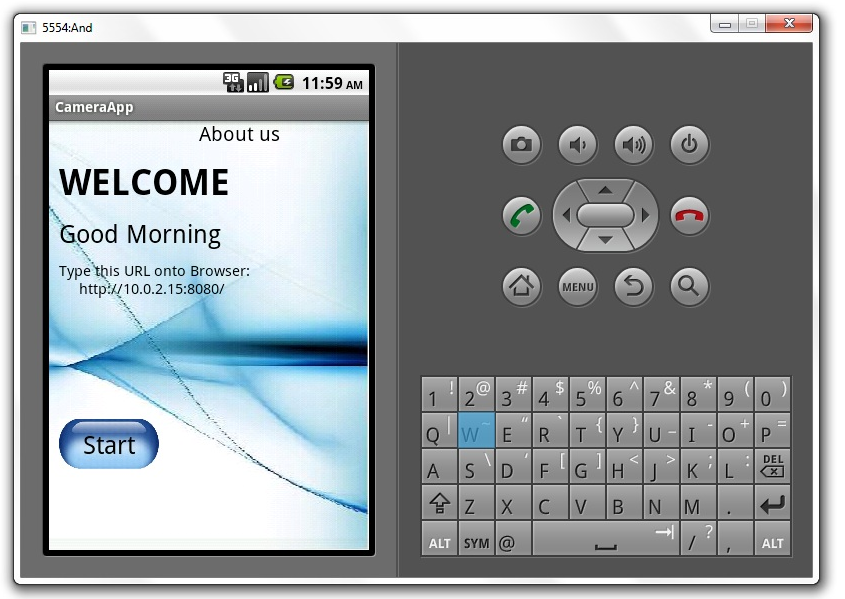
# Start button clicked:



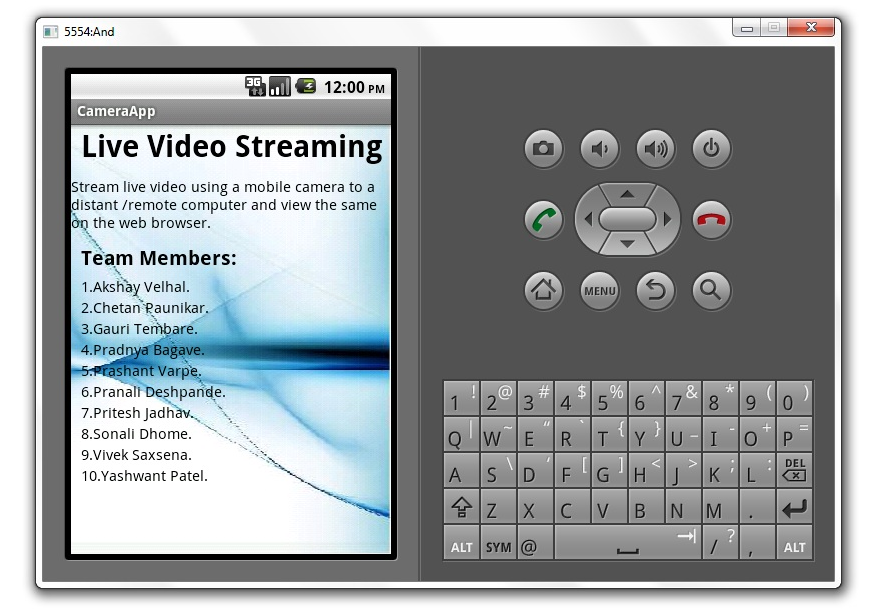
**Fig 10.3: Start button clicked**

# 11. Screen Shots:

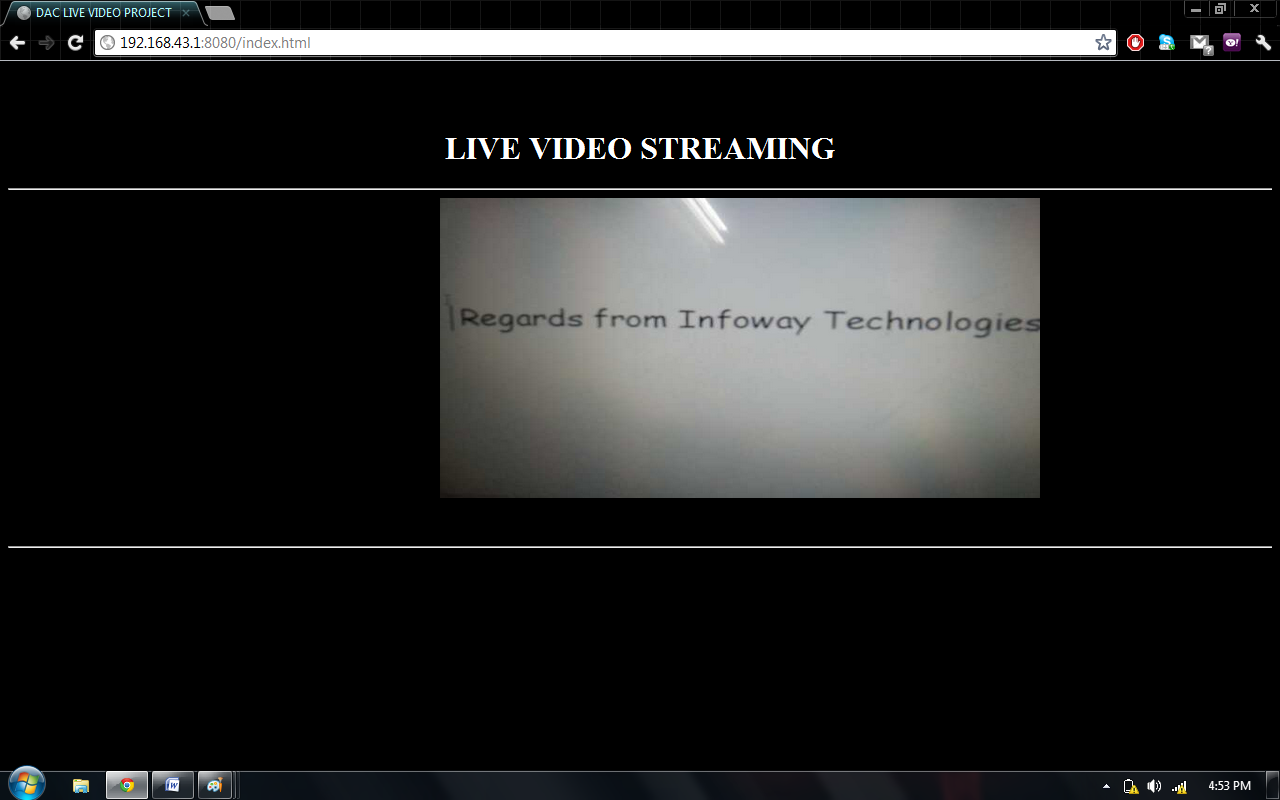
* **Home activity:**

****

* **About us Activity:**

****

* **Sample Output:**

****

# 12. Use of Quality Assurance Practices:

Quality Assurance is a broad practice used for assuring the quality of products or services. Quality Assurance makes sure the project will be completed based on the specifications, standards and functionality requirement without defects and possible problems.

It monitors and tries to improve the development process from the beginning of the project.

Quality assurance function of an organization uses a number of tools for enhancing the quality practices. These tools vary from simple techniques to sophisticated software systems. To achieve better quality we are using Google SVN (subversion).

**13. Software and Hardware Requirements:**

**Software:**

* Eclipse (Helios)
* ADT plug-in(Android Development Tools)
* Android SDK
* Web Browser

**Hardware**:

* Android based mobile with embedded camera.
* An implementation of NanoHTTPD Web server in order to transfer the frames to viewing clients.
* 500 kilobytes space on Android mobile phone.

**14. Team Members:**

* 1. Akshay Velhal.
  2. Yashwant Patel.
  3. Chetan Paunikar.
  4. Pritesh Jadhav.
  5. Vivek Saxsena.
  6. Prashant Varpe.
  7. Pranali Deshpande.
  8. Sonali Dhome.
  9. Gauri Tembare.
  10. Pradnya Bagave

**15.References:**

|  |  |
| --- | --- |
| **Topic** | **URL** |
| Telephone Manager | <http://developer.android.com/reference/android/telephony/TelephonyManager.html> |
| Connectivity Manager | <http://developer.android.com/reference/android/net/ConnectivityManager.html> |
| Network Info | <http://developer.android.com/reference/android/net/NetworkInfo.html> |
| Network Interface | <http://developer.android.com/reference/java/net/NetworkInterface.html> |
| WIFI | <http://developer.android.com/reference/android/net/wifi/package-summary.html> |
| WifiInfo | <http://developer.android.com/reference/android/net/wifi/WifiInfo.html> |
| YUV to RGB Conversion | <http://en.wikipedia.org/wiki/YUV>  <http://www.fourcc.org/yuv.php>  <http://support.microsoft.com/kb/294880> |
| Uml diagrams | <http://www.smartdraw.com/resources/tutorials/uml-diagrams/#/resources/tutorials/Introduction-to-UML>  <http://www.uml-diagrams.org/> |
| Rtsp protocol implementation | <http://www.agilentia.ch/images/stories/documents/Agilentia_Basic_Streaming_Technology_and_RTSP_Protocol.pdf>  <http://www.ietf.org/rfc/rfc2326.txt> |
| NanoHTTPD | <http://elonen.iki.fi/code/nanohttpd>  <http://en.wikipedia.org/wiki/NanoHTTPD> |
| Frame extracting | <http://android.bigresource.com/Android-Extract-video-frame-at-particular-time-4JbvFdgXg.html>  <http://android.bigresource.com/Android-Capture-frames-from-video-file--8BnOyan8X.html> |
| Image Compression | <http://developer.android.com/reference/android/graphics/Bitmap.html>  <http://www.coderanch.com/t/579019/Android/Mobile/compress-image-android> |
| Preview Callback | <http://developer.android.com/reference/android/hardware/Camera.PreviewCallback.html>  <http://stackoverflow.com/questions/3398363/how-to-define-callbacks-in-android> |
| SVN | <http://tortoisesvn.net/>  <http://subversion.apache.org/>  <http://en.wikipedia.org/wiki/Apache_Subversion> |

**16.Project Log:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Akshay** | **Yashwant** | **Pritesh** | **Chetan** | **Vivek** | **Prashant** | **Pranali** | **Sonali** | **Gauri** | **Pradnya** |
| 11Jul | SVN R&D | SVN R&D | Pipelining(Research) Joining two IP’s on the server | Pipelining(Research) Joining two IP’s on the server | RTSP Protocol | RTSP Protocol | JSP (How to play local video on the hard disk to a browser). | JSP (How to play local video on the hard disk to a browser). | Hardware, Software requirements and Project scope. | Hardware, Software requirements and Project scope. |
| 12Jul | Bugzilla,Review of documents, SVN ppt | Bugzilla, SVN ppt | Feasibility Study (Documentation) | DFD level 0,1,2(Documentation) | RTSP Protocol | DFD level 0,1,2(Documentation) & RTSP | Quality Assurance and SDLC(Documentation) | Quality Assurance and SDLC(Documentation) | Feasibility Study(Documentation) | Absent |
| 13Jul | Java code to send Video, Review Documents | Java code to send Video,Review Documents | Documentation & Arch Diagram | Connect Camera & Display On Canvas(Camera On Off) | Connect Camera & Display On Canvas(Camera On Off) | GUI Design of Android | Embed Flash Plugin & code to stream video | Embed Flash Plugin & code to stream video,Review Documents | Documentation & Arch Diagram | GUI Design of Android |
| 16Jul | Creation of Host | Creation of Host | Documentation & Arch Diagram | Camera GUI | Streaming Video | Streaming Video | Streaming Video | Absent | Streaming Video | Camera GUI,Streaming Video |
| 17Jul | Unit Testing, Streaming Video with file Descriptor | Unit Testing, Streaming Video with file Descriptor | Unit Testing | Absent | Streaming Video with file Descriptor | NanoHTTPD | Unit Testing, Streaming Video with file Descriptor | Unit Testing | NanoHTTPD | Camera GUI,Streaming Video |
| 18Jul | Establish Connection from mobile, Streaming Video with file Descriptor | Establish Connection from mobile | Program to Find Mobile IP | Absent | Streaming Video with file Descriptor | Camera GUI | NanoHTTPD, Streaming Video with file Descriptor | NanoHTTPD, Streaming Video with file Descriptor | Program To find Mobile IP | Media Recording |
| 19Jul | Establish Connection from mobile, Find Mobile IP, RTSP R&D, | RTSP R&D,Media Recording | RTSP R&D,Media Recording | Writing Test Cases | Video Header Problem | Streaming Video with file Descriptor | NanoHTTPD server | NanoHTTPD  server | Absent | RTSP R&D |
| 20Jul | Integration & Testing over various situations. Creating separate plans for both the projects | Integration & Testing over various situations. Writing Test Cases | Writing Test Cases | Absent | Video Header Problem | Writing Test Cases | Send Video over socket | Send Video over socket | Absent | Integration & Testing over various situations |
| 21Jul | Pick out farmes from video, send image over network through socket. Streaming Video with file Descriptor | Pick out farmes from video(plan 1), Streaming Video with file Descriptor  (plan 2) | To extract frame from video(plan2) | Display Image on jsp & change it using javascript(plan1) | Absent | Absent | send image over network through socket (plan 1) | Streaming Video with file Descriptor  (plan 2) | To extract frame from video (plan 1) | To extract frame from video (plan 1) |
| 23Jul | Optimize the Image sending process & Testing (plan 1) | Redocumentation, Streaming Video creating small files of 5 seconds (plan 2) | Streaming Video creating small files of 5 seconds (plan 2) | DFD | Streaming Video with file Descriptor  (plan 2) | DFD | Optimize the Image sending process & Testing (plan 1) | Streaming Video with file Descriptor  (plan 2) | Redocumentation | To extract frame from video (plan 1) |
| 24Jul | Save Image Preview & send it over socket(Integeration & Testing). (plan 1) Streaming Video creating small files of 5 seconds (plan 2) | Display Image on html & change it using javascript (plan 1), Streaming Video creating small files of 5 seconds (plan 2) | Reduce Image size by reducing the camera resolution and quality (plan 1) | Reduce Image size by reducing the camera resolution and quality (plan 1) | Streaming Video creating small files of 5 seconds (plan 2) | Reduce Image size by reducing the camera resolution and quality(plan 2) | Optimize the Image sending process & Testing (plan 1) | Save Image Preview & send it over socket(Integeration & Testing). | Display Image on html & change it using javascript, (plan 1) | Streaming Video creating small files of 5 seconds(Plan 2) |
| 25Jul | Integrate Connection Class with Project & Display IP Address onto GUI. Test the Delay caused in frames & reduce it  (plan 1) | Test the Delay caused in frames & reduce it (plan 1) | UML Diagrams (Use Case & Activity & State Diagram), (plan 1) | Re-documentation | Streaming Video creating small files of 5 seconds (plan 2) | Re-documentation | Test the Delay caused in frames & reduce it (plan 1) | Test the Delay caused in frames & reduce it (plan 1) | UML Diagrams (Sequence & Deployment) | Attractive UI Design containing a welcome screen, Start button & Display IP Address. Streaming Video (plan 1) |
| 26Jul | Creating a player to display video on html page as a playlist (plan 2) | New & Improved HTML design(plan1), Testing (plan 1) | Testing and test cases (plan 1) | Absent | Creating a player to display video on html page as a playlist (plan 2) | Testing and Test cases(plan 1) | Class Diagram & Documentation review | Class Diagram & Documentation review | Documentation Integration | Testing and test cases |
| 27Jul | Minor Bug fixes in plan1 and plan2 | New & Improved HTML design for plan1, plan2 | Creating a final documentation folder.(includes integration of document files) | Absent | Minor Bug fixes in plan1 and plan2 | Testing | Creating of a product description file, which can be used while presentation of the project | Creating of a product description file, which can be used while presentation of the project | Creating a final documentation folder.(includes integration of document files) | Writing Test cases for plan 1, & fixing some minor bugs. |