CITS3403 Web and Internet Technologies – Project 2

Functionality and Design Decisions

Functionalities that are included in the system.

- Users can create an account and log into the system.
- Users can create a todo list.
- Users can add, edit, and delete items from the todo list.
- Users can mark items as completed.
- Users can set a due date for each todo items.
- The system will send out an email at 7.00am everyday to all users with a list of all tasks due that day.
- Have complete unit and functional/integration tests.

Functionalities that are not included in the system.

- Inviting friends to view the list.

Design Decisions

Rails already determine all the architectural design decisions, and we have no qualms with how it is handled, so we left it as it is. Therefore, this makes the database schema the only design decision we have to make.

We initially started off with 3 models: User, List, and Item. However, we decided to scrap off List as it gave us some problems with accessing data between models, as well as making the application unnecessarily complex. The initial concern we had was that it would render the functionality where a user can invite friends to view the list impossible. However, we soon figured out a way to circumvent this problem - to give the User model an extra attribute called “secret_key” (it is assumed that each user can only create one list) that contains a randomly generated key. This key can then be emailed to a friend who when trying to access the user’s todo list without being logged in, will be presented with a page that requires him/her to enter this secret key shared by the user.

Review and Further Development

Team's Development Process

We started off by doing our own static sites for the first stage of the project. After that part is completed, we came together and reviewed each other's work and picked our favourite, while incorporating some of the better aspects of the other versions into the main one.

For the actual Rails application itself, we worked on our own through a really comprehensive Rails tutorial until we're familiar with developing a Rails app, then we just pick the best done version and worked together to improve it in terms of further functionalities not covered in the tutorials, as well as fine-tuning things to fit with our requirements.

What We Learnt

We basically learned how to develop a Ruby on Rails (RoR) application, from beginning till finish, setup until deployment.

We also learned about how the MVC architecture pattern works, why we should use it, and really appreciated how it made our lives easier when we need to go back and modify the code, as well as just during general development where it helps us to compartmentalise our thinking into the 3 different parts of MVC.

Another important thing we learned about software development is the Test-Driven Development (TDD) methodology. This really gave us the peace of mind when we add functionalities or refactor our code, as we can be sure that nothing goes wrong if all the tests pass. This makes testing a lot easier for us too as we do not have to go back and test the system manually for regression every time a major change is made. One important to note though is that just because all the tests pass it does not mean that the system is error-free, it just means that we are not perceptive enough to write tests that caught the errors.

In addition, we also learned how to refactor our code so that nothing gets repeated, as far as possible, in line with Rails' Don't Repeat Yourself (DRY) principle. This made things easier later on when we want to change stuff as there is only a small part of the code that we have to change as opposed to have to made multiple identical changes throughout the entire codebase due to all the repetitions.

In general, we really enjoyed working with RoR, as we had to do everything manually before this, although it took some getting used to. We now learned to appreciate how Rails' convention over configuration paradigm, and also just frameworks in general, can really make programming so much more fun as they can really help software developers focus on the big picture software logic part as opposed to worrying about all the tiny little details, as well as greatly boosting a developer's productivity.

What we would have done given more time

We would love to add the inviting friends to view list functionality if given more time as well as making the code a little more polished.

Other ideas for extra functionalities that we would like to implement if given more time includes:

- Drag and drop reordering.
- Syncing a copy of the database for local use.
- Let people send in an email to a special email address linked to their account to add their todo items.
- Let users create different lists for different projects.
- Set todo items to be preceded by and succeeded by other items, etc for the use of project managers, as well as the ability to generate Gantt charts and stuff.
- Letting users set recurring dates for their todo items.