Mitul Patel – PhotoBox Project

Application (Prod): http://www-test.cs.umanitoba.ca/~patelm5/index.html

This project is a proof of concept of how distributed system can be used for persistence which is separate from the clients machine. This is PhotoBox, cloud storage for users to store their most memorable photos and can be accessed from anywhere using any device (What a Distributed World!). Uses have a quota limit of 10 photos. Users can register, login. view photos, upload photos, delete photos and share photos via a sharable link or email.

Please note apart from using Bootstrap and some minor JS code snippets from w3schools, all the code is written from scratch by me.

Back End Architecture

Please pardon me for using .txt files instead of an actual SQL Database, I understand that using text files like these halt backend performance.

"textDb" Folder

This folder can be thought like an SQL Database and the files inside can be thought of as Tables.

Everything in the files is stored as JSON.

- ***userData.txt**:* Stores all the usernames, passwords and names of the user's photos
- ***sessions.txt**:* Stores the different session UUIDs for each user (users can have more than 1 session). **Also stores a session's state (whether a given user session has up to date photos of that user).**
- *locks.txt:* Stores a username and user's session UUID that is currently uploading/deleting an image to avoid the same user uploading/deleting at the same time from a different session (Mimics locking).

"cloudDb" Folder

This folder contains folders where each folder belongs to a user and the folder name is the username (usernames are unique). Each user folder contains the photos uploaded by their respective user (photo names are unique). To search for a photo we need the username and folder name. The structure goes like this:

/~patelm5/cgi-bin/cloudDb/USERNAME/PHOTONAME

identifier.cgi

This CGI script deals with authentication. It presents the user with a login/registration screen if the user doesn't have a session based on two cookies. The cookies are "guestid" which is the username and "sessionid" which is a unique UUID for the user's session. If the correct cookies are present, the user will see the PhotoBox Dashboard. **Note that if the user fraudulently makes up the "guestid" and "sessionid" cookies, they won't be able to access the PhotoBox Dashboard because those cookies don't exist on the server's session Database. Also, if the user deletes any of those cookies from their browser, the script will log them out immediately.**

serve.cgi

This CGI script serves all the XHR Requests which transfer everything as JSON. Methods include:

- ***GET (User's Photos):*** Returns the User's photos only if:
    1. The User has logged in, Registered or Opened a New Tab
    2. Another session of the same user has uploaded/deleted a photo (States Tracked in sessions.txt)

  Otherwise, it returns a message like "Cloud Data Unchanged"

- ***POST (Verification Request):*** Returns "OK" if the user has valid "guestid" and "sessionid" cookies. Otherwise returns something like "Must Log In".
- ***POST (Upload Image):*** Receives a base64 encoded image and makes a file in the user's respective folder in cloudDb. It will send something like "OK" if everything went fine but will return "User Quota full" or "image with the same name already exists".
- ***POST (Delete Image):*** Deletes the image from the user's respective folder in cloudDb. It will send something like "OK" if everything went fine but will return "image not in the folder" if the image isn't there.

NOTE: The POST methods for Upload and Delete both uses a locking mechanism to avoid a user concurrently uploading/deleting an image at the same time with two different sessions.

shared.cgi

In PhotoBox, a user can share a photo with a friend by generating a sharable link. This CGI script solely serves the browser's GET requests that are sent when someone wants to open the sharable link. Each sharable link has a query with a unique UUID which is uniquely matched to a photo on the PhotoBox cloud. Example:

http://www-test.cs.umanitoba.ca/~patelm5/cgi-bin/**shared.cgi?photoId=78a76fb7-5d33-4158-99db-ee569b5bbab1**

email.cgi

This CGI script solely serves XHR POST requests sent from the JS front end when the user wants to share a picture with a friend via email. It takes the username, photoname and recipient email address. **It then sends an email to the recipient's email address with the photo in the attachments and a small message which includes a shareable link to the photo in case the user doesn't want to download the attachment.** It uses the SMTP server that is set up on port 25 of the Linux Machines (Aviary). Its also using MIMEMultipart, MIMEImage and MIMEText from python's email library.

forget.cgi

This CGI script runs when the user clicks Logout. It clears the "guestid" and "sessionid" cookies from the servers cookie store (sessions.txt) and the browser. It also redirects the user to the login page.

Thank you for reading