

FIT5225 Cloud computing and security

ASSIGNMENT 2: Aussie EcoLens: A Multi-Cloud

Serverless Wildlife Observation Platform

Group Report

Due date: Monday 8 June 2026, 23:55

Group 58

Student's name and ID:

Jinyuan Chen: 35598433

Peng Mu: 34451994

Keshu Zhang: 36436763

You Ni: 35358076

Department and University: Faculty of Information Technology, Monash University

Data of submission: Monday 8 June 2026

GenAI Usage Statement

We used generative AI (Claude, Gemini, Figma Make) in a limited way to support this assignment.

GenAI were used to

- Programming assistance;
- Improve the structure and flow of our writing;
- Refine report

Table of Contents

GenAI Usage Statement.....	2
Architecture.....	4
Contributions.....	6
App User Instruction.....	7
Sign In.....	7
Main Navigation.....	9
Gallery and Library.....	10
Upload Media.....	12
Query Media.....	15
Modify Tags.....	19
Delete Media.....	20
Subscribe & Notification.....	21

Architecture

The Architecture of this project (Figure 1.1) is based on the classic B/S model. Business across three cloud providers, AWS, GCP and Azure.

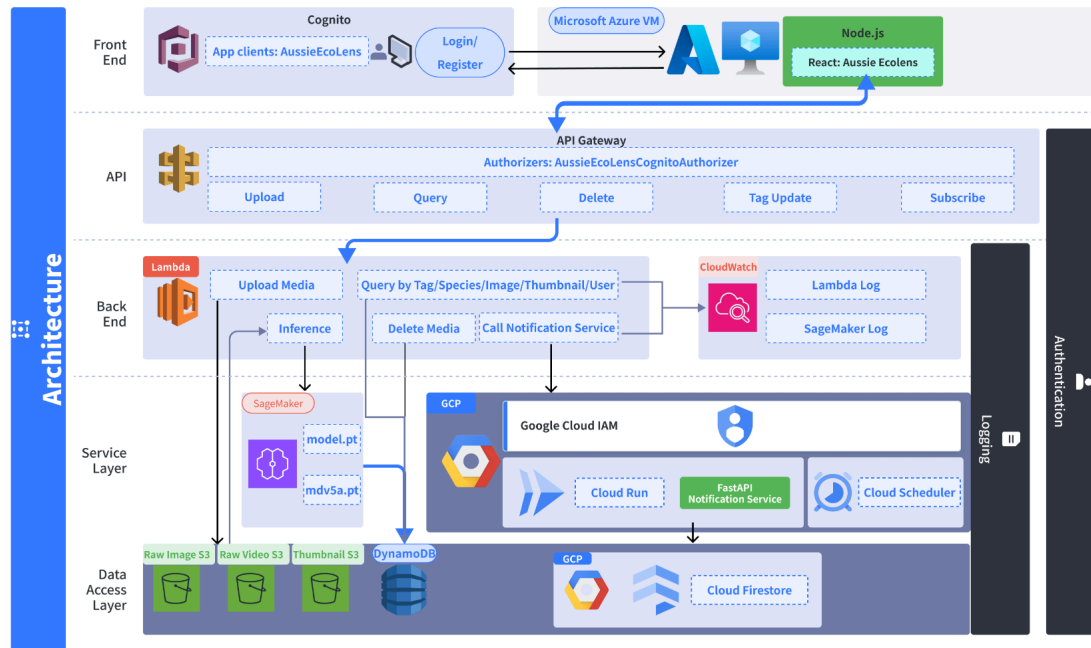


Figure 1.1 Architecture

The Frontend is based on React Framework, deployed on an Azure instance with a domain name. It can be publicly accessible via this link:

<https://aussieecolensgroup58.japaneast.cloudapp.azure.com/>. The login and registration will redirect to Cognito.

All API endpoints are secured by a Cognito Authorizer. They can be divided into 5 types.

The amount of active lambda is 16. The business logic is robust to validate filename and large (up to 70MB) or corrupt image files. To optimize the cold-start of lambda, we deployed the mdv5a.pt and model.pt to SageMaker ml.m5.xlarge instance. The CloudWatch also reveals that the memory usage is about 7.5GB, which suggests that deployment on the SageMaker instance is reasonable. The update process of the model is isolated with the lambda. When the model is updated, there will be no need

to stop the code.

The notification service is implemented by FastAPI, containerized and deployed on Google Cloud Run. The subscribe and notification endpoints are additionally secured by Google Cloud IAM.

The S3 is divided into 3 buckets. The video and image will share the same bucket to store their thumbnails. The S3 Buckets are under the control of the Policy.

The DynamoDB table design illustrated in Figure1.2. Any access to the DynamoDB must go through the lambda.

Attribute	Type	Key Type	Example
image_hash	String	Partition key	9a8f2c3e1b...
owner_user_id	String	Sort key	xxx-xxxxx-xxxx
owner_email	String	-	user@example.com
file_type	String	-	jpg, png, bmp / mp4
original_url	String	-	s3://.../id_hash_raw.png
thumbnail_url	String	-	s3://.../id_hash_thumb.jpg
animals_detected	Map (JSON)	-	{"koala": 3}
manual_tags	Map (JSON)	-	{"magpie": 1}
upload_time	String	-	2026-06-08 16:11:20
view_permission	String / Boolean	-	'owner_only' / 'registered_users'
tag_edit_permission	String / Boolean	-	'owner_only' / 'registered_users'

Figure 1.2 Database design

Contributions

The workloads of each student are well allocated. The contributions are indicated in Table 2.1.

Table 2.1 Students Contribution

Student Name and ID	Contribute Rate	Contribution
Jinyuan Chen 35598433	25%	I mainly work as the media uploading and processing lambda developer, and the model inference function developer. Specifically, I implemented 6 active lambdas which allow Frontend to request a presigned URL to store media to S3, ensuring robustness and efficiency. And, I designed the validation logic to ensure the files upload to S3 and Inference is corrected. Also, I packaged the model and implemented inference code to SageMaker. In addition, I implemented a lambda to interface with GCP notification service. Furthermore, I launched and configured 3 main S3 Bucket, as well as deploying public-accessible React Front-end to Azure.
You Ni 35358076	25%	Responsible for authentication, security, notifications, and multi-cloud integration. Configured AWS Cognito Hosted UI, API Gateway authorisers, and token-based access control. Designed and implemented the notification subsystem using GCP Cloud Run and Firestore. Implemented secure AWS–GCP communication through Workload Identity Federation. Performed integration and edge-case testing, identifying defects involving uploads, duplicate detection, validation, and tag management. Contributed to architecture discussions and system integration across AWS and GCP services.
Keshu Zhang 36436763	25%	Implemented backend APIs for media search and management, including search by species, tag-count conditions, thumbnail URL, uploaded query file, uploader ID, and public gallery items. Developed bulk tag add/remove and delete functions, with Cognito-based user identification, permission checks, input validation, and clear error handling. Integrated the functions with DynamoDB media metadata and ensured consistent response formats for frontend use.
Peng Mu 34451994	25%	Responsible for frontend UI, user-facing integration, and demo support for Aussie EcoLens. Designed and implemented the React interface for Library, Gallery, Upload, Search, Edit Tags, Subscribe, Notifications, and Media Detail pages. Integrated presigned S3 uploads, multiple search modes, bulk tag editing, subscription flows, and owner-restricted deletion feedback with backend APIs. Also contributed to the app user guide, report integration, and end-to-end testing for the final demo.

App User Instruction

Sign In

Users must sign in through AWS Cognito before accessing protected features.

1. Open the Aussie EcoLens website.
2. Click **"Sign in / Sign up with AWS Cognito"**.
3. Sign in with an existing account or create a new account.
4. After authentication, the user is redirected to the application.

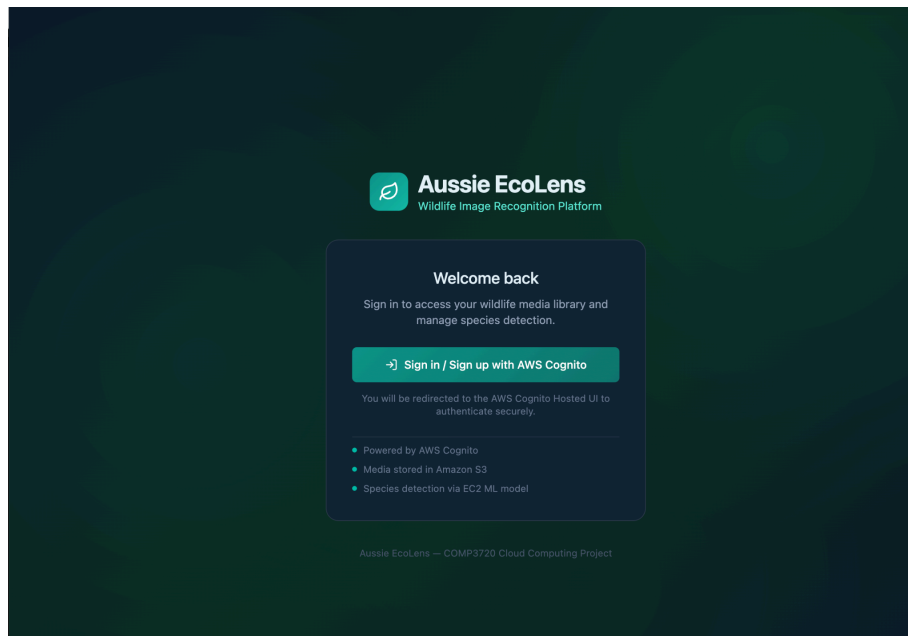
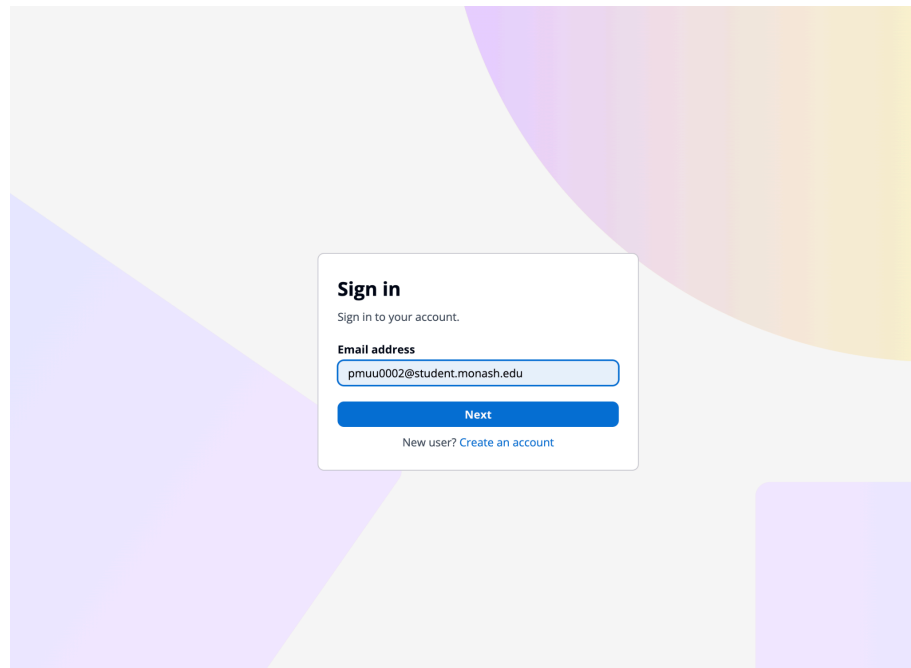


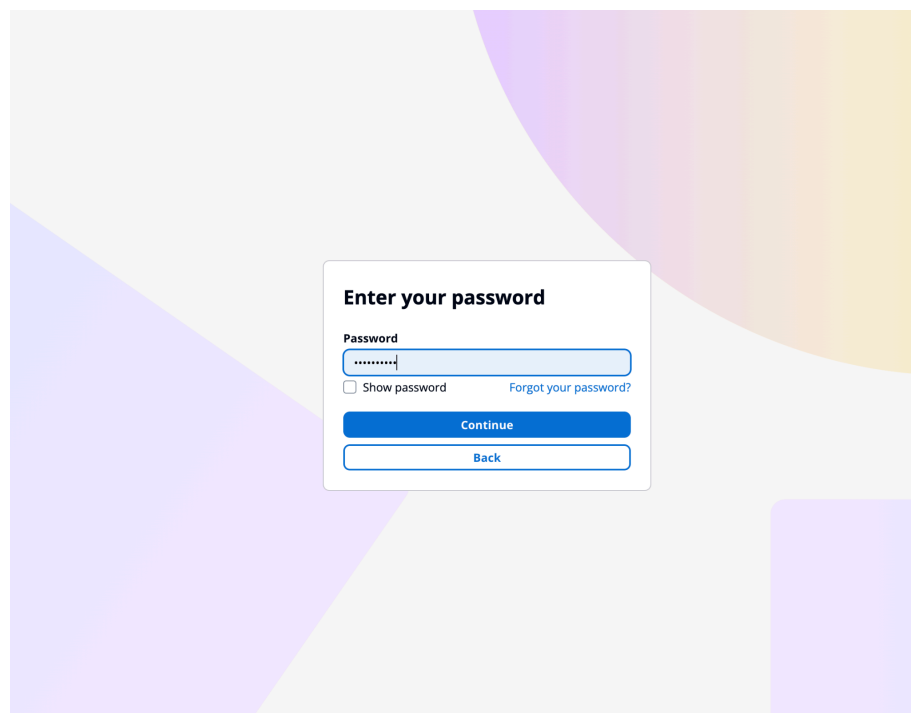
Figure 3.1 Redirect Link



The image shows a 'Sign in' form on a page with a light gray background and large, abstract, overlapping shapes in shades of purple, blue, and yellow. The form is a white card with a thin gray border. It contains the following elements:

- Sign in**: A bold heading.
- Sign in to your account.: A sub-heading.
- Email address**: A label for the input field.
- : A text input field containing the email address.
- Next**: A blue button with white text.
- New user? [Create an account](#): A link for new users.

Figure 3.2 Cognito Login/Register Page



The image shows an 'Enter your password' form on the same background as Figure 3.2. The form is a white card with a thin gray border. It contains the following elements:

- Enter your password**: A bold heading.
- Password**: A label for the input field.
- : A password input field with masked characters.
- ☐ Show password: A checkbox to toggle password visibility.
- [Forgot your password?](#): A link for users who have forgotten their password.
- Continue**: A blue button with white text.
- Back**: A button with a blue border and blue text.

Figure 3.3 Input password

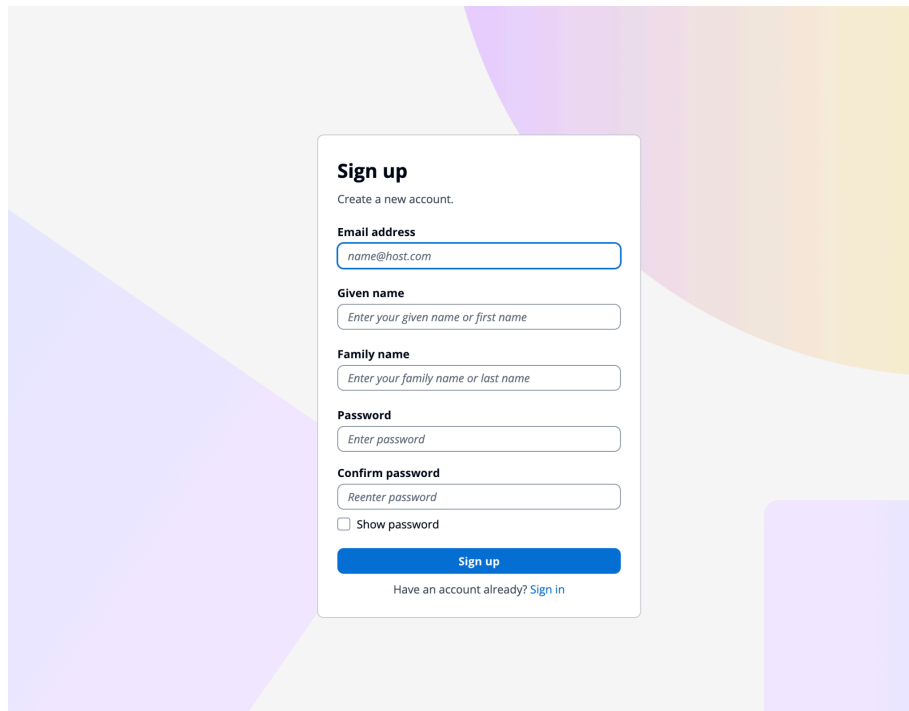
The image shows a 'Sign up' form for a new account. The form is centered on a background with abstract purple and yellow shapes. It includes fields for 'Email address' (with placeholder 'name@host.com'), 'Given name' (placeholder 'Enter your given name or first name'), 'Family name' (placeholder 'Enter your family name or last name'), 'Password' (placeholder 'Enter password'), and 'Confirm password' (placeholder 'Reenter password'). There is a 'Show password' checkbox and a blue 'Sign up' button. Below the button is a link: 'Have an account already? Sign in'.

Figure 3.4 Cognito Register Page

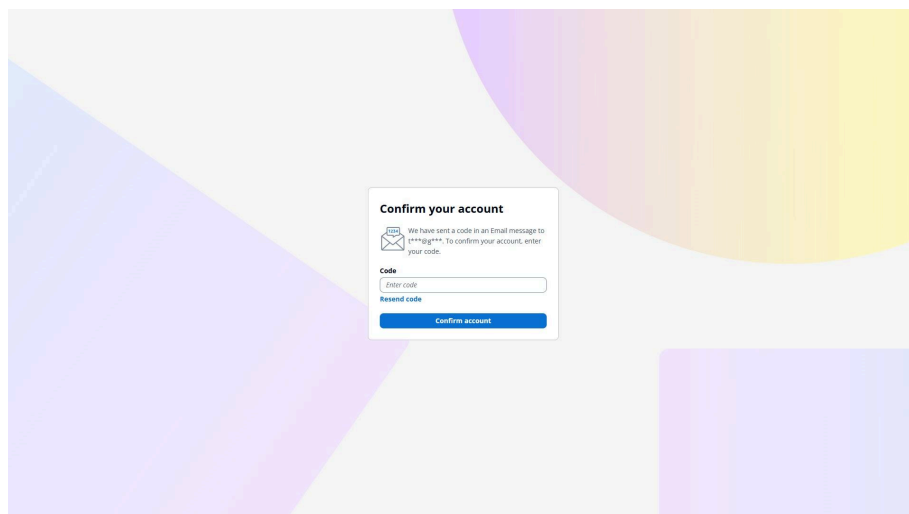
The image shows a 'Confirm your account' form. It features an email icon and text: 'We have sent a code in an Email message to ****@****. To confirm your account, enter your code.' Below this is a 'Code' input field with a placeholder 'Enter code', a 'Resend code' link, and a blue 'Confirm account' button.

Figure 3.5 Email with code is sent

Main Navigation

The main application contains:

- **Navigation Bar** – Access major application features.
- **Species Sidebar** – Select a species to automatically populate the Search page.
- **User Area** – Displays the signed-in user and sign-out option.

- **Status Bar** – Shows the status of API, model service, storage, authentication, and notification services.

Gallery and Library

Users can view their uploaded media in the **Library** page, and view all public media in the **Gallery** page. Click the thumbnails to get more details in the media page.

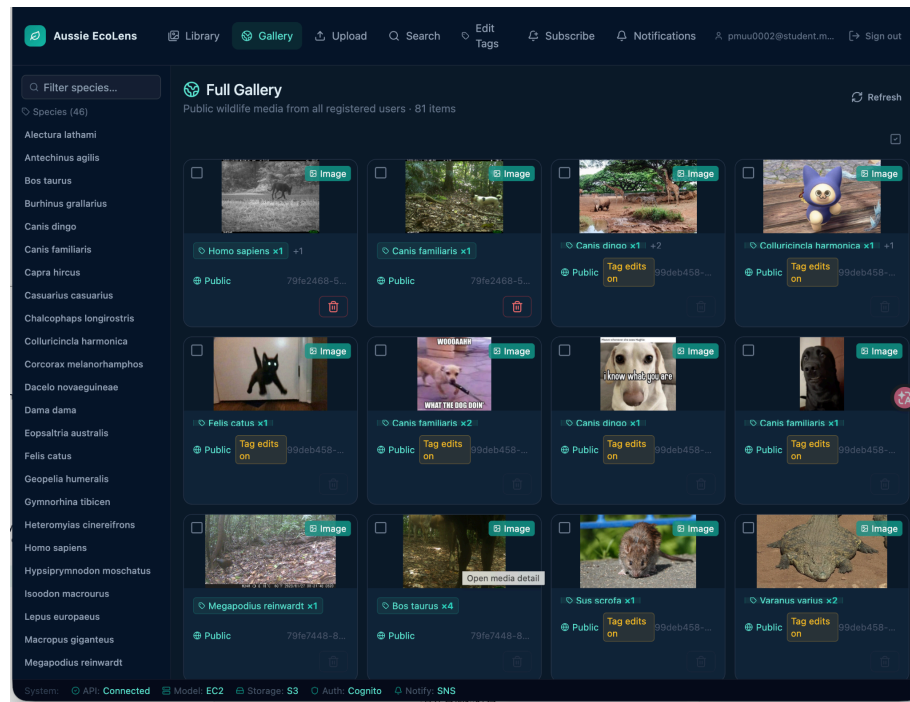


Figure 3.6 Gallery page

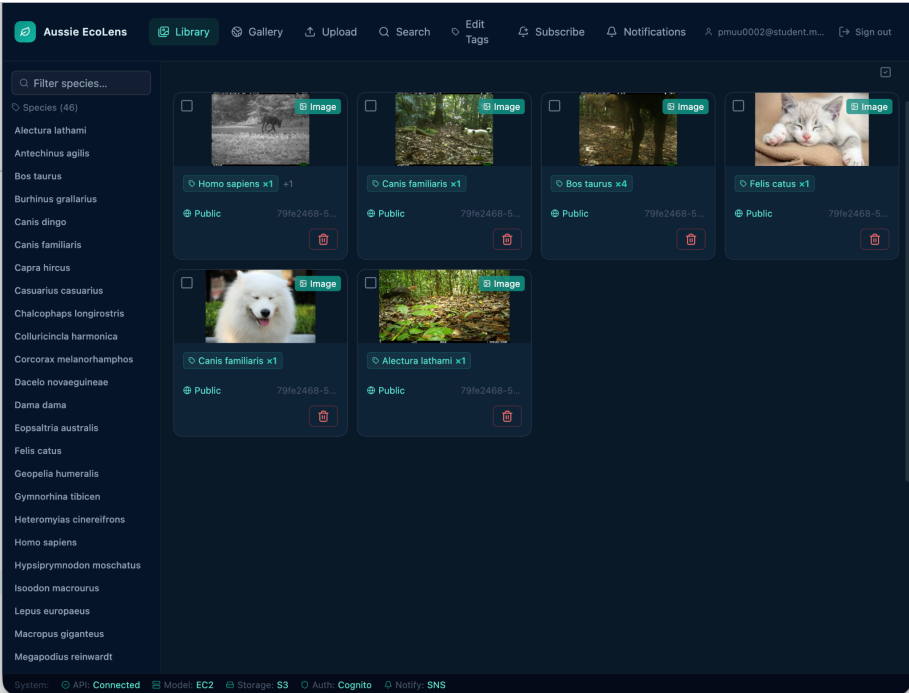


Figure 3.7 Media Library

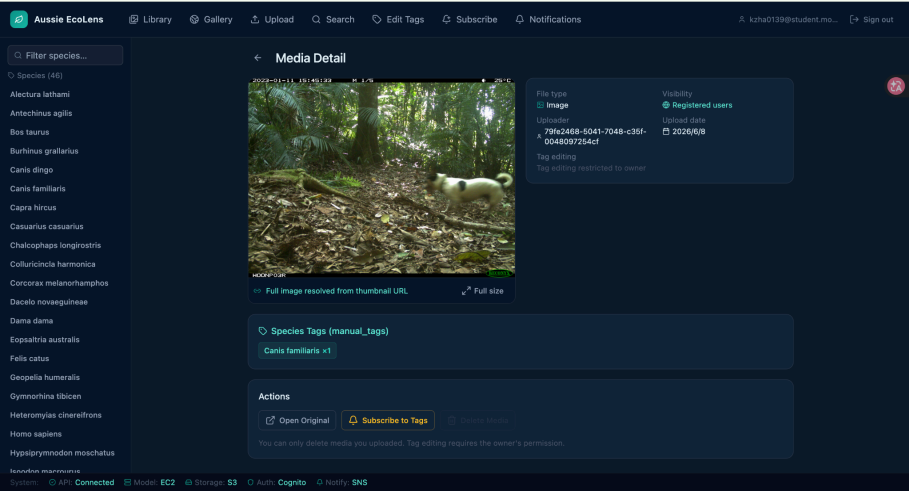


Figure 3.8 Media detail page

Upload Media

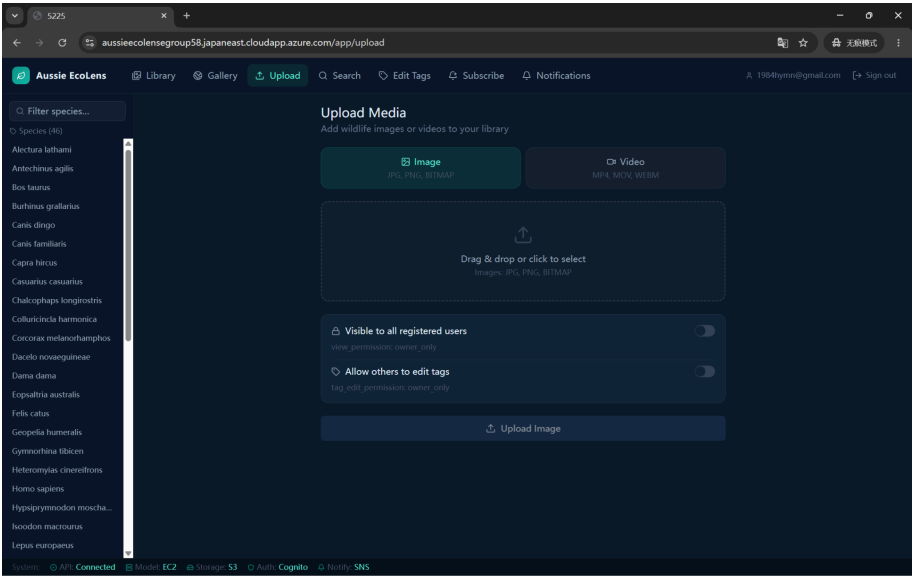


Figure 3.9 Upload Page

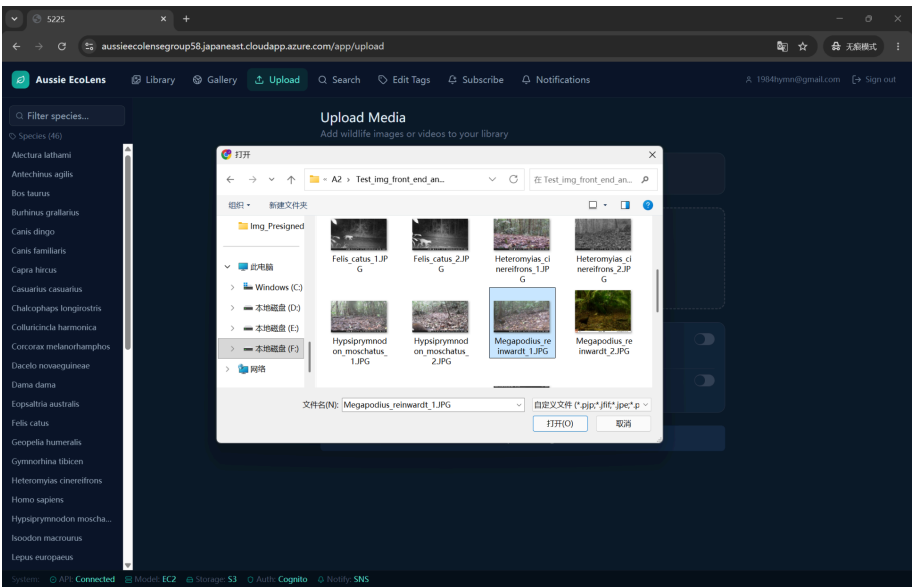


Figure 3.10 Choose from filesystem

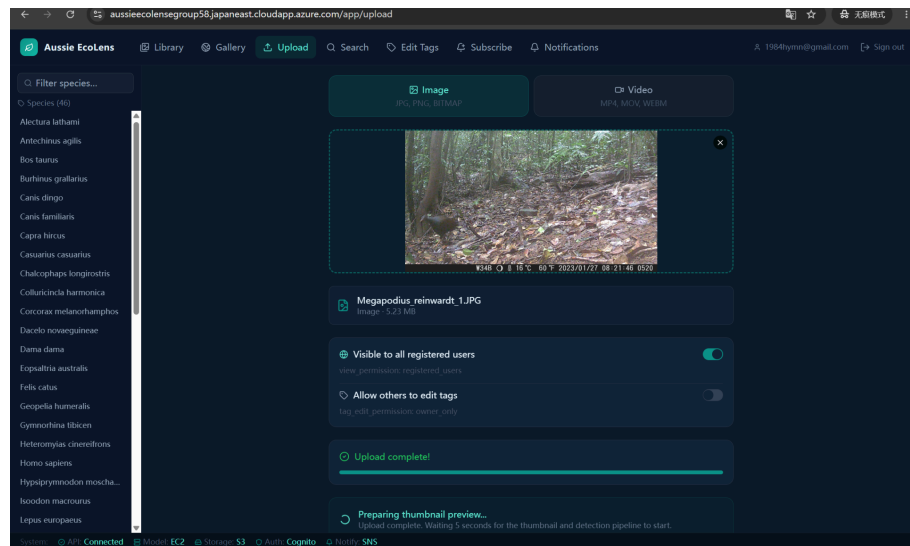


Figure 3.11 Configure permission and upload

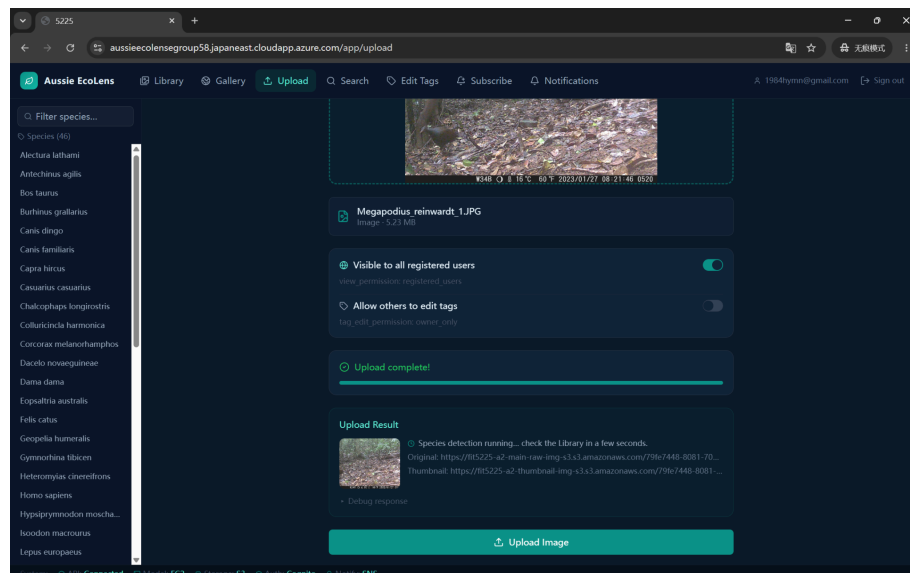


Figure 3.12 Upload successful

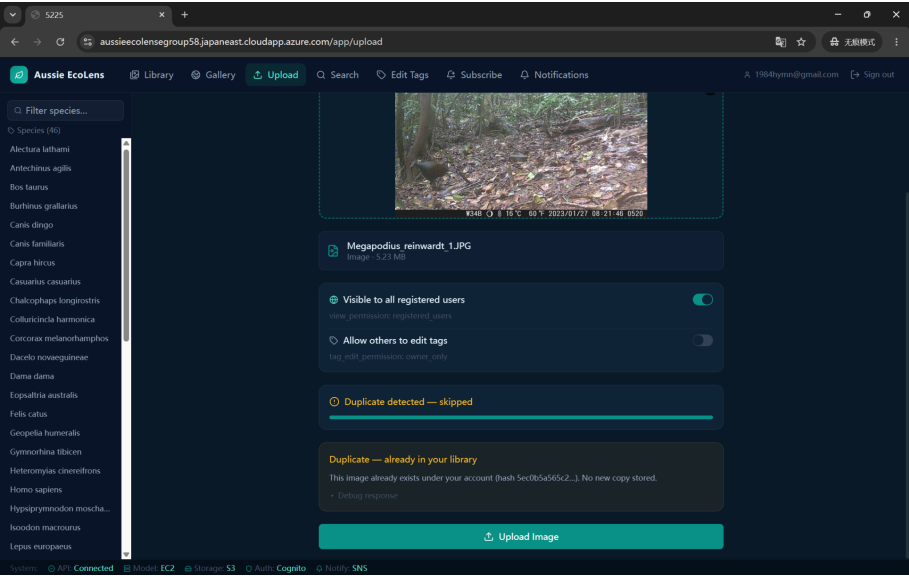


Figure 3.13 Try upload duplicate

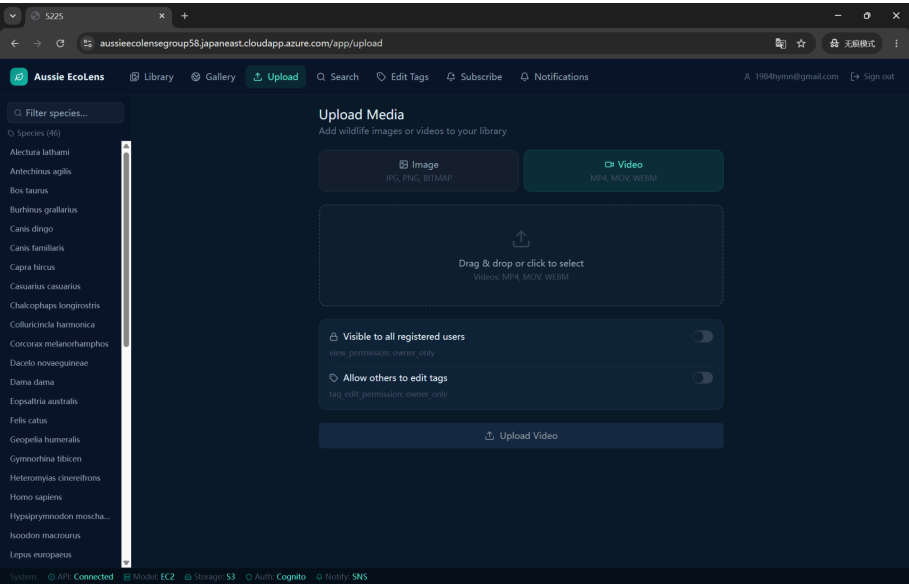


Figure 3.14 Upload video

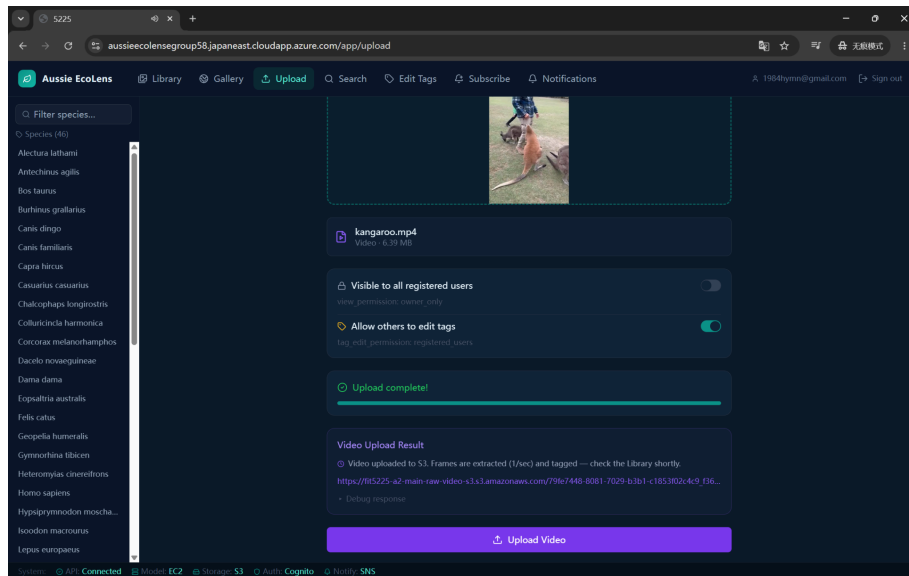


Figure 3.15 Upload video successful

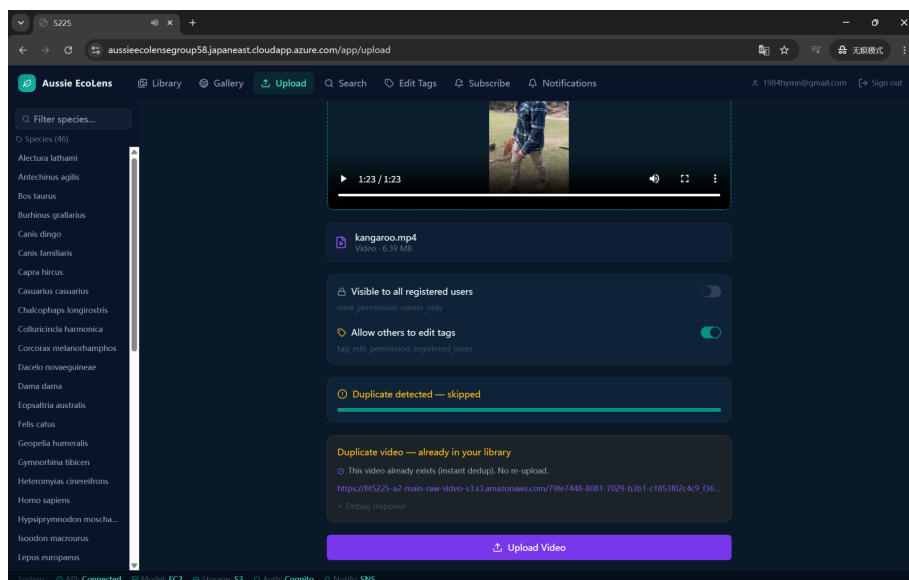


Figure 3.16 Try upload duplicated video

Query Media

Users can view their uploaded media in the **Library** page, and view all public media in the **Gallery** page. Click the thumbnails to get more details in the media page.

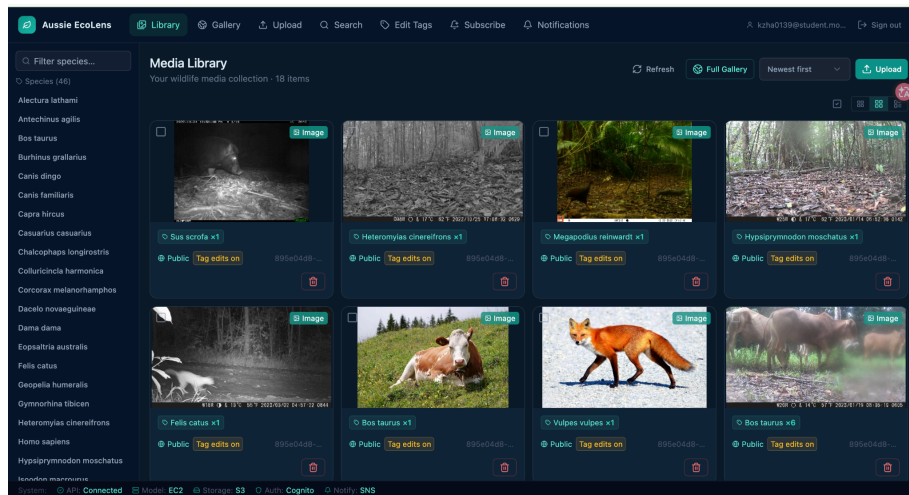


Figure 3.17 Own updated media library

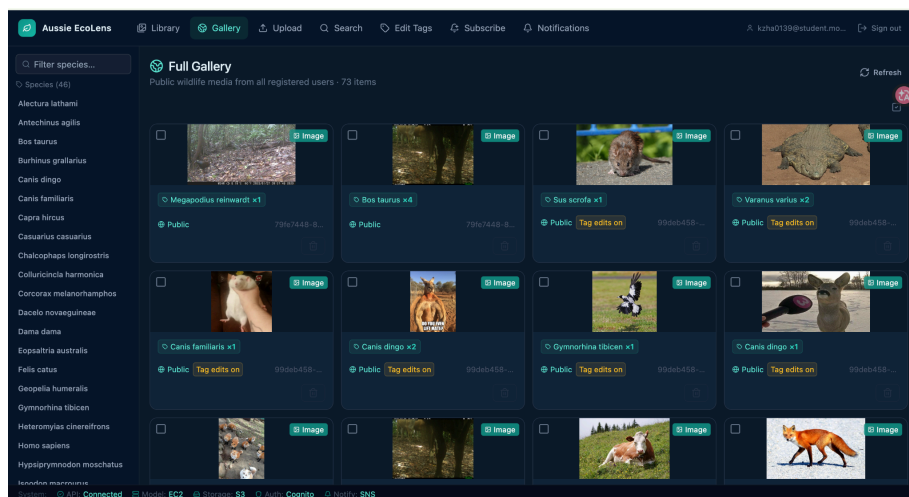


Figure 3.18 All public media Gallery

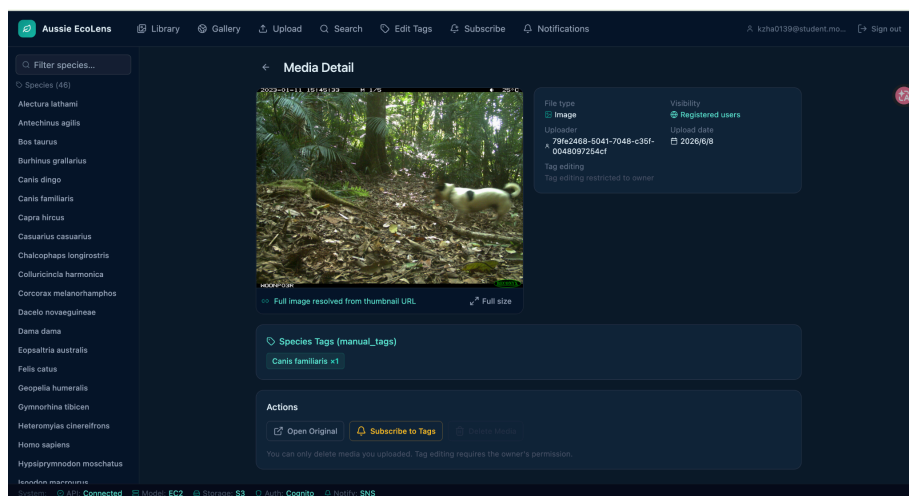


Figure 3.19 Media detail page

Users can **Search** media in several ways.

“**By Tags & Count**” supports multiple tag-count conditions with AND conditions.

“**By Species**” finds media containing at least one selected species.

“**By Thumbnail URL**” returns the specific media based on the thumbnail url.

“**My Uploads**” can view users own media, same as “Library”.

“**By User ID**” finds accessible public media uploaded by a specific user.

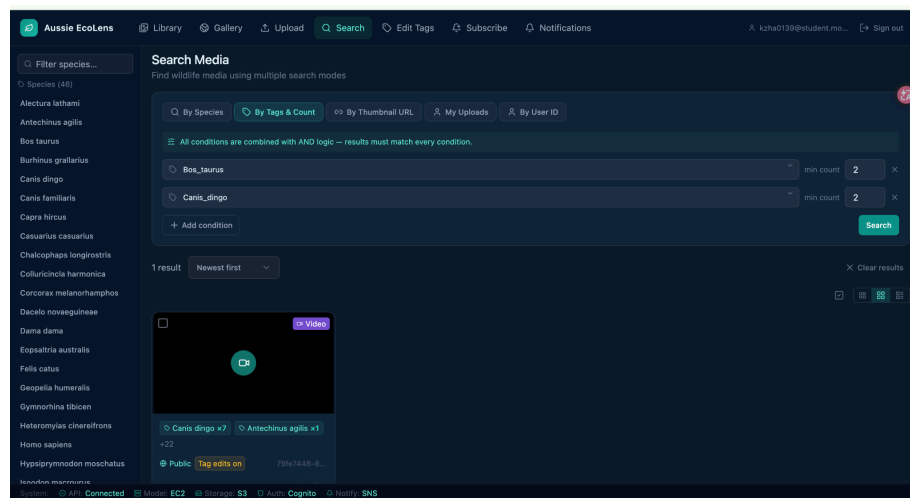


Figure 3.20 Query by tags (public)

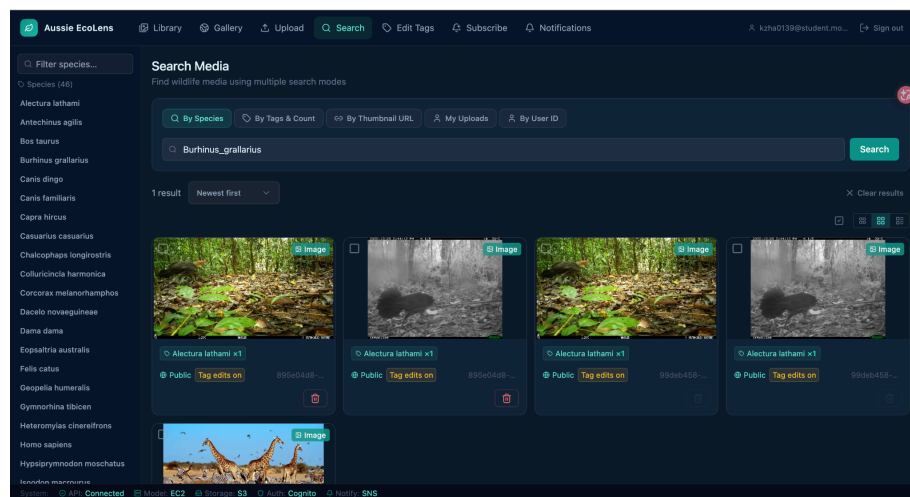


Figure 3.21 Query by species (public)

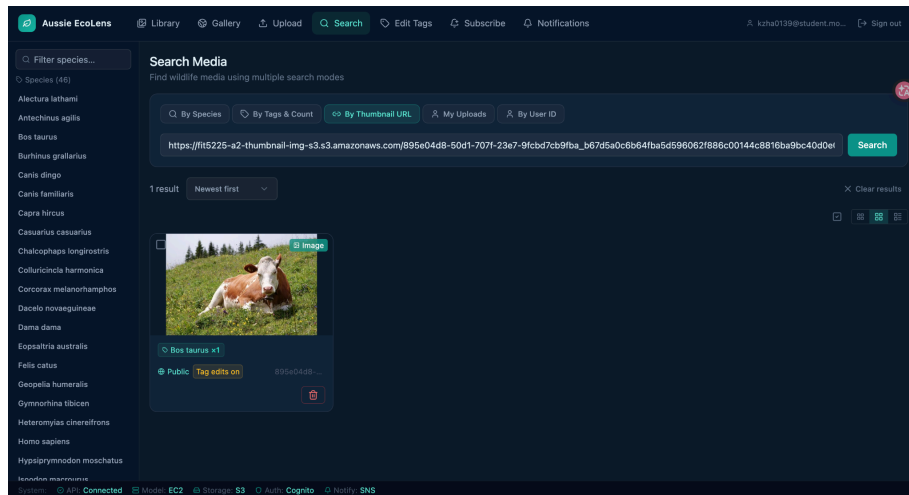


Figure 3.22 Query by thumbnail (public)

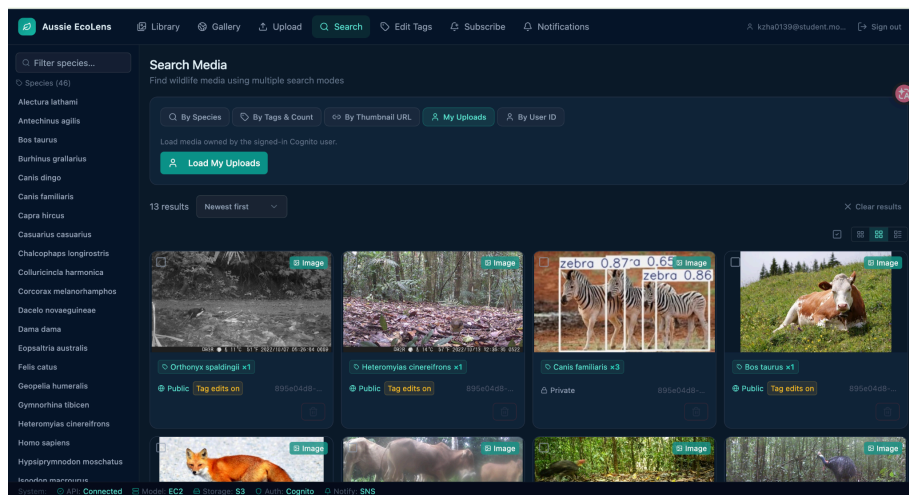


Figure 3.23 Query own uploads (Library)

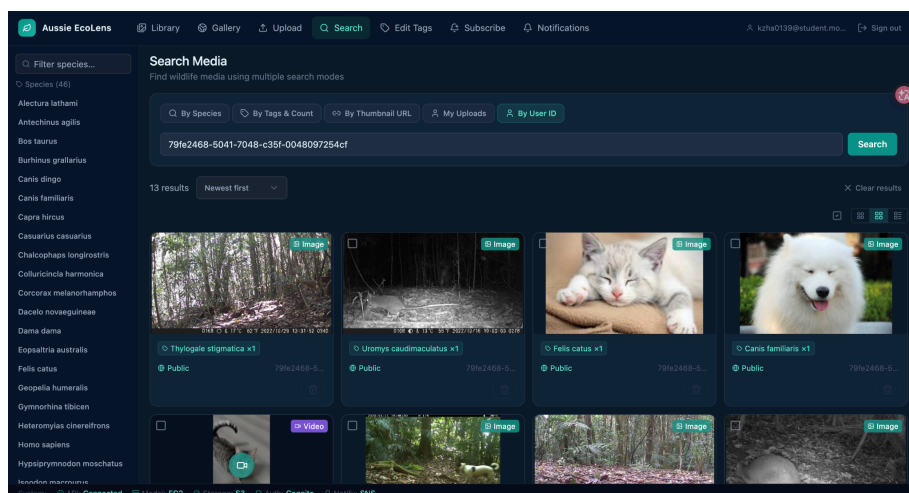


Figure 3.24 Query by owner_id (public)

“By Query Image”: Users can upload a query image, and the system temporarily detects its species tags to search for matching existing media without saving the uploaded query file.

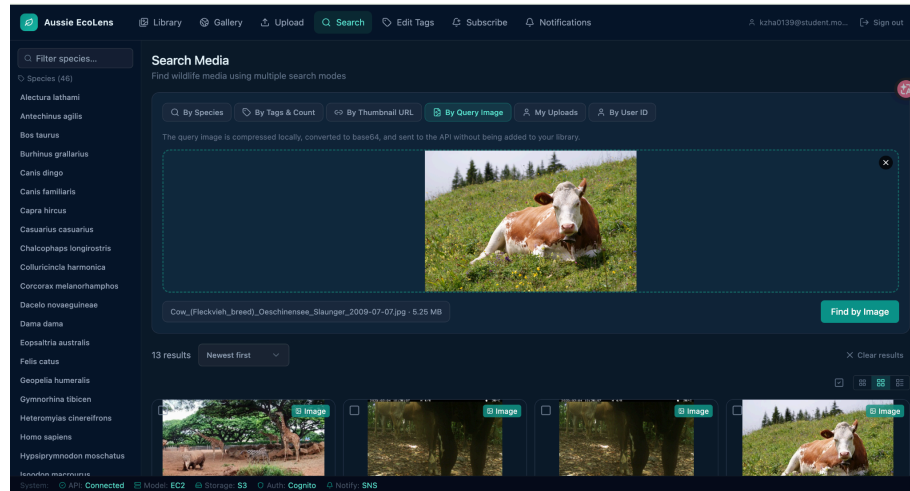


Figure 3.25 Query by upload image

Modify Tags

Users can use **“Edit Tags”** to select multiple media items and update or remove species tags in bulk. Items without edit permission are skipped automatically.

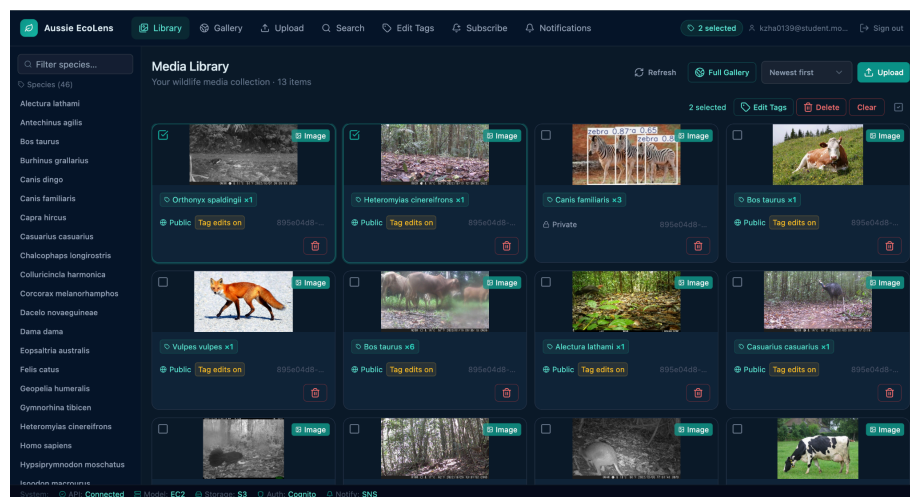


Figure 3.26 Select multiple media for delete/edit tags

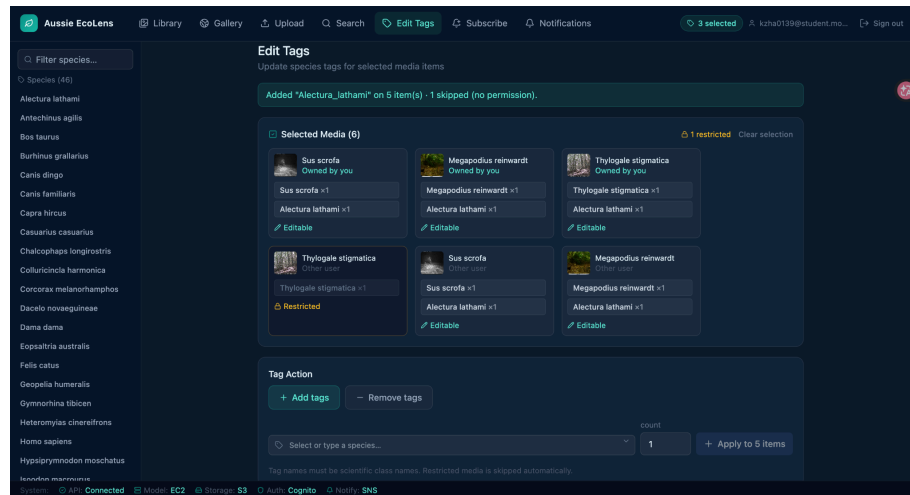


Figure 3.27 Bulk update tags (table edits on)

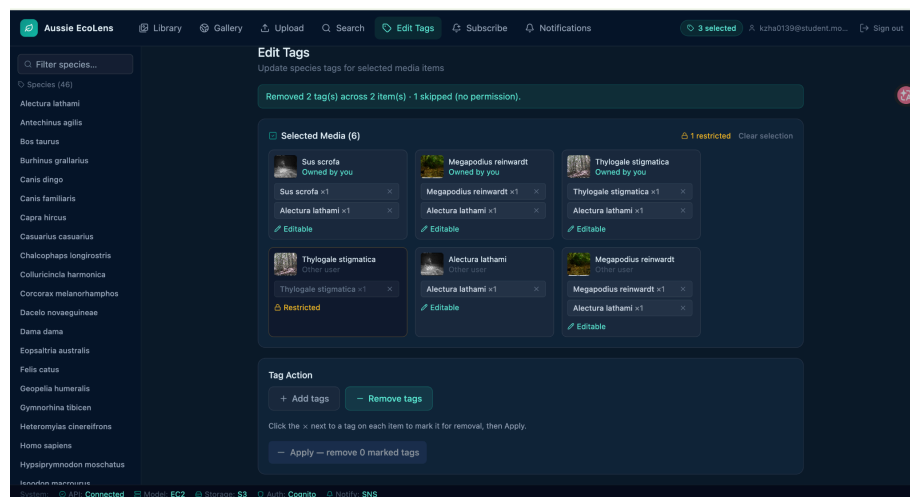


Figure 3.28 Bulk remove tags (table edits on)

Delete Media

Users can multiple **Delete** only their own uploaded media. Other users' public media can be viewed but cannot be deleted.

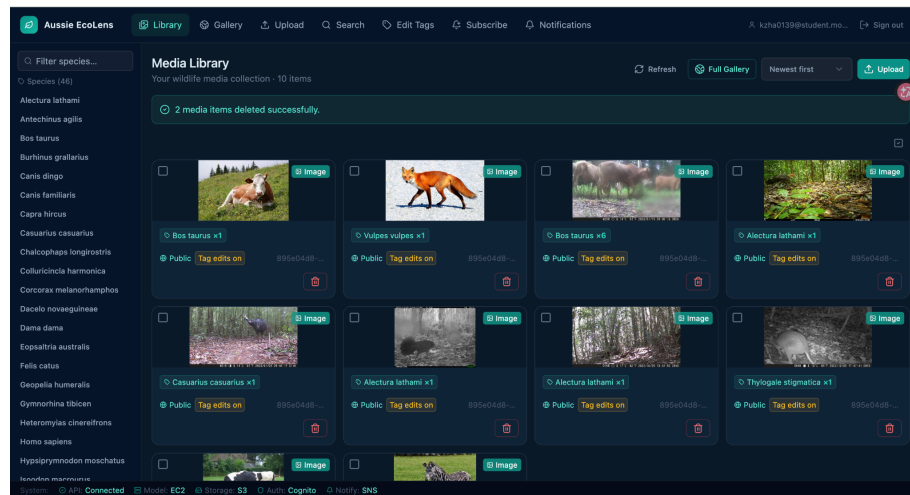


Figure 3.29 Delete own media

Subscribe & Notification

Users can subscribe to wildlife species tags and receive a weekly email digest every Sunday containing links to newly uploaded images that match their subscriptions.

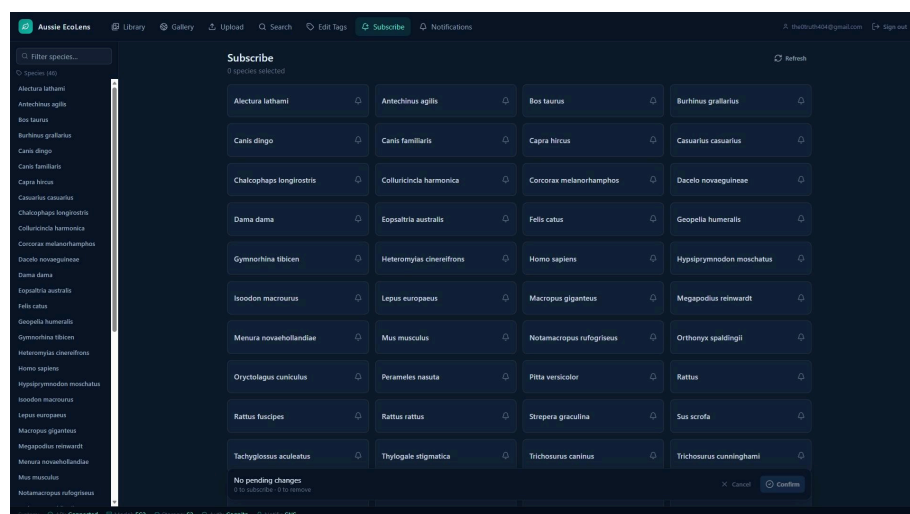


Figure 3.30 Subscription Page

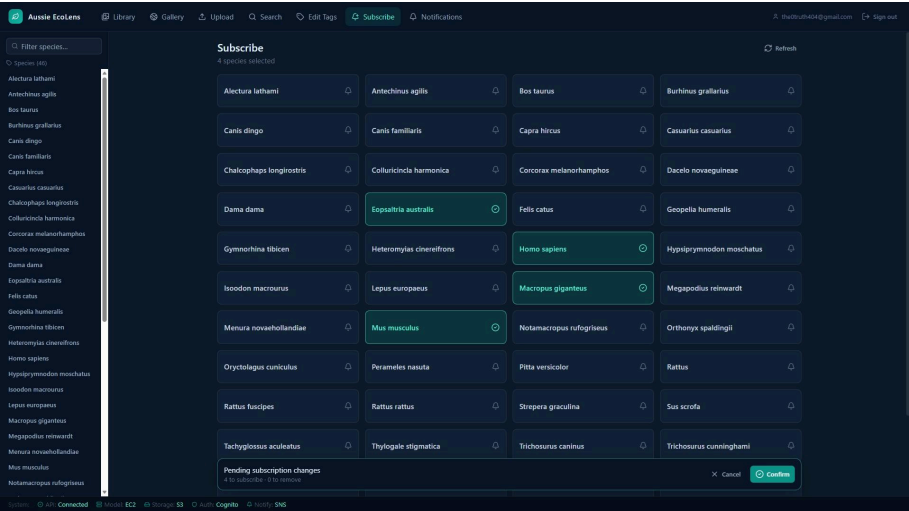


Figure 3.31 Bulk Selection

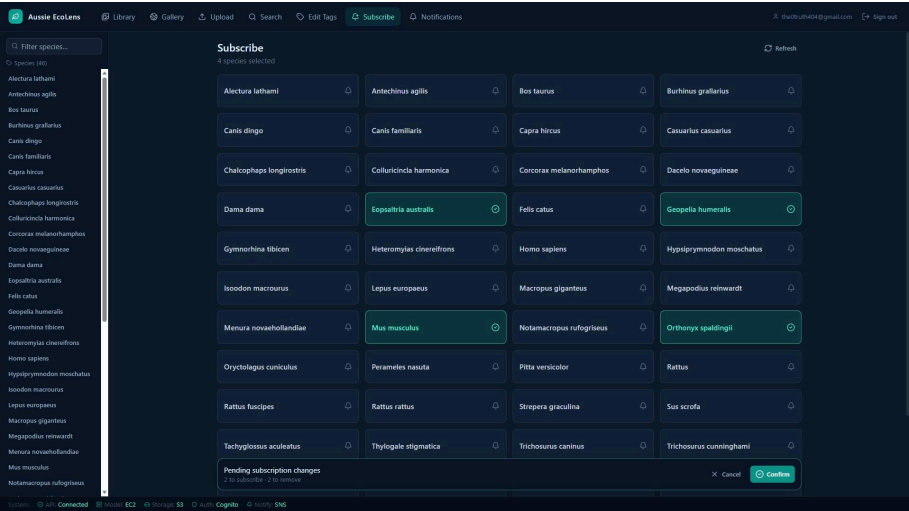


Figure 3.32 Bulk edit(remove/add) Page

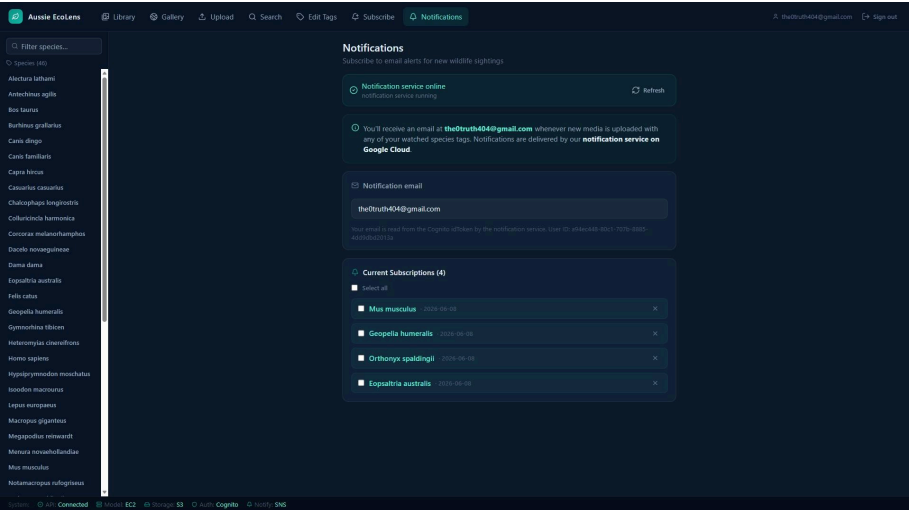


Figure 3.33 My Subscriptions

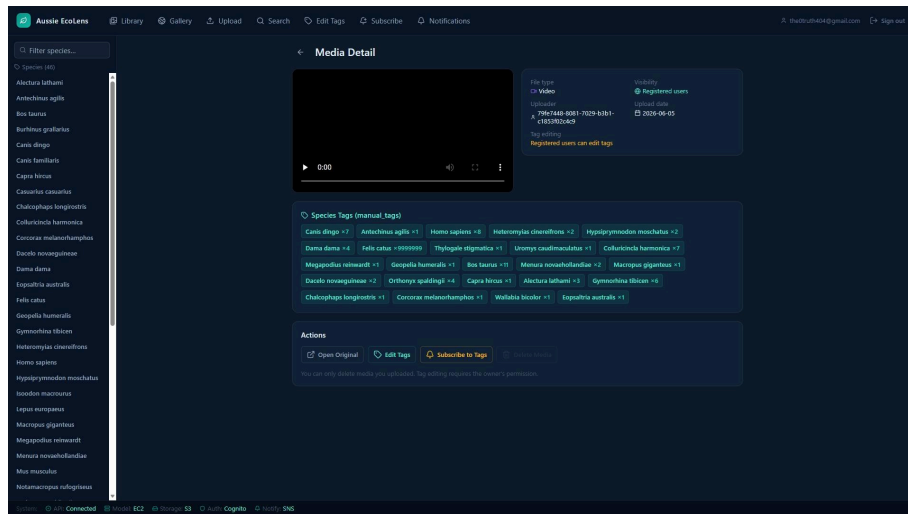


Figure 3.34 Subscribe from image

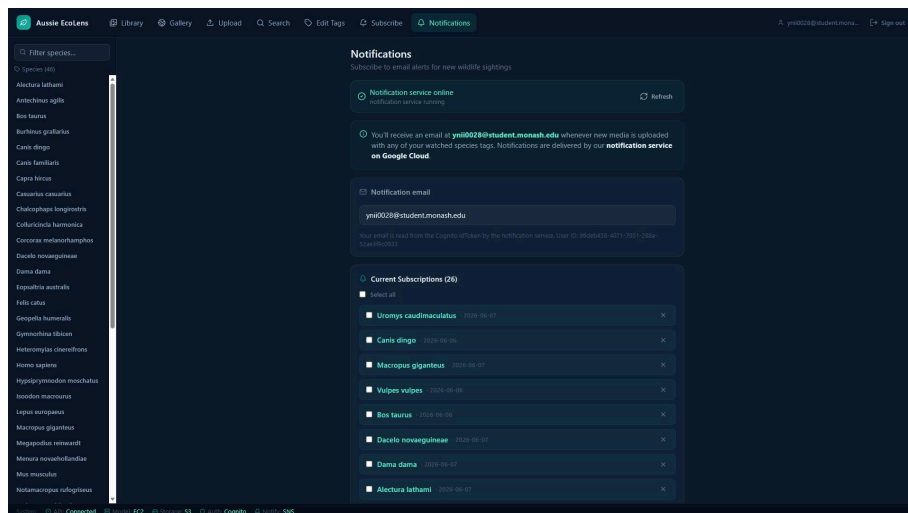


Figure 3.35 All tags from image are added

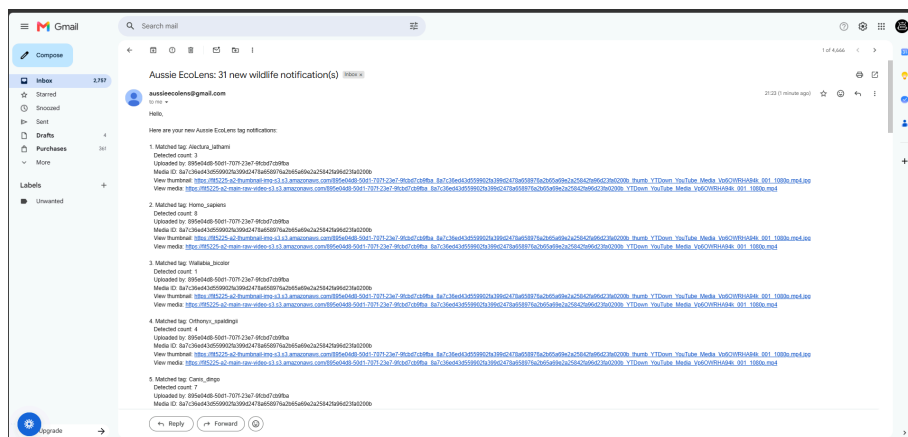


Figure 3.36 Automatic Email Notification