CLOUDBUCKET

SEMANTIC MEDIA SEARCH WITH MOBILE CLIENTS USING CLOUD COMPUTING

PROPOSAL

The City College of New York

by

Sahat Yalkabov and Emily Bodden

May 17th, 2013

TABLE OF CONTENTS

EXECUTIVE SUMMARY

In this project a user-friendly mobile-effective media search system will be implemented. This system will use cloud computing to store media and use semantic search to allow efficient searching of media stored on the cloud. The approach to carry out this system has merit above all others as it is using current tools to create an efficient media search system.

This user friendly mobile effective media search system is of great importance as most people own mobile devices and try to manage their time effectively. We want fast and relevant results when using a search feature. Being able to use mobile devices for uploading different kind of files such as video, audio, text and pictures to a cloud storage is very useful. Searching files based on the file content among the cloud is also very handy. Semantic capabilities to automatically tag media files based on their content creates a high standard of search quality for the individual. Files will also be available for download from the cloud.

OBJECTIVE

Create a user-friendly mobile-effective media search system using semantic search and cloud computing. It will incorporate both automatic and manual tagging by allowing the user to add manual tags and also use natural language techniques to add automated tags to the media stored on the cloud. This system will allow efficient searching of media stored on the cloud due to the tagging functionalities used.

BACKGROUND

Dating back to 2001, an article was written in the Scientific American magazine about *The Semantic Web*. [1] This article spoke about a new form of Web content meaningful to computers that would release many new possibilities. Computers at the time were not able to gather meaning from web pages, they instead saw web pages in terms of their layout as routine processing was done. Tim Berners-Lee said the Semantic Web to come would bring structure to the meaningful content of Web pages. The Semantic Web would not be seen as a separation from the Web but instead an extension of the current one.

In 2006, an article called *The Semantic Web Revisited* was released by the IEEE Computer Society. [2] At this point, the Semantic Web was being seen as attainable as its first stirrings began to show. It referenced the Tim Berners-Lee article in saying that initially the web consisted of documents for humans to read then it evolved to one that included data and information for computers to manipulate. The article mentioned that the Semantic Web still had not reached its full potential. The progress of The Semantic Web consisted of deploying languages for shared meaning, this is when the Resource Description Framework (RDF) was introduced. RDF assigns specific Universal Resource Identifiers to its individual fields. URIs identify resources and are crucial the to Semantic Web. When a URI is linked to an Resource it allows individuals to refer to it or retrieve a representation of it.

In 2009, Scientific American posted another article termed *The Semantic Web in Action*. [3] This article once again referred back to the first article done in 2001. It spoke of the technology for Semantic Web and how it is now of age. The tagging systems developed on MySpace and Flickr were

mentioned as this was seen as a big development. However, the two systems did not communicate and this was a drawback. There has been many technologies released to cross this boundary and Friend of a Friend (FOAF) is an example as it uses a Semantic Web vocabulary which describes peoples names, ages, relationships and jobs for finding a common interest among them. This allows information to be posted and connected unlike MySpace and Facebook who can not due to incompatible fields. RDF language and ontologies, which represent the relationships between concepts have matured greatly. The article notes that the Semantic Web is emerging greatly at this point and is making online information more useful than ever.

Towards the Semantic Web [4], written in 2010 speaks of the vision of Tim Berners-Lee coming into play. The Semantic Web would be like a database where everything is organized and categorized which would allow queries to be combined in every way possible. Searching for a specific restaurant and listing different criteria for instance, would be done easily. This database would not be controlled by anyone but the individual themselves. This would allow you to choose your own data. Possible issues were discussed as the individual may not organize their database themselves. The RIF, Rule Interchange Format was introduced and is a new Semantic Web Standard and gives rules for translating between data on different sites.

Companies such as Bing, which Microsoft is said to be investing their money into utilize Semantic Search. In the article *Bing: Re-organizing the Web to Get Stuff Done* [5], Weitz, the Search Director for Bing says that a big part of how Bing directs you to tasks is in detecting intent, also known as semantic search. Semantic search has to determine the meaning of the words as a unit, and in context. The search system has to understand the difference between Jaguar the car, the Florida team,

the animal, or any other jaguar. Bing is using the newsfeed of social services such as Facebook and Twitter to gather them into their results.

Google is another company that utilizes semantic search immensely. A recent article written this year in The New York Times, *Google Introduces New Search Tools to Try to Read Our Minds* speaks about the companies new search tools. [6] When an individual asks certain questions on Google, not only will those questions be answered but also there will be predicted follow up questions which will also be answered for the user. For instance, when asked for the population of India, the population of China and the United States will also be given. This is because Google knows that these are the most common follow up questions.

Semantic media search with cloud computing was done was done in June 2009 by a student whose project entailed the use of a Dell mobile device. [7] The system utilized semantic search by implementing a general purpose semantic API which allowed for the generation of a list of tags for a given file. This allowed tagging software for text files. An algorithm for detected inter-tag relationships was also implemented to suggest related tags for a given user query. Our system will be a lot more current and will be compatible with both Android and iPhone mobile devices. Better technology will also be used which creates a more diverse use of the application as native API's are then incorporated.

PROGRESS

Weekly Timetable for Summer Semester 2013

June 3rd – Beginning of Summer Semester

- Read MongoDB tutorials (Emily)
- Read ElasticSearch documentation on querying (Sahat)

June 10th –

- Read tutorials on parsing documents for MongoDB purposes (Emily)
- Implement mobile app search user-interface (Sahat)

June 17th –

- Learn how to connect MongoDB and ElasticSearch (Emily)
- Connect MongoDB with ElasticSearch (Sahat)

June 24th –

- Storage of manual tags (Emily)
- Use NLTK to automatically create tags from a short text summary (Sahat)

July 1st –

- Index files using MongoDB (Emily)
- Figure out how to parse PDF, DOCX, and other documents to plain text (Sahat)

July 8th –

- Read tutorials on WordNet (Emily)
- Add EchoNest API for music recognition (Sahat)

July 15th –

- HTML tweaking of user interface (Emily)
- Implement OpenCV face detection (Sahat)

July 22nd –

- Run tests on search function (Emily)
- Implement partial support for video file analysis and recognition (Sahat)

July 24th – End of Summer Semester

- Final product testing and polishing (Emily)
- Final product testing and polishing (Sahat)

TECHNICAL APPROACH

The Semantic Media Search project will allow users to upload files from their mobile devices to a remote storage on the cloud. The users will have an ability to manually tag their files, but automatic tags will also be generated by the use of natural language processing techniques. These tags will allow for efficient searching of media stored on the cloud.

The server side of the system will be implemented using *Express Web Framework* [8] that's built on top of the Node.js platform. Express gives us very useful features such as routing for HTTP methods, templating, caching and security. Express web framework will be the central system that will connect together MongoDB, ElasticSearch, Web Client, Natural Language Processing Toolkit, OpenCV and other components. The Express project consists of 2 folders - public and views and 1 main file - app.js. The public folder contains stylesheets, javascripts and images, while the views folder contains template files for specific application "pages". Templates are coded in the Jade language syntax (Jade is a derivative of the HAML language). Jade templates are much more concise than HTML while also providing with special features such as an if-statement logic, server-side variables, for-loop iterations to display dynamic content. To launch the application you have to run *node app.js* from the command line (you must have node.js installed locally). Additionally, prior to launching the application you must install prerequisite Node.js packages, specified in *package.json* by running *npm install* from the command line. Express is a very minimalistic, but modular web framework, and for that reason it doesn't provide much features out of the box. In order for us to have Google OAuth2 authentication, query the

MongoDB database, upload files to Amazon S3, we have to install modules separately and manually add them into the Express application ourselves. Besides the module imports and some basic configuration settings, the *app.js* essentially consists of HTTP functions (GET, POST, PUT, DELETE) that take URL (Uniform Resource Locator) as its first parameter and a callback function that gets executed when that URL is visited from the browser.

The Amazon S3 (Simple Storage Service) will be used for storing user's files on the cloud. We will use Amazon's official SDK (Software Development Kit) for Node.js [9] to easily integrate Amazon S3 with our Express application. When a user selects and uploads a file from a mobile device, the file will be sent forward as a POST request to the */files* route, which Express will in turn execute a corresponding function that will upload a file to Amazon S3. There is no storage limit on Amazon S3, but the more storage you use, the more money you have to pay. There will be a fixed storage limit of 1 gigabyte for end-users in order to prevent abuse of the system.

The mobile web application will use the Google OAuth2 protocol for user authentication and authorization. This simplifies things a lot on our side, as we don't have to worry about potential security threats. Furthermore users prefer to use a 3rd party login system such as Google instead of creating a new username and password for every web application that they use. In order to get Google authentication to work we are using *Passport* node.js module as well as *Google Client ID* and *Google Client Secret* keys that were created via Google API Console web page. Client ID and Client Secret uniquely identify our application among million of other applications that use Google Authentication.

Storage of the tag information for the files will be maintained in the *MongoDB* [13] database. File database schema is located in the *schema.js* file which is used by the *Mongoose* node.js module. Mongoose greatly simplifies access to MongoDB via Node.js. It is a layer of abstraction, similar to ORM (Object Relational Mapper) for relational databases, that makes database query, update, create, delete very easy and intuitive for developers. File metadata that is stored in MongoDB includes *name*, *extension*, *type*, *size*, *path*, *last modified date*, *keywords*, *summary* and *user* who uploaded the file. Physical files will not be stored in MongoDB, only information about the file is stored. Actual physical files are stored on the Amazon S3. Additionally, MongoDB will also store user information to handle the case of new user vs returning user. Existing users will bypass registration process, because the information about them will already be stored in the MongoDB database.

ElasticSearch [12] is a powerful search engine which will be used for searching of files and to find similar tags of media uploaded on the cloud. It is a flexible and powerful open source, distributed real-time search and analytics engine for the cloud. ElasticSearch will allow us to search the entire documents for a particular phrase and get results in record fast times, because it is optimized exactly for this scenario. Before we could use ElasticSearch to query for information, it has to first build an index of the MongoDB database. *Mongoosastic* node.js module will help us bridge MongoDB and ElasticSearch together for easier file indexing. So, whenever a new file is created in MongoDB it will automatically be indexed by ElasticSearch as well. Another advantage of using ElasticSearch is we gain access to using WordNet database for related words. WordNet is a large, open-source, lexical

database of synonyms. ElasticSearch supports WordNet synonyms out of the box. Having synonyms support in a search engine will greatly improve search results for users.

A dedicated Python back-end server will be used along with the *Natural Language Toolkit (NLTK)* [11] to allow automatic tagging. NLTK is a leading platform for doing natural language processing. It allows you to do classification, tokenization, stemming, tagging, parsing, and semantic reasoning on a block of text. NLTK will be most useful for text documents, but not so much for photos and videos.

While NLTK provides excellent natural language processing supports for text documents, we will use OpenCV to do face detection in order to allow efficient tagging of individuals in photos. OpenCV is an open-source computer vision library that allows you to perform camera calibration, tracking, image segmentation, and other general image processing tasks. While it is impossible for us to detect whose face is in the photo, it is helpful to know that there exists a face in the photo so we could tag it appropriately.

ImageMagick and Ffmpeg will be used for image and video thumbnails to give user a preview of media file. These are both open-source tools used for image and video processing. We need ImageMagick to resize a thumbnail instead of sending a full image resolution to preview a file, ultimately in order to save some mobile bandwidth. Ffmpeg library will allow us to extract a single thumbnail from a video file that could be used to preview a file. This is similar to when you go to YouTube and see video thumbnails before actually see the video.

GenSim is an open-source topic modelling library for Python that will allow us to perform latent semantic analysis and latent dirichlet allocation. It can also analyze text-documents for semantic structure and retrieve semantic similar documents which I believe is crucial for our application. GenSim in combination with NLTK and ElasticSearch will undoubtedly provide an amazing user experience when searching for file documents.

EchoNest API will be used for music identification where an audio file can be uploaded to EchoNest via their API, then song metadata such the artist, song name, duration of the song and other details will be returned back from the API. It is the only free service as of this writing that allows music recognition. Depending on the song, size may vary from 3 megabytes to 20 megabytes, and to avoid sending music files back and forth to retrieve metadata information, Expres application will first examine the audio file locally as it usually contains at least the most basic information such as Track, Album and Artist names to save time and bandwidth.

Flask is a small Python web framework that will send results from NLTK, OpenCV, *GENSIM* [10], ImageMagick, ffmpeg over the web to our Node.js server where Express application is located, which will then finally pass it on to the mobile device. The reason why we need a second web framework is because we have the Express application written entirely in JavaScript, meanwhile NLTK, GENSIM are purely Python libraries. This means there is no way to communicate between each other without writing a separate wrapper application for each library that will convert JavaScript to Python language.

We have decided that it is far easier to set up a secondary web server that is able to communicate directly with NLTK, GENSIM, OpenCV libraries, while also have an ability communicate via a REST API with the Express application.

STATEMENT OF WORK

Users: Administrator, Unregistered, Registered

As an administrator I should be able to manage the storage quota on the cloud. (Estimated time: 15hrs)

1. Set a quota database field for a user and set it to 1GB by default.
2. On each file upload, check whether used has exceeded his/her quota.
3. Create an error dialog for the event when user has exceeded his/her quota.

As an administrator I should be able to delete users. (Estimated time: 25hrs)

1. Create a small admin dashboard.
2. Display a list of all registered users with a delete button next to the user's name.
3. Implement a function that will delete user from the MongoDB database and clean up all files on Amazon S3 that belong to that particular user.

As an unregistered user I should be able to register for an account. (Estimated time: 10hrs)

1. Use passport library for node.js to handle user authentication.
2. Configure passport to work with Google OAuth2.
3. Store user information in MongoDB such as Google ID, access token, first and last name, email etc.

As a registered user I should be able to login to my account. (Estimated time: 10hrs)

1. Create HTML page with "Sign in with Google" button.
2. Write a server side function to handle login route.

As a registered user I should be able to view media files uploaded on cloud. (Estimated time: 20hrs)

1. Create homepage layout using Ratchet UI framework.
2. Write a server side function to retrieve files for the current logged in user from MongoDB.

As a registered user I should be able to upload photos to the cloud. (Estimated time: 20hrs)

1. Create "Upload" button using Ratchet UI framework.
2. Write a server side function to allow uploaded files to be stored.

As a registered user I should be able to tag the photos I upload. (Estimated time: 120hrs)

1. Create a comma separated tags upload field.
2. Parse and split comma separated tags on server side and save them to MongoDB for the particular file.

As a registered user I should be able to upload documents to the cloud (Android users only). (Estimated time: 25hrs)

1. Detect whether uploaded file is a document and what type of document it is.
2. Extract plain text information from the document.

3. Store extracted text inside the MongoDB database along with other file metadata.

As a registered user should I be able to tag the documents I upload (Android users only). (Estimated time: 25hrs)

1. Create a textfield form during the upload process to store comma-separated tags.
2. Parse and split the comma-separated tags on server side and save them along the file metadata in MongoDB.

As a registered user I should be able to upload videos to the cloud. (Estimated time: 30hrs)

1. Detect whether uploaded file is a video file.
2. In addition to regular file metadata, store additional video information: aspect ratio, format, video resolution, audio codec, etc.

As a registered user I should be able to tag the videos I upload. (Estimated time: 10hrs)

1. Create a textfield form during the upload process to store comma-separated tags.
2. Parse and split the comma-separated tags on server side and save them along the file metadata in MongoDB.

As a registered user I should be able to upload music files to the cloud (Android users only). (Estimated time: 60hrs)

1. Detect whether uploaded file is music filetype (mp3, aac, mp4).
2. Extract and store music metadata on the server side, if no metadata is present use EchoNest API to do music recognition.
3. Use a 3rd party API to request lyrics (if available) for that particular music file, and store it in MongoDB along with other file metadata.

As a registered user I should be able to tag the music files I upload (Android users only). (Estimated time: 15hrs)

1. Create a textfield form during the upload process to store comma-separated tags.
2. Parse and split the comma-separated tags on server side and save them along the file metadata in MongoDB.

As a registered user I should be able to search for my media files based on keywords. (Estimated time: 50hrs)

1. Search bar created for user to enter keywords.
2. Queries sent to ElasticSearch with key words taken as input by user.

As a user I should be able to retrieve semantic results from searching. (Estimated time: 175hrs)

1. Create an HTML page showing search results based on user's searching criteria.
2. Semantic results given by the use of MongoDB, NLTK (Natural Language Processing Toolkit for Python) and ElasticSearch database in conjunction with WordNet lexical database for word synonyms support.
3. Searching will be performed not only through the use of "full-text search" support provided by ElasticSearch but also utilizing automatically generated tags and manually created tags by the

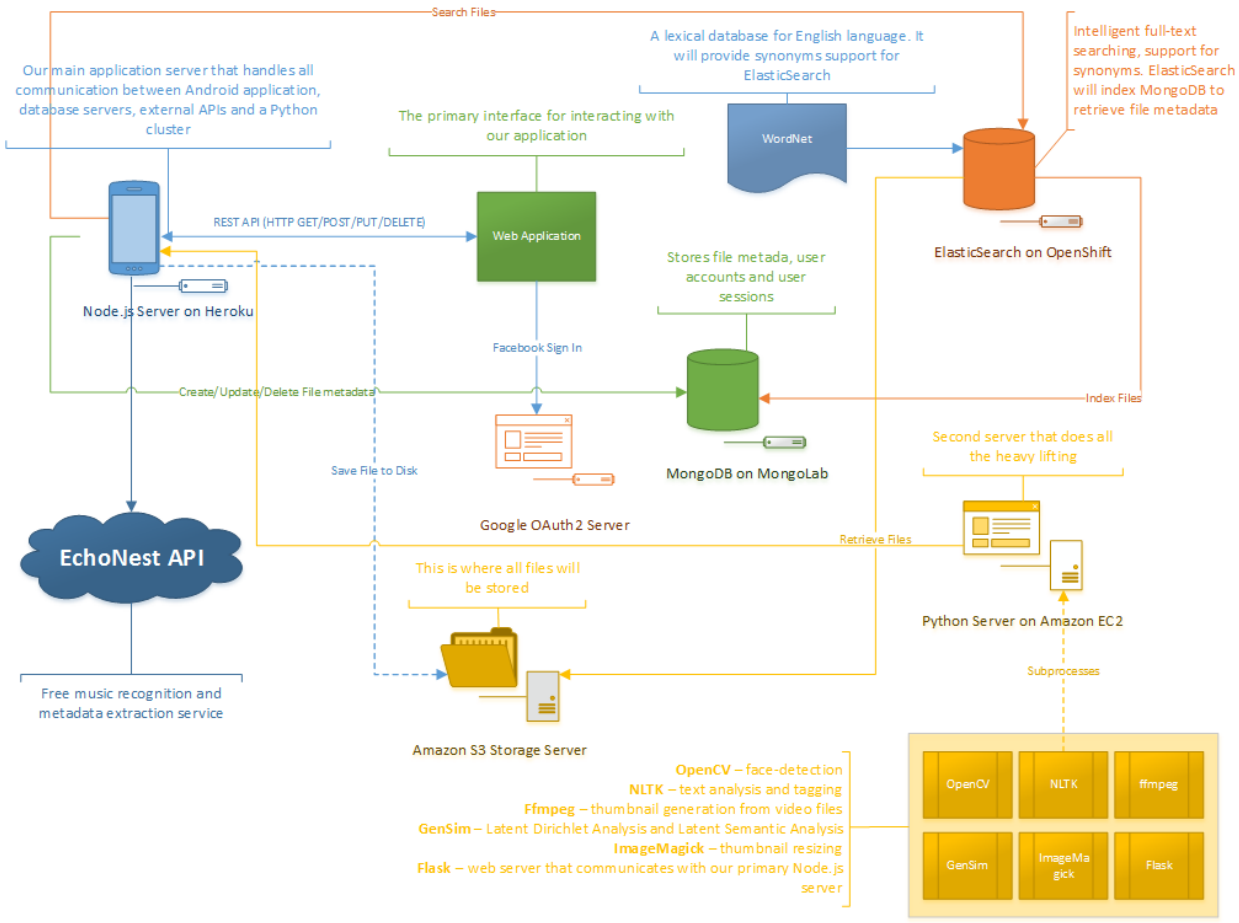
user in order to improve search accuracy.

As a user I should be able to download files stored on the cloud. (Estimated time: 70hrs)

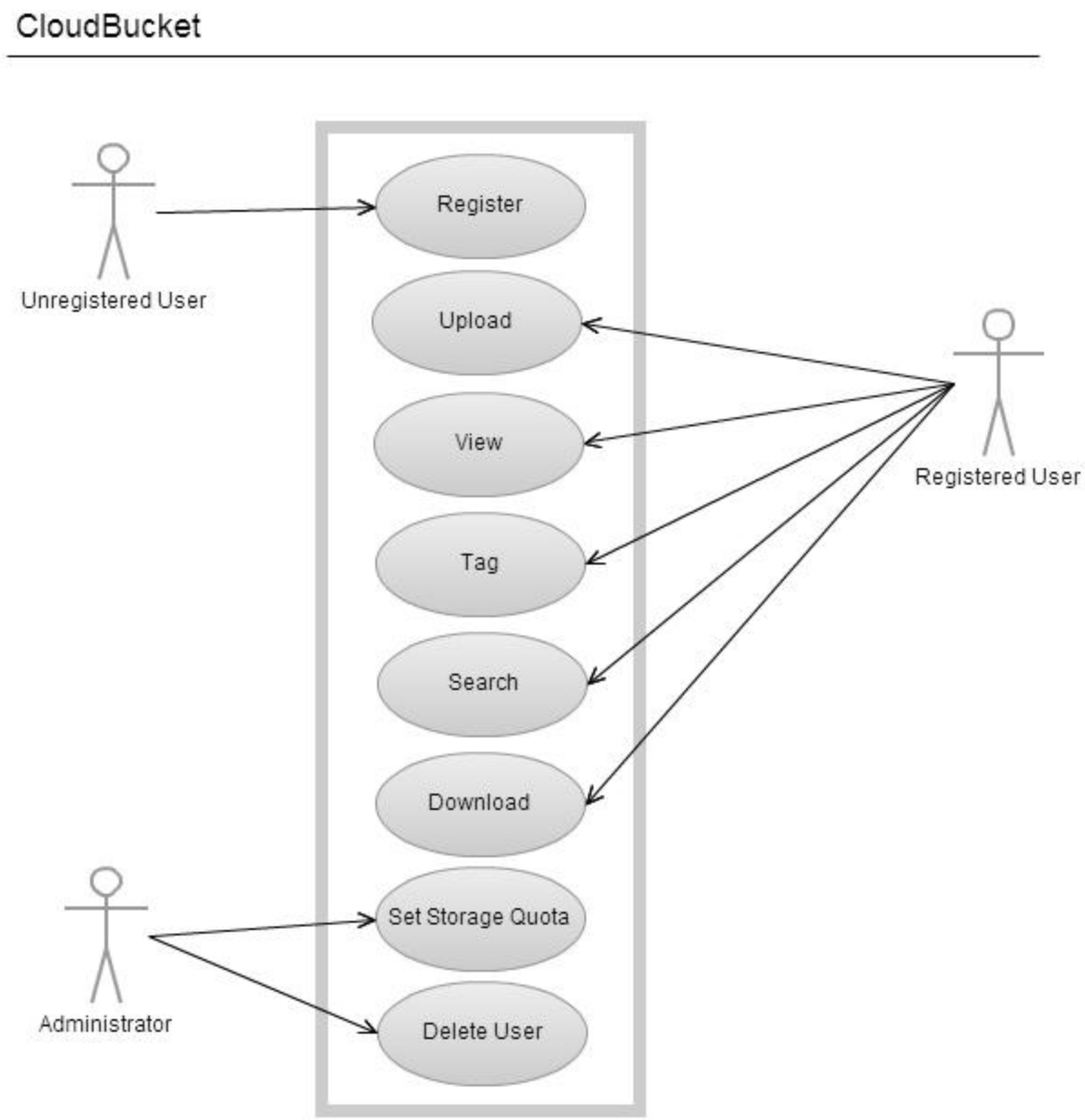
1. Create a dropdown menu for each file in HTML to allow user download a file.
2. Link each file to a file location path located on Amazon S3.
3. Implement a security policy so that other users cannot download files that do not belong to them.

DIAGRAMS

General Diagram: Shows the overall flow of the system and short descriptions for each sub-system.



Use Case Diagram: Showing main functionality of CloudBucket system.



FACILITIES

The facilities that will be used for this project will be:

- Laptop
- Google Nexus 4
- iPhone 5
- Ubuntu 13.04
- Android Developer Tools r21
- JetBrains IntelliJ IDEA 12

SUPPORTING INFORMATION

Working as a team of two, attending the City College of New York enrolled in *Senior Project 1* (CSC 59866), both students are majoring in Computer Science.

SUMMARY

In this project we expect to implement a user-friendly mobile-effective media search system. This system will use cloud computing to store files which will include videos, audio, text and pictures. These files will be uploaded to the cloud and each user will have the ability to view their files, tag their files, search their files and download their files.

Files one user uploads to the cloud will not be visible to other users. Semantic search to allow efficient searching of the media files stored on the cloud will be incorporated. This will allow users to search for data based on its content. The use of automatic and manual tagging will give rapid search results. The approach to carry out this system has merit above all others as it is using current tools in technology to create an efficient system.

BUDGET

Time budget of 4 hours a day, Start date: June 1st, 2013.

REFERENCES:

[1] Tim Berners-Lee, James Hendler, Ora Lassila. (2001) *The Semantic Web.* Scientific American.

[2] Nigel Shadbolt, Wendy Hall, Tim Berners-Lee. (2006) *The Semantic Web Revisited*. IEEE Computer Society.

[3] Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, Susie Stephens. (2009) *The Semantic Web in Action*. Scientific American.

[4] Larry Hardesty. (2010) *Towards the Semantic Web*. MIT News Office.

[5] Pete, Pachal. (2012) *Bing: Re-organizing the Web to Get Stuff Done*. Mashable.

[6] Claire Cain Miller. (2013) *Google Introduces New Search Tools to Try to Read Our Minds*. The New York Times.

[7] Nagaraja, Radha. (June 2009) *Semantic Media Search with Mobile Clients Using Cloud Computing.* Master of Science (Computer Science) thesis. City College of the City University of New York.

[8] *Express - node.js web application framework*. Retrieved from http://express.js.com.

[9] *AWS SDK for Node.js*. Retrieved from http://aws.amazon.com/sdkfornodejs.

[10] *GenSim: topic modeling for humans*. Retrieved from http://radimrehurek.com/gensim.

[11] *Natural Language Toolkit.* Retrieved from http://nltk.org.

[12] *ElasticSearch real-time search for the cloud*. Retrieved from http://www.elasticsearch.org.

[13] *MongoDB*. Retrieved from http://www.mongodb.org.

PERSONNEL

Sahat Yalkabov, Computer Science (BS). Graduation date: May 2014.

- TechCrunch Volunter: Helped out TechCrunch Dirsrupt NYC 2013 with general tasks, including help desk, reception, registration, taking care of vendor companies and the conference tear-down/packing.
- Research Internship: C-SURP (CUNY Summer Undergraduate Research Program) is a summer research opportunity for undergraduate students from many diverse fields. During my time in this summer program, I worked closely with Professor Ioannis Stamos and graduate students on exploring various image segmentation techniques necessary for image classification and photorealistic modeling. Econofy Hackathon Project Contributor: Econofy debuted at CleanWeb Hackathon in January 2012 where it was chosen best by both jury and audience. The idea behind Econofy is to help consumers find more energy efficient appliances and calculates ROI on buying them instead of keeping old ones.
- CMACS 2012 Research internship: Computational modeling and analysis of complex systems involves studying and performing simulations of cellular signaling pathways, in my case the T-cell receptor.
- Final project for the CSC32200 Software Engineering class at City College of New York. It has been coded entirely in Node.js in accordance to the specification document provided to us. Working on this project has strengthened my skills in JavaScript, jQuery, Node.js, MongoDB, Session Management, Web Security.

Emily Bodden, Computer Science(BS), Graduation date: Feb 2014.

- Deans List 2012 - 2013, The City College of New York
- Cayman Islands Government Scholarship, Study at the City College of New York, 2010 - 2014.
- Associates of Science, Natural Science, University College of the Cayman Islands, 2009.