The test application itself is in Ruby on Rails as I enjoy working with that (and I had some ideas to extend the app further using it), but that's not at all a requirement. I developed the frontend/visualisation JS code in plain JS files, and just served it via Rails for the actual user tests. The backend could easily be just a PHP file that accepts POST requests and saves records to a database of your choice.

Here's an overview of relevant files in the Rails app. Most of the other files are the default auto-generated Rails files and can be ignored.

**app/assets/stylesheets/application.css** – All styles

**app/controllers/answers\_controller.rb** – Simple controller that handles the POST request to /answers, and saves an answer instance to the database

**app/views/application/index.html.erb** – Initial page shown when starting the app

**app/views/application/[bar,box,eid]chart.html.erb** – Files for showing the respective charts; they just set template values, used by the file below

**app/views/layouts/application.html.erb** – Shared layout which the [bar,box,eid]chart.html.erb files above are rendered into

**config/database.yml** – Database settings

**config/routes.rb** – Maps URLs to controllers

**db/schema.rb** – Auto-generated database schema file

**public/color\_images/** – Coloured versions of images (unused)

**public/data/** – Place generated JSON files here, and choose the one you want to use by setting `dataSource` in `generic.js`

**public/generate\_data.html** – Page used for generating data sets.

**public/images/** – Images used in the tests

**public/info/** – Content for each of the information pages in the tests

**public/javascripts/generate\_data.js** – Used to generate a new JSON dataset, by altering the data definition

**public/javascripts/generic.js** – Main frontend application code, used by all chart types

**public/javascripts/{bar,box,eid}chart.js** – Code specific to each chart type

* **bars.enter().append('rect').attrs()** – Where color is given to the bars of each graph
* **function(d) { return color(d.key)** – Inherited from generic.js
* **class: "bar"** – Inherited from applications.css, overwrites **function(d) { return color(d.key)**

The visualisations are done with D3.js. The code which I inherited for this project for rendering the specific charts was using D3 v3, which I continued with. After finding some problems potentially due to using D3 v3 (e.g. a memory leak if certain pages are left open for long durations), I experimented with migrating the app to D3 v4, but didn't have enough time to complete that. However, I've left some commented out lines of JS ending with `// D3 V4`, so if you want to implement your own application with D3 v4, those will be some good starting points and suggestions (although not tested or finished).

To run the app locally:

- Install PostgreSQL

- Modify the `database`, `username`, `password`, `host`, and `port` settings in the `development` section of `database.yml`, as necessary

- Ensure you have a recent Ruby version installed

- Open a terminal in the directory of the application

- Run command `gem install rails`

- Run command `bundle install`

- Run command `rails db:setup`

- Run command `rails server`

- Visit <http://localhost:3000>