Northern Arizona University

Capstone Team Project

Design Document

Bit Tag

Temitope Alaga, John Dance, Joshua Frampton, Jun Rao

CS 486c

Version 1.3

2/12/2016

Table of Contents:

Introduction:

Bit Tag Technologies has sponsored our team to develop a mobile app in order to extend their current functionality of a web site tagging system to further platforms to link website content like Facebook “tags” photos. Their current platform is limited to a Chrome extension and so they are trying to build out their platform to include a wider audience. Building the Bit Tag application into a mobile app will allow for more people to easily access the technology. Since the current desktop application is built into an extension, it is not currently easy or possible to use the application outside of Chrome. This mobile application will show that it is possible to port this technology to a variety of platforms and show that the technology can be viable.

This document will outline the design of the Bit Tag mobile application architecture. The Bit Tag mobile application aims to extend the uses of the online webapp into the mobile environment. It will take the high-level concept of Bit Tag tagging and apply it in a format that is relevant to the mobile platform. There are two core parts needed for the application to work correctly. The first part is the ability to pull from and review the Bit Tag "tag repository” which hosts all of the tag data in a central location. The "tag repository" will be accessed via an API. This information should be pulled into the application and displayed in such a way that is comparable to the web version of Bit Tag. The second portion of the application should be able to accurately display the tags on the tagged content where it is formally hosted.

The ability to perform this task natively on a mobile platform is very limited so this will be solved by implementing a web browser within the application itself. The "tag repository" of the application should be able to send the data that was pulled from the Bit Tag servers through to the web browser side of the application, the web browser side should then place visual tags on the content where it is appropriate. To ensure portability of the application, the application will

be developed in a WebView context. A WebView object is effectively a web viewer built into an app. To achieve this, we will be using Apache Cordova to wrap the application into a target platform native application. The system will primarily be built in web languages such as HTML, CSS, and JavaScript because of this. The primary messaging architecture of the application will be designed around the Model View Controller (MVC) architecture. The MVC architecture allows us to abstract a level of communication in the app and separate these lines of communication into three core portions, ie. the Model, the View, and the Controller. Using this architecture will allow us to more effectively and efficiently create and destroy views as it is applicable while maintaining a good level of organization in workflow and data. The application will also be designed in such a way that it does not violate the terms and conditions of mobile application stores such as Google Play and the iOS App Store.

Architectural Overview:

Figure 1 represents the internal architecture of the mobile app. This architecture represents a Model View Controller pattern. The core logic of the application is done on the controller level. This means that as the application is interacted with, the controller is able to react to these interactions controlling

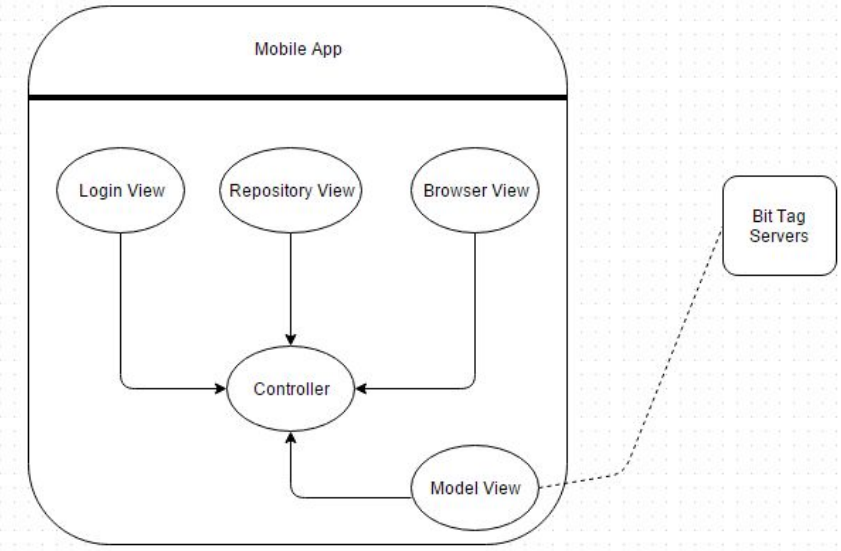


Figure 1: Architecture Diagram

the flow of the application. The interactive portion of the application is handled by the views. Each of the views defined in Figure 1 show the core portions of interactivity. The views themselves will be reasonably lightweight and focus primarily on displaying the desired data at the desired time. Finally the model is the datastore of the application. This will handle the on-board storage of data along with the accessing of data from the Bit Tag servers. The Bit Tag server's job is to contain and store every tag that has ever been created and uploaded to Bit Tag's server, this is known as the "tag repository". This will allow our app to gather and save tags to upload to the server. Thus, the app will have the access to tags on every device.

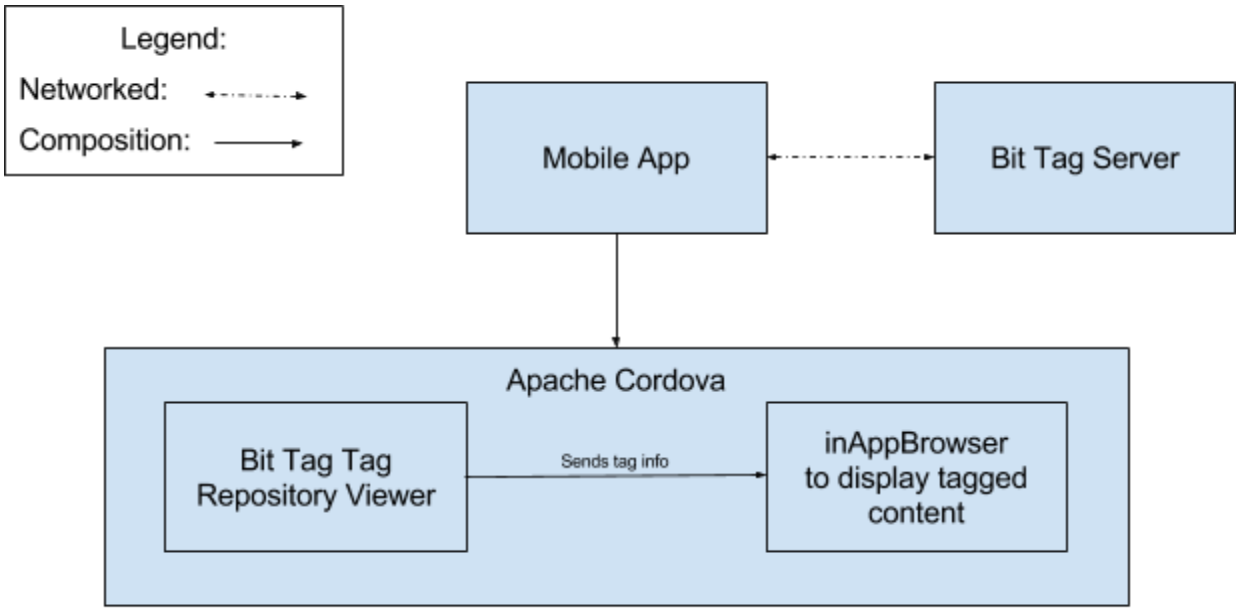


Figure 2: System Level Diagram

As we abstract up a layer we have the larger technology interaction stack. This can be seen in Figure 2. As we are developing the application using a mobile WebView, we have the ability to use a mobile development framework called *Apache Cordova*. This allows us to target multiple mobile platforms with one source. At a high level, the app will consist of two key parts, the "tag repository" viewer, and a browser to view these tags. The repository viewer will be built

from the ground up. The browser however will be using a Cordova plugin called *inAppBrowser* that supports a variety of popular browser features along with code injection which is critical for displaying tags.

Module and Interface Descriptions:

Login View:

We will authenticate each login attempt by sending an Ajax request to the Bit Tag server in order to check the credentials with their database. The server will then generate an authentication token and send that along with the sign in information back to the application. After authenticating the user, the Bit Tag server will send a response back to our application which will redirect the user to the Repository view. By having our login view (Fig. 3), we will be able to allow users to create and manage their own tags within the entire application and when going to the in-app browser.

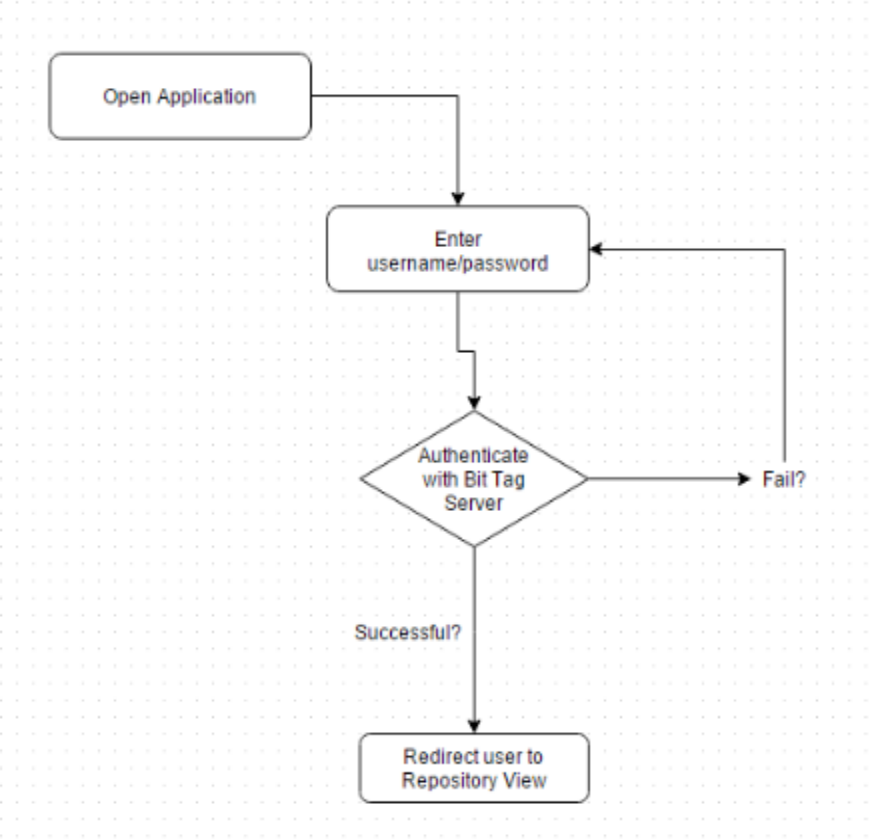


Figure 3: Login View

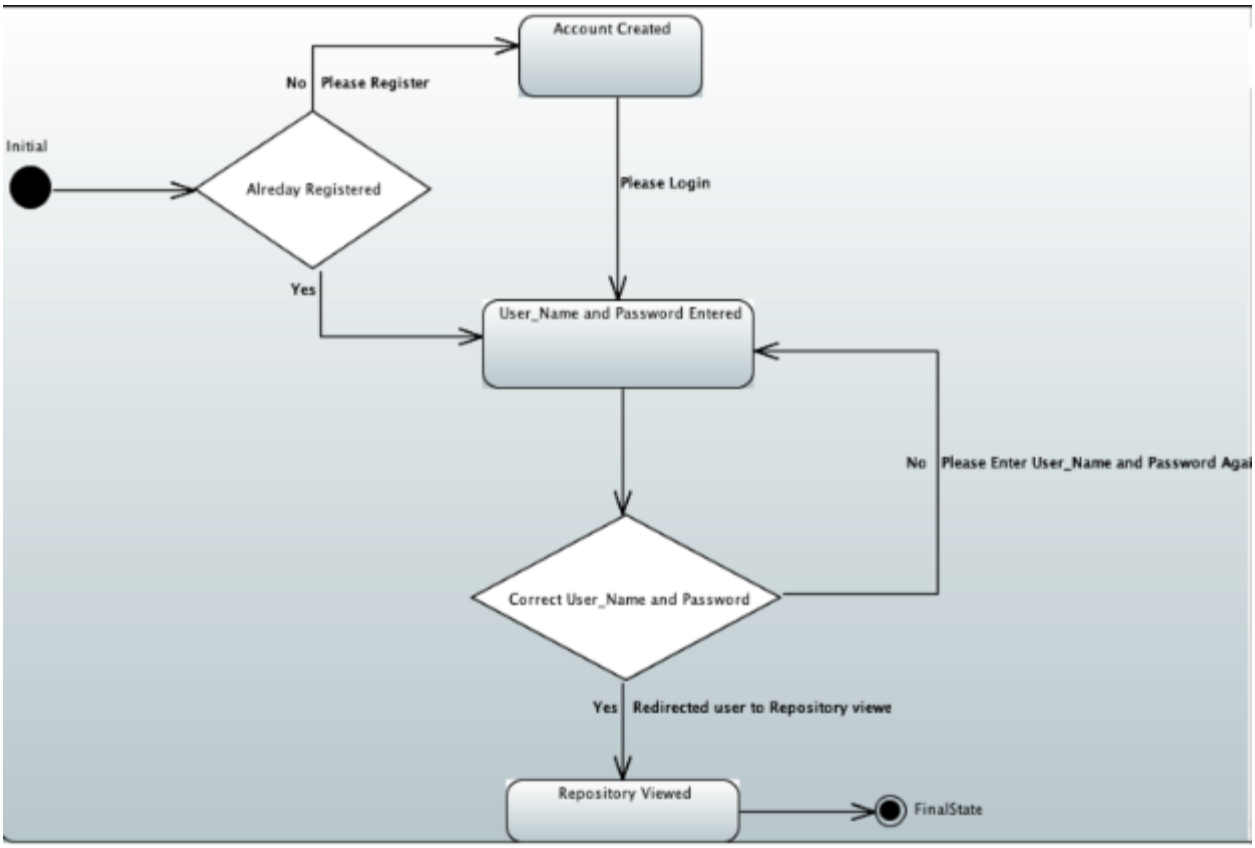


Figure 4: Repository View

Repository View:

The repository view (Fig. 4) will contain many of the information which has been already created or stored through the Chrome Extension onto the Bit Tag server. This will include tags and content that the users create while using the software. We will pull the entire repository from the Bit Tag server using the Bit Tag APIs and then display each tag on the mobile app's repository view. The user will then be able to modify, delete, and share tags they have created in the past through this view which will immediately reflect on Bit Tag's servers. Modifying tags will consist of organizing and commenting. As a user you can organize your tags and other tags by category in order to help you sort them, this will reflect on the repository view and will help

the user sort and organize the content more efficiently. Most of the methods needed for this functionality has already been created on Bit Tag's end so using their libraries and the Bit Tag API we will be able to implement this rather easily into the mobile app.

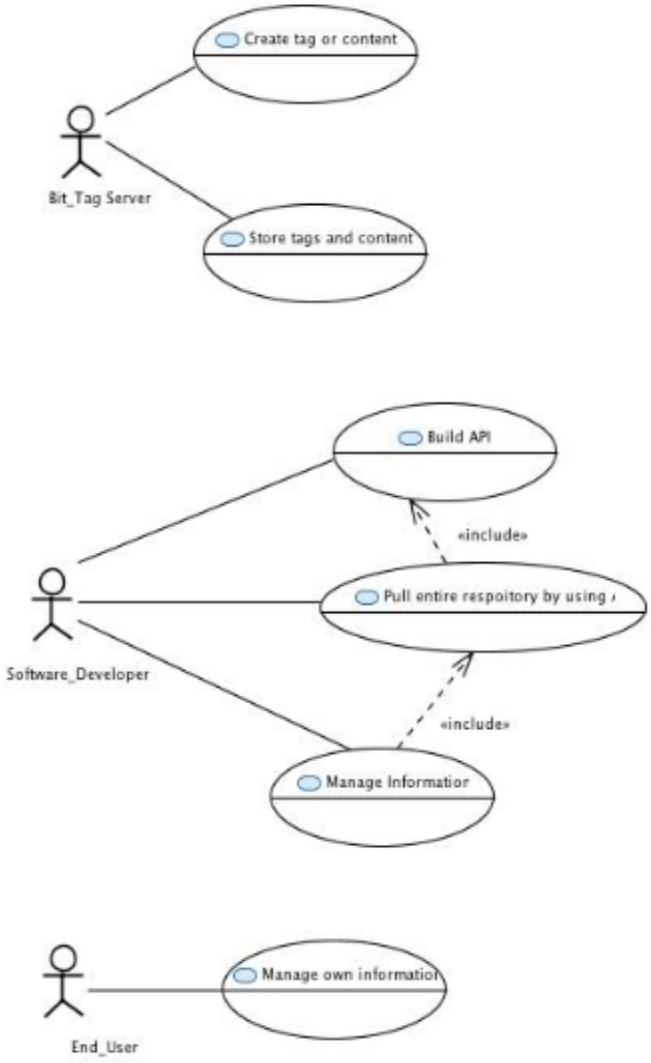


Figure 5: Browser View

Browser View:

The browser view's (Fig. 5) responsibility will be to ensure that the user can easily create and view tags from the repository in a separate in app web browser. By allowing the user to do this, we are giving them the ability to have full functionality of Bit Tag's services from within the mobile application. When the user clicks on a specific tag from the repository view, it will take them to their in app browser and will show the page in which the tag was created on and will display the portion of that page with the tag and the comment left by the previous user. When creating the tag, the user will have an option to create a tag for most content while in the in app browser.

In the browser view, we will be using Apache Cordova's *InAppBrowser*, which is a web browser that affords many of the same features that users are used to in their normal web browsers. By using *inAppBrowser's* executeScript and addEventListener we will inject our JavaScript code into the browser window

which will allow the user to view and create new tags which will then immediately reflect into our repository view and then will allow the user to manage and organize their specific tags within that view. In order to create the specific actions for the tags, we will need to use JavaScript to manipulate the DOM elements by adding elements in order to create and view tags easily.

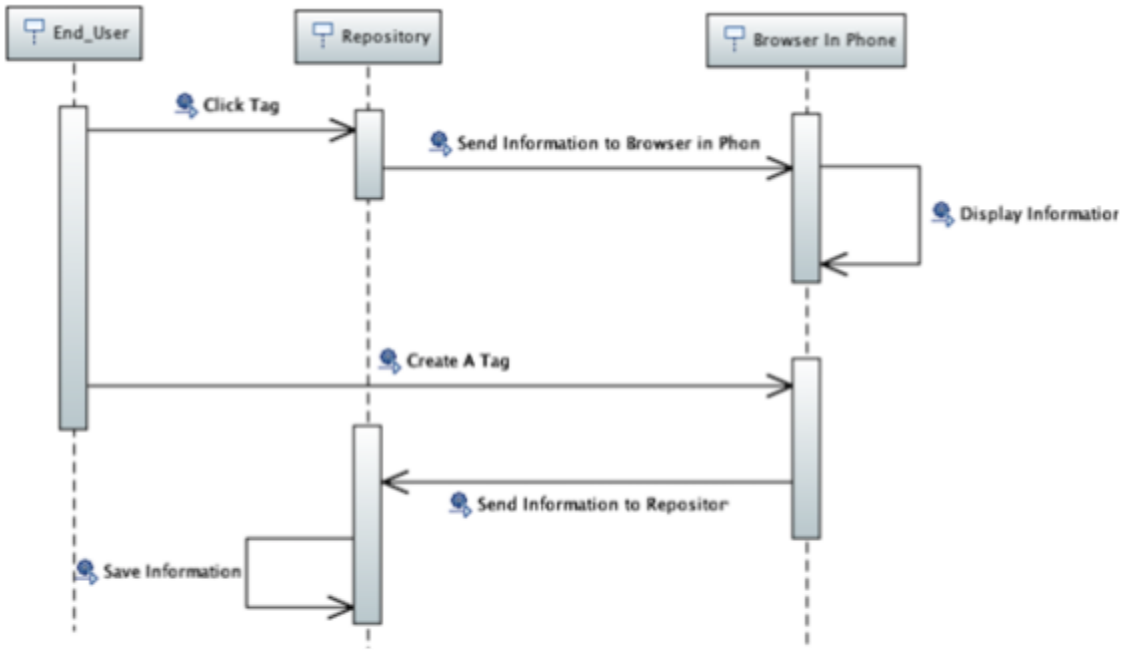


Figure 6: Sequence Diagram

Implementation Plan:

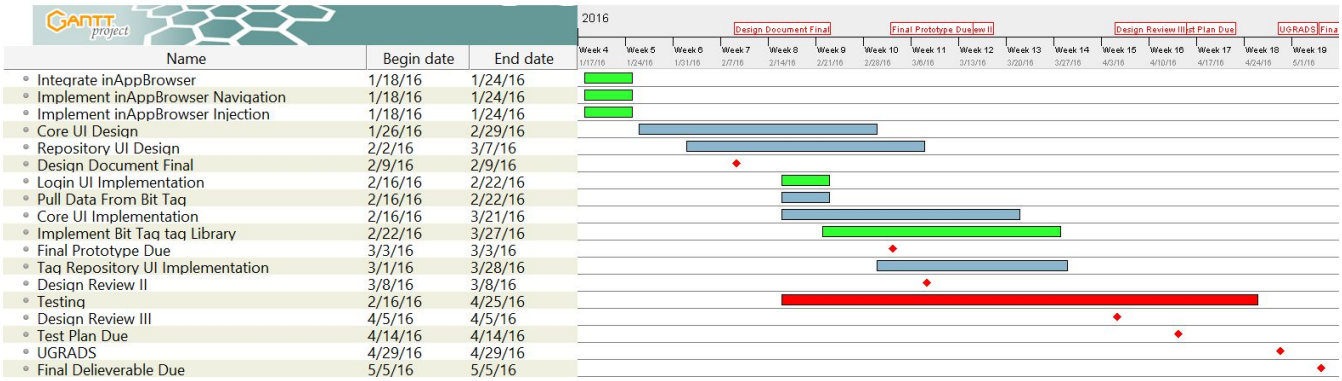
At a high level, the work will be broken down into two key sections:

1. The development of the "tag repository" portion
2. The development of the tag viewer portion

The breakdown of work within this can be viewed as the following:

- "tag repository"
    - UI design
        - Core UI - includes login
        - Repository UI
    - Core UI implementation
    - Login UI implementation
    - Pulling and formatting data from Bit Tag API
    - "tag repository" UI implementation
- Tag Viewer
    - Integrate *inAppBrowser* plugin
    - Implement browser navigation
    - Implement browser injection
    - Implement Bit Tag tag library

To start with, the work will be split up into the two main groups as described. The "tag repository" portion will be worked on by Jun Rao and Temitope Alaga. The tag viewer portion will be worked on by Josh Frampton and John Dance. By splitting the work like this we can effectively work in parallel without having conflicting workflows. The subteams themselves will be responsible for splitting their sub-workloads.



Name	Begin date	End date
Integrate inAppBrowser	1/18/16	1/24/16
Implement inAppBrowser Navigation	1/18/16	1/24/16
Implement inAppBrowser Injection	1/18/16	1/24/16
Core UI Design	1/26/16	2/29/16
Repository UI Design	2/2/16	3/7/16
Design Document Final	2/9/16	2/9/16
Login UI Implementation	2/16/16	2/22/16
Pull Data From Bit Tag	2/16/16	2/22/16
Core UI Implementation	2/16/16	3/21/16
Implement Bit Tag tag Library	2/22/16	3/27/16
Final Prototype Due	3/3/16	3/3/16
Tag Repository UI Implementation	3/1/16	3/28/16
Design Review II	3/8/16	3/8/16
Testing	2/16/16	4/25/16
Design Review III	4/5/16	4/5/16
Test Plan Due	4/14/16	4/14/16
UGRADS	4/29/16	4/29/16
Final Delieverable Due	5/5/16	5/5/16

Figure 7: Gantt Chart

Jun and Al: Gray

Josh and Keevan: Green

Entire Team: Red

References:

Apache Cordova:

https://cordova.apache.org/

Bit Tag:

http://www.bittagbox.com/

inAppBrowser:

https://www.npmjs.com/package/cordova-plugin-inappbrowser\

WebView:

http://developer.android.com/reference/android/webkit/WebView.html

https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIWebView_Class/

Diagram Tool:

https://www.draw.io/