Frontend Take Home Exercise

Background

enview.ai is building a driver monitoring solution. Our customer has a fleet of delivery vehicles. They have had problems where drivers drive rashly or use their phones while driving, leading to accidents.

We have built a model that detects this behavior. Further, we have a backend service that exposes the inferences of the model with APIs.

Vehicles

A vehicle is identified by a unique ID that is generated by us when we add it to our database. We also store the number plate of the vehicle.

Our backend service has a REST endpoint **GET /vehicles** that returns a list of vehicles. Here is a sample response:

```
[ { "friendly_name": "KA12A3456", "id": "dd70a7e5-8397-4914-bbbb-4d6bb521ec67", }, {
"friendly_name": "MH12A3456", "id": "cc70a7e5-8397-4914-bbbb-4d6bb521ec67", } ]
```

Alerts

- An **alert** is when the model detects risky driver actions.
- Alerts have a **type**, which is a string field indicating the reason for the alert, eg. “Unsafe driving” or “Distracted driver”.
- Alerts are associated with a **driver** and a **vehicle.**
- In summary an alert is when a **driver** driving a **vehicle** shows some **type** of risky action.

Our backend service has a REST endpoint **GET /alerts** that returns a list of alerts. Here is a sample response:

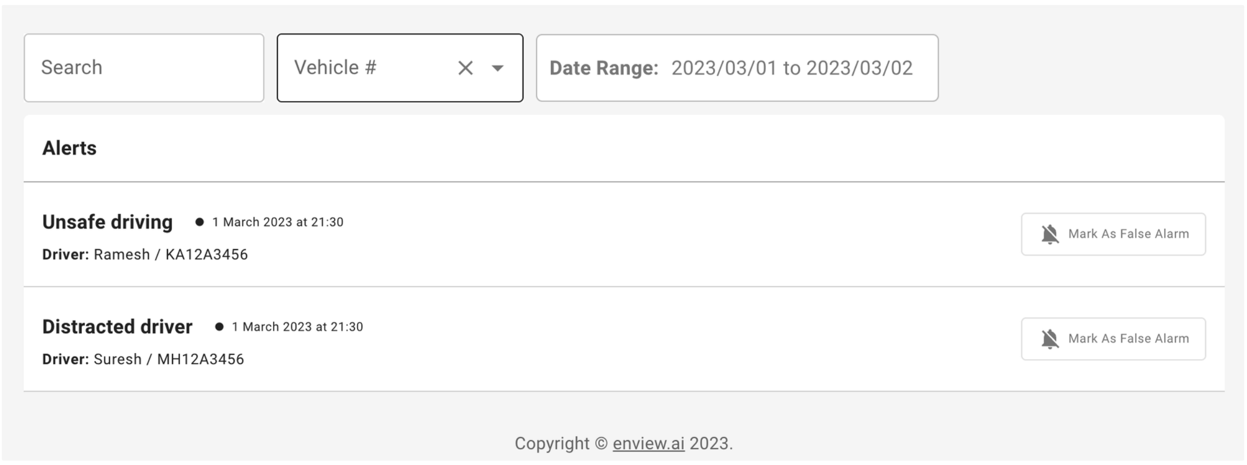
```
[ { "id": "6049dbd2-45bc-4e34-9ea2-c82ced0279f1", "alert_type": "Unsafe driving",
"vehicle_id": "cc70a7e5-8397-4914-bbbb-4d6bb521ec67", "driver_friendly_name":
"Ramesh", "vehicle_friendly_name": "KA12A3456", "timestamp": "2023-03-
01T04:25:45.424Z" }, { "id": "5149dbd2-45bc-4e34-9ea2-c82ced0279f1", "alert_type":
"Distracted driver", "vehicle_id": "dd70a7e5-8397-4914-bbbb-4d6bb521ec67",
"driver_friendly_name": "Suresh", "vehicle_friendly_name": "MH12A3456", "timestamp":
"2023-03-01T04:24:45.424Z" }, ]
```

For the purposes of this exercise, **you may mock these endpoints in any manner you choose.** You will not be judged on how you call the APIs- it can be as simple as reading JSON blobs in text files.

The Task

Your task is to develop an interactive, single page application where our customer can view what our model has detected.

The page should look like this:



Unsafe driving ● 1 March 2023 at 21:30
Driver: Ramesh / KA12A3456

Distracted driver ● 1 March 2023 at 21:30
Driver: Suresh / MH12A3456

Copyright © enview.ai 2023.

The page should have the following functionality:

1. Alerts must be displayed in the layout shown above.

2. The user can search for alerts through three interfaces:
    a. Free text search: When the user enters a string here, the app searches all the text in the alerts for that string. If there is a match, the alert is displayed.
    For example, if the user enters “Unsafe”, only the unsafe driving alert should be visible.
    If the user enters “Suresh”, only the second alert should be visible.
    If the user enters “Driv”, both alerts should be visible as “Driv” matches both “Unsafe **driv**ing” and “Distracted **driv**er”.
    b. Search by vehicle number: This should only search the vehicle_friendly_name field in the alerts.
    c. Search by date range: This should show the alerts whose timestamp falls in between the two dates.
3. The user can mark an alert as a false alarm, ie, the model’s inference is a false positive.

Bonus Points

1. In the search by vehicle box, show a dropdown of current vehicles (using GET /vehicles) in addition to searching by text.
2. Display timestamp and date range in the user’s time zone, but call the backend using UTC timestamps.

Evaluation Criteria

- First and foremost, your code will be evaluated for readability.
- Code must be working. We should be able to follow the instructions in your README and run the code locally on our machines.
    - If you aren’t able to complete all features, a partial working solution is better than a non-working solution.
- User friendliness of the UI