CIS 4911 – SENIOR PROJECT

Picture Marketing’s Social Wall

**Design Document**

Members: Steve Noel

Mentor: Carlos Ocampo

Instructor: Masoud Sadjadi

**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 chromcast 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 |  |
| 1.1. Design Methodology |  |
| 1.2. Terminology |  |
| 1.3. Overview of the Document |  |
| 2. System Design |  |
| 2.1. Overview |  |
| 2.2. Subsystem Decomposition |  |
| 2.3. Hardware and Software Mapping |  |
| 2.4. Security/Privacy |  |
| 3. Detailed Design |  |
| 3.1. Overview |  |
| 3.2. Static Model |  |
| 3.3. Dynamic Model |  |
| 3.4. Code Specification |  |
| 4. Glossary |  |
| 5. Appendix |  |
| 5.1. Appendix A - Use Case diagram |  |
| 5.2. Appendix B - Use Cases |  |
| 5.3. Appendix C - Class Interfaces |  |
| 5.4. Appendix D - Diary of Meetings |  |
| 6. References |  |

**1. Introduction**

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

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

**1.1 Problem Definition**

The problem of our project, the Social Wall, is the inability for clients to display photo albums in a slideshow from their Fotozap accounts in a fast and convenient manner on any screen or display. While it is possible to accomplish, it is normally tedious to set up and requires many different components to control the slideshow.

**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

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

PM - Picture Marketing

Xcode – Apple Integrated Development Environment

SW - Social Wall

CD – Cordova software

h/w - Hardware

s/w - Software

OS - Operating System

App - application

CC - Chrome Cast

AJS - AngularJS Framework

JS - Javascript language

CSS - Cascading Style Sheets language

ST – Sencha Touch 2 Mobile Framework

CS – ConnectSdk Framework

MVC- Model-View-Controller Pattern

IOS – iPhone OS (Operating System)

HTTP – Hypertext Transfer Protocol

HTTPS – HTTP Secure

**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 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.

A Package Diagram of the sender application. Showing MVC Architecture.

The system as a whole can be described as a client server architecture, where the client is a Chromecast device, which makes requests to a webserver to display a Chrome cast receiver application and that can communicate with an IOS mobile application.

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. The swcarousel subsystem uses object-oriented JavaScript.

**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 user’s 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. 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 section 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 up.

**3.1. 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.

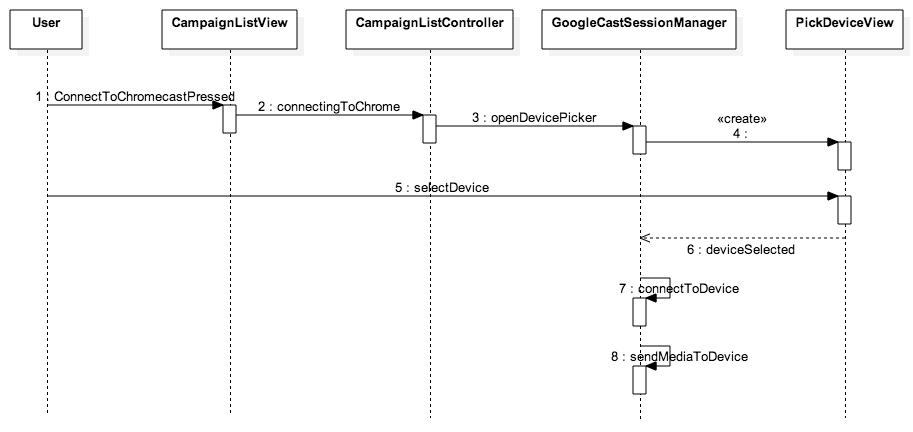
**3.2 Static Model**



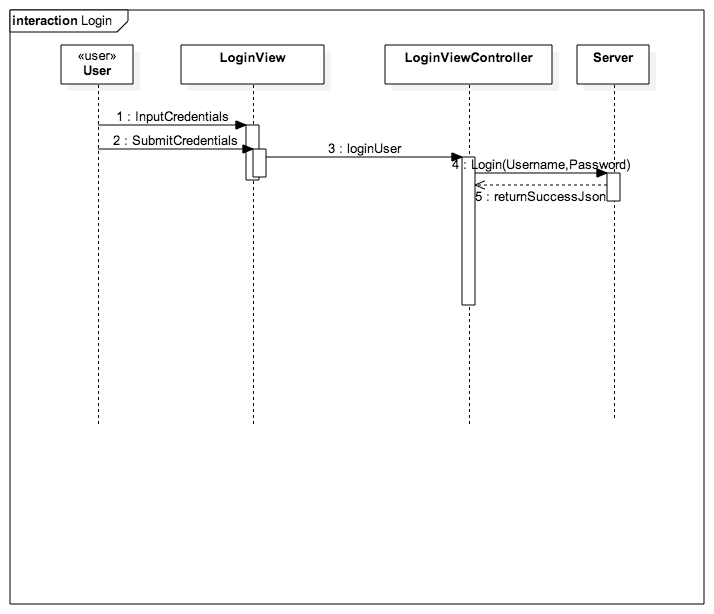
Sender Application Class Diagram

* **Sender Application Subsystem** 
  + HTML5-CSS
  + Framework: Sencha Touch 2
  + Architecture: MVC & Cordova(App Only)
  + The sender application subsystem is made up of an IOS 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.

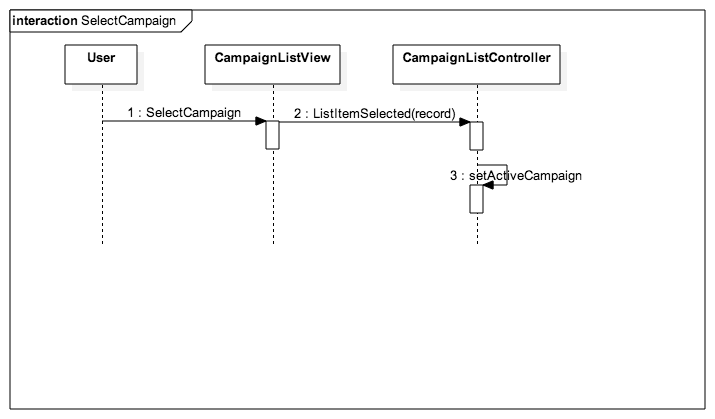
**3.3 Dynamic Model**



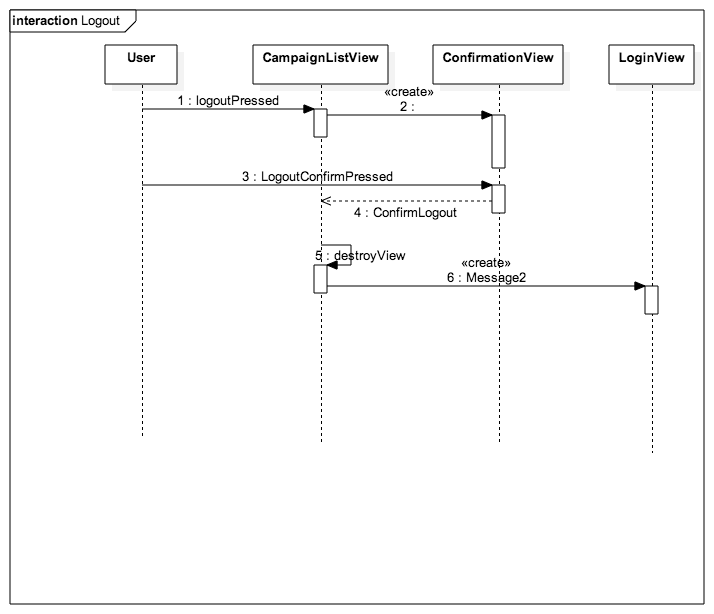
Sequence Diagram – Connect to Chrome cast device Use Case.



Sequence Diagram – Login Use Case.



Sequence Diagram – Select Campaign.



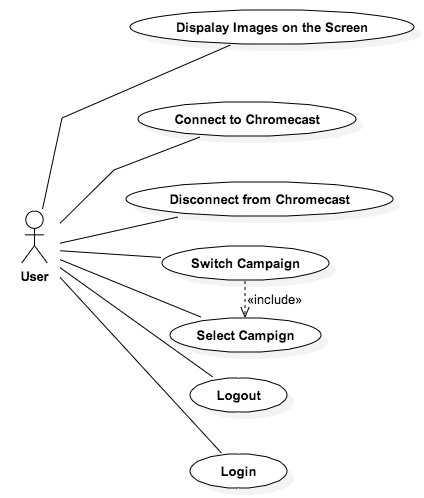
Sequence Diagram – Logout

**3.4 Code Specification**

**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**

**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 message and the system changes view.

**Postconditions:**

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..

**Related Use Cases:**

**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:** September 8, 2014
* **Last Modified:** October 5, 2014

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

**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 login screen.

**Postconditions:**

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

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

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used mostly every time the software is used.
* **Criticality:** High. Core functionality of software.
* **Risk:** Low. Fundemental .

**Constraints:**

**Modification History:**

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

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

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

**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 Chromecast 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 to connect to.

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 webAppSession with the running application.

**Postconditions:**

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

2. The user has an active webAppSession with the custom receiver application.

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

**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. Fundemental .

**Constraints:**

**Modification History:**

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

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