Application for Manual Annotation of Camera Trap Images

Grace Cheung

CPSC 490 Final Project Report

Advisor: Professor Aaron Dollar

Acknowledgments

Thank you to my advisor, Professor Aaron Dollar, for his guidance and support in creating this project, and thank you to the members of his lab for their guidance as well, especially Diego Ellis Soto and Dave O'Connor. Also, thank you to my senior project partner, Elizabeth Brooks.

Background

Ecological research, especially conservation research, depends heavily on image data gathered from "camera traps." A camera trap is a motion, infrared, or other remote sensing activated camera stationed to capture images of organisms in a certain area over a certain period of time without researcher presence, for the purpose of later analysis. For example, the images gathered from camera traps can be used to determine the biodiversity of an area, or to estimate the population of a certain species, and thus determine the necessity of conservation attention. Camera traps can collect thousands of images, which need to be parsed through and converted into usable, quantitative data. First, many of these images are irrelevant, containing no organisms of interest or are blurry or blank. Relevant images then need to be examined and annotated for features of interest to the research project. Examples of such features are: "number of organisms," "number of spots on the bird shown, "number of petals on the flower shown." Commonly, images are also used for taxonomic purposes as well.

To this day, this image analysis is predominantly done manually, using analog methods. There are many reasons for this - the areas that conservation research is interested in are often in less developed countries, resulting in older or unreliable technology as well as low-paid field researchers who are responsible for supervising these camera traps. A typical workflow for a field researcher would be to manually download the camera trap images and parse through them one by one, removing irrelevant or damaged images. Then, the field researcher will reiterate through the images, recording and annotating the features of interest, often by hand in a field notebook. This analog data is then manually entered into an Excel spreadsheet for further data manipulation.

There are many problems and inefficiencies with this current method. Most significantly is the amount of time and effort (and thus, financial burden) necessary for this workflow. For the typical thousands of images captured in one camera trap session, this data collection process can take hundreds of menial hours. Second, it is very prone to human errors that are difficult to retrace and resolve. There are two transfers of data, from camera trap to analog notebook to Excel spreadsheet. At each phase, there is a high possibility of convoluting the data through handwriting or typo errors. But then, because the only reference to the original image the data was collected from is a file name or image ID, it is often too tedious to recheck. The data is essentially separated into, at minimum, three different places and three kinds of storage, making retrieval, rechecking, or editing and appending to the data very tedious. Third, the final data format in the Excel sheet is very convoluted. Different researchers working on similar projects may use very different data collection formats or calculation processes, making data sharing, an important part of conservation research, time-consuming and difficult.

There have been a few technologies that aim to address these or similar problems in ecological research. The two that I will discuss are Microsoft's *AI for Earth* and iNaturalist's volunteer-based *Zooniverse*. The two also have a joint venture attempting to combine machine learning and volunteer annotations. They are two different approaches to image data parsing for ecological research that encompasses most current technology-based approaches. Microsoft's *AI for Earth* is an umbrella project that houses various machine learning based approaches to ecological image data. The project within *AI for Earth* that is most similar to the problems we are addressing is the project, "Conservation Metrics." "Conservation Metrics" uses deep learning to detect and identify species from camera trap images, and then uses Bayesian statistical models to estimate classification errors as well as determine reproducible metrics of ecological trends. *AI for Earth* also offers relevant APIs for camera trap image processing (particularly filtering out unusable images and identifying the presence and location of animals in images) and species identification for both plants and animals. These are external APIs that can support field researchers in processing image data. *AI for Earth* addresses the problems of filtering images and taxonomic processing of images, as well as providing helpful statistical analysis. However, *AI for Earth* does not address field researchers' need for specific features and annotations. There are usage barriers to the APIs, because they require existing technological infrastructure and knowledge to integrate into a field researcher's workflow. Using "Conservation Metrics" means exporting images to Microsoft, and waiting an unspecified period of time for the images to be processed and for data to be returned, in a format that may not necessarily be conducive to the researchers' existing data infrastructure. In conclusion, *AI for Earth* is valuable as an external tool for specific utilizations, but cannot streamline the entire process.

On the other hand, iNaturalist's *Zooniverse* crowdsources volunteer labor to annotate ecological images. Researchers (or anyone) can upload images with the features and annotations of interest, and a network of up to 1.6 million volunteers can answer these questions. This approach does better fit the typical field researcher's workflow and produces data that is more relevant to specific research projects. However, similar to *AI for Earth*, it requires external exporting of images and waiting for data to be returned. In a way, it appears to just be the external outsourcing to volunteers of the typical field researcher's job. While this lessens the financial burden of such tedious image processing for research labs, it introduces a greater margin for human error, as volunteers are not as trained or familiar with the needs of the research project. In our user research (as conducted by my senior project partner, Elizabeth Brooks), Nelson Rios, Yale Peabody Museum Head of Biodiversity Informatics and Data Science, also mentioned that Zooniverse's data output was an unfamiliar format and difficult to integrate into research projects. In general, Zooniverse suffers from a lack of researcher ownership, primarily customizability and convenient reusability for data addendums within a project.

Thus, after review of the problem and the current solutions, there is a clear need for a technology that is more tailored to specifically ecological researchers. Such a technology would be focused on the head researcher's need to customize needed questions and annotations and the field researcher's need for easy, in-house data collection, storage, and annotation. Ideally, this technology would also provide easy, customizable data download and manipulation, as well as some integration of machine learning to speed up the most tedious parts of image parsing.

Deliverables

I worked with Elizabeth Brooks, another senior in Computer Science to complete this project. We aimed to create a prototype of a web-application (with the option to be packaged as a desktop application) that would address the issues outlined above.

Features/General Workflow:

- User creation and authentication (register and login)

- Project creation (title, description), view list of all user's projects
- Admin researchers and field researchers for each project - the admin researcher(s) can specify what annotations and questions the project will be looking for, the field researchers cannot. Both types of researchers can upload images, annotate images, and download data
- Upload multiple images into a project
- Annotate images by answering specific annotations/questions provided by the admin researcher(s), using keyboard-based annotation (more user-friendly and efficient than the typical click-based annotations, also more easily transferable to a future iOS or Android version)
- Download data and basic statistical overview for a project

Our responsibilities are delineated as follows: Elizabeth is responsible for the user-facing portions of the project, specifically the frontend (https://github.com/gracec10/Seniorproject.github.io) and user-research. This included interviewing various ecological researchers to best determine their needs for such an application. My responsibilities were primarily the backend, specifically server-side logic, deployment, data storage/API creation, and database planning, creation, integration, and management (https://github.com/gracec10/seniorprojectbackend). Elizabeth and I collaborated to connect the frontend and the backend. For the remainder of this final report, I will be focusing on my responsibilities and contributions.

Technologies Used:

Frontend: ReactJS, Babel, Webpack, HTML/CSS, Redux

Backend:

- Node.js: JavaScript runtime environment to run JavaScript outside a browser
- ExpressJS: a web application framework for Node.js
- MongoDB: a No-SQL database program
    - Mongoose: a schema-based solution to model MongoDB objects
- JSON Web Tokens and passport.js: for authentication
    - Bcrypt.js: to hash passwords
- Body-parser: Node.js body parsing middleware
- CORS: Cross Origin Resource Sharing enabling middleware
- Nodemon: monitor script for development of a node.js app

The database was hosted using the free (data limited) version of MongoDB's Atlas, a cloud database service.

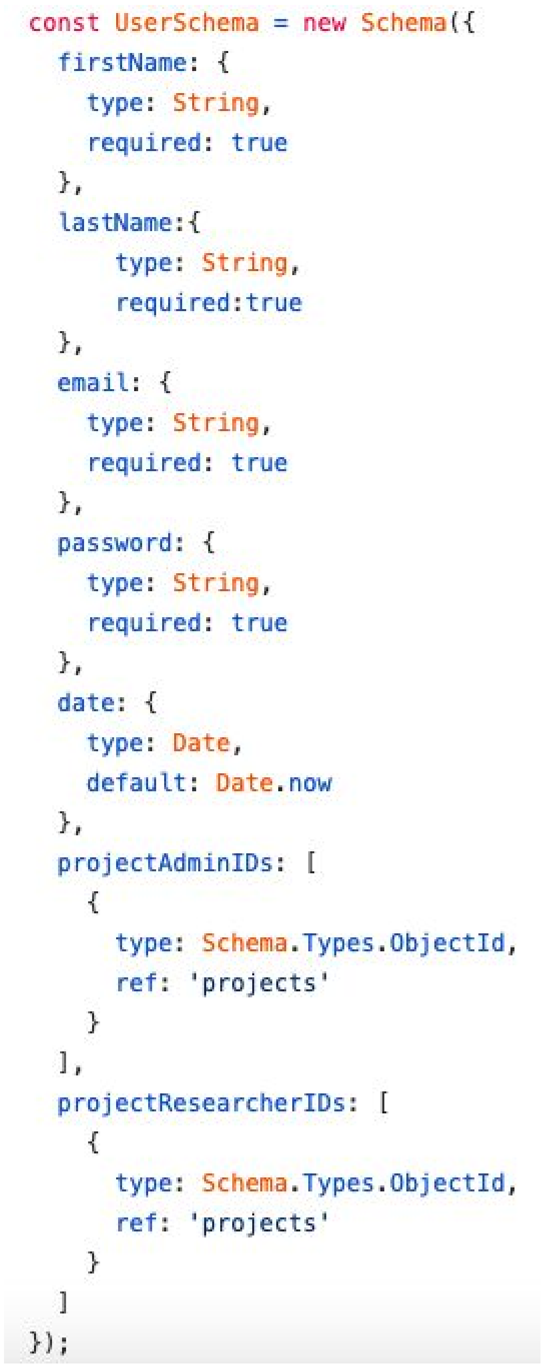
Screenshots and more specific details about the frontend can be found in Elizabeth Brooks' report.

Database Structure/Challenges:

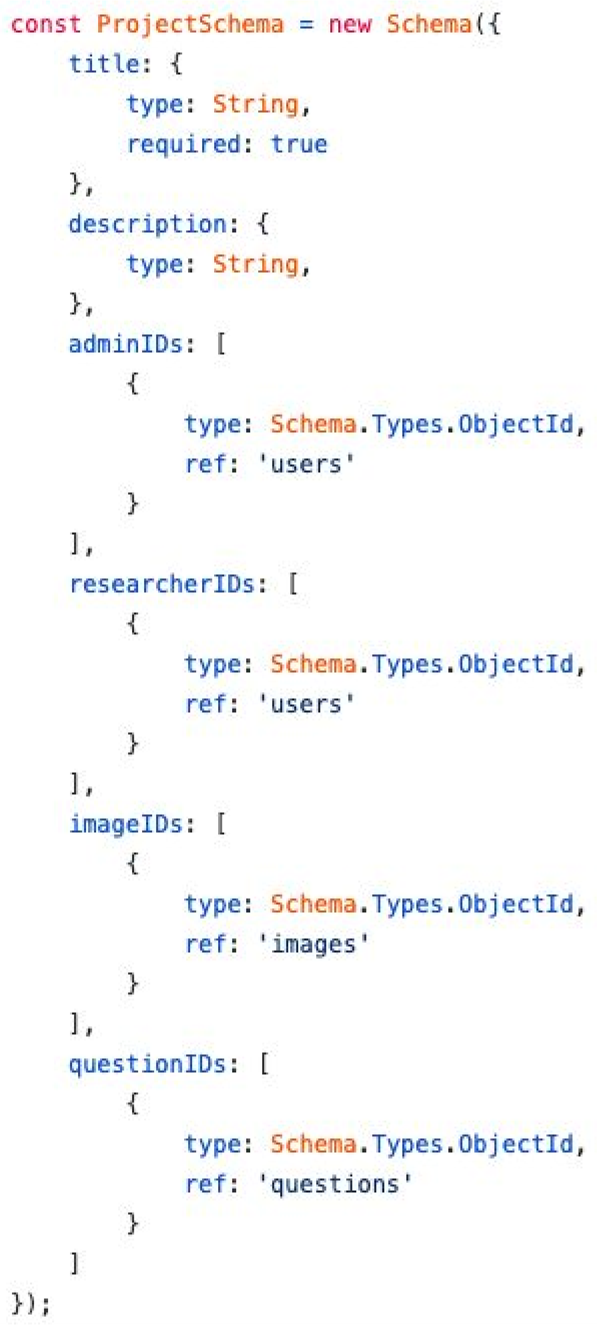
I chose to use a No-SQL, non-relational database, primarily because of the free cloud storage features of MongoDB. In addition, non-relational databases are much more scalable. As this is just the beginning stages of the full application, flexibility in database structure is important. However, this created many unique challenges to fitting what seems most comfortably a relational set of data (User → Project → Image → Question → Answer) into a low-latency, low-space non-relational set of schemas within JSON files (using mongoose and MongoDB).

An overview of my solution:

I used five models: User, Project, Image, Question, Answer



```
const UserSchema = new Schema({
  firstName: {
    type: String,
    required: true
  },
  lastName:{
      type: String,
      required:true
  },
  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  date: {
    type: Date,
    default: Date.now
  },
  projectAdminIDs: [
    {
      type: Schema.Types.ObjectId,
      ref: 'projects'
    }
  ],
  projectResearcherIDs: [
    {
      type: Schema.Types.ObjectId,
      ref: 'projects'
    }
  ]
});
```



```
const ProjectSchema = new Schema({
    title: {
        type: String,
        required: true
    },
    description: {
        type: String,
    },
    adminIDs: [
        {
            type: Schema.Types.ObjectId,
            ref: 'users'
        }
    ],
    researcherIDs: [
        {
            type: Schema.Types.ObjectId,
            ref: 'users'
        }
    ],
    imageIDs: [
        {
            type: Schema.Types.ObjectId,
            ref: 'images'
        }
    ],
    questionIDs: [
        {
            type: Schema.Types.ObjectId,
            ref: 'questions'
        }
    ]
});
```

These are the User and Project models, as examples of my solution. The relationship between User and Project is a many-to-many relationship, which is the most complicated relationship to fit into a non-relational database. A one-to-many or one-to-one relationship can be easily represented by using nested arrays within a singular model. However, if I used nested arrays in this case, it would take up a lot of space to store a different copy of the project for each user that had access to it, especially if there were inactive users and a growing amount of field researchers with access. To prevent that now, I decided to use very small, separate models for each component of the data, and then have each model reference IDs of the relevant pieces of data they are related to. This does bring up the question of a growing number of API calls from the frontend, but Elizabeth and I worked to mitigate that by using Redux. The store feature of Redux allows an external "global" state for the entire frontend, minimizing the number of API calls and state passing.

The routes can be found in routes/api folder and in server.js. I used body-parser to submit data as well as to use req.params to track relations. For example, to access all questions specific to a user's project, the API route would look like: api/:projectID/, where :projectID is the param in the route that specifies projectID. So, this call would go to the project model with that specific ID, and then pull up Project.questionIDs, then call each question model with those IDs and display those contents.

Authorization was addressed using passport.js, bcrypt.js, and JWT. I decided to host most of the authorization logic (in the validation folder) within the backend, for higher security.

Conclusion and Future Developments

As the scope of this web application is very large and the possibilities for it are ever-expanding, there are many future developments that I can see for it, and that I hope to implement.

1) Full integration of Microsoft's *AI for Earth* APIs: this would help filter out empty or unusable images, saving a lot of time. In addition, it could provide semi-automatic taxonomy, a useful tag for all images.
2) Machine learning based direct image annotation: a feature where researchers can click on segments of the image (as determined by machine learning) and annotate those specific segments. A much more advanced version of this would be for the admin researcher to provide a 3D model or many 2D images of what they are looking for in the images, and then have those be automatically counted within the image.
3) Computer vision: direct integration with the camera trap, so that images are captured and processed on the spot

+ Many other small user functionality features such as email validation for authentication, more customizability of projects, packaging into an offline desktop/iOS/Android app for wifi-scarce areas, etc.

In conclusion, I learned a lot about the challenges of combining technology with different fields of scientific research. This kind of problem is not one that is specific to the field of ecology - it is often difficult to integrate technological solutions because of the steady reliability of analog and manual work - there are no bugs or accidental data deletion if everything is kept in a physical notebook, the difficulty of migrating data from the established form of collection to a new technological form, and finally, the unique and specific needs of researchers in specific scientific fields.

I also learned a lot about structuring and managing databases all while focusing on lowering costs, latency, and storage needs. I really had to plan carefully and think ahead to the future needs of the project, while allowing space for future changes and need for adaptability.

Links/Sources:

Microsoft's *AI for Earth*: https://www.microsoft.com/en-us/ai/ai-for-earth-tech-resources

iNaturalist's Zooniverse: https://www.zooniverse.org/

Github (Elizabeth's frontend): https://github.com/gracec10/Seniorproject.github.io

Github (Grace's backend): https://github.com/gracec10/seniorprojectbackend