BeakPeek

Millennium
University Of Pretoria

Algorithmic Innovation Award

In Partnership With
AgileBridge

Group:
22

Mentor:
Ayaaz

2024/10/18

Contents

1 Explanation of Algorithms in BirdMap

The BirdMap feature utilizes several key algorithms that enhance both performance and user experience. These algorithms ensure that the map is interactive, efficient, and responsive, making the app user-friendly and scalable.

1.1 Geolocation and Dynamic Camera Adjustment

The `_getCurrentLocation` function leverages the *Geolocator* package to obtain the user's real-time location. The algorithm first checks if location services are enabled and if the necessary permissions are granted. Once a valid location within the boundaries of South Africa is obtained, the camera on the map is dynamically adjusted to center on the user's location.

Why it's effective:

- **Efficiency:** By updating the map's camera position based on the user's current location, the feature provides real-time relevance without unnecessary user input.
- **User-Centric Design:** Automatically centering the map improves navigation and location-based features, enhancing the overall user experience.

1.2 Dynamic KML Parsing for Polygon Data

The `_loadKmlData` function loads KML data corresponding to the selected province. This data is parsed into polygon shapes using a custom *KmlParser*, and the resulting polygons are then drawn on the map.

Why it's effective:

- **Scalability:** The algorithm dynamically loads KML data for specific provinces, avoiding the need to load unnecessary data, which improves memory usage and performance.
- **Modularity:** By separating the parsing logic from the map itself, the design remains flexible and allows for the reuse of this functionality in other regions or datasets.

1.3 Filtering by Province and Month

Users can filter the map's content based on province and month. When a new province is selected, the camera is adjusted to focus on that province, and the corresponding KML data is reloaded. The month filter allows for further refinement of bird data shown on the map.

Why it's effective:

- **Interactivity:** The filter system allows users to explore different regions and times with ease, keeping the interface intuitive and engaging.
- **Performance:** The dynamic reloading of only the necessary map data (polygons) helps in reducing the computational load and enhances the speed of the application.

1.4 Interactive Polygon Tapping

The `_onPolygonTapped` function is triggered when users tap on any polygon on the map. This action opens a modal sheet with detailed bird data for the selected region.

Why it's effective:

- **User Engagement:** The ability to interact with specific map regions encourages users to explore the data, creating a more immersive experience.
- **Clean User Interface:** By using modal sheets, the app avoids cluttering the map with too much information, providing details on demand.

1.5 Overall Design

These algorithms work together to optimize both the performance and usability of the BirdMap feature. By loading data dynamically and offering interactive filters, the system is able to present relevant information to users while minimizing unnecessary resource usage. The modular and scalable design allows for easy future expansion and adaptability to other datasets or regions.

2 Explanation of Algorithms in the HeatMap

The HeatMap feature incorporates several effective algorithms to ensure efficient display and interactivity for bird population data, with dynamic filtering by month and species. Here's an overview of the key components and why they work so well.

2.1 Dynamic Pentad Data Loading

The `loadPentadData` function dynamically loads bird sighting data. It retrieves population information for different species, computes polygons for regions (called pentads), and colors these polygons based on the reporting rate, i.e., how often birds are sighted in a particular region.

Why it's effective:

- **Dynamic Data Handling:** The map only loads the data it needs when required, keeping the app fast and responsive. It processes and adds polygons in batches, ensuring that the user experience remains smooth even with large datasets.
- **Modularity:** The function that retrieves bird data is separate from the map rendering logic. This modular design makes the system more flexible and reusable in other parts of the app.

2.2 Color Palette and Visualization

The `getColorForReportingRate` function assigns colors to the polygons based on the percentage of bird sightings (called the reporting rate). The color scale ranges from low to high, helping users quickly identify areas with high bird activity.

Why it's effective:

- **Visual Clarity:** The color-coded map makes it easy to understand bird distribution patterns at a glance. It turns raw data into meaningful visual insights.
- **User Engagement:** Users can customize the color palette through the `PaletteSelector`, allowing them to personalize their experience and making the app more interactive.

2.3 Interactive Map Filtering

The map allows users to filter the displayed data by month. By selecting a month, users can see how bird populations change throughout the year. This is managed via a dropdown menu that updates the polygons and the displayed data accordingly.

Why it's effective:

- **Real-time Updates:** The polygons are cleared and redrawn every time the user changes the month, ensuring the map always reflects the latest data.
- **User Flexibility:** Users can easily explore different time periods, providing a more engaging and informative experience.

2.4 Efficient Map Rendering

The Google Map widget efficiently renders polygons on the map. The polygons represent pentads and are generated dynamically based on bird sighting data. To maintain performance, the polygons are processed and added to the map in small batches.

Why it's effective:

- **Efficiency:** By only rendering relevant polygons and doing so in small increments, the map stays responsive and avoids performance bottlenecks.
- **Interactivity:** The map allows for zooming and panning, ensuring users can explore the data interactively without any lag or slowdowns.

2.5 Endangered Species Alerts

The app includes a dynamic alert that informs users if the bird species they are viewing is endangered. The app checks the bird's population size and displays an appropriate message, enhancing the user's awareness of conservation efforts.

Why it's effective:

- **Dynamic Warnings:** Users are alerted about endangered species, adding a layer of education to the app and increasing awareness of conservation issues.
- **Seamless Integration:** These warnings are displayed alongside the map in a way that feels natural, without interrupting the user's experience.

2.6 Overall Design

These algorithms ensure that the HeatMap feature is not only efficient but also interactive and user-friendly. By dynamically loading data, using intuitive visual cues, and providing interactive filtering, the app offers an engaging experience that adapts to the user's needs.

3 Explanation of Algorithms in LifeList

The LifeList's algorithms are geared towards performance ensuring that the user always has access to the data with minimal delays, ensuring that it the App remains efficient and responsive.The SQLite database that is used has also been designed to be scalable whilst still maintaining efficiency.

3.1 SQLite Tables

The LifeList database is comprised of 3 tables Birds, allBirds and Provinces

3.1.1 AllBirds Table

The allBirds table stores all the data for the 800+ birds in South Africa.Images are also store as a Blob of base64. The user also has the ability to request for this table to be updated giving the latest data when they require it.

Why it's effective:

- **Offline access:** By keeping this table the user is able to access all information about all the birds in South Africa at anytime regardless of internet access.
- **Better response time:** Having no need to constantly make request to the online API means that the information is always readily available and able to be displayed.
- **Lower storage requirements:** By storing images as a string of base64 we drastically decease the size of the database
- **Redundancy:** All birds also stores the online URL for the images this means that should the image fail to load the online image is retrieved and stored in the table for next usage.

3.1.2 Bird Table

The `Bird` table stores rudimentary bird data for birds that have been seen.

Why it's effective:

- **Saves storage:** By keeping this table small we minimize storage requirements.
- **Seamless Integration:** As the `Bird` stores minimal information it make it easier to upload to the users online account allowing for the transfer of data to be quicker and more cost effective.

3.1.3 Provinces Table

The `Provinces` table stores the Bird is and a list of provinces that the bird is found in.

Why it's effective:

- **Efficiency :** This table allows for 1 query to be made to find number of birds in a province and which birds are found in which provinces.
- **Storage Efficiency:** This allows the `AllBirds` table to be made smaller and faster.

3.1.4 Online Profile Update

The `fetchUserLifelistString` retrieves the necessary data from the `Birds` table that is need for online profile storage.

Why it's effective:

- **Efficiency :** This allows for efficient retrieval of the life list to be stored online .
- **Storage Efficiency:** This allows the online storage to be smaller and more manageable.

3.1.5 Duplicate inserting

The `isDuplicate` checks if the bird to be inserted is in the life list already.

Why it's effective:

- **Efficiency :** This prevents duplicate data from even being attempted to be inserted and it allows the front end to identify if a bird is in the Life List without being queried.

3.1.6 Bird Provinces

The `getBirdProvinces` return all the provices that a bird is in.

Why it's effective:

- **Efficiency :** Allow for 1 query that retrieves all provinces that a bird is seen in.

3.1.7 Achievement Progress

The `precentLifeListBirds` returns the percentage of the current achievement. By querying `Provinces` and `Birds` tables we are able to calculate the percentage of the achievements.This query is called upon inserting a bird into life list and the value is stored in the user model.

Why it's effective:

- **Efficiency :** Allow for 1 query that retrieves the current percentage.
- **Seamless Integration :** By storing this data in the user model it is always available when need so loading time is minimized.

3.1.8 Overall Design

These tables and algorithms ensure that the LifeList as well as all bird information is available at all times. By loading the LifeList as the app starts we ensure that the data is always readily available to the user anytime the user needs it. These algorithms are also geared to minimize loading times and save storage space by breaking the data down into smaller and more manageable tables.