Uber ATC Homework Assignment

Below are two tasks that you can choose to do for your homework assignment. **You only need to do one of these tasks!** We expect you to spend about eight to ten hours doing this task, and don't expect you to do all of the extra "bonus points" features. It is better to focus on doing fewer features very well vs. doing many features.

General guidelines:

- You are allowed to ask questions to clarify the assignment.
- What you deliver should be a working product and have detailed documentation for how to build and run the application. We will be using your tool as part of the evaluation process! In your documentation please explain any major design decisions you've made and why you made those decisions. These can be technical decisions (a choice of a particular library or framework), as well as any API structure or user experience choices.
- The assignment may not be fully specified. If making a simplifying assumption would help you to complete the assignment, you may make it – just describe the assumptions you made in the documentation you submit with your code.
- Allocate time to thinking through the whole solution so that it is easily extensible when you want to implement extra features.
- Usability is key to the work we do. We will be critically looking at some of the user interface design decisions you make.
- You can use any resources, frameworks, or libraries that you want so long as you document which parts you borrowed.

Task Choice 1: Building an Image Labeler

We want you to make a web interface for labeling objects in images. For this task it is sufficient to design a tool with which users can outline objects using a bounding box.

A minimal implementation will provide:

- Ability to load and display an arbitrary image for labeling (can be configured as a variable somewhere in the code, or extracted from a URL)
- A tool for marking bounding boxes around pedestrians in an image
- Instructions for how to build and run the application

- Instructions for how to use the interface. Having a simple help page is encouraged.
- A mock 'save' button that will output a list of all labeled objects and their coordinates. We will define the top-left corner of an image is (0,0) (this is consistent with many computer graphics and image processing conventions). You should also provide a function that would POST or PUT this data to a server and specify the format of this data. You do not need to implement any back-end system to receive this request.

Here's an example image that we would like you to use when designing and implementing the labeler. If you get to any of the bonus items, feel free to choose other images and include them with your submission.
https://upload.wikimedia.org/wikipedia/commons/8/83/King's_Cross_St_Pancras_underground_station_entrance_-_IMG_0746.JPG

Bonus points:

- A polygon tool with which we can produce detailed outlines instead of bounding boxes
- Statistics about how long the task takes (per-label and per-image timing)
- Ability to edit existing labels
- Different classes of objects: pedestrians, vehicles, animals, signs (stop sign, parking sign, etc.), motorcycles, etc.
- A design that lends itself to the addition of new tasks, such as providing additional details per label (type of vehicle, stance of the person, etc.)
- The ability to load and save labels for multiple images.
  Note: without writing a back-end service you can use cookies or local storage to save intermediate labels.

We will be looking for:

- Use of best-in-class frameworks. We want to move quickly and use frameworks and libraries that are maintained and widely used as much as possible.
- Readable, modular, and tested code
- User interface decisions- is the tool easy to understand? Can someone quickly perform the task at hand?

Task Choice 2: Uber Trip Search Engine

As you probably know, people take trips with Uber. Your task is to create a simple search engine with which we can look up trips by geolocation, driver, rider, or time of day. You will create a backend and frontend for this search engine such that we can run your solution with our own trip data for evaluation.

Each trip is a dictionary with the following keys:
driver_id: The driver's user ID
driver_name: The driver's name
passenger_id: The passenger's user ID
passenger_name: The passenger's name
path: A list of (latitude, longitude, unix timestamp) triples, ordered by timestamp, that describes the route the driver took.

When the server starts we would like you to read in this data as a JSON blob. The server should accept some argument on the command line which is the filename containing the data. The contents of this file will be a JSON serialized list of trip dictionaries. That means that if you are using Python, you should simply do a 'json.loads(file_contents)'.

A minimal solution will include:

- A backend that reads in the trip data as described above and exposes access to this data through a simple API that you will design and document.
- A frontend that implements the following functionality:
  1) Display a list of trips taken in a table with the driver, passenger, start time, and trip duration.
  2) A search API and UI that allows for searching trips by the driver, passenger, and time of day
  3) A geo-search API that returns trips that pass within a given geo-rectangle. You only need to deal with the case in which a path point is inside the rectangle; you don't need to interpolate between points.

Bonus points:

- Display the trip on a map
- Add the ability to tag trips with arbitrary key/value pairs. For example, we may be interested in the weather at the time of the trip
- Generate statistics for the system. Some questions we might want to answer include: What's the most popular time of day to take trips? Which drivers have given the most number of rides? Which passenger has spent the most amount of time in an Uber?