Count Cat App: Installation and deployment

01.08.2021 v.1.0

Jorge Alberto Serrano Paul
Jorge Alberto Serrano Paul
DUNS: 951598088
NCAGE: SKQL4
Cedro No. 263
Ciudad de México, 06400

Considerations

Count Cat App is a web application created to categorise observation data from camera traps in a determined area. The application was designed to be easy to deploy, maintain and use.

This manual covers the process of installation, configuration and deployment of the components that make the entire application.

Architecture

The application was created as a Single Page Application that consumes two independent APIs, connected to the same database.

The frontend is powered by React, while both backend services run on a Node/Express setup connected to a MongoDB cluster.

Backend: Data digest and report download service

Source: https://github.com/Smithsonian/Cat-app-data-digest

This application is in charge of processing the raw data into the **Count Cat App** database. It runs on **Node 14.x LTS.** And it consists of a schedulable job to fetch observations from an API endpoint and an exposed endpoint to download a CSV report guarded by an authentication verification middleware.

Installation

```
npm install
```

Environment

Provide the following environment variables before running the application

```
MONGO_URI=Valid MongoDB connection string. Same database as second service.
JWT_SECRET=String to encode JWT across the entire application
CAT_API=Endpoint from where the app will fetch data
LOGS_PATH=Local directory to create logs
CRON=Frequency of data digest in cron format
ORIGIN=Allowed origins to send requests (CORS)
PORT=Will fallback to 8000
```

Start application

```
npm start
```

Schedulable data digest

In order for the application to run the data digest on a regular and automatic basis, an environment variable named **CRON** needs to be provided. Scheduling the task to run the first day of every month is recommended.

Raw data definition

The application is prepared to send a request to an endpoint on **CAT_API** that takes two URL parameters:

- month: Number (1-12)
- year: Number (>1970)

A response in JSON format is expected as follows:

```
[
{
        "image_id": String,
        "date_time_original": "YYYY-MM-DD HH:MM:SS",
        "sub_project_id": String,
        "deployment_id": String,
        "sequence_id": String,
        "count": Number,
        "latitude": Number,
        "longitude": Number
}
]
```

Where every object in the response is an image within a sequence in a camera trap.

- **image_id**: unique id that identifies the image within the sequence
- **date_time_original**: date of the observation
- **sub_project_id**: Id that identifies the project within your institution
- **deployment_id**: Id of the deployment and/or camera trap
- **sequence_id**: Id of the sequence to which the image belongs to.
- **count**: number of individuals in the sequence
- **latitude**: latitude
- **longitude**: longitude

Observation model

The application will group images in the same sequence into a single observation and will save it into the database as per the following model.

```
{
  project_id: { type: String, required: true },
  deployment_id: { type: String, required: true },
  sequence_id: { type: String, required: true },
  location: {
      type: {
      type: String,
      enum: ['Point'],
      default: 'Point'
      },
      coordinates: {
      type: [Number],
      required: true,
      index: '2d'
      }
  },
  date_time_original: { type: Date, required: true },
  forReview: { type: Boolean, default: false },
  reasonReview: {
      type: String,
      enum: ['None', 'Not cat', 'New cat', 'New match', 'Unidentifiable'],
      default: 'None'
  },
  specimen: { type: ObjectId, ref: 'Specimen' },
  pattern: {
      type: String,
      enum: [
      'Black/Gray',
      'Tabby/Spotted',
      'Orange/White',
      'Tortoiseshell/Calico',
      'Siamese',
      'Unknown'
      ],
```

```
      default: 'Unknown'
  },
  bicolor: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  longHair: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  sex: {
      type: String,
      enum: ['Male', 'Female/Neutered', 'Unknown'],
      default: 'Unknown'
  },
  notched: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  collar: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  isCat: { type: Boolean, default: true },
  notID: { type: Boolean, default: false },
  images: { type: [{ image_id: { type: String, required: true } }],
required: true }
}
```

Download report (CSV)

The application exposes an endpoint from which a CSV report can be downloaded as an attachment. A JWT needs to be present in the header of the request as **token** in order to authenticate the user. This token needs to be retrieved by sending a POST request to the **/signin** endpoint of the second backend service.

Endpoint: `/startDate/:startDate/endDate/:endDate`

startDate: Date

endDate: Date

Report content

The CSV report contains the following columns as specified in **controllers/observations.js**

```
[
      {
      label: 'Observation ID',
      value: '_id'
      },
      {
      label: 'Deployment ID',
      value: 'deployment_id'
      },
      {
      label: 'Sequence ID',
      value: 'sequence_id'
      },
      {
      label: 'Cat ID',
      value: 'specimen'
      },
      {
      label: 'Pattern',
      value: 'pattern'
      },
      {
      label: 'Bicolor',
      value: 'bicolor'
      },
```

```
    {
    label: 'Long hair',
    value: 'longHair'
    },
    {
    label: 'Sex',
    value: 'sex'
    },
    {
    label: 'Notched ear',
    value: 'notched'
    },
    {
    label: 'Collar',
    value: 'collar'
    },
    {
    label: 'Latitude',
    value: 'location.coordinates[1]'
    },
    {
    label: 'Longitude',
    value: 'location.coordinates[0]'
    },
    {
    label: 'Date',
    value: 'date_time_original'
    }
]
```

Backend: main API

Source: https://github.com/Smithsonian/Cat-app-backend-2021

This application is in charge of providing data for the **Count Cat App**. It runs on **Node 14.x LTS.** And it consists of a collection of RESTful endpoint for authentication and observations.

Installation

```
yarn install
```

Environment

Provide the following environment variables before running the application

```
APP_NAME=Name of application, used for emailing and navbar
APP_URL=URL of production application, used for emailing
MONGO_URI=MongoDB connection string, shares database with data digest
service
JWT_SECRET=String to encode JWT across the entire application
ORIGIN=Allowed origin (CORS)
SMTP_HOST=For emailing
SMTP_PORT=For emailing
SMTP_USER=For emailing
SMTP_PASSWORD=For emailing
PORT=Fallsback to 5000
```

Start application

```
yarn start
```

Auth

User Model

```
{
  name: { type: String, required: true },
  role: { type: String, required: true, enum: ['user', 'admin', 'master']
},
  email: { type: String, required: true },
  password: { type: String, required: true, select: false },
  active: { type: Boolean, default: true }
}
```

Sign up | POST /auth/create-user

To register new users. Expected response JWT with user payload. An email is sent to the new user with an auto-generated password.

A **token** header belonging to an admin account needs to be sent.

Sample request

```
{
      "name": String,
      "role": String,
      "email": String,
      "active": Boolean
}
```

Sign in | POST /auth/signin

To login users. Expected response JWT with user payload.

Sample request

```
{
      "email": String,
      "password": String
```

```
}
```

Toggle status | PATCH /auth/status/:id

Toggles user **active** property.

A **token** header belonging to an admin account needs to be sent.

Toggle role | PATCH /auth/role/:id

Toggles user **role** property.

A **token** header belonging to an admin account needs to be sent.

Get users | GET /auth/users

Retrieves list of users in application

A **token** header belonging to an admin account needs to be sent.

Verify session | GET /verify-session

Validates current JWT for resetting auth state in frontend.

Observations

Observation Model

```
{
  project_id: { type: String, required: true },
  deployment_id: { type: String, required: true },
  sequence_id: { type: String, required: true },
  location: {
      type: {
      type: String,
      enum: ['Point'],
      default: 'Point'
      },
      coordinates: {
      type: [Number],
      required: true,
      index: '2d'
      }
  },
  date_time_original: { type: Date, required: true },
  forReview: { type: Boolean, default: false },
  reasonReview: {
      type: String,
      enum: ['None', 'Not cat', 'New cat', 'New match', 'Unidentifiable'],
      default: 'None'
  },
  specimen: { type: ObjectId, ref: 'Specimen' },
  pattern: {
      type: String,
      enum: [
      'Black/Gray',
      'Tabby/Spotted',
      'Orange/White',
      'Tortoiseshell/Calico',
      'Siamese',
```

```
      'Unknown'
      ],
      default: 'Unknown'
  },
  bicolor: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  longHair: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  sex: {
      type: String,
      enum: ['Male', 'Female/Neutered', 'Unknown'],
      default: 'Unknown'
  },
  notched: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  collar: {
      type: String,
      enum: ['Yes', 'No', 'Unknown'],
      default: 'Unknown'
  },
  isCat: { type: Boolean, default: true },
  notID: { type: Boolean, default: false },
  images: { type: [{ image_id: { type: String, required: true } }],
required: true }
}
```

Get deployments | GET /observations/deployments

Returns a list of deployment with active observations by an aggregation pipeline on the Observations collection.

A **token** header needs to be sent.

Get dashboard counters | GET /observations/counters

Returns values representing the total numbers of deployments, observations and observations for review to be visualised in the frontend.

A **token** header needs to be sent.

Get candidates | POST /observations/candidates

Returns list of potential candidates for an observation based on location, distance and metadata.

A **token** header needs to be sent.

Sample request

```
{
      "coordinates": Array,
      "distance": Number,
      "pattern": String,
      "bicolor": String,
      "longHair: String
}
```

Get observations | POST /observations/

Returns list of observations based on location, distance and metadata.

A **token** header needs to be sent.

Sample request

```
{
      "coordinates": Array,
      "distance": Number,
```

```
      "pattern": String,
      "bicolor": String,
      "longHair: String
      ...
}
```

Get observation | GET /observations/:id

Returns a single observation based on its id.

A **token** header needs to be sent.

Get cat | GET /observations/cat/:id

Returns a single specimen based on its ID. An specimen is a collection on observations:

Specimen Model

```
{
  matches: [{ type: ObjectId, ref: 'Observation' }]
}
```

A **token** header needs to be sent.

Update observation (medatada) | PATCH /observations/:id

Updates observation metadata.

Sample request

```
{
      "pattern": String,
      "bicolor": String,
      "longHair: String
      ...
}
```

A **token** header needs to be sent.

Create new cat | POST /observations/:id/newcat

Creates new specimen based on the passed observation id

A **token** header needs to be sent.

Match observation to cat | PATCH /observations/:observationId/cat/:catId

Inserts observation reference in existing cat

A **token** header needs to be sent.

Remove observation from cat | POST /observations/:id/removeid

Removes observation reference from existing cat

A **token** header needs to be sent.

Frontend

Source: https://github.com/Smithsonian/Cat-app-frontend

React 17 SPA.

Installation

```
yarn install
```

Environment

Provide the following environment variables before running the application

```
REACT_APP_OBSERVATION_API=Base URL of Backend API
REACT_APP_IMAGE_BUCKET=Base URL for images hosting bucket
REACT_APP_CSV_SERVICE=Base URL of data digest API
```

Build application

```
yarn build
```

Setting initial search bounds

Modify only the fields pertaining to the location. Rebuild the application afterwards.

```
// utils/searchFormHelpers.js
export const initialForm = {
  minLon: -76.88644409179689,
  maxLon: -77.14256286621095,
  minLat: 38.849333913235476,
  maxLat: 38.95406929344106
};
```

Setting map center and bounds

Modify only the fields pertaining to the location. Rebuild the application afterwards.

```
// utils/leafletConfig.js
export const center = [38.9072, -77.0369];
export const bounds = [
  [50.505, -29.09],
  [52.505, 29.09]
];
```