Frontend Take Home Exercise

Simulation Log Viewer

nuTonomy runs simulations of different car builds in different scenarios. Even if the car build and scenario match, the results can differ. Your task is to create a web interface that presents the scenario logs and performs basic operations.

Requirements

Every scenario has an id and a maximum number of "stops" (times when the car unexpectedly braked).

Every simulation run has a start time, end time, car build and a corresponding scenario. A run is considered successful if

- It does not exceed the maximum number of stops for the scenario
- It does not exceed the maximum running time for the scenario. (Calculated by `<end_time> - <start_time>`)
- It does not have a collision

A set of logs has the following format:

```
{
  "simulationRuns": [{
    "startTime": number (UTC timestamp),
    "endTime": number (UTC timestamp),
    "scenarioId": string,
    "carBuild": string,
    "result": {
      "numberOfStops": number,
      "hasCollision": boolean,
    }
  }],
  "scenarios": [{
    "scenarioId": string,
    "maxNumberOfStops": number,
    "maxRunningTime": number,
  }]
}
```

Interface

The data should be presented as a table of simulation runs with the following format:

| scenarioId | carBuild | startTime | runningTime / maxRunningTime | numberOfStops / maxNumberOfStops | hasCollision | doesScenarioPass |

Each run should be sortable on every column e.g. ascending order of `startTime` or descending `numberOfStops`.

The data should be filterable by

- `scenarioId`
- `carBuild`
- `scenarioId` and `carBuild`

Selecting any filter should show the relevant runs and present the following statistics

- percentage of runs that exceed the maximum number of stops
- percentage of runs that exceed the maximum running time
- percentage of runs that have a collision
- percentage of runs that do not pass

Technical requirements

- Project should be installable and runnable with npm i.e. `npm install && npm start`
- Project should be testable with `npm test`
- Project must be written using a modern JavaScript framework e.g. Angular, React, etc
- The interface should be clean. Advanced HTML/CSS is not a priority