CIS 4911 – SENIOR PROJECT Ver. 2

Picture Marketing’s Social Wall

**Design Document**

Members: Steve Noel

Mentor: Carlos Ocampo

Instructor: Masoud Sadjadi

April 30, 15

**Abstract**

The Social Wall is a IOS mobile application that allows registered users to show their images as a slideshow on any display using a Chrome cast device. The purpose of this document is to outline in detail the Social Wall software, in particular, the architecture, subsystems, and code specifications of the Social Wall system.

**COPYRIGHTS AND TRADEMARK NOTICES**

This is the work of Steve Noel, unless specified otherwise. Carlos Ocampo and Louis Zuckerman of Picture Marketing provided aid and guidance.

**Table Of Contents**

|  |  |
| --- | --- |
| **Title** | **Page Number(s)** |
| 1. Introduction | 4 |
| 1.1. Problem Definition | 4 |
| 1.2. Design Methodology | 5 |
| 1.3. Terminology | 5 |
| 1.4. Overview of the Document | 5-6 |
| 2. System Design | 7-10 |
| 2.1. Overview | 7 |
| 2.2. Subsystem Decomposition | 8 |
| 2.3. Hardware and Software Mapping | 9-10 |
| 2.4. Persistent Data Management | 10 |
| 2.5. Security/Privacy | 10 |
| 3. Detailed Design | 11-23 |
| 3.1. Overview | 11 |
| 3.2. Static Model | 11-15 |
| 3.3. Dynamic Model | 16-21 |
| 3.4. Code Specification | 21-23 |
| 4. Glossary | 24 |
| 5. Appendix | 25-46 |
| 5.1. Appendix A - Use Case diagram | 25 |
| 5.2. Appendix B - Use Cases | 25-34 |
| 5.3. Appendix C - Class Interfaces | 34-42 |
| 5.4. Appendix D - Diary of Meetings | 42-46 |
| 6. References | 47 |

**1. Introduction**

This chapter pertains to introducing the project and defining the scope, purpose,

acronyms and background on the project. It also deals with explaining the overview of the entire document through our chapter-based format.

**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 Design Methodology**

The software process model being utilized is the agile development process. Regarding system requirements, utilizing agile development allows for dynamic development of the system, as well as easing the requirement elicitation with the client. Developing an application with various unknowns, the agile development model allows the development to be created in a dynamic, reoccurring, method using the sprint concept. The models that will be used to represent the Social Wall design are as follows

- Class Diagrams

- Sequence Diagrams

- Use Case Diagrams

- Package Diagrams

- Detailed/Minimal Diagrams

- State Chart Diagram

**1.3 Terminology - Definitions, Acronyms, and Abbreviations**

PM - Picture Marketing Inc. project Sponsor

Xcode – Apple Integrated Development Environment

SW - Social Wall system

Cordova - native mobile application framework

h/w - Hardware

s/w - Software

OS - Operating System

App - Application

CC - Chrome cast device

AJS - AngularJS Framework

JS - JavaScript language

CSS - Cascading Style Sheets language

ST – Sencha Touch 2 Mobile framework

CS – ConnectSdk Framework

MVC- Model-View-Controller architectural pattern

IOS – iPhone OS (Operating System)

HTTP – Hypertext Transfer Protocol

HTTPS – HTTP Secure

FotoZap – Picture Marketing Photo Capture and storage application

**1.4 Overview of Document**

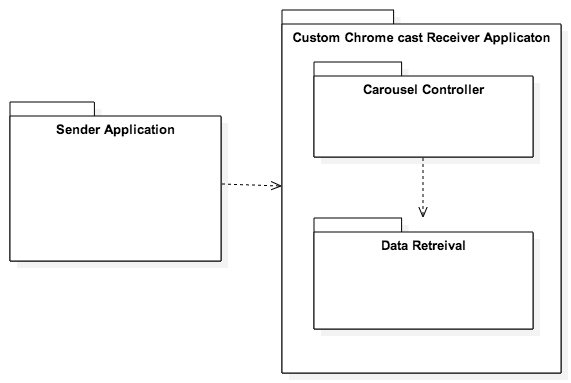
This document is divided into 5 main chapters. Each chapter is further divided into individual sections. This section marks the end of the first chapter. Chapter 2 focuses on the proposed software architecture of the *Social Wall*. Section 2.1 provides an overview of the chapter 2. Section 2.2 details the decomposition of the system into subsystems and identifies use cases associated with each subsystem. Section 2.3 focuses on the software and hardware utilized in our development process and maps out each of the subsystems to that hardware and software. Section 2.4 concerns itself with the persistent data of our system and how it will be managed. Section 2.5 discusses the security concerns of the *Social Wall*. Chapter 3 is focused on the design of the system in terms of the detailed models of the system. Section 3.1 introduces the classes we have decomposed the system into and identifies the design patterns used in the class structure. Section 3.2 is concerned with the control flow of the system and how the various objects interact with each other. Section 3.3 is the heart of the chapter and focuses on the detailed design of the system and explains the purpose of each class. Chapters 4 and 5 are the metachapters in that they concern themselves with the content in the previous chapters. Chapter 4 serves as the glossary of terms, and Chapter 5 contains the appendices.

**2. System Design**

A piece of software’s architecture is a description of its overall structure. The *Social Wall* uses two different architecture patterns, which make up its high level structure. An overview of this design will be described in section 2.1, with the succeeding sections detailing our system’s decomposition in terms of subsystems.

**2.1. 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. 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.

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.**

**2.2 Subsystem Decomposition**

The system is composed of four subsystems:

* Sender Application Subsystem
  + 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
* 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
  + 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
  + 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.

**2.3. 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.
    - Chrome cast - allows hdmi capable devices of running custom receiver applications, and allowing interactions through devices such as PCs and mobile devices.
    - 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.

**2.4. 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 login was successful. Persistent data is not necessary; as the system requires a network connection to function thus any data can be retrieved on the network.

**2.5. Security/Privacy**

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

However, 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. In addition, the system uses the HTTPS encrypted protocol for its network communications.

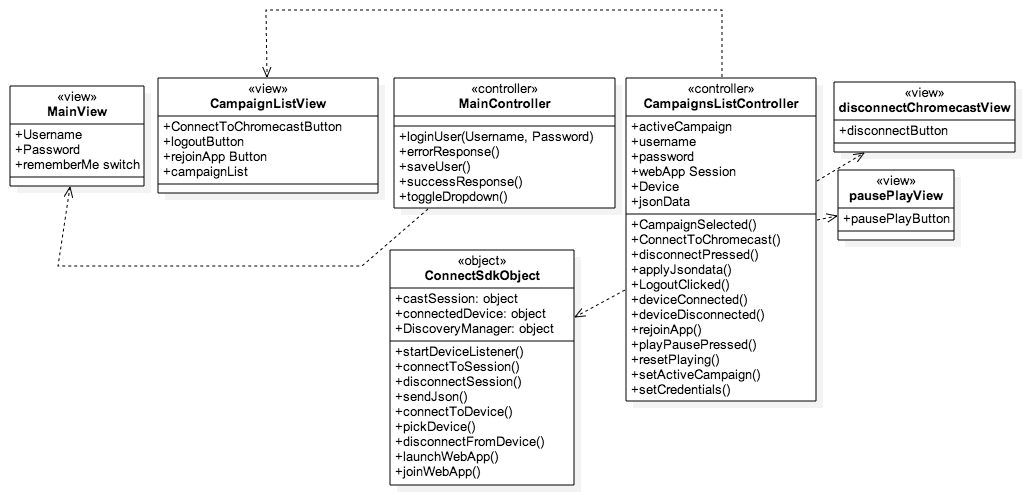
**3. Detailed Design**

This chapter deals with the detailed design of the system in terms of how the various classes interact with each other and the control flow in the system. It includes the static and dynamic models that describe the system, as well as the documented code that it makes it up.

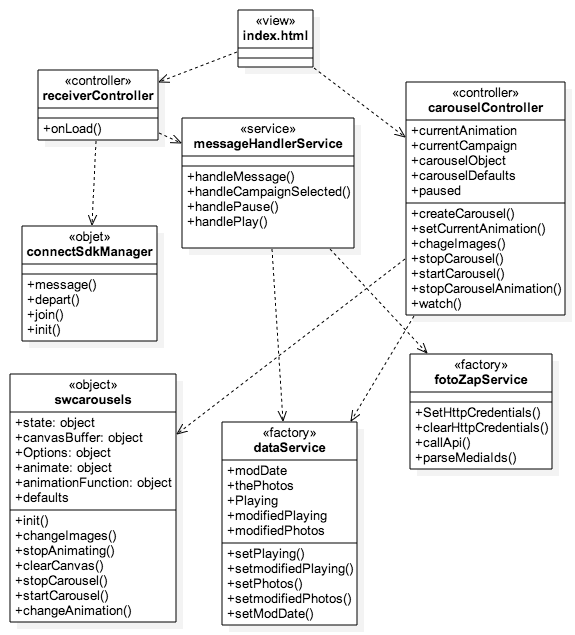
**3.1. Overview**

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.

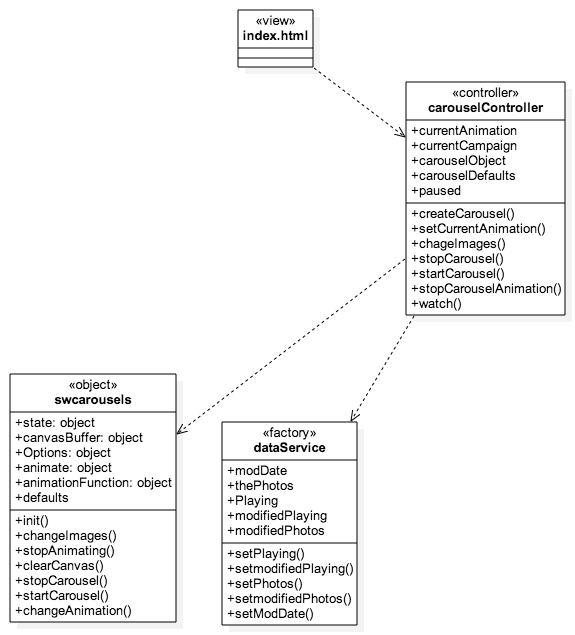
The design patterns used in the Social Wall system are the singleton, mediator, state and Iterator design patterns. The Singleton pattern was used because it centralizes the code and provides a single interface to access an object thereby reducing the complexity and increasing the simplicity and readability 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.

**3.2 Static Model**

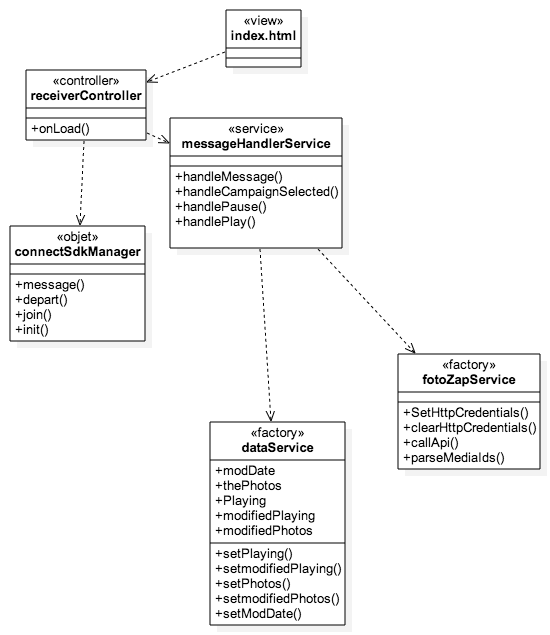
* **Sender Application Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Sencha Touch 2
  + Architecture: MVC & Cordova (App Only)
  + The sender application subsystem is made up of Cordova IOS application. This system provides the user interface that will allow for users to input information and 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.



* **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 communicates 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 framework which allows for applications to connect to the application.

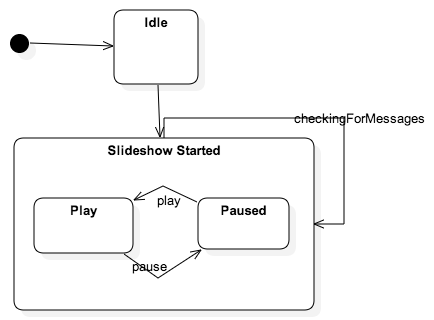


* **Carousel Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Angularjs
  + Architecture: MVC
  + This system is in charge of the entire Image slideshow of the receiver application. It creates the image carousel. It is in charge of pausing and playing the slideshow , displaying the images to the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

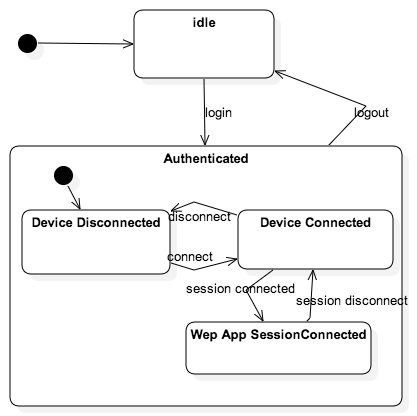


* **Data Retrieval Subsystem** 
  + HTML5-Javascript-CSS
  + Framework: Angularjs
  + Architecture: MVC
  + This subsystem is mainly concerned with retrieving and parsing the necessary data for the receiver application. This includes calling external API functionalities, parsing the data and then setting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

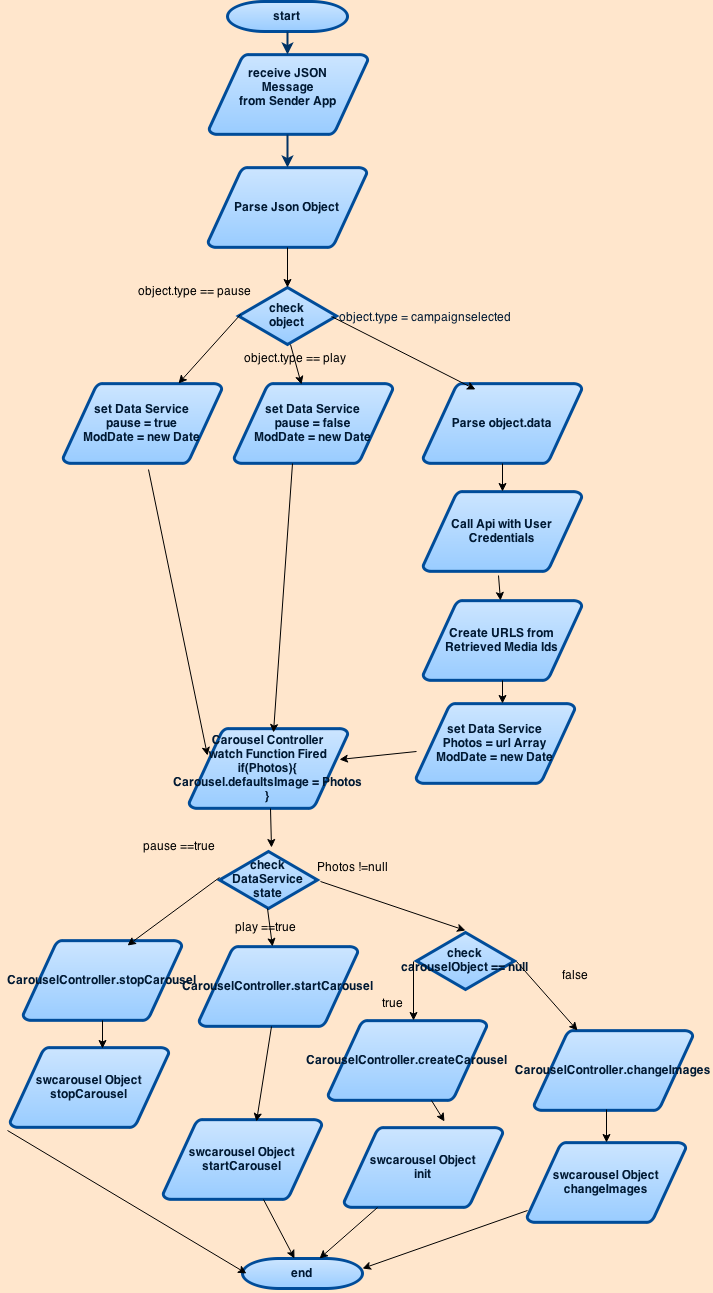
**3.3 Dynamic Model**

****

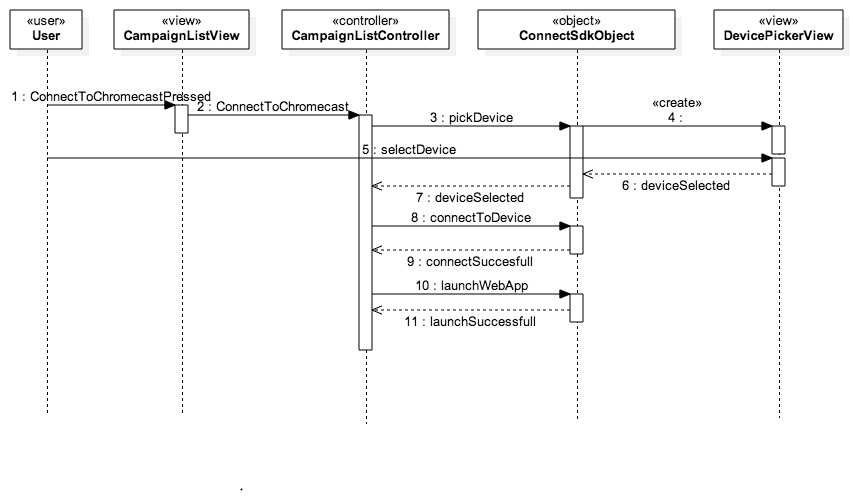
**State Machine Diagram – Carousel Controller.**



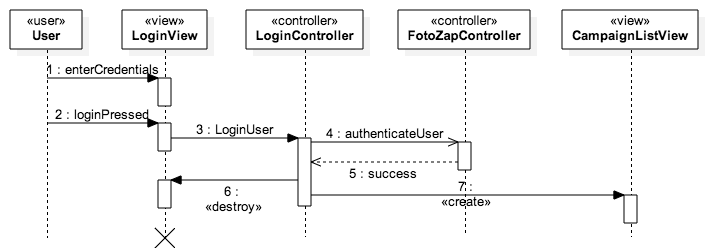
**State Machine Diagram –Campaign List View Controller.**



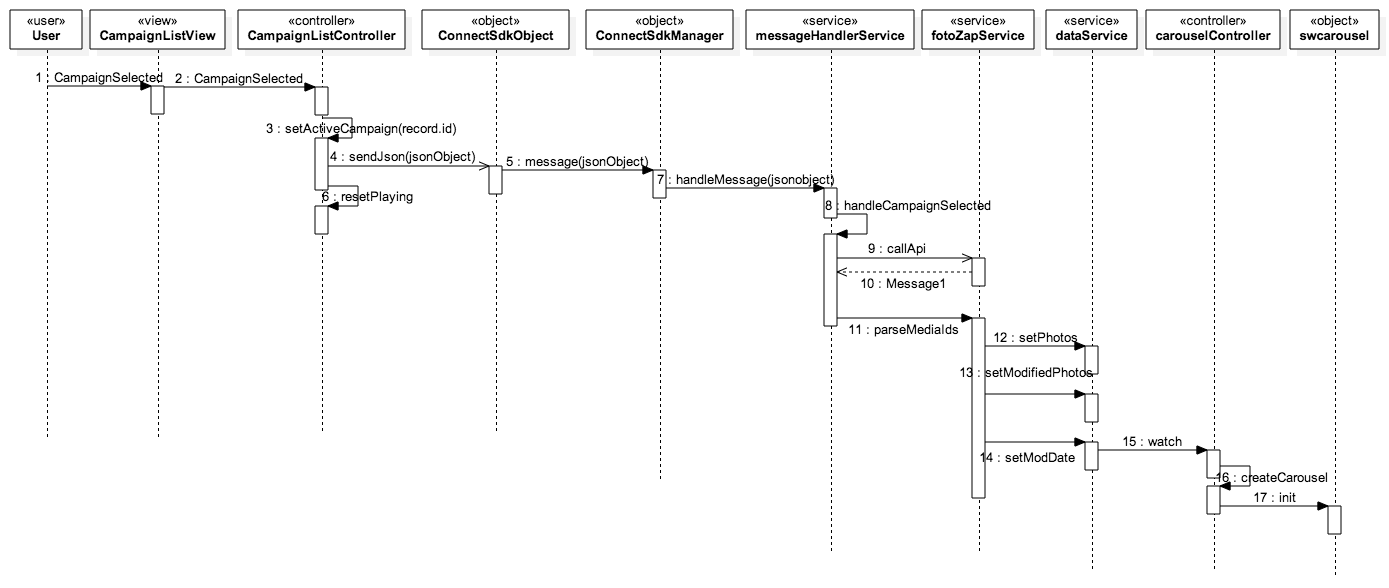
**Main Algorithm of System Flow Chart**



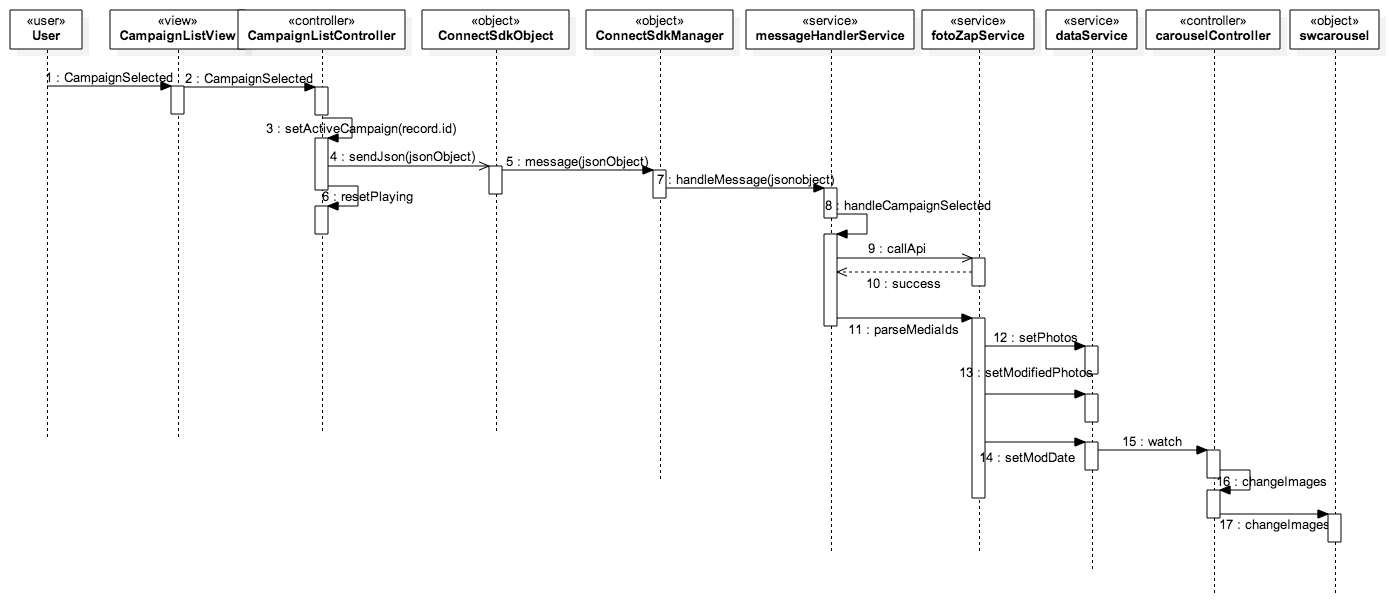
**Sequence Diagram of S-WALL/2-003/Connect to Chrome cast**



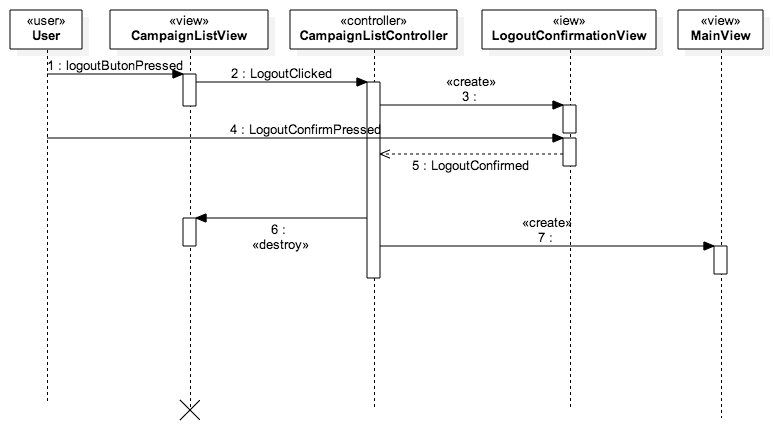
**Sequence Diagram of S-WALL/2-001/Login**



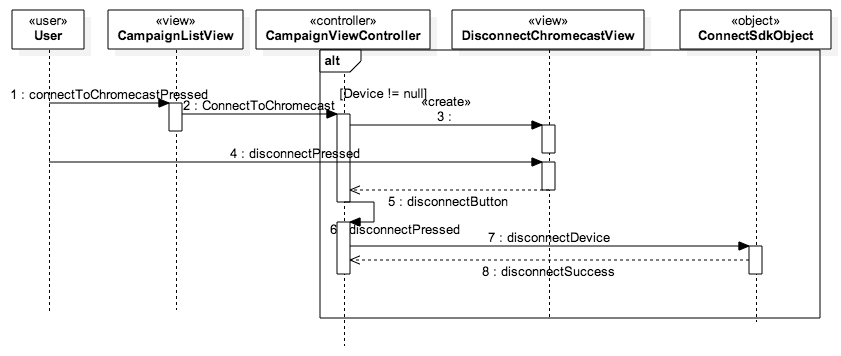
**Sequence Diagram of S-WALL/2-005/Select Campaign**



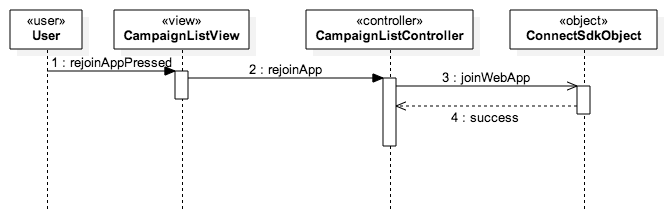
**Sequence Diagram of S-WALL/2-006/Switch Campaign**



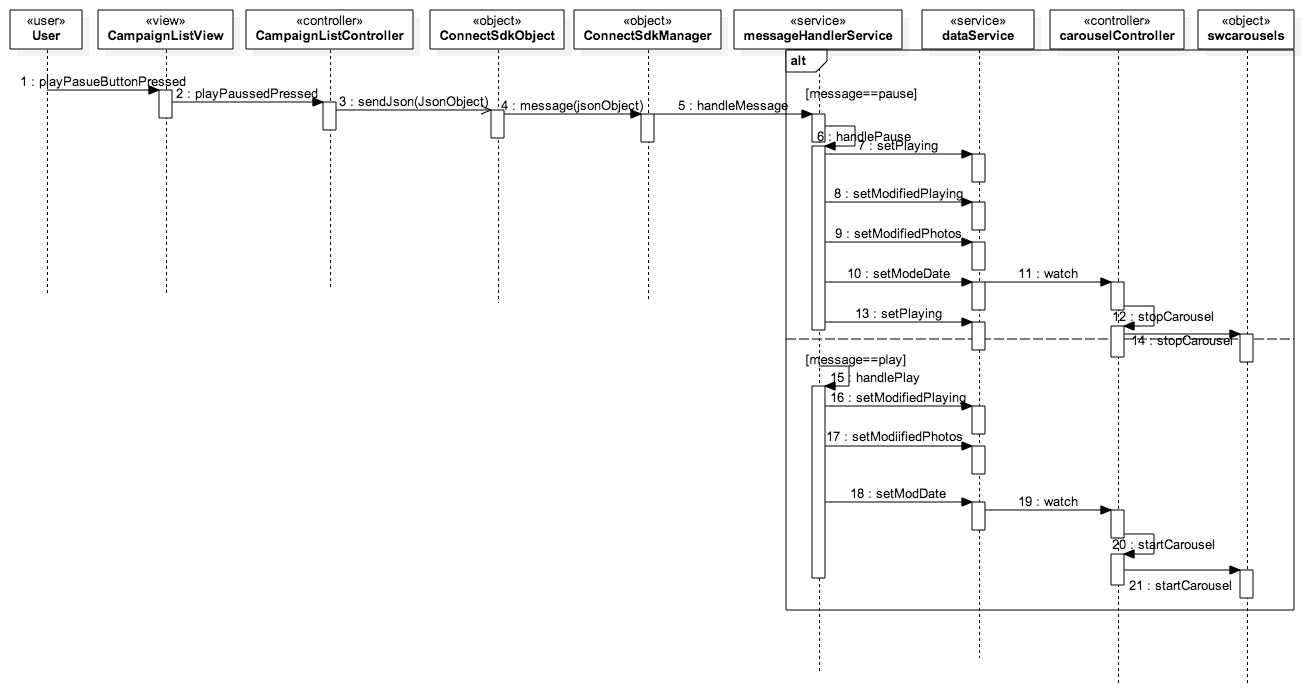
**Sequence Diagram of S-WALL/2-002/Logout**



**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**

**3.4 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. 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 the begins the disconnect Chrome cast feature.
* **disconnectPressed**
  + Direct trigger for the disconnect Chrome cast function.
* **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.

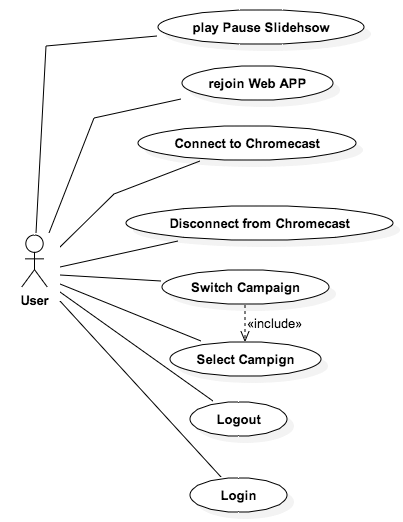
**Carousel Controller (methods)**

* **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.

**4. 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. |

**5. Appendix**



**5.1 Appendix A - Use case diagram of System.**

**5.2 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.

**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

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

**5.3 Appendix C – Code Specification**

**//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);

}]);

})();

**5.4 Appendix D – 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.

**6. 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