CVWO Assignment 3: Final Submission

Shen Yichen (A0091173J)

In this assignment I created a basic TODO app, which allows lets the user manage their TODO tasks. The app supports multiple sets of TODO lists, allows labeling of TODOs, as well as support deadlines and marking TODOs as done without deleting them.

User Manual

List

The landing page of the app will show 2 lists of TODOs, pending and completed (completed is collapsed by default). Each TODO in the pending list will appear as either red, orange, or green, if they are overdue, due today, or not due and without due dates respectively. If a TODO has a due date, it will be displayed accordingly. Completed TODOs are always grey. Clicking on the header will cause the TODO to expand, revealing more information, such as the extra description, as well as the labels that were associated with the TODO.

TODOs can be added with a form via the button on the top right. The icons on the right of the header allows the user to edit, delete, or mark the TODO as done. When a TODO is marked as done, it is moved to the completed section.

Searching

A user can search for TODOs via the search bar on the top of the list. The user can enter the search term and press enter. Results will then be filtered to TODOs that contain the search term in the title or in the extra description. Clearing the search bar restores the full list.

Adding and Editing

Adding or editing for a TODO is done via a form. In the form, the user must specify a title for the TODO. On top of that, he may select a due date via a date picker, give an extra description, as well as assign the TODO a label.

Labels

Labels can be accessed from the sidebar, which will expand upon clicking on the icon on the top left. Labels will be listed in the sidebar, together with a "Manage Labels" entry that brings the user to the label admin page. This page lists out the labels in a collection, and the user can add/edit/delete them in much the same way as TODOs. Clicking on the labels themselves in the sidebar brings the user to a list of TODOs filtered by that label.

Sessions

Each new user to the app will be assigned a new TODO list. The ID of his list will be saved in his cookies and the TODO list will restore itself to the state he left it at when he revisits the site sometime later. Each separate TODO list can be accessed with its own URL, specified at the bottom of the page. A user may visit another TODO list, then click on a link near the save URL field to set it as his default, allowing him to be directed to that list instead when he arrives at the root URL. He can still access his original default via URL.

Sessions that have not been accessed for over a month will be deleted.

Accomplishments

This particular assignment was both a revision and an exploration for me in a way. I do have some previous experience with Rails, thus, relearning and expanding upon that knowledge was much less painful than when I first started out with it. I started by visualizing how to and later attempting to satisfy the basic requirement of a simple TODO list with a association to labels. The process allowed me to refresh my understanding of Rails considerably.

During the mid assignment writeup, I laid out a basic plan that included deadlines and the ability to mark TODOs as done. These additional criteria eventually guided me to use more aspects of the functionality set provided by Rails as compared to the basic requirements. For example, for marking a TODO as done, I used a custom route, and ended up learning a little more about routing in Rails.

On top of the basic plan, I also laid out 3 possible expansions: a mechanism to allow each user to have their own TODO list, front-end with react.js, and deployment on a server from scratch. I managed to complete the first and the third expansion.

Sessions

I implemented a mechanism that assigned each new user a ID to be stored in their cookie store. A corresponding entry is then created in the database, which will serve as the parent of all TODOs and labels the user create. As such, the whole list itself is accessible solely from the URL path of the session model. The advantage is that a user can easily access his TODO list anywhere and share it by simply saving the URL. The disadvantage is that there is no access control, and anyone with the URL can also access the same TODO list.

I'm actually not sure if this method is even the recommended way to accomplish a session based app. For example, the database can be bloated with old application sessions that were long abandoned. To alleviate the issue, I created a rake task, activated by cron once a week to clear out sessions that have not been accessed for a month. I read up on cron, rake, as well as how to write the callback to update the accessed time correctly to accomplish this addition. An improvement that could be made would be to find a way to resolve the problem of a malicious attacker creating a large number of useless sessions, which is not tackled in my implementation yet.

Server

I had been curious about how a Rails app is actually deployed ever since I started with it. For this assignment, I decided to explore the option of deploying my app from scratch, on my own server.

Currently, I have the app deployed on my old laptop, running Fedora 23 Server Edition, using Phusion Passenger. It's connected to the NUS network, and is accessible at `http://ycholocron.ddns.net/chimney`. A cron task is set on the server to pull in changes from Github, and update the Rails app accordingly.

Playing with the server has allow me to appreciate the details needed for an production server, instead of the simple `bin/rails server` used in development environments. In particular, bundler gave me some issues, as the Gemfile is locked in a production environment. Furthermore, SELinux on Fedora also interfered with some of the native gem libraries, until I configured it properly. Both of the issues taught me that just because the whole app is running fine on the production machine does not mean it will run without errors during actual deployment. Thus it would be wise to plan for some time that will be devoted to deployment itself.