CVWO Mid-Assignment Submission

Chaitanya Baranwal (A0184716X)

December 31, 2018

1 Key Details about the App

1.1 Basic layout of the app

The core todo app will have two models, namely a Task and a Category model. Further plans include integrating an authentication system which will end up requiring a User model as well. It consists of two pages, a task page which lists all the added tasks and a category page listing all the categories. A tagging system is implemented through which tasks, while being generated, can be sorted into available categories.

The app will use Bootstrap as a front-end framework to make it mobile-responsive, and also use HAML instead of HTML to make linting and code-reading easier.

1.2 Execution Plan

Having worked previously on Django, I had some idea of how the Model-View-Controller structure works before beginning working on the Todo assignment. As such, understanding the model, view and controller relationship in Rails did not take so much time, which is why I have ended up finishing a significant portion of the application.

My plans for having a basic skeleton of the app before the mid-assignment deadline have worked out, so that I have enough time to work on front-end on which I have essentially no experience. For the remainder of the vacations,

I will work on the front-end and other functionalities and refinements which will make the app more user-friendly.

2 Problems Faced

2.1 Authentication System

Initially I implemented an authorization and authentication system using the high-level Rails gem called Devise, which made the process fairly straightforward. However, I ended up learning through forums that Devise might not be the best thing to use if someone is a beginner to Rails, and it is important to learn how the concepts of users, sessions and authentication works in Rails.

To make sure I learn such important concepts, I will thus be implementing an authentication system from scratch, something I have yet to begin on.

2.2 Marking tasks as completed

Something a todo app should intuitively contain is the ability to mark tasks as done without having to go to the edit form for the task. This would most probably consist of a checkbox for each task in the task list, marking which would dynamically change the completed field of the task model's instance. This would require some kind of Javscript solution I haven't been able to work out yet.

3 Suggested improvements to the app

3.1 Search Functionality

Something which would make the app way easier to navigate through is a search functionality, which would filter tasks/categories through what is entered in the search bar. This kind of search functionality can also be included in the create/edit task form, where suggestions from categories would come when the user enters categories the task should belong to.

3.2 Test-driven development

Writing unit tests for different functions and components of the web-app would make is extremely easy to ensure a proper working of the todo app. However, I am not at all familiar to how testing works on Rails.

3.3 Automatic creation of categories

During the creation of a new task, if a category is entered in the categories option which does not exist, a new category will automatically be created and associated with the task being created.

3.4 A distinct backend API

Initially I was planning to create the front-end of the app using React, with a backend JSON API using Rails. Such a distinction would allow easier debugging and structure, however I could work this out due to my unfamiliarity with React.

Note: A working copy (several improvements yet to be implemented) of the todo-app has been deployed to Heroku, and can be found at: `http://gentle-coast-15254.herokuapp.com/users/sign_in`