Chan Kok Jing, Daryl
A0183998B
31 December 2018

CVWO AY2018/19
Mid-assignment Write-up
Task Tracker

1 Main Idea

The goal is to build a simple and highly-performant task tracker with a focus on user experience. The task tracker should allow the user perform basic CRUD operations such as creating, viewing, editing and deleting tasks. It should also allow the user to categorize and filter tasks using tags.

2 Architecture

2.1 Back-end

The back-end of the application should serve two main purposes – a RESTful API server, and to serve the front-end.

It will be built using Ruby on Rails, with PostgreSQL as the database. A test suite will also be built using RSpec, Factory Bot, Faker, Database Cleaner and Simplecov.

2.2 Front-end

The front-end of the application should allow the user to interact with all of the task tracker's features in a fuss-free and aesthetically-pleasing manner.

It will be built using React, with Bulma as the CSS framework.

2.3 Process

The application will be built using a test-driven development process. This consists of the standard red-green-refactor loop. Continuous integration will be provided by CircleCI and continuous deployment will be provided by Heroku.

For the back-end, models and controllers should have full test coverage. For the front-end, React components should also preferably have complete test coverage if time permits.

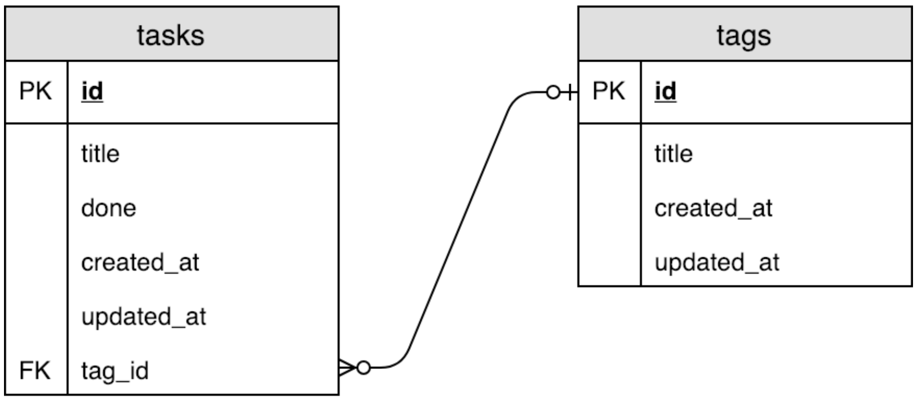
Each time code is pushed to GitHub, the test suite should be automatically run in an independent environment provided by CircleCI. If all tests pass successfully, Heroku should automatically pull the latest commit from GitHub, build it and deploy it to production.

3 Data

The application should have two models – Task and Tag.

A task must have a title, a Boolean indicating whether or not the task is done, and may belong to a tag.

A tag must have a title, and may have many tasks. When a tag is deleted, the associated tasks should not be deleted. Instead, the associated tasks' `tag_id` should be set to `null`, as if the tasks do not belong to a tag.



4 Current Progress

4.1 Back-end

The back-end is fully built, complete with full test coverage.

4.2 Front-end

The front-end is about 80% complete at the current point.

Completed –
- Basic structure
- CRUD for tasks
- Filtering tasks using a keyword

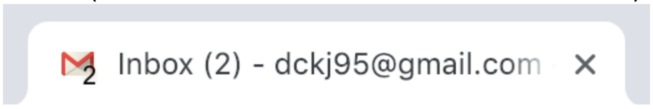
Incomplete –
- CRUD for tags
- Filtering tasks using a tag
- Improvements to UI / UX

5 Future Plans

5.1 Additional Features

The following is a list of additional features that may be added to the application if time permits.

- Inline creation / modification of tasks instead of using a modal
- Split complete and incomplete tasks into separate lists
- Dynamic modification of `<title>` and favicon to reflect number of incomplete tasks (similar to what can be found on Gmail)



Inbox (2) - dckj95@gmail.com

5.2 Challenges and Remarks

Even though I've had prior experience with both Rails and React, this was my first time putting them both together in a single application. I've always built the back-end and front-end as independent applications so it was an interesting experience integrating them both together.

Learning how to use the `react-rails` and `webpacker` was a challenge, especially since I've had a pre-conceived notion of how React is supposed to work (no `ReactDOM` but the `react_component` view helper instead, etc.).

Going forward, I think Heroku deployment is going to be interesting, for the same reasons as above. I'm also quite interested in writing tests for the React components but have no experience with that. If time *really* permits, I might also look into using React `v16.7.0-alpha` to try out the new Hooks API since I've been wanting to do that for a while now.