
Video Tag Portal

System Design Document | Current Version [0.3.0]

Prepared By:
Anudeep Potlapally
Travis Rous

Revision History

<i>Date</i>	<i>Authors</i>	<i>Version</i>	<i>Comments</i>
9/20/12	Travis Rous & Anudeep Potlapally	1.0.0	Initial version
10/4/12	Travis Rous & Anudeep Potlapally	1.1.0	Added prototype 2
11/2/12	Travis Rous & Anudeep Potlapally	1.2.0	Added prototype 2
12/6/12	Travis Rous & Anudeep Potlapally	1.3.0	Added prototype 3
02/08/13	Travis Rous & Anudeep Potlapally	1.4.0	Added Prototype 4
03/15/13	Travis Rous & Anudeep Potlapally	1.5.0	Added Prototype 5
04/25/2013	Travis Rous & Anudeep Potlapally	1.6.0	Added Final Product

Table of Contents

1.0	Overview	5
1.1	Scope.....	5
1.2	Purpose	5
1.2.1	Major System Component #1: Database	5
1.2.2	Major System Component #2: APIs	5
1.2.3	Major System Component #3: YouTube and Vimeo.....	6
1.2.4	Major System Component #4: Popcorn.js	6
1.3	Systems Goals	6
1.4	System Overview and Diagram	6
1.5	Technologies Overview	7
2.0	Project Overview.....	9
2.1	Team Members and Roles	9
2.2	Project Management Approach.....	9
2.3	Sprint Overview.....	9
2.3.1	Sprint 1: Initial prototype.....	10
2.3.2	Sprint 2: Database Design	10
2.3.3	Sprint 3: Design and Connect pages	12
2.3.4	Sprint 4: Giving the users an attractive interface	15
2.3.5	Sprint 5: Authentication and Finalizing Layouts	19
2.3.6	Sprint 6: Finalization and Testing.....	24
3.0	Requirements.....	26
4.0	Design and Implementation.....	26
4.1	Client Browser.....	26
4.1.1	Technologies Used	26
4.1.2	Component Overview	26
4.1.3	Sprint Overview.....	26
4.1.4	Architecture Diagram	27
4.2	jQuery.....	28
4.2.1	Technologies Used	28
4.2.2	Architecture Diagram	28
4.3	Popcorn.js	29
4.3.1	Technologies Used	29
4.3.2	Component Overview	29

4.3.3	Sprint Overview.....	29
4.3.4	Architecture Diagram.....	30
4.4	Database Management.....	30
4.4.1	Technologies Used	30
4.4.2	Component Overview	30
4.4.3	Sprint Overview.....	31
4.4.4	Database Schema.....	32
4.4.5	Architecture Diagram.....	33
4.5	User Authentication	34
4.5.1	Technologies Used	34
4.5.2	Component Overview	34
4.5.3	Sprint Overview.....	34
4.5.4	Architecture Diagram.....	35
4.6	Video Source	36
4.6.1	Technologies Used	36
4.6.2	Sprint Overview.....	36
4.6.3	Architecture Diagram.....	37
5.0	System and Unit Testing	38
5.1.1	General Testing	38
5.1.2	Manual Test Cases	38
5.1.3	Automated Test Cases	43
5.2	Dependencies.....	44
5.3	Test Setup and Execution.....	44
6.0	Development Environment.....	45
6.1	Development Tools	45
6.2	Source Control	45
6.3	Build Environment	45
7.0	Release Setup Deployment	46
Appendix I:	Supporting Information and Details.....	46

1.0 Overview

The goal of the document is to show the way that we have implemented video tag portal. There will be details on the purpose, goals, as well as requirements and implementation.

1.1 Scope

There will be a detailed discussion on how the project is implemented and what tools are necessary to create the project.

We have 4 major system components

- Database – this is where tags, users, and other locally needed objects are needed
- APIs – These APIs are need for a number of things such as login and video uploading
- YouTube and Vimeo – source of videos for the project
- Popcorn.js – used to implement the tags that our project is based around

1.2 Purpose

We have created a video tagging system that allows users to tag points in videos with a various number of predefined tags. After tagging a video, the user then has the ability to share the video with his or her friends. The friends then will have the ability to also tag the shared video. The goal of the system is to make videos more socially connectable between people.

1.2.1 Major System Component #1: Database

The Database holds all of the information that the system will need in order to function. The database contains all of the information to control the users, friends and the tags used by the player. The database also contains the tag timings and the list of the user's tags, as well as the information on who tagged what videos. In the database there are separate tables for separate videos venders (e.g. YouTube and Vimeo).

1.2.2 Major System Component #2: APIs

Here is a description of the APIs that we used for the project

Google data API

- Allows us to use the videos that are hosted on YouTube
- Allows us to upload videos to YouTube
- Allows us to manage users that are logged into our site

Facebook API

- Allows us to manage users that are logged into our site
- Allows us to map friends to different accounts

1.2.3 Major System Component #3: YouTube and Vimeo

We have support for the two major video hubs: YouTube and Vimeo. We are able to get the timings and embed both video sources which helps us get the tags to show up at the right points in the videos. Most of the functionality favors YouTube because it has a more complete API, so we use YouTube in searches and uploads. However, the actual tagging there is almost no difference to the user except for where the videos are from.

1.2.4 Major System Component #4: Popcorn.js

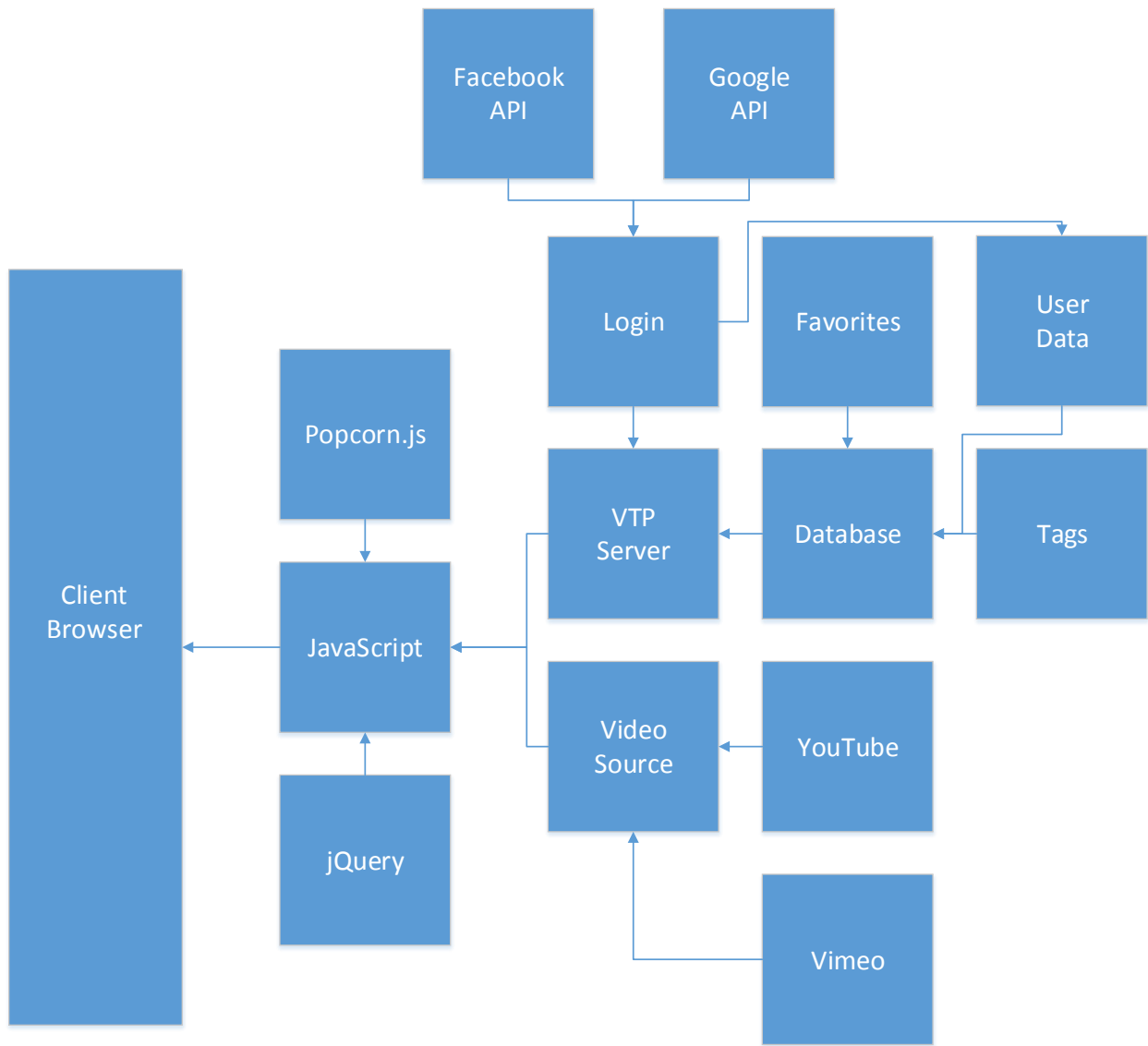
The Popcorn.js framework is what allows us to make the different tags appear and disappear based on the timing within the videos. There was support for the videos from YouTube and Vimeo. This framework has the ability to check an embedded video in a sub-second manor and be able to trigger events based on the timing.

1.3 Systems Goals

System goals include giving the user the ability to use our application to make video part of the online social experience. We want the user to have a “Natural” user interface to make the tagging process simple for the user. Also one major goal is to make the product as much as possible a social networking site.

1.4 System Overview and Diagram

Much of the project will depend on the various APIs that are needed to make the videos display and show up well on the web page. We will also need to use the APIs to manage the users that access the site using Facebook and Google. In addition to the APIs needed in the project there will be a database that is used to manage the different users and the tasks that they will be able to do. There are many different things that will be going on per user such as knowing there favorites to logging in to the site. The most significant system component is the Popcorn.js.js framework that will actually allow us to make time based tags on the fly which is the main part of the project.



1.5 Technologies Overview

PHP

- Stands for PHP: Hypertext Preprocessor(recursive acronym)
- It is a server side scripting language that is designed for web development.
- Supports cross platform development

HTML5:

- A markup language that is for structuring and presenting content for websites
- Contains many more features than the previous versions of HTML
- Supports cross platform development

JavaScript:

- An interpreted client-side scripting language
- Used to make user interfaces more interactive
- Used to communicate asynchronously
- Used to alter document content that the user sees dynamically

Popcorn.js:

- A JavaScript frame work for editing media in real time
- Able to edit media based on timing of the media
- We use it to control the main functionality of our site by displaying tags at specific times
- Support for YouTube and Vimeo
- Checks media in fractions of a second

Google APIs:

- Google Maps API v3
 - Using the maps API to display the map tags in our application
- Google+ API
 - Added to allow the users to login with Google accounts
- YouTube Data API v3
 - Allows the access of the YouTube search
 - Allows users to upload videos to YouTube
 - Allows checks users uploads to YouTube

Facebook API:

- Allows users to login using their Facebook accounts
- Allows users to see the friends that they have on Facebook
- Allows users to view profile pages from Facebook

jQuery:

- A feature rich JavaScript library
- Allows for simpler dynamic HTML traversal/modification
- Allows for simpler dynamic animation and Ajax coding
- Multi-browser support

Vimeo:

- A website for hosting videos.

YouTube:

- A website for hosting videos.
- Contains many useful APIs

MySQL:

- Open source relational database management system
- Allows for multi-user access to a number of databases

2.0 Project Overview

This section will provide some of the team roles and the way the project is managed and kept on track.

2.1 Team Members and Roles

The two members of the project are Anudeep Potlapally and Travis Rous. Each member is responsible for attending the scrum meetings and doing what is assigned to them after each meeting. There are only two team members so there is much overlap in the roles, however Anudeep was the official contact to the client. Travis had more of a testing role as well.

2.2 Project Management Approach

We are using the Scrum development cycle to manage the project. This consists of almost daily scrum meetings that are less than 5 minutes each (minimum bi-weekly). There are sprints that range from 2 to 3 weeks. We are using GitHub and Trello to manage our code and progress over time. Trello is like a digital board with task card that are able to be moved from to show the progress we have done in the current sprint and assign some cards to have a higher priority than others to show the significance of each task or what is left for each task at hand.

2.3 Sprint Overview

The major phases in the project are as follows (We will explain in greater detail in the next few sections):

Sprint I:

- Embed videos from 3rd party sources.

Sprint II:

- Implement feature to play videos from YouTube using Popcorn.js.
- Link video player to HTML elements to display hard-coded tag information.
- Use Database to save and retrieve tags.

Sprint III:

- Implement feature to play videos from other possible sources.
- Display multiple tag types simultaneously.

Sprint IV:

- Started work on an upload page
- YouTube search was implemented.

Sprint V:

- Updated favorites page to include the users profile picture from Facebook.
- Added a friends page.
- CSS experimentation.
- Vimeo integration.
- Search and Loading of videos now in the same bar
- Linked Facebook and Google accounts.

Sprint VI:

Finished Requirements and started testing

2.3.1 Sprint 1: Initial prototype

The initial prototype was needed to get our idea out to the client and make sure that we were on the same page. We were also able to get the videos embedded into the site. There was also research for finding the many different APIs and other frameworks that would be useful to us in the future of the project.

Overview of sprint:

- Created prototype
- Finished the client agreement
- Prepared for client presentation
- Cut up the project (know when to do what)
- We defined some project boundaries
- Also defined a set of tools that are were going to be used

Some of the early goals

- Play videos in the application
- Play videos from major sites
- Create a database to store some video information
- Social network ready
- Selecting video tag points
- Create the list of tag types
- Have a simple way to make the tags editable
- Filters
- Possible mobile development

2.3.2 Sprint 2: Database Design

Summary:

For this sprint there was mostly research and learning of how to implement some of the features that we will be trying to use in the project. Anudeep had discovered and Implemented the Popcorn.js which is used create videos with an features such as what we are trying to implement.

Sprint 2 Backlog:

- Database Design (In Progress...)
- Creating a favorites page (In Progress...)

Sprint 2 Burndown:

RESEARCH

- Anudeep had researched the YouTube data API and found that we needed to get permission from Google to use it which allows VTP to search, upload videos. Travis sent Google a request to use the data API and we are currently waiting for a response.

DATABASE DESIGN

- The current host location is just local on our personal machines. There is a database that we can both access that is being hosted on Travis' desktop. We are using MySQL via the phpMyAdmin tool.
- We currently have a database with three tables; users, yttags, favorites. In the users table there an auto incrementing ID number, the user id that is a 50 char array for the userId, a dateRegistered date that is auto timestamped, and a hostSite variable that is going to tell us what site the user is logging in with(e.g. Facebook, Google account, ...etc.). In the ytTags table we have an auto incrementing ID number, a videoId which is an the ID that the site that we get the video from uses to ID the video, a userId is used to associate the tag that is added with a specific user, and other video details used to create the tags. In the favorites table there is another auto incrementing ID, along with a userId to tell us which user's favorites this is, an auto timestamp, and a videoId to tell us the video ID that is a favorite.
- Anudeep created a database connector class to interact with MySQL from PHP and as we go we will be adding functions for frequent DB queries.

INTEGRATION OF POPCORN.JS

- Popcorn.js is an open source library developed by Mozilla Corporation which allows audio, videos to interact with HTML elements. Anudeep researched and integrated Popcorn.js.js into the system and also automated displaying image tag's for videos with tag entries in database. Popcorn.js supports HTML5, YouTube, Vimeo videos in future adding these supporting

CREATING A FAVORITES PAGE

- Travis started the favorites page, it currently gets the favorites for a dummy user named "123" who has five videos saved as favorites. Currently the page will only display the thumbnails and titles of the favorites of the videos.

LOGIN VIA GOOGLE AND FACEBOOK

- Anudeep Integrated Facebook PHP SDK and Google PHP API to allow users to login via Facebook or Google. Google API does not provide access to user's friends list. We might need to disable filter tags by friends option to Google users or create additional feature to support managing friends.

ERROR LOG

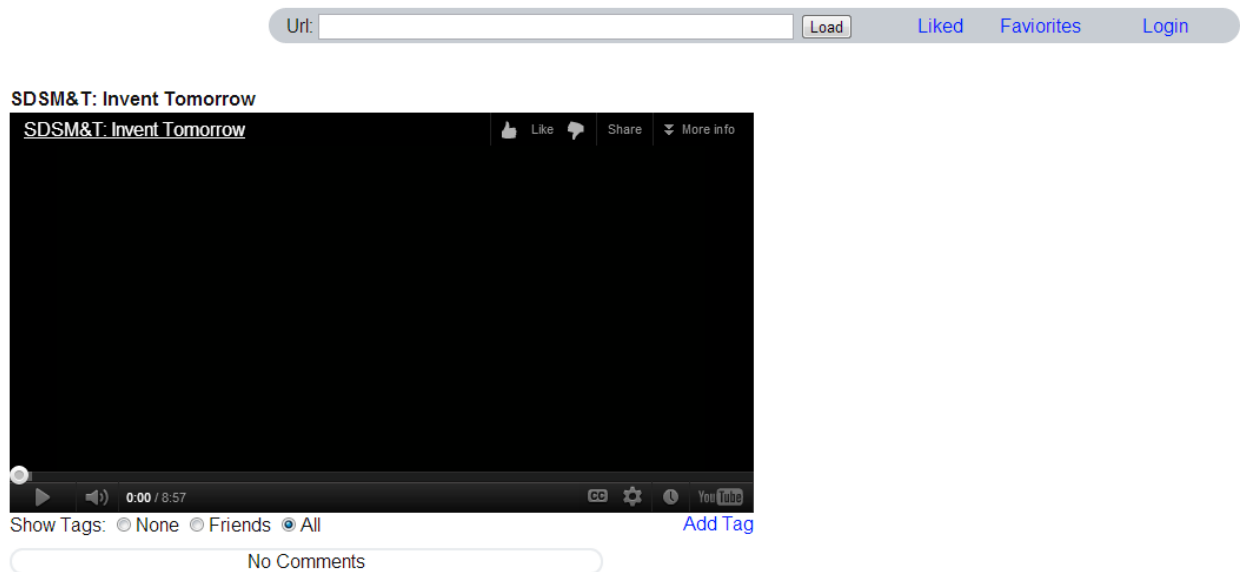
- Anudeep created error log function to keep track of all PHP generated errors. This function will create detailed entry for each error so that developers can attend the server-side bugs.

Current Prototype

The current prototype has gained a favorites page and is now using a database to get and store the information about users. We also can now login using Facebook and Google accounts. Anudeep has begun use of the Popcorn.js.

Prototype images

Sprint 2 look:



2.3.3 Sprint 3: Design and Connect pages

Summary:

In this sprint we created features for logged in users to add videos into their favorites list and also they can go through the list of their favorite videos and just click on them to watch again. We created a form to add image tag to the video currently being watched and play the image tag automatically from the database. We hosted the code on a server to see the product directly on the internet.

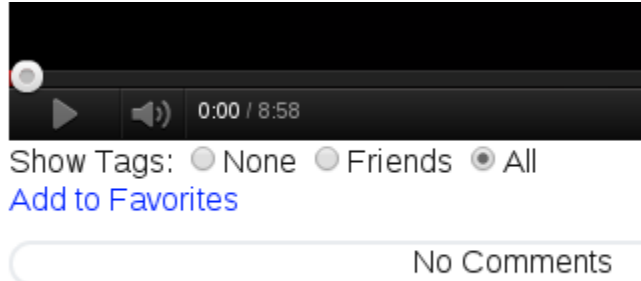
Sprint 3 Remaining Backlog:

- YouTube Data API (Permission pending...)

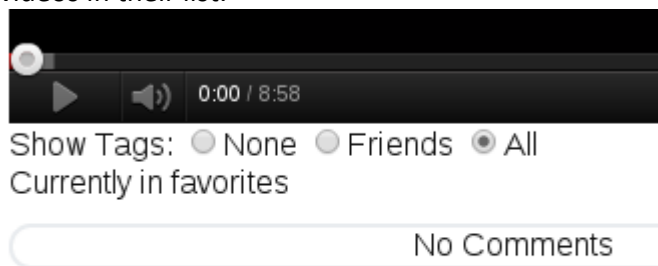
Sprint 3 Burndown:

ADD TO FAVORITES

- For logged in users Travis created option to add videos to their favorites list, and wrote required functions to add video information linking to user in the database.



- Videos can be added to favorites only once so when the video is already in favorites list the text would change to "Currently in favorites" forcing users to not make duplicates videos in their list.



FAVORITES PAGE DESIGN:

- Travis created a page where logged in users can look at the list of their favorite videos. Favorites page displays a thumbnail for the video and the title for the video and users can click on either video or the title and video and it would take user to the homepage to start watching the video again. To save memory on server thumbnail and video titles are not saved on database but retrieved from YouTube

server every time the page is loaded.

My Favorites



B.J. Mullens throws
it down on
LaMarcus Aldridge



Top 10 Dunks of the
Month: November
2012



SDSM&T: Invent
Tomorrow

ADD IMAGE TAG FEATURE

- Anudeep created a form for users to add Image tags and link it with a video and save information in the database. Wrote code to pull the saved tag information from database and show on the page at appropriate time. Tags should be played between appropriate timings of the video so linked the form to the player so that users can use seek bar on video to set exact time the tag should be played. Wrote JavaScript function to test the valid image link provided by the user and class to handle Image uploads to the server.

Add Tag

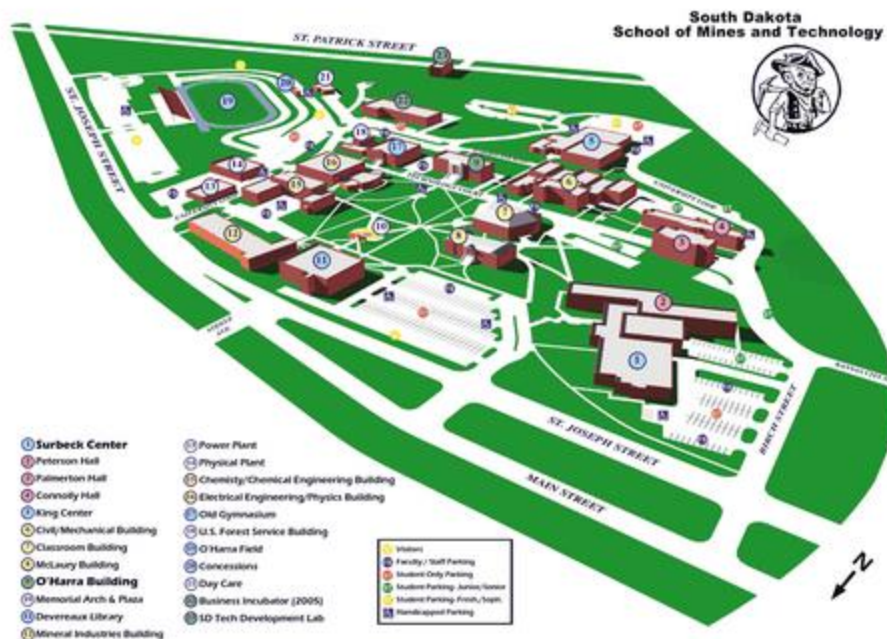
Start Time: 0:0:14

End Time 0:6:18

Type: ☐ Comment ☒ Image ☐ Map ☐ Web Link

Source: ☒ Web ☐ My Computer

Link:



HOSTED ON SERVER:

- Anudeep found a website to host the code which can run PHP and use MySQL database. Made required changes in Google API and Database classes to make them work on the new server.

2.3.4 Sprint 4: Giving the users an attractive interface

Summary:

Travis started the upload page, however it took more time than what was anticipated and still needs work before moving on. On the upload page, originally the user needed to put their credentials into the YouTube upload page on the VTP site. This is not done well so Travis is in the

process of getting the user authentication to work with the upload. Anudeep had implemented the YouTube search with user authentication and also implemented other tags.

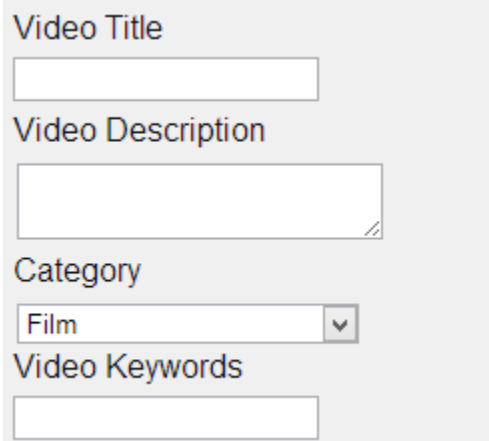
Sprint 4 Remaining Backlog:

- Video Upload fix-it list
 - error checking
 - Graphics
 - Add the users Video to the database
- Friends Page
- Grabbing new users videos from YouTube
- Displaying Map and web link tags

Sprint 4 Burndown:

YOUTUBE UPLOAD


- Added Upload to the headed
- Once clicking you are brought to a page to input the Video details.

A screenshot of a web form for uploading a video. The form is titled 'YOUTUBE UPLOAD' and contains five input fields: 'Video Title' (a single-line text box), 'Video Description' (a multi-line text box), 'Category' (a dropdown menu with 'Film' selected), 'Video Keywords' (a single-line text box), and a 'Video Upload' button. The form is styled with a light gray background and red text labels.


- Once the video fields are filled out then you will be able to authorize the upload to YouTube. Then you can choose the file and select upload. This will then take you to a page where you get the details about how to find the video on YouTube and a disclaimer about the video taking a long time to get done processing. Also on this page there is a link put the user back to the player page with their newly uploaded video.

SEARCH YOUTUBE VIDEOS


- YouTube made the Data API V3 publicly available for all the developers Anudeep implemented the search functionality for the videos. Now users can search YouTube videos and they can also change the search filters for relevant content.




NBA Best of Bloopers: January 2013
Check out the best of all the whacky plays from the NBA last month as we bring you all the jokes, pranks, on-court goofs and mishaps in January's ...




Top 10 Plays of the Night: January 26th
Two game-winners, two gravity-defying slams from Barnes & Griffin & a Matt Bonner sighting all jam packed into tonight's Top 10 Plays of the Night ...



NBA's Top 10 Crossovers: January 2013
Lets count down the 10 best ankle breaking crossovers from January. Which shake and bake crossover was your favorite? ... nba ... nba highlights ...



LeBron tackles Heat fan who hits \$75000 shot!
Check out this wild highlight of one lucky Heat fan who throws up a one-handed hook shot from half court and sinks it for 75000 dollars, much to ...



Top 10 Plays of the Night: January 30th
Check out the 10 best plays from around

Fig: screenshot of Search page

ADDING TAGS TO VIDEOS

- All tags are saved in database and for all of the following tags error handling has been implemented.
- Comments: Users can add comment and maximum text allowed has been set to 512 characters. When the video is being played these comments will be scrolled under the video.

Add Tag

Start Time:

End Time:

Type: ☒ Comment ☐ Image ☐ Map ☐ Web Link

Comment:

MAPS

- Anudeep implemented map tags using Google Maps API for users to drag and drop the marker to their preferred location. Users can also set the title for markers.

Add Tag

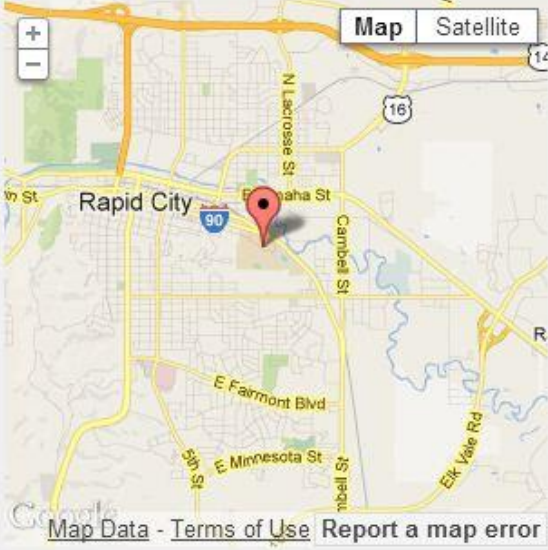
Start Time:

End Time:

Type: ☐ Comment ☐ Image ☒ Map ☐ Web Link

Marker Title:

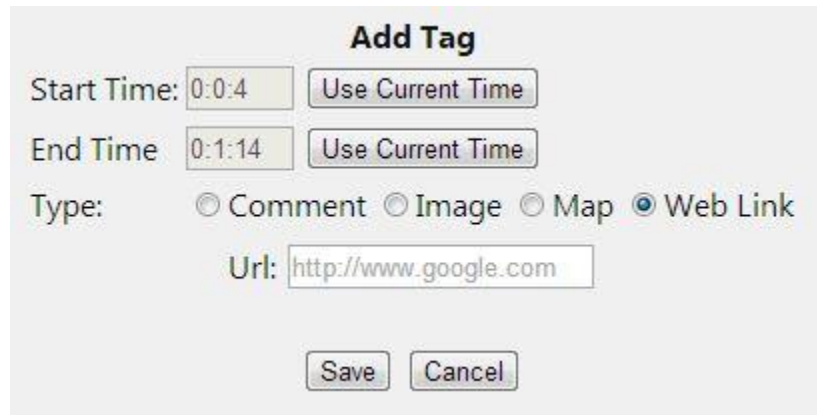
Drag and drop the marker to the preferred location



[Map Data](#) - [Terms of Use](#) [Report a map error](#)

WEB LINKS

- Users can add URL links to their websites.



The screenshot shows a web form titled "Add Tag". It contains the following elements:

- Start Time:** A text input field with the value "0:0:4" and a "Use Current Time" button next to it.
- End Time:** A text input field with the value "0:1:14" and a "Use Current Time" button next to it.
- Type:** A label followed by four radio buttons: "Comment", "Image", "Map", and "Web Link". The "Web Link" radio button is selected.
- Url:** A text input field containing the value "http://www.google.com".
- Buttons:** "Save" and "Cancel" buttons at the bottom.

2.3.5 Sprint 5: Authentication and Finalizing Layouts

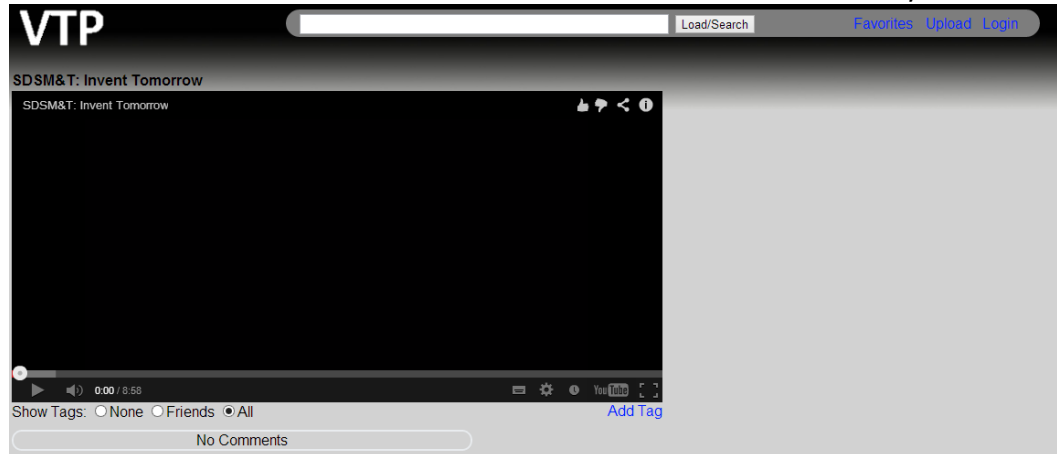
Summary:

- Google authentication continues to be an annoying task. Travis was not getting anywhere with the Google authentication so handed to Anudeep to play with. Travis was able to make a profile page for users to see other users pages and will be changing the favorites page to look more like a profile page for a more individualized page. Travis also made the friend page that displays the user's friends from Facebook that have linked their Facebook accounts to VTP. Travis also fixed some small bugs with the tags when being added to the database. Then Travis also played with some of the CSS in preparation of adding themes to the web pages. Anudeep integrated Vimeo player and wrote required code to add and display tags, and also combined the search and load bar to search YouTube videos or load videos from either of the sources. Anudeep added feature to link Google and Facebook accounts.

Sprint 5 Burndown

CSS changes (later cut due to time restrictions)

- This one of the themes that a user would be able to select for the GUI layout.



Made CSS changes to the forms to make them look better.

A screenshot of a web form titled 'Upload Video: Step 1'. The form is set against a light gray background. At the top, there is a search bar with 'Load/Search' and links for 'Favorites', 'Upload', and 'Indelam'. The form fields include: a 'Title' field with the text 'Testing video Upload'; a 'Description' field with the text 'This is my description for video upload test #14'; a 'Category' dropdown menu currently showing 'Film'; and a 'Keywords' field with the text 'test, upload, fake'. A 'Proceed to Step 2' button is located at the bottom of the form.

Load/Search Favorites Upload Indelam

Upload Video: Step 2

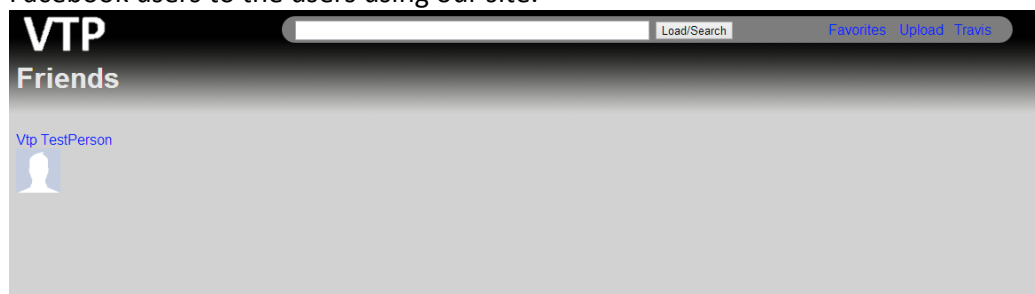
Title Testing video Upload

Description This is my description for video upload test #14

Upload Video No file chosen

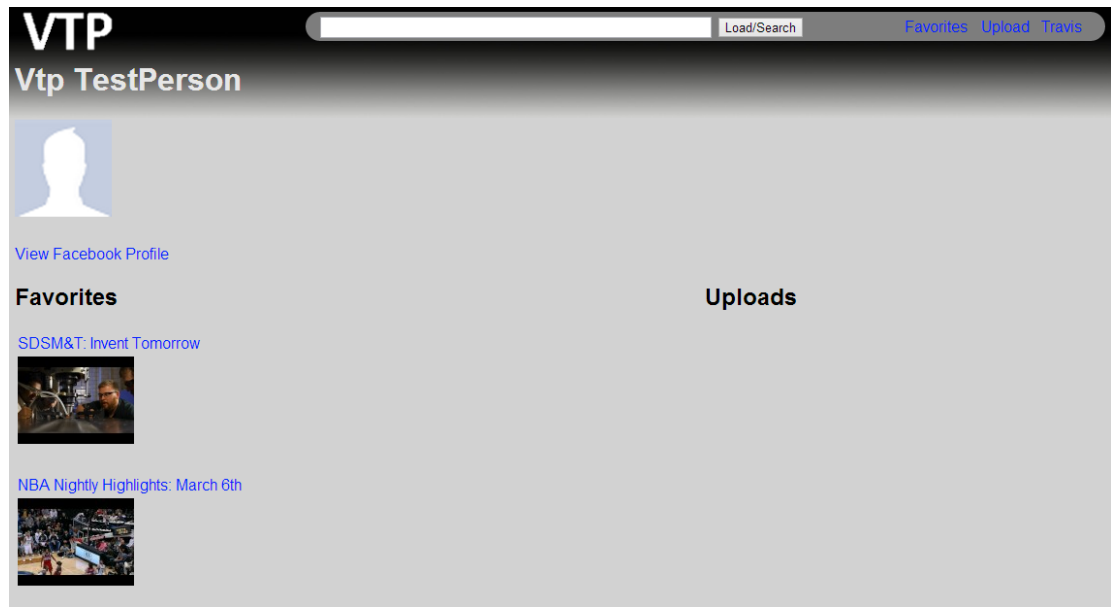
Friends page

- This page is able to select one of the friends that you have from Facebook that is also using the VTP site. This is a simple query to our database that searches for the people have linked their Facebook ID's to our site. When a user is logged in we can ask Facebook who their friends are, this then allows us to cross reference the Facebook users to the users using our site.



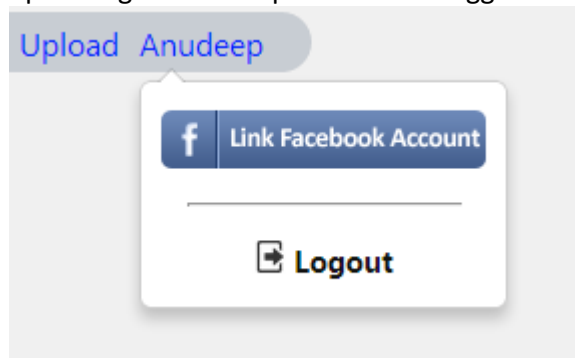
Profile Page

- This page contains the videos that a user has favorited as well as all of the videos that a user has ever uploaded to YouTube. We are currently storing the favorites on our site. Then for the Uploads we will be able to use the YouTube API to do a query to YouTube for the upload listing. The profile picture is the one that is from Facebook.



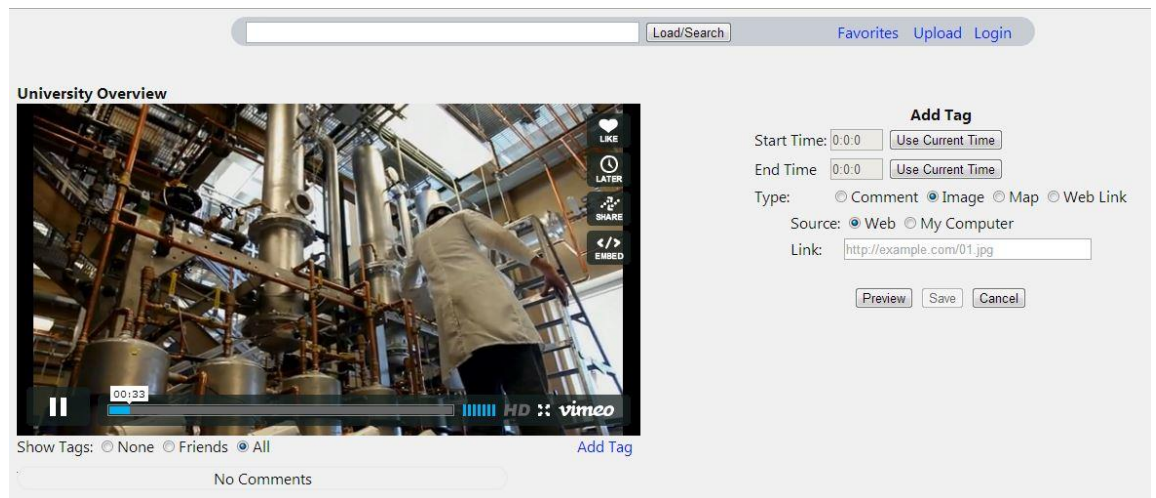
User Authentication

- Anudeep made changes to the user authentication method, now users can login into Google and link their Facebook account or the other way around. This gives the ability for users to upload videos into YouTube and tag, share them with Facebook friends. However due to the restrictions of Google authentication for offline access, uploading a video requires users to be logged into the VTP via their Google account.



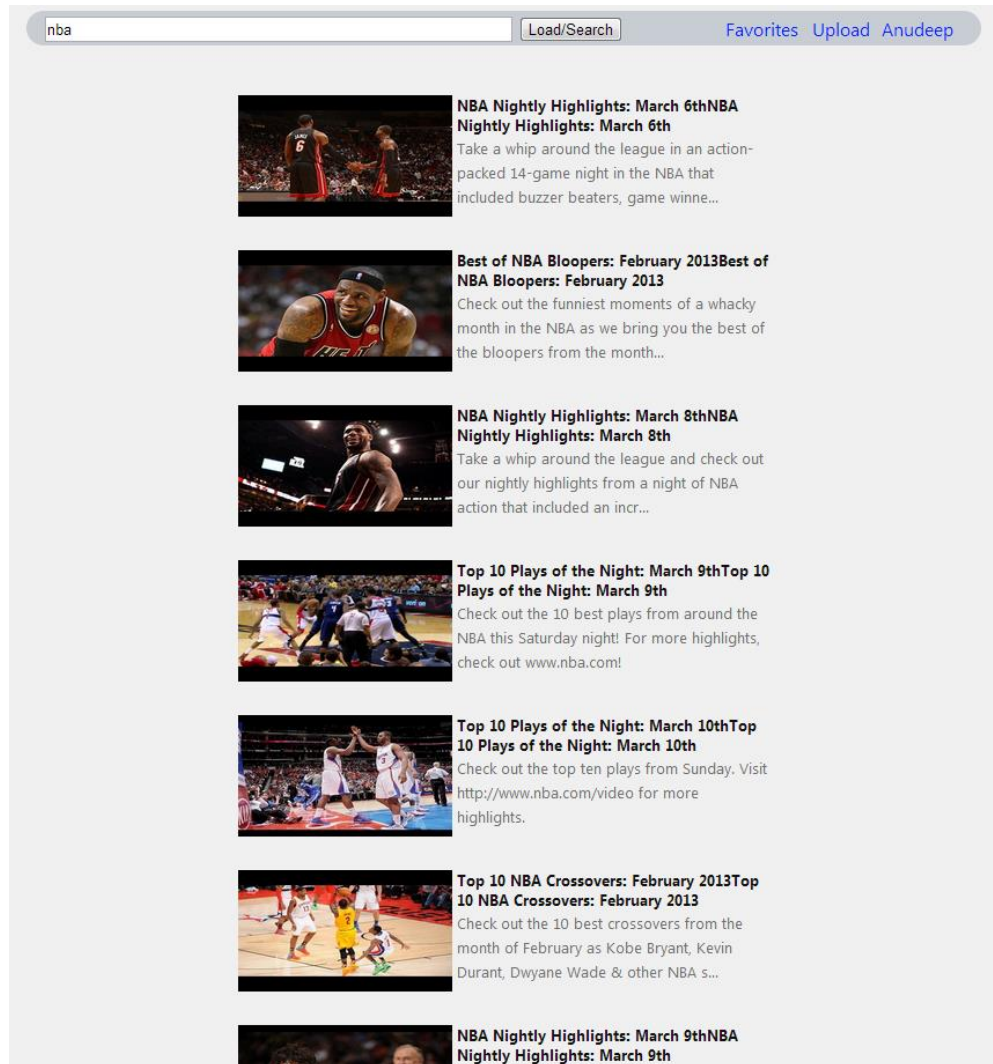
Vimeo Integration

- Vimeo is fully integrated into the system. Anudeep integrated the Vimeo player and added ability to add tags to the video and display them along with the video. Vimeo does not give permissions for embedded players to suggestions after the videos are being played and when user clicks on the suggestion it takes user to the new webpage (Vimeo.com).



Search/Load

- Unified the search and load bar, users can search and load videos from the same bar on the header. If the given input is a YouTube or Vimeo video link then the video is directly loaded if the given input any other search term/text then the YouTube search results will show up.



2.3.6 Sprint 6: Finalization and Testing

Summary:

- Worked out map tag bugs:
- Worked out comment tag bugs (Anudeep)
- Worked out friends page bugs (Anudeep and Travis)
- Worked on presentation materials: (Travis)
 - Poster
 - Fair Video Loop
- Testing: (Travis and Anudeep)
 - Manual
 - Automated

Sprint 6 Burndown

Filters:

Anudeep implemented filtering tags by friends, now users can choose option to see no tags, or display tags added by their friends. We observed that Facebook does not give offline access to the Facebook friends so we started saving the VTP userIds of Facebook friends in DB and update those on every successful Facebook login. So users signed in using Google accounts can also use friends filter for tags.

Friend page bugs

There was originally a bug with the friends page that would cause a user that was logged with their Google account to see a PHP exception to be thrown. The initial fix was to make the user login to Facebook again to see friends. However this was a little annoying for the user so we keep a friends list within the database to have offline access to the friend list.

Google Map Bugs:

Fixed bugs found on the map tags, rectified the offset observed on the map positioning while displaying the markers. Changed the default zoom level on requested of the client.

Worked out comment tag bugs

Fixed bugs caused when a user enters special characters in comment.

Presentation materials

Travis designed the poster and the Fair Video Loop. Anudeep brought the screen for displaying the video loop. Travis brought an extra tablet to run the video loop.

Testing

For testing results check the testing section of the design document. There will be a section for Automated and Manual tests that were performed. Fixed all the bugs found during the testing process.

3.0 Requirements

For the detailed list of requirements check the Software Requirements Document. Here is the overview of requirements (User stories that were initially provided).

- Play the videos within a web application
- Make the application have the ability to play local videos
- Possible integration with sling.com -- *unimplemented*
- The first time viewing of the video should store all the information of the video that is being played – *unneeded*
- Design some kind of login system that allows us to know who one's friends are
- We need to be able to pause a video at any one moment and place one of many kinds of tags
- We have a set of predefined Tags to choose from
- Need to have the ability to change the list of tags
- We need to set up a filter system to filter out tags placed on a video

4.0 Design and Implementation

This section is used to describe the design details for each of the major components in the system. This section is not brief and requires the necessary detail that can be used by the reader to truly understand the architecture and implementation details without having to dig into the code.

4.1 Client Browser

4.1.1 Technologies Used

Browsers Supported:

- Most latest browsers are supported
- Target/development browser: Google Chrome 24 ,25 ,and 26
- Tested Browsers: Internet Explorer 10, Mozilla Firefox 20

4.1.2 Component Overview

Browser Features:

- JavaScript needs to be supported and enabled
- Must have current version of flash player
- Must have support for HTML5

4.1.3 Sprint Overview

Sprint I:

Started development in Google Chrome

Sprint II:

Continued development in Google Chrome

Sprint III:

Checked Internet Explorer compatibility – almost everything seemed to function correctly

Sprint IV:

Continued development in Google Chrome

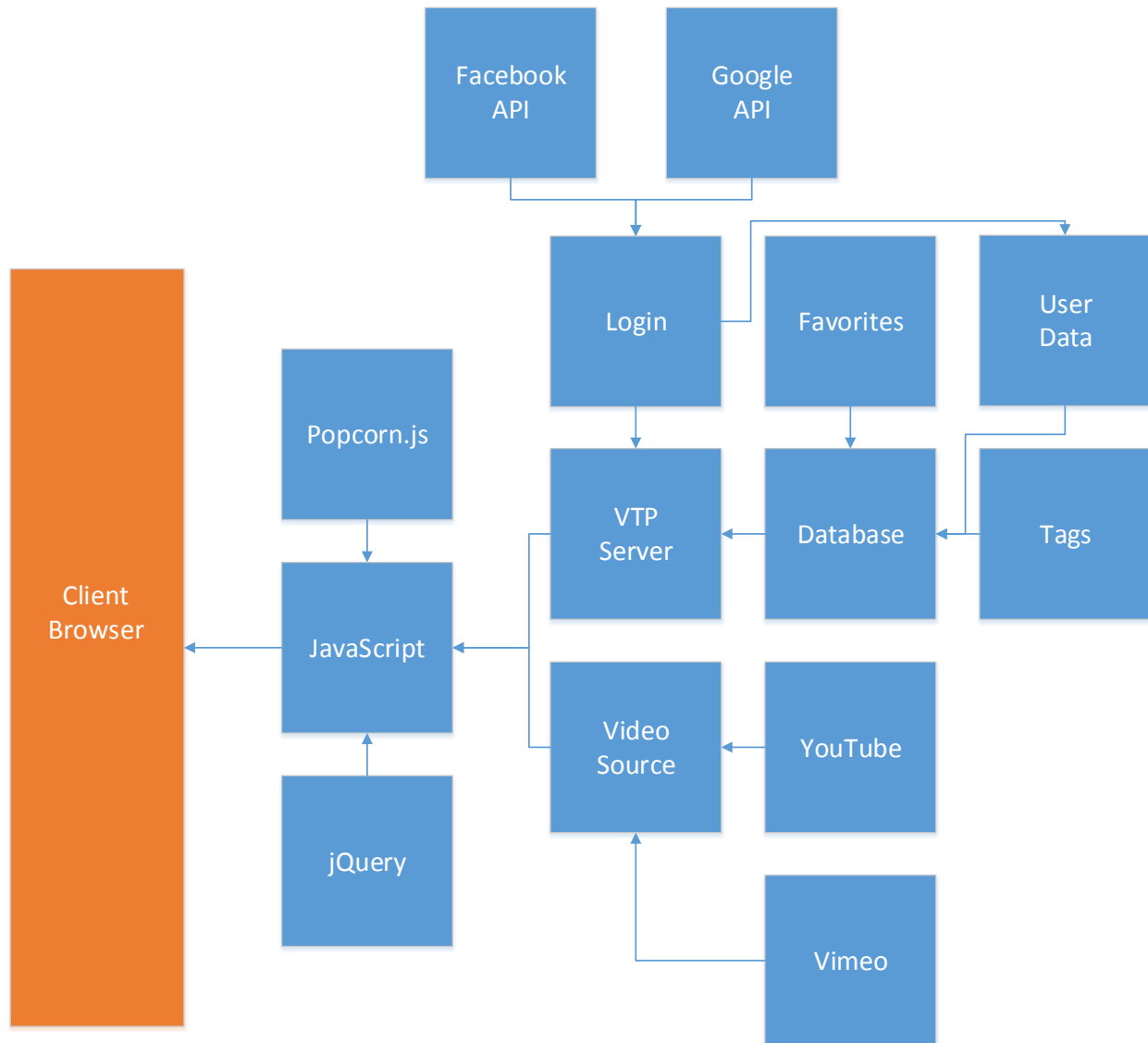
Sprint V:

Continued development in Google Chrome

Sprint VI:

Continued development in Google Chrome and tested other browsers

4.1.4 Architecture Diagram



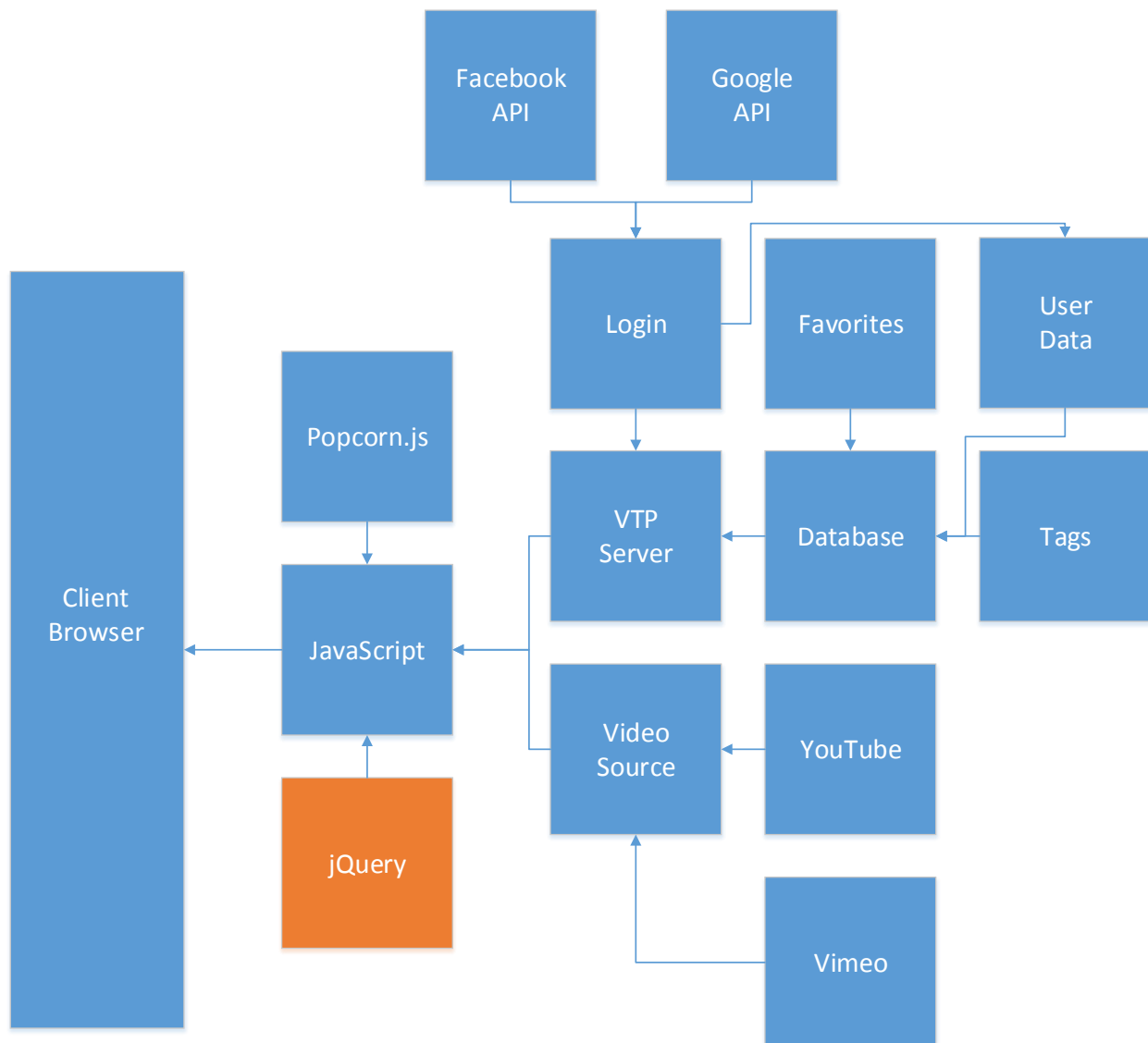
4.2 jQuery

4.2.1 Technologies Used

jQuery:

- A feature rich JavaScript library
- Allows for simpler dynamic HTML traversal/modification
- Allows for simpler dynamic animation and Ajax coding
- Multi-browser support

4.2.2 Architecture Diagram



4.3 Popcorn.js

4.3.1 Technologies Used

Popcorn.js:

- A JavaScript framework for editing media in real time
- Able to interact with media based on timeline
- We use it to control the main functionality of our site by displaying tags at specific times
- Support for YouTube and Vimeo
- Checks media in fractions of a second

4.3.2 Component Overview

Features:

- Play local videos in HTML5 and interact with them
- Play both HTML5 and flash videos from YouTube and Vimeo.
- Triggers JavaScript functions when video is on a particular time.

4.3.3 Sprint Overview

Sprint I:

Embed videos from 3rd party sources.

Sprint II:

Implement feature to play videos from YouTube using Popcorn.js.

Link video player to HTML elements to display hard-coded tag information.

Use Database to save and retrieve tags.

Sprint III:

Implement feature to play videos from other possible sources.

Display multiple tag types simultaneously.

Sprint IV:

Map tags added using Popcorn.js

Web Link tags added using Popcorn.js

Comment tags added using Popcorn.js

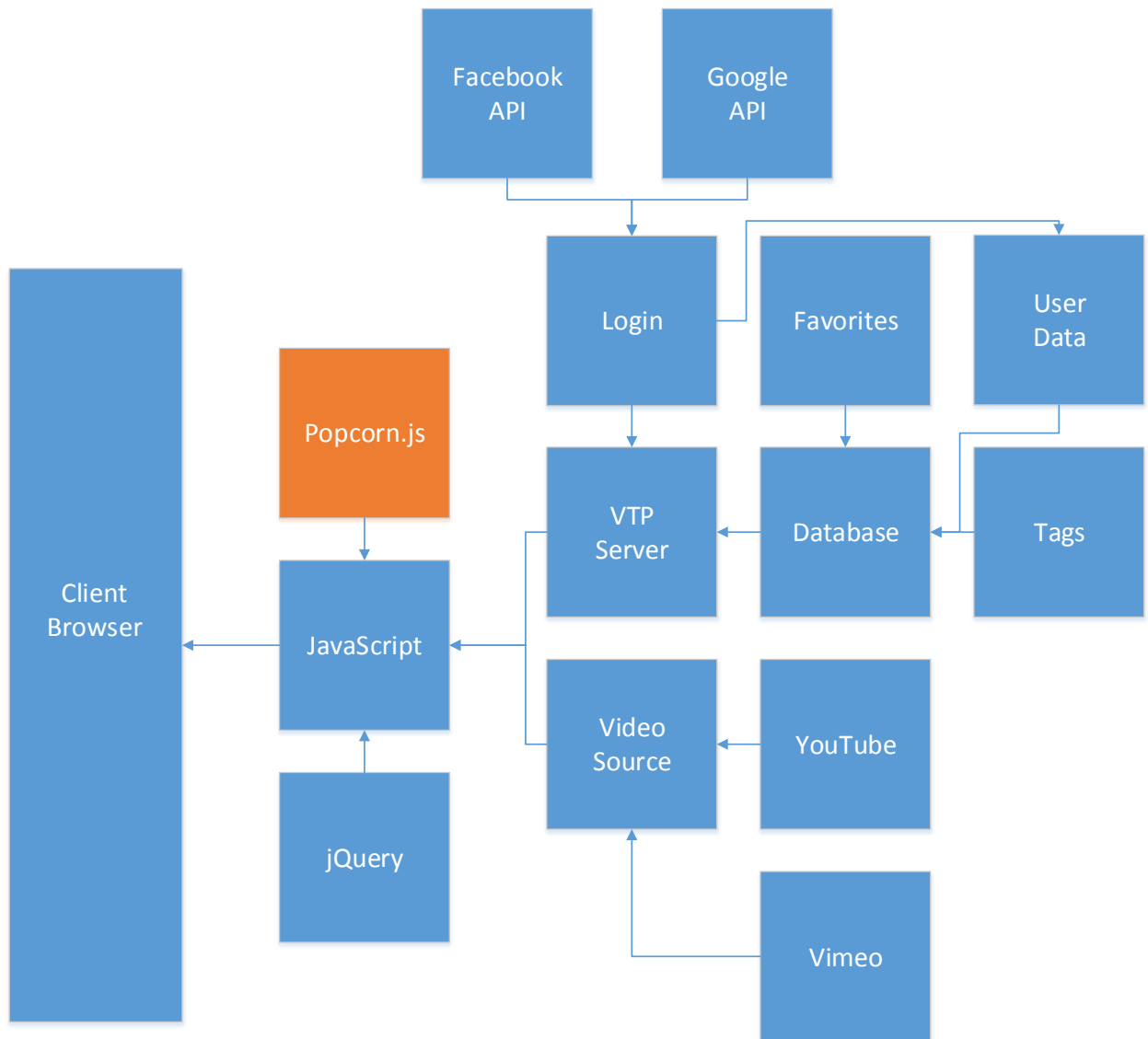
Sprint V:

Vimeo integration

Sprint VI:

Fixed bugs with current tags.

4.3.4 Architecture Diagram



4.4 Database Management

4.4.1 Technologies Used

Created DbConnector class in PHP to interact with MySQL Database. All queries are written as individual functions in DbConnector class.

4.4.2 Component Overview

Features:

- Connect to database and authenticate
- Execute frequent queries from functions
- Disconnect when PHP execution is completed

4.4.3 Sprint Overview

Sprint I:

- Start database design

Sprint II:

- Implement Database design.

- Write DbConnector class

Sprint III:

- Add queries as functions in DbConnector class

Sprint IV:

- Added Favorites table and functions

- Continued to add functions to the DbConnector

- Minor modifications to the database design

Sprint V:

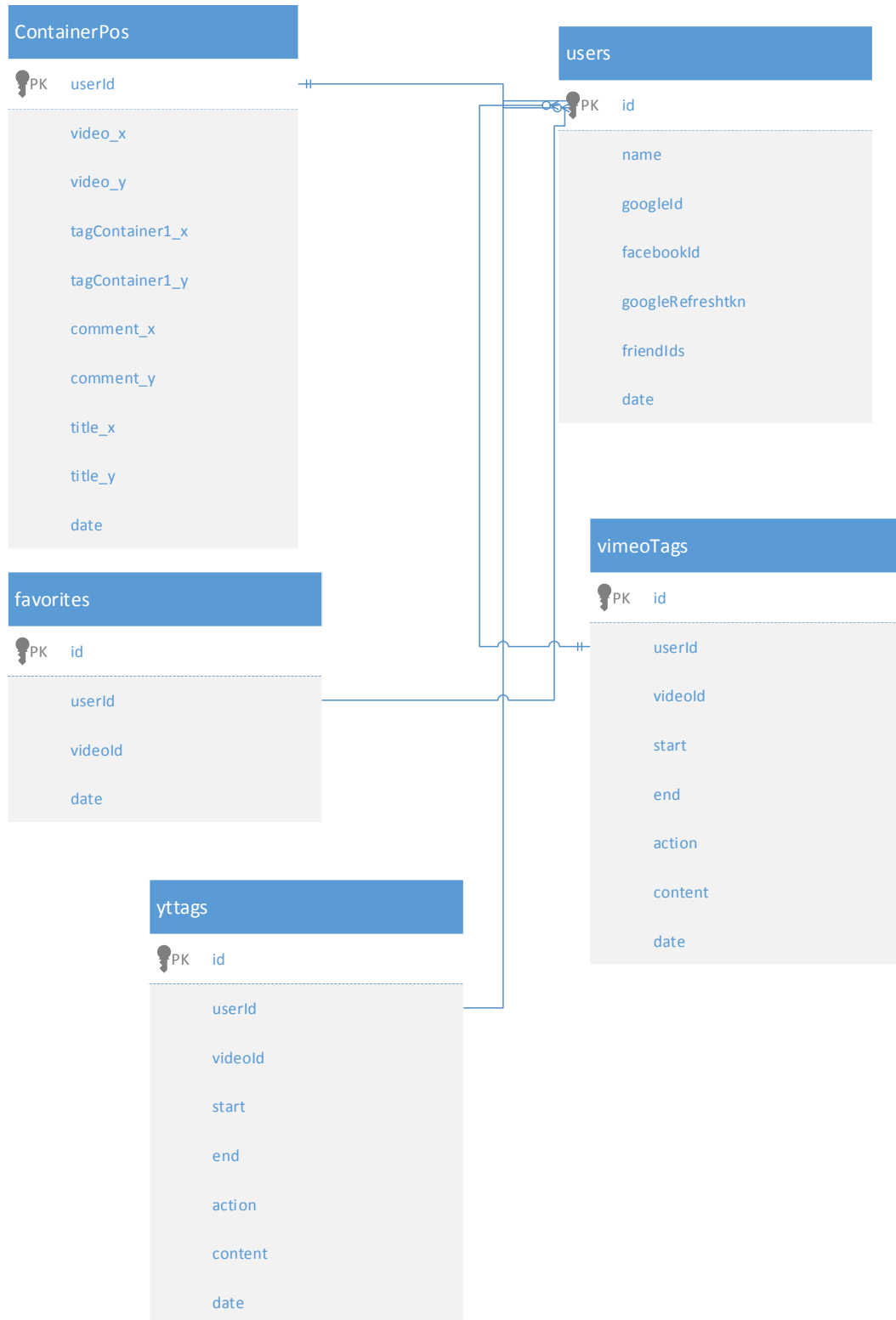
- Separated Video tags into YouTube tags and Vimeo Tags

- Added containerPos table for storing the positions of the components on the main page (this feature was not implemented fully due to time restrictions).

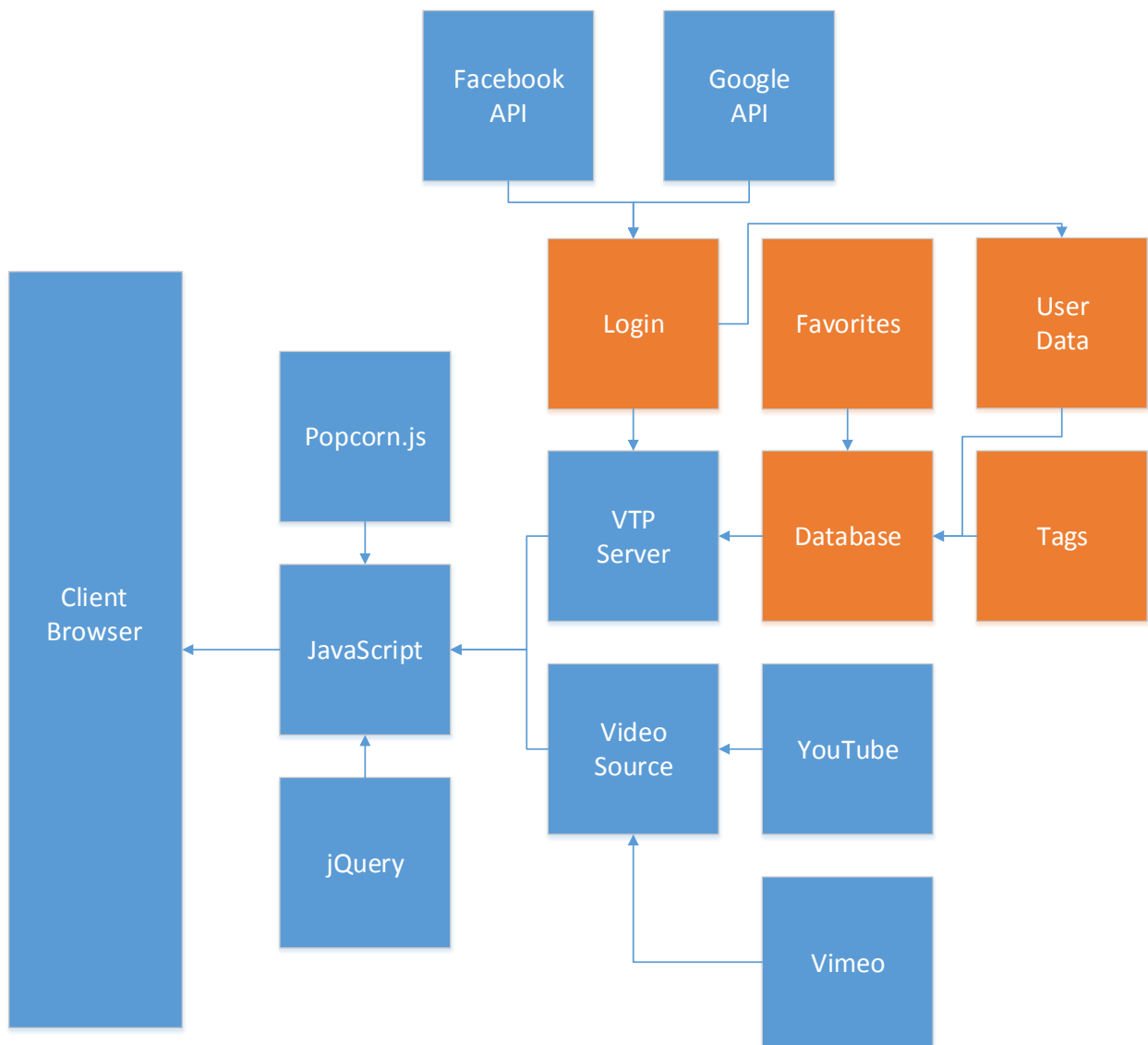
Sprint VI:

- Added a friend listing instead of having to be logged into Facebook you can see your friends from the last time you were logged in to Facebook.

4.4.4 Database Schema



4.4.5 Architecture Diagram



4.5 User Authentication

4.5.1 Technologies Used

Implement user authentications using Google API PHP Client and Facebook SDK.

4.5.2 Component Overview

Features:

Facebook SDK:

- Authenticate users; access basic information

- Get friends list to compare with the VTP database for filtering video tags

Google API PHP Client:

- Authenticate user accounts, access basic information

- Upload videos into their YouTube account. (Requires YouTube Data API)

4.5.3 Sprint Overview

Sprint I:

- Research on both the API's

Sprint II:

- Implement both API and save basic information into Database.

Sprint III:

- No major changes

Sprint IV:

- Used Google API to upload videos to YouTube

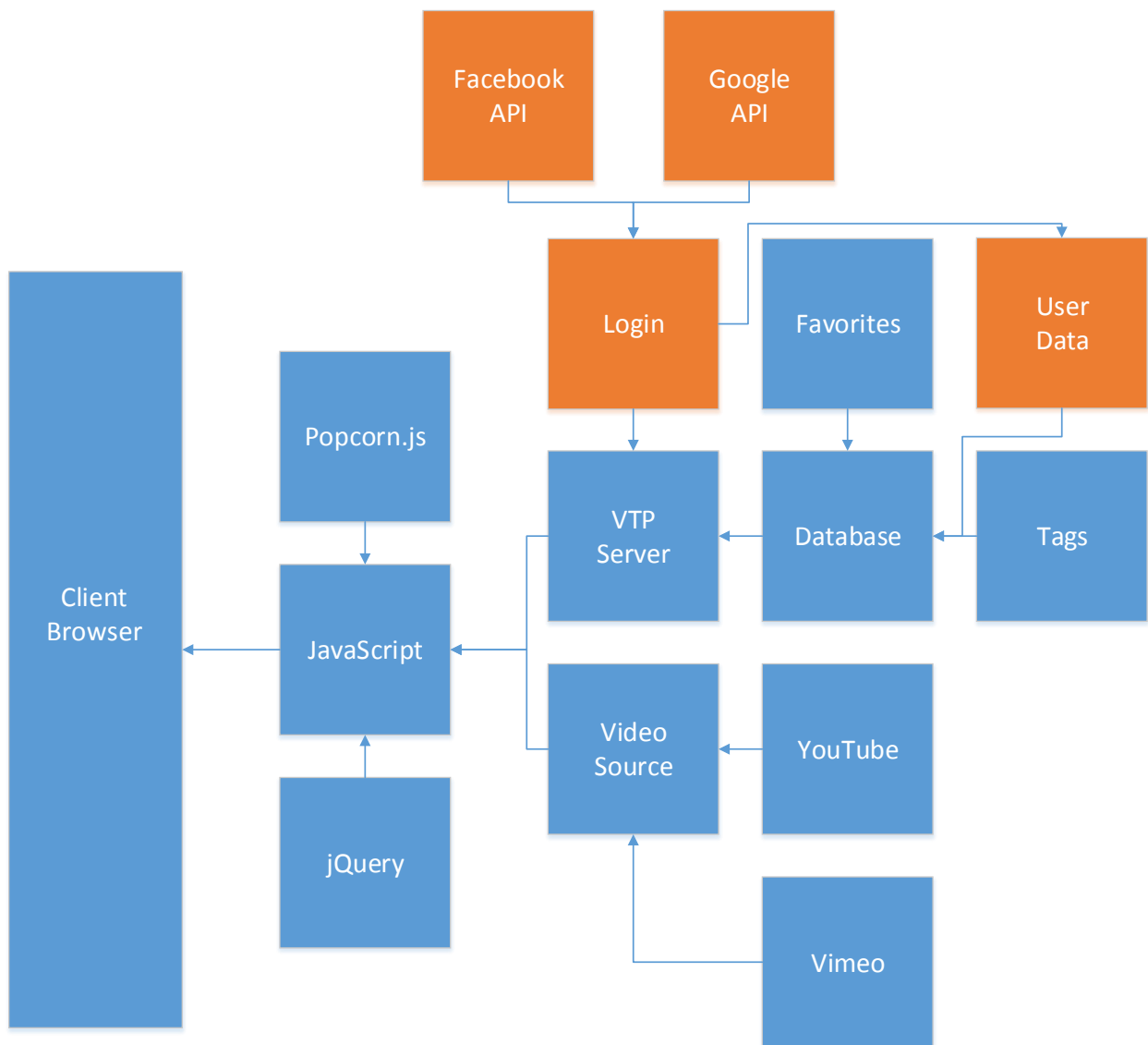
Sprint V:

- Modified and Finalized the YouTube Upload

Sprint IV:

- Fixed Known Bugs with APIs

4.5.4 Architecture Diagram



4.6 Video Source

4.6.1 Technologies Used

Popcorn.js:

- A JavaScript framework for editing media in real time
- Able to edit media based on timing of the media
- We use it to control the main functionality of our site by displaying tags at specific times
- Support for YouTube and Vimeo
- Checks media in fractions of a second

Videos are played from the following sites:

- YouTube
- Vimeo

4.6.2 Sprint Overview

Sprint I:

Able to use YouTube embedded videos

Sprint II:

Research on how to edit embedded videos

Sprint III:

Only supporting YouTube

Sprint IV:

Only supporting YouTube

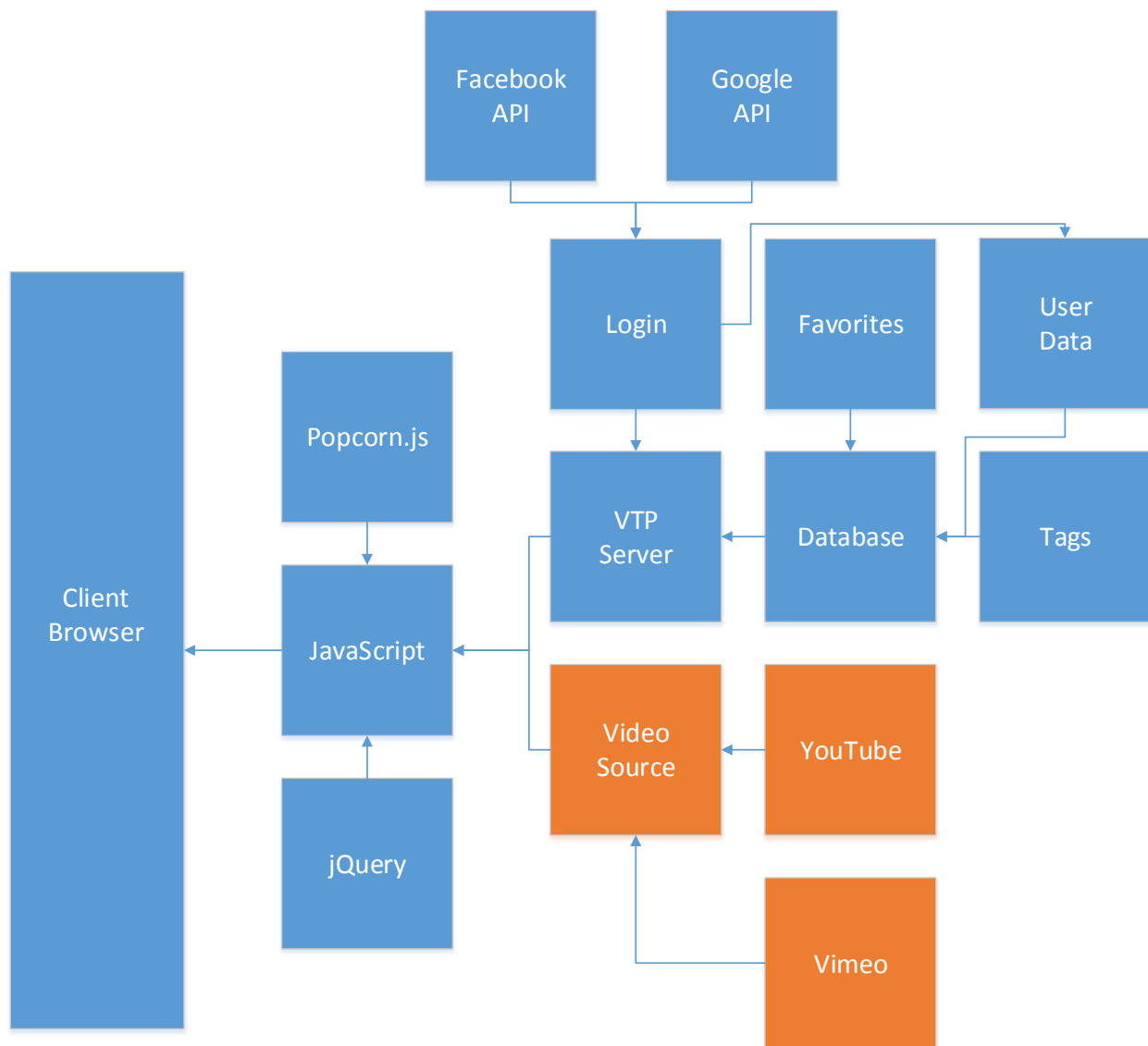
Sprint V:

Added support for Vimeo

Sprint VI:

Finalized and tested different videos from different sites.

4.6.3 Architecture Diagram



5.0 System and Unit Testing

5.1.1 General Testing

5.1.1.1 Popcorn.js

We will need to make sure the Popcorn.js is behaving as we expect, by checking the timings and making sure that they are consistent with what is happening on-screen.

5.1.1.2 Videos

We need to make sure that the videos are being displayed correctly and are sized properly. The videos need to be able to be played while tags are being updated.

5.1.1.3 Database

We need to make sure the database is secure as well as able to be responsive for the users accessing it.

5.1.1.4 Host Server

Security is the major concern that is associated with the Host Server.

5.1.1.5 JavaScript

We will be using the JSLint to test the quality of the JavaScript functions.

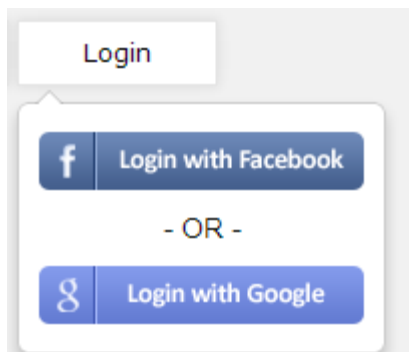
5.1.1.6 Range of Browsers

We will need to test the functionality of the different browsers on our site.

5.1.2 Manual Test Cases

The overall purpose of the test cases is to try to break the code. So here are the methods that are used and the results of trying to break the code. These are test cases that will need to have human interaction in order to complete them and see if they are correct.

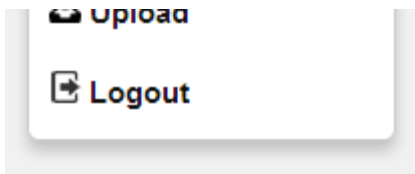
Login



Many early test were promising when a user could only login with either Facebook or Google. Once the two need to be linked there was a number of issues that needed to be addressed. One issue is

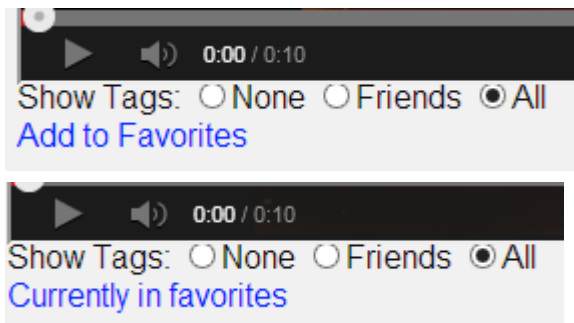
that Google authentication cannot be easily used at the same time the Facebook Authentication is active. Another problem was that users could only login to Google first if they wanted to link their accounts. One problem that can arise is if a user is logged in but a database admin deletes their account, they are still logged in because the browser will store the login information and will attempt to access account specific things with and get errors however still seem to be logged in. This problem is not critical because there should be no cases where a person is deleted from the database.

Logout



There were no problems with the logout because it only deletes things that are on the client side that tell the user that they are logged in. The user account information is all saved on the database server and is only accessed the next time the user is logged in.

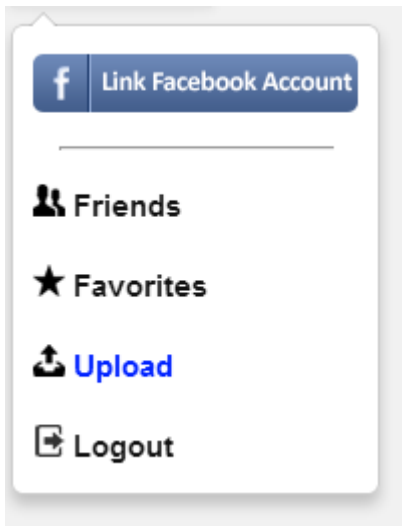
Add to favorites



The functionality is as simple as clicking a button to a user, however there was an early problem where a user could click the button multiple times and add the favorite multiple times to the database. However there are currently no known bugs with this feature.

- **If no favorites**
 - There are no bugs to report with adding to the empty database
- **If one favorite**
 - There are no bugs to report with adding to a database with one item.
- **Two or more**
 - There seems to be no problems with the addition into the database with two or more items.

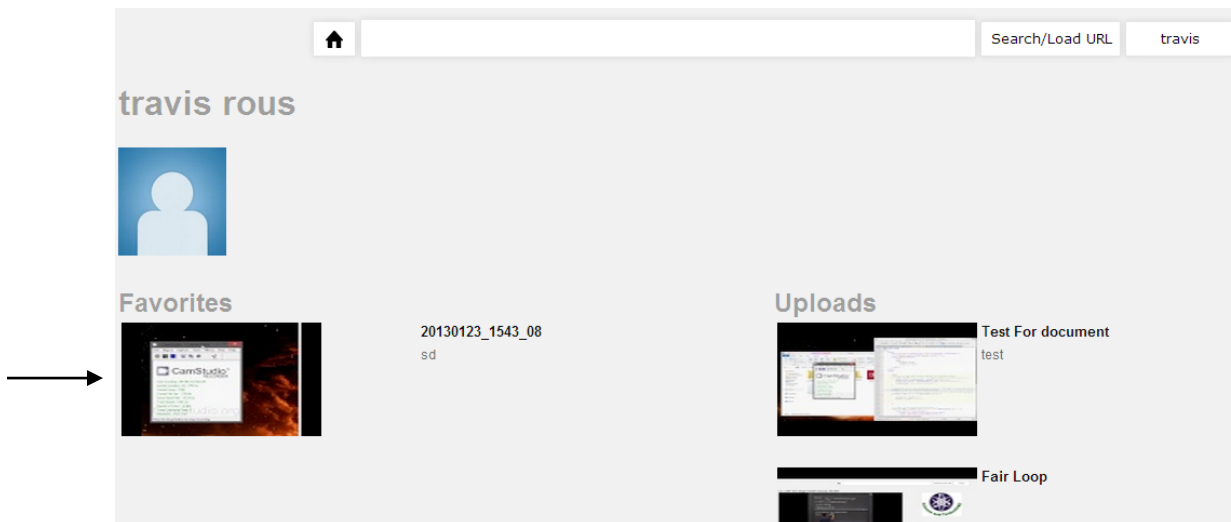
Link accounts



As stated before there were problems with the direction in which you linked the accounts.

- **Facebook to Google**
- **Google to Facebook**
 - There seems to be no bugs with the linking of a Facebook account to A Google account.

Open a favorite

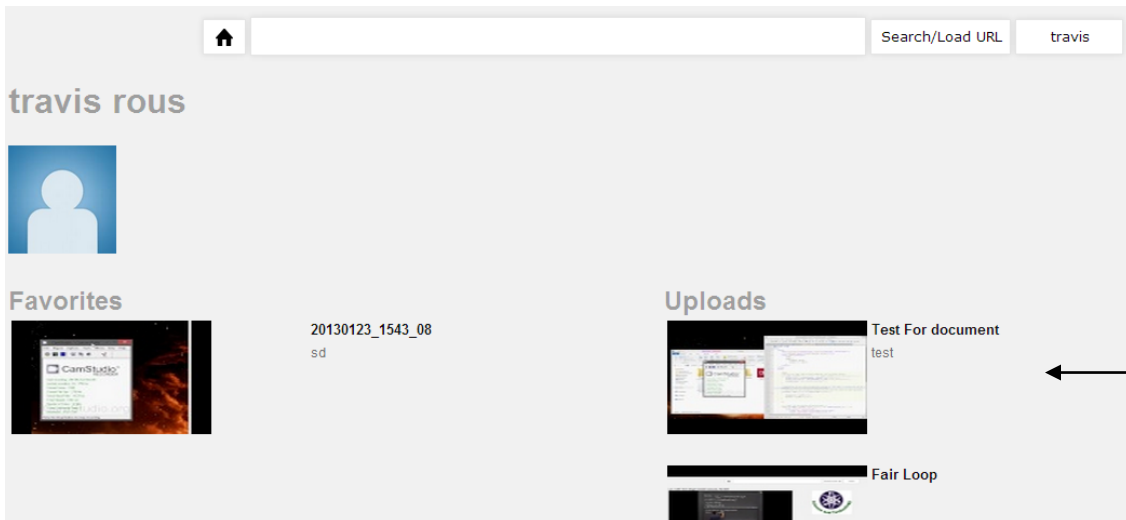


Opening a favorite consists of going to the favorites page and clicking on the link of the favorite you want to see. The following are just the extreme cases that need to be double checked.

- **Open the first one**
 - There seems to be no problem choosing the first one.
- **Open the last one**
 - There seems to be no problem choosing the last one in the list.
- **Check for only one favorite**
 - There seems to be no problem choosing a favorite to open if there is only one favorite to choose from.

- **The empty case**
 - When a user has no favorites then the user is notified that they have no favorites.

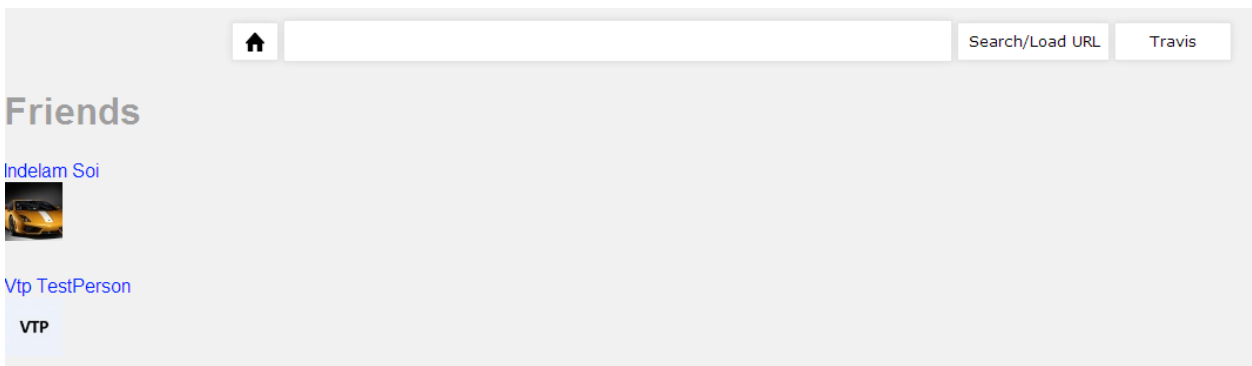
Open an upload



Opening an upload consists of going to the favorites page and clicking on the link of the upload you want to see. The following are just the extreme cases that need to be double checked. Also the user is required to be logged into Google in order to see the uploads that they have because they pull directly from YouTube.

- **Open the first one**
 - There seems to be no problem choosing the first one.
- **Open the last one**
 - There seems to be no problem choosing the last one in the list.
- **Check for only one upload**
 - There seems to be no problem choosing an upload to open if there is only one upload to choose from.
- **The empty case**
 - When a user has no uploads then the user is notified that they have no uploads.

Open a Friend



This test is much like the test cases that go with the favorites. These need to be checked by going to the user's friends page then choosing the friends.

- **Open the first one**
 - There seems to be no problem choosing the first one.
- **Open the last one**
 - There seems to be no problem choosing the last one in the list.
- **Check for only one friend**
 - There seems to be no problem choosing a friend to open if there is only one friend to choose from.

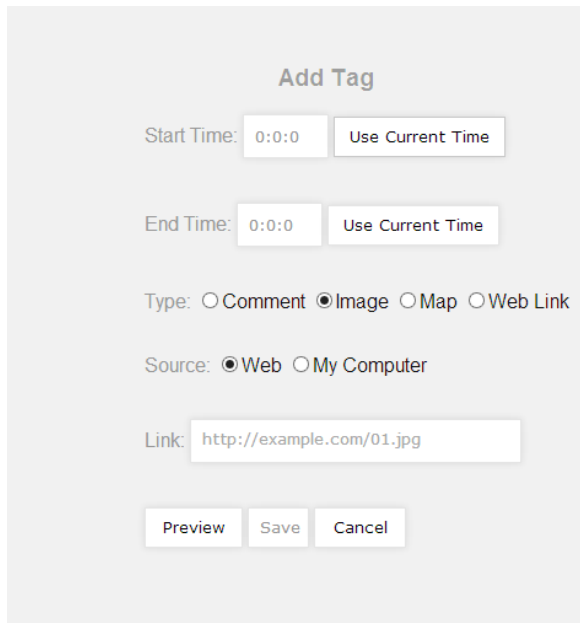
Upload a Video

The screenshot shows a web interface for uploading a video. At the top, there is a navigation bar with a home icon, a search bar labeled 'Search/Load URL', and a user name 'travis'. Below this, the main heading is 'Upload Video: Step 1'. The form contains the following elements: a 'Title' text input field, a 'Description' text area, a 'Category' dropdown menu currently set to 'Film', and a 'Keywords' text input field. Below these fields is a checkbox labeled 'Allow VTP to upload this video into my YouTube account'. At the bottom of the form are two buttons: 'Proceed to Step 2' and 'Cancel'.

Upload is definitely one of the most difficult parts of the project and contained many flaws. Here we will check for the errors that a user could generate.

- **Check for incorrect format**
 - This test fails because YouTube does not return an error to our application. This is because YouTube attempts to convert any file format that it receives to a useable format. So as long as YouTube gets a file it will attempt to process it. The user will be notified when they try to play the video that the video is invalid.
- **Check for duplicate video**
 - This test fails because Google does not return an error to our application. So the only way for the user to find out is to try to play the video then YouTube tells the user through the embedded player that it is a duplicate video.
- **Just a sample video to make sure it works**
 - Seems to work Correctly

Adding tags to a video



The screenshot shows a form titled "Add Tag" with the following fields and options:

- Start Time:** A text input field containing "0:0:0" and a button labeled "Use Current Time".
- End Time:** A text input field containing "0:0:0" and a button labeled "Use Current Time".
- Type:** Radio buttons for "Comment", "Image" (which is selected), "Map", and "Web Link".
- Source:** Radio buttons for "Web" (which is selected) and "My Computer".
- Link:** A text input field containing "http://example.com/01.jpg".
- Buttons:** "Preview", "Save", and "Cancel" at the bottom.

Some limitations on the tags are that they cannot be added at sub-second level. A video with a large number of tags will have tags that are stored in a scrollable box. The timing on the addition of the tags does not allow for impossible timings such as smaller ending times than starting times. The tags are also impossible to automate for the testing because there are too many time dependent pieces required to test them.

- **Comment Tag**
 - Tags are added seem to show up when they supposed to.
- **Hyperlink Tag**
 - Tags are added and seem show up at the correct timings.
- **Image Tag**
 - **From The Web**
 - Allows users to add an image using nothing more than a URL the preview allows the use to see if the link is a good image link.
 - **From Users Computer**
 - Tags are added and seem to show up at the correct timings.
- **Map Tag**
 - The tags seem to be added to the correct location and at the correct timings. Sometimes Google maps are timed out before they load. But the page just needs to be refreshed.

5.1.3 Automated Test Cases

Once a user is logged into the site many of the actions can be automated using a tool called selenium. Selenium is used to automating web applications. All though most of the Manual test cases could be automated using this tool they need to have interaction with the database and the tester would need to check for correct results both on screen and on the database. So the test will consist of navigating the website and adding random videos to the database for different users and the scripts will be located in the Appendix. The scripts only failed when the website could not load everything in time

5.2 Dependencies

JSLint for testing the individual JavaScript functions to make sure that they are living up to some standards. Most of the JavaScript is untestable to this site because it does not except jQuery code in the JavaScript.

<http://www.jshint.com/> --- the webpage for the tester.

Selenium is used to automating web applications.

<http://docs.seleniumhq.org/> --- for more information on selenium

5.3 Test Setup and Execution

The setup of the JSLint is go to the web sited and paste in the code that you want to test. (There are also command line versions for download)

Selenium is a Firefox add-on that contains an IDE for creating the testing scripts that automate a web site.

6.0 Development Environment

The basic purpose for this section is to give a developer all of the necessary information to setup their development environment to run, test, and/or develop.

6.1 Development Tools

For the current implementation the Developer will need to have access to the FTP server that we host the code on. The Developer will also need an editor of their choosing. The server has many database admin functions that a developer can use as well. Besides those few tools the developers will need to know how Facebook and Google APIs work, as well as, basic web development abilities with PHP, HTML, JavaScript, and jQuery. Our recommend tools are as follows.

We used FileZilla for our FTP transfers.

For the code editing we choose our favorite code editor, such as Notepad++ or Sublime Text 2.

For database management we used PhpMyAdmin which is in the list of database admin tools on the hosting site.

6.2 Source Control

For the source control we are using the GitHub repository that uses a windows interface to access the repository. GitHub is completely open to the public to see unless you pay for a private service. A few problems with the source control was that we could not directly host the code on GitHub because we need the code to be located in a place that the team could easily get to at any point in the day and from anywhere. We found a free domain where we hosted the server. Our typical strategy for development was to have two copies of the project so that we could both work on it. Then we could copy our code to GitHub and resolve any conflicts that happened between our two copies. Using this method of development caused some pauses of work on GitHub even though there was still work being done.

6.3 Build Environment

The project will need to be deployed on a PHP server and a MySQL server will need to have the tables setup using a .sql script that will be sent with the project. There will need to be a connection between the server and the PHP using a database connection in PHP.

7.0 Release | Setup | Deployment

Upon release the product will be hosted on a server that will have the bandwidth high enough to support thousands of users at a time. The project will need to be deployed on a PHP server and a MySQL server will need to have the tables setup using a .sql script that will be sent with the project. There will need to be a connection between the server and the PHP using a database connection in PHP.

Appendix I: Supporting Information and Details

This section contains the testing scripts that are used by selenium.

There are three that run in the order that they are presented here. They are used in this order because if the first one fails then the rest will fail as well. The last script is an all-encompassing test that goes through the whole site multiple times.

Basic Functions

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://VTP.host-ed.me/VTP/src/" />
<title>BasicFunc</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">BasicFunc</td></tr>
</thead><tbody>
<tr>
<td>open</td>
<td>/VTP/src/index.php</td>
<td></td>
</tr>
<tr>
<td>click</td>
<td>//input[@value='Travis']</td>
<td></td>
</tr>
<tr>
```

```

        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Travis']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Travis']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[3]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Travis']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>link=Indelam Soi</td>

```

```

        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Travis']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>xpath=(//span[@id='title'])[3]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>css=area[title=&quot;MIT&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Travis']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>

```



```

<tr>
  <td>type</td>
  <td>id=query</td>
  <td>mit calculus</td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>id=search-button</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>id=title</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>css=area[title=&quot;MIT&quot;]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>css=#actualMap &gt; div &gt; div.gmnoprint &gt; div.gmnoprint &gt; div[title=&quot;Zoom
out&quot;]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>xpath=(//img[contains(@src,'https://maps.gstatic.com/mapfiles/szc4.png')])[2]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>css=#actualMap &gt; div &gt; div.gmnoprint &gt; div.gmnoprint &gt; div[title=&quot;Zoom
out&quot;]</td>
  <td></td>
</tr>

```



```

        <td>css=area[title=&quot;MIT&quot;]</td>
    </td></td>
</tr>
</tbody></table>
</body>
</html>

```

Add To Favorites

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://VTP.host-ed.me/VTP/src/" />
<title>New Test</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">New Test</td></tr>
</thead><tbody>
<tr>
        <td>open</td>
        <td>/VTP/src/index.php</td>
        <td></td>
    </tr>
<tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
<tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>
<tr>

```

```

        <td>type</td>
        <td>id=query</td>
        <td>mit calculus</td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>id=description</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=favLink</td>
        <td></td>
    </tr>
</tbody></table>
</body>
</html>

```

Stress Test

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://VTP.host-ed.me/VTP/src/" />
<title>New Test</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">New Test</td></tr>
</thead><tbody>

```

```

<tr>
  <td>open</td>
  <td>/VTP/src/index.php?ytUrl=http://www.YouTube.com/watch?v=jblQW0gkx0</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=#logoutDropdown &gt; ul &gt; li</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>

```

```

        <td>css=input[type=&quot;button&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>
    <tr>
        <td>type</td>
        <td>id=query</td>
        <td>mit calculus</td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=img[alt=&quot;Lec 1 | MIT 18.01 Single Variable Calculus, Fall 2007&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>link=Add Tag</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Cancel']</td>
        <td></td>
    </tr>

```

```

</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>link=Travis Rous</td>
  <td></td>
</tr>
<tr>

```

```

        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=input[type='button']</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>

```



```

        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=#logoutDropdown &gt; ul &gt; li</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=input[type=&quot;button&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>
    <tr>
        <td>type</td>
        <td>id=query</td>
        <td>mit calculus</td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>

```

```

<tr>
  <td>click</td>
  <td>id=search-button</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=img[alt=&quot;Lec 1 | MIT 18.01 Single Variable Calculus, Fall 2007&quot;]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>link=Add Tag</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='Cancel']</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>

```

```

        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=span.link</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link=Travis Rous</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=span.link</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
    <td></td>

```

```

</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=input[type=&quot;button&quot;]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=#logoutDropdown &gt; ul &gt; li</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>

```

```

        <td>clickAndWait</td>
        <td>css=input[type=&quot;button&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>
    <tr>
        <td>type</td>
        <td>id=query</td>
        <td>mit calculus</td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=img[alt=&quot;Lec 1 | MIT 18.01 Single Variable Calculus, Fall 2007&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>link=Add Tag</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='Cancel']</td>

```

```

        <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=span.link</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
    <td></td>
</tr>
<tr>
    <td>click</td>
    <td>//input[@value='VTP']</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=span.link</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>link=Travis Rous</td>
    <td></td>
</tr>

```

```

<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=input[type='button']</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
  <td></td>
</tr>
<tr>
  <td>click</td>

```

```

        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=#logoutDropdown &gt; ul &gt; li</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=input[type=&quot;button&quot;]</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=query</td>
        <td></td>
    </tr>
    <tr>
        <td>type</td>
        <td>id=query</td>
        <td>mit calculus</td>
    </tr>
    <tr>
        <td>click</td>
        <td>id=search-button</td>
        <td></td>

```



```

</tr>
<tr>
  <td>click</td>
  <td>id=search-button</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=img[alt=&quot;Lec 1 | MIT 18.01 Single Variable Calculus, Fall 2007&quot;]</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>link=Add Tag</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='Cancel']</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>
  <td>clickAndWait</td>
  <td>css=span.link</td>
  <td></td>
</tr>
<tr>
  <td>click</td>
  <td>//input[@value='VTP']</td>
  <td></td>
</tr>
<tr>

```

```

        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>link=Travis Rous</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>css=span.link</td>
        <td></td>
    </tr>
    <tr>
        <td>click</td>
        <td>//input[@value='VTP']</td>
        <td></td>
    </tr>
    <tr>
        <td>clickAndWait</td>
        <td>//div[@id='logoutDropdown']/ul/li[2]/span</td>

```

```
        <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>css=input[type=&quot;button&quot;]</td>
    <td></td>
</tr>
</tbody></table>
</body>
</html>
```