CIS 4911 – SENIOR PROJECT

Picture Marketing’s Social Wall

**Final Document**

Members: Steve Noel

Mentor: Carlos Ocampo

Instructor: Masoud Sadjadi

April 30, 15

**COPYRIGHTS AND TRADEMARK NOTICES**

This is the work of Steve Noel, unless specified otherwise.

**Executive Summary**

This is the Final Document for the Social Wall application project done for Picture Marketing. This document gives an introduction to the project, showcases the feasibility study done for this application, highlights the project plan created for completion of this app, determines the software requirements for the app, details on the overall design of the app, and shows how we validated our app. In general, it gives a thorough and detailed report on the progress from start to finish.

**Table of Contents**

|  |  |
| --- | --- |
| **Title** | **Page Number(s)** |
| 1. Introduction | 6-7 |
| 1.1. Problem Definition | 6 |
| 1.2. Background | 6 |
| 1.3. Definitions | 6-7 |
| 1.4. Overview of Document | 7 |
| 2. Feasibility of Study | 8-10 |
| 2.1. Description of Current System | 8 |
| 2.2. Purpose of New System | 8-9 |
| 2.3. Description of Alternatives | 9 |
| 2.4. Recommendations | 10 |
| 3. Project Plan | 11-13 |
| 3.1. Project Organization | 11 |
| 3.2. Identifications of Tasks and Deliverables | 12 |
| 3.3. Cost of the Project | 13 |
| 4. System Requirements | 14-26 |
| 4.1. Functional and Nonfunctional Requirements | 14 |
| 4.2. Analysis of System Requirements | 18-26 |
| 5. System Design | 27-31 |
| 5.1. Overview | 27 |
| 5.2. Subsystem Decomposition | 26-30 |
| 5.3. Hardware and Software Mapping | 30 |
| 5.4. Security/Privacy | 30-31 |
| 6. Detailed Design | 32-44 |
| 6.1. Overview | 32 |
| 6.2. Static Model | 33-36 |
| 6.3. Dynamic Model | 37-42 |
| 6.4. Code Specification | 42-44 |
| 7. System Validation | 45-57 |
| 7.1. Subsystem Tests | 45 |
| 7.2. System Tests | 46-56 |
| 7.3. Evaluation Tests | 56-57 |
| 8. Glossary | 58 |
| 9. Appendix | 59-91 |
| 9.1. Appendix A - Project Schedule | 59 |
| 9.2. Appendix B - All Use Cases | 59-67 |
| 9.3. Appendix C - UI Designs | 68 |
| 9.4. Appendix D - Analysis Models | 69-73 |
| 9.5. Appendix E - Design Models | 74-78 |
| 9.6. Appendix F - Documented class interfaces | 78-86 |
| 9.7. Appendix G - Documented code for test driver | 86-88 |
| 9.8. Appendix H - Diary of Meetings | 88-92 |
| 10. References | 93 |

**1. Introduction**

This chapter deals with introducing our project and defining our purpose, scope, terms, and acronyms. Also this chapter discusses the background of the project. It also deals with describing how the rest of this document is organized and structured.

**1.1 Problem Definition**

The Social Wall system aims to solve the inability of our users to easily and efficiently display their online photo albums in a slideshow format on any large screen they desire without the hassle of a complicated setup process. The system should also allow users to seamlessly customize their slideshows on the fly without hindering user experience. With todays highly accessible software and hardware, the Social Wall project combines the latest innovations in mobile and web technologies to help users free and view their media wherever they want.

**1.2 Background**

Our Sponsor for this mobile application Picture Marketing Inc. specializes in creating applications such as FotoZap a social marketing platform that allows companies the ability to promote their brand through customized images and videos. The FotoZap application allows companies to take customer photos on their mobile devices, brand them with their custom logos and send them to their clients through social media, email and other forms thus promoting the companies brand. The, Social Wall we will help them in aggregating images from FotoZap and repurpose them for event slideshows, website galleries and other events so that Picture Marketing has another form of delivering the images to their clients.

**1.3 Definitions, Acronyms, and Abbreviations**

PM - Picture Marketing – Social Wall Sponsor

SW - Social Wall

FotoZap – Picture Marketing app for taking branded photos

Cordova – Native App Platform

h/w - Hardware

s/w - Software

OS - Operating System

App - application

ConnectSdk – Framework used to connect to Chrome cast

CC - Chrome Cast device

AJS - AngularJS MVC JavaScript Framework

JS - Javascript language

CSS - Cascading Style Sheets language

Sencha Touch – Mobile Development Framework

**1.4 Overview of document**

This document has 10 chapters. Each chapter explains a particular aspect of the system. Chapter 2 discusses our Feasibility Study and chapter 3 the project plan for the Social Wall system. Chapter 4 of the document outlines the requirements the system should have. Chapter 5 details the design of the system in general. Chapter 6 outlines the detailed the design of the system or the classes and objects of the system and how they interact. Chapter 7 explains the testing done on the system to make sure the system is bug free. Chapter 8 is the Glossary of the document and shows all the terms used throughout the document. Chapter 9 encompasses the Appendices of the document and Chapter 10 the references.

**2. Feasibility of Study**

**2.1 Description of Current System**

Currently, there are two major ways of displaying an image slideshow on a large screen: the local solution, which uses applications such as Microsoft PowerPoint to display a image slideshow from a computer and the web solution which uses a web application such as Tintup to show the slideshow using a web service. The local approach requires the user to download the images on their computer and connect to the screen or connect a flash drive device to the screen, which requires some initial setup and limits the control of the slideshow. This approach forces companies to dedicate time and resources for the creation of an appealing final product.

The second option using web applications such as Tintup or Postano which offer similar functionalities as the Social Wall System however, these, services while useful, can be rather costly, and can cost upwards of $1,000 a month (or more). This option may be viable for some companies, however it may not be a viable solution for smaller companies and some individuals.

Moreover, there are mobile applications available that can accomplish slideshow streaming to screens such as AllCast or Photocast. These are open source solutions that offer similar features however they are limited in the sources that can be used for the slideshows, they do not offer FotoZap users to view their pictures and their limited in the actions that can be performed on the image slideshow to dynamically customize it. Finally, the Social Wall system version one used a web application instead of a mobile application, which offers very similar functionalities that the mobile system provides but the mobile system is more intuitive and easier to use since most people have mobile devices and are used to using mobile applications.

**2.2 Purpose of New System**

The purpose of the new system is to allow for the automation of an image slideshow through a user-friendlier and completely dynamic system. It involves the creation of a mobile application that can connect to a Chrome cast device and display the images on any screen. The new system should also integrate seamlessly in the Picture Marketing suite of applications giving them yet another delivery method for their client’s media. The expensive and unintuitive alternatives discourage individuals from using this service, and force companies, both large and small, to spare hundreds and even thousands of dollars for a service they may only use a handful of times. The new system will eliminate this by making the mobile application open-source and reinventing the user interface of the Social Wall so that the user can easily use the application. Besides the u

**2.2. Description of alternative solutions considered**

Many solutions were considered for this project. One of them was a native desktop app for Windows machines. In this alternative, the consumer would enter in his/her FotoZap credentials; the app would store it, and generate a slideshow in the application itself. Also the application would have features to customize he slideshow in any way. The application would encompass all of the features of the Social Wall in a single application. This desktop app can be developed using Microsoft Visual Studio using C# and .the NET framework.

Another possible solution is a web application, which can be utilized by any OS and mobile OS through their respective web browsers, giving s/w the largest possible user base. The application will allow users to login and view heir images in the browser this alternative also provides the option to cast to Chrome Cast, which allows the user another simpler way to display their slideshow. The web app would be developed in Javascript, CSS, and HTML, using the AngularJS framework.

A third solution is to have a native app for iOS, Android, and/or Windows Phone using the Cordova framework. This provides an even simpler alternative for consumers to generate their desired slideshow through a simple mobile interface. It would also be an inexpensive alternative for the users, since the app would be open source and could be downloaded from any mobile device app store. A Cordova app would cast the image slideshow to a Chrome Cast, which would make presenting the slideshow even easier. This solution can be developed like a web app using (JS, HTML, CSS, and AJS) and converted into a mobile app with the aid of s/w like Cordova.

**2.3. Recommendations**

For the Social Wall project, we recommended making a mobile IOS application that retrieves photo albums from the users FotoZap account, and displays it on a slideshow engine running on a Chrome cast device. Most users have a mobile device and are accustomed to using applications on the device. Creating a mobile application would make the project more user-friendly and convenient as well. Other types of applications such as web applications or desktop applications require a computer to view and run correctly this hinders the convenience of the application because you need to carry a heavy laptop or use a desktop computer but with mobile apps you simply need your mobile device which is easy to carry and highly accessible. The Cordova framework will allow for the development of the application as a web application and then port it to the mobile device allowing for a cross-platform application that is easy to implement and maintain.

**3. Project Plan**

**3.1. Project Organization**

Team Member Roles:

Steve Noel – Scrum Master, Document Editor, UML Diagram Creator, Interface designer, and lead Developer.

Carlos Ocampo- Mentor

**3.1.2. Hardware and Software Resources**

Hardware:

* Mac Computer (OS X or higher)
* Chrome Cast device
* Television with HDMI port

Software:

* StarUML (version 2.0.0)
* Mingle (Project Management Tool)
* Google Chrome Cast IOS Framework
* Google Drive
* GitHub (version 2.0 for Mac)
* Apache Server
* Xcode IOS Development Environment
* Cordova Native App Framework
* Sencha Touch Mobile App Framework
* Sublime Text 2 (Text Editor)
* AngularJS MVC Javascript Front-End framework
* ConnectSdk Framework

**3.2. Identification of Tasks, Milestones, and Deliverables**

**Tasks:**

* Build and run IOS project on Device
* Run Chrome cast receiver application on Chrome cast
* Implement ConnectSdk bridge on IOS and receiver app
* Implement image Slideshow on receiver app
* Develop Connect and Disconnect device logic
* Handle Message from sender application to receiver application

**Milestones:**

* Create Hybrid IOS Application
* Develop Slideshow Engine
* Connect/Disconnect from Chrome Cast device
* Launch Chrome cast receiver app on Chrome cast
* Send Messages to Chrome cast receiver app
* Rejoin receiver App from Mobile Application

**Deliverables:**

* Feasibility and Product Plan Document
* Design Document
* Software Requirement Document
* User Manual
* Installation Guide
* Social Wall Source Code
* Project Poster
* Social Wall Videos

**3.3. Cost of the Project**

For this project, Steve Noel has decided to forgo payment for his services. As a result, there is no cost calculation in terms of effort and labor in this section. There is, however, a cost for the h/w and s/w resources needed for this project. These costs are covered in detail below:

|  |  |
| --- | --- |
| Resource | Price (USD) |
| Mac Laptop running OS X or better, 4 GB RAM, Intel i5 processor or better | $1300 |
| HDTV with HDMI port | $107.50 |
| Chrome Cast device | $34.99 |
| iPhone 5s | $600 |
| Total (taxes and fees included): | $3449.99 |

**4. System Requirements**

In this chapter, the proposed system requirements are presented and discussed in detail. The systems functional requirements along with their non-functional requirements are explained. These are the functionalities and constraints that the system should have in order to be viable solution to my product owner. The system being proposed is a mobile application that allows users to display their FotoZap account images on any screen using a Chrome cast device. The system will consist of the sender application and the receiver application.

**4.1 Functional and Nonfunctional Requirements**

1) The system shall allow the user to login to the system.

Non-Functional Requirements:

Usability- No previous training time needed.

Readability

* 1. Mean time to Failure – 5% failures for every twenty-four hours of operation is acceptable.
  2. Availability – Down time for Login Back-up 15 minutes in a 24-hour period.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 100 request in 1 minute.

Supportability-

a) System should be supported on IOS mobile devices.

2) The system shall allow the user to logout from the system.

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 3% failures for every twenty-four hours of operation is acceptable.
  2. Availability – Down time for 10 minutes in a 24-hour period.

Performance

1. Request should be sent within 5 secs.
2. System should be able to handle 1000 request in 1 minute.

Supportability-

a) System should be supported on IOS mobile device.

3) The system shall allow the user to Connect to a Chrome cast device.

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 10% failures for every twenty-four hours of operation is acceptable.
  2. Availability – Down time for 15 minutes in a 24-hour period.

Performance

1. Request should be sent within 5 secs.
2. System should be able to handle 100 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile device.

4) The system shall allow the user to Disconnect from a Chrome cast device.

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
  2. Availability – Down time for 15 minutes in a 24 hour period.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 100 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile device.

5) The system shall allow the user to select a campaign.

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
  2. Availability – Down time for 15 minutes in a 24 hour period.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 100 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile devices.

6) The system shall allow the user to switch a campaign.

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 5% failures for every twenty-four hours of operation is acceptable.
  2. Availability – Down time for 15 minutes in a 24 hour period.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 100 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile devices.

7) The system shall allow the user to pause play the image slideshow

Non-Functional Requirements:

Usability- No previous Training time needed.

Readability

* 1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
  2. Availability – Down time for 10 minutes in a 24 hour period.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 100 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile devices.

8) The system shall allow the user to rejoin a running Web App

Non-Functional Requirements:

Usability- No previous training time needed. Simple interface.

Readability

* 1. Mean time to Failure – 5% failures for every twenty-four hours of operation is acceptable.
  2. Availability – Down time for 15 minutes in a 24 hour period is acceptable.

Performance

1. Request should be sent and received within 5 secs.
2. System should be able to handle 1000 requests in 1 minute.

Supportability-

a) System should be supported on IOS mobile devices.

**4.2 Analysis of System Requirements**

**4.2.1. Scenarios**

Scenario 1) Login

Purpose: Scenario that describes the use of the login system of IOS app by a user.

Individual: User registered on the system.

Equipment: IOS mobile device running IOS 6.0 and above

Scenario:

1) User opens application on mobile device.

2) System displays login form.

3) User enters their username and password into the form.

4)User presses login button

5) System authenticates user and displays the users campaigns selecting page.

6) As an alternative the user can submit incorrect credentials then the system notifies the user that the creadentials submitted where incorrenct and allows the user to try again.

Scenario 2) Display Campaign Image Slideshow on Screen

Purpose: Scenario that describes the use of the display images system of IOS app by a user.

Individual: User registered on the system.

Equipment: IOS mobile device running IOS 6.0 and above

Scenario:

1) User opens application on mobile device.

2) System displays login form.

3) User enters his username and password into the form.

4) User presses login button

5) System authenticates user and displays the users campaign selecting page.

5)User selects the connect to Chrome cast button

6)System connects to Chrome cast device

7) The user selectst a campaign to view.

8) The user selects the show on Chrome cast button

9) System shows the seleced campaign on the Chrome cast screen.

Scenario 3) Logout

Purpose: Scenario that describes the use of the logout system of IOS app by a user.

Individual: User registered on the system.

Equipment: IOS mobile device running IOS 6.0 and above

Scenario:

1) User opens application on mobile device.

2) System displays login form.

3) User enters his username and password into the form.

4) User presses login button.

5) System authenticates user and displays the users campaign selecting page.

6) The user selects the logout button.

7) System destroys current view and redirects user to login screen.

Scenario 4) Connect t to Chrome cast device

Purpose: Scenario that describes the use of the connect to Chrome cast feature of the mobile app.

Individual: User registered on the system.

Equipment: IOS mobile device running IOS 6.0 and above

Scenario:

1) User opens application on mobile device.

2) System displays login form.

3) User enters his username and password into the form and presses login button.

4) System authenticates user and displays the users campaign selecting page.

5) The user selects the connect to Chrome cast button

6) System returns a confirmation whether the system was connected to the Chromecast.

Scenario 5) Disconnect t to Chrome cast device

Purpose: Scenario that describes the use of the Disconnect to Chrome cast feature of the mobile app.

Individual: User registered on the system.

Equipment: IOS mobile device running IOS 6.0 and above

Scenario:

1) User opens application on mobile device.

2) System displays login form.

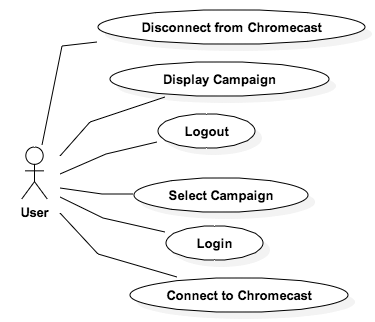
3) User enters his username and password into the form.

4) System authenticates user and displays the users campaign selecting page.

5) The user selects the disconnect to Chrome cast button

6) System returns a confirmation whether the system was disconnected from the Chromecast.

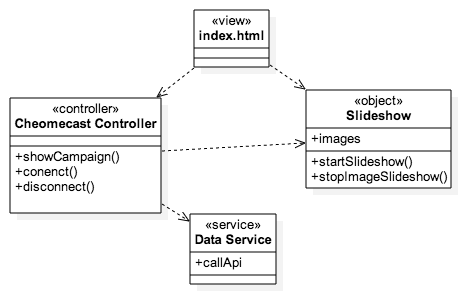
**4.2.2. Use case model**

****

**4.2.3. Static model**



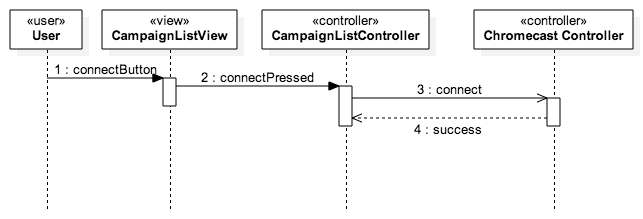
Sender Application Class Diagram



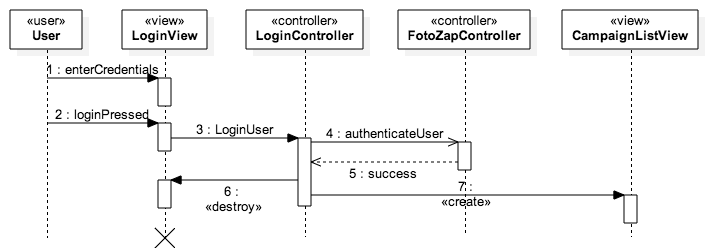
Custom Chrome cast receiver application Class Diagram.

**4.2.4. Dynamic model**

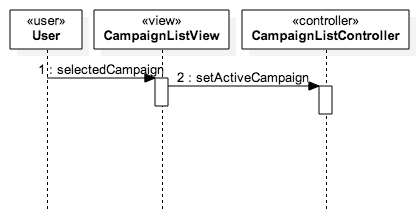
**Connect to Chrome cast Sequence Diagram**



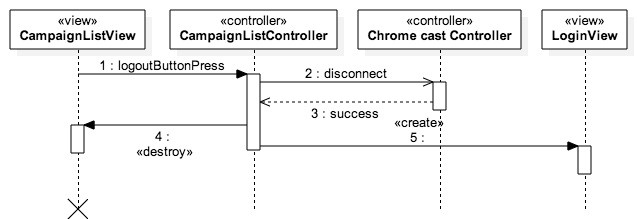
**Login Sequence Diagram**



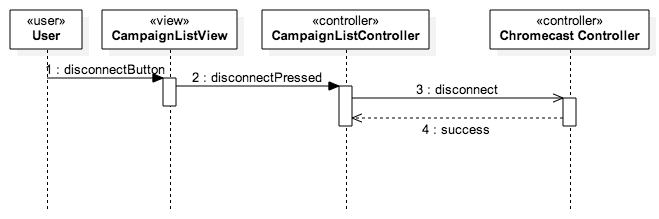
**Select Campaign Sequence Diagram**



**Logout Sequence Diagram**



**Disconnect from Chrome cast Sequence Diagram**



**Show Campaign on Chrome cast**

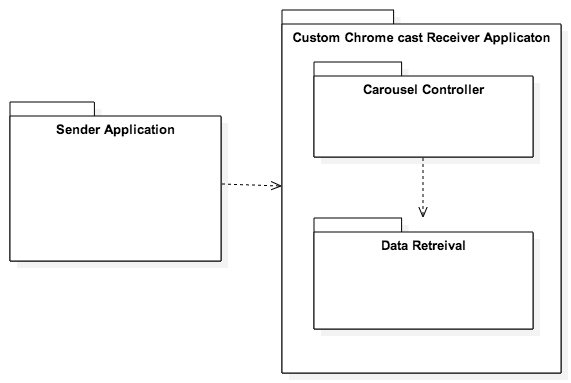


**5. System Design**

A piece of software’s architecture is a description of its overall structure. The Social Wall utilizes the MVC (Model-View-Controller) and the Server-Client architectural patterns. An outline of the Social Wall System design will be described in this chapter. As part of this system, 8 different use cases were implemented. The user can login to the system, connect to a Chrome cast device, disconnect from a Chrome cast device, select a campaign to view, switch the campaign, pause or play the image slideshow being displayed and rejoin the web application if disconnected.

**5.1. Overview** The high level architectural patterns used for the project are the Model-View-Controller (MVC) and Server-Client patterns. The MVC architectural pattern divides the objects of the system into three main groups, each responsible for a specific role in the system. The three parts are the Models, Views, and Controllers, respectively. The model is responsible for managing the data of the application. The view is ultimately the user interfaces that represent the visual aspect of system. The controller of the system acts as the bridge between the models and views, it is in charge of responding to user action and input, and is in control of what is shown in the view part of the application. Hence, the controller receives the data, validates the data, and modifies the model based on occurrences in the view. This architecture was chosen because it makes the updating of models when the views change much easier since the views are bound to the controllers so anytime the views change the controllers are immediately notified no listeners have to be set on the views. Also, the code becomes more flexible since the views are only used to represent data they should not have the logic to retrieve the data or what its model looks like.

The Server-Client pattern differentiates the system into resource providers or servers and service requesters called clients. Both actors usually communicate through a computer network on separate hardware. The usual example of the Server-Client architecture is a web application. This architecture was chosen because of its accessibility any client can connect to server and also allows for Backup and Recovery since all the data is stored on the Server.The system as a whole can be described as a client server architecture, where the client is a Chrome cast device, which makes requests to a webserver to retrieve a Chrome cast receiver web application that can communicate with an IOS mobile application. The Social Wall has four main subsystems, the chrome cast sender system, the custom Chrome cast receiver system, the Carousel system and the Data Retrieval system. For the creation of the chrome cast receiver subsystem, the AngularJS framework was utilized. For the sender subsystem the Cordova and Sencha Touch 2 frameworks were used.



**Package Diagram of the Social Wall System.**

**5.2 Subsystem Decomposition**

The system is composed of four subsystems:

* Sender Application Subsystem

Related Use Cases:

* Login
* Logout
* Connect to Chrome cast
* Disconnect to Chrome cast
* Rejoin Web App
* Pause or play slideshow
* Select Campaign
* Switch Campaign
  + Sender application subsystem is an IOS mobile application, which plays the role of interacting with the user and allowing them to interact with the entire system. The sender application retrieves the input from the user such username, password and campaign id. The sender application also allows the user to connect to a Chrome cast device connected on the same network, Launch the receiver Application on the Chrome cast device and communicate with it.
* Data Retrieval Subsystem

Related Use Cases:

* Pause or play slideshow
* Select Campaign
* Switch Campaign
* The data retrieval subsystem is in charge of receiving the actual messages from the sender applications, parsing this message, retrieving the necessary data and making it available for the Carousel subsystem to get it and display it on the screen.
* Carousel Subsystem

Related Use Cases:

* Pause or play slideshow
* Select Campaign
* Switch Campaign
  + The carousel subsystem is in charge of the image slideshow, which is the main content that will be displayed on the screen. This subsystem handles the loading of the images, the creation of the canvas element and the rotation of the images in the slideshow. In addition, it works alongside the Chrome cast receiver subsystem to provide the users with the content that they requested using the sender application.
* Chrome cast Receiver Subsystem

Related Use Cases:

* Pause or play slideshow
* Select Campaign
* Switch Campaign
  + The receiver subsystem is a custom chrome cast receiver application in charge of handling the sender subsystems requests and fetching the necessary data and displaying it onto the screen. It is made up of the carousel subsystem and the data retrieval subsystem.

**5.3. Hardware and Software Mapping**

* Users Hardware/Software
  + Hardware: Apple Computer, IOS Device, Google Chrome cast, HDMI capable device.
    - Apple Computer and IOS device allow users to run the sender application that allow interaction with other parts of system. This involves the sender subsystem.
    - Chrome cast - allows hdmi capable devices of running custom receiver applications, and allowing interactions through devices such as PCs and mobile devices. This is where the Chrome cast receiver application run on as well as the Data Retrieval subsystem and the carousel subsystem.
    - HDMI capable device - is necessary in order for the Chrome cast functionality, plays the role as the host device displaying the custom receiver application.
  + Software:
    - IOS Operating System - Allows for the mobile sender application to run and allows user interaction with custom receiver application and chrome cast.
    - Chrome cast stripped Chrome Browser – Allows for the custom receiver application to run and shown on the desired screen.

**5.4. Persistent Data Management**

The Social Wall System does not have a database, which stores large amounts of structured data persistently. However, if the user selects the “Remember Me” feature, the system will cache the username and password of the user if the user selected the “Remember Me” option when login in. Persistent data is not necessary; as the system requires a network connection to function thus any data can be retrieved on the network.

**5.5. Security/Privacy**

As the system stands, security and privacy is not a major concern. The data transmitted by the system is not sensitive enough to warrant high security standards.

That being said, the system does have a user authentication feature that uses the HTTP Basic protocol. This protocol encodes the users username and password information before it is sent across the network using Base64 encoding. In addition, the system uses the HTTPS encrypted protocol for its network communications. It uses the latest TLS 1.2 (Transfer Layer Security) protocol for its communication security.

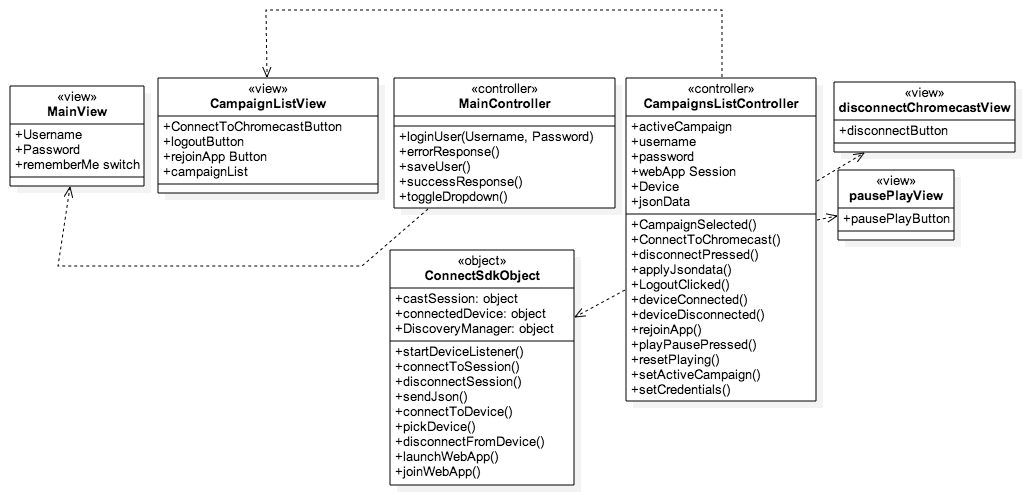
**6. Detailed Design**

This chapter deals with the detailed design of the system in terms of how the various classes interact with each other, the subsystems that comprise the Social Wall, the control flow in the system. It includes the static and dynamic models that describe the system and the design patterns that were used in the system, as well as an explanation of he code and how the classes are implemented.

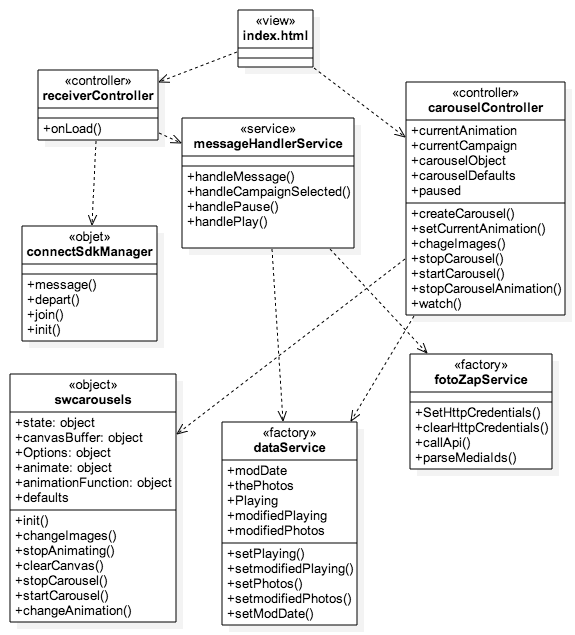
**6.1. Overview**

The Social Wall system is comprised of four subsystems. A Data Retrieval subsystem that retrieves messages from the sender subsystem, and fetches necessary data from the Picture Marketing servers, a Sender subsystem that sends the information the user inputs and is the main user interface for the Social Wall, a Receiver system that encompasses the web application running on the Chrome cast device and a Carousel subsystem that controls the state of the image slideshow.

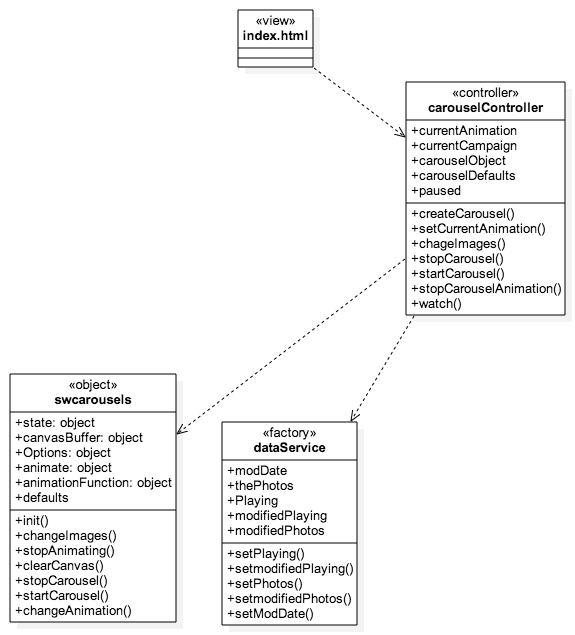
The design patterns used in the Social Wall system are the singleton, mediator, state and Iterator design patterns. The Singleton pattern was used because it centralizes the code and provides a single interface to access an object thereby reducing the complexity and increasing the simplicity and readability of the code. The mediator pattern was chosen to simplify the communication among objects in the system. The state pattern was used to avoid large conditional statements at runtime.Finally the iterator pattern was used to make looping through objects easier.

**6.2 Static Model**

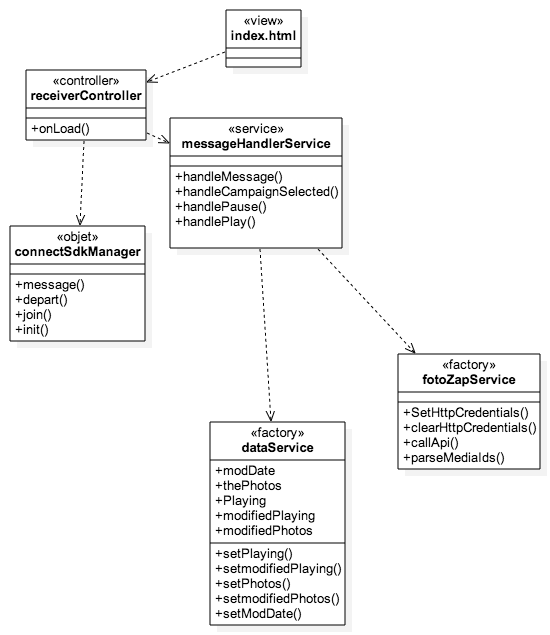
* **Sender Application Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Sencha Touch 2
  + Architecture: MVC & Cordova (App Only)
  + The sender application subsystem is made up of a Cordova IOS application. This system provides the user interface that will allow for users to input information and later retrieve the desired result. This subsystem interacts directly with the Chrome cast receiver subsystem and communicates messages along to it. These messages contain the user input data that will later be used in the other subsystems. This subsystem requires the ConnectSdk Object in order to connect to the Chrome cast device. Each view in the system has its own dedicated controller which handles the logic of that view. For Example, the login or main controller is in charge of logging in the user, the CampaignList controller is in charge of all the other main functionalities like sending messages, connecting to the chrome cast etc.



* **Chrome cast Receiver Application Subsystem** 
  + HTML5—Javascript -CSS
  + Framework: AngularJS
  + Architecture: MVC
  + This system is run on the Chrome cast device. It interacts directly with the sender subsystem and can also communicate messages along to the sender application as well. This system is made up of the Carousel and Data Retrieval systems allowing it to fetch data and display it in a slideshow format. The system depends on the ConnectSdk Manager object which allows for applications to connect to the application and receive messages from connected clients.The main objects are the messageHandler object which receives and parses the messages from the sender application.The FotoZap Service which is responsible for retrieving data from Picture Marketing servers, the data Service which is used as a mediator between the message Handler and the carousel controller, and the carousel cotntroller which handles the state of the image slideshow object.

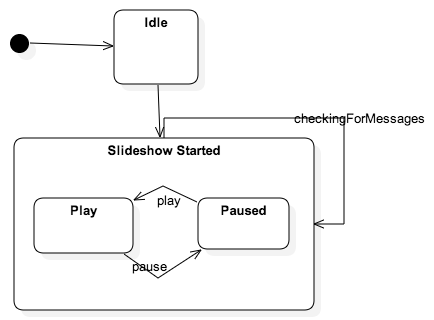


* **Carousel Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Angularjs
  + Architecture: MVC
  + This system is in charge of the entire Image slideshow of the receiver application. It creates the image carousel. It is in charge of pausing and playing the slideshow, displaying the images to the user and switching the images in the slideshow.

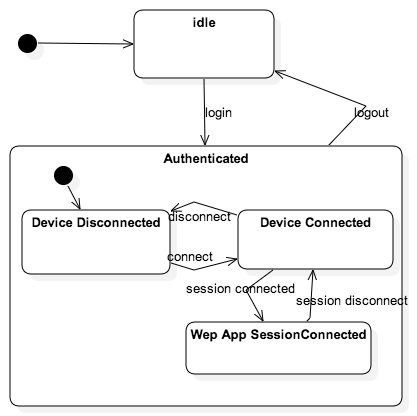


* **Data Retrieval Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Angularjs
  + Architecture: MVC
  + This subsystem is mainly concerned with retrieving and parsing the necessary data for the receiver application. This includes calling external API functionalities, parsing the data and then setting the data in the Data Service object so the carousel subsystem can receive it and display it.

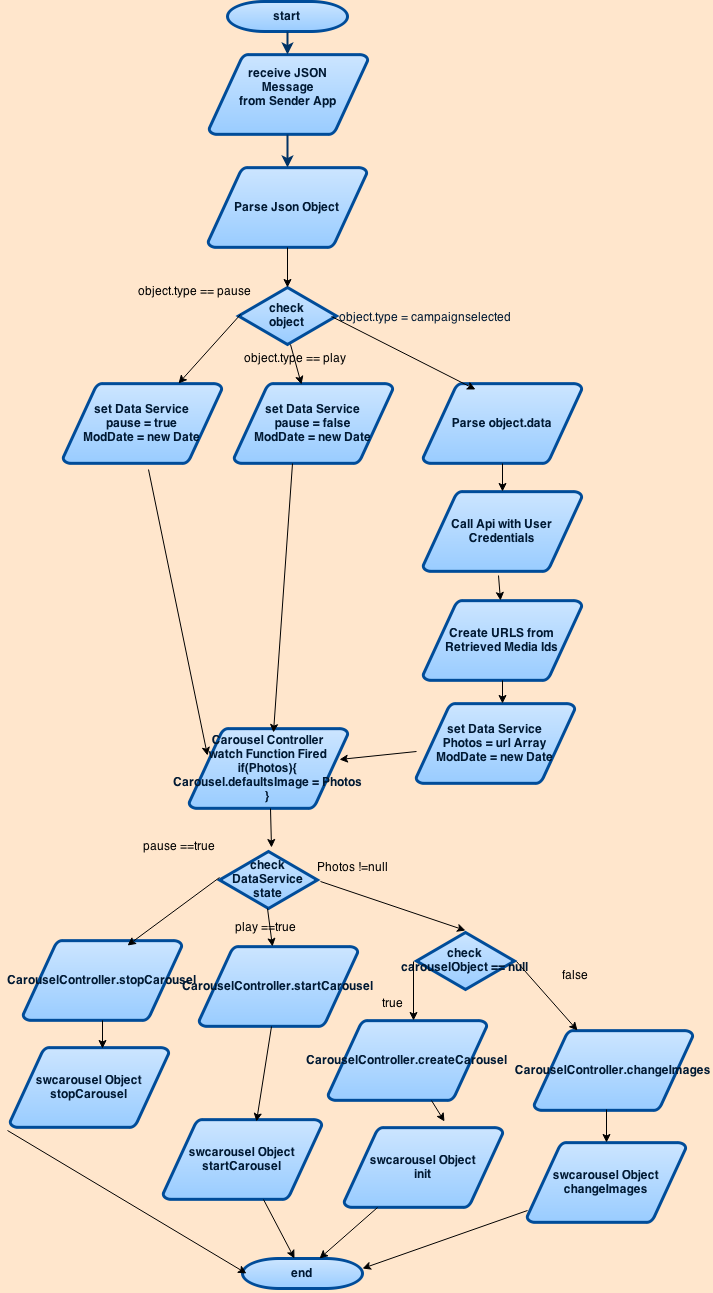
**6.3 Dynamic Model**

****

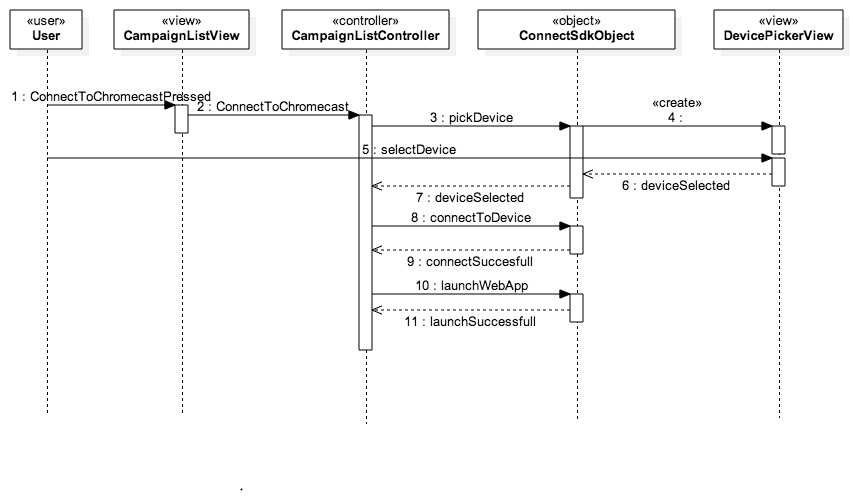
**State Machine Diagram – Carousel Controller.**



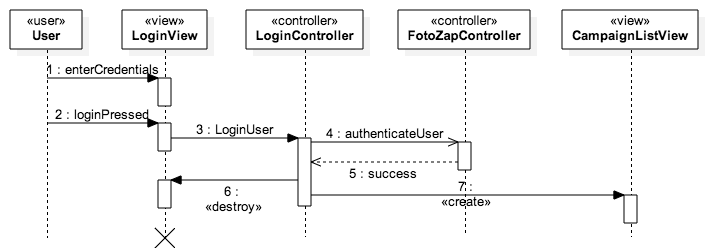
**State Machine Diagram –Campaign List View Controller.**



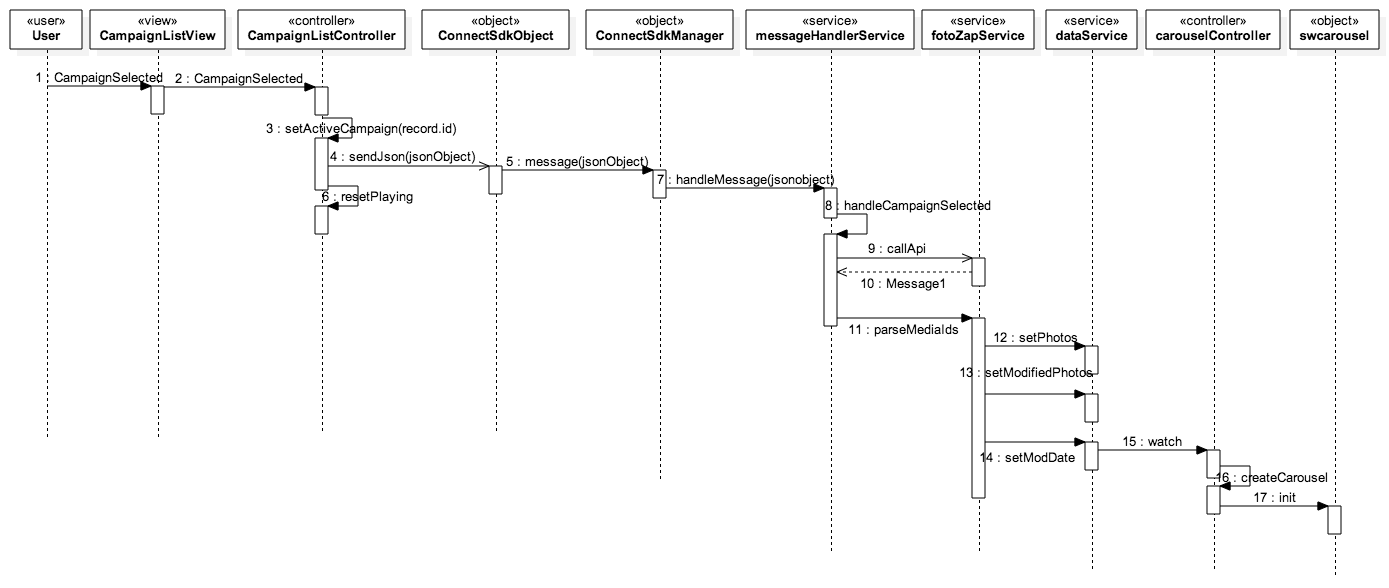
**Main Algorithm of System Flow Chart**



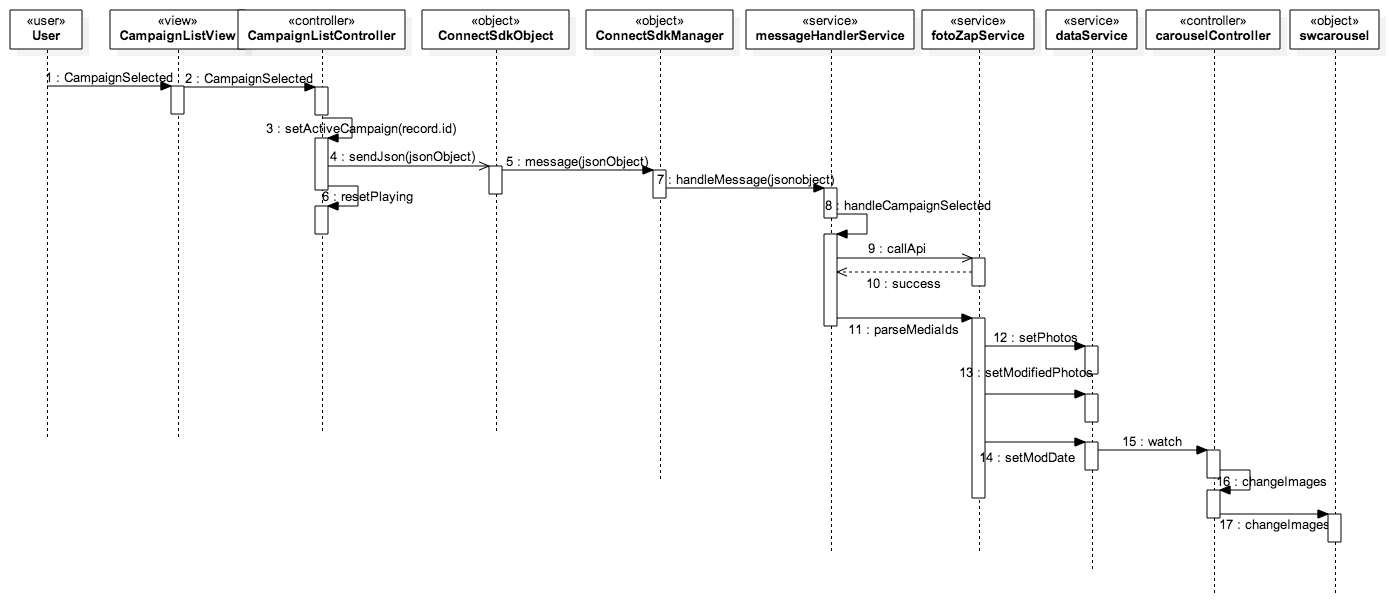
**Sequence Diagram of S-WALL/2-003/Connect to Chrome cast**



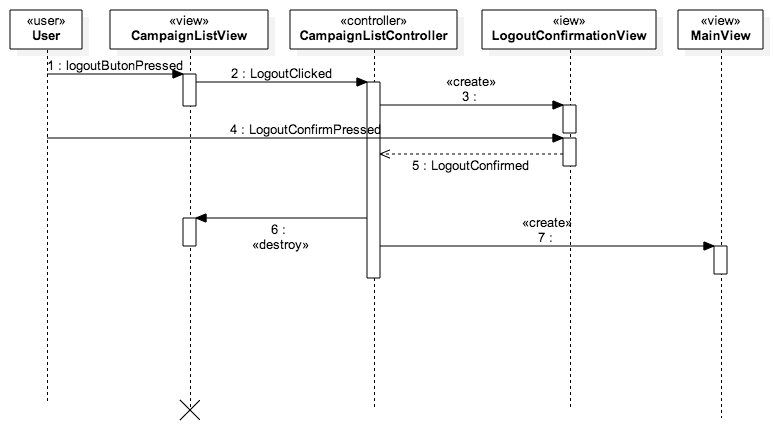
**Sequence Diagram of S-WALL/2-001/Login**



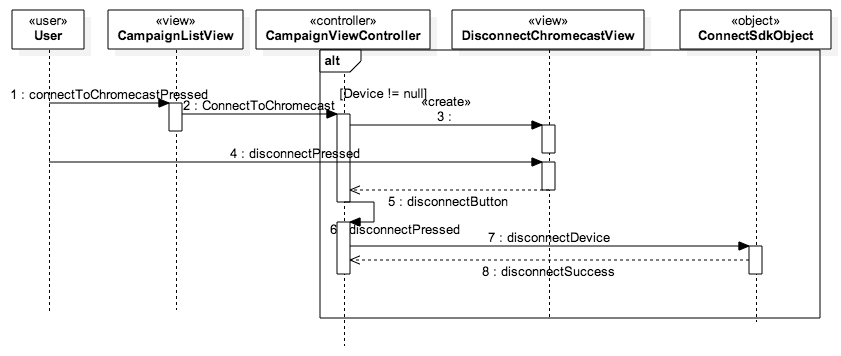
**Sequence Diagram of S-WALL/2-005/Select Campaign**



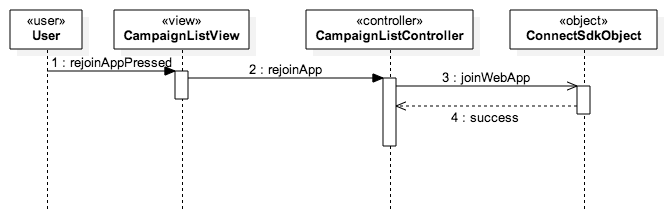
**Sequence Diagram of S-WALL/2-006/Switch Campaign**



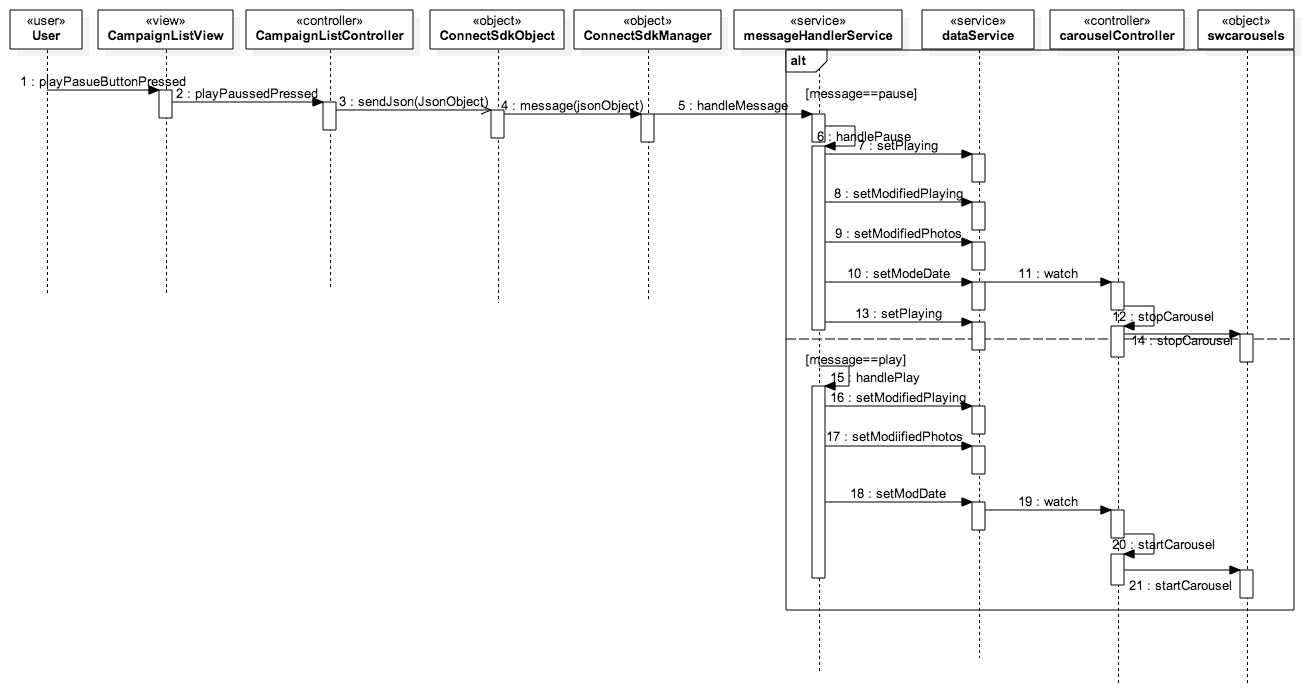
**Sequence Diagram of S-WALL/2-002/Logout**



**Sequence Diagram of S-WALL/2-004/Disconnect from Chrome cast**



**Sequence Diagram of S-WALL/2-008/Rejoin Web App**



**Sequence Diagram of S-WALL/2-007/ Play Pause Slideshow**

**6.4 Code Specification**

The two main controllers in the Social Wall system are the campaign List view controller in the sender application subsystem and the carousel controller in the Chrome cast receiver subsystem. The campaign list controller is in charge of the campaign list page of the sender application, which is the most important view in the sender application it is the one that sends messages to the receiver and connects to the Chrome cast device. The carousel controller oversees the image slideshow in the receiver application. Below is a detailed description of the two controllers:

**Campaign List View Controller (Attributes)**

* **WebAppId** 
  + **Id of Google Chrome cast receiver application**
* **Playing** 
  + **Boolean variable for pause and play functionality**
* **appSession** 
  + **Communication channel between apps.**
* **Jsondata** 
  + **Campaign list data**
* **Username** 
  + **Successful username of user**
* **Password** 
  + **Successful password of user**
* **Device** 
  + **Chrome cast device reference**
* **activeCampaign** 
  + **Active campaign selected by user**

**Campaign List View Controller (Methods)**

* **CampaignSelected(record)** 
  + Implements the select Campaign and switch Campaign functionality.
* **ConnectToChromecast**
  + Implements the connect to a Chrome cast device as well as begins the disconnect Chrome cast feature.
* **disconnectPressed**
  + Direct trigger for the disconnect Chrome cast button.
* **applyJsonData**
  + Sets campaign list with data.
* **LogoutClicked**
  + The Trigger for the Logout User Story.
* **deviceConnected**
  + Callback method when the chrome cast device is connected. The method sets some state variables for the sender application.
* **deviceDisconnected**
  + Callback method when the chrome cast device is disconnected. It sets some state variables for the sender application.

**Carousel Controller(attributes)**

* **currentAnimation**
  + The animation currently being used by the image slideshow.
* **currentCampaign**
  + The campaign or album being shown by the image slideshow.
* **carouselDefaults**
  + The options such as animation, transition speed, images etc. that the image carousel uses at default.
* **carouselObject**
  + The carousel object in charge of the image slideshow.
* **Paused**
  + Boolean variable describing whether the carousel is paused or is playing.

**Carousel Controller (methods)**

* **stopCarousel**
  + Pauses the image slideshow on receiver application.
* **startCarousel**
  + Resumes the image slideshow on receiver application.
* **chageImages** 
  + Modifies the images in the image slideshow.
* **createCarousel**
  + Instantiates the image slideshow object with the default settings.

**7. System Validation**

In this chapter we discuss our testing methods and evaluate the results of our testing. Section one describes the subsystem testing method that was used during the project. In the section after we provide a description of the system tests through sunny day and rainy day test cases. In final section we evaluate the results of our testing in a tabular form.

**7.1. Subsystem Tests**

For the Individual subsystems we used the Unit Testing method whereby individual segments of source code as well as the subsystems interfaces were tested to make sure that the methods are functioning in the proper way and that the systems are working together correctly. For the Social Wall Project we used two different testing frameworks, Siesta and Jasmine. Jasmine is a behavioral driven development (BDD) testing framework for testing JavaScript code. It does not rely on any browsers or other frameworks it runs wherever JavaScript runs. It was used together with the Karma Test Runner, which allows for the testing JavaScript in the browser. These were used to test the Chrome cast receiver subsystem especially the interaction between the Data Retrieval subsystem and the Carousel Subsystem mocking a message from the sender and following the response of the system.

The other testing framework that was used was the Siesta framework. The Siesta framework is not only a unit-testing tool but also can test the DOM (document object model) and simulate user interactions. This framework was chosen because it is designed for Sencha Touch applications was the framework used to develop the mobile application. The tests where implemented to test the functionalities of the controllers in the sender application.

**7.2. System Tests**

**S-WALL/TC-001/Connect to Chrome cast Sunny Day**

**Tested Use Case:** SW/2-003/Connect to Chrome cast

**Purpose:**

Ensure that the user can connect to the Chrome cast device running.

**Test Setup Environment:**

Test run using Siesta test framework, which ran on a localhost, displayed on Google Chrome browser.

**Test Input(s):**

* User presses the Connect to Chrome cast device button.
* User selects the Chrome cast device to connect to.

**Expected Output(s):**

User is connected to the Chrome cast device and has a reference to the device.

**Actual Output(s):**

The device is connected to the Chrome cast device and has a reference to the. connected device. **(TEST PASSED)**

**S-WALL/TC-002/ Connect to Chrome cast Rainy Day**

**Tested Use Case:** SW/2-003/Connect to Chrome cast

**Purpose:**

Ensure that a user cannot connect to a device that is not a Chromecast.

**Test Setup Environment:**

Test run on the IOS simulator, with a APPLE TV device as the device to connect to.

**Test Input(s):**

* User presses the Connect to Chrome cast button.
* User selects a Device that is not a Chrome cast device.

**Expected Output(s):**

An error message saying that the device selected by the user is not supported by the system.

**Actual Output(s):**

App displays the following error: Error:Invalid Device. **(TEST PASSED)**

**S-WALL/TC-003/ Connect to Chrome cast Sunny Day**

**Tested Use Case:** SW/2-003/Connect to Chrome cast

**Purpose:**

Ensure that the system shows an error message when the device the user is trying to connect to is offline.

**Test Setup Environment:**

Test run on actual device, with Chrome cast device turned off right before the sender application is to connect to it .

**Test Input(s):**

* User selects the Connect to Chrome cast device.
* User select sthe Chrome cast device they want to connect to.

**Expected Output(s):**

An Error gets thrown saying the device is not available to connect.

**Actual Output(s):**

User is not prompted an error and the sender application is abe to connect to the device. **(TEST FAILED)**

**S-WALL/TC-004/Disconnect from Chrome cast Rainy Day**

**Tested Use Case:** S-WALL/2-004/Disconnect from Chrome cast

**Purpose:**

Ensure that when the user tries to disconnect from the device when the system is already disconnected the system does not cause an error.

**Test Setup Environment:**

Test run on actual device, with Chrome cast device turned off right before the sender application is to disconnect from it .

**Test Input(s):**

* The user selects the Disconnect from Chrome cast button.
* The user selects the Disconnect button.

**Expected Output(s):**

The system does not raise an error message.

**Actual Output(s):**

The system did not raise an error.**(TEST PASSED)**

**S-WALL/TC-005/Disconnect from Chrome cast Sunny Day**

**Tested Use Case:** S-WALL/2-004/Disconnect from Chrome cast

**Purpose:**

Ensure that when the user tries to disconnect from the device when the system is able to disconnect from the device succesfully.

**Test Setup Environment:**

Test run on actual device, with Chrome cast connected to the screen.

**Test Input(s):**

* The user selects the Disconnect from Chrome cast button.

**Expected Output(s):**

The user is disconnected from the device and cannot send messages to it.

**Actual Output(s):**

The device is disconnected from the Device and is not able to connect to it. **(TEST PASSED)**

**S-WALL/TC-006/Disconnect from Chrome cast Sunny Day**

**Tested Use Case:** S-WALL/2-004/Disconnect from Chrome cast

**Purpose:**

Ensure that the sender application can disconnect from the device and afterwards reconnect to it.

**Test Setup Environment:**

Test run on actual device, with Chrome cast connected to the screen.

**Test Input(s):**

* User presses the Disconnect from Chrome cast button.
* User presses the Connect to Chrome cast button.

**Expected Output(s):**

The user us able to disconnect t and reconnect to the same Chrome cast device.

**Actual Output(s):**

The user able to reconnect to Chrome cast device after disconnecting.**(TEST PASSED)**

**S-WALL/TC-007/Select Campaign Sunny Day**

**Tested Use Case:** S-WALL/2-005/Select Campaign

**Purpose:**

Ensure that the user can select a Campaign using the sender application and it is displayed on the Chrome cast receiver application.

**Test Setup Environment:**

Test run on Karma Test Runner, which ran on a localhost, displayed on Google Chrome browser. The message was stubbed in order to test the system without actually receiving message.

**Test Input(s):**

* A valid api response is used to test the systems response to Apl calls. Test simulates message by proving a stubbed message object.

**Expected Output(s):**

Message is received, Data Retrieval service shows the state change and the carousel controller shows that the carousel object holds the images from the message.

**Actual Output(s):**

Slideshow display the state changes**(TEST PASSED)**

**S-WALL/TC-008/Select Campaign Rainy Day**

**Tested Use Case:** S-WALL/2-005/Select Campaign

**Purpose:**

Ensure that when the user selects a Campaign using the sender application it is not received when the receiver application is not connected to the sender.

**Test Setup Environment:**

Test setup using the actual mobile device with the Chrome cast device disconnected to screen.

**Test Input(s):**

* User selects a Campaign from the Campaign List.

**Expected Output(s):**

System does not receive the message.

**Actual Output(s):**

System did not receive message.**(TEST PASSED)**

**S-WALL/TC-009/Select Campaign Sunny Day**

**Tested Use Case:** S-WALL/2-005/Select Campaign

**Purpose:**

Ensure that the sender application can send a message when the receiver application is connected to it.

**Test Setup Environment:**

Test setup using the actual mobile device with the Chrome cast device connected to screen.

**Test Input(s):**

* User selects a campaign from the Campaign List.

**Expected Output(s):**

A message should be sent from the sender application

**Actual Output(s):**

Message received by Receiver Application **(TEST PASSED)**

**S-WALL/TC-010/Switch Campaign Sunny Day**

**Tested Use Case:** S-WALL/2-006/Switch Campaign

**Purpose:**

Ensure that the user can switch a Campaign using the sender application and it is displayed on the Chrome cast receiver application.

**Test Setup Environment:**

Test run on Karma Test Runner, which ran on a localhost, displayed on Google Chrome browser. The message was stubbed in order to test the system without actually receiving message

**Test Input(s):**

* User selects A Campaign from the Campaign List Page.

**Expected Output(s):**

Receiver application receives message and changes the images on the image slideshow.

**Actual Output(s):**

Slideshow displays the new images **(TEST PASSED)**

**S-WALL/TC-011/Switch Campaign Sunny Day**

**Tested Use Case:** S-WALL/2-006/Switch Campaign

**Purpose:**

Ensure that the sender application can send a message when the receiver application is connected to it.

**Test Setup Environment:**

Test setup using the actual mobile device with the Chrome cast device connected to screen.

**Test Input(s):**

* User selects a Campaign from the Campaign List

**Expected Output(s):**

The receiver receives a message.

**Actual Output(s):**

Receiver received message **(TEST PASSED)**

**S-WALL/TC-012/Switch Campaign Rainy Day**

**Tested Use Case:** S-WALL/2-006/Switch Campaign

**Purpose:**

Ensure that when the user selects a Campaign using the sender application it is not received when the receiver application is not connected to the sender.

**Test Setup Environment:**

Test setup using the actual mobile device with the Chrome cast device disconnected to screen.

**Test Input(s):**

* User selects a Campaign to view.

**Expected Output(s):**

An error message that tells the user to input a valid transition time before continuing.

**Actual Output(s):**

Reciever application received message **(TEST FAILED).**

**S-WALL/TC-013/Login Sunny Day**

**Tested Use Case:** S-WALL/2-001/Login

**Purpose:**

Ensure that the user can login to the system given correct input.

**Test Setup Environment:**

Test run using the Siesta Testing Framework, running in the Chrome browser

**Test Input(s):**

* User enters credentials (username and password)
* User presses the Login Button

**Expected Output(s):**

User is successfully logged in to the system and redirected to the Campaign List page.

**Actual Output(s):**

User prompted with login failed message .**(TEST FAILED)**

**S-WALL/TC-014/Login Sunny Day**

**Tested Use Case:** S-WALL/2-001/Login

**Purpose:**

Ensure that the user cannot login with incorrect credentials.

**Test Setup Environment:**

Test run using the Siesta Testing Framework, running in the Chrome browser

**Test Input(s):**

* User enters credentials (username and password)
* User presses the Login Button

**Expected Output(s):**

An error message that tells the user login failed.

**Actual Output(s):**

Login failed error message.**(TEST PASSED)**

**S-WALL/TC-015/Login Rainy Day**

**Tested Use Case:** SW/1-008/Generate Slideshow

**Purpose:**

Ensure that the user cannot login given the internet is not connected.

**Test Setup Environment:**

Test run using the Siesta Testing Framework, running in the Chrome browser

**Test Input(s):**

* User enters credentials (username and password)
* User presses the Login Button

**Expected Output(s):**

An error message that tells the user to internet disconnected.

**Actual Output(s):**

Login failed **(TEST PASSED).**

**S-WALL/TC-016/Logout Sunny Day**

**Tested Use Case:** S-WALL/2-002/Logout

**Purpose:**

Ensure that the user can logout to the system given correct input.

.

**Test Setup Environment:**

Test run using the Siesta Testing Framework, running in the Chrome browser

**Test Input(s):**

* User seletcts the logout button.

**Expected Output(s):**

User is successfully logged out and redirected to login page.

**Actual Output(s):**

User successfully logged out **(TEST PASSED)**

**S-WALL/TC-017/Logout Rainy Day**

**Tested Use Case:** SW/1-008/Generate Slideshow

**Purpose:**

Ensure that the user can logout when internet is down.

**Test Setup Environment:**

Test run using the Siesta Testing Framework, running in the Chrome browser

**Test Input(s):**

* User selects the logout button.

**Expected Output(s):**

User can logout successfully.

**Actual Output(s):**

User logged out successfully. **(TEST PASSED)**

**S-WALL/TC-018/rejoin Web App Sunny Day**

**Tested Use Case:** S-WALL/2-008/Rejoin Web App

**Purpose:**

Ensure that the user can rejoin running receiver web application.

**Test Setup Environment:**

Set up on actual IOS device and Chrome cast device connected.

**Test Input(s):**

* User selects the rejoin web App button.

**Expected Output(s):**

User us reconnected and the system establishes new session with receiver app.

**Actual Output(s):**

System successfully rejoined wit the receiver application. **(TEST PASSED)**

**S-WALL/TC-019/rejoin Web App Sunny Day**

**Tested Use Case:** S-WALL/2-008/Rejoin Web App

**Purpose:**

Ensure that the system does not rase error when trying to join a web app that is not running on the system.

**Test Setup Environment:**

Set up on actual IOS device with the chrome cast not running the receiver application.

**Test Input(s):**

* User selects the rejoin Web App Button.

**Expected Output(s):**

The system does not raise error message when rejoining receiver app that is not running.

**Actual Output(s):**

System showed no error message .**(TEST PASSED)**

**S-WALL/TC-020/rejoin Web App Rainy Day**

**Tested Use Case:** S-WALL/2-008/Rejoin Web App

**Purpose:**

Ensure that the system causes error message when the user tries to rejoin Web App that where the Chrome cast device is disconnected.

**Test Setup Environment:**

Set up on IOS device with Chrome cast running but disconnected from sender app.

**Test Input(s):**

* User selects the rejoin Web App button

**Expected Output(s):**

An error message that tells the user cannot rejoin web app.

**Actual Output(s):**

Error message showed user cannot join web app .**(TEST PASSED)**

**S-WALL/TC-021/Pause Play Slideshow Sunny Day**

**Tested Use Case:** S-WALL/2-007/ Play Pause Slideshow

**Purpose:**

Ensure that the slideshow engine gets paused when the user selects the pause play button.

**Test Setup Environment:**

Test run on Karma Test Runner, which ran on a localhost, displayed on Google Chrome browser. The message was stubbed in order to test the system without actually receiving message

**Test Input(s):**

* User presses the pause play button

**Expected Output(s):**

System pauses the slideshow.

**Actual Output(s):**

Slideshow paused .**(TEST PASSED)**

**S-WALL/TC-022/Pause Play Slideshow Sunny Day**

**Tested Use Case:** S-WALL/2-007/ Play Pause Slideshow

**Purpose:**

Ensure that the sender application actually sends the message to the receiver application when connected.

**Test Setup Environment:**

Set up using actual IOS device and Chrome cast device running.

**Test Input(s):**

* User presses the pause play button

**Expected Output(s):**

Receiver application message Handler receives message.

**Actual Output(s):**

Receiver received message**(TEST PASSED)**

**S-WALL/TC-023/Pause Play Slideshow Rainy Day**

**Tested Use Case:** S-WALL/2-007/ Play Pause Slideshow

**Purpose:**

Ensure that the receiver applicaton does not receive the pause message when disconnected.

**Test Setup Environment:**

Set up on IOS device and Chrome cast running on the large screen.

**Test Input(s):**

* User presses the pause play button

**Expected Output(s):**

Receiver application does not receive message.

**Actual Output(s):**

The pause play message ws not received**(TEST PASSED)**

**7.3. Evaluation Tests**

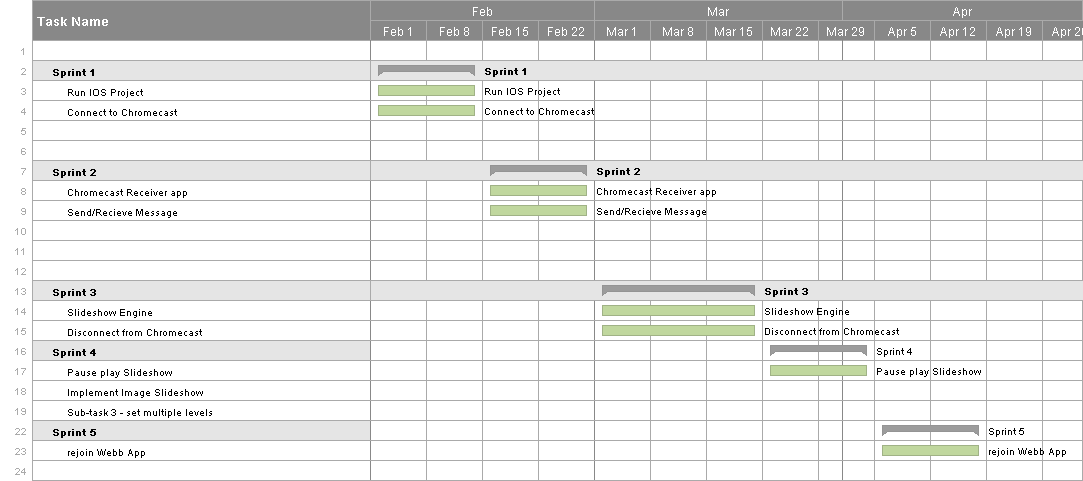
|  |  |  |
| --- | --- | --- |
| Test Case ID | Sunny or Rainy Day | PASS or FAIL |
| **TC-001** | Sunny Day | PASS |
| **TC-002** | Rainy Day | PASS |
| **TC-003** | Sunny Day | PASS |
| **TC-004** | Rainy Day | PASS |
| **TC-005** | Sunny Day | PASS |
| **TC-006** | Sunny Day | PASS |
| **TC-007** | Rainy Day | PASS |
| **TC-008** | Sunny Day | PASS |
| **TC-009** | Sunny Day | PASS |
| **TC-010** | Sunny Day | PASS |
| **TC-011** | Sunny Day | FAIL |
| **TC-012** | Rainy Day | FAIL |
| **TC-013** | Sunny Day | PASS |
| **TC-014** | Sunny Day | PASS |
| **TC-015** | Rainy Day | PASS |
| **TC-016** | Sunny Day | PASS |
| **TC-017** | Rainy Day | PASS |
| **TC-018** | Sunny Day | PASS |
| **TC-019** | Sunny Day | PASS |
| **TC-020** | Rainy Day | PASS |
| **TC-021** | Sunny Day | PASS |
| **TC-022** | Sunny Day | PASS |
| **TC-023** | Rainy Day | PASS |

**8. Glossary**

|  |  |
| --- | --- |
| Term | Definition |
| Slideshow | An array of images that are displayed one at a time. |
| Dynamic | Capable of action or change |
| Static | Stationary, fixed |
| Functional Requirements | Features that are integral to the desired output of the system |
| Non-functional Requirements | Constraints on the system that determine the quality of said system. |

**9. Appendix**

**9.1. Appendix A - Project Schedule (Gantt chart)**

****

**9.2. Appendix B - Use Cases**

**Use Case ID:** S-WALL/2-001/Login

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

* **Description:**

1. The use case begins when the user enters his credentials, username and password for example johndoe and password321.

2. The user then presses the “Login” button.

3. The system shall send the credentials to the server for authentication.

4. The use case ends when the server responds with a success json message and the system destroys the current view and changes to the campaign list view destroying the old view.

**Post conditions:**

1. The user is in the Campaign List View of the system.

**Alternative Courses of Action**: N/A

**Exceptions:**

* The system is unable to send the credentials to the server.
* There is no Internet connection on the device.

**Related Use Cases:**

**Logout**

**Decision Support:**

* **Frequency:** Will be used every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent upon external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** February 6, 2015
* **Last Modified:** April 9, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-002/Logout

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the System.

* **Description:**

1. The use case begins when the user presses the “Logout” button.

2. The system prompts the user whether he wants to continue or not.

3. The user then presses the okay button.

4. The use case ends when the system resets the state variables, closes the current view and navigates to the main or login screen.

**Post conditions:**

1. The user is in the login screen of the system.

**Alternative Courses of Action**:

1.User can press the “NO” button in step 2 and cancel the logout.

**Exceptions:**

**Related Use Cases:**

**Login**

**Decision Support:**

* **Frequency:** Will be used on average 2 times the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Fundamental functionality.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** February 26, 2015
* **Last Modified:** February 26, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-003/Connect to Chrome cast

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the system.

3. There is a Chrome cast device on the same network as the mobile device.

* **Description:**

1. The use case begins when the user presses the “Connect to Chrome cast“ button.

2. The system then shows a list of Chrome cast devices on the network.

3. The user then selects a device from the list of devices.

4. The system then hides the list view and establishes a connection with the selected device.

5. The use case ends when the system launches the custom receiver application on the Chrome cast device and establishes a web App Session with the running application.

**Post conditions:**

1. The user is connected to the Chrome cast device.

2. The user has an active web App Session with the custom receiver application.

3. The custom receiver application is running on the Chrome cast device.

**Alternative Courses of Action**:

1. The user can select the “Cancel” button on step 2 to not connect to a device.

**Exceptions:**

**Related Use Cases:**

**Disconnect from Chrome cast**

**Decision Support:**

* **Frequency:** Will be used every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent on external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** February 5, 2015
* **Last Modified:** February 19, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-004/Disconnect from Chrome cast

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the system.

3. User must be connected to a Chrome cast device.

**Description:**

1. The use case begins when the user presses the “Connect to Chrome cast“ button.

2. The system then displays the disconnect Chrome cast view.

3. The user then presses the “Disconnect” button.

4. The use case ends when the system hides the disconnect Chrome cast view, disconnects from the Chrome cast device and sets some state variables.

**Post conditions:**

1. The user is disconnected from the Chrome cast device.

**Alternative Courses of Action**:

1. The user can press outside the view at step 2 to cancel the use case.

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used on average 3 times the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent on external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** March 17, 2015
* **Last Modified:** April 8, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-005/Select Campaign

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the system.

3. User must be connected to a Chrome cast device and have an active web app Session.

* **Description:**

1. The use case begins when the user selects a campaign from the campaign list.

2. The system then sets the campaign as the active campaign.

3. The system then sends a JSON message to the custom receiver application.

4. The system then receives the message and parses the message.

5. The system identifies the message and fetches the necessary data.

6. The use case ends when the system creates and displays a image slideshow object with the fetched data.

**Post conditions:**

1. The system stores the selected Campaign.

2. The system displays a image slideshow based on the campaign selected.

**Alternative Courses of Action**: N/A

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent on external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** March 20, 2015
* **Last Modified:** March 20, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-006/Switch Campaign

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the system.

3. User must be connected to a Chrome cast device and have an active web app Session.

* **Description:**

1. The use case begins when the user selects a campaign from the campaign list.

2. The system then sets the campaign as the active campaign.

3. The system then sends a JSON message to the custom receiver application.

4. The system then receives the message and parses the message.

5. The system identifies the message and fetches the necessary data.

6. The use case ends when the system modifies the image slideshow object with the new fetched data.

**Post conditions:**

1. The system stores the selected Campaign.

2. The system displays a image slideshow based on the campaign selected.

**Alternative Courses of Action**: N/A

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent on external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** March 22, 2015
* **Last Modified:** March 22, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-007/ Play Pause Slideshow

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must be logged into the system.

3. User must be connected to a Chrome cast device and have an active web app Session.

* **Description:**

1. The use case begins when the user selects a campaign from the campaign list.

2. The system then sets the campaign as the active campaign.

3. The system then sends a JSON message to the custom receiver application.

4. The system then receives the message and parses the message.

5. The system then identifies the message.

6. The use case ends when the Image Slideshow is either paused or resumes depending on the message sent.

**Post conditions:**

1. The system Image slideshow on the Chrome cast receiver application is either paused or resumed.

**Alternative Courses of Action**: N/A

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent on external entities.

**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** March 24, 2015
* **Last Modified:** March 24, 2015

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case ID:** S-WALL/2-008/Rejoin Web App

**Use Case Level:** High Level

**Details:**

* **Actor:** User
* **Preconditions:**

1. User must have opened the application on their mobile device.

2. User must have logged into the system.

3. System must be connected to a Chrome cast device.

4. Custom receiver app must be running on the Chrome cast device.

* **Description:**

1. The use case begins when the user presses the “Join Web App” button.

2. The system shall send the custom Chrome cast application a rejoin signal.

3. The use case ends when the system establishes a web session with the Chrome cast receiver application.

**Post conditions:**

1. The user has an active web app session with the receiver application.

**Alternative Courses of Action**: N/A

**Exceptions:**

* The system is unable to establish a web app session.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used on average 3 times the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Dependent upon external entities.

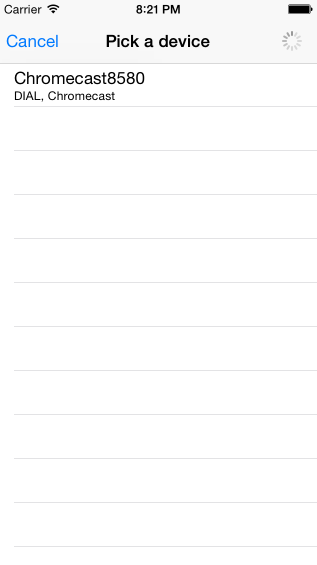
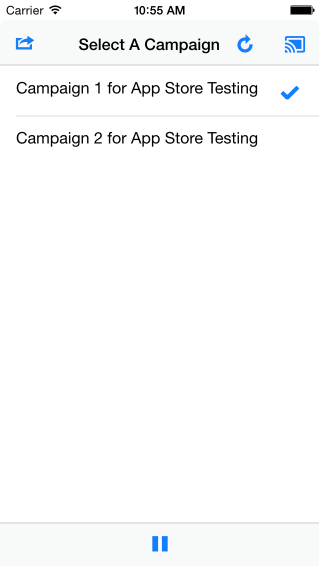
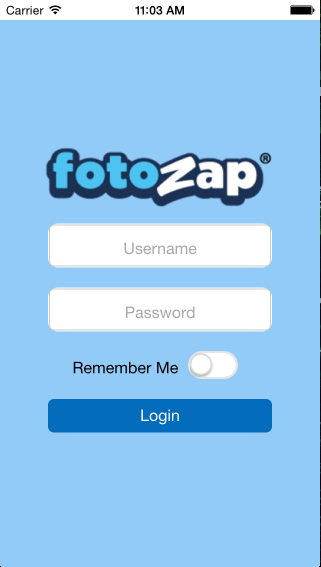
**Constraints:**

**Modification History:**

* **Owner: Steve Noel**
* **Initiation Date:** April 9, 2015
* **Last Modified:** April 9, 2015

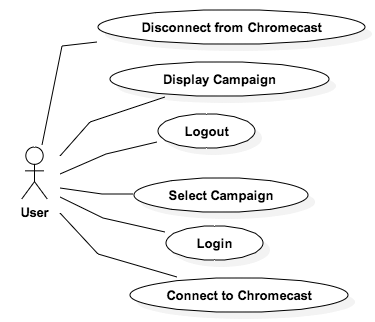
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9.3 Appendix C - User Interface Designs**

****

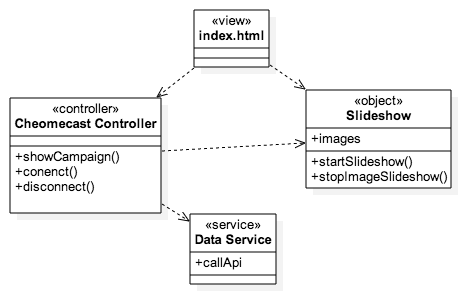


**9.4 Appendix D - Analysis Models (Static and Dynamic)**

****

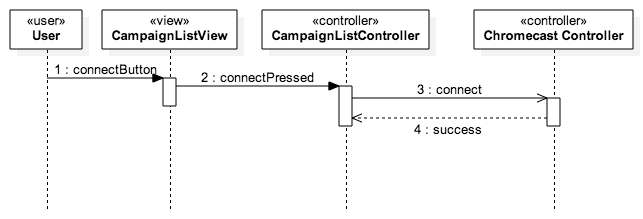


Sender Application Class Diagram

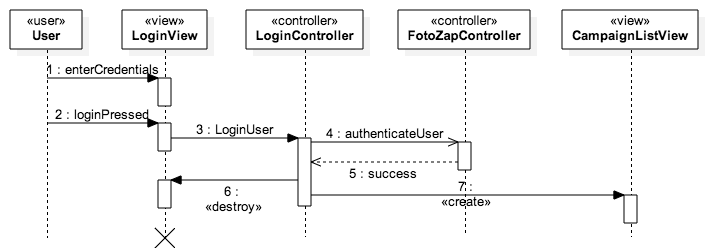


Custom Chrome cast receiver application Class Diagram.

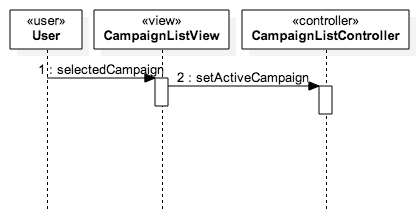
**Connect to Chrome cast Sequence Diagram**



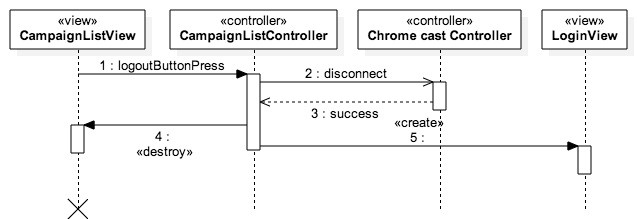
**Login Sequence Diagram**



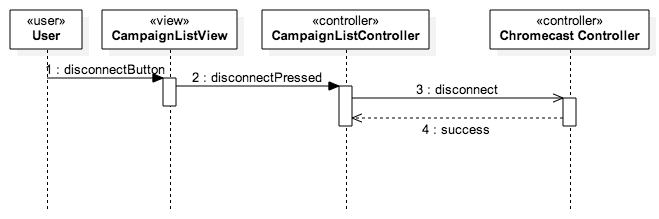
**Select Campaign Sequence Diagram**



**Logout Sequence Diagram**



**Disconnect from Chrome cast Sequence Diagram**

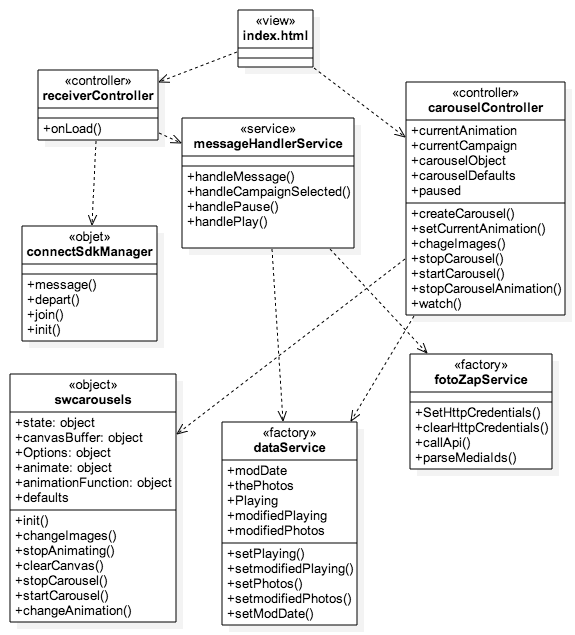


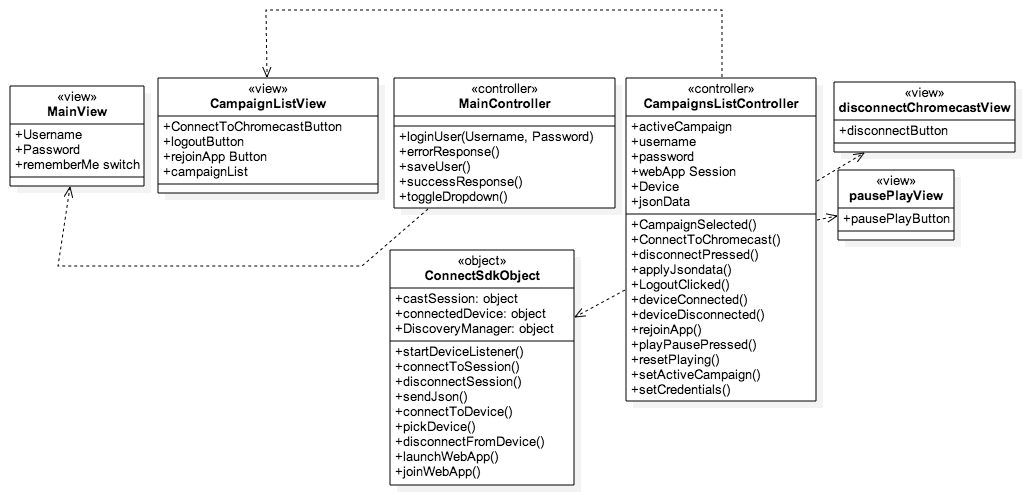
**Show Campaign on Chrome cast**



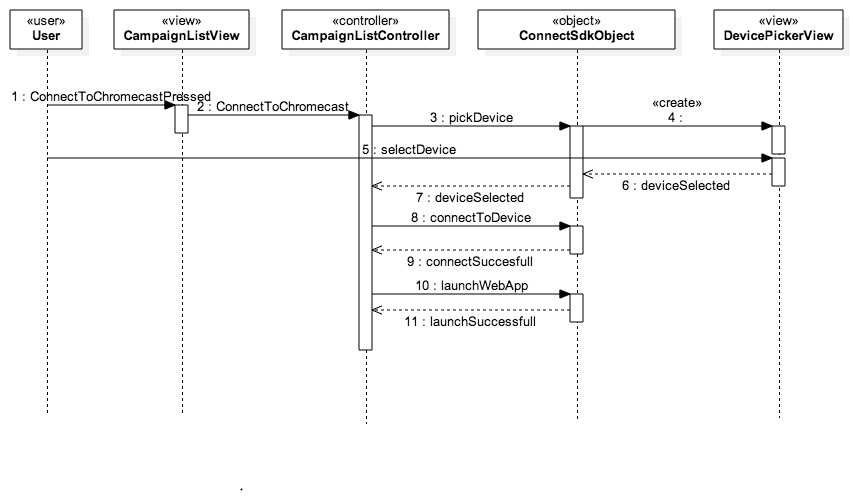
**9.5 Appendix E - Design Models (Static and Dynamic)**

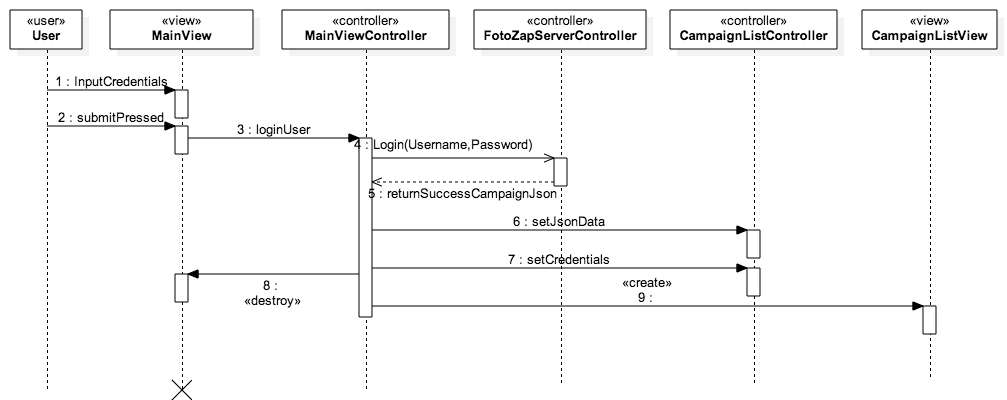
**Receiver Subsystem class diagram**



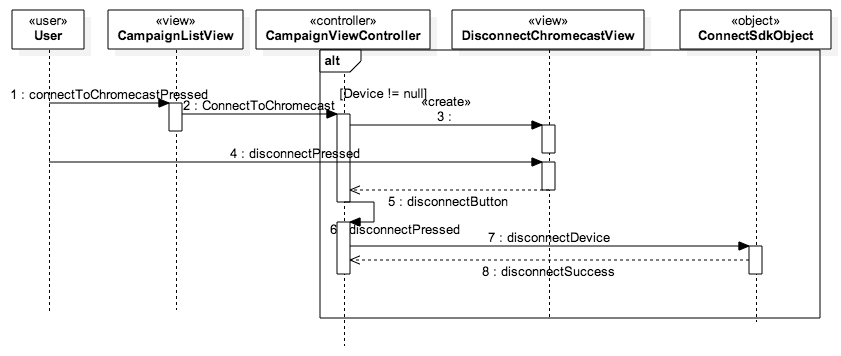


**Connect to Chromecast Sequence Diagram**

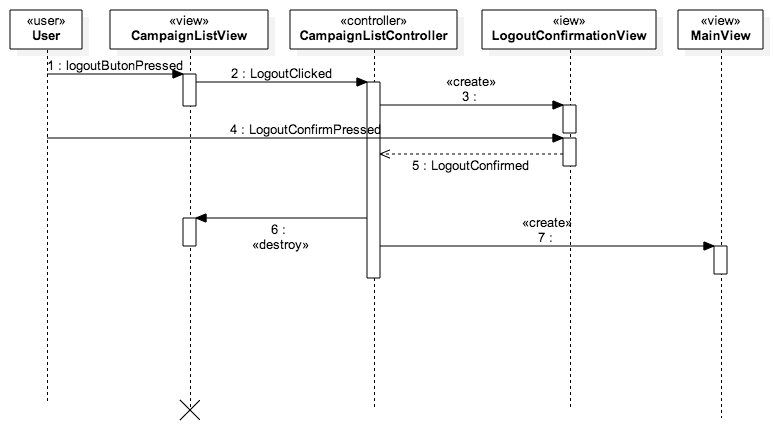


**Login Sequence Diagram**

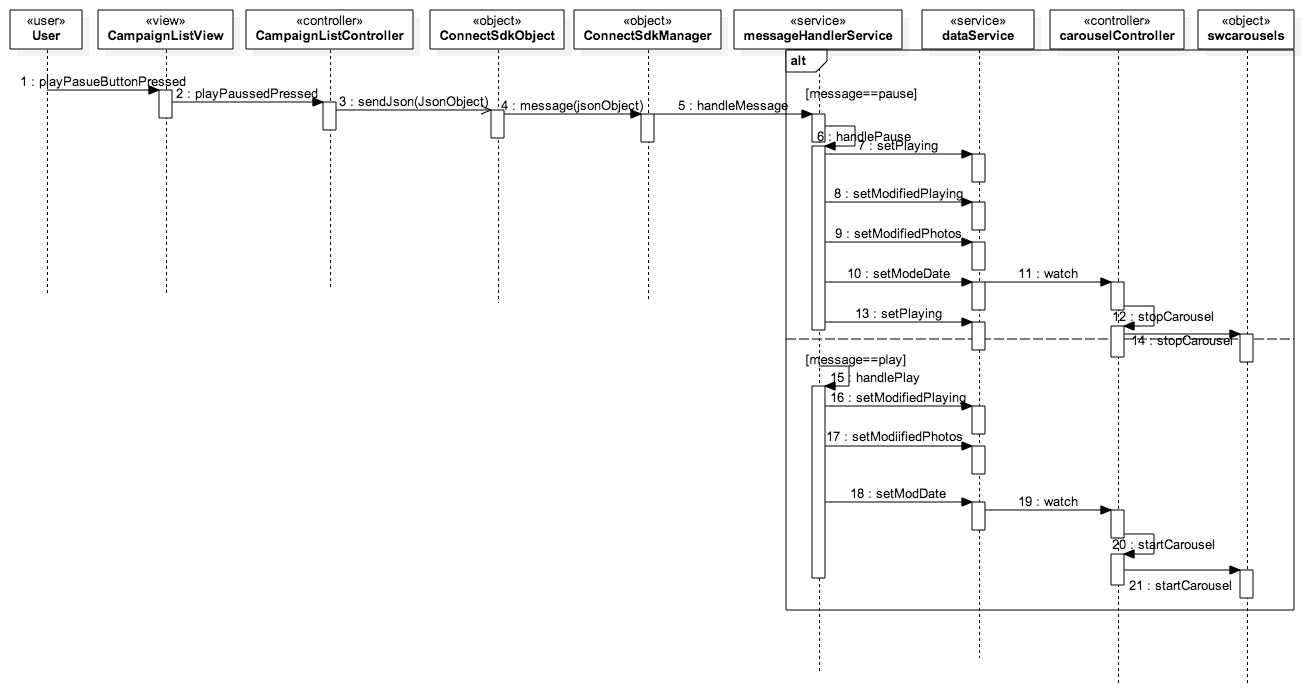
**Diconnect from Chromecast Sequence Diagram**



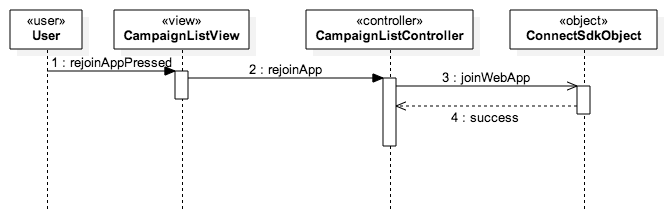
**Logout Sequence Diagram**



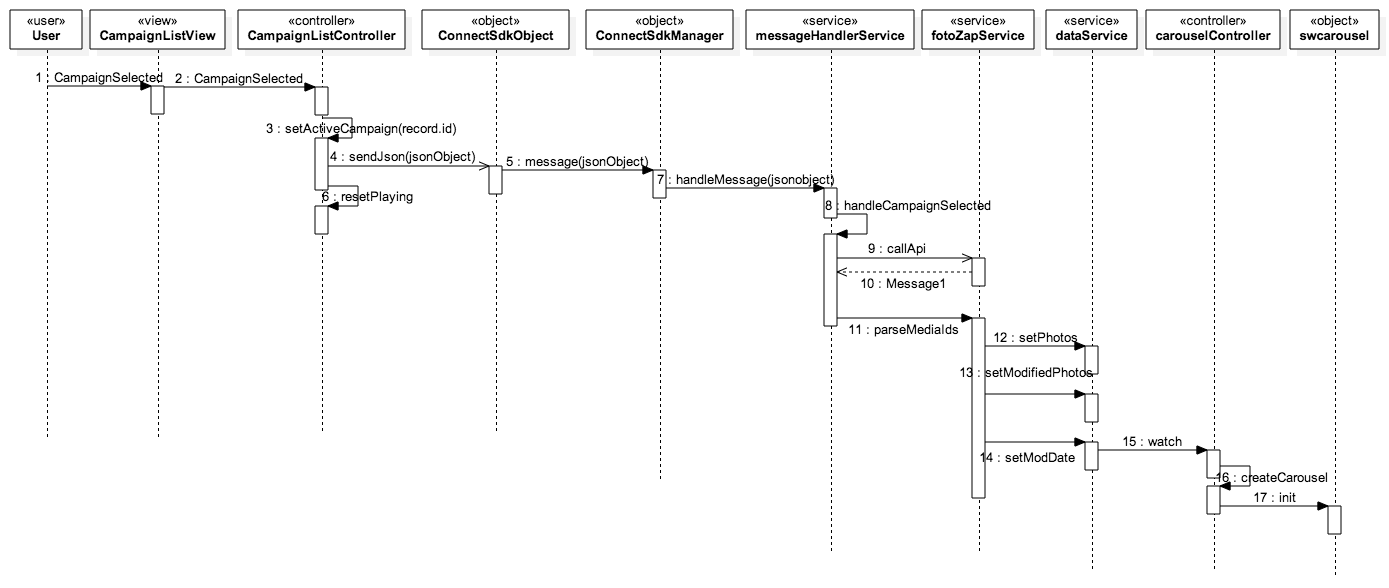
**Pause Play Sequence Diagram**



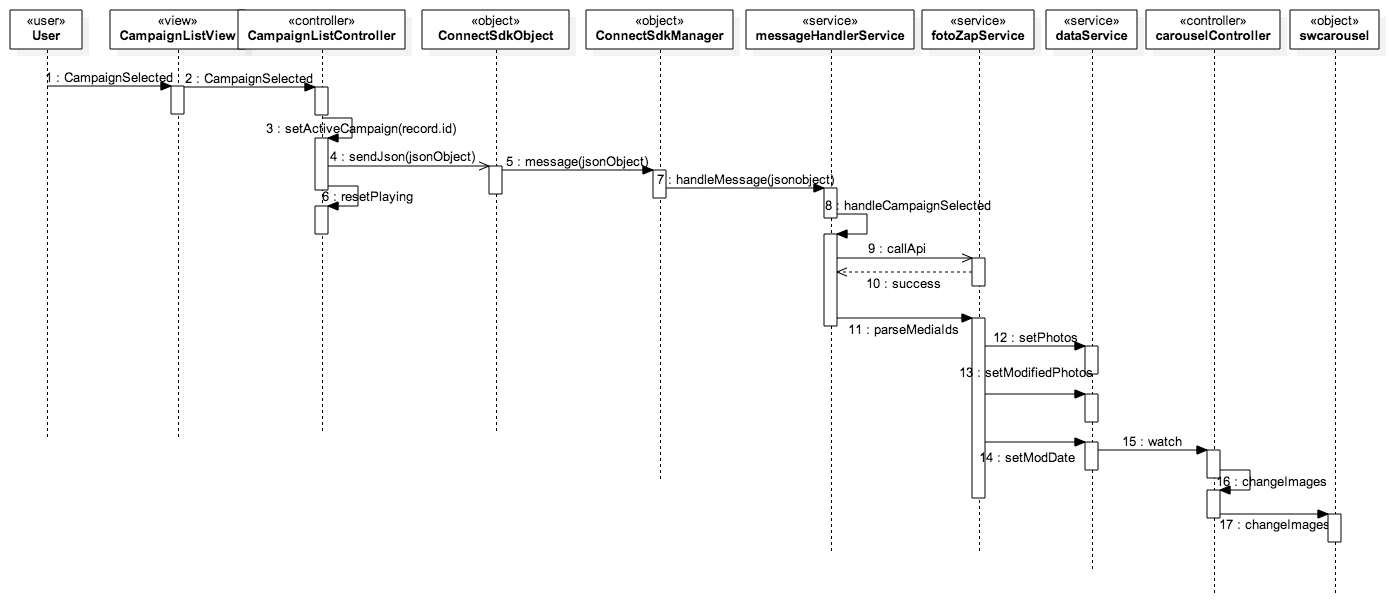
**rejoin Web App Sequnce Diagram**



**Select Campaign Sequence Diagram**



**Switch Campaign Sequence Diagram**



**9.6 Appendix F - Documented Class interfaces (code) and Constraints**

**//Code Specification for the Campaign List Controller**

Ext.define('FotoZap.controller.CampaignListController', {

extend: 'Ext.app.Controller',

requires: ['Ext.MessageBox', 'FotoZap.model.Campaign'],

config: {

webAppId: '1E0F8D69',

deviceIsConnecting: false,

playing: true,

lastSelectedRecord: null,

appSession: null,

jsondata: null,

username: null,

password: null,

device: null,

activeCampaign: null,

refs: {

logoutButton: 'titlebar #logoutbutton',

joinButton: '[itemId="joinbutton"]',

campaignList: '[itemId="theCampaignList"]',

castButton: 'titlebar #thecastbutton',

disconnectButton: '[itemId="disconnectChrome"]',

discoModal: {

selector: '#diconnectWindow',

xtype: 'disconnectModal',

autoCreate: true

},

campaignPage: 'campaignpage',

playpausePage: 'playpausebar',

playpauseButton: '[itemId="playpause"]'

},

control: {

joinButton: {

tap: 'rejoinApp'

},

playpauseButton: {

tap: 'playPausePressed'

},

disconnectButton: {

tap: 'disconnectPressed'

},

campaignList: {

select: 'CampaignSelected',

initialize: 'ListInit'

},

castButton: {

tap: 'ConnectToChromecast'

},

logoutButton: {

tap: 'LogoutClicked'

}

}

}

**//Code Specification for the Carousel Controller**

(function() {

'use strict';

//This controller handles the creation and

angular.module('social-wall-receiverApp')

.controller('carouselController', ['$scope', 'dataService', function($scope, dataService) {

$scope.paused = {};

$scope.currentCampaign = null;

$scope.currentAnimation = 'scroll';

$scope.isCurrentAnimation = function(aninm) {

return aninm === $scope.currentAnimation;

}

$scope.stopCarousel = function() {

$scope.paused.isPaused = true;

var pauseElement = angular.element(document.querySelector('#pauseIcon'));

if (pauseElement.hasClass('ng-hide')) {

pauseElement.removeClass('ng-hide');

}

if ($scope.carouselObject) {

$scope.carouselObject.stopCarousel();

}

}

$scope.startCarousel = function() {

$scope.paused.isPaused = false;

var pauseElement = angular.element(document.querySelector('#pauseIcon'));

if (!pauseElement.hasClass('ng-hide')) {

pauseElement.addClass('ng-hide');

}

if ($scope.carouselObject) {

$scope.carouselObject.startCarousel();

}

}

$scope.stopCarouselAnimation = function() {

$scope.carouselObject.stopAnimating();

}

$scope.chageImages = function(newimages) {

$scope.carouselObject.changeImages(newimages);

}

$scope.carouselDefaults = {

switchInterval: 5000, // millisecs between switch

width: window.innerWidth, // pixels

height: window.innerHeight, // pixels

speed: 8, // pixels/16 millisecs

animationFunction: 'hardcut'

};

$scope.createCarousel = function(parentElement, options) {

return new swcarousels(parentElement, options);

}

$scope.setCurrentAnimation = function(anim) {

$scope.carouselObject.changeAnimationFunc(anim);

$scope.currentAnimation = anim;

}

$scope.$watch(function() {

return dataService.getModDate();

}, function() {

console.log('In the data service watch function');

console.log(dataService.getmodifiedPhotos());

console.log(dataService);

if (dataService.getmodifiedPhotos()) {

var images = dataService.getPhotos();

if (images.length > 0) {

$scope.carouselDefaults.images = images;

if ($scope.carouselObject) {

console.log(images);

$scope.chageImages(images);

} else {

$scope.carouselObject = $scope.createCarousel(document.getElementById('main'), $scope.carouselDefaults);

}

}

}

console.log(dataService.getmodifiedPlaying());

if (dataService.getmodifiedPlaying()) {

if (dataService.getPlaying()) {

$scope.startCarousel();

} else {

$scope.stopCarousel();

}

}

}, true);

}]);

})();

**9.7 Appendix G - Documented Code for test drivers and stubs**

describe("Receiving a Message", function() {

beforeEach(module('social-wall-receiverApp'));

beforeEach(inject(function(\_fotoZapService\_,\_dataService\_,\_messageHandlerService\_,$controller,$rootScope,$httpBackend){

// The injector unwraps the underscores (\_) from around the parameter names when matching

fotoZapService = \_fotoZapService\_;

dataService = \_dataService\_;

messageHandlerService = \_messageHandlerService\_;

scope = $rootScope.$new();

controllerService = $controller;

// $scope:scope

//});

httpMock = $httpBackend;

}));

describe('Select Campaign Works', function () {

it('wea',function(){

httpMock.expectGET("https://zap-rest.fotozap.com/campaigns/3206/media?offset=0&limit=50").respond([{"id":"3412ffb3-0dd1-4cb9-ad09-34896dfa58c5","dateUploaded":"2015-02-16T17:49:20+0000","campaignId":3207,"campaignTitle":"The 2nd Campaign (stevenoel)","contentType":"PHOTO","photographer":"snoel006@fiu.edu","enabled":true,"hitCount":0,"photerranIds":["a19b1cf2-8599-48ff-a095-b57f692c8712"],"barcodes":[],"attributes":[{"key":"UploaderIP","value":"54.221.26.218","type":"RESERVED","dateCreated":"2015-02-16T17:49:26+0000"},{"key":"SerialNumber","value":null,"type":"RESERVED","dateCreated":"2015-02-16T17:49:26+0000"},{"key":"Filename","value":"phpmJPSg8.jpg","type":"RESERVED","dateCreated":"2015-02-16T17:49:26+0000"},{"key":"UserAgent","value":"Zend\\Http\\Client","type":"RESERVED","dateCreated":"2015-02-16T17:49:26+0000"}]},{"id":"0f022bb3-fa1a-46de-8e8c-099b6d9ad48a","dateUploaded":"2015-01-29T18:28:33+0000","campaignId":3207,"campaignTitle":"The 2nd Campaign (stevenoel)","contentType":"PHOTO","photographer":"snoel006@fiu.edu","enabled":true,"hitCount":3,"photerranIds":["9a4d7b93-c59b-4ae4-b51d-972d28c7551e"],"barcodes":["FZ435252142"],"attributes":[{"key":"Filename","value":"sample-3.jpg","type":"RESERVED","dateCreated":"2015-01-29T18:28:38+0000"},{"key":"SerialNumber","value":null,"type":"RESERVED","dateCreated":"2015-01-29T18:28:38+0000"},{"key":"UserAgent","value":"Zend\\Http\\Client","type":"RESERVED","dateCreated":"2015-01-29T18:28:38+0000"},{"key":"UploaderIP","value":"54.163.136.245","type":"RESERVED","dateCreated":"2015-01-29T18:28:38+0000"}]}]);

messageHandlerService.handleMessage({

type:'campaignseleted',

data:'snoel006@fiu.edu fotozap 3206'

});

ctrl = controllerService('carouselController', {$scope: scope});

httpMock.flush();

expect(dataService.modifiedPhotos).toBe(true);

expect(scope.carouselObject).toBeDefined();

});

});

});

**9.8 Appendix H - Diary of Meetings**

Diary Entry 1:

Date: February 8, 2015

Location: Picture Marketing Offices

Start time: 12:00 pm

End time: 1:30 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Introduction and initial meeting
* Learn about Picture Marketing.
* Explanation and details of the project
* Go over requirements of the system.
* Talk about expectations of the project.

Summary of Discussion:

Introduced to the Picture Marketing Developers and Management. Discussed the Social Wall project, what is needed, what tools to use and what is expected.

Diary Entry 2:

Date: February 13, 2015

Location: Picture Marketing Office

Start time: 12:00 pm

End time: 1:30 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Talk about implementation so far.
* Discussed use cases.
* Discussed requirements and constraints

Summary of Discussion:

Discussed the implemented user stories thus far and the issues that have been discovered during development.

Diary Entry 3:

Date: February 19, 2015

Location: Picture Marketing Offices

Start time: 12:30 pm

End time: 1:30 pm

In Attendance: Carlos Ocampo, Louis Zuckerman

Late: N/A

Agenda:

* Discuss Login Bug.
* Demo of current system.
* Discuss application control flow and current

Summary of Discussion:

A demo of the current system was shown. Discussed how to fix the login use case problem. Discussed the performance of the slideshow engine. Decided to change the slideshow to a canvas implementation.

Diary Entry 4:

Date: February 27, 2015

Location: Picture Marketing Offices

Start time: 12:00 pm

End time: 1:30 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Fixed Logout Use Case
* Discuss current user stories implemented.

Summary of Discussion:

Demoed the current version of the system. Discussed the implementation of the fotozapService and how the images are loaded from the server. Decided the development of the image slideshow most critical user story.

Diary Entry 5:

Date: March 6, 2015

Location: Picture Marketing Offices

Start time: 12:00 pm

End time: 1:00 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Slideshow engine

Summary of Discussion:

Discussed different canvas frameworks that can be used for the project especially the react-canvas framework for a crisp looking hardware- accelerated slideshow.

Diary Entry 6:

Date: March 20, 2015

Location: FIU ECS Computer lab

Start time: 12:00 pm

End time: 2:00 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Discuss progress
* Slideshow engine and Disconnect Chrome cast demo

Summary of Discussion:

Talked about the new slideshow engine and the Disconnect Chrome cast features. Demoed the current system. Talked about the efficiency of using a sliding animation on the Chrome cast device and decided to change the animation used to a hard cut animation.

Diary Entry 7:

Date: April 1, 2015

Location: Picture Marketing Offices

Start time: 12:00 pm

End time: 2:00 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Zuckerman

Late: N/A

Agenda:

* Demo current version of system
* Discuss bugs

Summary of Discussion:

Demoed the current system, added some functionalities that the system should have, discussed the aspect ratio bug and also discussed the joining the web app bug.

Diary Entry 8:

Date: April 9, 2015

Location: FIU SCS Computer Lab

Start time: 12:00 pm

End time: 1:30 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Discussed Refactoring Code
* Discussed future developer setup instructions
* Discussed UI fixes

Summary of Discussion:

Made sure developer setup instructions were clear so future developers can follow it. Also, made sure code was readable and self-documenting.

Diary Entry 9:

Date: April 17, 2015

Location: Picture Marketing

Start time: 12:00 pm

End time: 2:00 pm

In Attendance: Carlos Ocampo , Louis Zuckerman

Late: N/A

Agenda:

* Demo of Final Product
* Discussed submitting App to store

Summary of Discussion:

Demoed the final version of the system. Also discussed the submittal of the app store but where not able to finally submit to store. Explained the setup Instructions to Picture Marketing Developers.

**10. References**

1. <http://angularjs.org/> - Javascript MVC Framework.
2. <http://docs.sencha.com/touch/2.2.0/> - Sencha Touch MVC Framework for developing HTML mobile applications.
3. <http://connectsdk.com/>- Connects to Chrome cast device.
4. <http://cordova.apache.org/> - Porting web app to IOS
5. <http://nodejs.org/> - Porting web app to IOS
6. <https://www.npmjs.com/> - Package manager for receiver web application
7. <http://www.techsmith.com/camtasia.html/> - Screen Recorder for videos
8. <http://newegg.com/> - Prices for hardware and software requirements
9. <http://photoshop.com/> - Photoshop CC
10. <http://www.bryntum.com/docs/siesta/> - Unit and Integration testing for Sencha Applications
11. <http://jasmine.github.io/> - Unit Testing Framework for Web Applications.
12. <http://angular.github.io/protractor/#/> - Integration testing Framework for Angularjs web Applications