|  |
| --- |
| **Sentinel** |
| **Design Document** |
|  |
| This document captures the design details of the Sentinel Application |
|  |
| **Rufus Gnana, Vijay Kumar, Fekri Kassem, Kong Tsao** |
| **11/1/2011** |
|  |

Contents

[Version Tracking 4](#_Toc307960683)

[1. Introduction 5](#_Toc307960684)

[2. Application Features 6](#_Toc307960685)

[2.1 Overview 6](#_Toc307960686)

[2.2 Application Snapshots 7](#_Toc307960687)

[3. Flow of control and data 10](#_Toc307960688)

[4. Use Cases 11](#_Toc307960689)

[4.1 Scenario 1: Video Door Bell 11](#_Toc307960690)

[4.2 Scenario 2: Baby monitor 12](#_Toc307960691)

[5. Design Choices 12](#_Toc307960692)

[5.1 Platform 12](#_Toc307960693)

[5.2 Video format 13](#_Toc307960694)

[5.3 Wireless IP camera used for testing 15](#_Toc307960695)

[5.4 Launching the Sentinel Application 16](#_Toc307960696)

[6. Schemes of Classes used 17](#_Toc307960697)

[6.1 Model 17](#_Toc307960698)

[6.2 View 19](#_Toc307960699)

[6.3 Controller 23](#_Toc307960700)

[7. Data Management 25](#_Toc307960701)

[7.1 Camera Authentication 25](#_Toc307960702)

[7.2 Camera Interaction 26](#_Toc307960703)

[8. Security 27](#_Toc307960704)

[9. Milestones and timeframes 28](#_Toc307960705)

**List of figures**

[Figure 1 Camera Selection 7](file:///C:\D_colon\Rufus\Studies\3rd%20sem\Mobile%20App\Idea\Design%20document.docx#_Toc307960808)

[Figure 2 User Authentication 8](#_Toc307960809)

[Figure 3 Live Camera Feed 8](#_Toc307960810)

[Figure 4 Removal of a configured camera 9](#_Toc307960811)

[Figure 5 Control/Data Flow 10](file:///C:\D_colon\Rufus\Studies\3rd%20sem\Mobile%20App\Idea\Design%20document.docx#_Toc307960812)

[Figure 6 System Architecture mandated by HLS format 14](#_Toc307960813)

# Version Tracking

|  |  |  |  |
| --- | --- | --- | --- |
| Sl No | Date | Version Number | Version updates |
| 1 | 11/01/2011 | V1\_0 | First Version |
|  |  |  |  |
|  |  |  |  |

# Introduction

This document captures the following details regarding the Sentinel

* Application features
* Application snapshots
* Flow of control and data
* Use cases
  + Video Door Bell
  + Baby Monitor
* Design decisions
  + Platform
  + Video format
  + Wireless IP camera used for testing
  + Launching the Sentinel Application
* Detailed description of the Classes used
  + Model
  + View
  + Controller
* Data management
  + Storage considerations
* Security Considerations
* Milestone and timeframes

# Application Features

# Overview

This section details the features that are intended to be supported as a part of the development activities for the Sentinel.

* Identification of Wireless IP Cameras connected on the network
  + The application will provide an interface for the user to scan the Wireless IP Cameras on the network the phone is currently connected to.
  + The user can then select the camera that needs to be connected to.
  + The user can also configure a particular camera to be the default. Feed from this camera will be displayed by default when the user starts the application.
  + The user will also be able to change the default camera
  + If the user does not configure a default camera, he shall be presented with the camera selection interface when the application is launched
* Authentication of the user credentials
  + Once the user selects a camera, if
    - The user has not configured or saved the camera credentials before, the application shall present an interface for the user to key in the user name and password for the camera
    - The user has configured and saved the camera login credentials before, the application shall automatically authenticate the user.
* Saving the user credentials for the camera locally on the phone
  + The user can choose to save the login credentials for the camera locally on the phone
* Capturing the MJPEG stream from the camera on the phone and displaying it in a 320x240 frame size
  + Each MJPEG image in the stream from the camera will be decoded and displayed on the screen
  + The frame rate of the camera can be configured by the user
* The following camera parameters will be configurable from the application (as a slider bar): Zoom, Pan, Tilt, Brightness, Backlight and Camera frame rate
* The user will also be able to take a snapshot at any instant while viewing the live video
  + The captured image will be stored locally and viewable later

# Application Snapshots

This section provides snapshots of the application and points to the application features captured in the previous section

* User Configurations

Once the user starts the application, the application shall provide the following interface to detect and select a camera

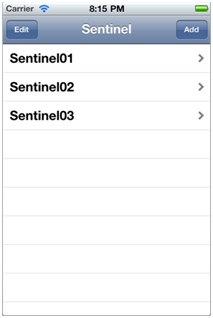


Figure 1 Camera Selection

* User authentication

The user will now be presented with the following interface to authenticate his credentials. The user can choose to save his details on the phone to ease the process of subsequent logins.

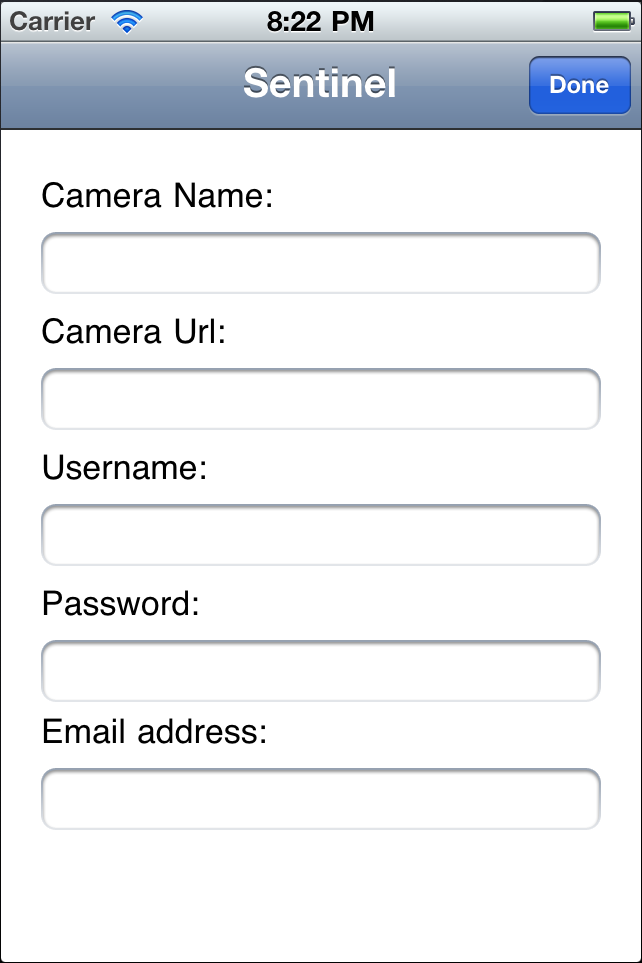
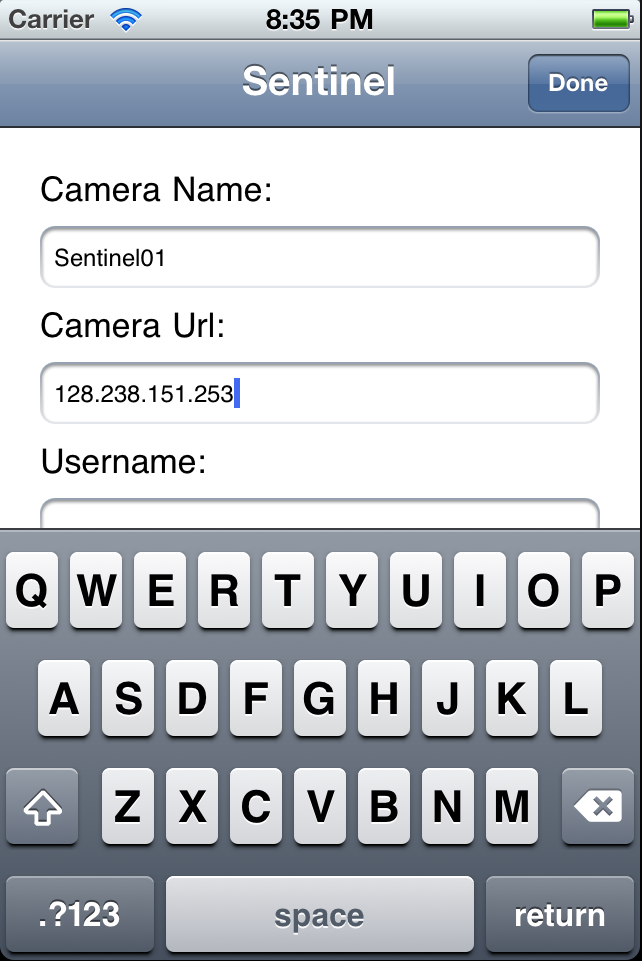
 

Figure 2 User Authentication

* Viewing live feed

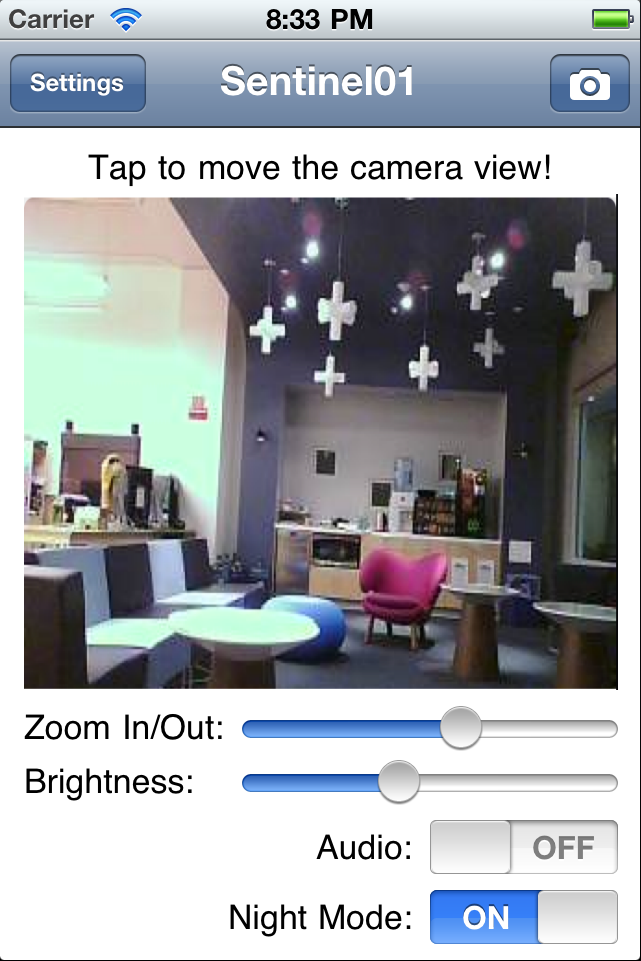
The user shall be presented with the following interface to view the live camera feed

Figure 3 Live Camera Feed

* Deleting a camera from the list

The user can also choose to remove a camera from the configured list

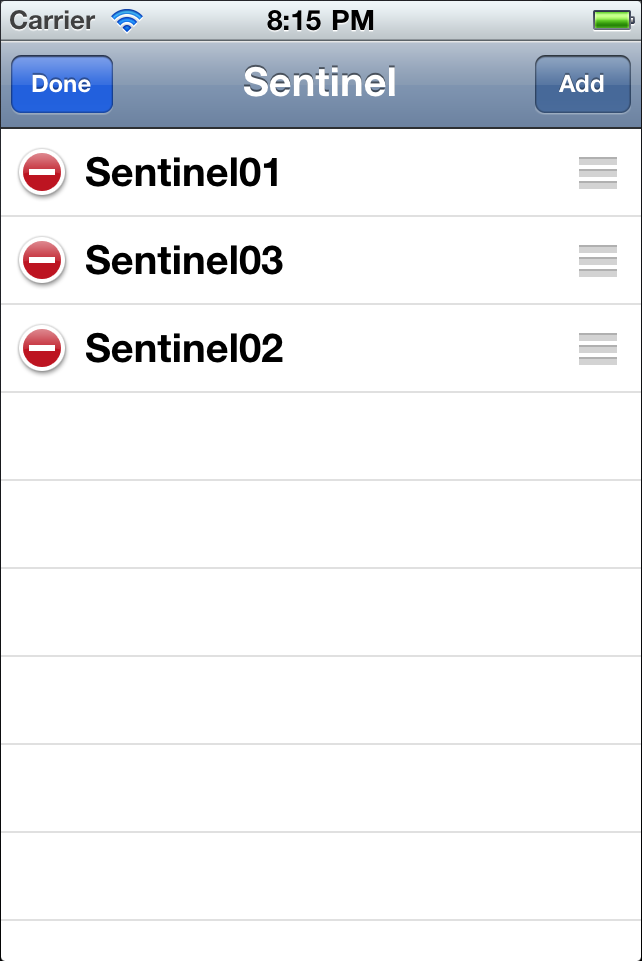


Figure 4 Removal of a configured camera

# Flow of control and data

The flow and data and control are detailed in the flowchart below:

Camera captures and transmits MJPEG frames over the wireless network

Flow of Video Data

Actions performed by the user

Door Bell is pressed



User starts the application

Wait for an MJPEG frame from camera

User Authentication

Decode the frame

Display the frame

Application enters loop

User changes camera controls (Pan, Tilt, Zoom)

Interpret and format an HTTP request to the camera

User closes the application

Application performs a clean exit

Save user credentials

Legend

Actions taken in the Sentinel Application

Flow of Control



Figure 5 Control/Data Flow

The following are the actions initiated by the user that the Sentinel application shall respond to:

* The user starting the application
* The user keying in the login credentials
* The user updating the camera parameters
* The user closing the application

The Sentinel application shall perform the following actions in response to the user’s inputs

* Store the login credentials
* Decode and display the MJPEG frames
* Configure the camera based on the configurations by the user in the user interface
* Clean exit when the user quits the application

# Use Cases

Prerequisites:

* The user should be connected to a wireless network
* A wireless IP camera shall be installed at the door and connected to the same wireless network the user is connected to
* The user shall know the login credentials for the camera
* The Sentinel application shall be installed on the user’s phone

# Scenario 1: Video Door Bell

* The wireless IP camera installed at the door shall be capturing AV data in a continuous fashion. When the user understands that there is someone at the door the shall start the application and
  + If the user has configured a default camera, start viewing the live feed from the camera
  + If the user has not configured a default camera, select a camera (login if necessary) and view the live feed from the camera
* The user can decide to take the “current snapshot” of the camera feed and store the image in the phone
* The user can also control the camera parameters while viewing the live feed from the camera

# Scenario 2: Baby monitor

* The wireless IP camera installed in the baby’s room shall be capturing AV data in a continuous fashion.
* The parent shall start the application and view the live camera feed
* The parent can also control the camera parameters while viewing the live feed from the camera

# Design Choices

# 5.1 Platform

The iOS platform was chosen due to the following considerations

* Support for various display resolutions for the captured video across different cameras will be relatively convenient due to well know dimensions of the display screens on Apple products
* Native support for image decode and display
* Support for a wide range of user interface options
* Increasing market of Apple products

# 5.2 Video format

The [Real-time Streaming Protocol](http://en.wikipedia.org/wiki/Real-time_Streaming_Protocol) (RTSP), [Real-time Transport Protocol](http://en.wikipedia.org/wiki/Real-time_Transport_Protocol) (RTP) and the [Real-time Transport Control Protocol](http://en.wikipedia.org/wiki/Real-time_Transport_Control_Protocol) (RTCP) were specifically designed to stream media over networks. RTSP runs over a variety of transport protocols, while the latter two are built on top of UDP.

HTTP Live Streaming (also known as HLS) is an [HTTP](http://en.wikipedia.org/wiki/HTTP)-based [media streaming](http://en.wikipedia.org/wiki/Media_streaming) [communications protocol](http://en.wikipedia.org/wiki/Protocol_%28computing%29) implemented by [Apple Inc.](http://en.wikipedia.org/wiki/Apple_Inc.) as part of their [QuickTime X](http://en.wikipedia.org/wiki/QuickTime_X) and [iPhone](http://en.wikipedia.org/wiki/IPhone) software systems. It works by breaking the overall stream into a sequence of small HTTP-based file downloads, each download loading one short chunk of an overall potentially unbounded transport stream. As the stream is played, the client may select from a number of different alternate streams containing the same material encoded at a variety of data rates, allowing the streaming session to adapt to the available data rate. At the start of the streaming session, it downloads an [extended M3U](http://en.wikipedia.org/wiki/Extended_M3U) [playlist](http://en.wikipedia.org/wiki/Playlist) containing the metadata for the various sub-streams which are available.

Both the above methods were discarded for the prototype due to the following reasons:

* Why use of RTSP and RTCP was decided against
  + RTSP , though a well proved video streaming format, is not supported by a wide variety of cameras
  + Almost all IP cameras support configuration of camera parameters via HTTP messages
* Why HLS was decided against
  + HLS, though easily compatible with Apple devices, required the following system architecture

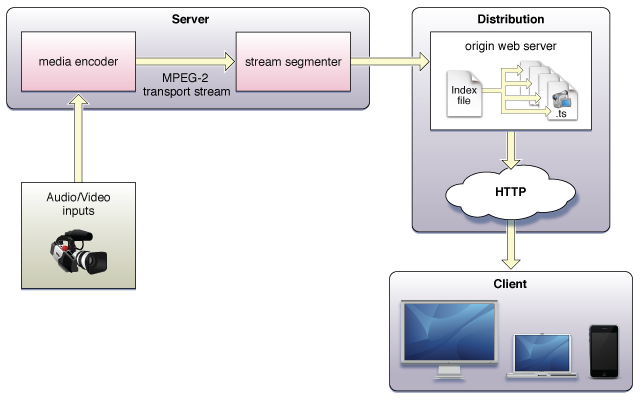


Figure 6 System Architecture mandated by HLS format

* + The above architecture needs a dedicated server between the streaming content provider (the camera) and the device. Hence HLS was decided against.

It was finally decided that we would support Motion JPEG in our initial prototype due to the following considerations:

* A thorough study of the wireless video cameras available in the market revealed that MJPEG was the most common format supported.
* The iOS webview class had native support for displaying a JPEG image received as a HTTP message

# 5.3 Wireless IP camera used for testing

We plan to use the “**Panasonic BL-C210**” to test the Sentinel. It is imperative that we describe what camera we plan to use to test the Sentinel and the reasons behind the decision because:

* The design of the Sentinel application is greatly influenced by the results we observe as we test the application with the camera.
* The test camera needs to be a good representative of the common cameras available in the market

The Panasonic BL-C210 supports the following features that make it a good representative of the common cameras found in the market. It supports the following features:

|  |  |
| --- | --- |
| Feature | Details |
| Image compression | H.264, MPEG-4, Motion JPEG for moving image display |
| Video resolution | H.264: 640x480, 320 x 240 MPEG-4: 640 x 480, 320 x 240, 192 x 144 JPEG: 640 x 480, 320 x 240, 192 x 144 |
| Frame rate | Max. 30 frames/sec |
| Security | User ID / Password/SSL |
| Supported protocols | IPv4: TCP, UDP, IP, HTTP, FTP, SMTP, DHCP, DNS, ARP, ICMP, POP3, NTP, Auto IP, SMTP Authentication, RTP, RTSP, RTCP IPv6: TCP, UDP, IP, HTTP, FTP, SMTP, NTP, RTP, RTSP, RTCP |
| Image transfer method | SMTP, FTP, HTTP |
| Triggers | Timer, Sensor, Alarm 1, Alarm 2, Motion Detection, Sound Detection |

# 5.4 Launching the Sentinel Application

The Sentinel application can be launched in the following ways

* The application auto-launches when someone rings the door bell
* The application will be started manually by the user when someone rings the door bell.

It has been decided that the Sentinel application will be launched manually by the user due to the following reasons:

* An auto launch would require that the Wireless camera to be connected to the door bell system of the user. This would then put the onus of installing the video camera with the door bell on the user.
* On the other hand if the user manually launches the application, an off the shelf camera can be used with the Sentinel application without any further hassles.

# Schemes of Classes used

# 6.1 Model

|  |  |  |
| --- | --- | --- |
| Class Name | NSURL | |
| Need | The NSURL object will be used to connect to the camera discovered on the network. This will also be used to refer to the MJPEG images available on the camera | |
| Method’s Used | URLWithString | Used to create a NSURL object with the string that represents the IP camera over the network |

|  |  |  |
| --- | --- | --- |
| Class Name | NSURLConnection | |
| Need | This class is used to load the URL request. This class enables asynchronous load of a URL request. | |
| Method’s Used | connection:didCancelAuthenticationChallenge: | This message will be sent to the delegate if the connection cancels the authentication challenge due to the protocol implementation encountering an error. This authentication of the user shall happen in this function |
| connection:didFailWithError: | This message will be used to track failure to connect to the camera |
| [connectionDidFinishLoading:](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/DeprecationAppendix/AppendixADeprecatedAPI.html#//apple_ref/occ/instm/NSObject/connectionDidFinishLoading:) | This message will be used to track successful connection establishment with the camera |

|  |  |  |
| --- | --- | --- |
| Class Name | NSURLRequest | |
| Need | Used to load request in a manner independent of protocol and URL scheme. | |
| Method’s Used | requestWithURL | This method is used to create and return a URL request for a specified URL with default cache policy and timeout value. |

|  |  |  |
| --- | --- | --- |
| Class Name | NSUserDefaults | |
| Need | This class is used to save and retrieve in the persistence storage default settings for users such as the default camera and default camera settings. | |
| Method’s Used | standardUserDefaults | Creates an instance of NSUserDefaults. |
| Set<type>: forKey: | Add a new setting or attribute for this user. |
| synchronize | Saves the settings an attribute. |
| <type>forKey | Retrieve a setting or an attribute. |

# 6.2 View

|  |  |  |
| --- | --- | --- |
| Class Name | UIView | |
| Need | The UIWebView class will be used to display the Motion-JPEG content on the main window. The MJPEG content will be available as an image on the networked camera. | |
| Method’s Used | initWithFrame | Used to initialize the video display window with the 320x240 frame size |

|  |  |  |
| --- | --- | --- |
| Class Name | UILabel | |
| Need | Used to display different text in the views. | |
| Method’s Used | Text | To get or set the text being displayed by the label. |

|  |  |  |
| --- | --- | --- |
| Class Name | UIButton | |
| Need | Used for supporting user interaction with application. Most of the attributes of the UIButton will be set in the Interface Builder. | |
| Method’s Used | addTarget | Add a handler for button events. |

|  |  |  |
| --- | --- | --- |
| Class Name | UITableViewController | |
| Need | This class is the controller for the UITableView. It will handle retrieving, editing and saving the cameras information. | |
| Method’s Used | add | Adds a new camera details to the table. |
| reorder | Reorder the list of cameras being displayed. |
| edit | Handle editing one of the camera entries. |
| delete | Delete camera information entry. |

|  |  |  |
| --- | --- | --- |
| Class Name | UITableView | |
| Need | This will be used to display the list of the cameras available along with their information. | |
| Method’s Used | viewForHeaderInSection | This will create the view for the new section, which will be the camera’s name. |
| didSelectRowAtIndexPath | Callback that will be handled when the user selects one of the cameras in the table. This will cause navigation to the new view to show the details of the camera. |
| editingStyleForRowAtIndexPath, willBeginEditingAtIndexPath, didEndEditingRowAtIndexPath | These events and callbacks that will be implemented in order to enable editing entries in the list of cameras tables, such as deleting an entry. |
| targetIndexPathForMoveFromRowAtIndexPath | This will be implemented in order to allow re-ordering cameras in the table view. |
| cellForRowAtIndex | This will be called to determine what should be displayed at a given cell in the table view. |
| numberOfRowsInSection | For a given section, this method will be implemented to determine the number of rows in that section. In our case, it will be just one since only have camera settings. |
| numberOfSectionsInTableView | This will be implemented to determine the list of section that should be displayed in the table view which is the list distinct cameras in this case. |
| canEditRowAtIndexPath | This will set all the table rows to be editable since all the cameras can be deleted. |
| canMoveRowAtIndexPath | This will set all the table rows to be moveable to support custom reordering. |

|  |  |  |
| --- | --- | --- |
| Class Name | UITextField | |
| Need | This class will be used to enable adding or editing the existing settings of cameras. | |
| Method’s Used | Text | All the interaction with object is retrieving the text that the user enters or setting the text when the user is modifying the settings of an existing camera. |

|  |  |  |
| --- | --- | --- |
| Class Name | UIAlertView | |
| Need | This class will be used to raise unexpected errors or confirmation with the user. E.g invalid logins, invalid camera details etc. | |
| Method’s Used | initWithTitle | After calling alloc, just set the text that should be displayed to the user. |

|  |  |  |
| --- | --- | --- |
| Class Name | UIBarButtonItem | |
| Need | This will be used along with navigation functionalities to help with moving between views and also for saving the camera details after finish will the fields. | |
| Method’s Used | Target | The object that should handle the interaction with this button, this will likely be the UIViewController or UINavigationController |

|  |  |  |
| --- | --- | --- |
| Class Name | UIScrollView | |
| Need | This class used to handle cases where the size of the view is not big enough to display all the data such as the list of the cameras. | |
| Method’s Used | initWithFrame | Creates an instance of UIScrollView instance and initialize it with frame. |
| pagingEnabled | Enable the paging when scrolling. |
| addSubView | Adds sub views the scroll view. |
| contentSize | To change the view content size. |

|  |  |  |
| --- | --- | --- |
| Class Name | UIWebView | |
| Need | The UIWebView class will be used to display the Motion-JPEG content on the main window. The MJPEG content will be available as an image on the networked camera. | |
| Method’s Used | loadRequest | The MJPEG content to be displayed will be available as a link over the network. This function shall be used to associate the view object with the MJPEG content to display live video in the application |

## 6.3 Controller

|  |  |  |
| --- | --- | --- |
| Class Name | UIViewController | |
| Need | This class is used to manage the view hierarchy. | |
| Method’s Used | loadView | Used to load various the view |

|  |  |
| --- | --- |
| Class Name | UINavigationController |
| Need | This class will be used to manage the navigation. Our app does not heavily use navigation so there is no need for a custom navigation controller. |

|  |  |  |
| --- | --- | --- |
| Class Name | UIViewController | |
| Need | This will handle the interaction with the camera, this is the main display view. | |
| Method’s Used | viewWillAppear/viewDidApear/viewDidLoad/viewWillDisappear/viewDidDisappear | These will be implemented to do the initialization when the view becomes visible and also the cleaning when view disappears. |
| connectCamera | Connect to specific camera. |
| snap | Take snapshot |
| zoom | Will handle the zooming in and zoom out. |
| changeBrightnewss | Will handle changing the brightness by sending http request to the camera with new attributes. Same for Night mode changing. |
| disableAudio | This will be called as result of user disabling the streaming of the audio to the camera. |

# Data Management

# 7.1 Camera Authentication

* Adding camera to the application

|  |  |
| --- | --- |
| Detail | Data Structure |
| Camera Name | NSString |
| Camera Url | NSUrl |
| Username | NSString |
| Password | NSString |
| Email Address | NSString |
| Error Handler | Use NSRegularExpression Class   * Password: More than six digits * Email: email format, should contain “@” * All fields are required, should not be blank |

* Save Camera Info

|  |  |
| --- | --- |
| Detail | Data Structure |
| Information corresponding to each Camera | NSMutableDictionary |
| The list of all cameras will be saved | NSMutableArray of NSMutableDictionaries |
| Camera information array to the local file within the device | NSKeyedArchiver |

* Show Camera Information

|  |  |
| --- | --- |
| Detail | Data Structure |
| To retrieve the camera information | NSKeyedUnarchive |
| Parsing Camera information | NSMutableDictionaries |
| Displaying Camera list | UITableViewCell |

* Delete Camera Information

|  |  |
| --- | --- |
| Detail | Data Structure |
| Locate the path index of the tableview and remove the object from the camera info | list(NSMutableArray) |

## 7.2 Camera Interaction

* Camera authentication:

|  |  |
| --- | --- |
| Detail | Data Structure |
| connect to the Camera | NSURLRequest |

* Retrieve camera live stream:

|  |  |
| --- | --- |
| Detail | Data Structure |
| to load and display the live stream source of the camera | UIWebView |

* 3. Take and store camera snapshot

|  |  |
| --- | --- |
| Detail | Data Structure |
| Displaying the snapshot | UIImageView |
| Saving the captured image on the phone | NSKeyedArchiver |

# Security

IP cameras offer secure data transmission through encryption and authentication methods such as WEP, WPA, WPA2, TKIP, and AES. In addition, the ONVIF Specification standardizes the network interface (on the network layer) of network video products. It defines a network video communication framework based on relevant IETF and Web Services standards including security and IP configuration requirements. Security is a primary component defined by this specification.

The ONVIF specification can be viewed at the following link

<http://www.onvif.org/Documents/Specifications.aspx>

Also, there is the security imposed by the wireless network the user’s phone and the camera are connected to.

# Milestones and timeframes

|  |  |
| --- | --- |
| Week | Task |
| 1 | Introduction |
| 2 | Learning iOS and Objective C |
| 3 | Learning iOS and Objective C + Initial Product Pitch |
| 4 | Learning iOS and Objective C + Team formation + Idea sounding at NYU |
| 5 | Study phase (IP Camera, iPhone native support, Streaming protocols, Core Data) |
| 6 & 7 | Design Discussion (with the Sentinel Team) + Tests on the iPhone with an IP Camera |
| 8 | Design Document Preparation |
| 9 – 11 | Continue the development phase + Testing and validation + Prepare for the first prototype demo |
| 12 | Integration testing + Complete the demo for the first prototype |
| 13 | First prototype presentation |
| 14 | Final presentation |