Project Proposal: Photo Recognition

Team Crimson (name | email | GitHub username)

- Adriana Pedroza | apedroza@utexas.edu | adripedroza
- Abhishek Mandal | abhi.mand@utexas.edu | abhimand
- Shashwat Pandey | shashwat.pandey@utexas.edu | shash-pandey27
- Matthew Machado | matthewmachado@utexas.edu | mmachado33
- Pranav Rama | pranavrama9999@utexas.edu | pranav12321
- Umer Salman | umer936@utexas.edu | umer936
- Brandon Pham | brandon.pham@utexas.edu | bpham1

GitHub Team URL: https://github.com/PhotoStorageSoftwareLab
Google Drive: Team Drive URL cannot be shared without adding a member to the folder. The Proposal itself is here:
https://drive.google.com/open?id=1alzzBXZNZL3PwdTbbmK1kxgPYL2RjuJxOSldBpvAOwQ

Vision

Cloud storage has reached tipping points, specifically in security/privacy and processing/storage equipment. There are ways to keep your data safe from cloud storage locations and companies, but those solutions do not provide the level of features that the cloud storage solution offers. One specific pain point in removing one's information from cloud storage is the photo gallery apps and tools, which provide excellent features like image feature tagging and classification, sharing across users, and location tagging of each photo. All of these tags are then searchable, which means the user can just search their gallery for "Colorado" or "bicycles" and get their images associated. These powerful features are only available if you commit to using the cloud storage options provided by major companies like Google or Apple. That becomes a limiter dependant on how much you care about your privacy or how much you are willing to pay. The tipping points for cloud storage set up a nice gap for a product that is secure/private and doesn't cost as much.

For one, when users store data on cloud photo galleries, they give up huge amounts of private information. The saying, "A picture holds a thousand words," is even more true with digitized photos, which contain metadata that include detailed camera information, GPS coordinates, and more. Storage providers can then extract more info from photos and associated data, like who one is with in the photo and what objects the photos contain. Who the person was with when the photo was taken does not even require the people to be in frame. If two users upload images from the same location at the same time or using the same WiFi network or cell towers, odds are the users are together. Companies can use all this info to make connections between people or figure out likes, dislikes, etc to package and sell. This presents privacy and security issues. Users no longer own their information, making it easier to be targeted, whether it be advertisers, governments, or political groups/actors. Google, Facebook, and most of the other cloud storage locations make money off selling your info. On the security

side, photos are constantly being transmitted over WiFi or cellular, both of which are susceptible to network, like Man-In-The-Middle attacks. Data centers cause other security issues by being bigger targets. While Google has some of the best security researchers working on keeping those data centers safe, one cannot guarantee that will continue to be the case. At one point, Yahoo! thought their security was impenetrable. Years go by and their infrastructure ages and we've now had multiple Yahoo! data breaches. A more direct example is the iCloud celebrity photo leaks from 2014. Those were gained through spear-phishing attacks, but spear-phishing attacks are easier when the storage location is well known, like iCloud. Since all these companies are closed source, we must trust that these companies handle our information properly. As companies like Equifax, whose hack consisted of *44.8% of the US population*, continue to get hacked, it's less easy to trust that these companies can keep our data safe.

Another tipping point for moving storage from the cloud to a local environment is cost. One aspect is the processing equipment. To run image recognition, which is required to implement the tagging features provided by the cloud storage options, one needs a GPU to process those images efficiently. But prices of GPUs continue to drop each day and computer vision gets easier, better, and more efficient. Now one does not need to rely on cloud servers for the processing when it can run on home or business desktops. Similarly, storage costs continue to drop as well. New 2TB drives can be had for $50. Apple's iCloud service requires monthly payment if one has over 5GB of data. For the US, a 2TB iCloud plan is $9.99/mo. If you like keeping your photos for more than 5 months, which most people do, it's far cheaper to keep them in your own home or business.

As mentioned above, most people use Google Photos or iCloud. Neither solve security or privacy issues. iCloud does better privacy-wise, but gets costly with more pictures. Nextcloud is a ready to go local storage application, but it sorely lacks in photo gallery features, like an image showcase UI and image recognition/tagging. If the goal is to make an alternative to the cloud storage providers, it must be able to do at least the core features of Google Photos or iCloud.

Our solution works in both homes and businesses. The backend we plan to use, Nextcloud, is HIPAA-certified and open source. Open source is a huge win for privacy and security because it allows auditing by millions. Google requires hiring the best in the field to keep their software secure, but hiring a limited number of people will always have blind spots. If users across the world can see the source code and submit pull requests for security issues, the solution gets more secure simply due to having more eyes on it diminishes the risk of blind spots by the developers. More people around the world hack for fun or for ethical purposes than malicious hackers. Open source leverages those ethical hackers to commit fixes before malicious hackers find them. Conversely, a closed source system, like Equifax, may have a security hole that a malicious hacker found and continued to exploit for months.

In terms of adoption, the first group would be privacy-oriented users. They like keeping stuff locally and many already have computer equipment that can function like a server. Later on, businesses would be able to serve their own flavor of the solution.

Data Sources, Scraping, and Database

The main data we will store are the photos that the users themselves upload. This can be from multiple sources, including desktops, laptops, iPhones, or Androids using the web interface. Anything on the same local network can upload to the Nextcloud instance.

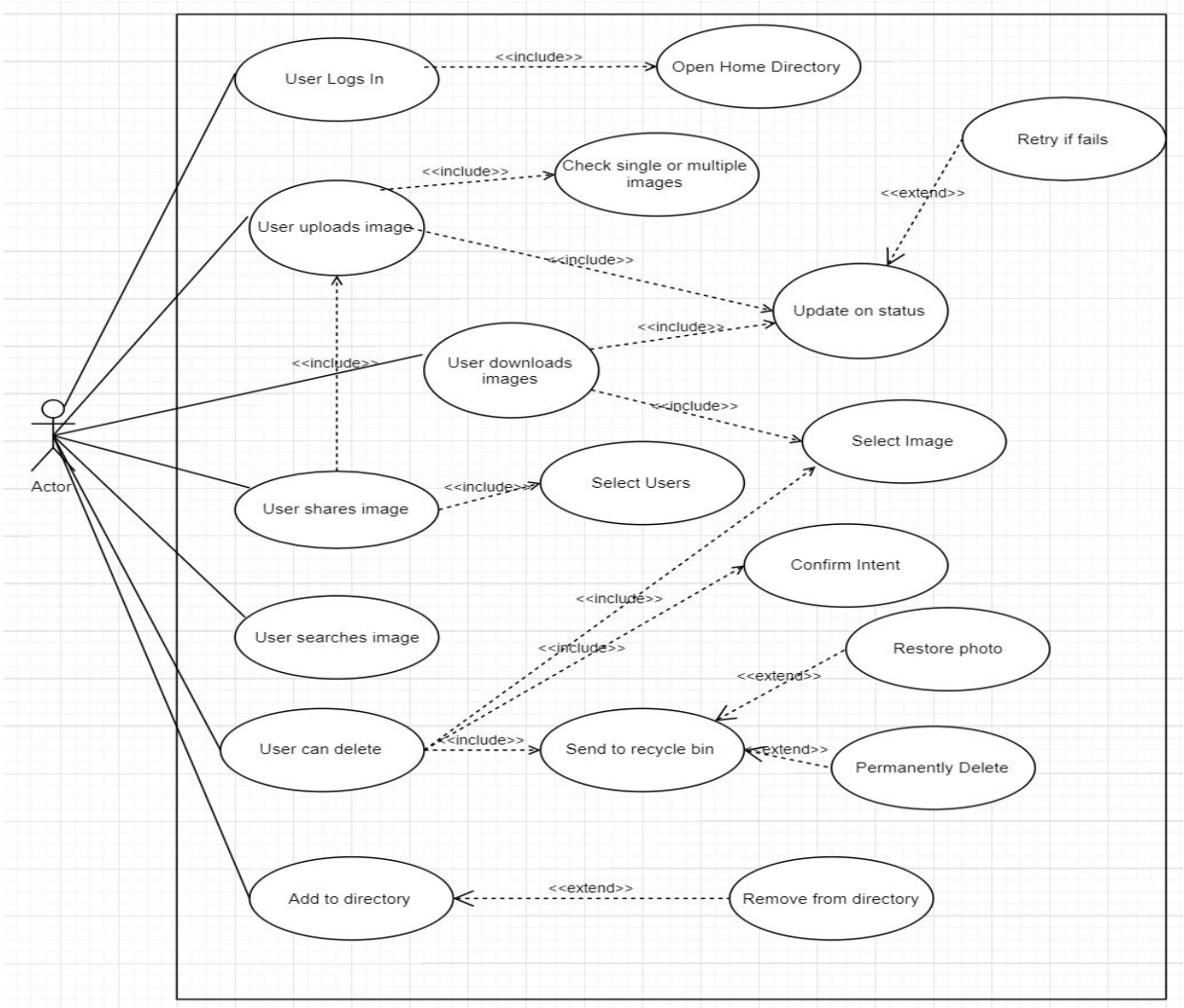
However, for data scraping, we can keep a copy of GPS coordinates to Points of Interest locations in the Nextcloud MySQL database. Searching the internet shows at least 3 sources we would be able to use. We would use this information so the user can search their photos for things like “Austin Convention Center,” which is something the image recognition would not be able to figure out on its own. This would function similar to Google’s Geocoding API, but is more beneficial for our use for four reasons. One, it helps the user be less connected to Google; two, it does not allow someone listening to network connections to figure out where the last image you uploaded was from as you have the entire set and are not looking for specific entries over the internet; three, keeps all your data locally, meaning the entire application can still run without internet access; and four, does not cost any money, unlike Google’s APIs.

Requirements

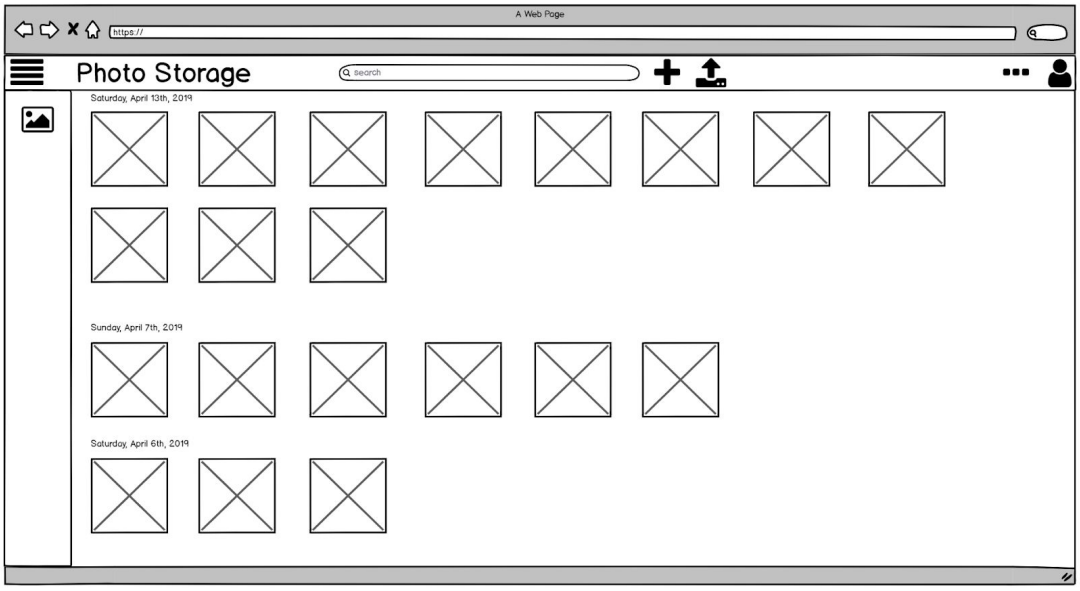
1. User logs in and accesses the home directory to view his/her photos
    a. Home directory by default displays the user’s images in reverse chronological order.
    b. **User Story:** “As a user, I want to log in to the home directory so that I can view my photos.”
2. User uploads images
    a. App asks user if single or multiple images are being uploaded
    b. Updates user on upload status
        i. If upload fails, ask user to retry or cancel
    c. **User Story:** “As a user, I want to upload an image/images for secure storage so that I can access them later.”
3. User downloads images onto local device
    a. Ask user to select the images to be downloaded; can select multiple images
    b. Inform user on download status
        i. If download fails, recommends user to select fewer images, cancel operation, or retry
    c. **User Story:** “As a user, I want to download images on my computer so that I can print them.”
4. User shares images on the shared drive
    a. User can upload images to a shared drive in home directory
    b. User can select which users in the network can access their shared images
    c. **User Story:** “As a user, I want to share my images so that my family see them.”
5. User searches for images based on category/tag/time
    a. User specifies in the search bar a time, location, or image category, and the best matching results are displayed
    b. **User Story:** “As a user, I want to search images so that I will not have a hard time finding it”
6. User can delete photos
    a. User can select list of images to delete

b. App confirms users intent to delete photo
c. User asked to undo/restore a photo from the recycle bin
d. **User Story:** “As a user, I want to delete photos so that I do not have ugly photos whatsoever.”

7. User can add photos to a favourites directory
    a. User can select list of photos to star and add to a favourites directory
        i. User can remove images from the favourites directory
    b. **User Story:** “As a user, I want to create a directory of 2019 photos so that I can look back on them later.”



Interface



Planning and Scheduling

Phase	Overview	Details
I. Oct 7	Implement parts of backend to connect and integrate with the cloud	- Develop an individual image interface that uses YOLO to classify an input image - Develop a wrapper around the individual image interface, that categorizes the image based on the caption provided by YOLO - Nextcloud, a local file hosting service, will be used for storing and accessing the image data - Data scrape GPS ↔ location data from host websites
II. Oct 25	Implement the use cases, while parallely developing the web app user interface	- 4 out of 7 use cases and their alternate flows implemented and tested - The use cases tested here are upload, download, share, and the user account/home directory organization - Backend invokes YOLO's deep neural network to tag the image, and the image is then categorized and stored in the Nextcloud database - Basic user interface for the use cases implemented - Web app source for use cases written in PHP - Begin use of PHPUnit for testing
III. Nov 15	Test all the features and specific units	- Implement and test the remaining use cases - Complete the user interface using Javascript and React in a Node environment - Implement Selenium design tests - Ensure web app is fully functional

IV. Nov 25	Refactoring & Applying Design Pattern	- Refactor and apply design patterns - Test the software with these new changes

Tools, Software, Frameworks

The primary tool used in this project is Nextcloud. Nextcloud will provide local storage to store photos on a machine in the user's local network. These photos will be uploaded through a dedicated web application. This web application will be created with a combination of Javascript and React, built for a Node environment. Image processing will be handled through YOLO, a C++ image recognition system. YOLO comes with a neural network named darknet developed by the same person. Additionally, YOLO has default weights with classifications for over 9000 objects. The YOLO associated set of tools handles this part for us so we do not need to reinvent the wheel. Nextcloud's web application toolkit also includes PHP (with PHPUnit for unit testing) and MySQL for the backend.

Because most of the front-end programming is built for the front-end, we intend to use web development tools that are included in the browser of choice. We may also utilize Selenium for comprehensive design tests.

We also are using tools for us to manage a project of this scope. We will be keeping our code on GitHub in a group so that we can have multiple repositories for different aspects of the project. Additionally, all non-code files, such as this report, are kept on a team Google Drive (semi-ironic). Lastly, we have created a Slack group with dedicated channels for things like the application development vs general communication.

Feasibility

The biggest challenge is YOLO (You Only Look Once), a C++ image recognition system, integration into the app. We cannot guarantee YOLO/web app integration without trying it. If YOLO does not integrate well, we can change YOLO source code, but that comes at a serious cost to timeline. We can minimize this risk by attempting YOLO integration first, giving time for changes. Other challenges are unfamiliarity with Linux and PHP. The Nextcloud server only runs on Linux and expects the web apps to be written in PHP. Some of us have considerable Linux experience while others do not. However, by segmenting the work so parts contain actual goals and stretch goals, we can adapt to keep to the deadline, yet still shoot for the moon if we can make it there.

Feedback

We have addressed all feedback in the sections above.