Architecture

This application is built using the *distributed microservices architecture* with Docker and deployed with Docker Cloud on Google Compute Engine.
The online deployment is accessible in two ways:
with the external ip: http://104.196.170.159/.
Docker Cloud's service endpoint exports: http://nginx.mapcch.98a2de36.svc.dockerapp.io

Services are defined in `docker-compose.yml` as:

web: a nodejs + express web application
redis: a redis server with data persistence
nginx: web server and proxy

Why the setup?

Microservices architecture highly increases the scalability of the application, meaning not only additional services (for example to handle the authentications, etc.) but also makes the application reusable (for example the form could be used as part of other projects).

Logic

I implemented the required functionality of checking if an email address exists simply with using redis `set` type/object. When submitting the form (after validation), an ajax call is made to the application server, passing only the email as the parameter. Expressjs then gets the email, and uses the `sadd` method to write the value into a redis set. A `set` cannot hold duplicate records, thus the redis server responds with a 0, or 1 indicating if the save to the set was successful or not, and app server responds to the client with this status, where 0 means record already exists, and 1 meaning record is now added. The client then handles this by showing a modal/prompt the user. Please note that using redis in this implementation is considerably faster than implementing an object model database like postgres, saving the email+records, and querying the responds to get to the same result. In practice, this could be used to cash the users data, in addition to a postgres database (or any other type of databases).

Breakdown of the time spent

I used my personal template, which is a stack using Nodejs + Semantic-ui. The template is available here: https://github.com/arminakvn/appntxt.git I also re used my recent project's docker-compose setup with nginx and postgres (here: https://github.com/civic-data-design-lab/atlas-lighting.git) with only replacing postgres with redis. Therefore the initial setup was very fast, and most of the time was spent documenting, design/aesthetic fixed, and deployment. The breakdown of the time spent is as follows:

- initial setup from the template and making the initial git push: 30 minutes.
- setting up a working form from a semantic-ui example/tutorial: 30 minutes.
- setting up docker-compose using my code from previous projects, edits/config, switching from postgres to redis and testing: 60 minutes.
- implementing the challenge requirements including the form validations, using semantic-ui api and the logic for using redis database: 60 minutes.
- documenting, cleaning the code, making setup instructions, some inline commentings in the code for better readability: 3 hours.