Harvard John A. Paulson
School of Engineering
and Applied Sciences

A-Eye App Project Outline

AC295
Advanced Practical Data Science, MLOps

Group Name: Pinkdrink
Anita Mahinpei, Yingchen Liu

Outline

- Problem Definition & Background
- Proposed Solution
- App Design
- Project Scope
- Project Workflow
- Process Flow
- Data
- EDA Results
- Models
- App Directory Structure
- Deployment
- App Demo (Images)
- Todos

Problem Definition & Background

The World Health Organization (WHO) estimated that 314 million people have visual impairment across the world, including 269 million who have low vision, and 45 million who are blind (Ono et al 2010). Many people with visual impairments rely on screen readers in order to access the internet through audio and thus depend on image captions (Yesilada et al 2004). Therefore, **accessibility**, as well as **automatic indexing** and other goals, make accurate **image captioning** an important priority (Hossain et al 2018).

Proposed Solution

We will explore the realm of image captioning by creating an app that allows users to *upload images and have them be captioned in parallel through multiple models*. We specifically focus on creating captions for images of objects and scenery. The app will generate and display three possible captions for the image. Since the intended audiences are visually impaired individuals, the app will provide a text to audio functionality so that the generated captions can be read out loud to the user if desired. In order to support a broad range of audiences, we will allow users to select the language they wish to use for the generated captions.

Proposed Solution

Our models will be trained with a dataset of more than 330k images with 1.5 million object instances, covering more than 80 entity categories. We will explore a full-stack data science process starting from deep learning, operations to deployment and eventually distribute the project app in a contained environment to a wider range of audiences.

App Design

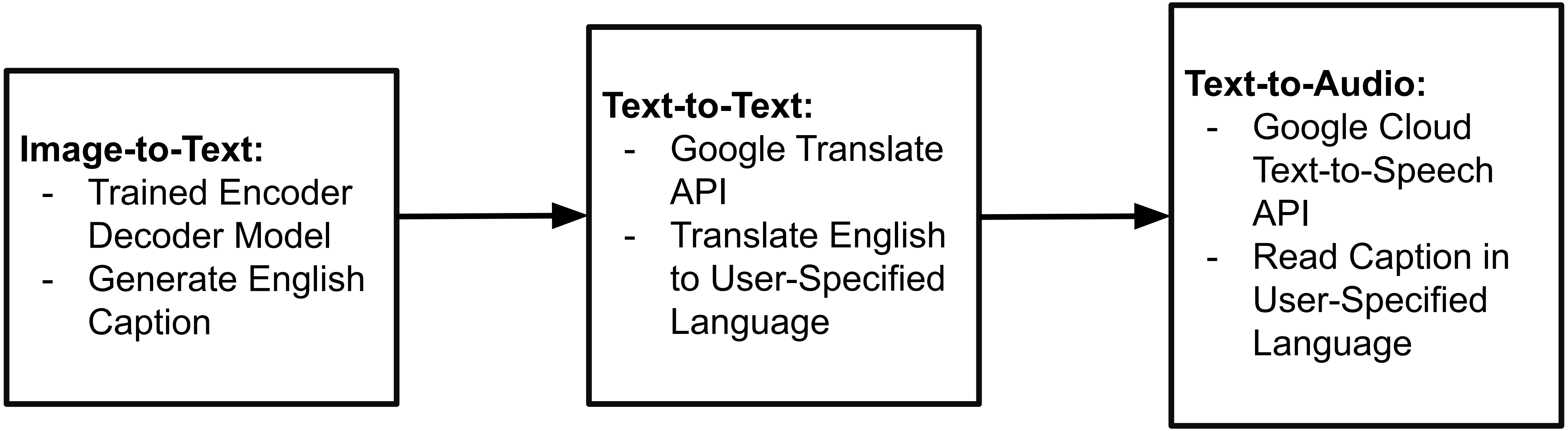


Image-to-Text:

- Trained Encoder Decoder Model
- Generate English Caption

Text-to-Text:

- Google Translate API
- Translate English to User-Specified Language

Text-to-Audio:

- Google Cloud Text-to-Speech API
- Read Caption in User-Specified Language

Project Scope (A-Eye App)

Proof Of Concept (POC)

- Download MSCOCO data
- Perform EDA to verify data
- Set up data pipeline (i.e. resize all images to a fixed size, normalize pixel values, tokenize the captions, store in tf dataset)
- Experiment with some baseline models trained on a subset of the dataset
- Validate captioning results on unseen images

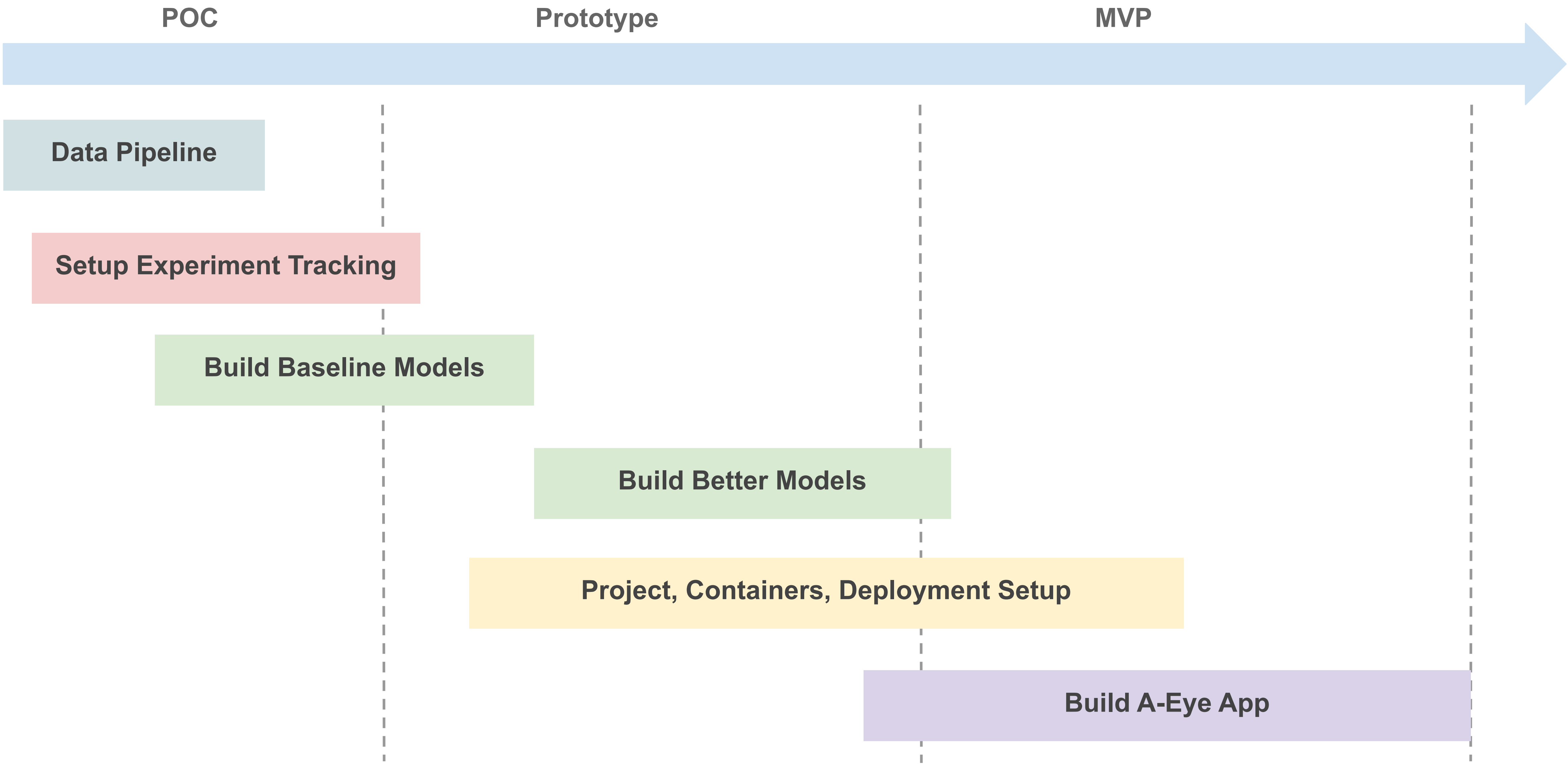
Prototype

- Create a mockup of screens to see how the app could look like
- Deploy one model to Fast API to service model predictions as an API

Minimum Viable Product (MVP)

- Create App to caption images
- API Server for uploading images and predicting using best model

Project Workflow



POC

Prototype

MVP

Data Pipeline

Setup Experiment Tracking

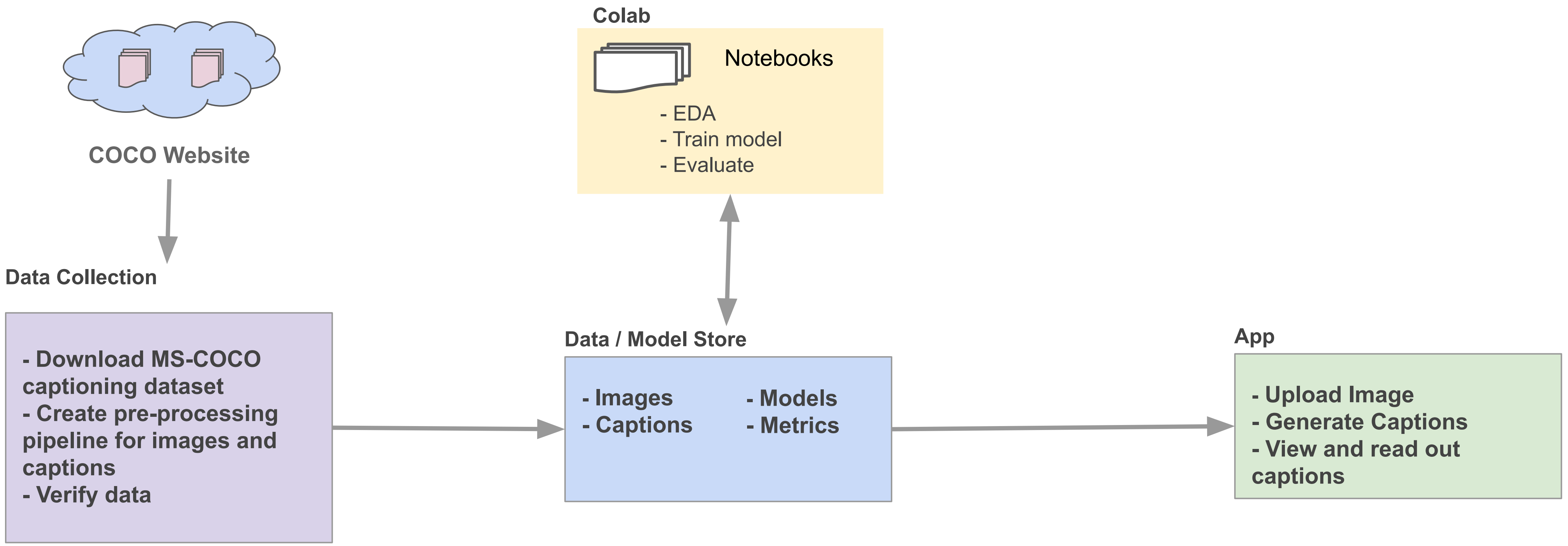
Build Baseline Models

Build Better Models

Project, Containers, Deployment Setup

Build A-Eye App

Process Flow



COCO Website

Data Collection

- Download MS-COCO captioning dataset
- Create pre-processing pipeline for images and captions
- Verify data

Colab

Notebooks

- EDA
- Train model
- Evaluate

Data / Model Store

- Images
- Captions

- Models
- Metrics

App

- Upload Image
- Generate Captions
- View and read out captions

Data

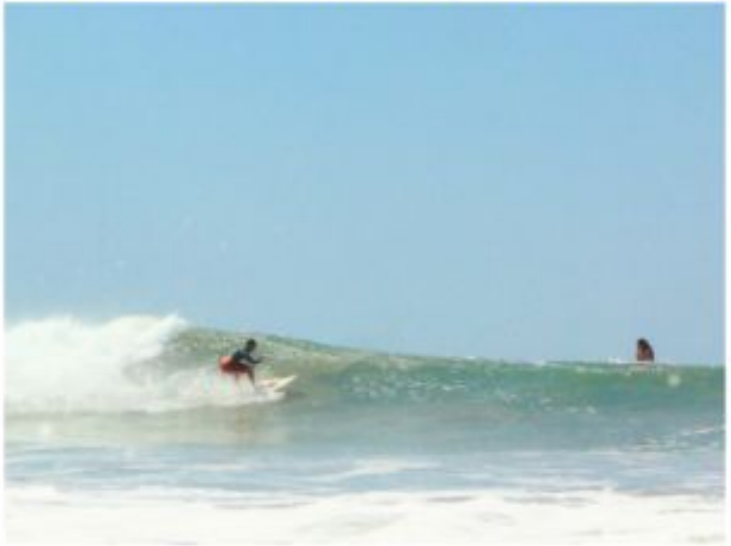
We will use the Microsoft Common Objects in Context (COCO) data for our project. COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 to 7 captions per image
- 250,000 people with keypoints

Data

We will be using the images with the caption labels for training models.

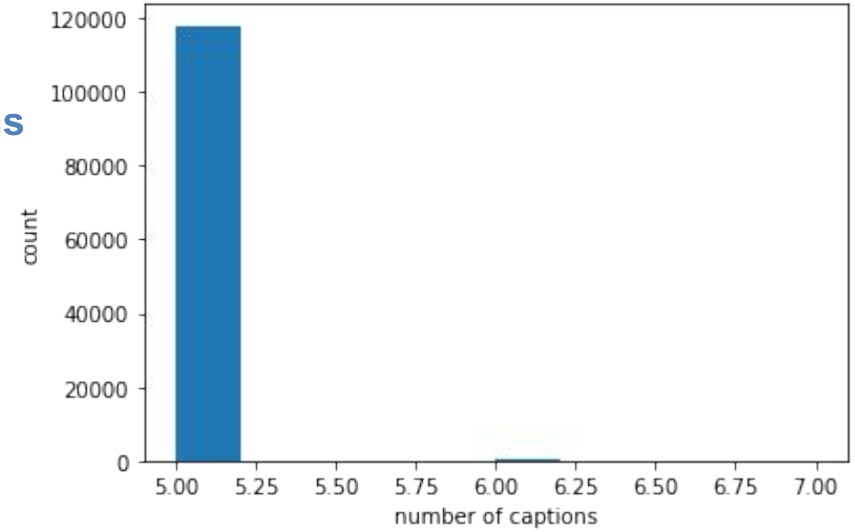
Sample image with 5 gold captions:

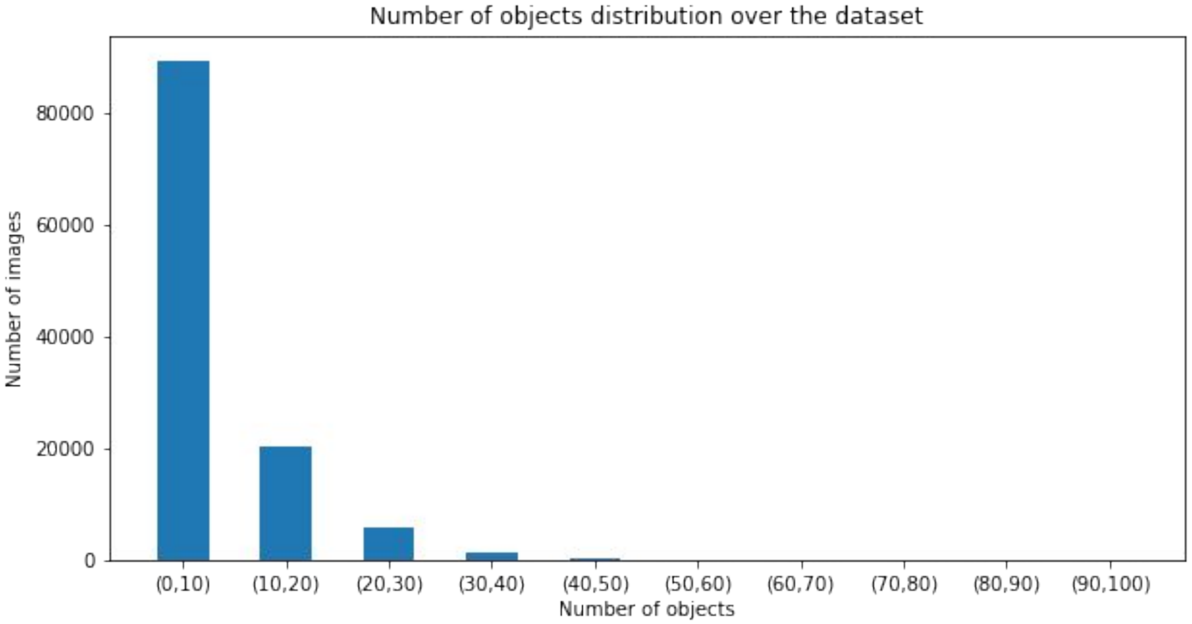


```
A couple of people riding on top of a wave on surfboards.
A man rides a white surfboard near another person in the ocean.
one person surfing one person laying on a surfboard
The guy is riding the wave as a girl watches.
Surfers surfing in the ocean on a clear day.
```

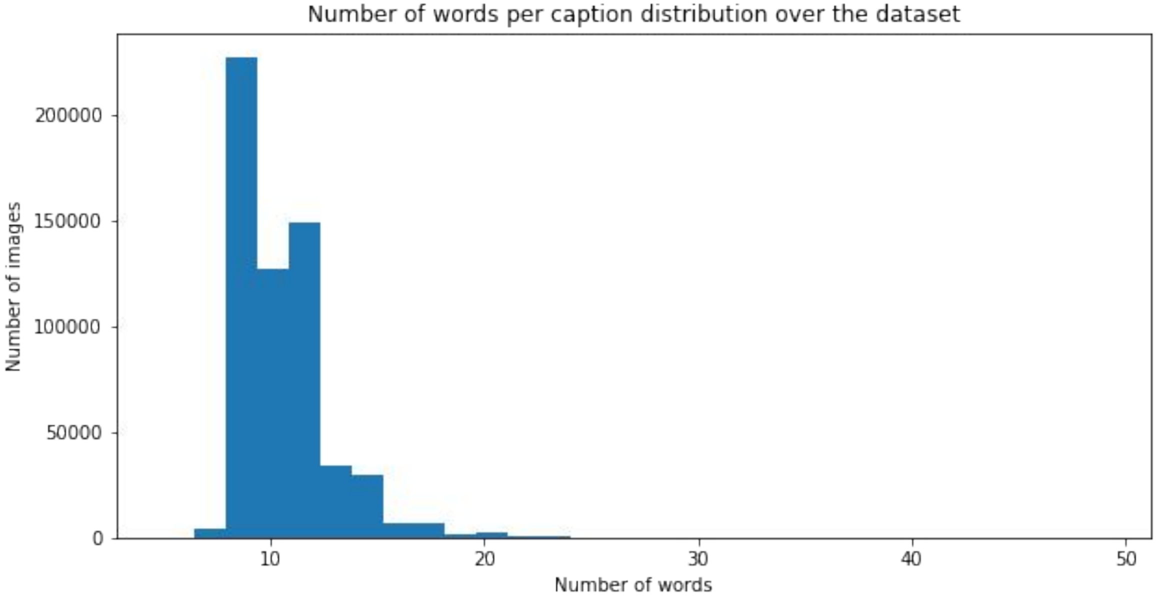
EDA Results

Most images have 5 captions



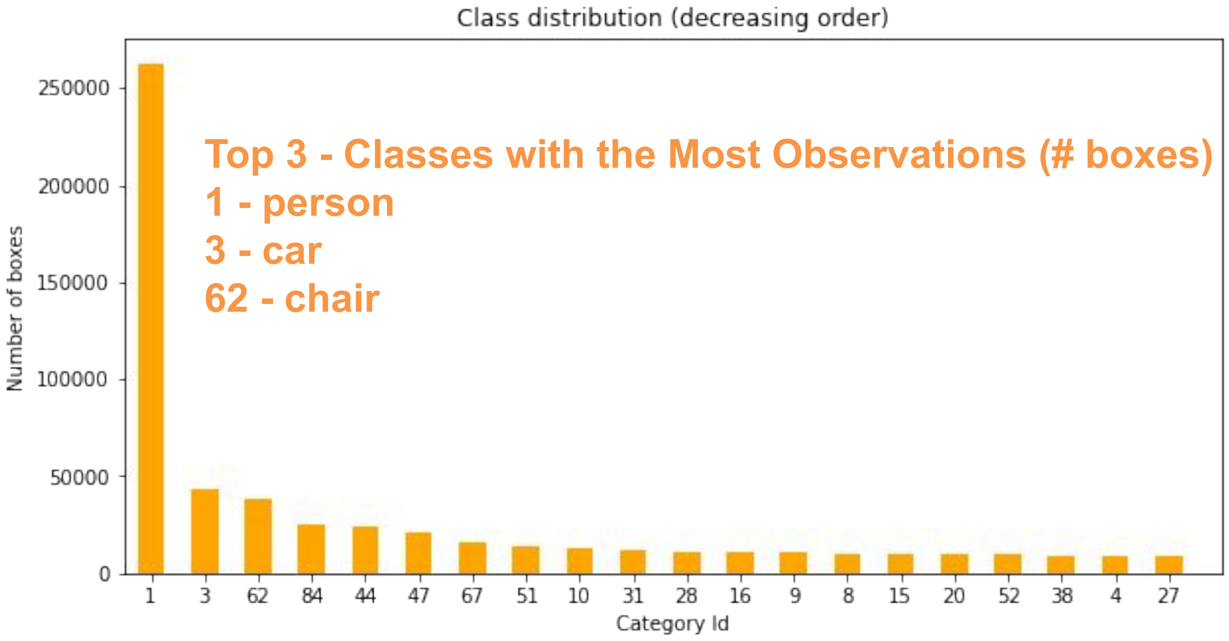


Most images contain less than 10 objects

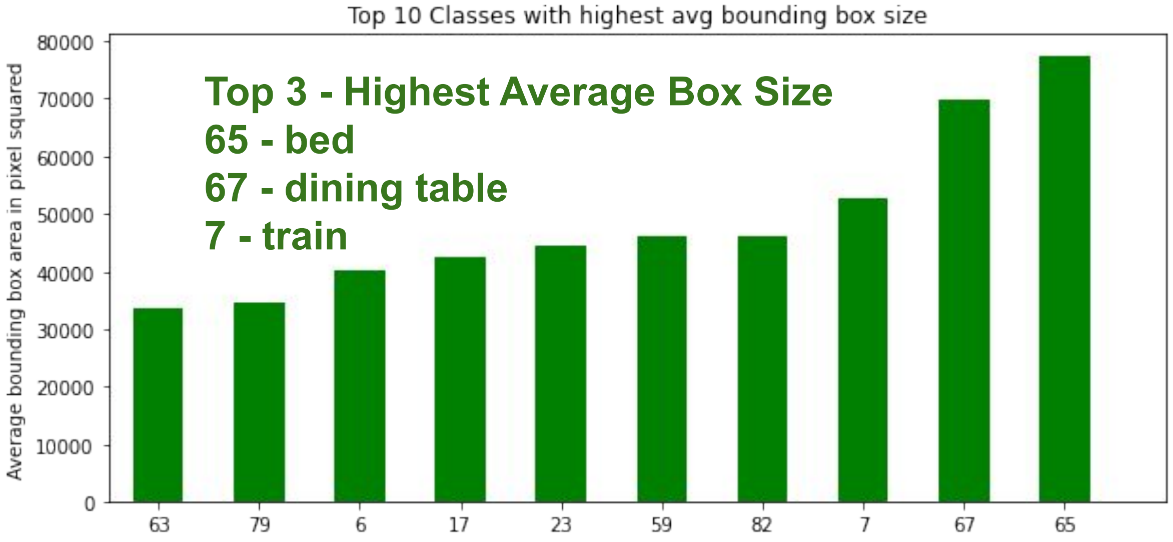


The distribution of the number of words in the captions is slightly right skewed but in general centered around 10

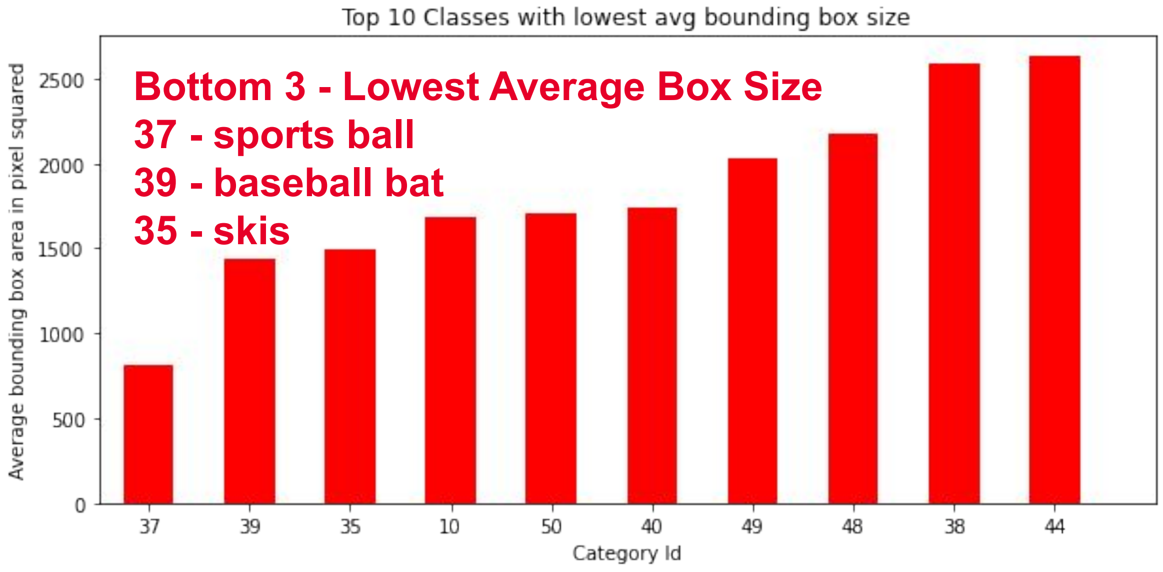
EDA Results



Top 3 - Classes with the Most Observations (# boxes)
1 - person
3 - car
62 - chair



Top 3 - Highest Average Box Size
65 - bed
67 - dining table
7 - train



Bottom 3 - Lowest Average Box Size
37 - sports ball
39 - baseball bat
35 - skis

Models

In terms of deep learning models, both computer vision and natural language processing models will be used.

- **Computer Vision:** Pre-trained, frozen CNN architectures (e.g. VGG, Inception) will be used to extract features from the images. (Note: we decided to use frozen encoders trained on ImageNet because we ran a trial without freezing layers and observed a lot of over-fitting.)
- **Language:** RNN and LSTM structures
- **Attention:** As described in the paper, "Show, Attend, and Tell", models that incorporate attention to the image feature map perform, have improved performance. We will use the attention mechanism described in this paper in one of our models.

Models

We have trained the following models:

1. **Inception-GRU:**
   - Extract feature map from the last CNN layer of frozen InceptionV3 with ImageNet weights
   - Attend to feature map
   - Decoder with an embedding layer (not pre-trained), a GRU layer and 2 fully-connected layers
2. **Inception-LSTM:**
   - Extract feature map from the last CNN layer of frozen InceptionV3 with ImageNet weights
   - Attend to feature map
   - Decoder with an embedding_layer(not pre-trained), a LSTM layer and 1 fully-connected layer
3. **VGG-LSTM:**
   - Extract feature map from the last CNN layer of frozen VGG16 with ImageNet weights
   - Language feature extractor with an embedding_layer(not pre-trained), a dropout layer, and an LSTM layer. (This extracts features from the previous caption words to predict the next word)
   - Combine the output of the language and image feature extractors and feed it into a fully-connected layer to predict the next word of the caption

App Directory Structure

- a-eye-app
    - api-service
    - frontend-react
    - deployment
    - frontend-simple (not kept up-to-date)
    - persistent-folder
    - secrets

Deployment

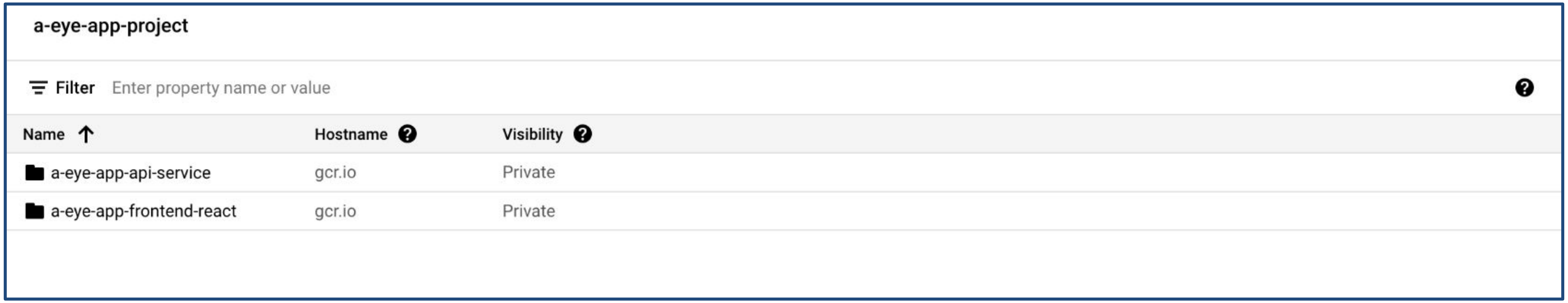
We used Ansible scripts to deploy our app on GCP and Kubernetes.

- Deployment to GCP
    - http://35.222.164.164/

- Deployment to K8s Cluster
    - http://35.202.124.222.sslip.io/

Note: we have shut them down for now to save GCP credit.

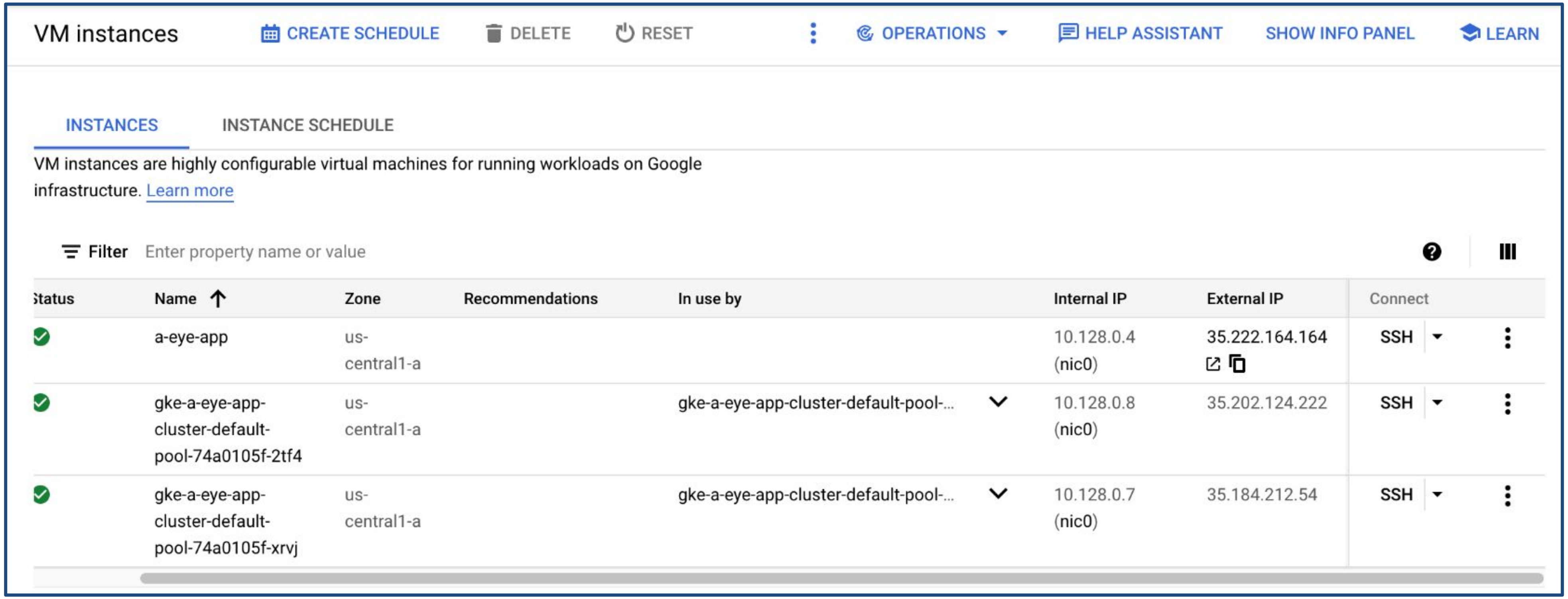
Deployment: GCP



a-eye-app-project

Filter Enter property name or value

Name ↑	Hostname	Visibility
a-eye-app-api-service	gcr.io	Private
a-eye-app-frontend-react	gcr.io	Private

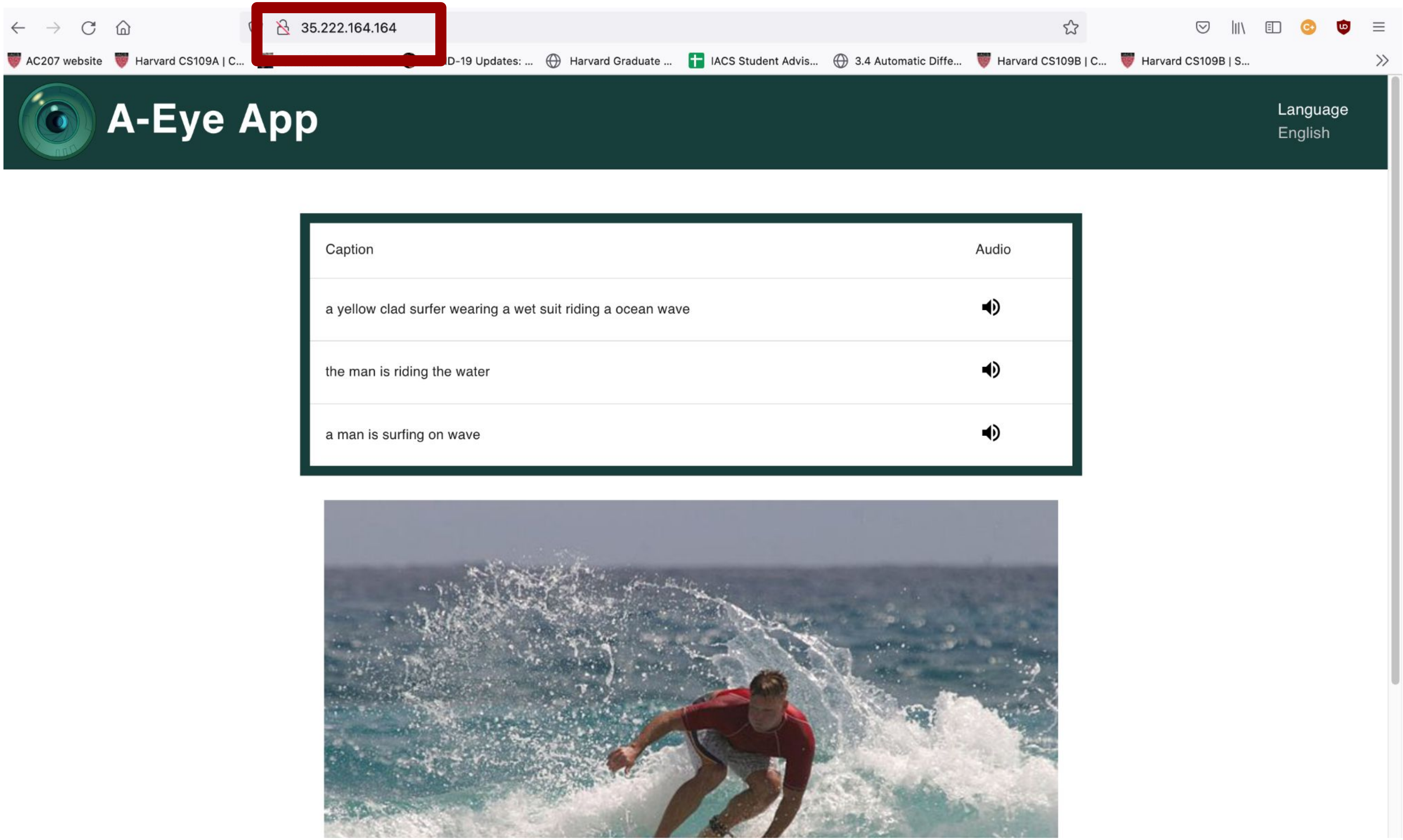


VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

Filter Enter property name or value

Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	a-eye-app	us-central1-a			10.128.0.4 (nic0)	35.222.164.164	SSH
✓	gke-a-eye-app-cluster-default-pool-74a0105f-2tf4	us-central1-a		gke-a-eye-app-cluster-default-pool-...	10.128.0.8 (nic0)	35.202.124.222	SSH
✓	gke-a-eye-app-cluster-default-pool-74a0105f-xrvj	us-central1-a		gke-a-eye-app-cluster-default-pool-...	10.128.0.7 (nic0)	35.184.212.54	SSH

Deployment: GCP

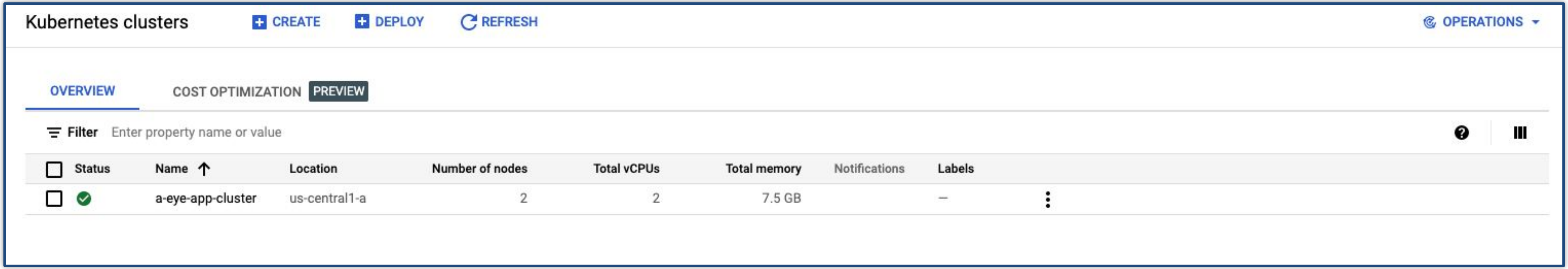


A-Eye App

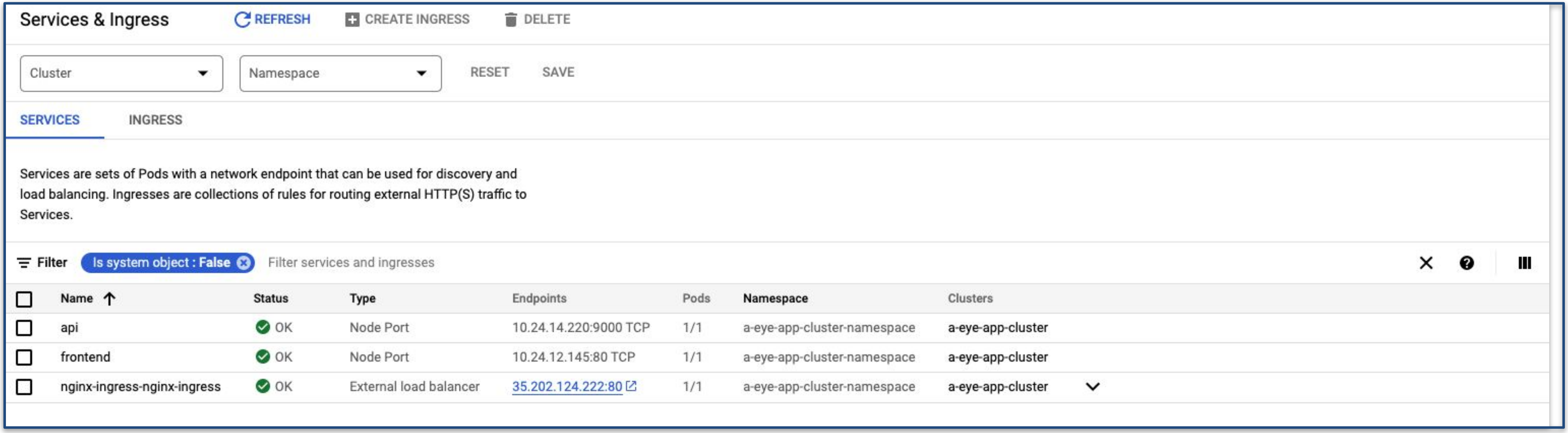
Language
English

Caption	Audio
a yellow clad surfer wearing a wet suit riding a ocean wave	🔊
the man is riding the water	🔊
a man is surfing on wave	🔊

Deployment: Kubernetes



☐	Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
☐	✓	a-eye-app-cluster	us-central1-a	2	2	7.5 GB		–

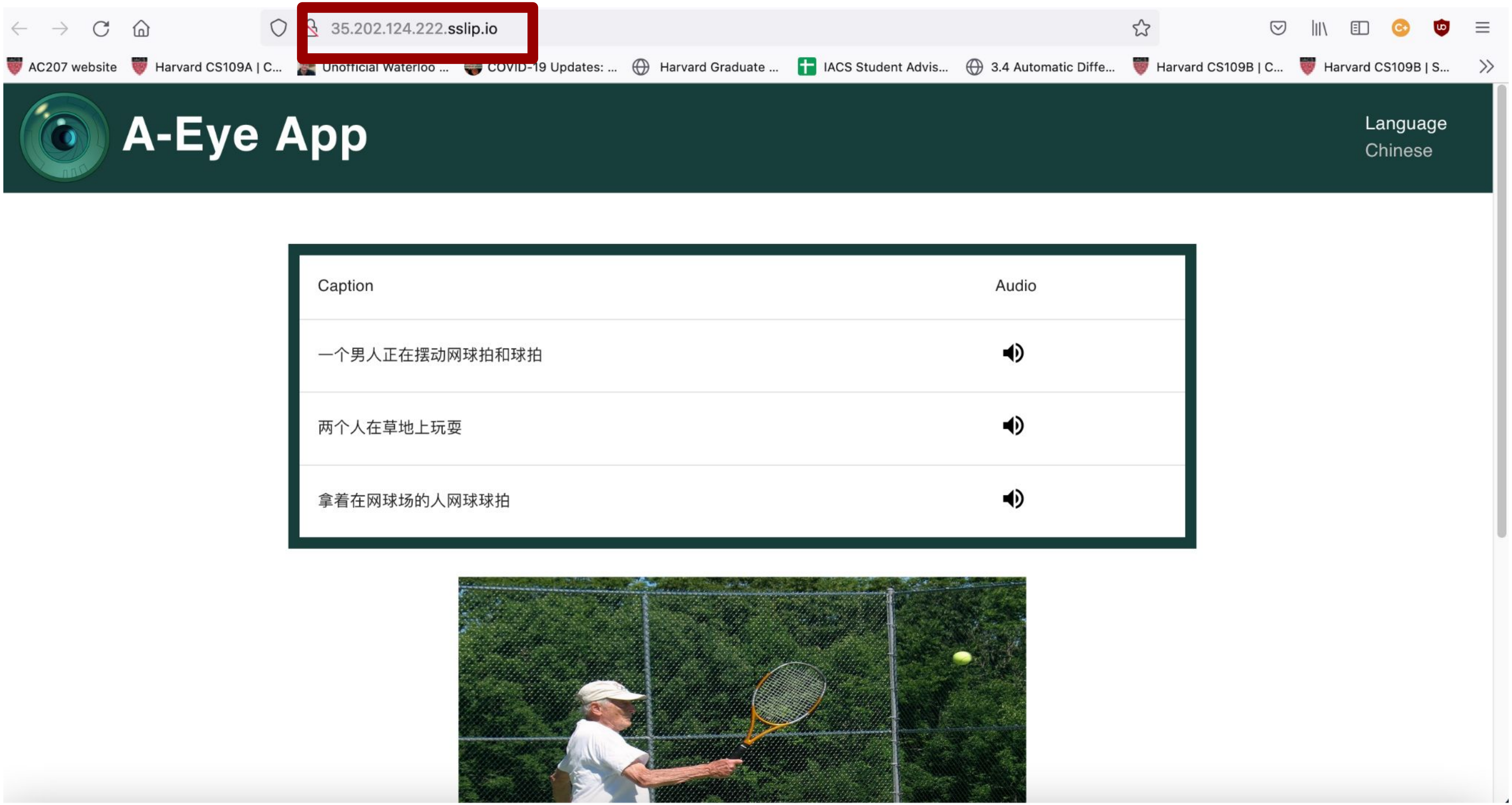


Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

Filter: Is system object : False

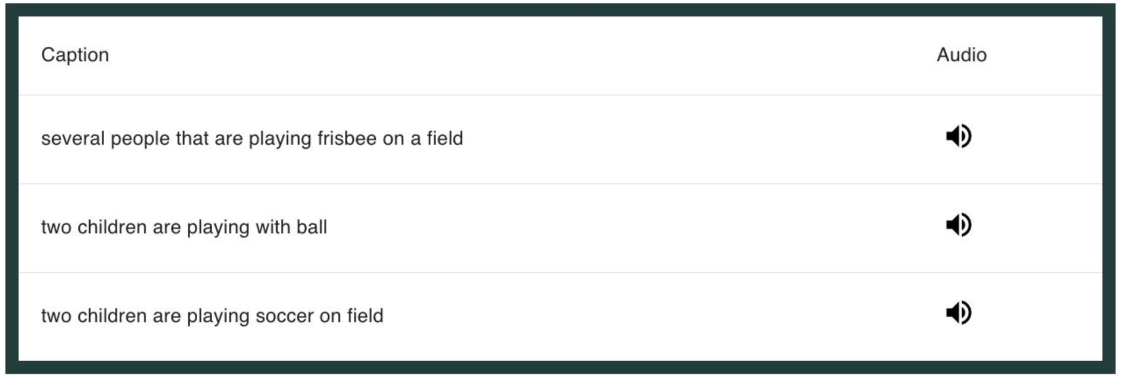
☐	Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
☐	api	✓ OK	Node Port	10.24.14.220:9000 TCP	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster
☐	frontend	✓ OK	Node Port	10.24.12.145:80 TCP	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster
☐	nginx-ingress-nginx-ingress	✓ OK	External load balancer	35.202.124.222:80	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster

Deployment: Kubernetes

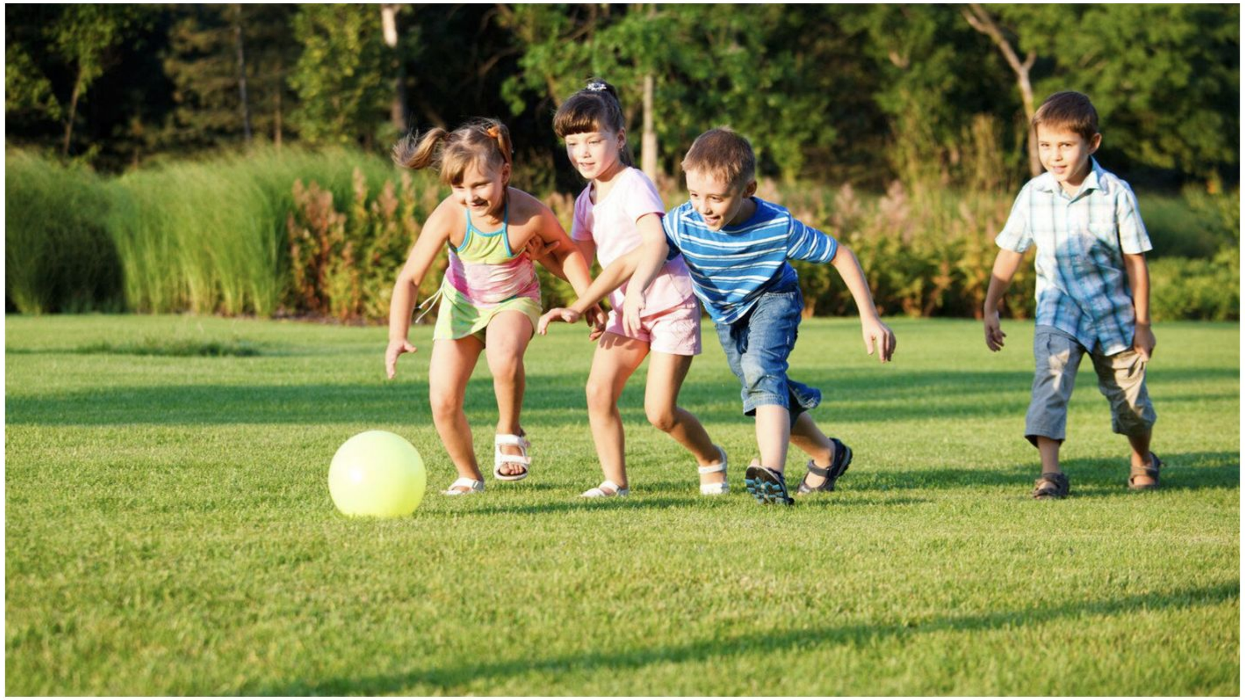


Caption	Audio
一个男人正在摆动网球拍和球拍	🔊
两个人在草地上玩耍	🔊
拿着在网球场的人网球球拍	🔊

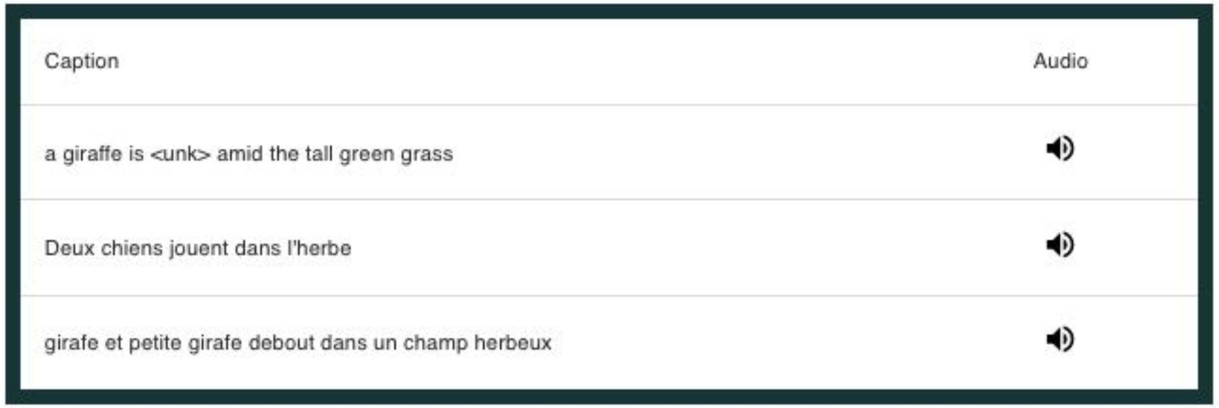
App Demo - English



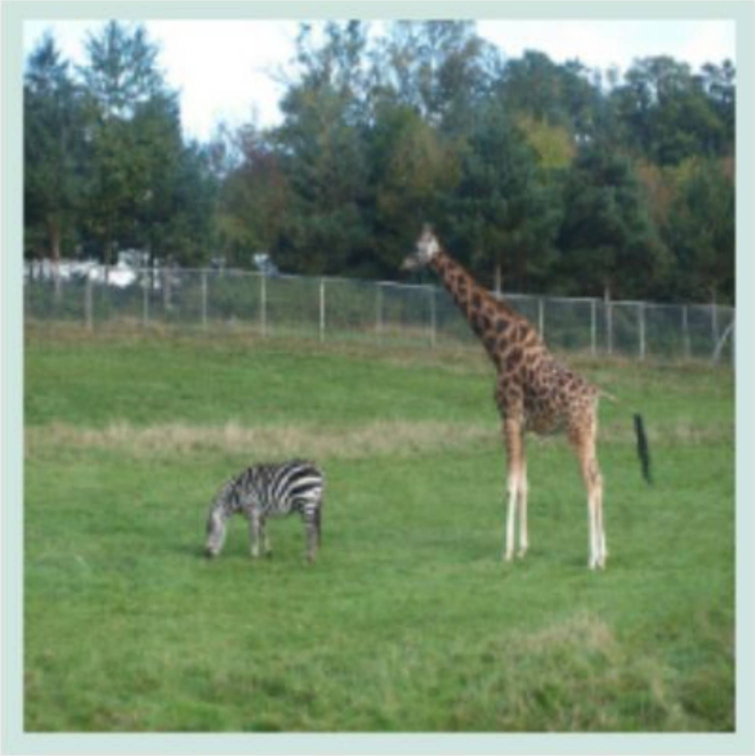
Caption	Audio
several people that are playing frisbee on a field	🔊
two children are playing with ball	🔊
two children are playing soccer on field	🔊



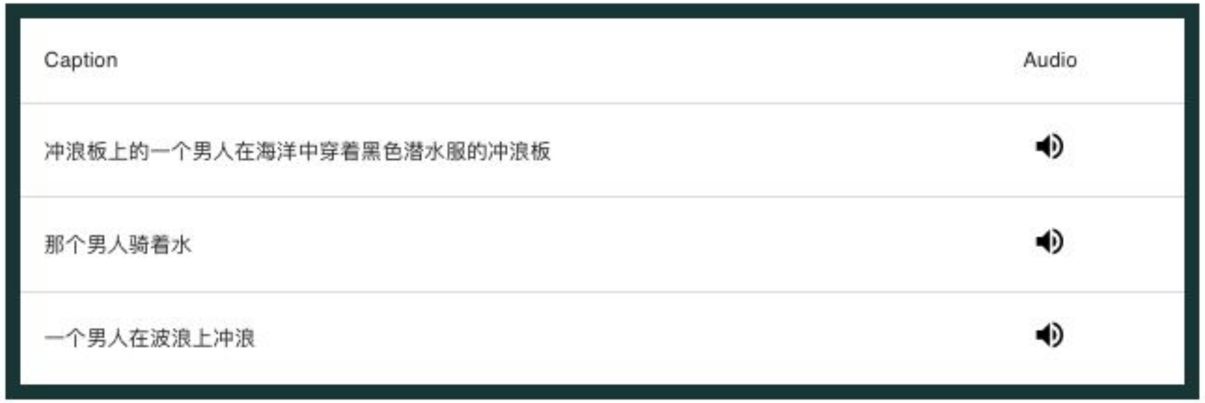
App Demo - French



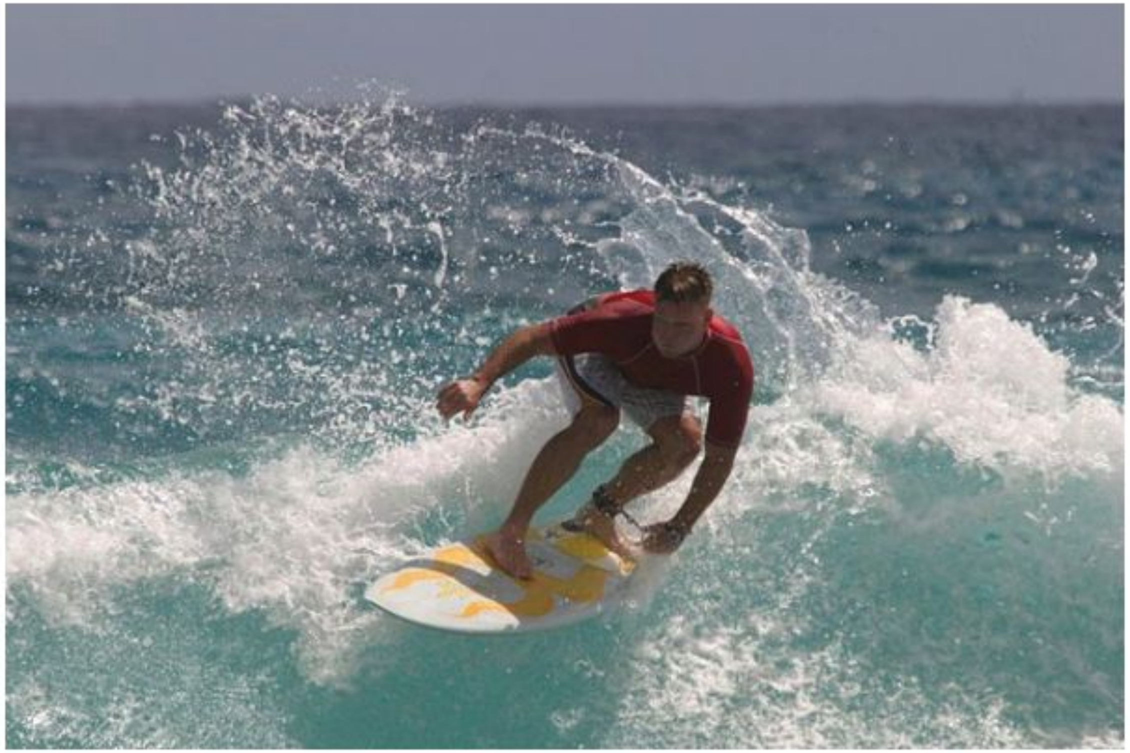
Caption	Audio
a giraffe is <unk> amid the tall green grass	🔊
Deux chiens jouent dans l'herbe	🔊
girafe et petite girafe debout dans un champ herbeux	🔊



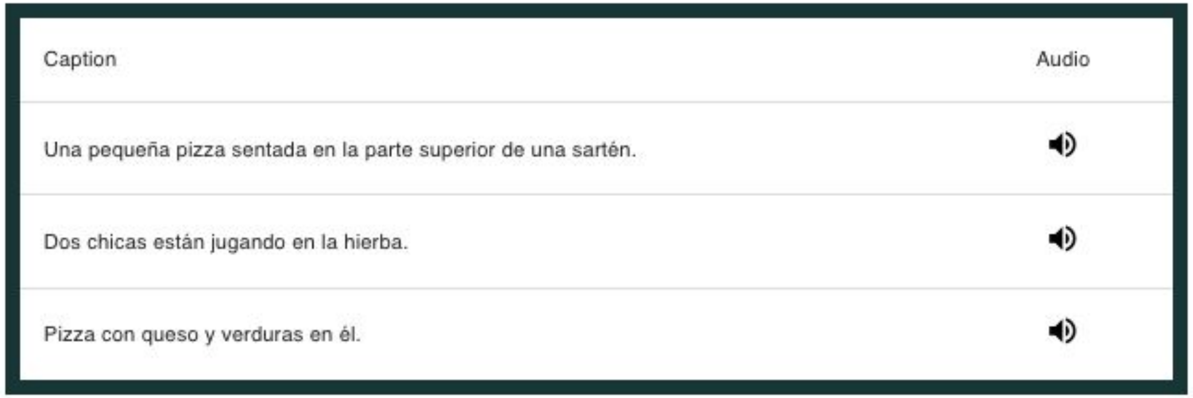
App Demo - Chinese



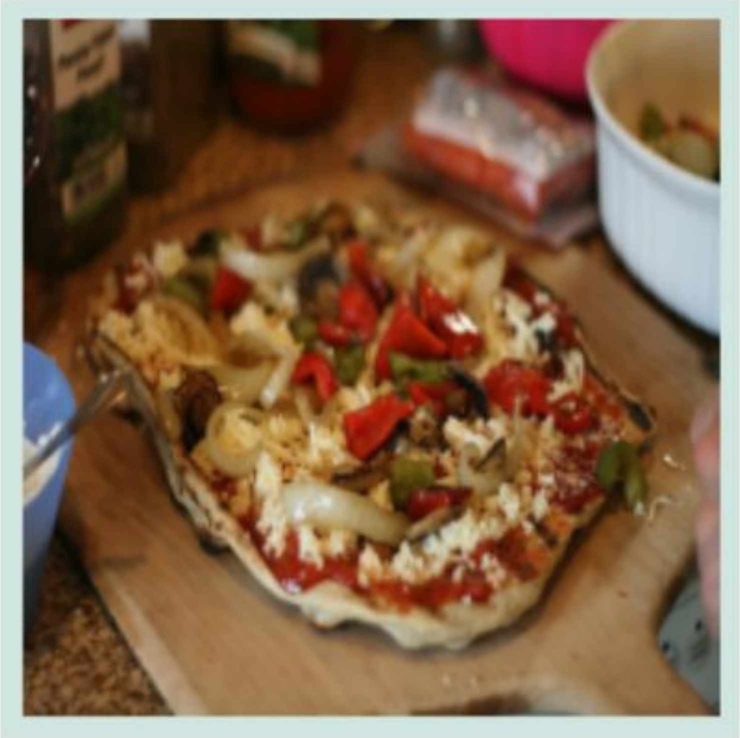
Caption	Audio
冲浪板上的一个男人在海洋中穿着黑色潜水服的冲浪板	🔊
那个男人骑着水	🔊
一个男人在波浪上冲浪	🔊



App Demo - Spanish



Caption	Audio
Una pequeña pizza sentada en la parte superior de una sartén.	🔊
Dos chicas están jugando en la hierba.	🔊
Pizza con queso y verduras en él.	🔊



Todos

- **Optimization:** Currently, the process of creating model predictions and audio files is a bit slow. We want to add a progress bar on the UI side for better user interaction and run the models in parallel to speed things up.
- **UI Design:** Make UI more aesthetically pleasing
- **Translation Bugs:** Google Translate API does not translate captions that have an <unk> token. Fix this bug.
- **Language Setting:** Since we don’t have user information, if two people use our website at the same time, the same language setting will be used for both. We want to change this to have different settings for different users.