CIS 4911 – SENIOR PROJECT

Picture Marketing’s Social Wall

**Final Document**

Members: Steve Noel

Mentor: Carlos Ocampo

Instructor: Masoud Sadjadi

**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 |
| 2.3. Description of Alternatives | 9 |
| 2.4. Recommendations | 9-10 |
| 3. Project Plan | 11-13 |
| 3.1. Project Organization | 11-12 |
| 3.2. Identifications of Tasks and Deliverables | 12 |
| 3.3. Cost of the Project | 13 |
| 4. System Requirements | 14-15 |
| 4.1. Functional and Nonfunctional Requirements | 14-15 |
| 4.2. Analysis of System Requirements | 15 |
| 5. System Design | 16-18 |
| 5.1. Overview | 16 |
| 5.2. Subsystem Decomposition | 16-17 |
| 5.3. Hardware and Software Mapping | 17-18 |
| 5.4. Security/Privacy | 18 |
| 6. Detailed Design | 19-35 |
| 6.1. Overview | 19 |
| 6.2. Static Model | 19-22 |
| 6.3. Dynamic Model | 22-26 |
| 6.4. Code Specification | 26-35 |
| 7. System Validation | 36-38 |
| 7.1. Subsystem Tests | 36 |
| 7.2. System Tests | 37-38 |
| 7.3. Evaluation Tests | 38 |
| 8. Glossary | 39 |
| 9. Appendix | 40-74 |
| 9.1. Appendix A - Project Schedule | 40 |
| 9.2. Appendix B - All Use Cases | 41-49 |
| 9.3. Appendix C - UI Designs | 50 |
| 9.4. Appendix D - Analysis Models | 51-52 |
| 9.5. Appendix E - Design Models | 53-58 |
| 9.6. Appendix F - Documented class interfaces | 58-67 |
| 9.7. Appendix G - Documented code for test driver | 67 |
| 9.8. Appendix H - Diary of Meetings | 67-74 |
| 10. References | 75 |

**1. Introduction**

This section deals with introducing our project and defining our purpose, scope, terms, and acronyms. It also deals with describing how the rest of this document will unfold, describing our feasibility study, as well as our project plan.

**1.1 Problem Definition**

The problem our project, the Social Wall, deals with is the inability for clients to display photo albums in a slideshow from social media sites and cloud storage apps in a quick, inexpensive, and efficient manner. While it is possible to accomplish, it is normally tedious to set up and gives you a very limited amount of customization on how you want to show your slideshow. It also usually costs hundreds or even thousands of dollars a month to use competing software (e.g., tintup).

**1.2 Background**

The company we are working with for this web application , Picture Marketing, is known for their apps and products, like PhotoZap, which provides companies with the tools to market and promote their brands. With our app, Social Wall, we will aid them in aggregating images from different locations (DropBox, Facebook page albums, and RSS feeds) and repurpose them for event slideshows, website galleries, and other types of repurposing, so that more people can view them.

**1.3 Definitions, Acronyms, and Abbreviations**

PM - Picture Marketing

SW - Social Wall

FB - Facebook

PG - PhoneGap software

h/w - Hardware

s/w - Software

OS - Operating System

App - application

RSS - Rich Site Summary

CC - Chrome Cast

AJS - AngularJS Framework

JS - Javascript language

CSS - Cascading Style Sheets language

**1.4 Overview of document**

In this document, we will discuss our Feasibility Study and Project Plan. For the Feasibility Study we will discuss the current system in place today, the purpose of the new system we are going to implement, define our user requirements, describe the alternative solutions, and offer recommendations. This gives us our purpose behind our project and helps us move forward.

In our Project Plan, we discuss how we will organize our project, which includes our personnel organization and our h/w and s/w resources, and we identify our task, milestones, and deliverables for the rest of our project.

**2. Feasibility of Study**

**2.1 Description of Current System**

In the current system, there are two ways to solve this slideshow problem: doing it locally, by using broad slideshow applications like PowerPoint, or by using an expensive web application like Tintup. To display a slideshow locally, the consumer has to either download the images directly into their device and run it through a desktop slideshow application, or click through each individual image directly through social media site and display the images that way. This current system forces companies forces companies to dedicate time and resources for the creation of an appealing final product.

Another option for these consumers are through web application that provides this slideshow service (like Tintup or Postano). These services, while useful, can be rather costly, and can cost upwards of $1,000 a month (or more). While it might be affordable for some companies, it is a price that is too much for others, as well as individual users.

**2.2 Purpose of New System**

The purpose of the new system is to allow for the automation of this slideshow creation process through much simpler and cost-effective means. The expensive and unintuitive alternatives discourage individuals from using this service, and force companies, both large and small, to spare thousands of dollars for a service they may only use a handful of times. The new system will eradicate that by offering a simplified, yet elegant alternative that costs very little to use.

Another purpose is to add more customization to the slideshows. The current system gives the consumer a bare-bones version of a slideshow. Our new system will give more power to the consumer to make their slideshow to their own liking.

**2.2. Description of alternative solutions considered**

Many solutions were considered for this project. One was to make a native desktop app for Windows machines. In this alternative, the consumer would enter in his/her photo source, RSS feed, Facebook, or DropBox, the app would save it, and generate a slideshow on the application itself. This desktop app can be developed using Microsoft Visual Studio using C# and .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 SW the largest possible user base. This alternative also gives us 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 native apps for iOS, Android, and/or Windows Phone. This provides an even simpler alternative for consumers to generate their desired slideshow through a simple select interface. It would also be an inexpensive alternative for the SW users, since the app would likely cost less than $5 to download from their cell phone’s app store. A phone app would also make it possible to cast the application to Chrome Cast, which would make presenting the slideshow even easier. For this solution, we could develop it like the web app (JS, HTML, CSS, and AJS) and convert it into a phone app with the aid of s/w like PhoneGap/Cordova.

**2.3. Recommendations**

For Social Wall, we decided to make a web application (so that it is compatible with all OS) that retrieves photo albums from specific media sources, such as Facebook, RSS, and DropBox, and sends it to a slideshow engine that displays it for them. This was chosen because it gives us the largest possible user base for the app. It also allows us to port the app into a native mobile application via Cordova/PhoneGap.

We chose to do a mobile app because of the aforementioned PhoneGap software that makes it a simple port, and because it simplifies our app to the general consumer and allows them to generate a slideshow very quickly. We chose to develop for the Android OS because of the high market share it holds globally.

Both alternatives were also chosen, in addition to the ones listed above, due to the fact that the can also be cast to Google’s Chrome Cast, opening up more options for our potential user base.

**3. Project Plan**

**3.1 Project Organization**

**3.1.1. Project Personnel Organization**

**Joseph Gonzalez** - Developer, Project Manager, Tester

**Juan Gonzalez-Llanos** - Developer, Project Manager, Tester

**Cortney Mills** - Mentor

**Louis Zuckerman** - Mentor/Consultant

**3.1.2. Hardware and Software Resources**

**Hardware**:

* PC - Windows OS (7 or 8) or Mac (OS X or higher)
* Google Chrome Cast
* Television that accepts HDMI

**Software**:

* GitHub (2.0 for Windows or Web application)
  + Open Source Code Repository
* StarUML (version 2.0.0)
* Trello (web application)
  + Project Management tool
* Java EE
  + Development/Testing of Application
* Tomcat 7.0
* Eclipse IDE
* Google Chrome
* Adobe Photoshop CS6
* Google Drive
* Wamp Server
* Android Development Tools
* Webstorm
* Cordova/Phonegap
* AngularJS Framework
* Bootstrap UI framework

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

**Tasks:**

* Develop ability to display slideshow
* Implement slideshow transitions
* Develop ability to display background music
* Implement ability to receive RSS image albums
* Obtain FB API
* Secure connection to FB to receive image albums
* Obtain Chrome Cast API

**Milestones:**

* Create UI design
* Develop Slideshow Engine
* Cast app to Chrome Cast
* Develop app within AngularJS Framework
* Receive all possible photo sources
* Send sources to SS engine
* Port web app to Android app

**Deliverables:**

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

**3.3. Cost of the Project**

For this project, both Joseph and Juan have generously decided to forgo payment for their 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) |
| Laptop PC running Windows 7 or better, 8+ GB RAM, Intel i7 processor or better (x2) | $899.99  x2 |
| HDTV with HDMI port | $297.99 |
| Adobe Photoshop CC for four months, for two licenses | $19.99/mo  x8 |
| Chrome Cast | $34.99 |
| WebStorm Licenses (x2) | $99  x2 |
| Total (taxes and fees included): | $2,617.96 |

For references on these prices, observe section 10, located on page 75.

**4. System Requirements**

In this section, the proposed system requirements are presented and discussed in greater detail. The systems functional requirements along with their non-functional requirements are explained. UML diagrams describing both the static and dynamic aspects of the system are shown and expanded upon.

**4.1 Functional and Nonfunctional Requirements**

**Functional Requirements:**

1) The system shall allow the user to generate a slideshow.

Refer to Appendix B: S-WALL/1-001/Generate Slideshow

2) The system shall allow the user to change the image cycle speed.

Refer to Appendix B: S-WALL/1-002/Image Cycle Speed

3) The system shall allow the user to change the background photo of the slideshow.

4) The system shall allow the user to digest photos from an RSS Feed

Refer to Appendix B: S-WALL/1-008/Digest RSS Feed

5) The system shall allow the user to choose the source of photos for slideshow.

Refer to Appendix B: S-WALL/1-009/Select Photo Source

6) The system shall allow the user to cast mobile app onto Chrome Cast device.

Refer to Appendix B: S-WALL/1-010/Cast to Chrome Cast

7) The system shall capture requests made by users, and handle them respectively.

Refer to Appendix B: S-WALL/1-011/Handle Cast Request

8) The system shall allow user to change the transition effect of their slideshow

Refer to Appendix B: S-WALL/1-012/Change Effect

9) The system shall allow the user to add music to slideshow

Refer to Appendix B: S-WALL/1-013/Add Music

**Non-functional Requirements:**

1) The system shall retrieve photo album from source in less than ten seconds

2) The system shall generate slideshow on web app in less than five seconds

3) The system shall generate slideshow on Chrome Cast in less than ten seconds

**4.2 Analysis of System Requirements**

The SW system requires basic function of receiving the photo album source, sending it to slideshow controller, which passes it to the page view to display. It also allows the user to change the settings, like transition animation and speed, as well as cast it to Chrome Cast. This is all done quickly, efficiently, and safely.

**5. System Design**

A piece of software’s architecture is a description of its overall structure. The *Social Wall* utilizes an MVC architecture. I.e. Model-View-Controller structure. An overview of this design will be described in section 5.1, with the succeeding sections detailing our system itself.

**5.1. Overview**

The MVC architectural pattern splits the system into three main parts, each responsible for a specific role in the system. The three parts are the Model, View, and Controller, respectively. The model is responsible for managing the data of the application. The view is ultimately the user interfaces which presents the data based on the controllers decisions. The controller of the system acts as the manager that 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 social wall has three systems that are the chrome cast sender applications, the slideshow engine, and the custom chrome cast receiver application.

The system as a whole can be described as a client server architecture. Where the clients utilize a mobile application and Chrome desktop browser to make requests to a server side application that can connect to chrome cast devices and displays content requested by the user.

**5.2. Subsystem Decomposition**

The system is composed of four subsystems:

* Sender Application Subsystem
  + Sender application subsystem plays the role of interacting with the user and allowing them to interact with the system. The third subsystem is the receiver subsystem. The sender application subsystem retrieves the input from the user such as the RSS Feed URL, transition, transition speed, as well as where he would like the data to be digested from. The sender application also allows the user to create a session with chrome cast devices connected on the same network, and sending the user input to the receiver subsystem.
* Slideshow Engine Subsystem
  + Slideshow engine subsystem is in charge of the photo manipulation which is the content that will be displayed after user requests. Slide show engine subsystem works alongside the receiver subsystem to provide the user’s with the content that they requested utilizing the sender applications.
* Data Retrieval Subsystem
  + Data retrieval subsystem is the subsystem that deals with the digesting of digital media, mainly photos, and feeds into the receiver subsystem and slideshow engine subsystems in order to display the intended content. Its main function is to retrieve data from existing media online.
* Receiver Subsystem
  + The receiver subsystem is a custom chrome cast receiver application in charge of handling the sender subsystems requests, working alongside the slideshow engine subsystem, and displaying content onto chrome cast devices. The receiver subsystem receives the data input sent by the sender application and then requests data from the data retrieval subsystem.

**5.3. Hardware and Software Mapping**

* Users Hardware/Software
  + Hardware: Computer, PC, Android Device, Google Chrome cast, HDMI capable device.
    - Computer and android device allow users to run the sender applications that allow interaction with other parts of system.
    - Chrome cast - is required in order to display content onto intended device.
    - HDMI capable device - is necessary in order for the Chrome cast functionality, plays the role as the host device running the custom receiver application.
    - Chrome cast - allows hdmi capable devices of running custom receiver applications, and allowing interactions through devices such as PCs and mobile devices.
  + Software:
    - Desktop Google Chrome browser - installed alongside google chrome cast plugin. Sender application is created for
    - Android Operating System - Allows for the mobile sender application to run and allows user interaction with custom receiver application and chrome cast.

**5.4. Security/Privacy**

As the current system stands, security and privacy is not a major concern. However, through program design, the system must be able to take account for future security concerns. The data retrieval subsystem may in the future allow for user authentication from existing media requiring authentication online. In order to combat this, a modular approach is taken in the design of this component that will allow for user data to either be temporary, as well as secure from user misuse.

**6. Detailed Design**

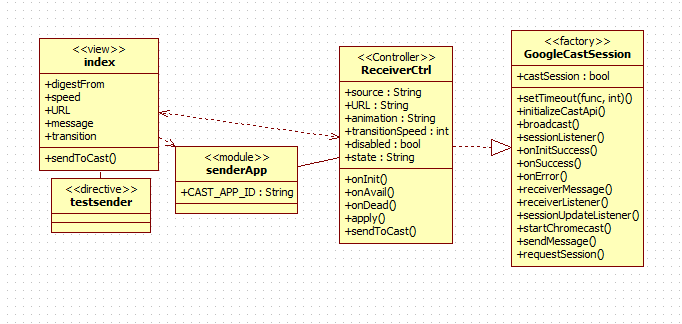
This section deals with the detailed design of our system. It includes the static and dynamic models that describe our system, as well as the documented code that it makes up.

**6.1. Overview**

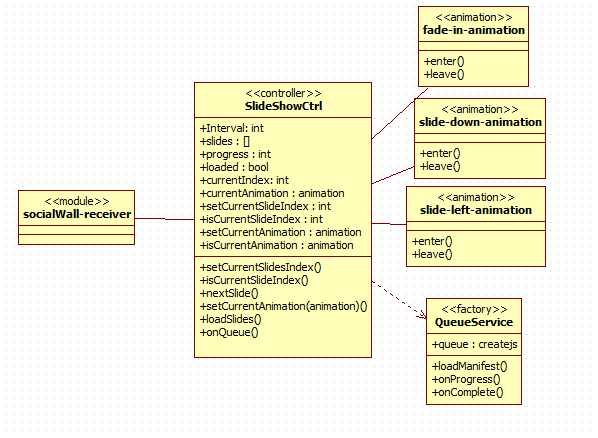
The Social Wall system is comprised of four subsystems. One (Data Retrieval) that retrieves the source of the photo album, one (Sender) that sends the information the user inputs, one (Receiver) that receives and store the data that is sent, and one (Slideshow Engine) that displays the data that was retrieved.

**6.2. Static Model**

As stated, the Social Wall system is composed of four subsystems.

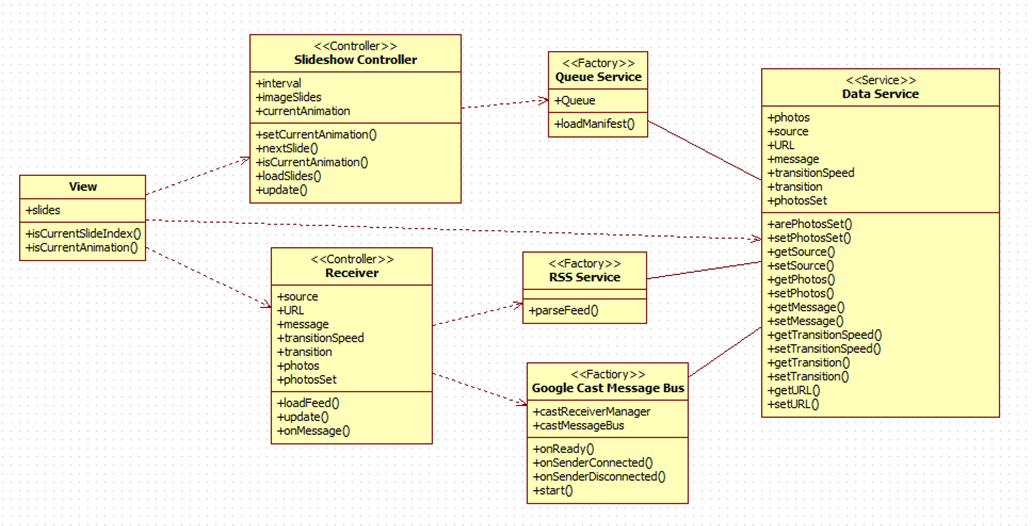
Sender Application Class Diagram

* **Sender Application Subsystem** 
  + HTML5-CSS
  + Framework: AngularJS
  + Architecture: MVC & Cordova(App Only)
  + The sender application subsystem is made up of an android application as well as a Chrome 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 receiver subsystem and communicates messages along. These messages contain the user input data that will later be used in the other subsystems.



Slideshow Engine Class Diagram

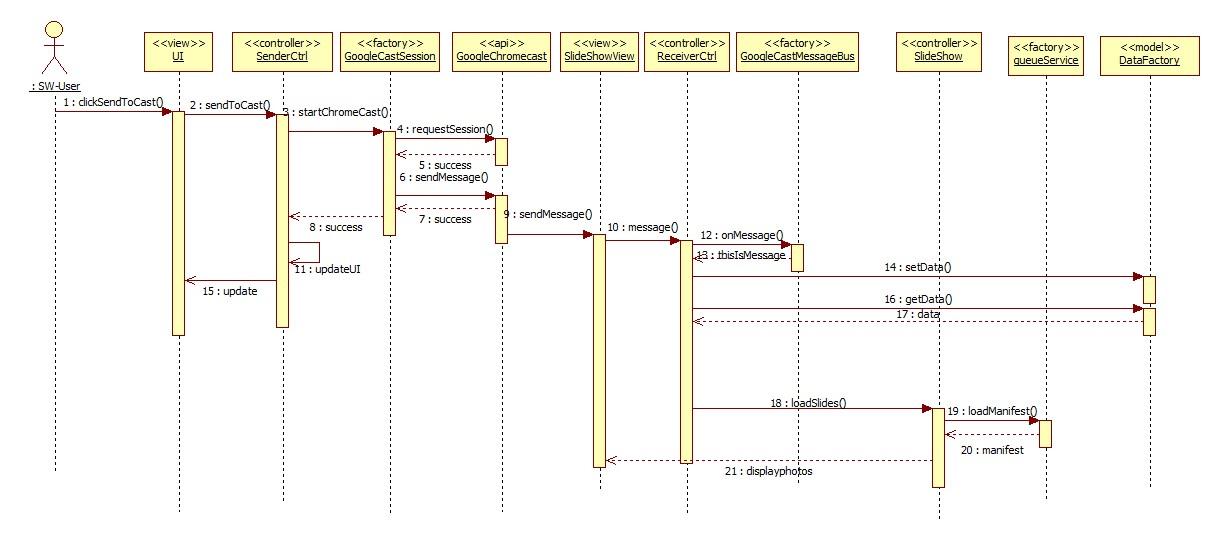
* **Slideshow Engine Subsystem** 
  + Framework: AngularJS
  + Architecture: MVC
  + Slideshow engine subsystem is in charge of the photo manipulation which is the content that will be displayed after user requests. Slideshow engine system utilizes an interface approach to work alongside other subsystems. The data retrieval subsystem feeds data directly into the data source that is utilized for the slideshow engine system, and the user input sent from the sender application is used in the generating of a slideshow that is displayed by the receiver application.
* **Data Retrieval Subsystem**
  + Framework: AngularJS
  + Architecture: MVC
  + Data retrieval subsystem is the subsystem that deals with the digesting of digital media, mainly photos, and feeds into the receiver subsystem and slideshow engine subsystems in order to display the intended content. Its main function is to retrieve data from existing media online. Depends on the sender subsystem as this subsystem retrieves data depending on the source that is decided on by the user. The receiver subsystem calls upon this subsystem and controls this subsystems actions.



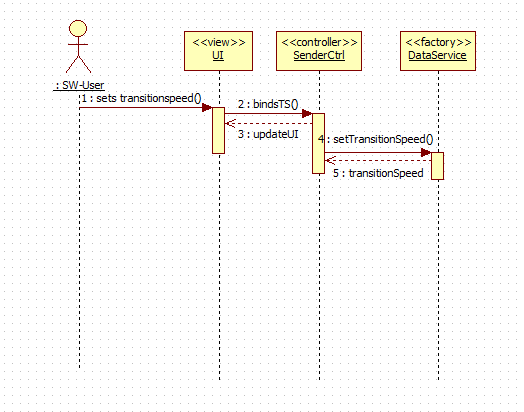
Class diagram of Custom Receiver Subsystem

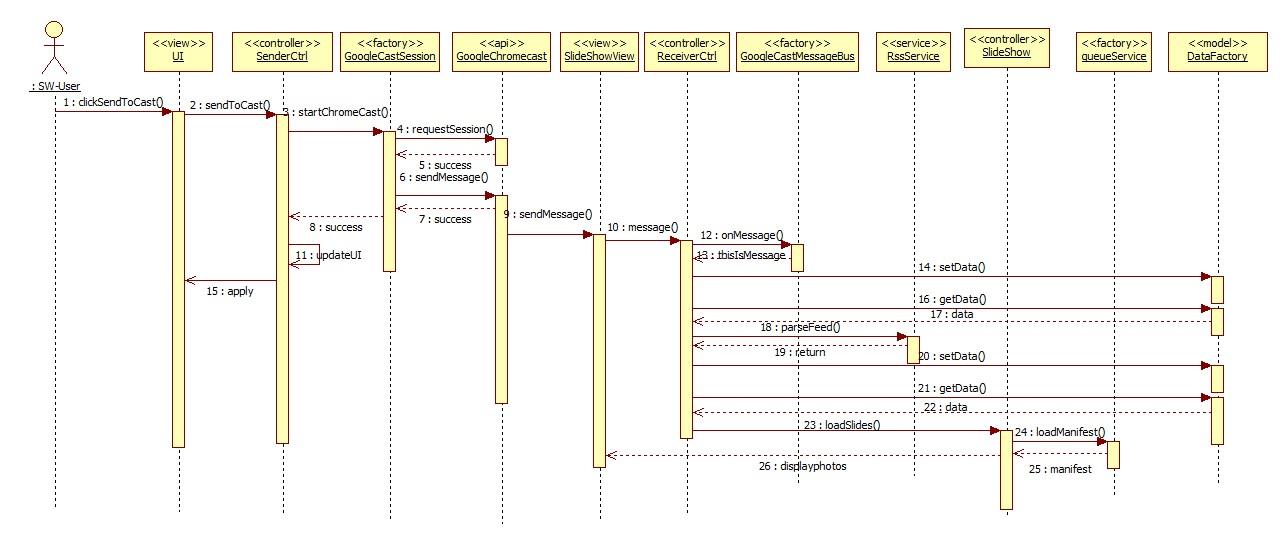
* **Receiver Subsystem**
  + Framework: AngularJS, HTML5
  + Architecture: MVC
  + The receiver subsystem is a custom chrome cast receiver application in charge of handling the sender subsystems requests, working alongside the slideshow engine subsystem, and displaying content onto chrome cast devices. The receiver subsystem receives the data input sent by the sender application and then requests data from the data retrieval subsystem. The receiver subsystem is a fully custom web application that has the capability of being deployed onto chrome cast devices. Receiver subsystem is in charge of handling the data sent by sender applications, as well as interacting with the other subsystems in order to generate the correct outcome, depending on what the user input.

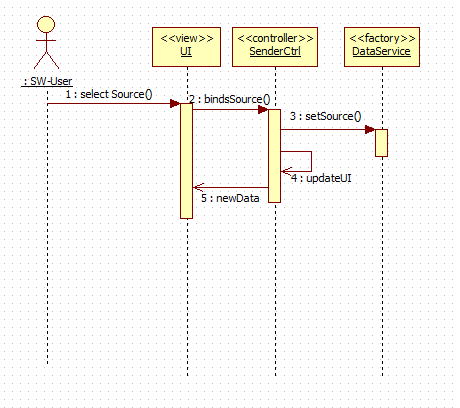
**6.3. Dynamic Model**

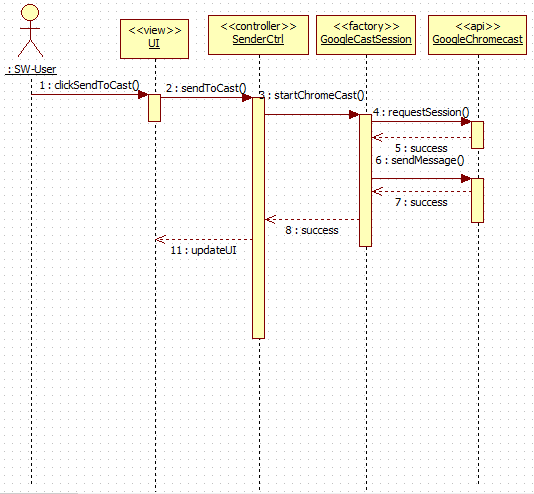
**Sequence Diagram of S-WALL/1-001/Generate Slideshow**

**Sequence Diagram of S-WALL/1-002/Image Cycle Speed**

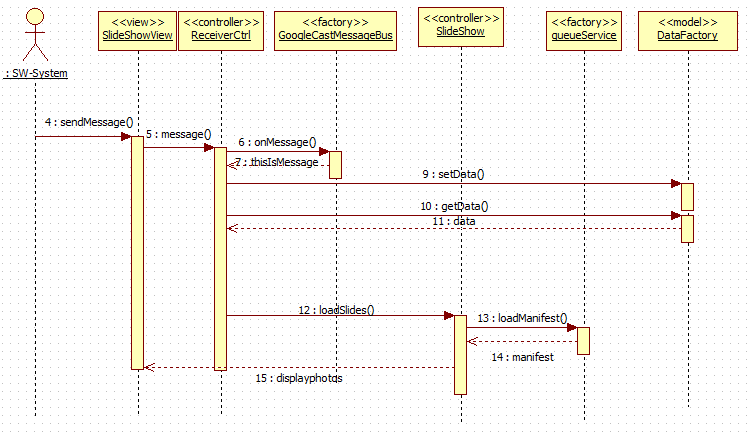


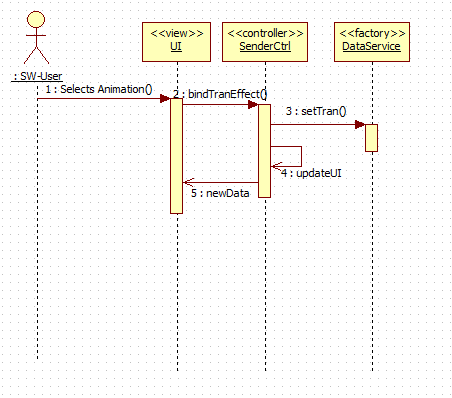
**Sequence Diagram of S-WALL/1-008/Digest RSS feed**

**Sequence Diagram of S-WALL/1-009/Select Photo Source**

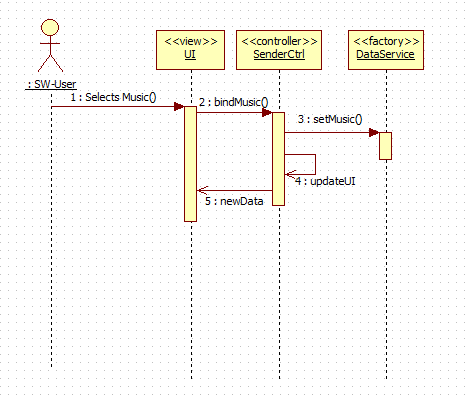


**Sequence Diagram of S-WALL/1-011/Handle Cast Request**

**Sequence Diagram of S-WALL/1-011/Handle Cast Request**



**Sequence Diagram of** **S-WALL/1-012/Change Effect**



**Sequence Diagram of S-WALL/1-013/Add Music**

**6.4. Code Specification**

//Filename: Receiver.js

//Author@JosephG

angular.module('socialWall-receiver')

.controller('ReceiverCtrl', ['$scope', '$timeout', '$http', 'GoogleCastMessageBus', 'DataService', 'rssService',

function ($scope, $timeout, $http, GoogleCastMessageBus, DataService, rssService) {

$scope.source = 'RSSFeed';

$scope.URL = "http://tinyurl.com/nynogx3";

$scope.message = '';

//On message received, perform function, e contains data.

GoogleCastMessageBus.onMessage = function (e) {}

//Load Feed

function loadFeed(){}

//Filename: sshow.js

(function () {

'use strict';

angular.module('socialWall-receiver')

.controller('SlideShowCtrl', function($scope, $timeout, $window, QueueService, DataService) {}

var IINTERVAL;

var slides;

//Sets currentslide index, tracker

function setCurrentSlideIndex(index){}

//Checks too see whether current slide is last

function isCurrentSlideIndex(index)

//Show next slide/photo

function nextSlide(){}

//Set animation from setting

function setCurentAnimation(animation){}

//if animation is current, contineu

function isCurrentAnimation(animation){}

progres;

loaded;

currentIndex;

currentAnimation;

//

//Filename: DataFactory.js

//Author: JosephG

//The model of the architecture, Contains data shared across controllers.

var photos;

var source;

var URL;

var message;

var transitionSpeed;

var transition;

var photosSet;

this.arePhotosSet = function() {}

this.setPhotosSet = function(newSet) {}

this.getSource = function() {}

this.getPhotos = function(){}

this.getMessage = function(){}

this.setPhotos = function(newPhoto){}

this.getTransitionSpeed = function(){}

this.setTransitionSpeed = function(newSpeed){}

this.getURL = function(){}

this.setURL = function(newURL){}

this.getTransition = function(){}

this.setTransition - function(newTran){}

//

//Filename: googlecastmessagebus.js

//Author: JosephG

//Handles chrome cast on network connection

angular.module('socialWall-receiver')

.factory('GoogleCastMessageBus', ['MESSAGE\_NAMESPACE', function (MESSAGE\_NAMESPACE) {

// Initialize the chromecast

cast.receiver.logger.setLevelValue(0);

var castReceiverManager ;

// Handle the 'Ready' event

castReceiverManager.onReady = function (e) {};

// Handle the 'SenderConnected' event

castReceiverManager.onSenderConnected = function (e) {};

// Handle the 'SenderDisconnected' event

castReceiverManager.onSenderDisconnected = function (e) {};

// Create the Cast Message Bus to handle messages for the custom namespace

var castMessageBus;

// Initialize the CastReceiverManager with a base application status

castReceiverManager.start(();

//Filename: rssService.js

//Author: JosephG

//Uses google api to retrieve photos

(function () {

'use strict';

angular.module('socialWall-receiver')

.factory('rssService',['$http',function($http){

return {

parseFeed : function(url){

return $http.jsonp('//ajax.googleapis.com/ajax/services/feed/load?v=1.0&num=100&callback=JSON\_CALLBACK&q=' + encodeURIComponent(url));

}

}

}]);

})();

//Filename: queueService.js

//Author: Juan G.

angular.module('socialWall-receiver')

.factory('QueueService', function($rootScope){

var queue

function loadManifest(manifest) {}

queue.on('progress', function(event) {}

queue.on('complete', function() {}

//Filename: main.js

//Author: JosephG

//Receiver Control for managing the custom receiver applicaiton and receing

//messages from the sender applications.

angular.module('socialWall-sender')

.controller('MainCtrl', ['$rootScope', '$scope', 'GoogleCastSession', function ($rootScope, $scope, GoogleCastSession) {

$scope.source

$scope.URL

$scope.animation

$scope.transitionSpeed

$scope.disabled

$scope.stat

// Listen to the different states

$rootScope.$on('INITIALIZING\_CAST\_API', function () {}

});

$rootScope.$on('RECEIVER\_AVAILABLE', function () {}

});

$rootScope.$on('RECEIVER\_DEAD', function () {}

// Send the data to the chrome cast

$scope.sendToCast = function () {}

//Filename: googlecastmessagebus.js

//Author: JosephG

//factory used by receiver application to handle requests made by chromecast

//returns a case message session

//messages from the sender applications.

angular.module('socialWall-sender')

.factory('GoogleCastSession', ['$rootScope', 'CAST\_APP\_ID', 'MESSAGE\_NAMESPACE', function ($rootScope, CAST\_APP\_ID, MESSAGE\_NAMESPACE) {

var castSession = null;

// Timeout to initialize the API

if (!chrome.cast || !chrome.cast.isAvailable) {}

// Initialize the Google Cast API for use

function initializeCastApi() {}

function sessionListener(e) {}

function onInitSuccess() {}

function onSuccess(message) {}

function onError(message) {}

function receiverMessage(namespace, message) {}

function receiverListener(e) {}

function sessionUpdateListener(isAlive) {}

return

}

}]);

})();

**7. System Validation**

In this chapter we discuss our testing methods and evaluate the results of our testing. Section one describes the subsystem testing we did through a series of test cases. In the following section we provide a description of the system testing also through test cases. In final section we evaluate the results of our testing.

**7.1. Subsystem Tests**

Due to time constraints, testing of the four subsystems was cut from the deliverable.

**7.2. System Tests**

**S-WALL/TC-001/Retrieve RSS Source Sunny Day**

**Tested Use Case:** SW/1-008/Digest RSS Feed

**Purpose:**

Ensure that the source is retrieved properly and placed into an image array for the Slideshow engine to use.

**Test Setup Environment:**

Set up a basic RSS feed with multiple different images of different sizes and resolutions. Test run on Eclipse IDE, which ran on a localhost, displayed on Google Chrome browser.

**Test Input(s):**

* Valid RSS image feed link: <http://bit.ly/1slNCq1>

**Expected Output(s):**

An array of images that are displayed in the order in which they appear on the RSS Feed.

**Actual Output(s):**

An array showing images from the RSS feed provided, in order of how they appear on the RSS feed. **(TEST PASSED)**

**S-WALL/TC-002/Retrieve RSS Source Rainy Day**

**Tested Use Case:** SW/1-008/Digest RSS Feed

**Purpose:**

Ensure that when an improper URL is input into the RSS feed retrieval field, that the system knows how to handle the error.

**Test Setup Environment:**

Provide a link that is not an RSS feed and does not include any images, preferably a fake URL. Test run on Eclipse IDE, which ran on a localhost, displayed on Google Chrome browser.

**Test Input(s):**

* Invalid RSS image feed link: [http://fakeURLhere.edu/](http://fakeurlhere.edu/)

**Expected Output(s):**

An error message saying that image album could not be retrieved from the source provided, instructing users to try another link.

**Actual Output(s):**

App displays the following error: Error: Incorrect URL. **(TEST PASSED)**

**S-WALL/TC-003/Display Slideshow Sunny Day**

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

**Purpose:**

Ensure that the slideshow engine displays as expected given a proper input (transition speed, transition animation, photo album source), and that it does quickly.

**Test Setup Environment:**

Set up a basic RSS feed with multiple different images of different sizes and resolutions. Test run on Eclipse IDE, which ran on a localhost, displayed on Google Chrome browser.

**Test Input(s):**

* Valid RSS image feed link: <http://bit.ly/1slNCq1>
* Transition effect: Fade In
* Transition time: 3 seconds

**Expected Output(s):**

A slideshow that displays all images, in order that they appear, from the RSS photo album, that fades in after every three seconds.

**Actual Output(s):**

As expected, output is a slideshow that displays all images, in order that they appear, from the RSS photo album, that fades in after every three seconds. **(TEST PASSED)**

**S-WALL/TC-004/Display Slideshow Rainy Day**

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

**Purpose:**

Ensure that the slideshow engine reports an error if certain required settings aren’t properly input into their respective fields.

**Test Setup Environment:**

Set up a basic RSS feed with multiple different images of different sizes and resolutions. Test run on Eclipse IDE, which ran on a localhost, displayed on Google Chrome browser.

**Test Input(s):**

* Valid RSS image feed link: <http://bit.ly/1slNCq1>
* Transition effect: Fade In
* Transition time: (Blank)

**Expected Output(s):**

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

**Actual Output(s):**

Slideshow displays without displaying an error of any sorts. System defaults to a transition time of three seconds **(TEST FAILED)**

**7.3. Evaluation Tests**

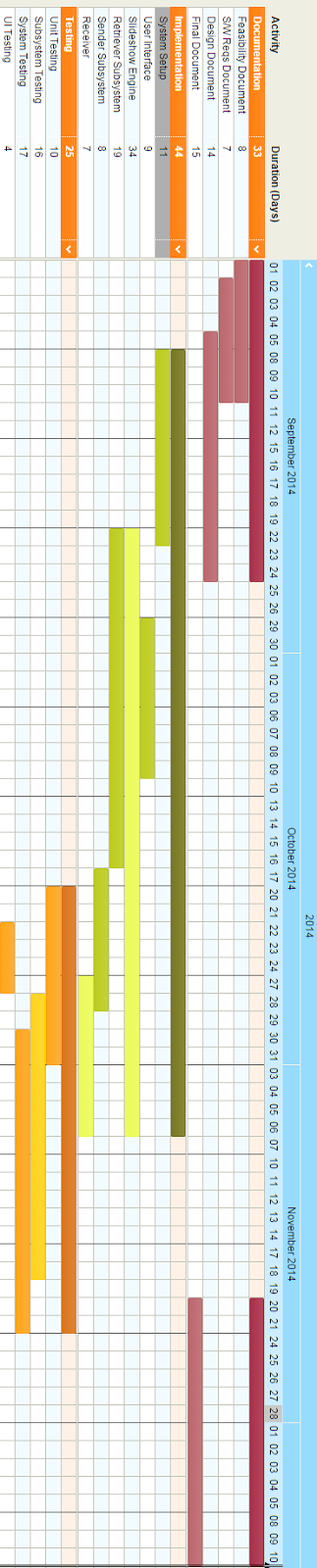
These tests have shown that our SW system, while not completely refined, functions as expected. All non-functional requirements are met, giving the system a quick and efficient run time. Most functional requirements were met and satisfied testing. Some, however, such as the retrieval of DropBox and Facebook photo albums were not completely finished due to time constraints.

**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/1-001/Generate Slideshow

**Use Case Level:** High Level

**Details:**

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

1. User must be on web app.

2. User must have selected source of images.

* **Description:**

1. The use case begins when the user clicks the "Generate Slideshow” button.

2. The system shall check if user has entered all necessary input.

3. The system shall display the images in a slideshow format.

4. The use case ends when the system presents the user with the slideshow of images.

* **Postconditions:**

1. The user shall be presented with with a slideshow of images according to the settings input by the user.

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

**Exceptions:**

* The system is unable to generate the slideshow.
* The system is unable to display images.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used almost everytime software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** High. Dependent upon external entities.

**Constraints:**

* System shall present user with slideshow within 5 seconds.

**Modification History:**

* **Owner:**
* **Initiation Date:** September 8, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-002/Image Cycle Speed

**Use Case Level:** High Level

**Details:**

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

1. User must be on web application.

2. User must be in slideshow settings.

* **Description:**

1. The use case begins when the user enters data into the “Image Cycle Speed” setting.

2. The system shall maintain user input as image cycle speed.

3. The use case ends when the system records the users input and displays the changed number.

* **Postconditions:**

1. The user can purchase the item from an external site or can simply close the window.

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

**Exceptions:**

* The system is unable to change image cycle speed.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Medium- High Frequency. Setting utilized at least once every application use.
* **Criticality:** Medium-High. Most users may utilize feature, but it is not the core feature of the site.
* **Risk:** Low.

**Constraints:**

* The system must change the image cycle speed in under .2 seconds.

**Modification History:**

* **Owner:**
* **Initiation Date:** September 8, 2014
* **Last Modified:** September 9, 2014

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

**Use Case ID:** S-WALL/1-006/Digest Facebook Page Album

**Use Case Level:** High Level

**Details:**

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

1. User must be on web application.

* **Description:**

1. The use case begins when the user clicks “photo source” button.

2. The system shall present the user with the different source options available.

4. The user shall select the “facebook page” option from the list of optoins

5. The system shall present the user with an input box.

6. User shall enter the name of the facebook page.

5. User shall click on the get photos button.

6. The system shall present the user with the albums available.

7. The use case ends when the user selects the album to digest.

* **Postconditions:**

1. User has set album from facebook page to be used in slideshow engine.

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

**Exceptions:**

* The system is unable to access source.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** High. Needed for core functionality of system.
* **Risk:** Medium.

**Constraints:**

**Modification History:**

* **Owner:**
* **Initiation Date:** September 8, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-008/Digest RSS Feed

**Use Case Level:** High Level

**Details:**

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

1. User must be on web application.

* **Description:**

1. The use case begins when the user clicks “photo source” button.

2. The system shall present the user with the different source options available.

4. The user shall select the “RSS Feed” option from the list of options

5. The system shall present the user with an input box.

6. User shall enter the RSS Feed information

5. User shall click on the “next” button.

6. Use case ends when user has set RSS Feed information and has proceeded.

* **Postconditions:**

1. User has set up the RSS Feed link information and system is ready to digest photos.

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

**Exceptions:**

* The system is unable to access source.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** High. Needed for core functionality of system.
* **Risk:** Medium.

**Constraints:**

**Modification History:**

* **Owner:**
* **Initiation Date:** September 8, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-009/Select Photo Source

**Use Case Level:** High Level

**Details:**

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

1. User must be on web application.

* **Description:**

1. The use case begins when the user clicks “photo source” button.

2. The system shall present the user with the different source options available.

4. The user shall select an option from those listed to use as the source.

5. The system shall present the user with the function of accessing source.

6. User shall complete source process.

5. The use case ends when the system shows the user that the source option that has been chosen is active.

* **Postconditions:**

1. Source of images is that which user has chosen.

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

**Exceptions:**

* The system is unable to access source.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** High. Needed for core functionality of system.
* **Risk:** Medium.

**Constraints:**

**Modification History:**

* **Owner:**
* **Initiation Date:** October 5, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-010/Cast to Chrome Cast

**Use Case Level:** High Level

**Details:**

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

1. User is utilizing mobile application.
2. User has setup source and settings.

* **Description:**

1. The use case begins when the user touches the “Cast” icon.

2. The system shall present the user with devices found on network.

3. The user shall select the Chrome Cast device wanted.

4. The system shall send a request to the server with users settings.

5. The system shall capture request and send the corresponding cast to device.

6. The use case ends when the device utilizing the Chrome Cast is displaying the correct photo feed.

* **Postconditions:**

1. Device running Chrome Cast is displaying photo feed.

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

**Exceptions:**

* The system is unable to access source.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** High. Needed for core functionality of system.
* **Risk:** Medium.

**Constraints:**

**Modification History:**

* **Owner:**
* **Initiation Date:** October 5, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-011/Handle Cast Request

**Use Case Level:** High Level

**Details:**

* **Actor:** Server
* **Preconditions:**

1. User has sent a Chrome Cast request to the server.

* **Description:**

1. The use case begins when the system receives a Chrome Cast request

2. The system shall handle corresponding request adequately.

3. The system shall then handle request, and cast the photo feed with the users settings onto intended device.

**Postconditions:**

1. Source of images is that which user has chosen.

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

**Exceptions:**

* The system is unable to access source.
* The system cannot connect to web.

**Related Use Cases:**

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** High. Needed for core functionality of system.
* **Risk:** Medium.

**Constraints:**

**Modification History:**

* **Owner:**
* **Initiation Date:** October 5, 2014
* **Last Modified:** October 5, 2014

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

**Use Case ID:** S-WALL/1-012/Change Effect

**Use Case Level:** High Level

**Details:**

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

1. User has selected a source for his/her slideshow

2. User is on slideshow settings page

* **Description:**

1. The use case begins when user clicks drop-down menu for effect

2. The system shall give them a list of transition effects to choose from

3. The user selects the transition effect they desire from the list

4. The use case ends records the user’s input and displays the name of the selected effect.

**Postconditions:**

1. Transition effect is displayed when user generates slideshow.

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

**Exceptions:**

* The system cannot connect to web.

**Related Use Cases:** S-WALL/1-001/Generate Slideshow

**Decision Support:**

* **Frequency:** High Frequency.
* **Criticality:** Medium. Feature desired for system.
* **Risk:** Low.

**Constraints:**

* Change to transition effect setting must take less than 0.2 seconds to apply.

**Modification History:**

* **Owner:** Juan Gonzalez-Llanos
* **Initiation Date:** October 5, 2014
* **Last Modified:** October 15, 2014

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

**Use Case ID:** S-WALL/1-013/Add Music

**Use Case Level:** High Level

**Details:**

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

1. User must be on web application.

2. User must be in slideshow settings.

* **Description:**

1. The use case begins when the user enters a YouTube URL in the “Add Background Music” field.

2. The system records the users input and displays the URL.

3. The use case ends when the user clicks “Generate Slideshow”

* **Postconditions:**

1. System shall play the youtube audio during the slideshow

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

**Exceptions:**

* The system is unable retrieve YouTube video
* YouTube video does not allow embedding
* System cannot connect to web

**Related Use Cases:** SW/1-001/Generate Slideshow

**Decision Support:**

* **Frequency:** Medium- High Frequency. Setting utilized at least once every application use.
* **Criticality:** Low. Most users may utilize feature, but it is not the core feature of the app.
* **Risk:** Low.

**Constraints:**

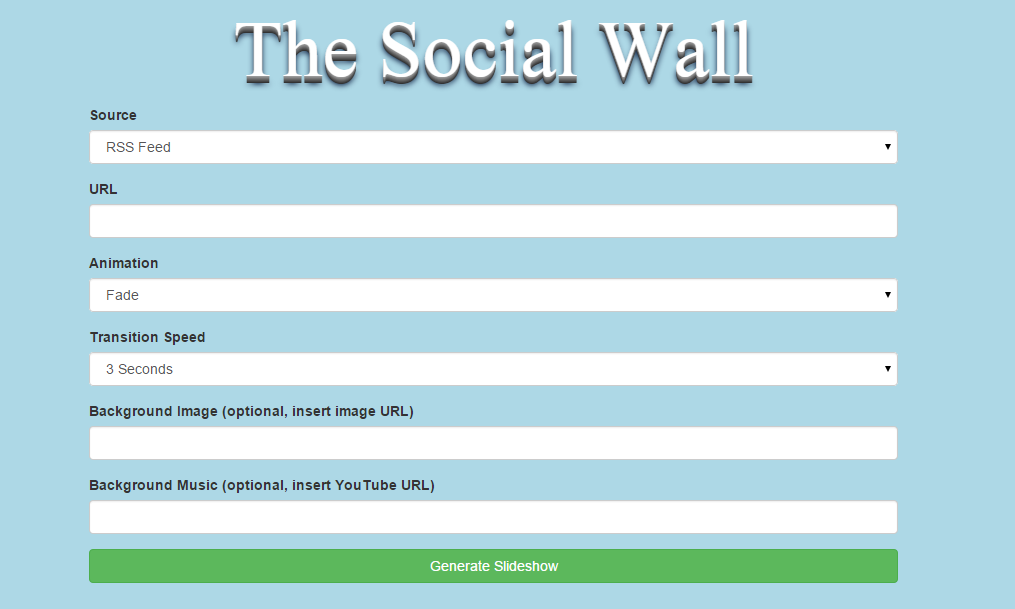
* The system must play audio in under .2 seconds.

**Modification History:**

* **Owner:**
* **Initiation Date:** October 17, 2014
* **Last Modified:** October 24, 2014

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

**9.3 Appendix C - User Interface Designs**

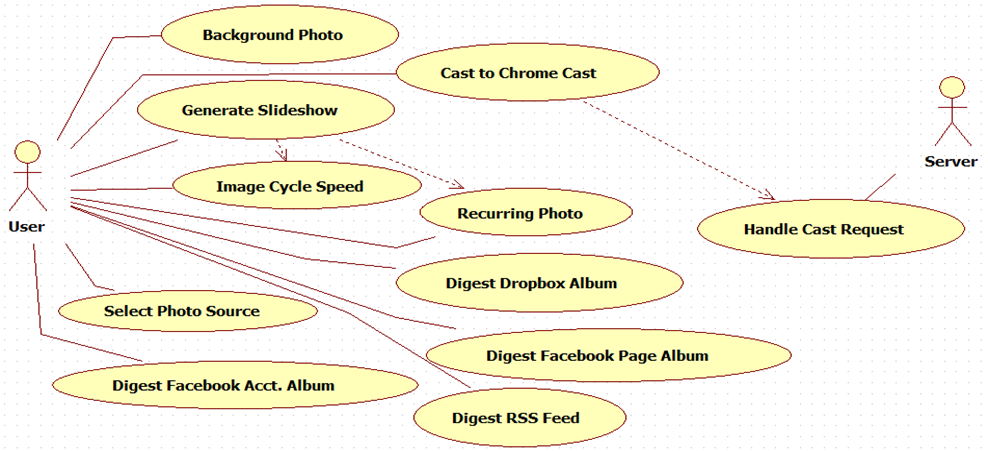


UI for Social Wall Web Application

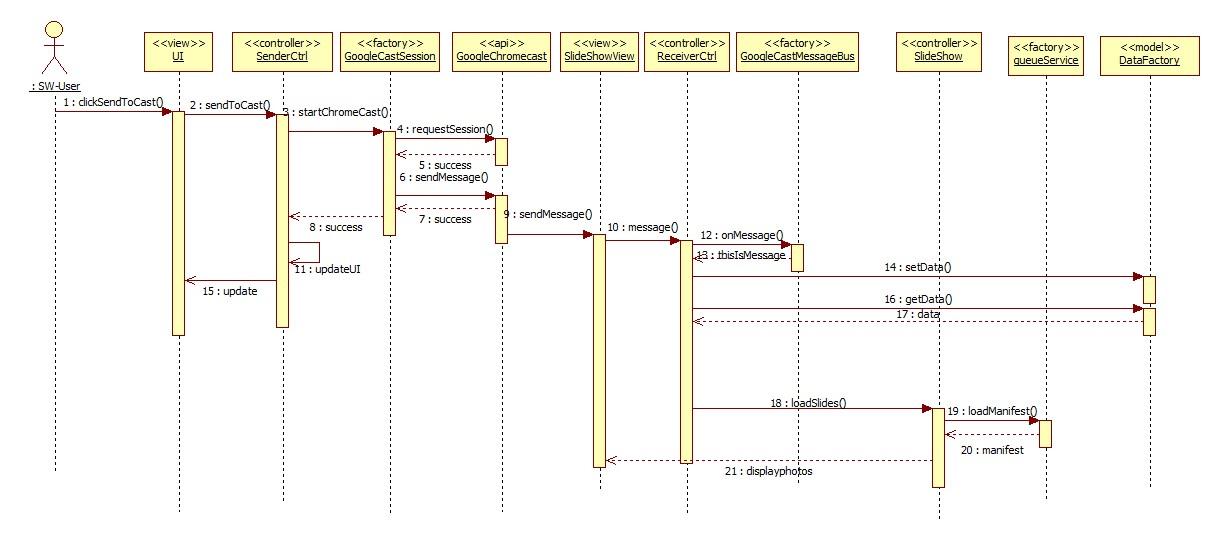


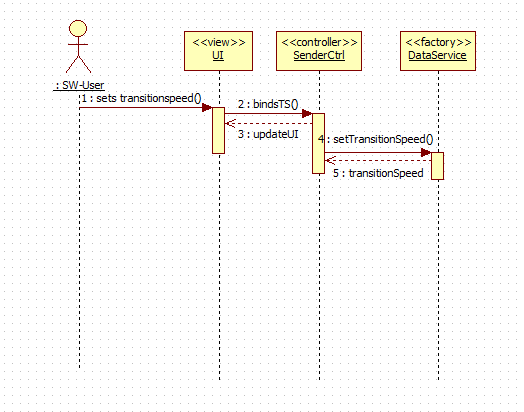
UI for Social Wall to send to Chrome Cast

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

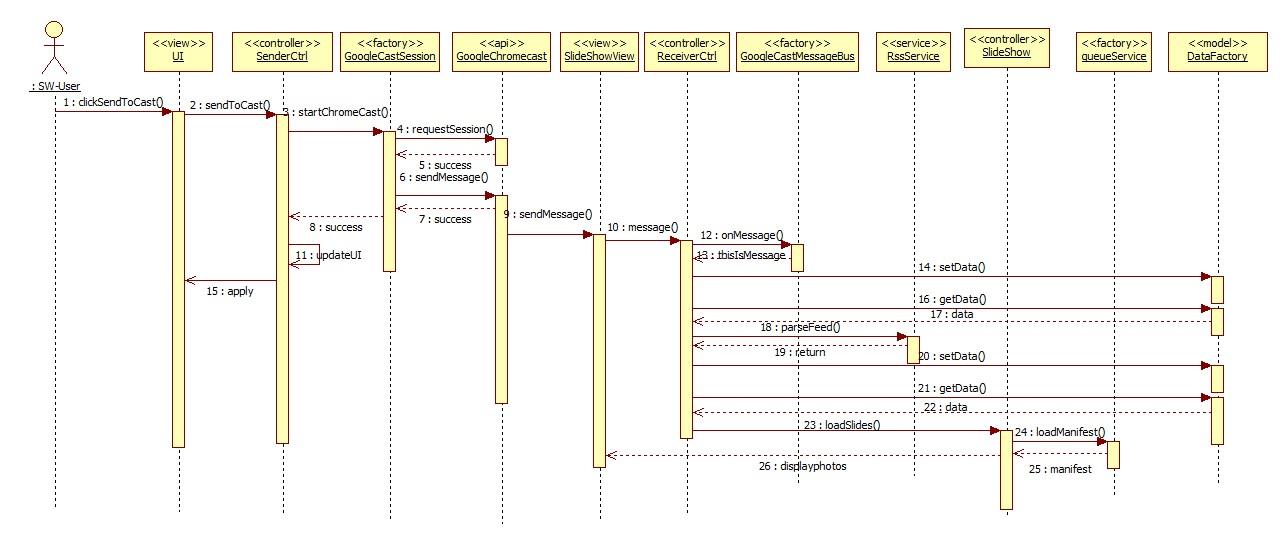


**Sequence Diagram S-WALL/1-001/Generate Slideshow.**

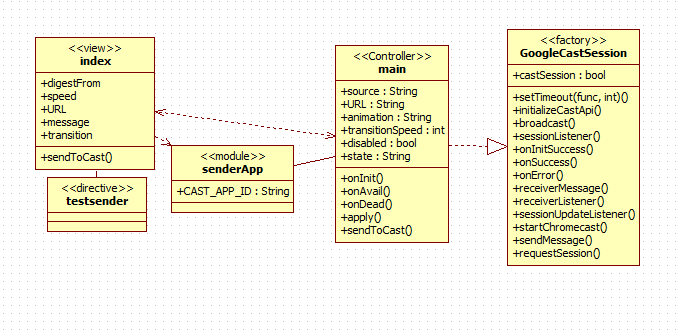


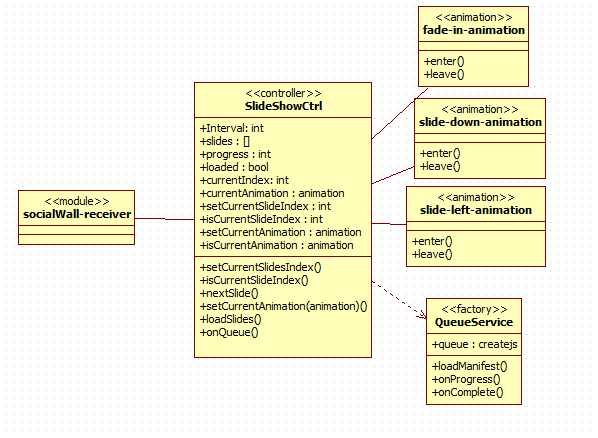
**Sequence Diagram S-WALL/1-002/Image Cycle Speed**

**Sequence Diagram-WALL/1-008/Digest RSS Feed**

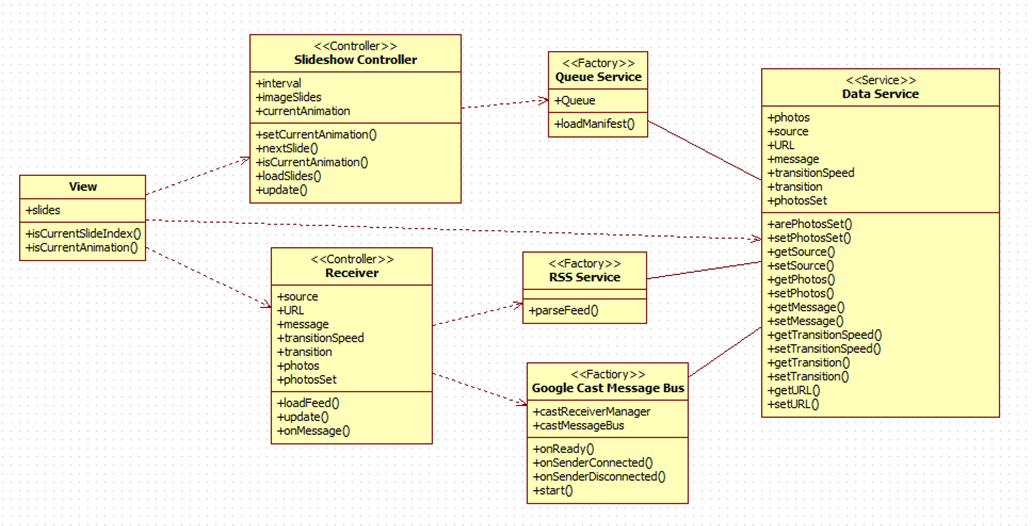
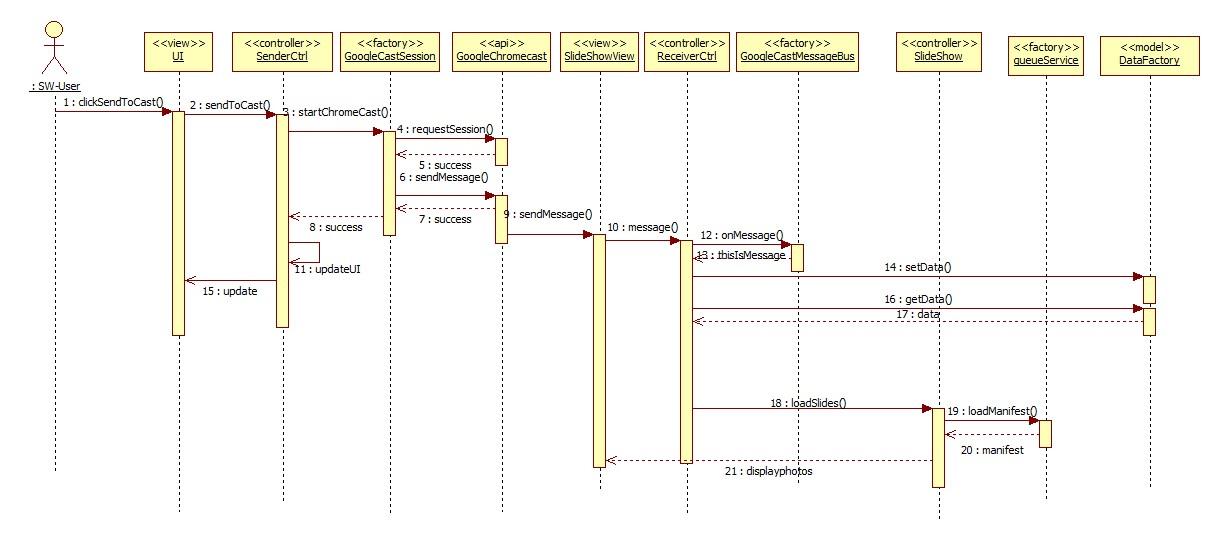


**9.5 Appendix E - Design Models (Static and Dynamic)**

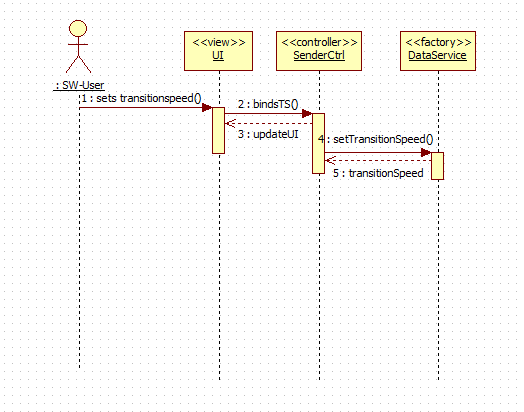


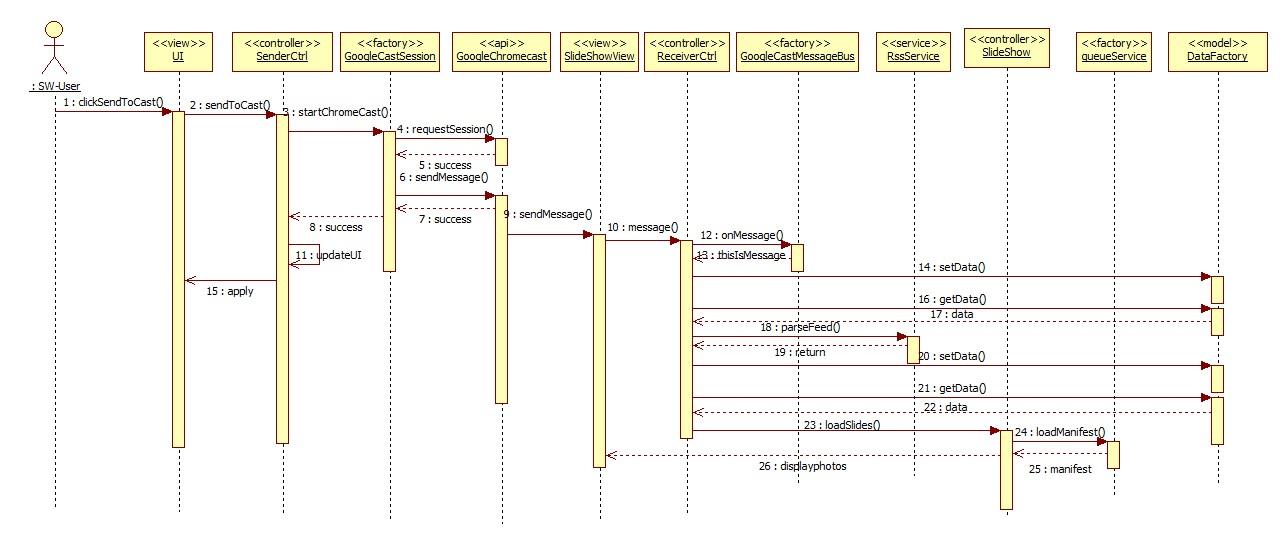
**Sender Application Class Diagram**

**Slideshow Engine Class Diagram**

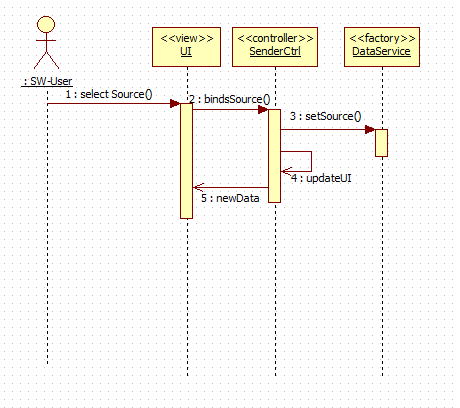
**Class diagram of Custom Receiver Subsystem**

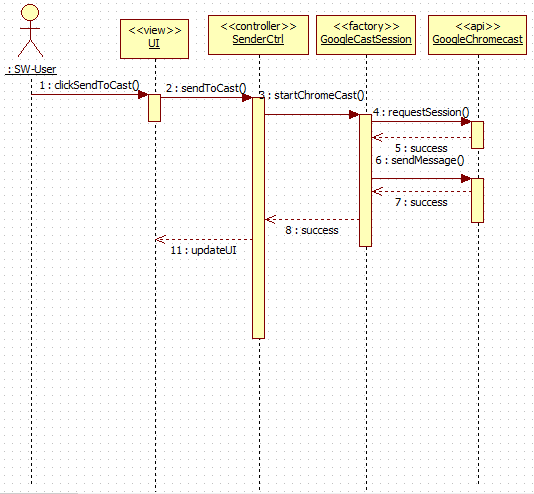
**Sequence Diagram of S-WALL/1-001/Generate Slideshow**

**Sequence Diagram of S-WALL/1-002/Image Cycle Speed**

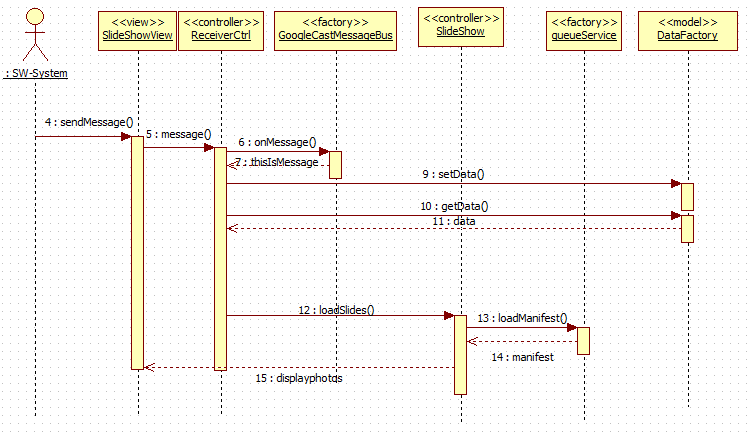


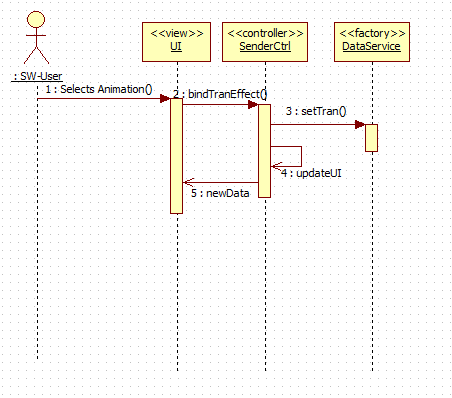
**Sequence Diagram of S-WALL/1-008/Digest RSS feed**

**Sequence Diagram of S-WALL/1-009/Select Photo Source**

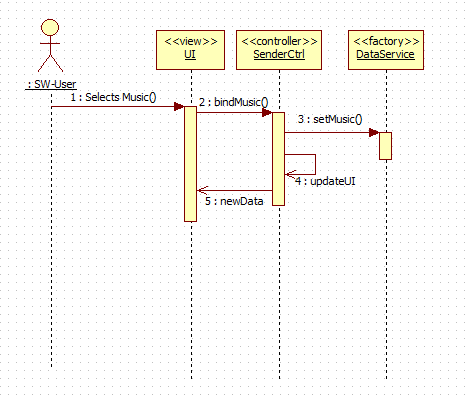


**Sequence Diagram of S-WALL/1-011/Handle Cast Request**

**Sequence Diagram of S-WALL/1-011/Handle Cast Request**



**Sequence Diagram of** **S-WALL/1-012/Change Effect**



**Sequence Diagram of S-WALL/1-013/Add Music**

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

//Filename: Receiver.js

//Author@JosephG

angular.module('socialWall-receiver')

.controller('ReceiverCtrl', ['$scope', '$timeout', '$http', 'GoogleCastMessageBus', 'DataService', 'rssService',

function ($scope, $timeout, $http, GoogleCastMessageBus, DataService, rssService) {

$scope.source = 'RSSFeed';

$scope.URL = "http://tinyurl.com/nynogx3";

$scope.message = '';

//On message received, perform function, e contains data.

GoogleCastMessageBus.onMessage = function (e) {}

//Load Feed

function loadFeed(){}

//Filename: sshow.js

(function () {

'use strict';

angular.module('socialWall-receiver')

.controller('SlideShowCtrl', function($scope, $timeout, $window, QueueService, DataService) {}

var IINTERVAL;

var slides;

//Sets currentslide index, tracker

function setCurrentSlideIndex(index){}

//Checks too see whether current slide is last

function isCurrentSlideIndex(index)

//Show next slide/photo

function nextSlide(){}

//Set animation from setting

function setCurentAnimation(animation){}

//if animation is current, contineu

function isCurrentAnimation(animation){}

progres;

loaded;

currentIndex;

currentAnimation;

//

//Filename: DataFactory.js

//Author: JosephG

//The model of the architecture, Contains data shared across controllers.

var photos;

var source;

var URL;

var message;

var transitionSpeed;

var transition;

var photosSet;

this.arePhotosSet = function() {}

this.setPhotosSet = function(newSet) {}

this.getSource = function() {}

this.getPhotos = function(){}

this.getMessage = function(){}

this.setPhotos = function(newPhoto){}

this.getTransitionSpeed = function(){}

this.setTransitionSpeed = function(newSpeed){}

this.getURL = function(){}

this.setURL = function(newURL){}

this.getTransition = function(){}

this.setTransition - function(newTran){}

//

//Filename: googlecastmessagebus.js

//Author: JosephG

//Handles chrome cast on network connection

angular.module('socialWall-receiver')

.factory('GoogleCastMessageBus', ['MESSAGE\_NAMESPACE', function (MESSAGE\_NAMESPACE) {

// Initialize the chromecast

cast.receiver.logger.setLevelValue(0);

var castReceiverManager ;

// Handle the 'Ready' event

castReceiverManager.onReady = function (e) {};

// Handle the 'SenderConnected' event

castReceiverManager.onSenderConnected = function (e) {};

// Handle the 'SenderDisconnected' event

castReceiverManager.onSenderDisconnected = function (e) {};

// Create the Cast Message Bus to handle messages for the custom namespace

var castMessageBus;

// Initialize the CastReceiverManager with a base application status

castReceiverManager.start(();

//Filename: rssService.js

//Author: JosephG

//Uses google api to retrieve photos

(function () {

'use strict';

angular.module('socialWall-receiver')

.factory('rssService',['$http',function($http){

return {

parseFeed : function(url){

return $http.jsonp('//ajax.googleapis.com/ajax/services/feed/load?v=1.0&num=100&callback=JSON\_CALLBACK&q=' + encodeURIComponent(url));

}

}

}]);

})();

//Filename: queueService.js

//Author: Juan G.

angular.module('socialWall-receiver')

.factory('QueueService', function($rootScope){

var queue

function loadManifest(manifest) {}

queue.on('progress', function(event) {}

queue.on('complete', function() {}

//Filename: main.js

//Author: JosephG

//Receiver Control for managing the custom receiver applicaiton and receing

//messages from the sender applications.

angular.module('socialWall-sender')

.controller('MainCtrl', ['$rootScope', '$scope', 'GoogleCastSession', function ($rootScope, $scope, GoogleCastSession) {

$scope.source

$scope.URL

$scope.animation

$scope.transitionSpeed

$scope.disabled

$scope.stat

// Listen to the different states

$rootScope.$on('INITIALIZING\_CAST\_API', function () {}

});

$rootScope.$on('RECEIVER\_AVAILABLE', function () {}

});

$rootScope.$on('RECEIVER\_DEAD', function () {}

// Send the data to the chrome cast

$scope.sendToCast = function () {}

//Filename: googlecastmessagebus.js

//Author: JosephG

//factory used by receiver application to handle requests made by chromecast

//returns a case message session

//messages from the sender applications.

angular.module('socialWall-sender')

.factory('GoogleCastSession', ['$rootScope', 'CAST\_APP\_ID', 'MESSAGE\_NAMESPACE', function ($rootScope, CAST\_APP\_ID, MESSAGE\_NAMESPACE) {

var castSession = null;

// Timeout to initialize the API

if (!chrome.cast || !chrome.cast.isAvailable) {}

// Initialize the Google Cast API for use

function initializeCastApi() {}

function sessionListener(e) {}

function onInitSuccess() {}

function onSuccess(message) {}

function onError(message) {}

function receiverMessage(namespace, message) {}

function receiverListener(e) {}

function sessionUpdateListener(isAlive) {}

return

}

}]);

})();

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

Could not perform testing for drivers and stubs due to time constraints.

**9.8 Appendix H - Diary of Meetings**

Diary Entry 1:

Date: September 4, 2014

Location: Picture Marketing Offices

Start time: 2:30 pm

End time: 5:00 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills

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 ourselves to Cortney and the Picture Management. Discussed the Social Wall project, what is needed and what is expected.

Diary Entry 2:

Date: September 8, 2014

Location: Virtual meeting via Skype

Start time: 7:30 pm

End time: 8:40 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez

Late: N/A

Agenda:

* Talk about document drafts due
* Discussed use cases
* Come up with requirements and constraints

Summary of Discussion:

Discussed and worked on drafts that are due on September 8th and we came up with the use cases.

Diary Entry 3:

Date: September 19, 2014

Location: Picture Marketing Offices

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Discuss Development Approach
* Create Schedule
* Discuss requirements/finalize

Summary of Discussion:

Louis would become a mentor as well of the project. Had meeting to discuss how to approach the project and finalize requirement elicitation. Decided upon an android application, chrome cast compatible, as well as a web application. Discussed tools needed as well as framework that will be utilized being angularjs. Joseph would be in charge of android application, Juan in charge of Slide show engine and web app.

Diary Entry 4:

Date: September 26, 2014

Location: Picture Marketing Offices

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Show Facebook demo
* Discuss development approach

Summary of Discussion:

Discussion and acceptance of major work must be done on core parts of system. Must get android application running phonegap and slideshow engine must begin development.

Diary Entry 5:

Date: October 3, 2014

Location: Picture Marketing Offices

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Discuss phonegap project
* Slideshow engine

Summary of Discussion:

Phonegap project discussed in detail. Along with getting application running on emulator. See benefits of angular and begin approach to develop using framework. Slideshow engine displayed, must be converted to angularjs.

Diary Entry 6:

Date: October 22, 2014

Location: FIU ECS Computer lab

Start time: 1:50 pm

End time: 3:00 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez

Late: N/A

Agenda:

* Discuss progress
* Slideshow engine demo

Summary of Discussion:

We spoke about our recent progress and breakthroughs. Juan spoke of his improvements to the slideshow engine, and Joseph spoke about his improvement of the mobile app.

Diary Entry 7:

Date: October 28, 2014

Location: Picture Marketing Offices

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Discuss phonegap project
* Slideshow engine

Summary of Discussion:

Phonegap project discussed in detail. Along with getting application running on emulator. See benefits of angular and begin approach to develop using framework. Slideshow engine displayed, must be converted to angularjs.

Diary Entry 8:

Date: November 2, 2014

Location: FIU SCS Computer Lab

Start time: 2:00 pm

End time: 4:00 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez

Late: N/A

Agenda:

* Discuss progress
* Develop UML diagrams

Summary of Discussion:

We discussed our work up until that point in time, displaying our progress in our respective portion of the project. Afterwards we began developing UML diagrams, mostly sequence diagrams, based on our progress so far.

Diary Entry 9:

Date: November 11, 2014

Location: Picture Marketing

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Discuss progress
* Discuss Chrome Cast
* Discuss Slideshow engine

Summary of Discussion:

Chrome Cast was the main focus of this meeting. Picture Marketing lent us their Chrome Cast so that we could begin developing the app so it could utilize it. Afterwards, we discussed the slideshow engine, and getting it working using the AngularJS framework.

Diary Entry 10:

Date: November 24, 2014

Location: FIU SCS Computer Lab

Start time: 1:50 pm

End time: 2:50 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez

Late: N/A

Agenda:

* Discuss progress
* Update documentation
* Discuss direction of project

Summary of Discussion:

Like all our meetings, we discussed how far we’ve done with our respective portions of the project. We updated our documentation accordingly. Once completed, we discussed the direction of our portions so we could schedule when we would be able merge our sections to finish the Social Wall app.

Diary Entry 11:

Date: November 29, 2014

Location: Skype (virtual meeting)

Start time: 5:30 pm

End time: 8:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph GonzalezLate: N/A

Agenda:

* Assemble final Social Wall app

Summary of Discussion:

We used this meeting to assemble both portions of the Social Wall app. We were able to get our application functioning as expected, with only minor bugs.

Diary Entry 12:

Date: December 4, 2014

Location: Picture Marketing

Start time: 2:30 pm

End time: 4:30 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Display our full demo
* Discuss current bugs
* Discuss shortcomings of the project

Summary of Discussion:

At this meeting, we were able to display our full demo to our mentors at Picture Marketing. After the demo, we highlighted our bugs, as well as the features we weren’t able to accomplish due to time restraints.

Diary Entry 13:

Date: December 10, 2014

Location: Picture Marketing

Start time: 1:30 pm

End time: 6:00 pm

In Attendance: Juan Gonzalez-Llanos, Joseph Gonzalez, Cortney Mills, Louis Zuckerman

Late: N/A

Agenda:

* Display demo with bug fixes
* Record videos
* Finish documentation

Summary of Discussion:

For this meeting, we put the final touches on the Social Wall project. We had already cleaned up most of the bugs from the past week, and showed our mentors a more complete Social Wall application. We used the remaining time to record our instruction videos and to finish documentation.

**10. References**

1. <http://angularjs.org/> - AngularJS Framework for Javascript
2. <http://ionicframework.com/> - Ionic Framework for developing HTML mobile apps
3. <http://phonegap.com/> - Porting web app to Android
4. <http://cordova.apache.org/> - Porting web app to Android
5. <http://nodejs.org/> - Porting web app to Android
6. <http://getbootstrap.com/> - Bootstrap Framework
7. <http://bower.io/> - Package manager for web app
8. <http://www.techsmith.com/camtasia.html/> - Screen Recorder for videos
9. <http://newegg.com/> - Prices on h/w and s/w
10. <http://photoshop.com/> - Photoshop CC