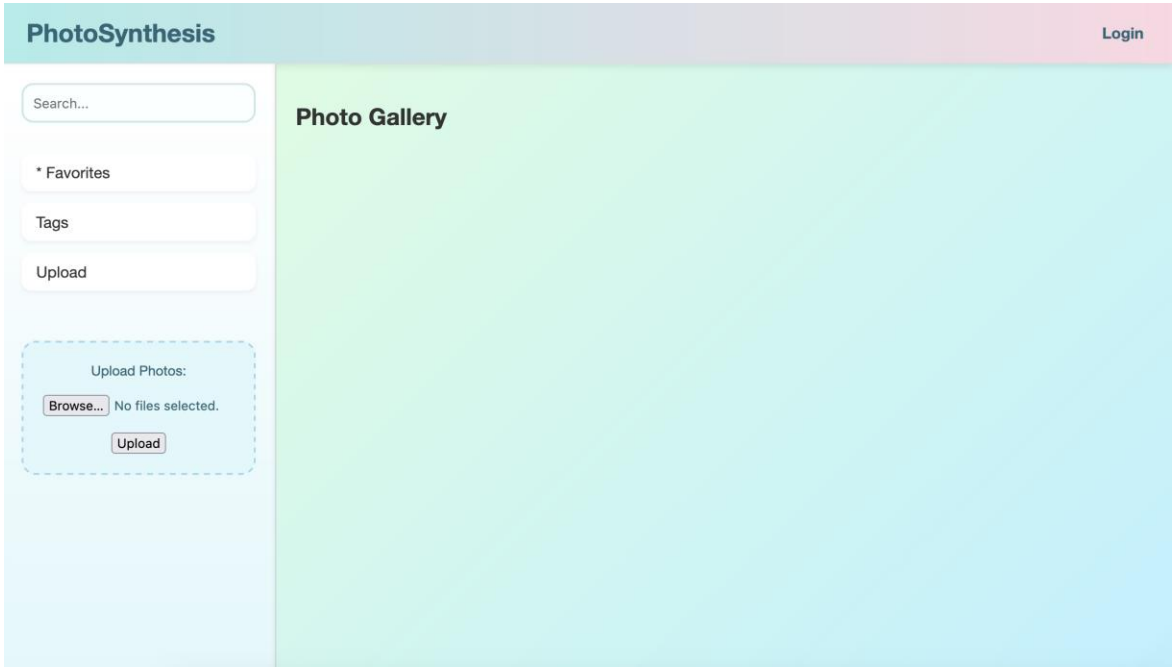
Photo App Project Presentation

Group 1 (Back-end): Zach Stofko and Alex Scalcione

Group 6 (Front-end): Fio Rigney and Catherine Wisniewski

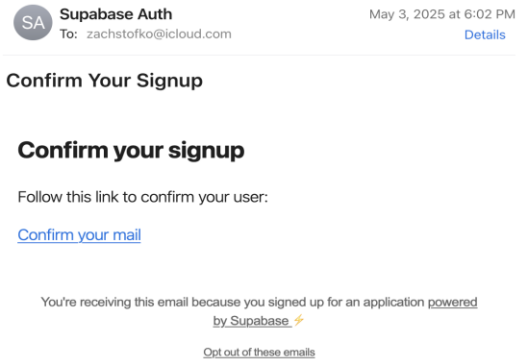
Overview

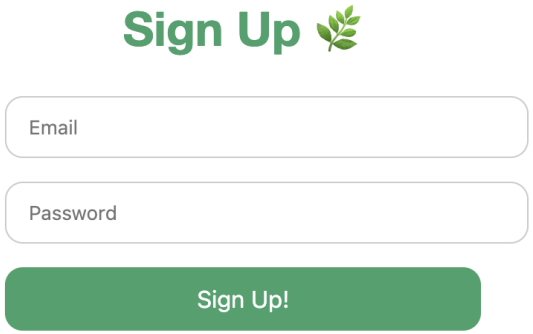
- Built a full-stack photo management website using Django and Supabase.
- Integrated AI tagging via HuggingFace API for smart photo categorization.
- Website allows for photo uploading and storage (imgur).
- Features Search, Favoriting, Organizing, and Removing of photos.



Features

- User Signup and Login (with Email Verification)
- Photo Uploading and Storage
- AI Tagging (Hugging Face)
- Photo Viewing
- Favoriting and Deleting Photos
- Photo Organization (by Tags and Dates Uploaded)





Technologies Used

Database Schema

- Supabase created the auth_user table as part of its built-in authentication. We had no control over it.
- We did not take advantage of all of Supabase's authentication features, like the "is_staff" or "is_active" attributes.

auth_user	
id	int4
password	varchar
last_login	timestamptz
is_superuser	bool
username	varchar
first_name	varchar
last_name	varchar
email	varchar
is_staff	bool
is_active	bool
date_joined	timestamptz

tag	
id	int8
name	text

photo	
id	int8
url	text
tags	_int8
date_created	text
creator	uuid
is_favorited	bool

AI Usage Status

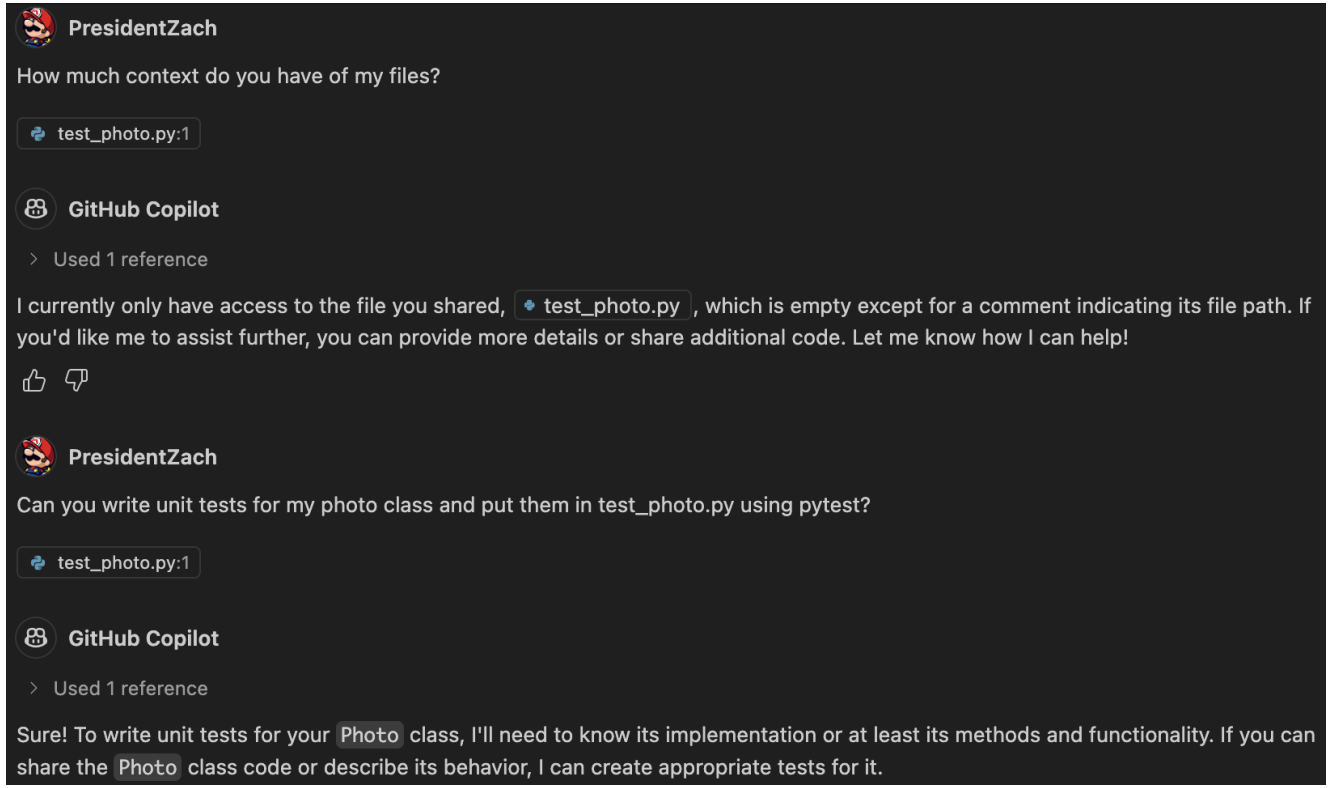
openai/clip-vit-base-patch32

Utilized hugging face API to generate tags

Call API with:

Photo	List of tags	API Key

AI Usage Status – Unit Test Writing



Using GitHub Copilot to Implement Unit Tests

- Integrated unit tests with GitHub Copilot (free tier).
- Performance was smooth on both my Windows 11 desktop (32GB RAM) and M2 MacBook Air (8GB RAM).

Gripes

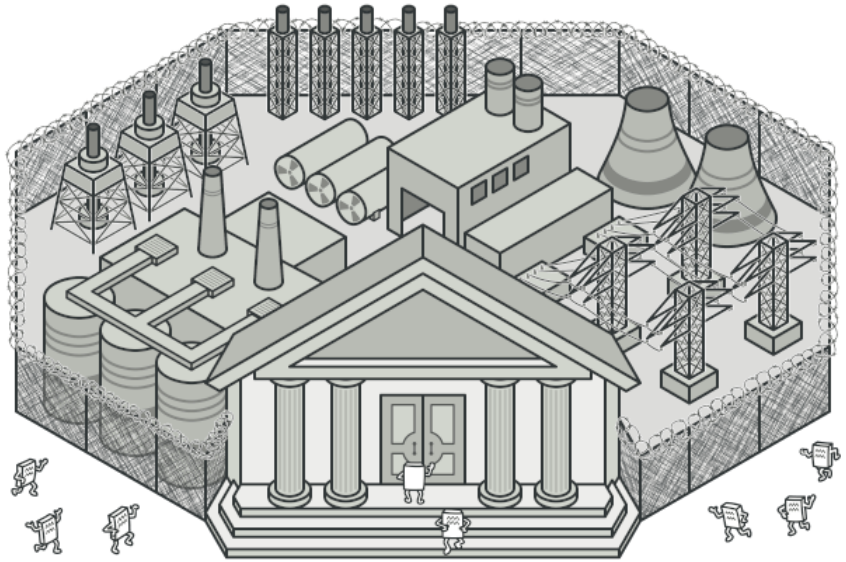
- Copilot defaulted to unittest instead of pytest, despite repeatedly saying to use pytest.
- Limited context – only current + attached files.

Resolutions

- Fixed test errors; photo class tests caught real bugs.
- When errors occurred, Copilot often returned a full, corrected file—making copy-paste fixes convenient.

Design Pattern - Facade

- The app provides a single interface to interact with complex systems like Supabase and the Hugging Face API.
- Users and other parts of the app don’t need to know about the underlying complexity, like uploading photos.
- The app centralizes responsibilities like user session handling and photo management behind a clean, simple interfaces.
- Developers can interact with high-level methods instead of dealing with multiple services or libraries directly



https://refactoring.guru/design-patterns/facade

Most Difficult to Implement

- Zach: Establishing a connection to Supabase: The official Python library lacked clear documentation for setting up the client. Vague error messages made debugging slow and frustrating.
- Alex: Making API call to Imgur. Documentation lacked information on what file type the image should be. Would sometimes fail randomly. Setting up the call in Postman helped with this roadblock.
- Fio: Personally, making the backend and the frontend connect to each other was the most difficult thing. It was confusing to try to write code in three different languages and make everything send information properly!
- Catherine: Getting the search bar too be functional. The search bar was not linking properly with the backend and was unable to actually find the tags.

Member Contribution

Zach	Alex	Fio	Catherine
• Created and managed the GitHub repository. • Instantiated the Django web app. • Implemented the Supabase API. • Created classes and methods for managing photos, tags, and users. • Created backend functionality for login and signup. • Implemented unit tests. • Some UI Functionality.	• Created the GitHub Projects taskboard. • Developed call to Hugging Face API for AI generated tags • Developed call to Imgur API to upload photos • Implemented base login/logout functionality • Made sure all tags/photos were uploaded to the database properly	• Defined tables and columns needed for the database. • Drafted website layout. • Added functionality for displaying images in a grid. • Created buttons that allow the user to remove images or update their data.	• Created the Basic index.html file • Created the .css style sheet for the app • Added functionality to the UI via upload button • Created a Search bar to allow users to filter through their images

Conclusion

- This project showcases a fully functional photo management web application with features like:
  - AI-powered photo tagging
  - User signup/login with email verification
  - Image uploading, favoriting, and organization
- None of the core functionality we aimed to complete is missing, but a feature we would've liked to add is the ability to filter based on selecting from a list of tags.
- Key takeaways:
  - Learned how to integrate modern backend services like Supabase in a Django project.
  - Strengthened our skills in version control, code organization, and collaborative development

Github Repository: https://github.com/PresidentZach/photo_app

Github Projects: https://github.com/users/PresidentZach/projects/2

DEMO TIME!