Welcome to Bird App!

# Introduction

The goal of our application is to help a user who is interested in bird watching to plan a trip from scratch with only their starting address. The user must choose a specific bird species and/or location that interests them, and then get back directions to the birdwatching spot. In terms of architecture, our application runs on serverless synchronous API RESTful architecture. The contributions of our tool are that it is user-friendly, and it releases the user from the burden of interacting with more complex APIs or understanding more complex outputs. Our tool integrates several APIs in one, interacts with a cloud database that is integrated in our architecture system, and manipulates the outputs of APIs such that they’re easily readable.

# API Architecture Design

Figure 1 shows the architecture of our API. As shown in the figure the client interacts with an API that we have created through API Gateway. When the client calls our API through its corresponding API URL, it automatically triggers the corresponding lambda function that is associated with the specific API resource method. The lambda functions will then interact with external APIs or with our RDS and perform the necessary computations or extract the necessary data. The lambda functions also have access to the necessary access keys for the Simple Email Service (SES) method to increase security of our service. The four external APIs that our application interacts with are as follows:

1. [Geoapify- Forward Geocoding API](https://apidocs.geoapify.com/docs/geocoding/):

This API was used to convert a place to global coordinates

1. [Geoapify- Forward Routing API](https://www.geoapify.com/routing-api/):

This API was used to convert route the user and create

1. [Ebird API](https://documenter.getpostman.com/view/664302/S1ENwy59):

This API was used to convert ….

1. [Nominatim API](https://nominatim.org/release-docs/develop/api/Overview/):

This API was used to convert …

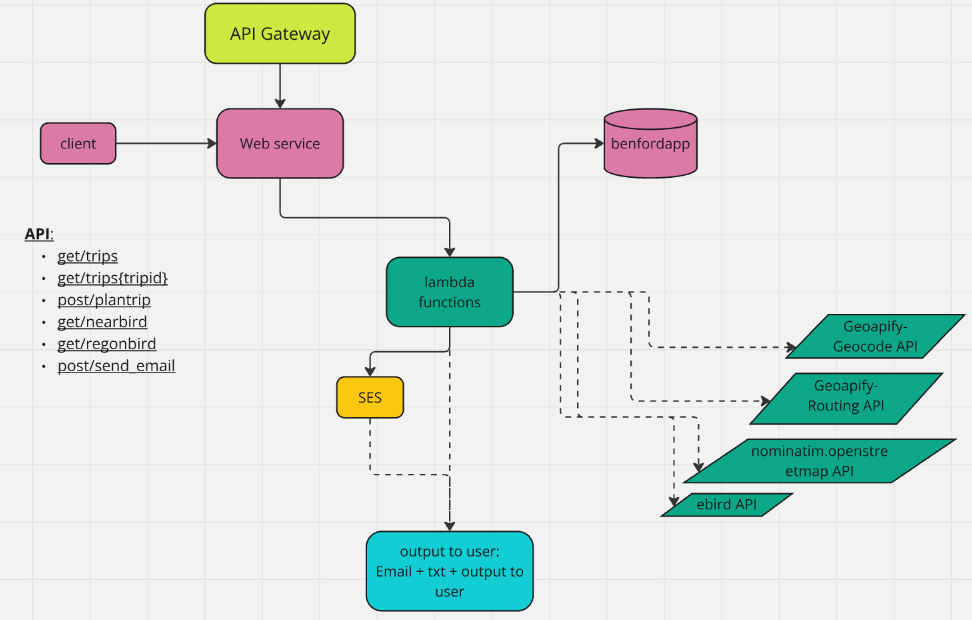


Figure 1: API Architecture for the Application

# Server-side

## Relational Database Service (RDS)

We utilized the RDS component of AWS where we built a table to save past trips’ data based on all previous requests by past users to guide new and existing users. The table was created using sql. It included the following columns:

* trip ID
* Common bird name
* Startin location
* Ending location
* Transportation mode
* Distance of the trip in Km
* Instructions to arrive at destination

## Non-trivial Operations

· Interacted with 4 APIs

· Extracted data and processed it and saved it in database & Downloaded text file

· Sent emails

·

· 1. calling an API or other tool that converts addresses to latitude and longitude coordinates 2. calling the eBird API to retrieve recent bird observations in an area and 3. calling a trip planner API that returns the optimal route from x to y given a specified preferred mode of travel (walk, bike, train, car). We are thinking of incorporating a database that stores already used addresses to save time computing their coordinates and also perhaps a table that stores already planned routes so that one can easily access them after computing them for the first time.

## Lambda Functions

Our tools is built on 6 lambda functions:

**Trip planner (/plantrip POST):**

* Interacts with our RDS to save data
* Interacts with Geoapify- Geocode API
* Interacts with Geoapify- Routing API

**Download Trip (/trips{tripid} GET):**

* Interacts with our RDS to extract data and report to user

**Past Trips (/trips GET):**

* Interacts with our RDS to extract data and report to user

**Nearby Birds(/nearbird GET):**

* Interacts with nominatim.openstreetmap API
* Interacts with ebird API
* Reports to user

**Region Birds(/regionbird GET):**

* Interacts with ebird API
* Reports to user

**Send\_email (/send\_email POST)**

* Gets recipient’s email address and trip details from Client.
* Sends user email with Trip details.

## Api config

· API url1: https://ssgmv6u3w1.execute-api.us-east-2.amazonaws.com/proj\_prod

* get/trips
* get/trips{tripid}
* post/plantrip
* get/nearbird
* get/regonbird

API url2: https://rpo6ydyluh.execute-api.us-east-2.amazonaws.com/prod

* post/send\_email

# Client-Side

## Prompts Given to client

The client has the option to choose from different functions:

1. Show past trip request by all previous users + Send email with a selected trip details and instructions
2. Plan trip to the location of interest for bird watching, by different modes
3. Download trip details and instructions
4. See nearby birds within 50 km radius from your address
5. See birds in region of your request

## Configuration file of client

The client is provided with two files for configuration that get called by the corresponding function. …

## 