

COM3014 Advanced Challenges in Web Technologies Coursework

Group 4 – 2014

Michael Hough – 6130813
Jade Beesley – 6125834

Table of Contents

Introduction.....	3
Required functionality	3
Solution Description.....	3
Stakeholders.....	3
System requirements	4
Roles and Contributions	4
Michael Hough	4
Jade Beesley	4
System Architecture.....	4
Application Layers.....	4
<i>Presentation Layer</i>	4
<i>Business Logic Layer</i>	4
<i>Persistence Layer</i>	5
Use Case Diagram	5
Behavioural View Diagrams	7
Third Party Libraries.....	10
Instruction Manual	10
Deploying the project.....	10
Welcome.....	11
Signing in.....	11
Signing out.....	13
Uploading an image	13
Browsing images.....	14
Using the RESTful API to fetch images.....	15

Introduction

This report documents group 4's submission for the coursework assignment as part of COM3014 Advanced Challenges in Web Technologies in the second semester of the 2013-14 academic year.

The purpose of this assignment is to demonstrate understanding of the topics covered in class by developing a web application using Spring MVC in groups.

Required functionality

According to the assignment sheet, marks are assigned for implementing the following functionality:

- Proper segregation of the core components of the application, for example user interface, application logic, and data storage.
- Usage of topics covered in class. This system implements:
 - Rich Client Experience through use of AJAX.
 - Usage of the Spring MVC framework.
 - Security.
 - RESTful services.
- Coding quality – code must be properly structured and documented. All HTML should comply with W3C standards.
- Impressive features. This system implements:
 - Enhanced security and integration with third party services via Google+ OAuth authentication.
 - Fully AJAX enabled user experience.
 - Uploading and displaying of images.

Solution Description

The produced application is an image sharing website. Users can upload images to the website, and then they are displayed on the website for other visitors to see. In order to upload a file, a user must first sign in to the website using their Google account, and then the website accesses information from the user's Google+ profile.

Stakeholders

The stakeholders for the project are:

- Michael Hough
- Jade Beesley

System requirements

The solution must:

- Allow a user to upload and view an image.
- Only allow a user to upload an image after the user has authenticated.
- Display multiple images at once, and enlarge images to allow a user to see the image more closely.
- Automatically load more images when the user scrolls down.
- Sort images by newest first.

Roles and Contributions

Michael Hough

Michael developed much of the server side logic. This involved creating the Spring MVC backend, and implementing controllers for server side application logic, beans, and the RESTful services. Michael was also responsible for developing the server side logic for the OAuth integration, producing the logic for the file upload feature, and providing properly formatted output for AJAX calls.

Jade Beesley

Jade was responsible for front-end design and processing. This includes developing the view, and developing the necessary CSS styling and JavaScript and JQuery scripting for the promised rich client experience, comprising of AJAX calls, and scripting for the client side implementation of the OAuth integration. Jade also implemented the user name lookup feature.

System Architecture

Application Layers

The application layers are as follows:

Presentation Layer

The Presentation layer contains all the logic for rendering a view, and encapsulates the JSPs and the front-end information.

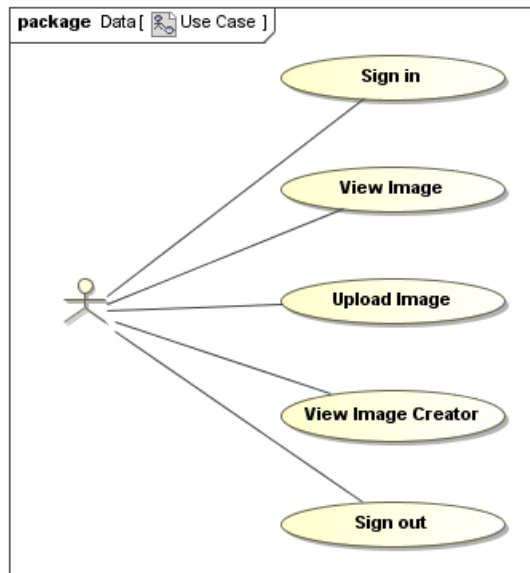
Business Logic Layer

The Business Logic layer contains all the logic for handling server side responses to client side events. The client sends and requests information through AJAX calls, which are handled as page requests by the server, and the required response is returned as a special view.

Persistence Layer

This layer serves to store the images that have been uploaded to the server. When an image is uploaded, it is renamed to contain the timestamp of when it was uploaded and the uploader's Google+ profile ID.

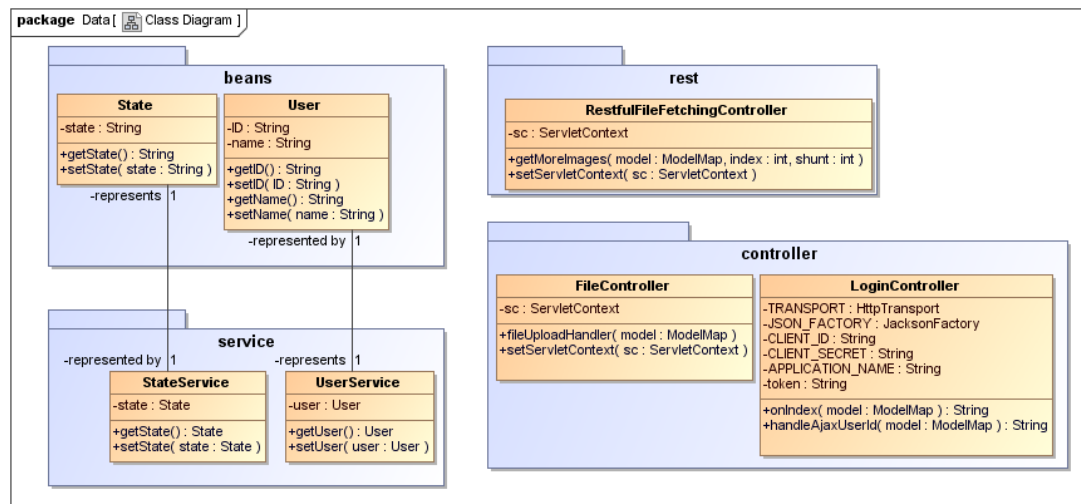
Use Case Diagram



For this system, there is only one actor, since there is no requirement for moderation or administration within the system itself. A user must be able to:

- Sign in – authenticate with the system by some reliable means
- View Image – look at images that other users have created.
- Upload Image – add their own images to the system. There must be a limit of 5MB per image.
- View Image Creator – see the name of the person who uploaded a certain image
- Sign out – remove the user's authentication from the system, requiring that they sign in again. Images uploaded in that session must **not** be removed.

Class Diagram



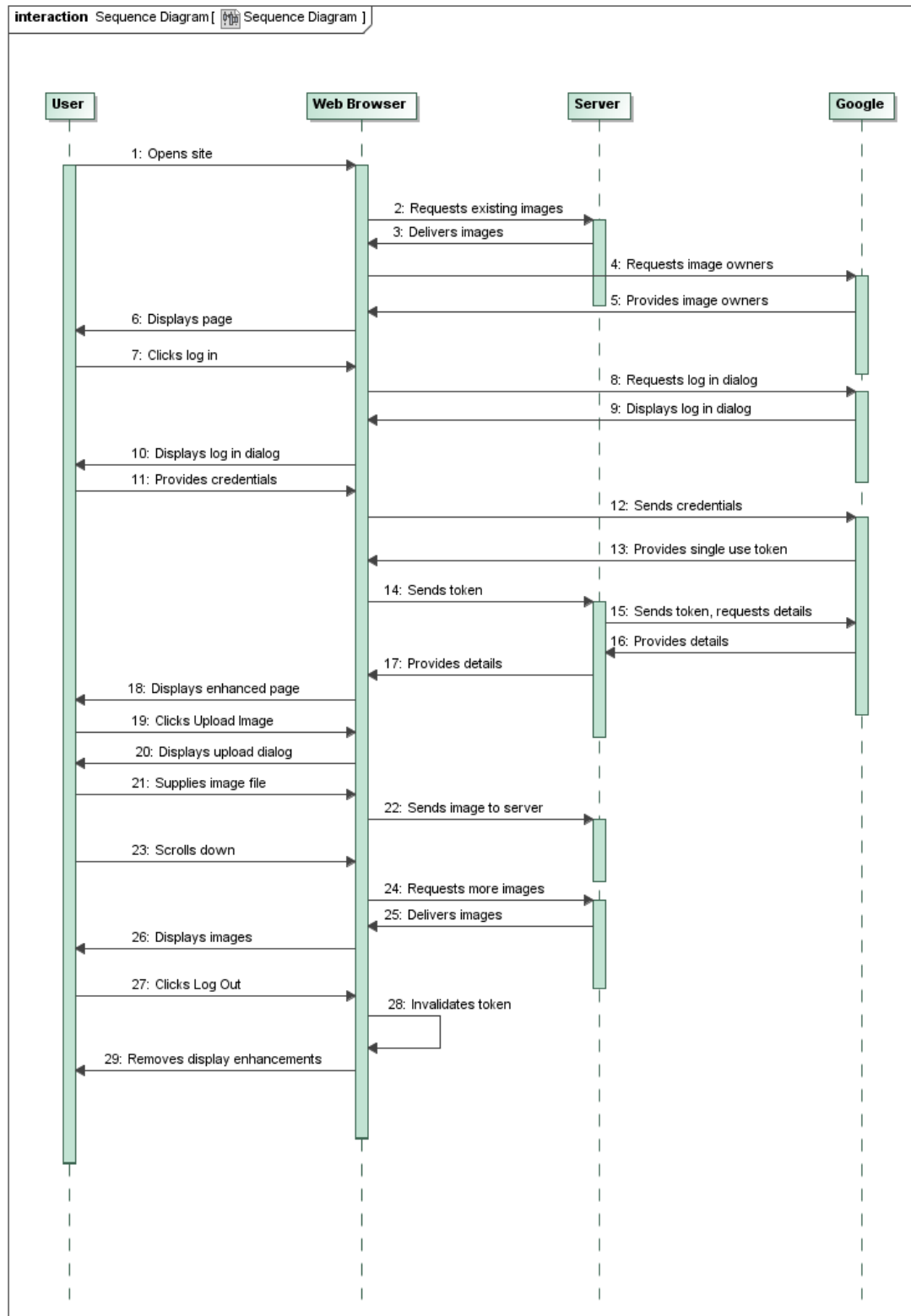
The system is separated into the frontend system and the backend.

The frontend as seen by the user is a single JSP view, within which all user interaction is handled by JavaScript and AJAX requests. There are other views created for returning information to AJAX requests, but these are not intended to be accessed by the user, and merely return information in JSON format.

The backend is built upon the Spring MVC framework, and includes controllers that capture and handle page requests from users and AJAX requests from the system. There are models for storing user data. Images are stored on the file system in a folder in the “*uploads*” folder in the built “*web*” directory.

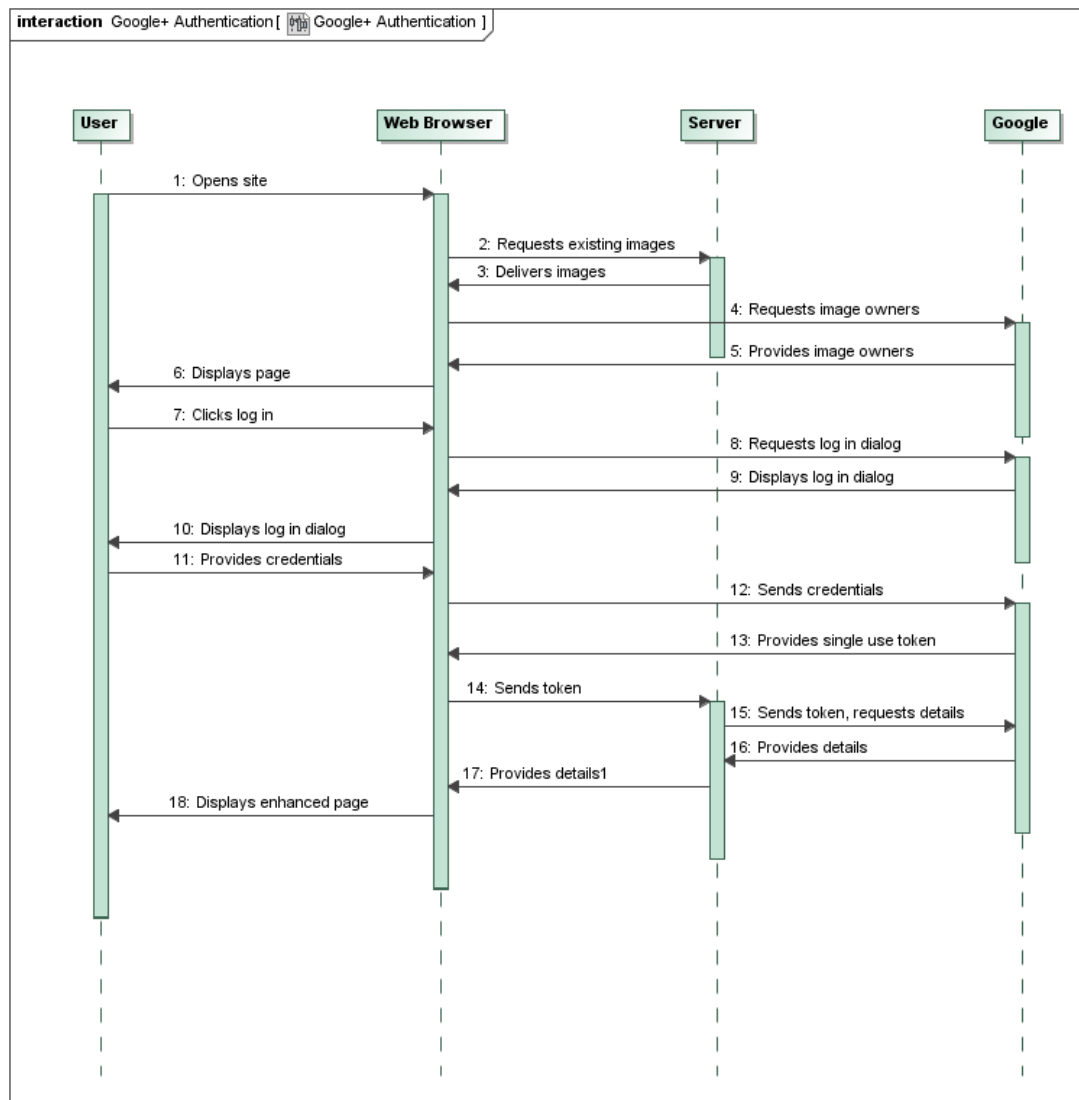
Sequence Diagrams

Overall System



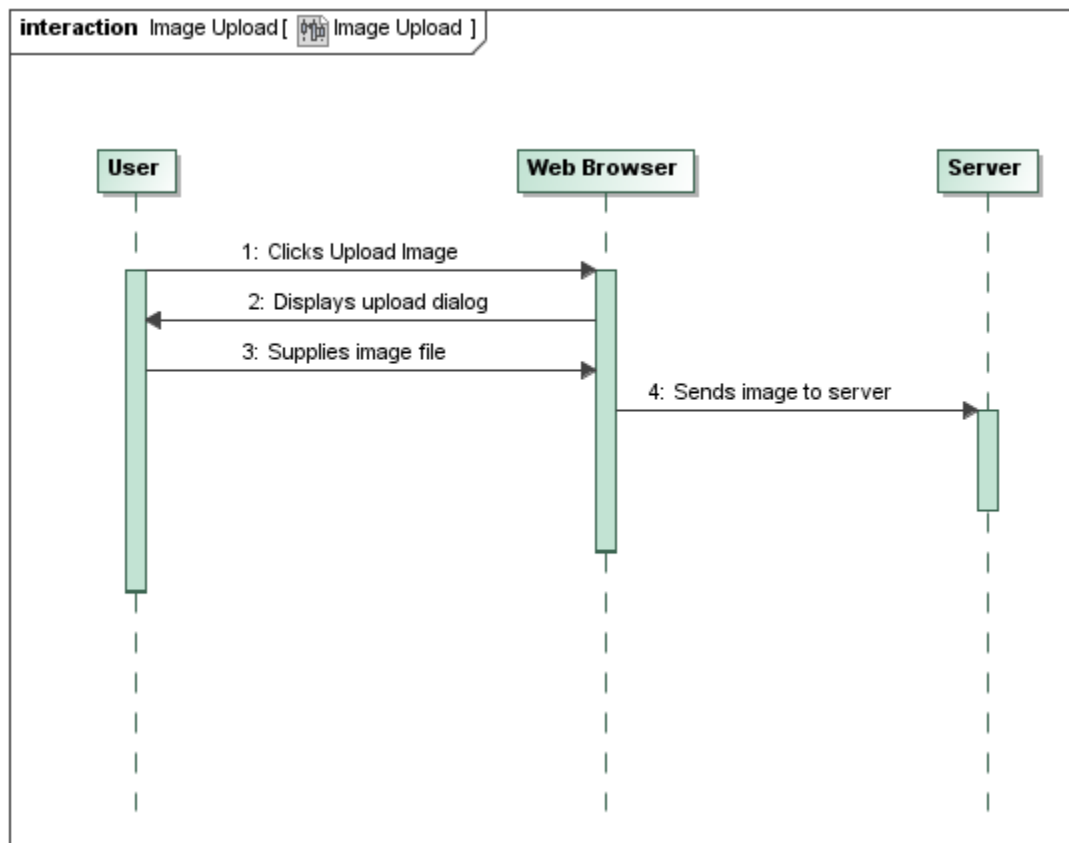
This sequence diagram shows how a user might interact with the entire system. The component parts of this sequence are explained in further detail below.

Google+ Authentication



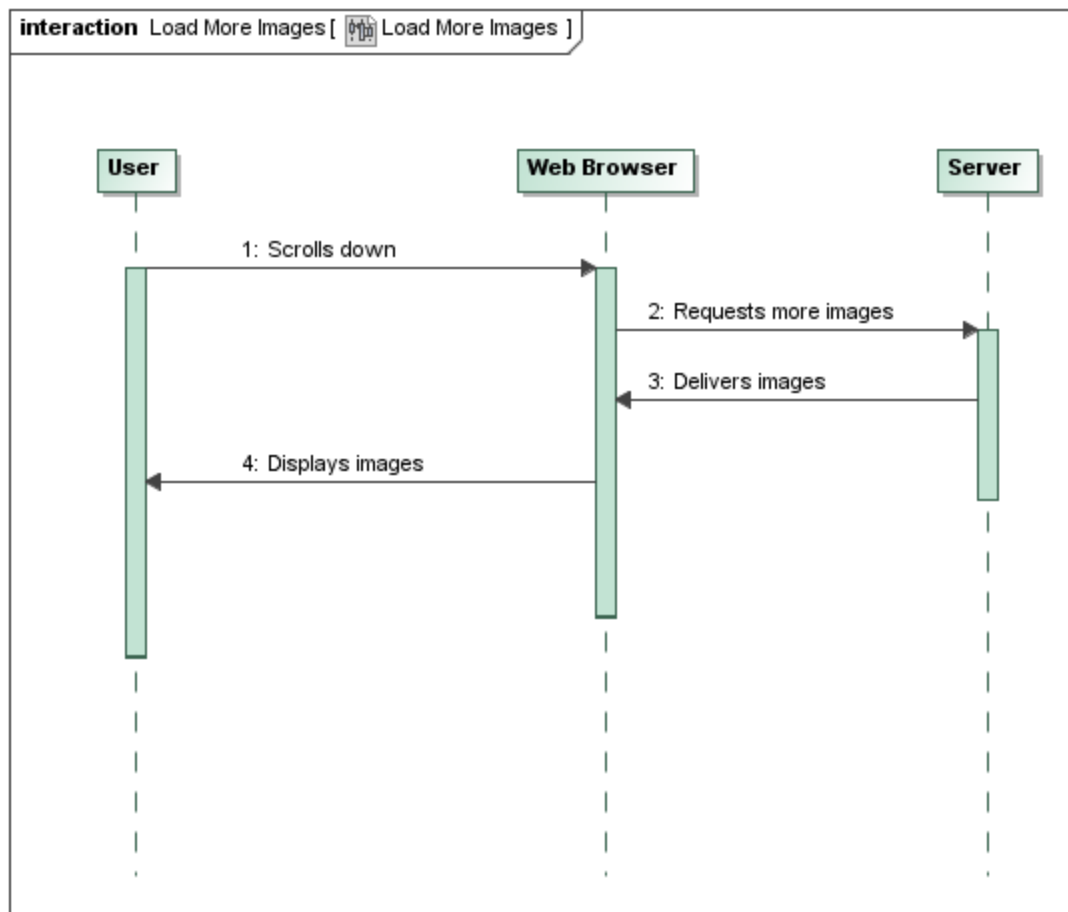
Google+ authentication requires interaction between both the user's system and Google, and the server and Google. When using Google's authentication system, their library produces the sign in box for the user and accepts their input. When the user has signed in correctly, Google's server sends a single use token to the web browser. This single use token is then sent to the application server, which sends the token to Google along with a request for access to the user's information. Once this is provided, the page is updated for the user showing their name and Google+ profile picture.

Image Upload



Compared to Google+ authentication, uploading an image is simple. When the user clicks the Upload Image button, a dialog appears requesting that the user provide an image to upload. When they do this, the image is sent to the server as POST data. The image is then stored on the server. After receiving a success message from the server, the client displays the image without having to download it from the server.

Loading more images



Loading images is performed as a GET request from the web browser. The client must include an identifier stating the last image they received, and the server returns the next six images. If there are less than six images left to display, the server returns only the images that exist.

Third Party Libraries

The following third party libraries were used for this project:

- Spring MVC
- JQuery
- Google+ Authentication API
- Google+ API
- Apache Commons FileUpload
- Apache Commons IO
- DropzoneJS

Instruction Manual

Deploying the project

Before you can run the project, you'll need to make sure you have all the third party libraries listed above added to your project. All the necessary jar files

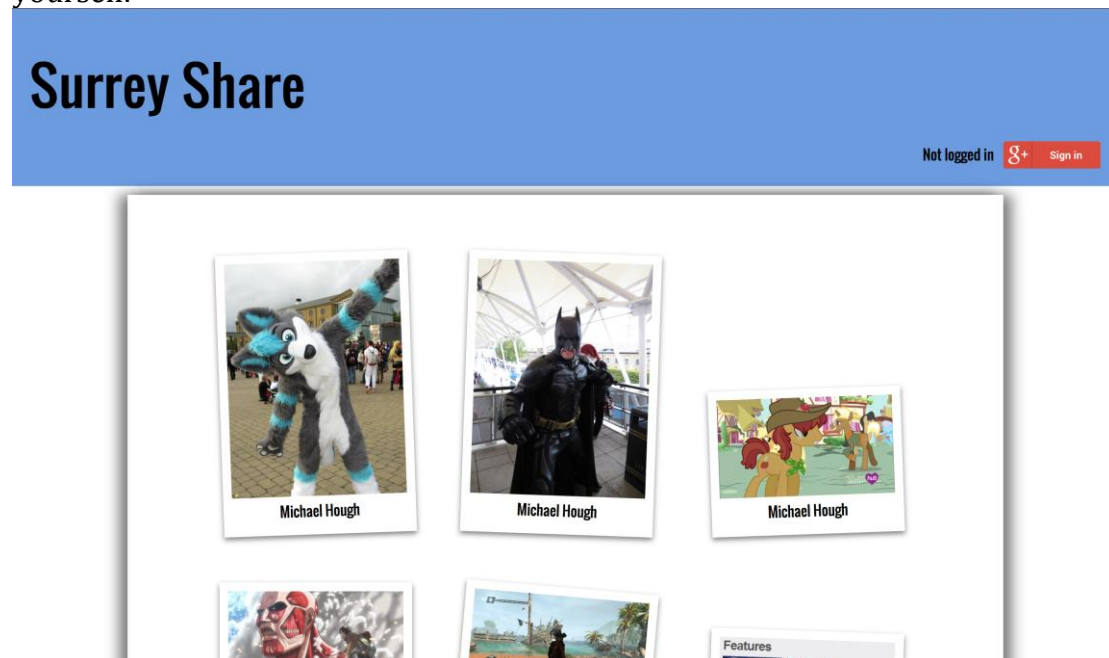
Michael Hough – 6130813
Jade Beesley - 6125834

should be included in WEB-INF/lib, but you might find that the project doesn't see them automatically. Create a library with all of the jars in the libs/jars directory (you don't need to, but you can include the source or javascript jars from their respective folders if you want to), and the project should build.

The project is designed to run with Apache Tomcat on localhost, port 8084 or 8080. If you use any other address or port, the Google+ integration features won't work, because the application's API key is only valid for running from those specific addresses.

Welcome

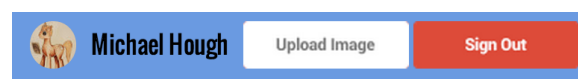
Welcome to Surrey Share. With this service, you can upload any photo and share it with other Surrey Share users. When you open the home page, you'll see a selection of the most recently uploaded images, captioned by the name of the person who uploaded each image. To upload an image, you'll have to sign in yourself.



Signing in

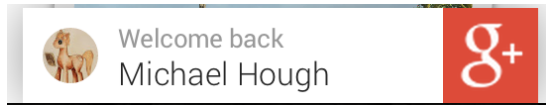
You can sign in to Surrey Share using your Google account. Before you can use Surrey Share, you must ensure that a Google+ profile exists for your Google account, since Surrey Share will access information in your Google+ profile.

If your Google account details are saved in your web browser, you'll probably be signed in automatically. If you see your name in the top right corner of the screen, you're already signed in, and you can skip this step.

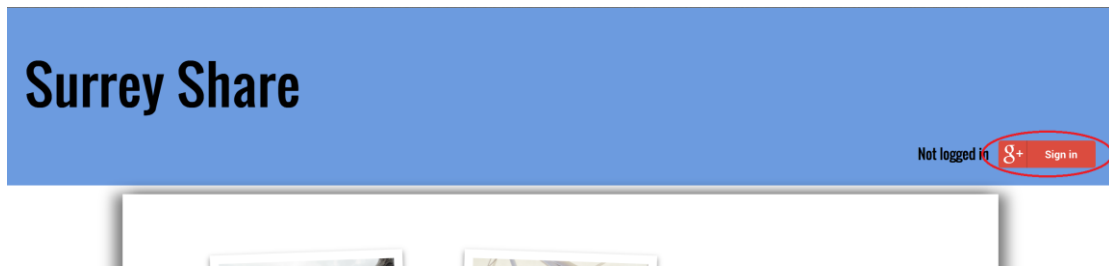


Michael Hough – 6130813
Jade Beesley - 6125834

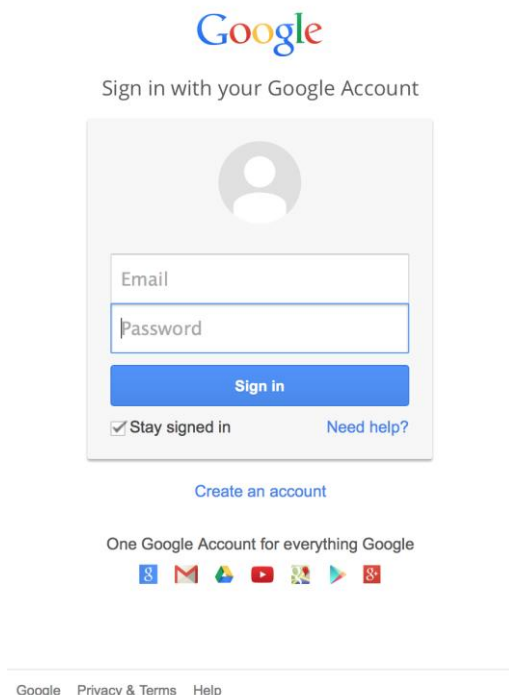
On Safari and Google Chrome, returning visitors are greeted by a “welcome back” message.



To sign in, click **Sign in** in the top right corner of the web page. The Google Accounts dialog box opens.

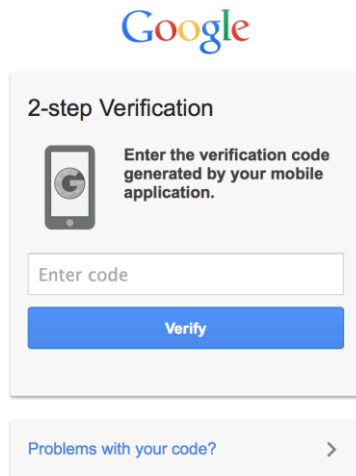


Type your Google account information in the boxes. Don't worry; Surrey Share doesn't get to see your Google account password. When you're done, click **Sign in**.



If you have two-step authentication enabled on your Google account, and you haven't signed into your account from this computer recently, you'll be asked to enter the code generated by your mobile application. Type it and click **Next**.

Michael Hough – 6130813
Jade Beesley - 6125834



Google

2-step Verification

Enter the verification code generated by your mobile application.

Enter code

Verify

Problems with your code? >

[Google](#) [Privacy & Terms](#) [Help](#)

When you're done, the Google authentication box will close, and you'll be back on the Surrey Share page. Now, though, you should see your name and Google+ profile picture in the top-right.



Signing out

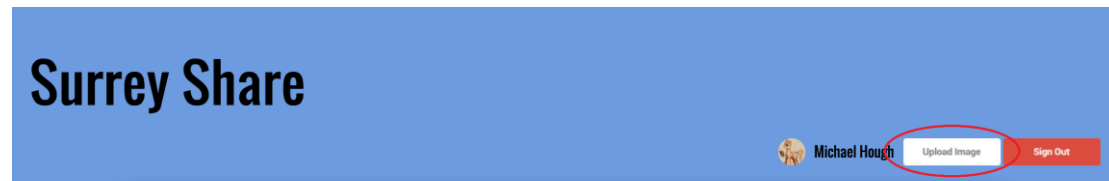
Signing out of Surrey Share is simple. Just click the Sign Out button.



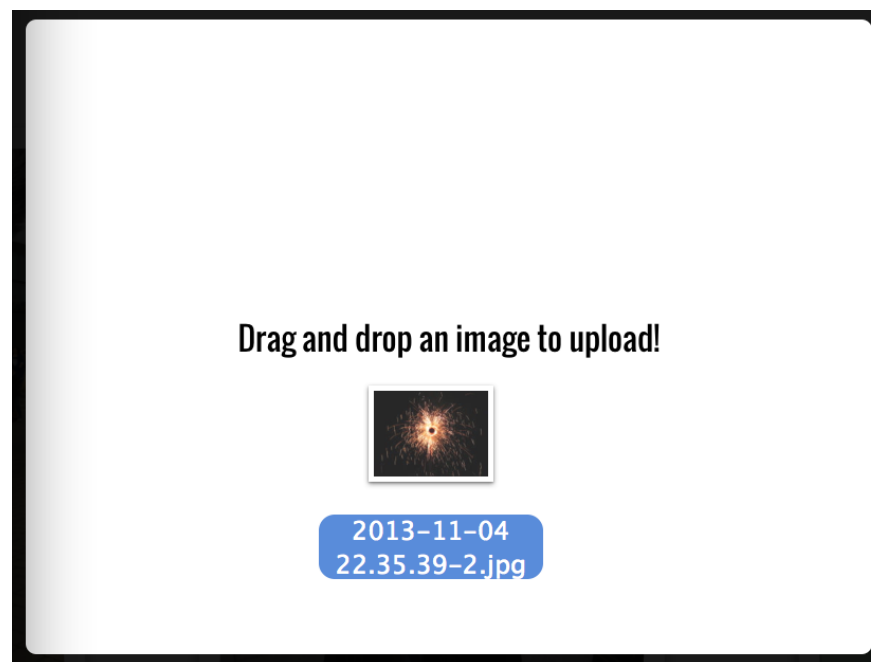
Note for Google Chrome users: Chrome *really* likes to sign you in to Google pages automatically, so when you sign out, it'll sign you back in again. This is part of Chrome, and we can't really do anything about it. If you really need to, try using a different browser, like Firefox.

Uploading an image

To upload an image, click the Upload Image button in the top-right corner. The file upload panel fades in.



Open the file browser on your computer, and find the image you want to upload on your system. When you've found it, click and drag it onto the white box. When you let go of the mouse button, your file is uploaded and appears first in the list of pictures.

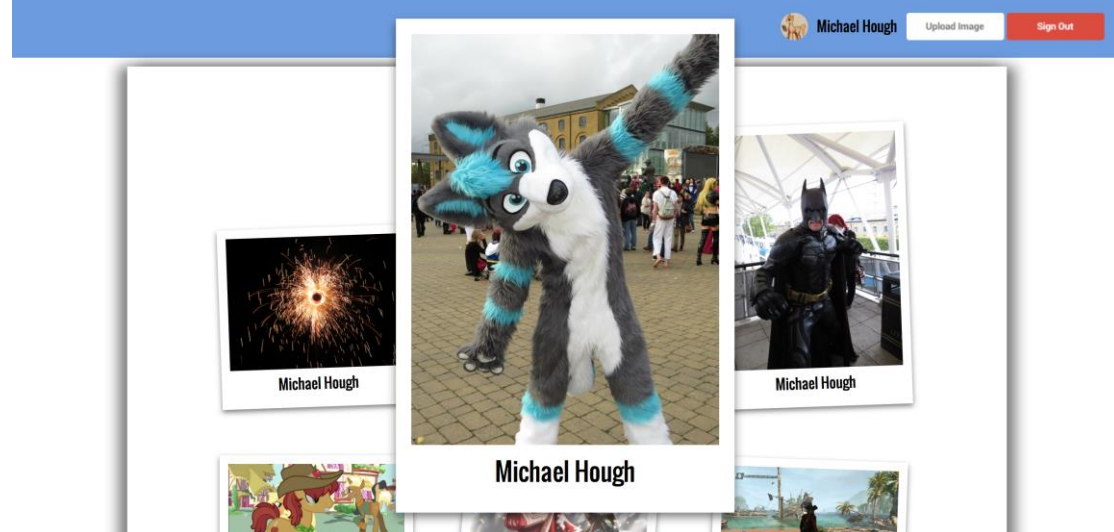


Browsing images

When you first open Surrey Share, you'll see the six most recently uploaded pictures. If you scroll down the page, it'll automatically load more, six at a time, until there's no more to show you.

If you want to take a closer look at a photo, just hold your mouse cursor over it. It will straighten out and get larger, so you can see more detail.

Surrey Share



You can click and drag images to move them around the screen. When you let go of them, though, they'll zip back to where they were before.

Using the RESTful API to fetch images

The REST API provides functionality to load images, six at a time, using a call to <http://localhost:8084/COM3014-source/rest/getMoreImages/{index}/{shunt}>. Information is returned as JSON. You need to provide the following parameters as path variables, filling in the spaces above:

- `{index}`: The index of which six images you want to fetch – if you specify zero for this, the API will return the most recent six images to have been uploaded.
- `{shunt}`: Shifts the image search by an amount. For example, if you want to fetch images starting from the second newest, you can specify 1 here (to skip the first image). In this case, the API will return the second newest through to the seventh newest images.

For example, to fetch six images starting from the eighth, you should specify an index of 1, and a shunt of 2, like so:

<http://localhost:8084/COM3014-source/rest/getMoreImages/1/2>.

The output JSON has the following format:

```
{ "images": [
  {
    "filename": "${filename}",
    "userID": "${userID}"
  }
  // And five more filename, userID pairs for the other five images.
]}
```

Michael Hough – 6130813
Jade Beesley - 6125834

The *filename* value stores the file name of the image. Note that images are stored within the `http://localhost:8084/COM3014-source/uploads` directory. The *userID* value stores the Google+ profile ID of the person who uploaded that image.

If there aren't enough images to fulfil your request, any JSON fields that can't be filled will be left empty.