Fallon & Byrne online shop project implementation report

Submitted by Ronan Carton and Evan McLaughlin

In support of Server Side Assessment Semester 1, July 2012

Project Logins

Administrator:	ronancarton7@gmail.com	Password: 233drove
User:	tedsmith@tedsmith.com	Password: tedsmith

Project workload split GitHub Repositories to be reviewed

https://github.com/RonanCarton/ECommerceBuild

https://github.com/RonanCarton/ECommerceBuildRevA.

Note two repositories to be reviewed above. Both contain stages pre and post GitHub detached head issue.

Implementation strategy.

Project Plan
Ronan prepared a project plan that outlined our examinable requirements and also what we had defined in the proposal as additional to core requirements, ie not essential to the correct operation of the site as defined in the proposal document. The plan was used to convey the scope of development and allocate initial tasks (see appendix).

Communication
Text documents were prepared and shared using google Docs, giving both partners access to edit and amend. Problems were discussed and solved by phone and in class discussion. On one occasion the problem encountered by one partner had already presented itself to the other partner so was easily corrected. Tasks were delegated via email or face to face meetings. Regular communication was necessary to ensure tasks were completed and tasks were not duplicated by both collaborators.

GitHub as shared repository
A Repository was created on GitHub. as https://github.com/RonanCarton/ECommerceBuild
We used the Windows GUI which made synching relatively straightforward.
Only one major problem encountered where we had a Detached Head issue which was caused by unstaged ie out of synch changes.. That was resolved by cloning the repository, modifying it locally and pushing the modified version up to GitHub. GitHub is a strong resource offers a lot but in our time using it we did not reach a comfortable understanding of all of the concepts but we did not abandon it and in future projects can see the value of using code sharing via Gist and of generating project documentation Wikis, two facilities the site offers.

We did come across a second Detached head problem which we resolved by reviewing the conflicting out of sequence commit, isolating the commited files and replacing them locally and loading a new repository as https://github.com/RonanCarton/ECommerceBuildRevA.

These solutions were not an effective use of GitHub best practice or a clean solution but this got us through a bottleneck to continue developing and learning.

We also made local dated copies of the project file as a tertiary backup.

Database persistence and GitHub.
We had problems running shared copies of our app when Synching and Cloning from GitHub. These occurred when the databases were out of synch. This issue was resolved by e-mailing. By default Rails places the DB folder, Temp folder and Log file under GitIgnore for good reasons as Active record is the core resource the needs to remain clean as a detached entity. The current DB directory to replace the locally outdated DB directory.

Software development path
The initial software frame was generated by adapting the five tutorials and additional functionality code was sourced from Agile Web Development with Rails.

Testing
We did modify some of the Functional testing files (as per agile book) this is an area that we need to look into further to gain a full understanding of its impact and usefulness throughout the development process.

Debugging Processes

Simple Errors
Ruby Mine highlighted simple formatting problems such as non termination of code ie closing “end” in Ruby code or leaving HTML tags open in ERB files.

The more difficult problems
Running our application with Rails in browser errors were regular. In resolving issues we did not look deeper than the application trace to track affected files. We were pointed to file that contained the errors and these errors were mostly resolved through initially commenting out and cross comparison with the structure of files in similar categories ie View and Controller and less frequently Model.

Server Side Forum
Keeping a close eye on problems other students were encountering and seeing possible solutions to these problems helped a good deal, also it was good to know we weren’t alone. We did hit frequent walls where a function that worked well previously broke.

Other sources for solutions
We found that someone has usually had the same or a similar problem http://stackoverflow.com/ proved a valuable asset if not providing direct solutions then pointing us in the right direction.

Code Sources
We did not use code other than from the tutorials and the Agile Development with Rails book.

Use of Active Record

Migrations
These were mainly error free whether they were scaffold generated Models Views and Controllers or Seed Migrations. Some tables currently have unused columns and it is recognised that these should be deleted to provide a cleaner data structure and avoid any future data errors

Site CSS.
We implemented a basic twitter Bootstrap. ie CSS only. The Bootstrap offers a lot with regards to CSS grid systems and Javascript but full integration is not supported by Rails 3.0.10 although we went as far as acquiring the Gem files before we realised.

Partials and Helpers
We used a partial to render the cart to the side. Partials and Helpers are two functions that we need to review in terms of abstracting code out.

Some of the many problems encountered

Importing Images
Inserting Images with image_tag was straightforward but we Ronan encountered problems in displaying the product images that were held as relative paths within the products table in the database. We raked various migrations to generate potential image holding columns including image_tag, image_url and binary_data before reaching a solution through discussion in class with the following modification to our products index.html.erb file.

```
<%= image_tag(product.image_url) unless product.image_url.nil? %>
```

One letter wrong issues
We encountered three problems where one incorrect or omitted character threw the code completely.
User Model Method:
The Digest class in our encrypt method was not one until we capitalised it

```
 def encrypt(string)
    Digest::SHA1.hexdigest(string)
  end
```

In config our routes file, we couldn't route to "Welome" as it was.

```
root :to => "Welcome", :action => "home"
```

And a seemingly innocuous non HTML € symbol that generated the 500 internal server error that locked us out of our products Index.html.erb in browser view.

Problems of a broader scope than the App

Evan had issues with running the project when it was cloned from GitHub. The problem was isolated as being a route path issue this was resolved by copying the Ap folder.
More routing problems were flagged as being issues with Devkit installation and the server seeking a Json gem. Both were resolved through bundle installs.

The final routing error was isolated as being a typo within the “welcome” view files.

As the scope of the site increased Ronan had issues with the Ruby Interpreter and Rails Server regularly crashing. This slowed development although not significantly.Ronan also encountered a 500 internal server error.
Ronan posted the issue on the Server Side Forum along with the Apps Development stage log file. Although responded to as a forum query the issue was not fully resolved. At the time of writing it still occurs but will be looked into.
Obviously this project could not move beyond development without this issue being isolated and corrected.

Variation from the Requirements and Specification
Two major non “sales pipeline” requirements were implemented.
Theses being The Meet the Suppliers and Blog pages.
Other specified pages intended to garner trust from users were not implemented due an underestimation of time required to develop basic functionality.

In conclusion.
This was our first run out with Rails and the Rails environment certainly offers great potential. Working with rails is challenging and it requires a great deal of time and effort to become familiar and comfortable with it. We recognise the power and usability of Ruby on Rails.

What we learned and how to correct our knowledge deficit.
This project allowed us to implement lessons learnt in class and through tutorials. Skills and knowledge taught during lectures were extracted and put to use in this project.
Knowledge deficit was minimised through communication with project partners, lecturer and forum. Knowledge deficit was bridged through resources such as Agile Web Development with Rails.

Get to know GitHub
GitHub is an exceptionally useful service and this project would have been been extremely difficult to complete without it. Like other parts of this project, it takes time and effort to become familiar with. There were some problems cloning and synching commits usually due to database conflicts. Hopefully we have learnt from this and will remember this when we begin our next rails project.
Next steps
Explore Rails Gems and Plugins and most importantly review Ruby language now that we are dealing with it in a practical way and getting our first look at how we can harness its potential within the Rails environment.

Report Breakdown Ronan Carton 70% Evan McLaughlin 30%