## During Holidays

During the holidays no meetings were held (as our group was separated), instead we communicated mainly via emails. These points summarise our decisions made and discussions we had via these emails:

* Made the decision to use email as our group’s main communication method. This was chosen as our group was widely separated and we had no chance of face to face contact
* **Brainstorming:**
  + We thought of general ideas of where we wanted to go with the system
  + Main role is to be a ‘News Feed Aggregator’ (as per the specification), as such it must be able to crawl, parse, store and display content from RSS feeds as its main functionality
  + Decided we would be making the product as a service rather than a system (so we could focus mainly on user functionality)
  + System could be broken down into 2 main parts, front-end for displaying the content to the user and backend for performing all the processing
  + Discussed the technologies we should use to construct the system:
    - PHP web server, accessible to the user via web pages
    - C++ for the backend
    - MySQL database for storing the data (MySQL was chosen as it was a cheap and simple option)
  + Discussed the architecture to use for the project. Repository structure, front-end and backend talking to database which stores feed and user information
  + Discussed possible users:
    - Standard users (business people, students, general public)
    - Administrators (for accessing the system)
  + Discussed the content sources we could possibly crawl:
    - RSS (most importantly)
    - Emails
    - Forums
    - Standard websites (HTML)
    - Social Media (Facebook, Twitter)
    - Weather feeds
* Decided our main task in the holidays would be to work on prototypes for each module of the project
  + This is to both test our ideas and learn how to correctly construct the module
  + The prototype for each section is a ‘throw-away prototype’, we will be building the final system when the session starts (and the prototype is done)
* Assigned some general roles for which section we would be working on:
  + Ian – Front-end web development
  + Alex – Backend parsing of RSS and other content and crawling of feeds
  + Aron – Backend threading
  + Michael – Algorithms for queuing and documentation
* Registered Domain Name (newsfeeder.co) in preparation
* Created SVN to store the work and simulations done in holidays

## Meeting Minutes – 27/2/12 (First Formal Meeting)

**Progress Reports:**

* **All**
  + Fleshed out requirements and refined ideas on architecture/technologies for their section/module
* **Ian**
  + Developed the initial front-end website (working login, display of dynamic content, etc)
  + Tested PHP PDF generation (is working, not the most efficient method though)
  + Tested Bing API for searching the internet for suitable feeds to add, works well (slow due to memory cache)
* **Alex**
  + Written up RSS parser
  + Currently using the library libXML (has found issues with memory leaks)
  + Written up basic crawler, tested and runs quickly
* **Aron**
  + Ran simulations of the priority queue. Is working on integrating them successfully with threading (and testing that)
* **Michael**
  + Thought of algorithm for priority queuing
  + Tested algorithm with the help of Aron’s simulations

**Agenda:**

* Completed set tasks for first week (get group, project and supervisor), all done beforehand
* Confirmed times for meetings:
  + 4:30 Monday, after the 321 lecture
  + 3:30 Wednesday with Luke
* Clarify any problems/questions we had with progress during break or initial brainstorming
  + Marked off any implemented functionalities (e.g. login and registering)
  + Added ‘Abuse Reporting Blacklist’ to requirements, this is the ability for web masters to report if the system is spamming/abusing their site
* Created Initial Use Cases
* Answered any questions on each section which had been worked on over the break:
  + Michael answered issue with how the priority algorithm handles initial start-up of the queue
  + Alex noted concerns with libXML and about whether to use Boost or P threads. Selected Boost
* Decided upon documentation to start:
  + Formalised requirements
  + Use cases
* Specified Ian as team leader, as he has the easiest means of communication with our supervisor Luke.
* Specified Michael as scribe
* Clarified questions to ask Luke at first meeting:
* Can we have copies of the project specification?
* Are we selling as a service or as a system?
* What do you think of development types (prototyping, agile, RUP)?
* Can we use VISIO?
* Can he purchase soft-drinks (priority Vanilla Coke)?
* Clarify Expectation of Time-Line.
* Clarify Roles.
* Clarify Scope.
* Clarify Requirements.

**Work for next week:**

* **Ian**
  + Continue testing and implementing high-level features to the front-end prototype
* **Alex**
  + Work on issue with RSS memory
  + Continue work on HTML parser
* **Aron**
  + Implement ‘start-up’ priorities on the queuing and test
* **Michael**
  + Work on creating the requirements documentation
  + Documenting discussed use cases

## Meeting Minutes - 29/2/12

* Went through previous questions for Luke:
  + **Can we have copies of the project specification?** Copies Given
  + **Are we selling as a service or as a system?** Either, we selected to sell as a service with possible smaller applications as systems/products
  + **What do you think of development types (prototyping, agile, RUP)?** Whatever, went with prototyping/agile method
  + **Can we use VISIO?** Yes, we will be using it for major documentation
  + **Can he purchase soft-drinks (priority Vanilla Coke)?** Possible ‘donations’
  + **Clarify Expectation of Time-Line.** Generally follow subject outline, already ahead in this regard
  + **Clarify Roles.** No specific roles, chose to specify areas of project to people rather than standard roles
  + **Clarify Scope.** Done
  + **Clarify Requirements.** Added some more requirements, confirmed current ones are suitable.
* Luke brought up the following points:
  + Force a print view of a feed item’s page to retrieve the additional content (so as to remove non-necessary features like advertising)
  + Look into stress testing methods
  + For adult content filters, look at hooking into existing browser filters
  + Look into creating payment scheme for project:
    - Pay to remove advertisements from page?
    - Additional features?
  + Add localised content filtering (for example in Germany Swastikas should be censored/filtered)
  + Add ability for users to customise sheets themselves (drag and drop interface)
  + Formatting sheets is at much higher priority, as is directly related to the user’s experience
  + Restrict sheet sizes (such as only 10 items per sheet)
  + Look into restricting user privileges (limiting sheets, feeds, etc, so as to not abuse)
  + SSL? Encrypting communications between system servers and the client

## Meeting Minutes – 5/3/12

* **Progress Reports**
  + Ian:
    - SSL Implemented
    - Problem with deletion, need to add dynamic SQL generation (databaseconnection.php)
  + Alex:
    - Socket designed, in the progress of implanting
    - Refactoring the backend
  + Aron:
    - Designed customisable UI layouts
  + Michael:
    - Designed customisable UI layouts
* **Agenda**
  + Aron and Michael discussed the ‘customisable UI layouts’ proposed by Luke
    - Essentially the user can drag and drop feed types into a defined layout
    - The layout itself is defined by dragging and resizing areas on a sheet to be filled with content
    - To use a standard column/row type format (as is similar to standard sheets)
  + Added additional requirements, formally added the ‘customisable UI layouts’ and what it entailed to the requirements
  + SSL
    - Group decided it would probably be by user preference and it would default to ON
    - Ask Luke for his suggestions on the security and how to implement it
* **Actions for next meeting:**
  + **Ian:** 
    - Correct any SSL issues and get prepare questions to ask Luke about the issues
  + **Alex:**
    - Continue working on Parsers (especially HTML parsing)
  + **Aron:**
    - Test threading in regards to the crawler more thoroughly (compare more threads vs fewer threads)
  + **Michael:**
    - Continue writing use cases / requirements documentation

## Meeting Minutes – 7/3/12

* Asked Luke about Security and SSL
  + Authentication for services (such as API and downloader application)
    - Given to registered users
  + Authentication only if we want to restrict access to registered users
  + SSL on important data (login, registering, etc)
  + No SSL on non-important data (its processing costs are too high to use for the whole site)
    - May put it on data about user preferences
    - Luke suggested defaulting to off instead of on
* Asked about initial documentation:
  + Nothing complex for the documentation, keep simple especially for user documentation
  + Format the requirements in table form
  + Start a WordPress for keeping diaries
  + Document main competitors
    - Google Reader
    - Mashable

## Meeting Minutes – 12/3/12

**Progress Reports**

* **Ian**
  + Implemented changes suggested by Luke about SSL
  + Found issue with PHP database connections class, currently trying to fix
* **Alex**
  + Basic HTML parser finished
  + Begun wrapping classes
  + Wrote up backend class diagram for discussion in this meeting
* **Aron**
  + Tested threads, more threads superior (provided we do not pool too many), performance hit is due to queue sorting
  + Started technical documentation
* **Michael**
  + Finished initial high-level use cases
  + Still writing up requirements documentation (working on priorities currently)

**Agenda**

* Analysed the current queuing architecture:
  + Is Crawler Controller getting the work from the MySQL updater or is it from the Crawler Thread? From the Crawler Thread
  + Polling (Crawler Controller polls from database every n seconds) OR When it initialises, have it add a PLACEHOLDER item onto the queue? Decided upon Ian’s suggestion, using the placeholder
  + Essentially when a new feed is added to the database there will be a message sent to the ‘crawler controller’ which will create a placeholder item in the queue for that new feed. This feed item will be filled in when it is first reached in the queue and crawled.
* Aron and Michael explained queuing architecture to group:
  + Continuous queue
  + Passes the top X items to X waiting threads, then resorts the remaining queue after incrementing the priority
  + Passed items are crawled in these threads and added back onto the end of the queue
* Analysed Backend design (and class diagram):
  + Refactored current design to fit more with the Model View Controller pattern
  + Using thread pools for all major services (crawler, SQL, etc)
  + General design involves running individual threads for each major service
* New executable to perform the Bing API search in C++ as opposed to PHP (very low priority), currently the search is being performed very slowly (10 seconds to bring up the first result) and mem-cache is still leaking.
* Discussed Iterations: initial throwaway prototype where each subsystem and its features are implemented (to test if they are feasible), followed by a standard iterative development cycle where the product is fully designed.
* Iterations are:
  + 1 (Prototype):
    - Get off the ground
    - RSS Parsing
    - Site Login
    - Sheet Management
    - Database connections (backend)
    - Database connections (frontend)
  + 2 (Prototype):
    - Threading
    - Queuing Simulation
    - Communication between frontend and backend (was not prototyped)
    - Interfacing with front and back ends (Marks completion of protyping)
    - Finalise Requirements (right at end)
  + 3
    - Finalised designs
    - Begin development of final products
      * BASE REQUIREMENTS
  + 4
    - More functionality
    - High level documentation
  + 5
    - TESTING
    - Bug squashing

**Actions for next meeting:**

* **Ian** 
  + Keep working on fixing that database controller
* **Alex**
  + Refactor backend to fit with new design
* **Aron**
  + Document the proposed backend redesign
* **Michael**
  + Continue working on requirements
  + Look into database controller with Ian

## Meeting Minutes – 14/3/12

* Discussed backend redesign with Luke, explained queuing method. No problems found
* Luke suggested we summarise our possible applications and addons:
  + Mobile Application – specific interface to browse feeds
    - Mobile support better option?
  + Java Download Application
    - Browser Addin or Desktop Application? Desktop Application
    - Main requirement: lets you view sheets offline
    - PDF or Offline Sheets
    - End of session
  + Sheet Designer Application
  + Storing Images
  + Add Image Searching
  + Social Network Integration
  + Public API Web Service
  + Pulling down email content (additional content type)
  + Email Notifications
  + Emailing sheets

## Meeting Minutes - 19/3/11

**Progress Reports**

* **Ian**
  + Database controller still broken, currently working on it
* **Alex**
  + Backend refactoring going well
  + RSS parser and RapidXML implemented (RapidXML not leaking so far)
* **Aron**
  + Architecture documentation begun
* **Michael**
  + Initial requirements done, looking for approval

**Agenda**

* Discussed the possible extensions suggested last week, and whether we would do them:
  + YES OR NO (or wait till later to decide)
    - Mobile Application - **very very low for individual application**
      * Should we just have mobile support instead? **YES we will provide mobile style support**
    - Java Downloaded Application **yes, will confirm at a later date**
      * PDFs or Offine? **Not yet decided**
    - Sheet Designer Application **Definitely Not (already done in UI)**
      * Is currently being done in UI should we add specific application?
    - Image Storage **could be done, have to do tests on storage**
    - Image Searching **Definitely Not**
    - Social Network Integration **Yes**
      * What does this entail?
        + Authenticating using apps and logons **Yes**
        + Adding social network content **Yes**
    - Public API Web Service **Yes if we can get the Java App done, since it will use the service**
      * Why and for what cases?
        + REST API
        + Already being done for Java Downloaded app
    - Email as Content Type **VERY VERY VERY LOW**
      * Already added as requirement
    - Email Sheets and Notifications **YES**
      * Notifications **YES**, Articles sent as PDFs? **YES**
        + User confirmed: Do you want to send notifications? If yes, Do you want to send a PDF copy of the article also sent?
        + Batch emails (Digest)
* Looked into Front End Design:
  + Major suggestion was the change to Smarty templating, to replace the GUI maker class currently being used, Ian is looking into this
  + Site Structure – fine, similar to other services
  + Classes – basic name changes, removal of init.php, change to Smarty
  + UI – not discussed
  + Sheets – not discussed
  + Custom Sheet Design – not to be discussed until better defined

**Actions for next week:**

* **Ian:**
  + Keep working on fix to database connections
  + Look into Smarty templating
* **Alex:**
  + Keep refactoring backend
  + Test RapidXML further
* **Aron:**
  + Look for Smarty guidelines to use for project
* **Michael:**
  + Add new applications into the requirements

## Meeting Minutes – 23/3/12

* Ian has fixed issue with database controller, was an issue with scoping of for loops and some SQL queries
* Short meeting with Luke, on Friday instead of Wednesday
* Alex brought up the issue of logging:
  + Only errors currently being written to log file
  + Should implement verbose logging
  + Luke suggested we provide auditing
* Showed Luke our requirements document:
  + Suggested rearrange requirements based on relation to other requirements rather than their priority
  + Added some additional requirements:
    - Logging and Auditing
    - Learning Algorithm (very low)
    - Printing support
    - Payment system
    - Feed categories
    - …

## Meeting Minutes – 26/3/12

**Progress Reports**

* **Ian**
  + Begun conversion to Smarty (needs Aron’s help to choose a templating style)
  + Found new design template to use for the front-end look
* **Alex**
  + Finished refactoring backend to fit with design documents
  + Added more functionality to RSS parser (can now handle more tags to retrieve the correct information)
* **Aron**
  + Found his old Smarty Templating guidelines, had fixed some issues with them to fit with the new project
* **Michael**
  + Added additional requirements suggested by Luke to documents

**Agenda**

* Chosen Smarty Templating Style
* Backend Requirements Finalizing
* Michael explained Priority Algorithm to Alex, to be implemented into the crawler
* Made decision to use Ian’s new design template
  + More professional
  + Provides additional JavaScript functionality the old template did not
* Implemented smarty naming convention
* UI design style
  + Boxes to hold articles
  + Similar to the customised sheet design style (placeholders using Smarty to fill)
* Discussed priorities for requirements, moved some applications to low, and functionality to medium

**Actions for next meeting**

* **Ian**
  + Convert frontend to use the new smarty template
* **Alex**
  + Implement the crawler queue with priority into the backend crawler
* **Aron**
  + Look into REST API’s for the Downloader Application
* **Michael**
  + Work on next iteration of priority algorithm

## Meeting Minutes 2/4/2012

**Progress Reports:**

* Ian:
  + UI with new template refactored
  + Separation between profile settings and account settings
* Alex:
  + Log manager
* Aron:
  + Requirements
  + Investigating web service for Java
* Michael:
  + Requirements
  + Meeting minutes

**Agenda:**

* Had a number of discussions concerning:
* Separation of the modules in the backend
  + Current modules in the backend are for statistics, crawling, admin (and the master for starting all other modules and threads)
* Messaging Interface
  + New table in schema ‘Notification’
  + SQL generation on item update
  + Front end generates document
* Gave the API higher importance (as suggested by Shaun)
* Instead of having individual database connectors for each subsystem of the project, thinking of integrating them all into a shared library
  + Will require a cross-language library
  + Looking into SWIG as a possible option for wrapping the shared library
  + We will only proceed with this if it is possible to run the shared library on all subsystems (C++, Java and PHP)
* Looked over the requirements to be submitted this week. Made no major changes to the document other than adding the ‘Notifications’ requirement to the front-end

**Actions for next week’s meeting**:

* **Aron**
  + Look into adding generalised Database Shared Library
* Ian
  + Notifications, add feed widget
* Alex
  + Fully implement master thread
  + Start administrator service and integration with the front-end
* Michael
  + Finalise requirements document