Critterycovery.me - Technical Report

CS373 - IDB (Phase 1)

William Crawford, Sahithi Golkonda, Shaharyar Lakhani, Daniel Qu, Brian Wang

Team: Group 16, 11 AM
Website: critterycovery.me
GitLab: https://gitlab.com/cs373-group16/critterycovery
Postman: https://documenter.getpostman.com/view/14742162/Tz5jfM1L

Table of Contents

Team

Name	EID	GitLab ID
William Crawford	wjc844	WilliamCrawford
Sahithi Golkonda	sg47288	sahithi-golkonda
Shaharyar Lakhani	sl46398	shaharyarlakhani
Daniel Qu	dq764	d.qu
Brian Wang	bw25755	bwang1008

Motivation

Many animals are on the verge of becoming extinct on Earth. In the past decade alone, almost 500 species have gone extinct, and it is unlikely to slow down anytime soon. There are many causes for this, such as deforestation, pollution, and global warming, to name a few.

Unfortunately, this is an issue that rarely receives attention, and most people are apathetic to the condition of severely endangered animals. We want this to change. The problem of endangered species is not something that can be resolved overnight; instead, we have to bring awareness of the seriousness of this problem to public attention so that collective action can be made to save these endangered species. It is our responsibility as living beings on Earth to take care of our home and to make sure every animal has a sustainable environment to live in.

Our goal is to create a website that will raise awareness for endangered animals and help people learn about the habitats and environments that they live in. Educating users about animals that are almost extinct is the first step towards resolving this global problem.

User Stories

Phase I - User Stories Received as Developer

The first of our user stories mentioned wanting to see the most endangered species (those with the smallest populations left). For Phase I, we have put some instance pages of a few endangered species, but we have no method of sorting them by attributes as of now. Currently, we have displayed characteristics of the three models that can be sorted, but the sorting is not implemented yet. We commented on this user story saying we would sort during a later phase, and they changed their request to just viewing some endangered species.

The second of our user stories mentioned wanting to see endangered species in their region. We do have our website have links between instance pages (such as between a species and the region where it's located), but as of now, we do not have filterable requests, nor do we have any endangered species-specific to Austin on our website yet. However, this is a feature that we thought would be great in the final product.

The third of our user stories mentioned wanting to see the diets of endangered species. As of now, some of the APIs we are scraping for data do not contain information about diets, but we will do our best to address this issue. In the worst case, we will change the attributes that are listed on our websites under species.

The fourth of our user stories wanted to see images of the endangered species. We are incorporating images of these species into our splash page, the model page for species, as well as our instance pages.

The fifth user story wanted to see the wealth of countries as an attribute under our model of countries. We represent this with the GDP of each country.

Phase I - User Stories Sent as Customers

We have also added our own user stories for the team Around Austin at (https://gitlab.com/jyotiluu/cs373-aroundatx/).

Our first user story was that we wanted to access the website through an URL. We wanted the developers to connect the domain obtained from Namecheap with their AWS or GCP server so that their website would be up for public users like us to see. The developer group marked it resolved on March 4.

Our second user story was that we wanted to see a homepage on their website. We wanted to be able to access their three model pages from their home page. The developer group marked it resolved on March 5.

Our third user story was that we wanted to see a model page for incidences (one of their three models). We wanted to see that it would lead to at least three instance pages and that the model page for incidences would be accessible from the home page. The developer group marked it resolved on March 5.

Our fourth user story was that we wanted to see a model page for apartments (another one of their three models). There should be links to some instance pages, and the model page should be accessible from the home page of the website.

Our fifth user story was that we wanted to see a model page for restaurants (the last of their three models). There should be links that lead to instance pages, and the home page should have a link to this model page as well.

Our sixth user story was that our user stories were not being marked with the Customer label because we did not have the right access permissions (of Reporter).

Restful API

We defined some GET requests that we would like to see implemented because they would be useful. In particular, we defined the following:

- `/api/species`: Retrieves all species
- `/api/species/:scientificName`: Given a scientific name of a particular species, this would retrieve information about that specific species, such as its common name, habitat, and endangered status
- `/api/habitat`: Retrieves all habitats
- `/api/habitat/:habitat`: Given a specific habitat, this would retrieve information about that habitat, such as its common name, ecosystem type, and where it is located
- `/api/country`: Retrieves all countries
- `/api/country/:stateID`: Given the ID, this would retrieve information about that region, such as its capital, population, and location.

Models

- Species

  This model displays all the different types of species in a grid view. Attributes of species include the name of the endangered species, the taxonomy, the weight, the height, the length, the number of the species left on earth, the regions in which it lives, its habitat, its diet, its behavior, as well as a picture of the species. For phase 1, our model included three instances:

  - Antelope
  - Zebra
  - Jaguar
  -

- Habitats

  The habitat model displays the different types of habitats in a table view. Attributes of habitats include temperature, rainfall, sunlight per day, the percent of earth it covers, the classification code of the type of habitat, region, locations, and the number of species as a measure of diversity. For phase 1, our model included three instances:

  - Desert
  - Grassland
  - Forest

- Countries

  The countries model displays the countries in a table view. Attributes of countries include name, the capital, population, area/size, the GDP, the habitats it has, and the endangered species within the state. For phase 1, our model included three instances:

  - Argentina
  - Australia
  - Germany

Tools

- GitLab: Our repository is located at (https://gitlab.com/cs373-group16/critterycovery/-/tree/master). In a later phase, we will use GitLab's continuous integration to run automated tests. We divided our project into the backend and frontend folders.

  In the backend folder, we use Python Flask to be able to update the About page programmatically with stats from GitLab. This is where we also will query APIs to obtain data to put on the frontend. For phase I, this is what we are doing to obtain the GitLab statistics (See file "gitlab.py").

  In the frontend folder, the file "package.json" details the dependencies needed for our frontend to run. Most of the source code is listed in the folder "src", where we use React with TypeScript to generate the structure of the website. The "pages" folder lists the individual files needed for Phase I. The "components" folder implements the functionality for a navigation bar and a table, while the "hooks" folder implements a button to scroll back to the top of our website.
- React Bootstrap: We are using React Bootstrap to stylize our website. We used Cards for our About page, Tables for our model pages, and React-player for our splash page.
- AWS EC2: We are using an AWS EC2 instance to host our website.
- AWS Route 53: We used AWS Route 53 to redirect "critterycovery.me" to our actual website.
- Postman: We used Postman to query from our various APIs, as well as create our API.
- TypeScript: We are using React with Typescript so that we can avoid type errors down the road.
- Flask: We used Python Flask for our backend and API calls.
- Docker: We are using Docker so that the EC2 can pull from our repository and run a container to host our website.

Hosting

Our site is being hosted as an AWS EC2 instance that will start a Docker image that hosts both our frontend and backend. The HTTPS authentication is provided by AWS Route 53. Our domain name of critterycovery.me was registered with Namecheap.

When first starting, we obtain a file with the extension “pem” to be able to remotely access the EC2 instance through ssh. The purpose of using EC2 is that even when we log out of the EC2 instance, it will continue to host the website.

Initially, the EC2 instance Amazon gave us was blank. For it to run our website, we downloaded Docker on it, so that it could run an image and host our website. The first step after accessing the EC2 instance was to install docker. Then we cloned our repository hosted on GitLab to the EC2 instance, and ran the Dockerfile in our repository with “docker build”. This file has all the commands needed for docker to set up the website. First, the file pulls from the master branch of the repository to get the latest version of our code. Next, the file runs “yarn install”, installing all the dependencies for the project and “yarn build”, which creates a static folder for the website to use for the frontend.

Finally, the EC2 instance starts our website with “docker start” followed by the name of our built image. This means our website (critterycovery.me) is accessible to the outside world!

Searching

We are planning to add user searches for species, habitats, and countries in a later phase.

Sorting

We are planning to sort the attributes that we have for each of our models in a later phase.

Filtering

We are planning to add filterable searches on our model pages in a later phase.

Pagination

We are planning to add pagination in a later phase. We have set up the infrastructure for this by building template Cards from React Bootstrap.

Testing

We are adding unit tests in a later phase.

DB

We are planning on using a database in a later phase. Currently, we are leaning towards using Google's database system, as it seems very user-friendly.