

# Software Modelling and Design

## *MediaMajik Design Reflection*

Edward Crupi 538156

Wenda Xiao 661252

For our 2014 Semester 2 Software Modelling and Design class we set out to design and implement a media sharing web app that could handle the uploading, viewing, sharing, editing and versioning of various images to the system. As part of the course requirement the app was to be implemented in Ruby On Rails, a framework built upon Ruby that is structured around the Model, View & Controller (MVC) design pattern. The project taught us a lot about Ruby, the Rails framework and most importantly the MVC design pattern. Numerous changes occurred in our architecture design as the project went on and we learned more about each. We eventually ended up deploying the app on a third party server so that others may view the project on the web at <http://mediamajik.ninefold-apps.com/>

The architecture we originally submitted for the app we were going to implement was quite different from the app we ended up making. Originally, we had intended for our app to contain extra classes to increase modularity, decrease coupling and increase cohesion. One such extra class was the File class from which the Image class was to extend from. This class was quickly scrapped from the design as we discovered that implementing persistent inheritances via a database proved to be very difficult due its required use of Multiple Table Inheritances (MTIs). As Rails requires a database upon which to build, our app would have required MTIs had we wanted to implement it according to our original specification by including class inheritance. Implementing inheritance via MTIs would have taken too long to set up and would prove to cumbersome for what we needed to get done in the 3 weeks we had been given to complete the project. Another class that got left out was the Filter class, it was found as we were implementing the 'apply effect' function for our images via the rmagick framework (anticipated in our class diagram as 'IEffectService') that filters and effects were essentially the same thing and that implementing the difference would have decreased cohesion and increased coupling, a non-ideal state of an architecture.

Other changes included interfaces we anticipated we would be sourcing externally but instead required building ourselves as well as just generally fields that we didn't realise we would need to be including in our models. An example of this was the ditching of the PaperTrail versioning framework that we were going to use as an interface to our Image model (referenced in the class diagram as 'IVersioningService'). This last-minute exclusion

arose out of a conflict between the other interface with our Image model that was handling file uploading, CarrierWave (referenced as 'IFileService'). CarrierWave required fields in our model that our proposed versioning interface PaperTrail was not able to persist in its own versions. It turned out that this required us to construct our versioning implementation by storing Images as a sort of Linked List or Tree/Stick that belonged to a parent\_image that had 0 to many child\_versions (representing the different edited versions of the once image). These changes were confusing to get set up but once working worked very well. It is believed that the app would have not have been able to include as many features as it does if we had stuck to not changing any one of these original design decisions.

Some things that worked well for our project were the Authentication service, the establishment of base scaffolding (setting up the Models, Views & Controllers) and associating the classes between each other. The authentication service implemented was a third-party sourced interface known as Devise (referred to in Class Diagram as 'IAuthenticationService'). This interface worked well as it was familiar due to being used once before in class and thus took less 'learning' to get going. The establishment of our Models, Views and Controllers was also quite painless as rails took care of most of the work for us by generating the requisite classes from what was essentially 4 'generate scaffold' commands. Implementing the associations between classes was a bit trickier but after learning about and setting up the requisite join tables the associations worked really well with our app and made development much easier than had they not been there.

Some difficult parts of the project that did not work as well was implementing the versioning feature of the app. This required rethinking how images were being stored in the system and altering the tables representing the model themselves. It was decided that the versions would be stored as a version-tree, each version represented as a node in the tree. It took a long time for this design to be worked out in comparison to the other parts of the project. Once it had been thought about it also took a long time to implement and debug.

Apart from these changes and setbacks it is thought that the app is quite successful functionally and could enjoy more success if it could undergo some cosmetic improvements via improving the CSS. Some minor CSS improvements were managed via importing the Bootstrap framework and it is thought that more time spent on the various links and divs throughout the app would yield some great result in terms of look and feel. Another way to improve cosmetic improvement would be to implement some javascript or jQuery interfaces that would increase responsivity of our app (allowing the display of images before uploading and editing).

For future editions of this app it is wished that the MediaMajik system be extended to more fully represent its name. This means extending the upload, edit, share and versioning functionality to other media types. Allowing other media types to be uploaded would be the next great milestone for this app. CarrierWave would definitely be able to handle this in terms of uploading the files and perhaps spending some more time learning about how to implement MTIs would enable us to use Images, Files, Audio and Video as inheriting classes extending from a MediaType class.

This project was an interesting project and we learned a lot about Ruby, Rails and the MVC design pattern. The Making the improvements specified here in the future would help us learn more about Multiple Table Inheritance and using Bootstrap to improve our CSS stylesheets.