CIS 4911 – SENIOR PROJECT

Picture Marketing’s Social Wall

**Design Document**

Members: Steve Noel

Mentor: Carlos Ocampo

Instructor: Masoud Sadjadi

**COPYRIGHTS AND TRADEMARK NOTICES**

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

**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 our scope, purpose, acronyms and background on the project. It also deals with explaining how the rest of the document is organized, by describing our feasibility study and 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 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.

**1.2 Design Methodology**

The software process model being utilized is that of agile development. Regarding system requirements, utilizing agile development allowed for dynamic development of the language, 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, testing oriented way. Models that will be used to represent the design are as follows

- Class Diagrams

- Sequence Diagrams

- Use Case Diagrams

- Package Diagram

- Detailed/Minimal diagrams

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

PM - Picture Marketing

SW - Social Wall

FB - Facebook

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

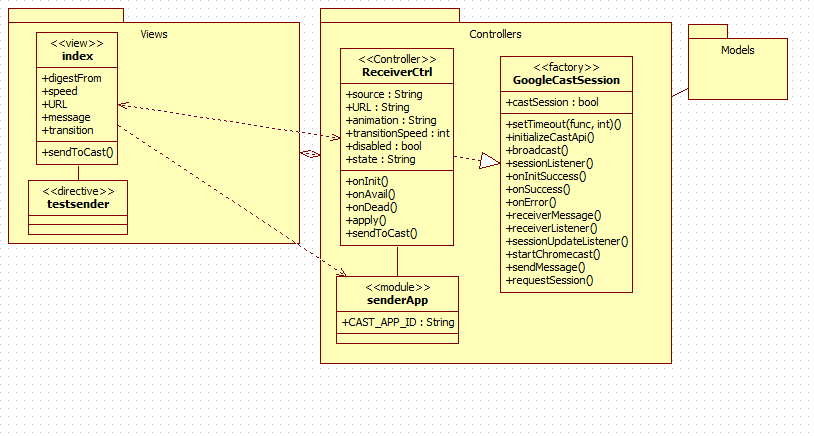
**1.4 Overview of Document**

This document is broken up into 5 main chapters. Each chapter is further broken up into sections. This is 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. 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 each of the aforementioned subsystems to that hardware and software. Section 2.4 concerns itself with the persistent data our system utilizes and how it will be managed. Section 2.5 discusses the security concerns of *Social Wall*?

Chapter 3 is focused on the design of the system itself. Section 3.1 introduces the classes we have broken the system down into and identifies the design patterns used in the class structure. Section 3.2 is more 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 a 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* utilizes an MVC architecture for its different components. I.e. Model-View-Controller structure. An overview of this design will be described in section 2.1, with the succeeding sections detailing our system itself.

**2.1. Overview** The design chosen to be used for the specific subsystems is the MVC architectural pattern, splitting the specific subsystems 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. 

Package Diagram of the sender application. Showing MVC Architecture.

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

For the creation of chrome cast receiver subsystem, the AngularJS framework was utilized. The sender subsystem, the IOS mobile application, utilizes the Cordova and Sencha Touch 2 frameworks to create a hybrid mobile application.

**2.2 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 username, password and campaign id, 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. Slideshow 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.

**2.3. Hardware and Software Mapping**

* Users Hardware/Software
  + Hardware: Apple Computer, PC, IOS Device, Google Chrome cast, HDMI capable device.
    - Apple Computer and IOS 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:
    - IOS Operating System - Allows for the mobile sender application to run and allows user interaction with custom receiver application and chrome cast.

**2.4. 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.

**3. 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.

**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.

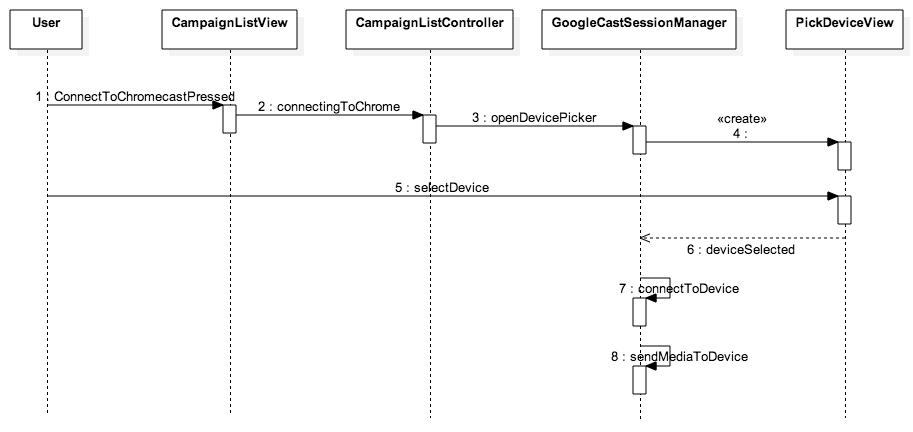
**3.2 Static Model**



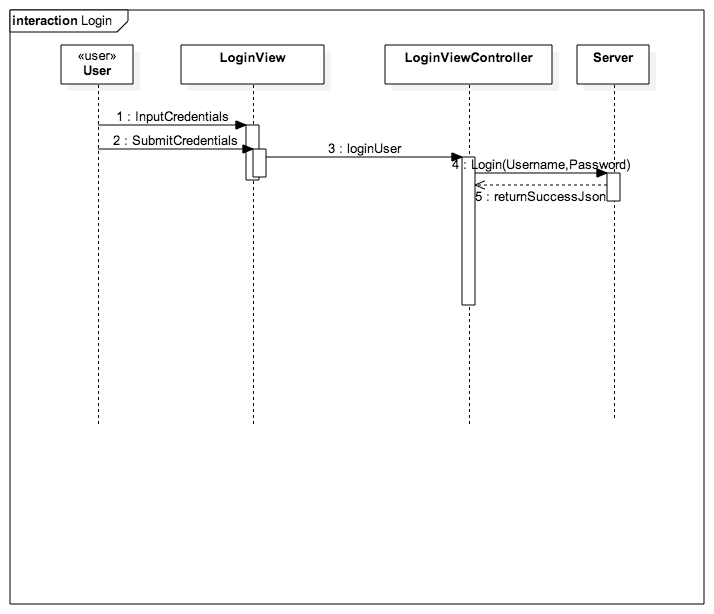
Sender Application Class Diagram

* **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.

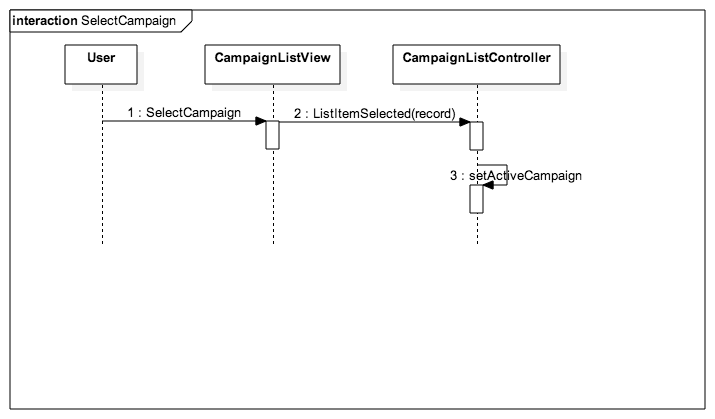
**3.3 Dynamic Model**



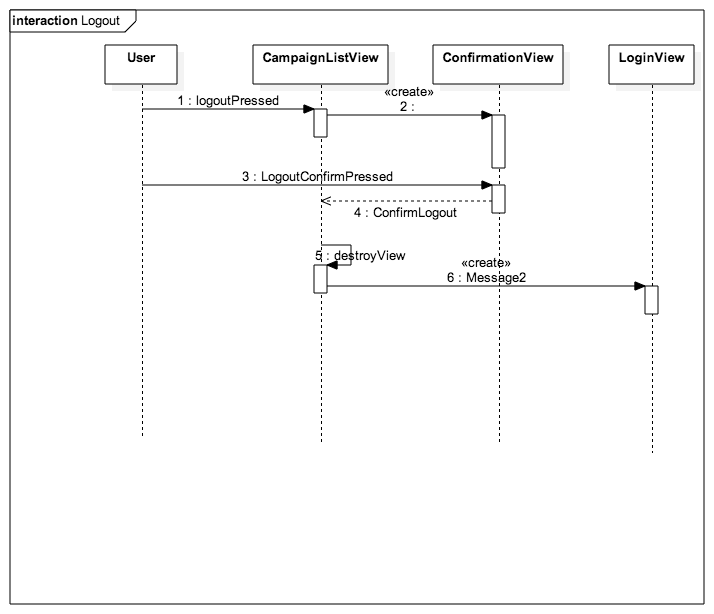
Sequence Diagram – Connect to Chrome cast device Use Case.



Sequence Diagram – Login Use Case.



Sequence Diagram – Select Campaign.



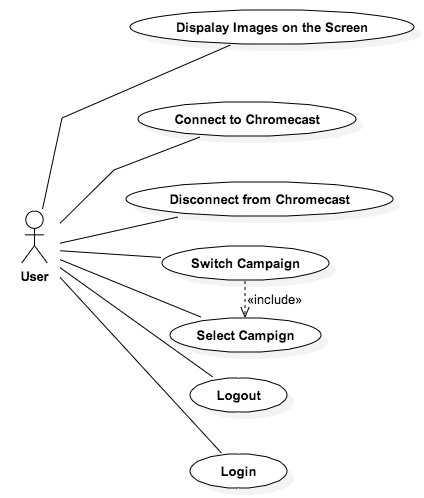
Sequence Diagram – Logout

**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.

2. The user then presses the submit 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 everytime 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 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 Chromecast“ button.

2. The system shows a list of chromecast devices on the network.

3. The user then selects a device from the list to connect to.

4. The use case ends when the system.

**Postconditions:**

1. The user shall have the application installed on their phone and have it open on their device.

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

**Exceptions:**

**Related Use Cases:**

**Decision Support:**

* **Frequency:** Will be used mostly everytime 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

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