SE31520 Assessed Assignment 2011-12

Enhancing the CS-Alumni Application

Chris Loftus
Hand-out date: Monday 24th October 2011
Hand-in date: Monday 12th December 2011
50% of SE31520 assessment

The problem

During the course of the module we have been looking at building the CSA Ruby-on-Rails application. In this assignment you will take my most recent version of the prototype and enhance its functionality or add scenario based test support (described below). I want to give you the freedom to be creative and decide on possible enhancements. That said, I have attached an initial set of requirements that I have been following when developing the prototype and whatever functional extension you choose must fit with those requirements. However, you may come and see me if you have a great idea that does not fit. I suggest you just go for one key enhancement or undertake Cucumber-based testing.

This assignment has been given out very early to give you time to think and develop the code incrementally. Some relevant lectures and worksheets will not be covered until several weeks into the project. In particular, lectures on Rails support for authentication and Ajax will not be dealt with until November (see the schedule). This may influence your choice of enhancements.

You must provide Rails unit and functional tests, even if you decide to also implement Cucumber-based testing. The current prototype's testing support is very weak and needs fixing and enhancing.

I also want you to analyse the existing prototype and provide a design section in your report that describes the final design. This must cover both my code and your extensions.

You are allowed to refactor my code. However your report must justify the refactorings. However, please do not completely re-implement my prototype!

Please try to respect the REST way of doing things when adding new functionality.

This assignment should take in the order of 50 hours, however the exact figure depends on whether you undertook sufficient preparatory work and study (worksheets, reading and attending lectures) and your software development abilities.

When testing the application, be careful that you don't swamp our email servers with emails. They will be treated as spam and you will be prevented from emailing for a half an hour or so. Instead, record emails in the development log file and just do a small test to make sure emails really work. I have commented out the configuration code that allows emailing. Also, make sure that if you do send test emails then send them to yourself and not to me!

Possible enhancements

Here are a few ideas. However, don't feel constrained by these suggestions.

- Explore and add Cucumber and RSpec test suites. I have not explored this recently and you may find installing the gems difficult on Windows. If this proves difficult don't waste too much time and switch to doing something else.
- Develop a REST-based web-service client program that communicates with the CSA application using XML or JSON as the data representation. The client may be written in any language you wish. CSA's authentication support may make this more difficult than you might initially think. You might consider commenting out the authentication/authorisation code during initial development.
- Change the broadcast function to broadcast a URL to a weblog entry rather than the text itself. The HTML text should be saved then in a place pointed to by the URL.
- Implement the Facebook feed.
- Implement the RSS feed.
- Improve authorisation to enable multiple logins to be considered administrators.
- Convert from Prototype Javascript to using JQuery and use Unobtrusive Javascript (UJS) to implement the CSA's Ajax features.

Steps

Undertake the following steps:

- Study the above brief and the attached requirements document.
- Download the current prototype and get it to run. It uses the gems installed during the worksheets (apart from the Twitter gem). These are declared in the *Gemfile* file. Two plugins can be found under *vendor/plugins*. These are to support the use of SSL (application_controller.rb has switched this off), and the use of legacy non-UJS approaches to using Javascript in Rails.
- Analyse the existing application. You will find existing design documentation in my Requirements slides (on Blackboard).
- Design and implement your enhancement. If necessary, speak to me if you're not sure.
- Make sure you have implemented some examples of unit and functional tests that test my code and your additions. Just provide a few examples for each model class and controller class.
- Write a report (five to eight pages of text with additional pages for screen shots):
    - Write an introduction.
    - Write a section on the architecture of the application and rationale for decisions made. As part of this, produce a UML diagram(s) that shows the architecture of your application. You can base this on the design from my slides with your changes and enhancements highlighted. I drew mine using Powerpoint, but feel free to use another tool or even to draw neatly by hand!

    - Write a section on your test strategy. IMPORTANT: Provide a screencast of your application working (some free screen-casting tools can be found online) and that focuses on your enhancements.
    - Write a critical evaluation section. Say what you found hard or easy, and what was omitted and why. Provide a critique on your design and the appropriateness, or otherwise, of the technologies used.
- Burn a CD that contains the report, the Rails project and the screencast. Include a README.txt file that describes what I need to do to run the application. No printed documentation is required, just the CD and a signed Declaration of Originality form (obtain from the Department Office).

Learning outcomes

By undertaking this assignment you will:

- Demonstrate that you know how to use Ruby-on-Rails technologies: ActiveRecord, view templates, controllers and REST-based web services.
- Demonstrate an ability to use Ajax.
- Demonstrate the application of design patterns.
- Demonstrate that you are able to produce a UML diagram(s) during the design process and then translate this (these) into code.
- Demonstrate that you know how to undertake suitable testing.
- Demonstrate an ability to maintain existing software.

Submission Date and Instructions

Your solution to this assignment must be deposited in the Department collection system on Monday 12th December 2011 and between 2-4pm. If you are late then please complete a Late Assignment Submission form and hand this in to the Department office. All required forms can be found at:
http://www.aber.ac.uk/~dcswww/intranet/staff-students-internal/teaching/resources.php

Note: this is an “individual” assignment and must be completed as a one-person effort by the student submitting the work.

This assignment is **not** marked anonymously.

I will attempt to provide marks and feedback by the Friday 13th January 2012.

Mark breakdown

Assessment will be based on the assessment criteria described in Appendix AA of the Student Handbook. However, the following table gives you some indication of the weights associated with individual parts of the assignment. This will help you judge how much time to spend on each part.

Successfully running the provided CSA prototype	Is there a screencast that shows the original prototype running?	5%
Design and documentation	Is there a design diagram(s)? Is there an associated textual description? Is there rationale for design choices made? Quality of the design. Documentation quality.	30%
Implementation	Is the enhancement implemented and running? Code quality: comments, identifier names ...	35%
Flair	You implemented and documented something in a way that really impresses me. I'll know it when I see it.	10%
Testing	Good set of unit and functional tests. Discussion of results and test strategy. Inclusion of screencast is essential. I need to see the proof.	15%
Evaluation	Critique of your performance. Critical discussion of approach taken and technologies used. Problems encountered and if and how you solved them. What did you learn?	5%