GlobalTourGuide App Executive Report

Date: August 19, 2025

1. Introduction & Background

GlobalTourGuide is designed for the moment a visitor needs to decide between several attractions and translate those decisions into a viable plan. Instead of contributing more unstructured tips, the app offers a map-first, globally consistent interface where each point of interest (POI) is tied to location, augmented by authoritative attributes (ArcGIS Living Atlas), and supplemented by structured community ratings (1–5) and tallies so signals are comparable across cities and nations. On an OpenStreetMap basemap, visitors can instantly compare near or distant sights by opening a callout to view vetted information and aggregated scores, then craft their itinerary by adding custom attractions—personal POIs with name, type, description, URL, and coordinates—that remain on the map alongside official features. The target audience is visitors and trip organizers who crave location-accurate, like-for-like comparisons and an easy way to log their own stops; a secondary audience is destination managers who appreciate standardized, georeferenced feedback. From day one the experience is global-ready: the comparison logic, data schema, and interactions all remain the same everywhere in the world, making the process of assessing and planning consistently feel the same whether in Rome or New York.

2. Methods

The development of GlobalTourGuide followed a full-stack approach, combining modern cloud services with an interactive mobile front end. The back end is built on Amazon Web Services. A relational database schema was designed using PostgreSQL to store attraction and review data. Two core tables were created:

- attraction – storing a unique identifier, name, description, tourism type, website, latitude, longitude and creation timestamp for each site.
- reviews – storing a review identifier, foreign key to the associated attraction, a double-precision rating between 1.0 and 5.0, and a text comment.

This schema was implemented on an Amazon RDS instance, configured with spatial extensions (PostGIS) to enable future spatial queries. Database administration was managed through pgAdmin.

Serverless computing formed the core of the application logic. Multiple AWS Lambda functions were created in Node.js to implement discrete API operations, such as adding an attraction, submitting a review, retrieving all attractions, retrieving an attraction by its ID and fetching top-rated attractions. Figure 1 shows the API Gateway stage configuration and the Lambda functions used in the deployment. Figure 2 shows deployed AWS lambda functions. Functions associated with transit data (getNearbyTransit and updateTransitData) were deliberately excluded from deployment due to the high cost and complexity of obtaining reliable, global real-time transit feeds. An API Gateway was

configured to expose these functions as RESTful endpoints, handling request routing, authentication and Cross-Origin Resource Sharing (CORS). The choice of a serverless architecture allows the system to scale automatically without the need to manage servers, reducing operational overhead and cost.

On the client side, the Android application was developed using Android Studio and the ArcGIS Runtime SDK for Android. The app initializes an OpenStreetMap base map and loads feature layers from the ArcGIS Living Atlas. A set of XML layout files defines the user interface, including map controls, dialogs for adding attractions and reviews and spinners for choosing tourism categories. The Kotlin code in MainActivity.kt handles map interaction, listens for user taps, displays callouts with attraction attributes, collects user input for new points and sends structured JSON requests to the API. Ratings are constrained to a 1–5 scale, and the app computes average scores on the fly. Asynchronous calls to the API ensure that data retrieval and updates occur seamlessly, maintaining a fluid user experience.

Throughout the development process, the project drew upon key areas of the GIS&T Body of Knowledge (BoK). It reflects knowledge areas in geospatial data management (GEOG574 database design, spatial indexing), geospatial computing (GEOG 576,GEOG378 serverless architecture, API design), cartographic design (GEOG370,GEOG475 map symbolization, interactive callouts) and spatial analysis (GEOG 574 aggregation of ratings). These foundations guide design decisions and align the project with state-of-the-art geospatial practice..

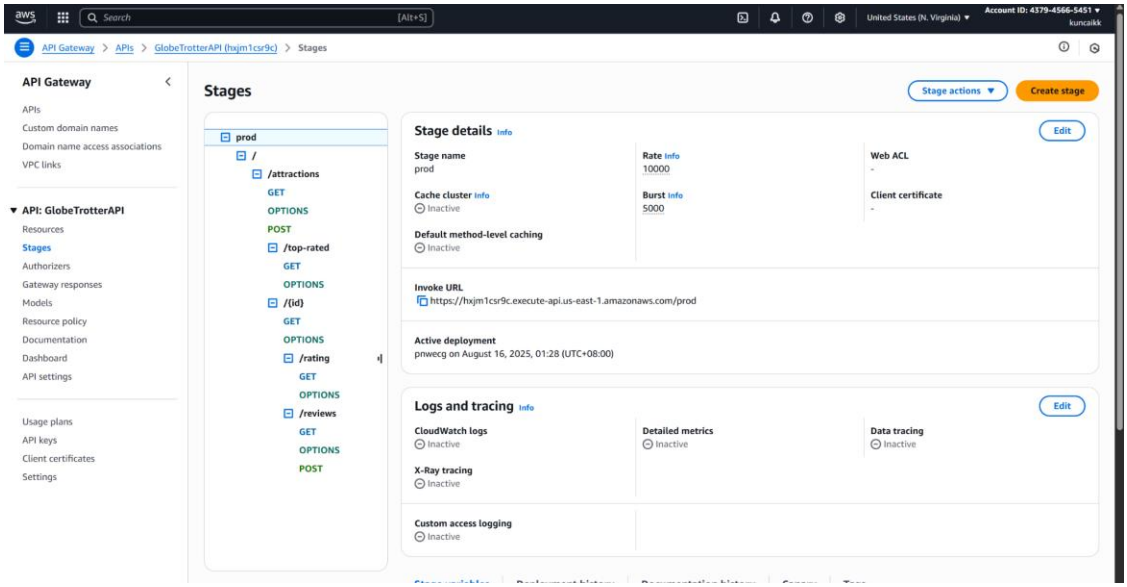

Figure 1: API Gateway stage and resource methods

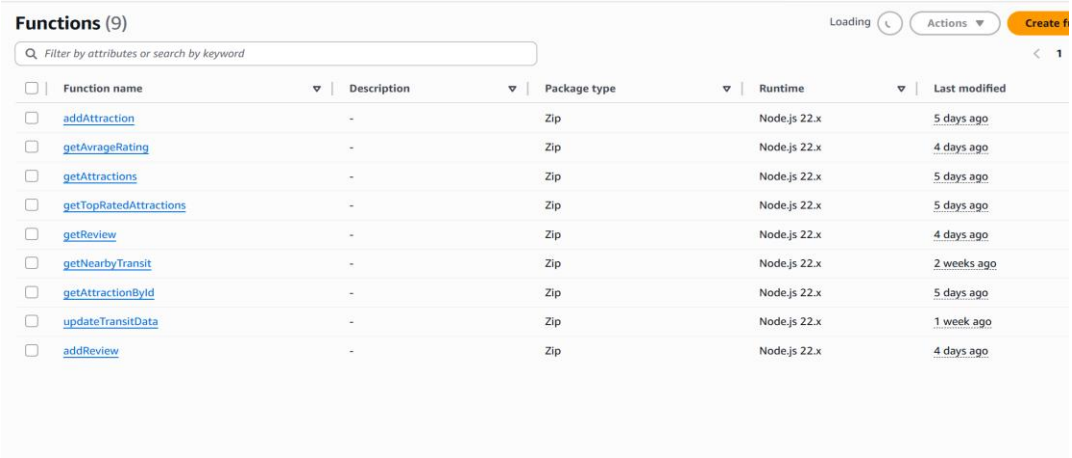

Figure 2: Deployed AWS Lambda functions

3. Architecture / Software Diagram

The software architecture of GlobalTourGuide follows a client–server model orchestrated through AWS. As depicted in Figure 3, the mobile app composes user input into JSON requests and sends them to the API Gateway. The API Gateway routes each request to the appropriate Lambda function, which executes the server-side logic and interacts with the PostgreSQL database. Responses are returned to the client as JSON, and the mobile app updates its map and user interface accordingly. The architecture emphasises modularity: each Lambda function performs a single operation, making the system easier to maintain and extend. Security and authorization are handled at the API Gateway, and the database layer isolates data storage from business logic, enabling future use cases or additional client applications (such as web or iOS) to reuse the same APIs.

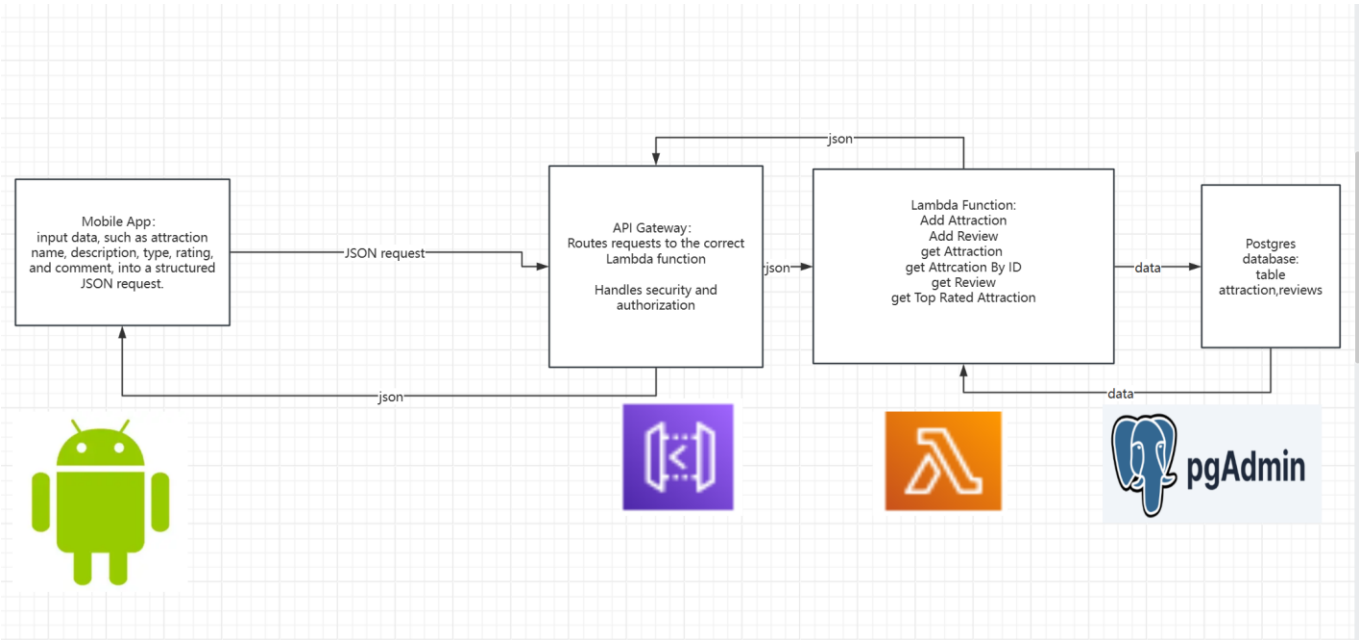

Figure 3: System architecture

Figure 3: High-level system architecture illustrating the interactions between the mobile client, API Gateway, Lambda functions and PostgreSQL database.

4. Results

During testing, the **GlobalTourGuide** app delivered a smooth sequence for exploring the world and planning trips. The experience begins with the OpenStreetMap basemap: upon launch the map centers on the user's approximate position, and the user can freely pan and zoom. A search box at the top enables jumping to any city or landmark by name, while a location marker indicates current GPS coordinates (Figure 4). Overlaid on the map are the orange star symbols of the **ArcGIS Living Atlas** feature layer, providing immediate visual access to attractions around the globe. The plus and minus buttons allow rapid changes in scale as users navigate from a continental view down to street level.

Figure 4: Basic Function (Basemap, Current Location, Zoom In/out, load Feature Layers)

Clicking an attraction opens a detailed callout containing authoritative attributes from the Living Atlas—such as the attraction name, classification and official website URL—along with an aggregated **average rating** and **review count** derived from the project's database (Figure 5). Within this callout, users can choose to **add a review** by selecting a 1–5 star rating and writing a comment, which is sent via the API and instantly updates the average score; or they can **view reviews**, which opens a list of existing feedback tied to that exact location. This direct coupling of community sentiment and map location supports meaningful comparison between sites across cities and countries.

Figure 5: Click on attraction, show attribute table

For sites not already represented, the bottom of the interface features an **Add Attraction** button. Tapping it presents a form where users enter the name, description, tourism type and an optional website for a new point of interest (Figure 6). When submitted, the app captures the coordinates of the tapped location, constructs a structured JSON request and posts it to the addAttraction endpoint. Upon success the new attraction appears immediately on the map as a custom symbol, inviting other users to rate and review it. This crowd-sourcing capability ensures the database grows beyond the Living Atlas to include hidden gems and recently opened destinations.

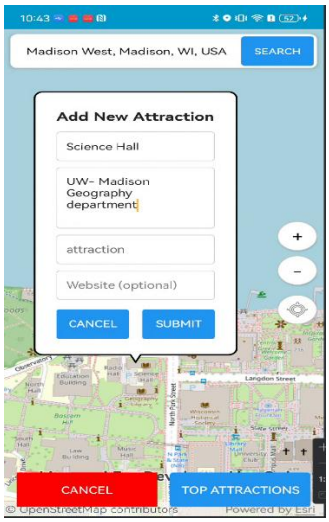

Figure 6:Add my custom attration

Rounding out the exploration workflow is the **Top Attractions** button. When tapped, the app requests the highest-rated attractions—computed from average scores and review

counts in PostgreSQL via the Lambda API—and presents them in a ranked panel (Figure X). Each row shows the attraction name, its current average rating, and the number of reviews. **Tapping any row immediately recenters and zooms the map to that POI and opens its callout**, letting the user inspect authoritative attributes or add a review without extra steps. Together with search, on-map pop-ups, and custom POIs, this tight list-to-map loop enables like-for-like comparison of recommended places and helps travelers plan an itinerary from a single, map-first interface.

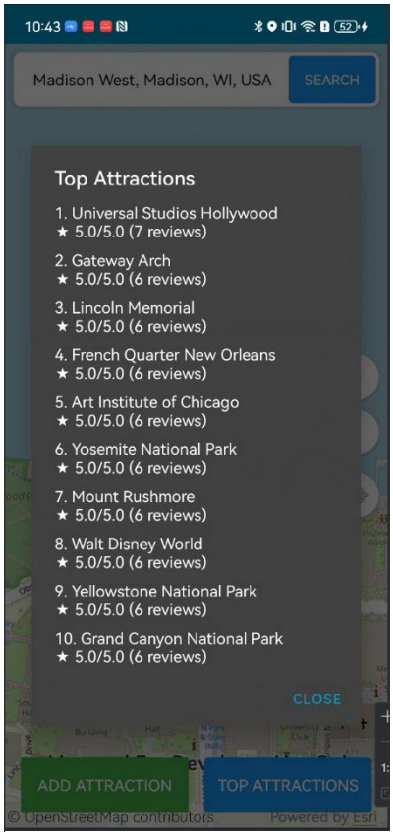

Figure 7:View Top Attractions

5. Conclusion

GlobalTourGuide demonstrates the possibility of using geospatial technologies and serverless cloud computing for realizing a comprehensive, community-driven tourism guide. The key lessons gained in the project are the advantages of following a modular, serverless design that scales automatically with the needs of the users, and the value of combining authoritative data and user contributions for making the data increasingly trustworthy. The main contributions of the project are:

1. A global platform that integrates authoritative geospatial datasets with crowd-sourced ratings.

2. A hands-on demonstration of end-to-end mobile GIS development with ArcGIS Runtime SDK and AWS.

3. An extensible data model that can accommodate other themes outside tourism.

However, there are restrictions. The choice to leave out transit functions indicates financial and technical obstacles to the inclusion of real-time information. The application assumes ongoing internet access and lacks offline caching or support for alternative mobile platforms as of yet. Quality of data is subject to the contributions of users, and moderation or spam detection mechanisms are still left for future work.

Future developments would involve incorporating offline functionality for regions with poor connectivity, incorporating low-cost transit or routing services, user authentication and profile management, expanding the domain to restaurants, events or cultural attractions, and offering analytics to discover emerging tourist patterns. Iterating from this foundation, GlobalTourGuide can develop into a rich, multi-layered geospatial platform that facilitates various place-based decision making.

6. Conceptual Knowledge & Technical Skills

The project utilizes several knowledge areas of the GIS&T Body of Knowledge, demonstrating state-of-the-art geospatial practice. In data management, it demonstrates good spatial database design in PostgreSQL and PostGIS, with primary keys, foreign keys and spatial columns defined. In geospatial computing, it demonstrates using serverless architecture (AWS Lambda) and API management (API Gateway) to host scalable geospatial services. In cartographic design, the app demonstrates principles of symbolization, map layering and interactive callouts in presenting geographic information effectively on mobile devices. For spatial analysis, it performs aggregation (mean ratings) and filtering (top-rated selection), and the design anticipates more complex analyses such as spatial nearest-neighbour queries when cost constraints are overcome.

Technical skills used include relational schema design, SQL programming, REST API development with Node.js, asynchronous networking with Kotlin, geospatial data visualization using ArcGIS Runtime SDK and integration of external data sources with ArcGIS Living Atlas. The project also required expertise in cloud infrastructure configuration (IAM roles, environment variables, cross-origin resource sharing) and mobile UI development with Android XML layouts. These skills cumulatively demonstrate expertise in geospatial data management, application development and cartographic design at a professional level.