Documentation

Functionality and Design Decisions

The main functionality includes:

- Ability to sign up (create user)
- Custom built authentication for signing in
- Ability for users to create / show / edit / destroy a todo list
- Ability for users to add/edit/destroy tasks inside a todo list
- Ability for users to sign out
- Authorization allowing only users to edit only their lists and tasks
- Fully implemented using HTML5 + CSS3

Left out functionality:

- Sharing lists (half-implemented)
- Offline tasks

The two left out items were done so mostly due to time constraints. Sharing lists has been half implemented by way of a table named “collaborations”, this contained obviously the list_id and user_id for the share. I found I ran into problems attempting to implement the associations for the models. Obviously collaborations belongs to both users and lists, lists have many users through collaborations and users have many lists through collaborations. The problem is that users already had many lists (user_id is a part of a list entry), by users having many lists twice rails seemed to get quite confused. Ideally I would have rethought the schema to allow for this, but attempting the sharing too late in the game meant it was too late to switch schemas.

I also attempted offline tasks using the rack-offline Gem but found that it almost worked too well (Firefox would not go back online). I had problems with the manifest, initially the manifest was fine and Firefox cached the page, something then went wrong with the manifest and Firefox refused to load it, serving only the cached page, even when I deleted the manifest.

Review and Further Development

I obviously had many problems with this project. Originally this was a group project, however unfortunately group dynamics fell apart and I ended up starting a new project a week before the deadline. In retrospect I should have seen and confronted

issues much sooner, ideally mitigating the problems and getting the group back on track or branching out on my own earlier in the timeline.

Obviously I have learnt a lot from this experience. Ruby was a completely new language for me, luckily Ruby on Rails is an extremely easy framework to learn and development was very rapid. Ruby on Rails also has the advantage of having a very large community with a vast amount of documentation and tutorials, all these things combined means it is very easy to tackle common problems in Ruby on Rails.

Beyond the basics of Ruby on Rails concepts, I also learnt some more advanced development methods, such as; how to create a custom based authentication system, basic associations in rails, as well as how to test your code.

I did however hit a number of walls when attempting more complex tasks and had trouble finding information online to solve these problems. Examples include that above of associations between users , lists and collaborations. Another example is performing functional tests on models that belong to another model, testing of creation of tasks fails because the test classes cannot resolve the url for adding a task.

I do feel I succeeded in providing a fairly secure and robust system with a lot of functionality given the time constraints, however given more time I would definitely work on fixing the sharing and adding the extra functionality. Beyond that I would have attempted to make the emails be sent with a template using the logo, and ideally integrate alternative authentications such as OAuth.