CS3354.502 Software Engineering
Final Project Deliverable 2

TwitTwat:
A Local Wildlife Photo Sharing Journal

Chloe Tatunay
Samara Sherven
Giovani Jonenson
Zachary Karanja
Akshu Ariga
Diya Jibu
Yoojin Lee
Tee Nguyen

Delegation of tasks:
Chloe Tatunay: Write about a test plan for software, add to conclusion, add references to the reference slide, and add presentation slides about the sequence diagrams as well as the test plans.
Samara Sherven: added to references, conclusion, and presentation slides
Giovani Jonenson: conclusion , slides for presentation regarding responsibilities from phase 1 ,
Zachary Karanja: Write about a test plan for software, add to conclusion, add references to the reference slide, and add presentation slides about the class diagrams as well as the test plans.
Akshu Ariga:Comparison to similar apps , cost estimation , slides for aforementioned things plus phase 1 responsibilities.
Diya Jibu: Software Process Model
Yoojin Lee: Cost Estimation, add to conclusion, add references to the reference slide, and add presentation slides about non-functional requirements and cost estimation.
Tee Nguyen: Write about a test plan for software, add to conclusion, add references to the reference slide, and add presentation slides about the architectural design (MVC pattern) as well as the test plans.

Project Deliverable 1 Content:

Proposed Implementation/Goal:

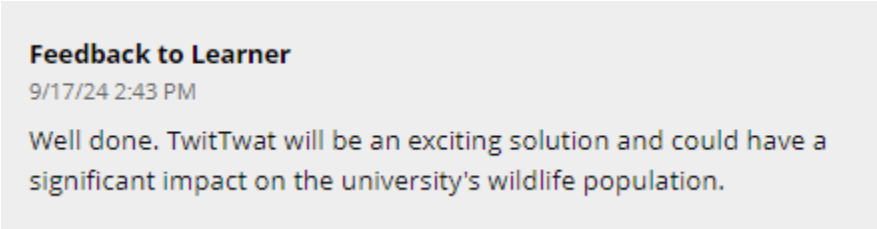
- TwitTwat: Local Wildlife Photosharing Journal
- “A mobile app for wildlife enjoyers to log and share their sightings with other local enthusiasts. (Like UTD birdwatchers)"

Motivations:

- We chose to make this software because we have noticed a lot of wildlife on campus and we thought that it would be interesting for students to have a software that will allow them to share photos and log their wildlife sightings.
- This software will primarily be used on campus as we designed it around it but can be scaled up to a regional/worldwide level.

Task Delegation -
Chloe Tatunay: Created the sequence diagram for the software, github repository, and contributed to project scheduling.
Akshu Ariga - Use case diagram, Read me file
Yoojin Lee - Non-functional requirements
Zachary Karanja - Class Diagram
Diya Jibu - Software Process Model
Giovani Jonenson - Functional Requirements, Project Goals and Motivations, Presentation Slides, Project Scheduling, & Software Process Model Determination.
Samara Sherven - Architectural Design
Tee Nguyen - Architectural Design

Proposal Feedback –
No issues to address



Github Repository: https://github.com/chloetat/3354-TwitTwat.git

Which software process model is employed in the project and why?
The Incremental Process Model seems like the best choice for this project as it combines both linear and parallel process flows. This will allow the application to be developed in small increments. For instance, the first version might include basic features like user login and photo uploading, while later increments would introduce additional functionalities such as photo sharing and location identification. This model supports quicker turnarounds by enabling continuous development and testing. It allows us to deliver a functional version of the application early on, test it within a small group and incorporate their feedback. This flexibility will help us refine the app iteratively as we aim for regional scaling.

Functional requirements:

1. A user shall be able to post a photo containing an animal that can be automatically identified via an AI-assisted scanning API.
2. A user shall be able to share and view their uploaded photos in a news feed-like interface, mixed in with other users as desired.
3. A user shall be able to share via an automatically generated HTTPS link their individual uploaded photos or profile outside the TwitTwat service itself.
4. A user shall have full control over the visibility of their profile to other users and outside the service such as through a search engine.
5. A user shall have the ability to delete their uploaded photos from the service as desired.
6. The service shall be able to reject uploaded photos that do not contain an animal, or allow for an appeal from a user via manual moderation from the development team if the AI-assisted scanning API detects a human face or doesn't detect an animal.

Non-functional requirements:
Usability requirements:

- The app should be easy to download and open by the users.
- The app shall have a simple and intuitive interface for users to interact easily with the app.

Performance requirements:

- The app shall take less than 2000 milliseconds to load images, posts, and screen refreshes.
- The app shall support at least 30 users simultaneously using the app's services.

Space requirements:

- The app itself shall take up less than 50 MB.
- The app should use less than 200 MB in cached data.

Dependability requirements:

- The app shall incorporate error handling and troubleshooting that addresses crashes.

Security requirements:

- The app shall encrypt user data and photos to secure sensitive information.

Environmental requirements:

- The app should properly adjust its interface for different screen types and sizes.

Operational requirements:

- The app shall sync data across devices for users to access content from the same account.

Development requirements:

- The app's code shall be maintainable for updates to be made easily as needed.

Regulatory requirements:

- The app should follow best practices of industry standards of photo-sharing apps and nature related software.
- In the case that the app is developed for a company, the app shall follow corporate policies.

Ethical requirements:

- The app shall prohibit sharing harmful content and content that promotes inappropriate behavior.

Accounting requirements:

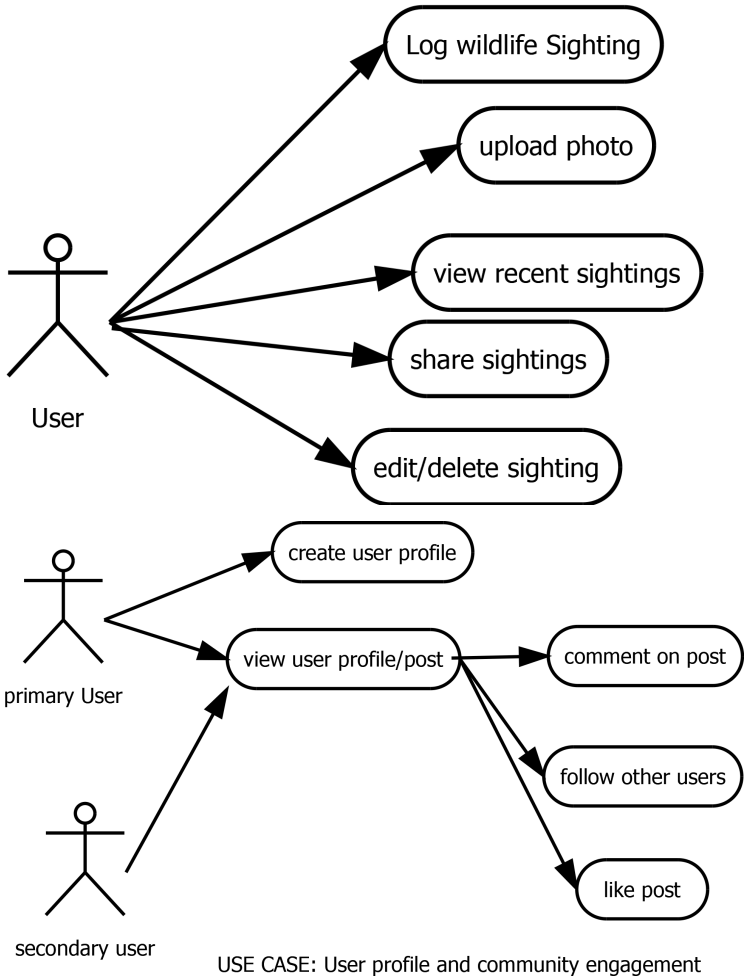
- The app shall comply with legal requirements regarding accounting in the case that the app incorporates finance related services.

Safety /Security requirements:

- The app shall moderate user generated content by allowing users to report and block other users.

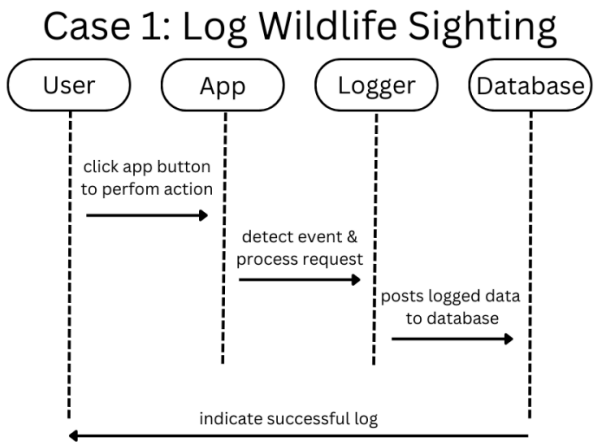
Use case diagram:

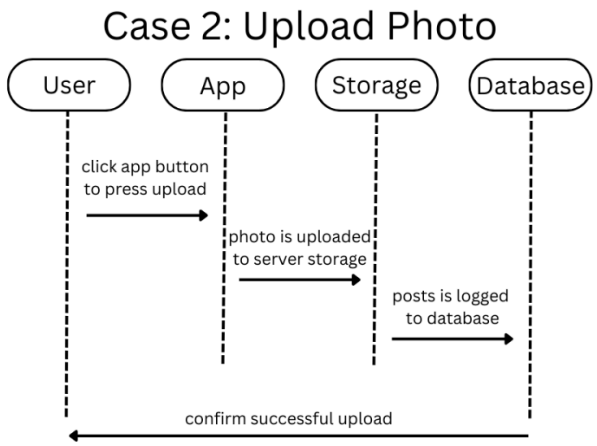
USE CASE : User Photo Log and Sharing

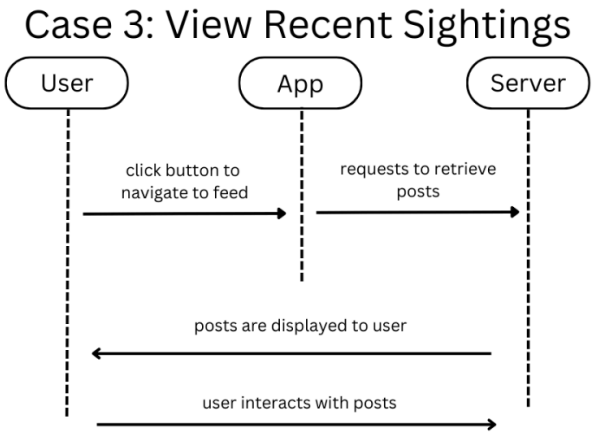


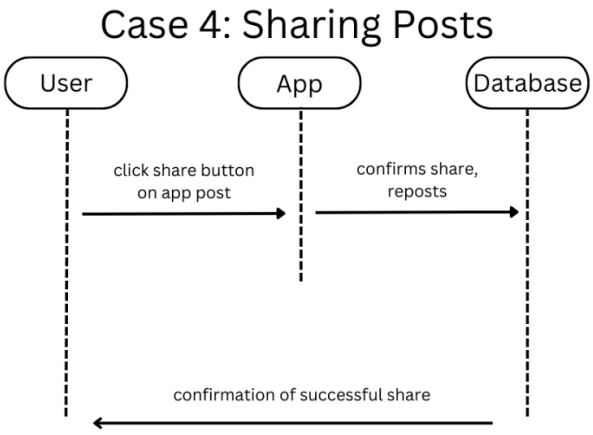
USE CASE: User profile and community engagement

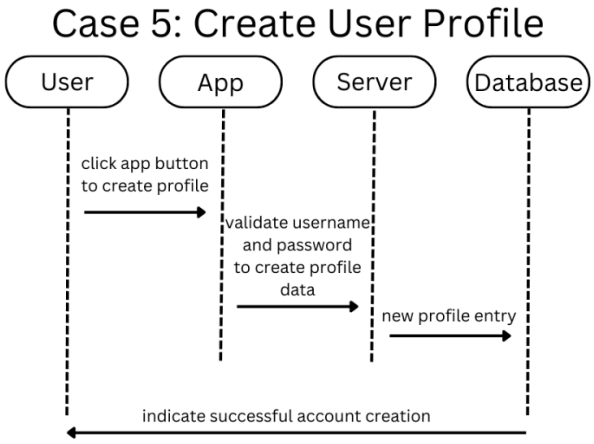
Sequence Diagram:

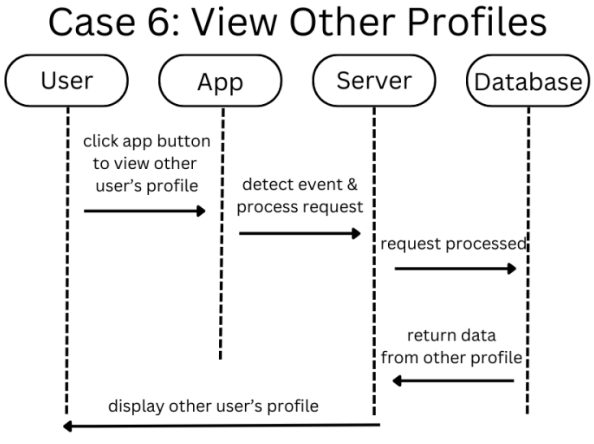




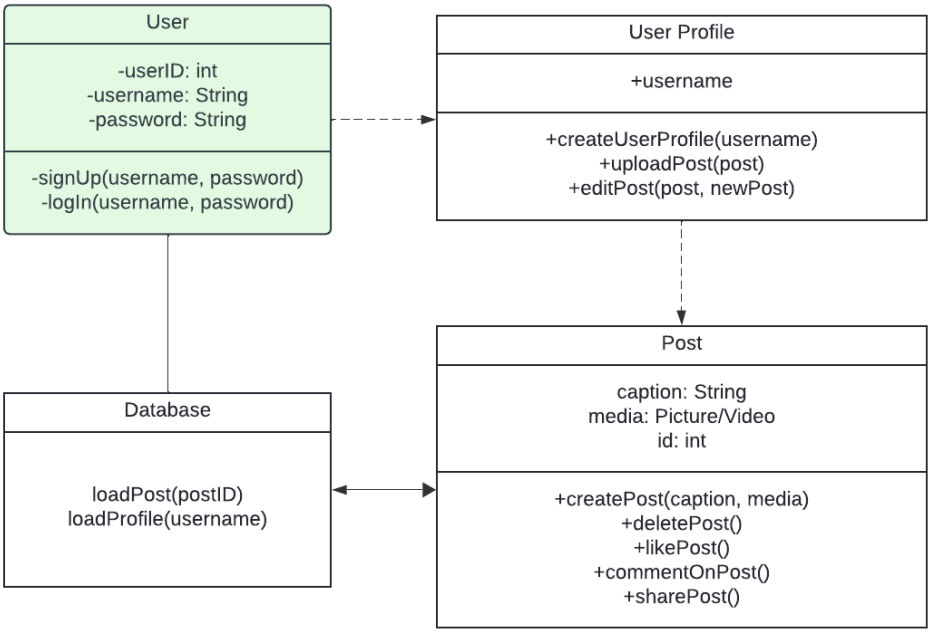




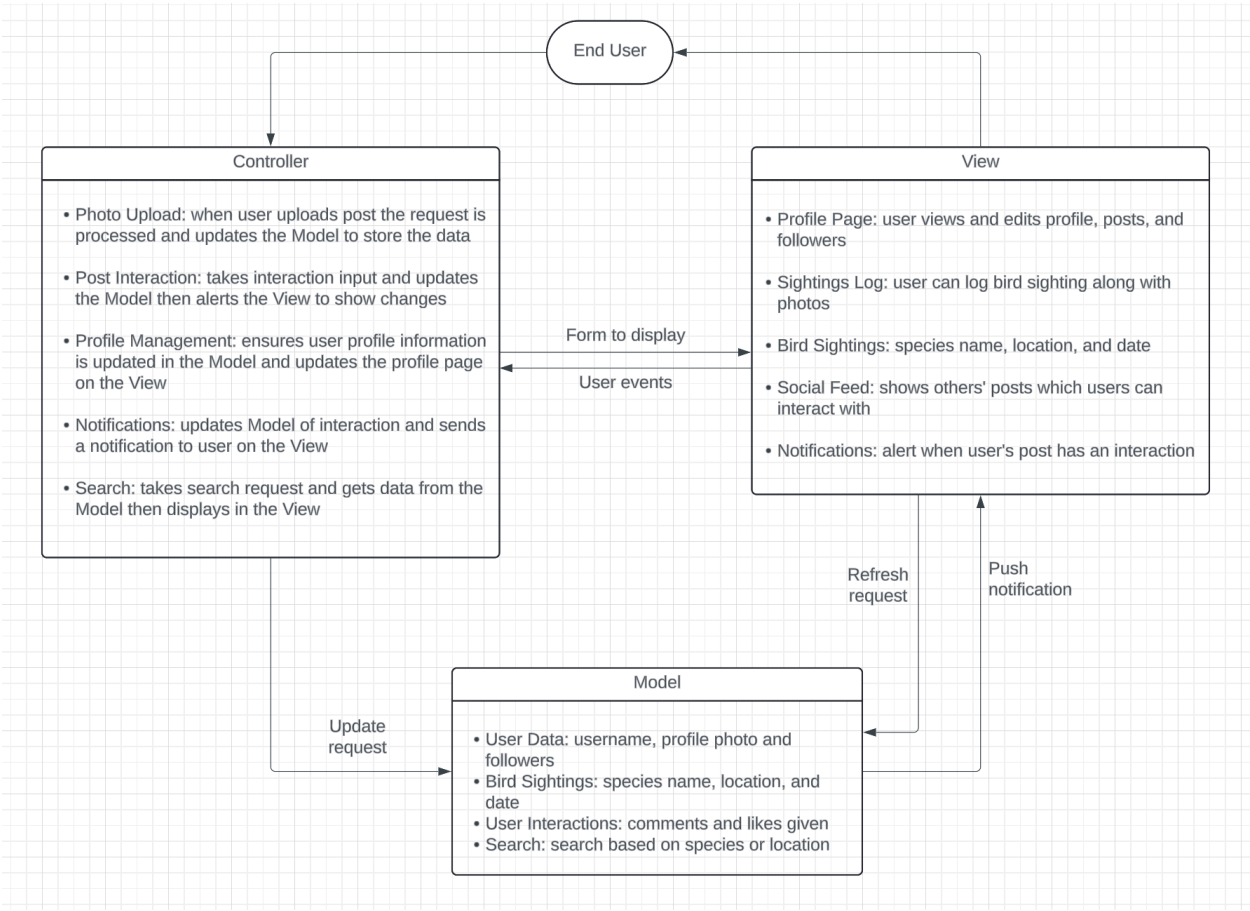




Class diagram



- Model-View-Controller (MVC) pattern



Project scheduling

Weekend Work:

- No weekend work is included. Work will be done Monday to Friday, excluding public holidays and weekends to allow a balanced work-life schedule.

Daily Working Hours:

- 8-10 hours per day is allocated to accommodate project tasks while managing flexibility for intensive or lighter tasks.

  Making the timeline six months allows the team to complete each project phase with greater focus, including additional time for user feedback, feature refinement, and detailed testing. We feel that a 6 month period also allows for the app to come out at the close to the start of summer which will coincide with the period in which the most wildlife can be found on campus during the year.

  ## Project Schedule (December 1, 2024 - May 31, 2025)

  Month 1 (Dec 1 - Dec 31, 2024): UI/UX Design

  - Develop wireframes and visual design for key screens (e.g., user profile, sighting logs, community feed).
  - Gather preliminary user feedback on wireframes to refine the designs.

  Month 2 (Jan 1 - Jan 31, 2025): Backend Development – Part 1

  - Set up database, implement user authentication, and create core API endpoints.
  - Develop backend logic for the sighting log feature, enabling users to log and store wildlife sightings.

  Month 3 (Feb 1 - Feb 28, 2025): Backend Development – Part 2 and Frontend Basics

  - Complete backend features, including photo uploads and data retrieval.
  - Begin frontend development, focusing on navigation and basic interface setup.
  - Integrate backend APIs with frontend for basic interactions (e.g., logging in, uploading sightings).

  Month 4 (Mar 1 - Mar 31, 2025): Frontend Development – Part 2

- Develop remaining frontend features, including community feed, profile views, and search/filtering.
- Add social functionalities like commenting and liking sightings.
- Conduct usability testing to identify and address UX issues.

Month 5 (Apr 1 - Apr 30, 2025): Testing and Debugging

- Perform comprehensive testing, including usability, performance, and security tests.
- Debug and optimize the app based on testing outcomes.

Month 6 (May 1 - May 31, 2025): Final Adjustments and Documentation

- Make final adjustments based on user feedback and test results.
- Prepare user documentation and finalize the app for launch or presentation.

We used application composition to estimate the price. This is a good technique to use since it works well with prototyping. The Application Composition approach focuses on estimating the effort required for each major component based on complexity and then calculating costs accordingly.

ii. Cost Estimation Using Application Composition

Estimated Effort per Component:

Component	Complexity	Effort (Person-Months)
User Profile	Low	1
Sighting Log Feature	Medium	2
Community Feed	High	3
Commenting and Liking	Medium	2
Search and Filtering	Medium	2
Notifications and Alerts	Low	1
Testing and Debugging	High	2
Documentation	Low	1
Total Effort		**14 Person-Months**

- **Total Estimated Effort**: 14 person-months

Cost Per Person-Month

Assuming an average cost of $7,000 per person-month (considering developers' salaries, benefits, and overheads)[1] :

- **Total Personnel Cost** = 14 person-months * $7,000 = **$98,000**

iii. Hardware and Software Costs

1. **Estimated Cost of Hardware**:
    - **Development Machines** (laptops/desktops for each developer)[2]: $1,500 each for 3 developers = **$4,500**
    - **Server** (for hosting the app during development and testing): Estimated at $160 per month[3] * 6 months = **$960**
2. **Total Hardware Cost** = **$5,460**
3. **Estimated Cost of Software**:
    - **Third-Party Libraries or APIs**[4]: Google maps api access: $200 a month * 6 months = **$1,200**
4. **Total Software Cost** = **$1,200**

iv. Personnel and Training Cost

1. **Personnel Cost**:
    - Estimated effort 14 person-months * $7,000 = **$98,000**
2. **Training Cost**:
    - Estimated effort of basic training after installation or launch for maintenance and updates: 1 person-month * $7,000 * 3 developers = **$21,000**

Total Personnel Cost (including training) = $98,000 + $21,000 = **$119,000**

Summary of Costs:

Cost Category	Amount
Personnel	$98,000
Training	$21,000
Hardware	$4,500
Software	$1,200
Total Estimated Cost	**$124,700**

Final Breakdown:

- **Total Project Cost Estimate**: **$124,700**

1. iNaturalist

Overview:
iNaturalist is a social network for people to record and share their observations of different species. The app also uses artificial intelligence to identify species in photos and offers a community of users who help confirm sightings.

Strengths: iNaturalist's identification feature is a big plus and helps users identify unknown species instantly. The community-based validation is ideal for creating a reliable database.
Weaknesses: iNaturalist may be overwhelming for casual users, with a broader scope that includes plants, fungi, and animals worldwide. TwitTwat, on the other hand, focuses on local wildlife, providing a more tailored experience.

2. Merlin Bird ID

Overview:
Developed by the Cornell Lab of Ornithology, Merlin Bird ID is specifically tailored for bird watchers. It offers an extensive database of birds, using AI to help identify species based on location, color, and behavior.

Strengths: Merlin's specialized bird database and sound recognition feature add high value for bird enthusiasts.

Weaknesses: It's exclusively for birds, limiting appeal to users interested in other wildlife. TwitTwat incorporates similar location-based features but for multiple species beyond just birds.

3.Seek by iNaturalist

Overview:
Seek is a simpler, gamified version of iNaturalist. It uses the same species recognition technology but is designed to be more accessible, focusing on encouraging exploration without requiring an account..

Strengths: The gamified approach makes Seek engaging and accessible for beginners. The instant identification feature is convenient.

Weaknesses: Seek lacks community interaction, so users can't share or discuss their sightings. TwitTwat fills this gap by fostering a social experience where users share and discuss sightings locally.
[5][6][7]

Test Plan:

- Our main objectives with TwitTwat is for the app to allow users to take, upload, view, and share photos of the campus wildlife effectively.
- The first case that we decided to make a test plan for was for uploading a photo and caption. The goal behind this case was to allow the user to effectively upload a photo (which is tested if it is a valid file, compatible with the app), and add a caption. After testing, the tester should result in the valid photo file and caption uploading successfully.
    - Code for this unit is TwitTwatUpload.zip on GitHub.
- The second case we decided to make was for creating user profiles and a user database to simulate how TwitTwat would handle storing users. The goal behind this case was to ensure that user profiles could be properly created with a username, password, and id. Additionally, these tests verify that creating a user would throw an error if there was already a user in the database with the same username or id.
    - Code for this unit is on GitHub.
- The third case that we decided to make a test plan for was reposting a photo or a post. The goal behind this case was to check the valid input from the database (userID, photoID,...), print out the notice that shows the user their photo has been reposted (with caption), and notify the user when they repost something. The test would let the user know if there is any invalid information they made (such as a wrong user ID).
    - Code for this unit is TwitTwatRepost.zip on GitHub

Conclusion

TwitTwat is a success in execution and design, it will leave aviary enthusiasts feeling connected to our feathered friends and each other. There were not many deviations from our original plan until now, which helped the team stay on track. Initially we wanted more custom features, but settled on what we knew would be more simple and necessary, cutting down our time requirements and budget. Overall, our project flowed smoothly resulting in our high quality bird sighting sharing mobile app that meets the needs of our target audience.

References

[1] “Entry Level Back End Developer,” *ZipRecruiter*, 2024. https://www.ziprecruiter.com/Salaries/Entry-Level-Back-End-Developer-Salary--in-Texas#Monthly (accessed Nov. 10, 2024).

[2] M. Pratt, “How Much Should a Business Computer Cost?,” *Business.org*, Feb. 15, 2023. https://www.business.org/finance/cost-management/much-computer-cost/ (accessed Nov. 10 2024).

[3] “Dedicated Servers Custom for Windows & Linux | ServerMania,” www.servermania.com. https://www.servermania.com/dedicated-servers-hosting.htm (accessed Nov. 10, 2024).

[4] “Platform Pricing & API Costs,” Google Maps Platform, 2024. https://mapsplatform.google.com/pricing/?_gl=1 (accessed Nov. 10, 2024).

[5] Cornell Lab of Ornithology, “Merlin Bird ID – Free, instant bird ID help for 4,500+ birds,” *Allaboutbirds.org*, 2019. https://merlin.allaboutbirds.org/

[6] Cornell Lab of Ornithology, “Merlin Bird ID – Free, instant bird ID help for 4,500+ birds,” *Allaboutbirds.org*, 2019. https://merlin.allaboutbirds.org/

[7] “Seek by iNaturalist,” *iNaturalist*. https://www.inaturalist.org/pages/seek_app