Product Backlog: Photo Storage Platform

Group 28: Alex Goswick, Rachel Hawley, Samara Saquib, Evan Shaw

Problem Statement:

Modern cloud apps exist to store and manage one's photos, however these services often mandate the recurring purchase of a cloud subscription, degrade image quality, or keep the user from migrating to another platform due to vendor lock-in. Applications available for purchase or download do not implement their management systems in a client/server model. This makes it impossible for multiple users to use the software at the same time, such as family members who share a single photo library. In addition, very few of these solutions expose an open API, which could be useful to integrate the system with third party apps. In this project, we propose a fully self-hosted Photo Management Platform, complete with many useful features.

Background Information:

Audience

Almost every citizen in the developed world has a camera in their hand, which has led to a world filled with pictures but no centralized way of storing, managing, and editing them. Modern platforms for photo storage and sharing often restrict their users in what they can do as well as charge them to use their services. The lack of a dedicated all-in-one software has led to people storing their photos over multiple cloud and desktop platforms. Anyone who enjoys taking pictures can benefit from having one place to store, manage, and edit their photos.

Similar Platforms

Google Photos and Adobe Lightroom are the two most similar platforms that exist. Google Photos is a free service that stores all your photos in the cloud, but has some limitations. Adobe Lightroom provides a robust editing platform for photos, but does not support advanced features for managing your data.

Limitations

The current platforms' main limitations are that not all platforms have all the features that users want to use. This means users are forced to use and pay for multiple platforms' services in order to meet their needs. Google Photos reduces the resolution of all the photos you upload, or charge you a monthly storage fee. This is bad because details in your photos could be lost forever. With the increasing resolution of displays, these photos may look inadequate on future devices. Adobe Lightroom lacks critical multiuser support, meaning that only one computer can access the local dataset at the same time. Also, Adobe Lightroom does not support datasets stored on a network drive

or a server, nor does it support automatically uploading photos from your phone. While Lightroom has the ability to backup the library catalog, it does not have an easy solution to backup the actual photos, leaving your data vulnerable to hardware failure, natural disasters, or a rogue user deleting everything. Our platform aims to fix all of these flaws.

Functional Requirements:

1. As a user, I would like the photos to be stored in a central location, and accessible at any time from any computer.
2. As a user, I would like to be able to set up the system on my own computer and not be tied to a cloud service.
3. As a user, I would like the system to be easy to set up with minimal install.
4. As a user, I would like this data to be password protected and available to authenticated users.
5. As a user, I would like to be able to have a library that can scale to 10+ terabytes.
6. As a user, I would like to be able to use the desktop app on multiple operating systems.
7. As a user, I would like to be able to upload photos from the desktop app.
8. As a user, I would like to be able to view photos from the desktop app.
9. As a user, I would like the metadata formats of many photos to be recognized.
10. As a user, I would like to be able to view the entire metadata from a single photo in the desktop app.
11. As a user, I would like to be able to browse photos by date from the desktop app.
12. As a user, I would like to be able to browse photos by geolocation from the desktop app.
13. As a user, I would like to be able to view videos from the desktop app.
14. As a user, I would like to be able to delete photos and videos from the desktop app.
15. As a user, I would like to be able to fix red-eye effects from the desktop app.
16. As a user, I would like to be able to adjust the color curves from the desktop app.
17. As a user, I would like to be able to retouch photos from the desktop app.
18. As a user, I would like to be able to edit the photos without changing the original photo (non-destructive) from the desktop app.
19. As a user, I would like to be able to undo and redo edits of photos.
20. As a user, I would like to be able to create and edit albums/folders of photos that are shared with authenticated users.
21. As a user, I would like to be able to view photos from a mobile device.
22. As a user, I would like to be able to manually upload photos from a mobile device.
23. As a user, I would like to be able to automatically upload photos from a mobile device in the background without any user intervention.

24. As a user, I would like the system to automatically recognize duplicate files and stop their upload to avoid clutter.
25. As a user, I would like the system to automatically recognize near duplicate files (like barely cropped files) and stop their upload to avoid clutter.
26. As a user, I would like to be able to download photos in multiple file formats.
27. As a user, I would like the system to load quickly and be responsive.
28. As a user, I would like the system to support Apple Live Photos (JPEG + HEIC).
29. As a user, I would like to be able to easily backup my photos.
30. As a user, I would like this backup system to keep track of the state of the photo library at different dates and be able to roll back to a specific date in time.
31. As a user, I would like this backup system to not take up too much more space than the actual photo library.
32. As a user, I would like the system to be compatible with third-party apps that anyone could write.
33. As a user, I would like to share photos with other people.
34. As a user, I would like to enable multi-factor authentication.
35. As a user, I would like to share photos directly to social media applications.
36. As a user, I would like to store an email and phone number with my account in case I forget my password.

Non-Functional Requirements:

Architecture and Performance

The desktop app will be written in React while the mobile application will be developed with a tool like React Native.
The back-end of the project will be a server written in Java implementing a custom written API to interface with client devices. The server will be connected to a MySQL database to store metadata, and will store the photo/video file directly on the filesystem. Our backend will also include a security framework to protect users' data from exploits. Our app will aim to complete requests at near network/storage speeds. The upper bound for acceptable performance would be 100 milliseconds latency and handle around 30 requests at a time. Each request would be either an API call to get some information about the data or an image request.

Security

As the case is with all photo storage applications, privacy is an important concern. Our application will ask for permission before accessing a user's Photo Library, Camera and Location Services. Our application will not make changes to photos in their device's photo library without their permission. Our application will not be tied to any Cloud services, and a password will be required to access your account. We will use a security framework to be sure that personal data will not be vulnerable to software exploits.

Usability

The desktop app will be the primary interface to the software, so it should have a simple layout and be easy to understand and navigate. Many photo management platforms have an abundance of tabs and features all in one place where it may be difficult to find what you want to use, so it is very important for the interfaces to be well organized so that it is very intuitive for users. It is also important for the application to be properly accessible across all browsers and platforms due to different screen sizes and resolutions affecting the appearance of the interfaces.

Hosting/Deployment

The server should be packaged in a Docker container and integrated into a continuous development workflow. It should be primarily targeted to run on a Linux distribution like Ubuntu, CentOS, or Arch, but in theory it could run on any server since it will be Java.