Team 066

Team Members:

Zhanwang zhou, Yuhang Li, Jimin Yang, Haoran Li

Demo Video Link:

https://youtu.be/C47DwezkJSQ

- **Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

  We followed the same directions from the original proposal and implemented the Bird Spotter application. We still allow users to record and explore bird sightings. Users can upload bird photos, enter details such as time and location, and save their observations. There is also an interactive map where users can view all recorded bird sightings.

- **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

  Our application achieved its main goal of allowing users to record and explore bird sightings through photo uploads, species identification assistance, and an interactive map. However, we did not complete extra features for filtering sightings by species and time, or for sorting events by time.

- **Discuss if you change the schema or source of the data for your application**

  We still used the ebird database for the Bird and the Event table. For the User table, we used a generated dataset.

- **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

  We add columns Country and State to the Event table instead of having only longitude and latitude in our ER diagram. This is a more suitable design because compared to merely the longitude and latitude, Country and State allow the users to more easily and intuitively locate the event.

  We also replace the schema Illustration in our ER diagram with schema Comment. We delete the Illustration schema because its function somewhat overlaps the function of

Image schema, while Comment schema enables users to interact with each other and offers better user experience.

- **Discuss what functionalities you added or removed. Why?**

  We added the functionality to show the latest spotted bird events at the home page. This allows the users to gain an overview of the latest happening events and boost their participation. We also add the functionality to display a random selection of species at our bird searching page, encouraging users to explore and discover birds and spark their interests.

  We deleted the functionality of filtering sightings by species, because we did not find the dataset that systematically specifies the category of birds according to Linnaean classification.

- **Explain how you think your advanced database programs complement your application.**

  The stored procedure enhances our bird-search feature by letting users enter either a scientific name or a common name without needing to specify which. The conditional logic in the procedure automatically detects the type of name provided and returns matching results accordingly.

  we implemented a trigger on the *`Event`* table to validate the latitude and longitude of newly uploaded events. The trigger checks that the coordinates fall within a reasonable range before allowing the insertion to proceed. This ensures that invalid or out-of-bounds data does not enter the database, maintaining the overall integrity and reliability of the stored information.

  We also implemented transactions to ensure data consistency during event uploads. All database operations in the *`/api/upload`* endpoint, including inserting the event, uploading the image, and updating bird information, are grouped into a single transaction. If any step fails, the transaction is rolled back, preventing partial or inconsistent data from being saved.

  We also implemented constraints by implementing primary keys and foreign keys. For the bird table, we are using birds' scientific name as primary key to keep each bird specie unique. For the event table, we are using event id as primary key, user id and bird scientific name as foreign key to keep each event unique and connected to the user and bird table. For the user table, we are using user id as primary key to keep each user unique. For the image table, we are using image id as primary key, bird scientific name as foreign key to keep each image unique and connected to the bird table.

- **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team**

could use this as helpful advice if they were to start a similar project or where to maintain your project.

Haoran Li: One technical challenge we encountered was ensuring consistent alignment of frontend and backend ports early in development. Initially, different developers set arbitrary ports locally, causing connection failures during integration tests. This delayed our debugging significantly. To mitigate this, we established a standardized configuration file specifying unified port numbers for both frontend and backend. Future teams should proactively agree upon and document port configurations clearly, ensuring all development environments consistently reference these ports to streamline integration and avoid connectivity issues during joint debugging sessions.

Jimin Yang: One technical challenge I encountered was integrating Leaflet.js with our bird sighting data on the home page. The main difficulty was managing the complex interaction between React's component lifecycle and Leaflets imperative map API. I solved this by implementing a robust reference system using useRef hooks to maintain the map instance and markers separately from React's state. This approach prevented memory leaks and ensured smooth updates when new sightings were added or selected. The solution also included custom marker management with dynamic popups displaying bird information and images, and efficient cleanup of map resources when components unmounted. This experience highlighted the importance of careful state management when integrating third-party mapping libraries with React's declarative paradigm.

Yuhang Li: Another challenge was correctly setting the database transaction isolation level within a Python function. We found that in a database session, the isolation level must be set *before* any other SQL operations are executed, even if previous cursors have been closed. In our `/api/upload` endpoint, we needed to execute `SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ` immediately after obtaining the connection, and before calling any helper functions like `init_trigger(connection)`.

Zhanwang Zhou: Another challenge I encountered is the initialization of stored procedures. When I implemented the procedure initialization function in Python, it raises various errors. This is because the grammar of SQL queries called by cursor in Python has a few differences from the queries in the MySQL database. For instance, we should avoid using DELIMITERS. Also, when checking if the procedure to be created already exists, we should write the checking and deletion query in the same query string as the stored procedure creation query, instead of running them separately using different cursors.

- **Are there other things that changed comparing the final application with the original proposal?**

  All major changes have been listed above.

- **Describe future work that you think, other than the interface, that the application can improve on.**

  For future work, we could add filters and sorting features on the event search functionality. We can also add an AI agent to determine the species of the uploaded birds.

- **Describe the final division of labor and how well you managed teamwork.**

  We have two team members working on the backend and the other two working on the frontend. One frontend worker is connected to one backend worker on the same functionalities. We also set a schedule on the process of the work and set up group meetings every week. In this way, we achieved maximum efficiency and were able to finish all major functionalities of our application following the schedule on time.