Executive Summary

This project describes the co-operative implementation of a new online academic social network, which allows users to upload, discuss and discover academic material. This thesis documents the analysis and thought process involved required to best engineer the tools with respect to privacy, security and scalability. I have worked with a fellow colleague Robert Thumpston, such that working together as a team we were able to allocate sufficient time required to complete the numerous aims and goals set out. This project was undertaken to not only explore which services currently exist, but by using a combination of modern technologies build a new foundation with other useful features. Its important that this text provides a detailed explanation of the patterns, tools and models used to ensure minimal time required in understanding the system, so that expansion is possible in the future.

Whilst contributing to the basis of the front and back ends of the server software, I have been specifically focused on the security and scalability aspects. In particular the correct use of cryptographic protocols, sensible access control policies and how to provide elastic access for unknown demand. I have also focused on system designs to allow ease of administration and provision of guidance with procedures to follow to resolve legal issues. However this project would be of interest to anyone that would like an insight into the role social networks and their performance have played in society, by impacting privacy, or encouraging collaboration through networking entities.

- I implemented the secure login and signup system using MYSQL and the PHP framework CodeIgniter, ontop of the authentication library IonAuth.
- Provided key social networking functionality required for users to communicate effectively across multiple platforms. Built an interface for an existing message threading library and used MathJax for displaying scientific symbols. I created interactive user profiles that dynamically generate content and respect legal rights to privacy with information disclosure controls.
- I have researched into the requirements for a scalable solution, and created a secure environment using Amazons Web Services. This allows users to upload research and collaborate in private using a role based access control system.
- I constructed significant parts of the Users, Groups, Papers and related database schemas; delivering presentation to clients through coded objects and methods and the front-end ability to search through this content.

Contents

List of Figures

List of Tables

Abbreviations

AES	**A**dvanced **E**ncryption **S**tandard
AJAX	**A**synchronous **JA**vaScript and **X**ML
API	**A**pplication **P**rogramming **I**nterface
AMI	**A**mazon **M**achine **I**mage
AWS	**A**mazon **W**eb **S**ervices
CA	**C**ertificate **A**uthority
CI	**C**ode **I**gniter
CBC	**C**ipher-**B**lock **C**haining
CSRF	**C**ross-**S**ite **R**equest **F**orgery
CSS	**C**ascading **S**tyle **S**heet
DBMS	**D**ata**B**ase **M**anagement **S**ystem
DNS	**D**omain **N**ame **S**ervice
DOM	**D**ocument **O**bject **M**odel
DoS	**D**enial **o**f **S**ervice
HTML	**H**yper**T**ext **M**arkup **L**anguage
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
HTTPS	**H**yper**T**ext **T**ransfer **P**rotocol **S**ecure
IaaS	**I**nfrastructure **a**s **a** **S**ervice
IP	**I**nternet **P**rotocol
LAMP	**L**inux, **A**pache HTTP Server, **M**ySQL, **P**HP
MAC	**M**essage **A**uthentication **C**ode
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
OWASP	**O**pen **W**eb **A**pplication **S**ecurity **P**roject
PHP	**P**HP **H**ypertext **P**reprocessor
RDMS	**R**elational **D**atabase **M**anagement **S**ystem
SID	**S**ession **ID**entifier
SN	**S**ocial **N**etwork
SQL	**S**tructured **Q**uery **L**anguage
SSL	**S**ecure **S**ockets **L**ayer
TLS	**T**ransport **L**ayer **S**ecurity
XSS	Cross-**S**ite **S**cripting

Chapter 1

Introduction and Scope

1.1 Project Summary

In general terms the aim of this project is to develop a web application that provides social networking specifically tailored for the common interactions amongst academics. Giving users the ability to share, discover and discuss academic research with other contacts at any location. Application on the web already enable millions of users the ability to connect in desirable ways, and we'd like to develop a tool specifically targeted towards improving awareness of scientific literature. Our aim is to provide an environment for authors to publish their material safely and then control how they wish others to view this information, using it as the content for electronic learning. Therefore I begin by researching how this is currently being achieved and where it can be improved. Already scientists collaborate and co-author in many ways to produce publications, and these links shape complex and interesting citation networks, the growth of which can have huge implications on civilisation. Background research also touches on how social graph theory can affect the sensitivity of these arrangements.

The web is being progressively used for educational purposes, and we aim to encourage this further. I have investigated current web solutions and the suitability of existing software within the large area of web development. I will aim to provide an overview of modern web procedures and techniques without requiring excessive background knowledge. However it's increasingly challenging to understand all the technologies involved, and whether or not they fulfil their requirements, so critical assessment of their performance is essential. As it turns out we found many infrastructures in existence that practices a subset of our desires; a primary concern is with providing privacy mechanisms to alleviate restraint in sharing personal data with the system. However privacy is not the only purpose of considering the

security of web applications, fundamental values and morals should also be upheld in accordance with jurisdiction so that no harm is done.

1.2 Aims

The system we are designing is a social networking website, as such it must provide essential functions to match this description. The main use cases that our system should cater for is:

1. Discussions related to published material, involving students, researchers and authors
2. Co-authouring sensitive research data with known scientists

Our initial design specification also identified other ways in which users may wish to use the service, such as wanting their research to be peer reviewed constructively by other users. Or to allow users to work together using version controlling. These use cases are desirable but are left as extensions to the core functionality.

1.3 Objectives

A primary objective is to build a working system that performs the use cases to a high standard for real users. Lesser functional target are needed so that each component can become a clear module which is to be programmed. In order to achieve this I have provided a comprehensive list of desirable goals that enable the web application to fulfil its system requirements:

1. We need to provide a method to sign up to the website securely and obtain an authentication token or username which they can use for future connection.
2. Users with an account should be able to communicate sufficiently through some means on our website. A user must be able to create a profile page that allows them to demonstrate their skills, interests, education, profile pictures, bibliography and contact information. They must also be able to 'follow' and stay up to date with other members of the community.
3. For reviewing to take place users should be able to upload, and view files in popular formats to the server storage in a secure manner. Our server must be able to handle large amounts of data to be stored, including extensive metadata on papers to allow for indexing and archiving.

4. The uploaded files may require a form of tagging by users, for efficient searching and organization. An author may wish to restrict access to a paper through allocation of custom permissions to different users.

5. The website should allow a user to enter discussions with an author relating to their material and interact with them by posting feedback or ratings on the published material. Discussions will take place located directly next to the papers and require useful formatting for Mathematical symbols.

6. The discussions should utilize a question/answer template that adopts the model of a reputation system. This allows users to gain rewards and moderate the discussion by flagging' inappropriate questions.

7. User generated website content must be optionally confidential, and give creators the ability to completely control their appearance on our website. Another aim of the project is to make sure that we consider all the legal systems which our parties will interact with, to act in strict accordance with the law.

8. A user should be able to create a group with private discussion and collaboration, also allow affiliation of known users and papers to the groups.

9. It's essential that the server can contact users via email in order to revoke their passwords, and later be used to inform a user of new content related to their site usage.

10. To guarantee related content and correct site usage, until moderators have been allocated, a user must be a member of a UK University or School. This is only an initial system requirement and can be delivered by providing email authentication using a known list of valid educational institutes email domains.

11. The environment should be accessible on many different devices and the website across multiple browsers so we should to strictly comply with web standards.

12. It must be designed using suitable software design practices, encouraging stability and performance. The source code requires thorough documentation to the ease possible future adoption by other software engineers.

13. Verifiable methods should prevent cyber-attacks from damaging or obtaining access to the system without permission.

14. An adequate policy for an administrator running the website should be in place; this includes updating the system, obtaining logs and backing up server data.

Chapter 2

Background and Context

Since the invention of the web the technological ability for users on different hardware to communicate through computers has been provided by standards and protocols. However, the interface in which the users interact consistently changes, and new platforms for collaborating programmers with new tools always arise. The architecture of the web typically operates using a client server model, which is enabled by the use of client side internet browsing software. Within the server computer is stored the algorithms that handle client requests and perform any additional desired functions. It is the server administrators role to setup the server and ensure that requests are handled accordingly.

One Internet service could include a message board or forum where users can post messages related to a specific topic. Additionally these environments can provide social network functions, such as allowing users to customize a profile space and then associate themselves with other contacts. All implementations should consider usage demands, correctness of desired outcomes and security when handling users private information.

2.1 Social Networks

In elementary terms a social network consists of participants and the relationships between them. From basic interactions between participants or people, complex macro level patterns can emerge. A simple social network can be represented as a graph, or *sociogram*, shown in figure 2.1: the vertices represent people and the undirected edges are mutual relations between them. The graph can be represented as $G_6 = (V_6, E_8)$ where the vertices $V_6 = (Andy, Sam, Alice, Greg, Adam, Eve)$ and the edges belong to the set $E_8 = ((Andy, Alice), (Sam, Greg), ...(Alice, Adam))$. Within network theory measures of centrality determine the importance of a vertex in the graph. An example measure of centrality is the *betweeness centrality*

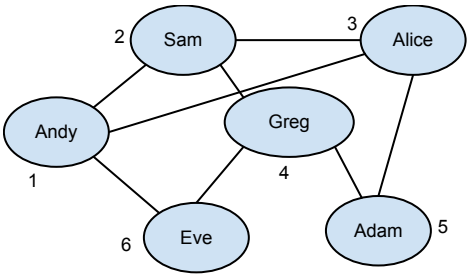


FIGURE 2.1: An undirected graph representing a social network

[1], this is calculated as the fraction of shortest paths through a vertex v, given by equation 2.1.

$$C_B(v_k) = \sum_{i \neq j \neq k}^{n} \frac{g_{ij}(v_k)}{g_{ij}} \tag{2.1}$$

where g_{ij} is the number of geodesics, or shortest paths, linking vertices i and j, and $g_{ij}(v_k)$ is how many contain vertex v_k . Therefore the betweenness centrality of the entire graph is $\{\frac{3}{2}, 1, \frac{3}{2}, 2, \frac{1}{2}, \frac{1}{2}\}$, and we can deduce node 4 (Greg) cuts the most shortest paths and perhaps is the most important player with the ability to react quickly to changes.

Real life social connections can be much more complicated than this, with weighted paths on growing networks. If we were to view a network as a weighted graph we could measure the strength or closeness of connections between people as the frequency of interaction, in this case Dijkstra's algorithm can be used to calculate the shortest distances. These models of networks have allowed us to discover remarkable results that have change our perspective on the world. A famous example is the contributions of Stanley Milgram, who's research is the inspiration for the phrase '6 degrees of freedom' [2], a suggestion that are persons are affiliated by an average of 6 connections. This has shown to apply to even topologically random networks and is one of the reasons social networks have been so successful in connecting people. We are proposing to build a social network where events and the flow of information is determined by the paticipants scholary intentions and the limits of the service. An important finding from my research is that social networking can have a positive effect on productivity of authors, as shown from a small sample of technology literature authors: a positive correlation between productivity and collaboration rates [3].

In the last few months there has been substantial headway made towards providing open access to research, initiated in the UK from the Finch report and the governments response [4, 5]. This aims to deliver material to a wider audience and improve scientific discovery, it will also significantly impact global economics and political thinking.

2.2 Academic Publishing

Academic publishing is the distribution of academic research in the form of articles, journals or books. Prior to release they require some form of peer review to ensure the integrity of the content. There are no set standards for reviewing and a typical selection process is performed by an editor, and involves picking experts on the topic to provide feedback on the document. Usually 3-6 scientists review papers and then rate them on different dimensions such as novelty, clarity, significance. There exist many useful guidelines such as those posted by the European Science Foundation [6], these provide advice on the process and platforms exist for organization such as the public knowledge project [7]. However there is no homogeneous approach and problems of various characters arise: conflicts of interests; anonymity/confidentiality, scoring metrics, and originality [8]. The issues are expanded during conference proceedings where scholars may engage in large discussions and presentations with the desire to publish related material. The links between the referees and the material form a network consisting of multiple links at the interpersonal level and also pseudo inter-organizational level [9]. The initial system design and the network features will stimulate grounds for social capital research. Specifically, the publication model is a form of reputation attainment that benefits the users with influence. The source of this capital is the output from the process of review, but alternative factors such as previous discussion and feedback are also contributions to gain. Working together and co-authoring is a popular method to enhance research, and it's understood that alternative actors, such as individuals deciding not to fully participate in the network thus forming cohesive subnetworks [10]. Its imperative that an official referee is impartial towards the author, this can be prevented through limitations of closeness centrality, strength of a network and overlap of authoring groups [11]. A hurdle that academics have to overcome is how to create the documents themselves, especially concerning complex and intricate symbols or alphabets required. The markup language LaTeXis a solution, and some good books have been written on the subject [12]. Even though it has been argued to be a convoluted workaround [13], it is widely learnt and new tools for creating LaTeX, such as `https://www.sharelatex.com/` where you can collaborate, emerge all the time. The W3C endorsed standard way for viewing mathematical expression on web pages is MathML, and could be a sensible choice for messages, question posts and errata.

The analysis of social networking sites used for educational purposes is relatively new, despite the intrinsic co-existence of education and Internet media. One study reveals that a third of 136 sampled students preferred staff not to use the website `https://www.facebook.com/` as a form of communication and that student-faculty relationships should remain purely professional' *Jeff Cain* [14]. In the same article another trial reveals the quantity of messages sent between students travel mostly within the same school. Speculation suspects that

learning though social media is a cover up for actual learning [15], and argues there is no model to measure the effectiveness of the results. On the other hand there do exist alternative practices of e-learning that are seeing success such as http://peerwise.cs.auckland.ac.nz. This website allows students to 'create and to explain their understanding of course related assessment questions, and to answer and discuss questions created by their peers'. They have members from over 200 educational establishments and understand the need for users to be able to access the content from a variety of devices.

Academic publishing credits scientists with status, and one such naive measure is the quantity of research published. Other methods used to analyse the impact of scientific research are the number of citations or content assessment, these methods belong to the set of *bibliometrics*. However these are only indicators as older papers will clearly have more time to obtain citations, also papers that are hmore popular are more visible so receive a cumulative advantage over other citations, this is known as the *Matthew Effect* [16]. This means new nodes connecting to the network of citations exhibits *preferential attachment* behaviour where a paper is more likely to connect to a popular one, making the network *scale free*. However recent studies of empirical data from citation networks have shown that there are many other factors which affect the growth and type of network [17]. Firstly citation networks are directly linked to co-authorship networks as authors tend to share ideas with people they have worked with and so reference them in their work, this is known as a 'referency bias' [18]. More recent papers are more likely to attain betweeness property as they have access to all of the up to date papers, this is called a 'recency bias'. We want to know how to networks evolve over time, so we can view the citations networks as complex dynamic, through which some degree of preferential attachment exhibits a power law distribution [17]. The cumulative advantage governs the power law probability distribution of the random variable x with $p(x) \propto Cx^{-\alpha}$, $2 < \alpha < 3$. Only paper citation networks have shown to obey the cumulative distribution function 'best fit' with power law, as seen in figure 2.2.

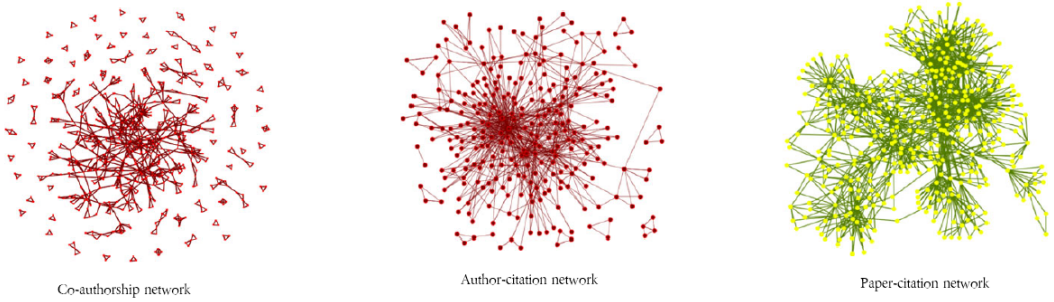


FIGURE 2.2: Nature of citation and author networks in 2002 [17]

The reason for paper networks to be scale free is suspected that anyone can cite any paper, and the network of paper citations is directed, permanent and acyclic. However collaboration co-authorship and author citation networks don't entirely follow power laws for any $x > x_{min}$, mainly because of social connections which form 'cliques' of 3-5 people. A service such as ours may decrease this fragmentation by limiting the exponential decay on the author citation network distributions shown by *M.Kas et.Al* [17]. We see this from the upper bounds of the current physical limit scholars can publish and collaborate within a timeframe. These limitations are clearer from spatial analysis which shows restrictions of the opportunities between countries, resulting in a more sparse co-authorship network. We must pay respect to the underlying network by establishing our key aims, one of which should be to increase academics productivity without significantly altering the quality of work from potential distractions. A certain level of responsibility is required when proposing a system that can alter the shape of underlying existing citation networks, if a paradigm shift occurs it could transform the temporal frequencies. By analysing the frequency of data, using techniques such as a FFT, we can discover when breakthroughs in subjects are near, or how far through the life-cycle a field is, or when new ones are born or connected. All of this causation could be a result of tweaking site features, and thus combinatorial innovation exacerbates security needs.

In modern citation networks, the impact of the work is used to determine how much funding is allocated for further research from governments. In the UK the Higher Education Funding Council uses a framework [19] of these 'indicators' for this exact purpose but it's creation was not without problems. As citations alone can be biased other factors like the ability to accumulate income, or the outcome of the work can be of more use. To be able to compare different fields or years a form of normalization known as *rebase impact* can be performed [20]. The relevance of this bring us into an overview of what services are provided by Google, the cloud computing giant, which run an estimated near 2 million servers [21] servicing over 1 billion search requests a day. Their products are well known, but one that is of specific interest to us is 'Google Scholar'. It attempts to globally index all scholarly literature across many fields and can calculate bibliometrics of authors. An example is the H-index [22]: if h of N papers have at least h citations and the others have less than $h + 1$ each. Issues with Google Scholar are that It doesn't report on how it does processes file or how frequently, nor do they provide access to all the documents. Application programming interfaces (API) exist for obtaining this information from sites of this nature [23].

2.2.1 Privacy concerns

A major consideration of publishing scientific material is the ability to avoid premature disclosure of new results. Privacy can be described as a position in which one is not disturbed by others and can be used to calculate the perceived cost of entering into an exchange relationship [24]. Contrary to common belief users have shown a concern for privacy, for example when 16,000 users signed a petition in February 2010 to leave Facebook [25], further experiments show reduced privacy settings can encourage interaction [26]. However as we are potentially handling personal and valuable intellectual property we must provide methods to hide this information. A possible solution is to offer the creator complete choice of how the documents are stored, licensed and who can view or perform operations on them. Then the implementation must provably adhere to the strict security model. An alternative proposed system is to give three categories in which to classify data: {show, hide or apathetic} [27], which would help reduce inconsistencies with the desired and actual visibility.

2.2.2 Present Academic Social Networks

There already exists online social networks designed for scientists, some of which achieve our goals and in various ways. I have compiled a comparison chart of the more popular sites including ResearchGate, myExperiment, Academia and Menderly with links found in Appendix A. Only a summary of comparisons is presented due to the vast number of evolving complex features, because of this the reader is invited to visit the websites for more up to date information. The two biggest websites I covered were Academia and ResearchGate, each with over 1.7 million users worldwide. However I would argue that these two were not the most feature rich, without an official way to query their website through an API or with dubious privacy settings. They all have some form of secure login, which is a main objective of ours, and there are methods to allow third party applications, such as Facebook or Google, authorization access using the open OAuth 2.0 protocol [28]. I found positive correlation between site popularity and how clear it was to connect via this method. An objective of our system is to allow for tagging, this is also enabled by each of the sites I visited and is used to sort through documents quicker by increasing their visibility. Something I have not considered yet is that authors already have existing works and publications, these can be found by searching the appropriate databases and collecting the information [29], see column 'Material Linking to user' Appendix A. It's assumed these sites overcome the many security, administration and storage concerns which we face. Other findings are that only myExperiment seems to provide a crediting system, complete privacy controls and licensing options. At ResearchGate university email domain authentication is present and social networking features were standard

throughout each website. Interestingly, I discovered that there was little platform support for content driven discussions, or any integrated mathematical symbols. Other learning driven environments without social networking features however do exist that have this support [30, 31].

2.3 Web Applications

2.3.1 Internet and Networking

2.3.2 As a Service

So far we have explored the social encounters of education with networking environments and the relevance towards our system, now we'll be concerned with the composition of web services. Modern networks are made up from connections between different nodes; they communicate using protocols to transmit IP packets between them. These are the standard data unit and are sent on the network layer of the Open Systems Interconnection mode. They are packet switched though hubs and routers to the correct location using an attached address. These addresses are in the form of 32 bit numbers (version 4), but which the Internet Assigned Numbers Authority [32] has run out of so 128 bit replacements exist (IPv6) allowing 340 trillion, trillion, trillion addresses. To setup a server a static IP provided by an ISP is sufficient to be visible on the Internet. A request from the client will be sent to the server which replies accordingly: with IP address and port number then perhaps by returning data by contacting the database management system. This traffic during a session can be made secure by encrypting data on the secure socket layer: digital certificates bind a servers {public key and identity} and can be verified for communication.

The Internet describes the hardware and software which joins computers in the world together, but the web itself is an application which runs on this framework. An application not on a server but over the Internet can be constructed in a browser supported language or could exist on the clients computers themselves and thus need continual updates. There have been substantial advances to guarantee quality to web end users, one notable technology is HTML5, a markup language still being developed because it needs to conform for humans and computers and encompasses many technologies, each laying the foundations of the web [**?**]. The appearance of a website page is determined by associating the HTML elements collected in the Document object model (DOM) tree structure with CSS style tags. The behaviour of a webpage in a browser can be controlled by using the language JavaScript. Examples are to implement form validation or to query the server and provide services without the need to refresh a HTML page.

The ability for web services to send independent and unsynchronized requests is known as *statelessness*, and is achieved by attaching data and context to HTTP headers. Using this a web server can function as a web service, with its entire application operating a network stack. Exchanges between web services then operate different layers of the stack: transport, messaging, description and discovery. For example the transport protocol defines mainly HTTP operations. In attempts to encourage good protocols for this exchange various specifications have been devised. The Simple Object Access Protocol (SOAP) allows for requested data to be formatted into XML over several transport layers using independent packaging. This model had been mostly replaced by the uniform interface *RESTful* web services [33]. REST is an attempt to standardize correct methods of usability, educating both the client and server promoting the following principles: 'Use HTTP methods explicitly, be stateless and expose sensible directory structure'. For example a HTTP GET request should only be used to retrieve a resource, avoiding unintentional server side changes. The REST application framework can be deployed in many ways; an attempt to restrict its use is the constraint layer 'Hateoas'. This stands for Hypermedia as the Engine of Application state, which is exactly what it promotes: REST clients connect to the server and receive nonlinear hypermedia. From there on control flow is determined through the representation of the files. This idea is important for the scalability of distributed networks, the client should know where to navigate between resources and whether or not it should store or cache information received. Web 2.0 defines the ability for users to generate their own content to promote participation in the web. The next step is to harmonize information meaning across the web so that resources can the web so that content is more useful. This concept has been described as the *Semantic Web* and is made possible through many technologies combined.

2.3.2.1 Open Source Licenses

As part of the system design it's important to identify existing available solutions and assess the functionality that is provided and the legal scope attached. As with all creations and developments there needs to be a consideration as to how the law will affect its existence and relation to current object populations. With regards to software, a license acts as the legal instrument which then determines how it's subsequently distributed and permissions that end users are granted. Under the [34] convention, original software is automatically assigned copyright protection for the programmer, however this can be disclaimed if it is to be placed within the public domain, not to be confused with 'free' and 'open source' licenses which contrast with proprietary modes. Therefore there exists descriptive software that has the designs made available freely, this movement is described as 'CopyLeft' and is promoted by the Free Software Foundation. The particular group of licenses which we are interested in are GPLv3, LGPLv3 and AGPLv3[35], with the latter

providing end users over a network power to view and modify the source code, and must be distributed under the same terms.

2.3.3 Social networking frameworks

In order for us to be able to accomplish the objectives we first need to choose a suitable set of tools that can accomplish the task. There are many factors affect our choice such as licensing, documentation, community support and time constraints. Additionally it's even more challenging to choose carefully given a time critical implementation. However there are many implementation challenges we face that constistute the foundation of a web application and have been solved time again. An example of one of these tasks is being able to securely login with a password and username, and there exist many libraries and frameworks which allow you to code a tested solution effortlessly, thus saving time. Communities have built more advanced versions of these that supply free pre-built social network structures such as those found in table: 2.1.

Table 2.1: Available PHP Social Networking frameworks

Name	Type	Docs	License	Description	Language	Popularity	Score/10
Elgg	Framewok	Good	GNU GPL2	SN engine	PHP 5.2+	3 mil d/l	7.5
Oxwall	Framework	Poor	CPAL 1.0	Comunity Soft	PHP 5.2.6+	200 d/l per day	5
WallFM	OxwallPlugin	See above	CCASA 3			30d/l a day	4
Etano	Simple script	Scarce	MIT	Dating site	PHP 4.4.0+	20 d/l per day	3
Beatz	Simple script	Scarce	CC GPL	SN Script	PHP 5+	Joomla CMS	3
Dolphin	Product	Limited	Cust License	Drag + drop	PHP 5.2+	3.2 mil d/l	2

The most useful choice we found was Elgg [36], which provides blogging tools out of the box, but has a more stable content agnostic file sharing approach. A notable dynamic content management system is `http://wordpress.com/`, both Elgg and WordPress are licensed under the very unrestrictive GNU GPLv2 and written in PHP, and the later exhibits data tagging and advanced blogging ability through a simplified interface. Some users even report on using it for research by setting the visibility of the posts to private and then accumulating research when discovered. As the source code is completely available modifications can be made to add additional features and there is vast support for making plugins. The effect of such a venture has resulted extensions like `http://buddypress.org/`, this allows for all the expected social networking extras such as friends, messaging, profiles but also private discussion forums and activity streams.

2.3.4 A decentralized free web

There are no requirements on having a centralised designated server and computers can connect directly in a peer to peer (P2P) fashion. An interesting idea is that

of decentralised social networking where all connected devices communicate and compute between each other like in P2P. The original vision of the web from its creator Tim Berners-Lee was that of "an interactive world of shared information through which people could communicate with each other and with machines" [37]. This is not in tune with the server being the central node and products alternative to this have been subject of huge attention [38]. A pure P2P network would have all the computers providing resources, so the storage capabilities are potentially unlimited. This premise has led to large scale file sharing which renders participants frequently in breach of copyright laws [39]. The approach gives clients freedom to establish their own authority on their data and how it is used, but P2P networks require careful planning to avoid security, content management, and latency issues [40]. The main disadvantage of this approach, which is inherent in the nature, is that there is no authority of content between the peers, and features to provide misuse detection disproportionately increase delay on the network. Therefore confidentiality typically involves technical means to accomplish, for example how do we backup private keys over a network of potentially untrusted servers? One approach is to delegate trustworthiness between servers, thus avoiding multiple adversaries teaming up to break the traditional threshold cryptography [41]. Therefore for performance the security must be chosen for P2P to be as minimal as possible, an unnecessary overhead is to encrypt publicly viewable posts.

2.3.4.1 Diaspora

An example of a hybrid P2P social network is *Diaspora* [42] , which becomes distributed at the server level, at nodes called pods. If a user wants to join the network they can just connect to exiting pods, where a maintained list of attached users and their files is kept. Peer to peer distributed technologies such as `http://www.gnu.org/software/social/` typically encounter several key issues: how to join the network, how to publish a file or search for one and then fetch its content. With Diaspora a search for a user performs a PULL request to get the basic info and their public key. This key is then used to encrypt private messages that can be sent and read by only by the recipient in possesion of a corresponding fresh private key. This is the same for all types of content in their framework and the container objects can be changed if needed. The quality of connections to pods can vary, not to mention failures can prevent persistence completely, therefore cloud storage for files is a serious issue. Although cross virtual machine attacks on the cloud, such as amazon Elastic Compute [43], required co-residence checks or blinding to prevent information leakage. The code is currently open source under the AGPL3 license, and the design team urge users to establish their own pods. Actually Diaspora works on the *Ruby on Rails* framework stack, which is described as Java meets PHP and which there exist some good books for learning [44]. In fact the ability to source friends from FaceBook constitutes a large number of new

members and can sometimes overwhelm a self-hosted server. The maintainer of a pod should ensure that their system is secure; this is analogous to the traditional system administrator. An extended implementation of Diaspora would require management of sensitive data using sufficient encryption; this is because the security rests in the hands of the server owners, who emits trust through experience. Yet there is no synchronization between the pods and if an attacker identified a pod as a point of failure, gaining subsequent access, then the security of the entire network would not be compromised. As the pods access control is maintained by the system administrator and adequate protection against processing of the users content cannot be guaranteed by existing services. It could be in our interest to align security policies between several nodes of the Diaspora network, a separate fork of the current git repository [42] could allow us to create access control and all our other required features.

2.3.4.2 Friendica

Another tool that gives users the chance to seize control of their data and privacy whilst encouraging sharing is http://friendica.com/. Again written mainly in PHP, It has features akin to Diaspora and is open source, but emphasizes consolidating existing social streams. Unique privacy tags on feed items such as 'private' and 'allow' notify the DFRN on the location of resources within the site. The owner of data items or feeds has the final say on how and by whom the item is accessed, this is truer in the sense of central servers as now the data is actually located on home devices. To alleviate trust issues more complex authentication procedures are required, the security layer 'Rhino' uses AES 128 bit with some form of challenge response validation.

2.3.5 Alternative Frameworks

We realize our system is very bespoke and by choosing a heavily developed framework this will abstract important computation that we may need to redesign ourselves. A substantial amount of the design process was to experiment with the open source solutions I have mentioned, and to see how adaptable is the programming language and the design patterns used. Simple community creators such as [45] are common, but these limit the ability for customization. We decided using pre-built samples is clearly not always the best way to go, as time can be spent forcing the system to do things it's not designed for. More appropriate is to use a framework that supplies with pre made database and validation methods. Some examples of PHP frameworks, with a comparison of features present is http://www.phpframeworks.com/.

2.4 Storage

It's fundamental to our success that we have some means to store persistent website information. One way this can be achieved is by using a database or structured collection of records that can be queried and retrieved on request. A preliminary requirement of our website is that it is driven dynamically by data; more precisely this means our website acts as an interface to a database for manipulation and presentation to the end users. The entire database system will consist of data, hardware, software and the users, and we'll need a method to manage the creation of data and control access and this can be handled by a Database Management System (DBMS). A relational model further abstracts the physical details and provides a logical data model so we can define the data structure, integrity and operations. A further method is required to synchronize concurrent manipulation of research material, and for the solution to be sufficiently fast and preferably open source.

If we model the size of a node as representing the how rich in features it is, the web has shown us that the traditional concepts of thin client/fat server and fat client/thin server no longer seem to have such distinct boundaries. The choice of where to process information as well as where to store it is vast. The recent explosion in cloud computing services has allowed for extremely accessible and scalable storage not just in terms of flat file mediums, like .txt files in Amazon S3, but more recently an entirely relational database in the cloud (RDS) [46]. Web applications now have the choice to entirely abstract away the complexities of infrastructure requirements through an 'Infrastructure As a Service' (IaaS), for example `http://www.heroku.com/`.

The web application proposed will need to support the storage of the relationships between the entities, such as an academic and their publications; it also must provide a scalable solution to store uploaded documents. Databases storing assets for delivery tend to be scaled vertically as opposed to the web application scaling horizontally in that the database 'box' tends to increase in size (more RAM, more processes) as the database grows. This is opposed to the application growing by adding more servers to load balance HTTP requests. Vertical scaling can continue up until current physical and technical implications limit its growth; at this point sharding can be employed [47]. Backups, maintenance and installation would all need to be tended to.

To reduce the risk that personal data is lost, it important certain disaster recovery steps are taken. Daily data backups should be kept to avoid tragedy, but need to be stored in a secure location, preferably geographically. This process should not interrupt access flow to data, which must also be protected. Since we represent a potential future project, technical measures need to be currently addressed to prevent against unlawful processing of data. For example a potential attack is to

intentionally crash the web server and then read the core dump that might contain private information. In this case the database can be taken off site and backed up on a separate provider such as [48]. A managed alternative is Amazon RDS [46] which deals with backups and scaling issues with more autonomy. Taking advantage of the number of growing *NoSQL* providers could solve many of the Big data scaling issues that could likely arise. *NoSQL* databases give the major advantage in that they defeat the scale up paradigm: now database servers and transactions can be split in parallel across multiple servers.

The storage of documents like PDF is another issue, and although it could easily be handled solely with a relational database, by storing records in 'blobs', but it makes more sense to use the file systems storage as it is designed for file operations. An option for document storage is the open source MongoDB [49], which stores structured data clearly in binary JavaScript object notation, another open source item for use is [50], which provides a distinct functional alternative.

2.5 Security Evaluation

Data must be subject to security considerations of some form, and there are many resources within the system that need control of access; . The requirements of the system design will be met with security mechanisms and a formal description of a security model should be provided that not only reflects the potential intentions of users but can be proven to be secure. Securing data should be considered as an integral part of the design process and leaving it as a later consideration, after flow procedures are selected, could be catastrophic. In spite of this, as I have breifly mentioned, its important not to provide excess security as it may affect performance. In this section I will describe accepted methods for providing protection between application and hardware layers. Eventually the platform independent security procedures will be implemented and provide the requirements of policies that describe subjects, objects and operations. Main areas of computer security involve confidentiality of data, for example cryptographic encryption using keys ensures that only authorised users can view the information. Another theme is integrity of data, whether or not its valid and has actually originated from the specific location. Finally access control can be enforced by authentication where the user has to provide something held, known or that is part of them. Accountability of actions on a system is also an important consideration, as users must not be able to repudiate a signature. This can be achieved through audit trails of network logs containing evidence. There are common mistakes made when securing web applications like not validating input, this can lead to buffer overflow weaknesses or cross side scripting attacks from malware. We must prevent these as a priority and not patch the system up later, this is a primary awareness goal for the new UK Software Security, Dependability and Resilience Initiative (*SSDRI*).

2.5.1 Authentication

We need to be able to verify users of the system for use when deciding whether to grant access to a specific resource. The most common form of this is a known username and password, these are popular because they are free and not dependent on location. The users identity is verified at the start of a session and then used to provide entity authentication for further access control based rules. Passwords should not be stored fully in a database, but encrypted instead to prevent obvious attacks. Even better is to use a collision resistant cryptographic hash function, this means it's computationally infeasible to find a different input that can hash to the same output, whereby learning about the input password. Also a good idea is to add random bits, called a salt, to prevent creation of a lookup or rainbow table. Standard web based login pages display information about the service and provide an option to register and create an account. Approximating a HTTP POST login request to the server to be of size 500 bytes with a dedicated 1Gbit full duplex server handling 250,000 requests a second, an attacker could break a 6 character alphanumeric password [A-Z,0-9,a-z] in $\sim$ 62 hours. Therefore it is not desirable for a server to be able to handle this many requests, a countermeasure is to limit login attempts on an account or from an IP address. There is little point in using passwords if HTTPS is not enabled as eavesdroppers can just view the passwords. Passphrases of large sizes can be difficult to remember so revocation is a cheap email operation, although they should be reset frequently. An argument against password policies is that attempts to measure strength regularly ignore populations of common patterns vulnerable to dictionary attacks [51], and that user education is vital to raise awareness of socially engineered attacks like phishing or key logging. In practise brute force attempts can be prevented by limiting the number of attempts, or setting extra challenges such as the Captcha mechanism [52]. The Captcha check, created by Von Ahn, attempts to set tasks that Artificial Intelligence struggles to complete, and so can be used to prevent non humans or scripts from overloading the server processor with requests or database with entries. A later creation was reCAPTCHA, which uses successful Captchas to digitize old texts from before the computer era. The 'Web Content Accessibility Guidelines' `http://www.w3.org/TR/WCAG/` aims to increase the usability of the web to those with disabilities or persons of age faced with difficult media, such as audio format for Captcha. By using challenges like Capture we increase the barriers to entry but user satisfaction is thus reduced, however DoS is a legitimate concern and this a small price to pay for protection of services. There is no denying that entity authentication is essential in an environment where users constantly share data; otherwise the fundamental assumptions about the system perish.

Another form of authentication can be established by using the uniqueness property of an email address. The email is used to generate a public key which can be authenticated using SMTP through an independent server [53]. To reduce the risk

of a man in the middle attack emails generated from our domain name could also be authenticated [54]. This idea has lead research into consideration a viable alternative to the Public key Infrastructure [55]. New functionality for maintaining networks is presented from IPv6, such as the stateless auto-configuration, where nodes check to see if an IP address is available on a LAN. This unfortunately is vulnerable to DoS, similar to *ARP* spoofing, as a corrupt node can pretend to be another or in a state that is not. A possible solution to this would be to contact a server that records the bindings of the current IP addresses used by which hardware.

2.5.2 Protection Mechanisms

There are numerous security concerns that will need to be addressed; I will cover a few of the key ones here. We cannot assume any end point on the Internet is safe as botnets are commonplace [56]. The capabilities of these new threats cause problems for many real systems, for example DoS attacks can render utilities unable to perform. These can be intentional such as flooding a TCP handshake with SYN requests [57], or through legitimate use such as over popularity. One solution to mitigate these potentially distributed attacks is to install a firewall that filter packets through certain ports and IP addresses. Highly sophisticated attacks can occur over commonly open internet ports, such as HTTP or port 80. Therefore stateful firewalls can work out if the recipient is in time with the handshaking [58], perhaps by using a statefull validating ‘syn proxy’.

Chapter 3

Work carried out

3.1 System Planning

3.1.1 Development Environment and Data structures

We have decided to use open standards and popular open source software, with adequate community support and for maximum portability of any code. The chosen programming language which we will be using to develop the server side application is the general purpose *Hypertext Preprocessor* (PHP) [59]. This is because even though its weak and dynamic typing generates slower code that's tolerant to errors, it's a very interactive and open source option with which we have experience in. At runtime scripts replace the PHP with HTML, this happens once the parser has compiled the code between the PHP delimiters into bytecode for Zend framework. The server will need to be configured to understand PHP and have the permission to run scripts. There have been concerns with scalability of PHP and its object orientated scripting design being too slow, but workarounds such as [60] exist. The main data structure we are using is a relational database and there are many options for Relational Database management systems. Presently the most popular is *MySQL*, this is also our choice because of its scalability, allowing multi-user access and compatibility with LAMP stacks under the GPL v.2. For that reason our development environments are made up of *LAMP* and *MAMP* stacks (Linux/Mac OS, Apache HTTP Server, MySQL,PHP), differing mainly at the physical design layer to reduce testing inconsistencies between ourselves and future deployment. We've chosen a compromising PHP framework *CodeIgniter*, known for its small footprint. This allows us to code freely in the style we prefer and design entire modules from scratch. We will be working as a team and there are great tools that allow us to collaborate co-operatively during development. Once we have planned the requirements and begin to program the source code, we have chosen to manage a distributed revision control system called *git* [61],

that will track all versions, called *commits*, into repositories. Assuming a centralized server model, a client can view a page by sending a HTTP request, which encapsulated the URL and parameters, to the server. The client connecting on the browser is then sent the required HTML needed to generate the page, along with any JavaScript or styling for presentation. On the server side a lot of this content is dynamically created to fit the needs of the user depending on the data they have sent our server already. The application is going to be written using a Model/View/Controller (MVC) design pattern. The application logic is located in an abstract *Model*, which is then requested from the controller which handles input and output to the view. MVC is described as a user interface that decouples these three objects for reuse and flexibility [62].

For example the main login page that users are presented with is generated by the server controller classes, which are connected to the models that communicate with the database using SQL. Then, in reverse, the relations from these models reach the objects in the controllers that are loaded into a view for a user, or details see figure 3.1. Another choice is to use the CodeIgniter framework (CI)[63], which adopts this design pattern and demonstrates high performance with the freedom to select coding style under the *Open Source License 3* [64].

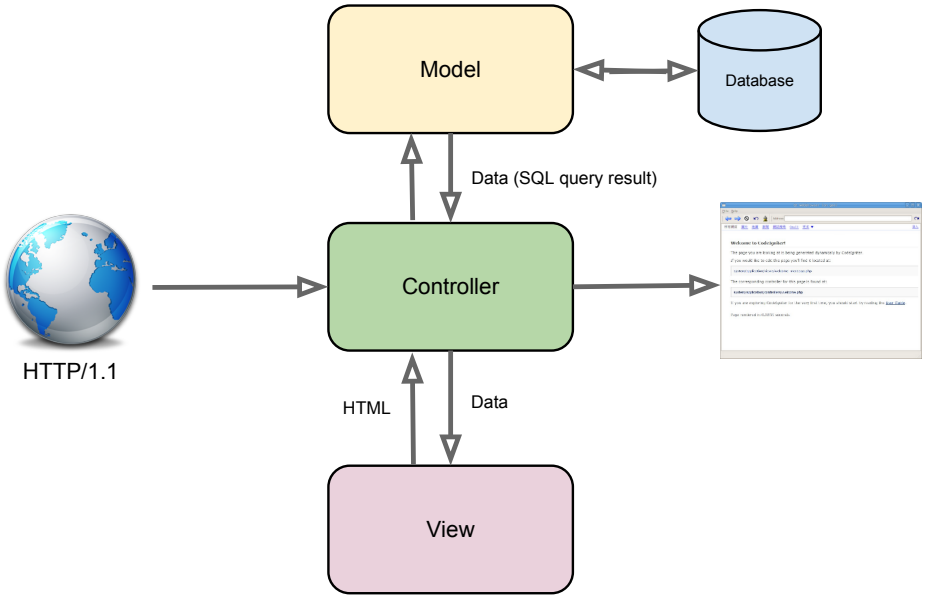


FIGURE 3.1: A theoretical illustration of the MVC design pattern used in our framework

3.1.2 User Database initial schema

I have described the main idea of our system, and the requirements can be further abstracted from Chapter 1. Nevertheless prior to engineering the software it's unclear which features are realistic, achievable and within our scope. Referring to our list of primary objectives, the first challenge is to construct a method of secure login. This requires a form of organized permanent storage, which will be provided by a relational database, namely the MySQL database.

The *User* database relation requires the following preliminary schema. Every attribute is of data type string, apart from *UserID* (UID) which will be an integer.

```
User(UID,username,password,email,university,firstname,last_name)
```

We have decided that the *username* will be a user's email address instead, reducing signup time and removing the need to remember a new username. However the downside to using email addresses as usernames is that it is more public than necessary, revealing information that could be used to hack into an account. It's then important that we respect a user's right to withdraw visibility of their email contact details from our site. An idea to encourage legitimate academics to the site was that authors that are not registered can be emailed questions asked about their papers; this could be achieved by extracting contact details from the papers and giving it to the users. A reason not to do this is because not only will it be disclosing private information, it might also be giving away future account usernames. A similar service could be offered whereby an email server could offer to send the questions via proxy to the authors email accounts. However we have not pursued this route as under UK legislation "a person shall neither transmit, nor instigate the transmission of, unsolicited communications for the purposes of direct marketing by means of electronic mail unless the recipient of the electronic mail has previously notified the sender that he consents" **Privacy and Electronic Communications 2003 s.22(2).**

An instance of the user relation can be uniquely identified by a primary Key called ID, this will be automatically incremented from the previous count upon database insertion, and will be used in reference to any associated user data. One such relation will be the profile page table, which should strictly be contained in the user table as it associates data which is unique to a user, however this table may contain heavy amounts of data and it will be more efficient to search just through the user data for authentication.

3.2 Preliminary Security Implementation

3.2.1 Email Address Domain Authentication

During signup it is desired that a user can only login by using a known affiliated email address from their current institute (*university, college or school*) as mentioned in the objectives. This can be achieved by linking their now *email* (username) with the institute name they have supplied. The validation step needs stored the institute names and the allowed domains for each Institute. An institute might have members which use different email address domains, like the University of Bristol which allows the use of 'bristol.ac.uk' but also 'my.bristol.ac.uk' under a new university for life scheme as part of the 'Going Google' project [65]. To be able to check a match: a relationship with cardinality many to many 'Domain' to 'Institute' is required. These tables all have integer keys and to model this in SQL a bridging table is required. This table consists of two keys: *Institute key* and *Domain Key*, which together form a composite set of foreign keys. The final tables are:'Institute', 'Institute to Domain' and 'Domain', an example set of tuples is given in figure 3.2. The validation check uses a custom call-back such as the sample in listing B.1 and the associated active record calls are in B.2. There are over

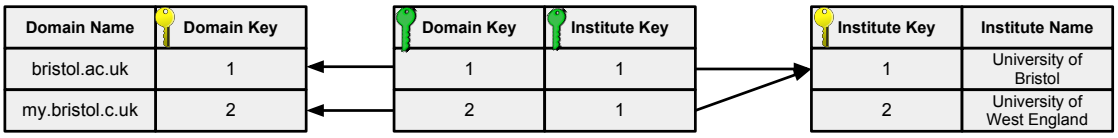


FIGURE 3.2: Example instances of the authentication email address domain names

300 Universities in the UK, each of which has at least one format of distributed email address domains. To accumulate this data we could seed the database with web scraped data, but this might violate intellectual property rights specified in a sites terms and conditions. A better approach is to allow potential users to request an institute and then appeal for permission from that establishment.

3.2.2 Signup and Login Forms

Before we can test to see if an email address is valid or not for an institute we need a way of accepting candidate registration data. A common method to accumulate this data is for the user to fill out a HTML form, consisting of different input fields and buttons, which can then be submitted using non-safe and non-idempotent HTTP requests such as POST [66]. The headers and body of the HTTP requests are broken down into packets and sent using TCP/IP to the address returned from

the DNS protocol. There are several things our server needs to check before we can safely register a user so that they can login and start sharing research.

3.2.3 Validation

The first step is to validate the user data, this is to restrict the type of information which they are supplying us with. In our application the 'auth' controller is handling these incoming requests and a 'create' method demonstrates a canonical validation procedure. The code that expresses not only the chosen input fields but also the proposed validation is given in Appendix B.3. For example the username should be a valid email address, and must be filled in. Also the password length is set to be between 8 and 20 characters.

3.2.4 XSS Protection

The input data is also subject to an 'xss_ clean' procedure, this is to prevent one of the most common attacks known as cross site scripting (XSS) where code could be injected into our application and enjoy elevated trust when run on our behalf. Unfortunately a resulting payload of an XSS vulnerability, persistent or reflected by the server, can execute any code a browser can handle, so could be designed to hijack user sessions [67]. Even though huge amounts of research is performed in this area, even giant sites like FaceBook have suffered from XSS vulnerabilities [68]. They are labelled as the second most critical web application security risk by the Open Web Application Security Project [69]. Popular browsers such as Chrome and Firefox (76.6 % of browser usage in July 2012 [70]) have built in filters to protect against the attacks, but these open source solutions are not sufficient to ignore defending our server against persistent XSS attacks [71]. In summary the 'xss_clean' tag protects us by essentially converting special characters into HTML entities, properly encoding them so they are not parsed, see 'htmlspecialchars' [59].

Throughout the web application many input fields are checked and sometimes we would like to accept HTML tags, but with currently mentioned input encoding this is not possible. A solution is with the use of an up to date and permissive 'white-list' allowing for standard compliant HTML entries, one such helper class is called *Html Purifier* [72], which safeguards against filter weaknesses in the application framework such as those discussed by Weinberger et. Al [73].

3.2.5 Signup Form Sanitization

An example signup form is seen in figure 3.3, which shows an attempted login that has been rejected for validation reasons stated. The warning message and its reasoning can be computed by a client's browser using JavaScript, or by our server. The latter option is essential as the JavaScript validation can be bypassed simply by constructing a custom request to our server. A final validation step is to sanitize the inputs, thus preventing SQL injection attacks or malicious scripts from damaging our server [67]. The likelihood of these attacks is high, and even the domain 'mysql.com' has been targetted successfully by a blind SQL attack back in March 2011, where highly regarded professionals usernames and passwords were disclosed [74]. To this end we're using active record for database query handling, which escapes from untrusted special characters before inputting SQL into the database [75].

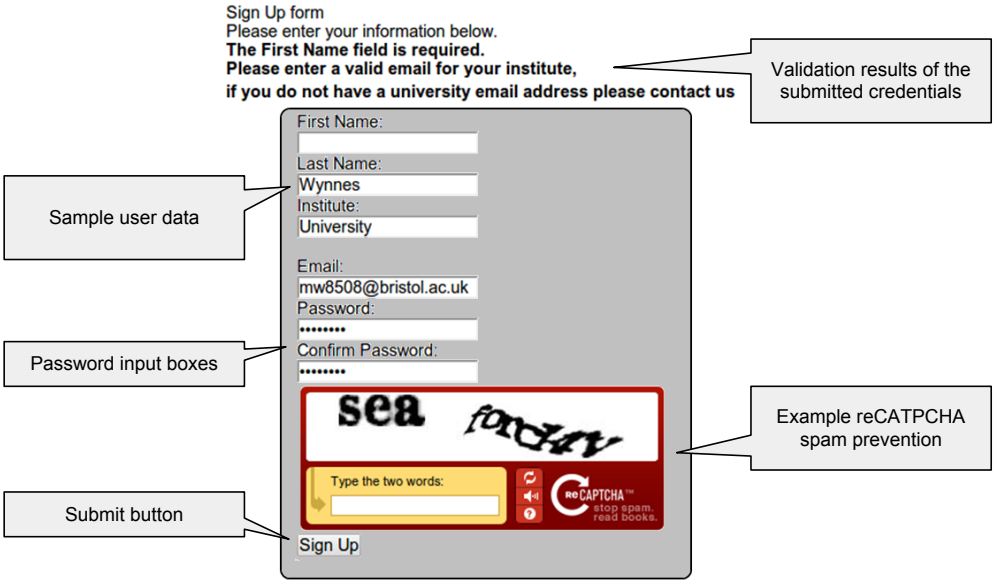


FIGURE 3.3: An example signup page form structure

3.2.6 User Authentication

The user attributes must be obtained securely on account creation and the fields ought to match those in the preliminary user relation table. We could construct a secure login system completely from scratch, but this would distract us from further implementation and community tested implementations already exist. The authentication library that we are choosing to use for this purpose is called *Ion Auth* [76]. This uses cryptographic low level primitives from PHP MCRYPT [59] or a CryptoAPI [77], such as encryption and hashing, to provide basic web

application user security. For example a general process for the creation of new users and the logic for logging in is supplied, but we'll decide how and where this is to be completed. Ion Auth is released freely under the Apache License 2.0 . This license is sufficient for our modification because any derivative work may be reproduced from the source, assuming the changed files contain the license and a notice saying they have been changed, 4.1-2 [78].

When a new account is created, we would like to certify their credentials as much as possible. In particular the email address they provide will act as a major and primary source for communication, so it's necessary the email account is accessible by them. Additionally we need to know that they are in current ownership of the account so that we can match the domain against affiliated Institutes, as mentioned earlier. Therefore they are sent an account activation link via email. To perform email activation and send forgotten password emails we use CI's email class, itself using the default PHP 'mail' class[59]. For local testing of this feature the chosen mail server is Google's Mail SMTP service, with SSL on port 465, until the site is deployed and a permanent website admin address is established. A sample of using the Ion Auth framework to reset a password is shown in 3.4, where the forgotten password form is at the beginning of the request and an expiring token is sent via email. Something not included in the diagram is that only one reset request is permitted until the last one expires, where the timeout set at five minutes, this is weak protection against an email account being bombarded by a DoS attack. When a user signs up to the website the activation link email, which is double

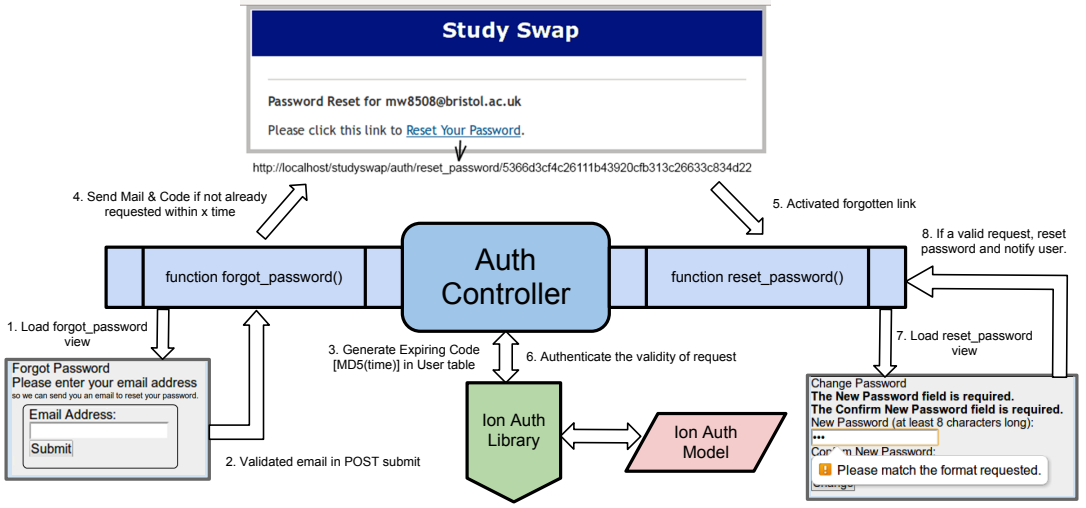


FIGURE 3.4: The sequence of events and steps required to reset a User password successfully

opt-in, has a similar appearance, and the HTML mark-up and CSS elements are dynamically generated in PHP to be cross compatible with multiple email clients [79].

The views, controller or models can be changed at any point in time, and the whole process would still function as the individual objects do not to be aware of the change. This framework adopts the MVC pattern, which by definition decouple the objects structures also known as the 'observer' design pattern [62]. For example the Ion Auth Library could be entirely altered with new methods to generate the forgotten links, such as a more random hash seeds, and the controller and forgot view objects would not need rewriting.

3.2.7 Spam Prevention

On user creation we need to protect ourselves from large numbers of pseudonymous account creations that could maliciously exhaust server resources. This is known as a *sybil* attack and there are unfortunately few ways to prevent this without reducing the user experience. An initial attempt in this direction is to employ a key that only humans can obtain, namely the 'Capture' system mentioned in background research. This requires a way to generate the pictures on the server, with a database table to link the current user's attempts to the capture images and their actual answers. We have found a default application of this mechanic in the CI helper files but soon found that the same effect can be achieved by querying the *reCAPTURE* API [80], thus reducing the need to manage this service on our end. There are two places that generally require this protection, the login and signup forms. The signup reCAPTURE is enabled by default on both and upon failure to pass the check the form submission data refills the fields, but the reCAPTURE image changes. The appearance of this on the login page however is more discrete and only comes into activation after x login attempts, using the new 'login attempts' database table. This prevents attacks against weak passwords, where there is debate over the term *weak*, by reducing the rate of password guessing to x login attempts per account, at which point further slowing guesses by adding the reCAPTURE check. An could hop to hacking another accounts with x allowed guesses, but the problem domain is refreshed and by further logging sessions we could signify the identity of the offending intruder.

3.2.8 Administrator Control Panel

To simplify adding new institutes and email domains for the email authentication procedure an initial admin account has been created with typical credentials. Also an admin user control panel form as seen in figure 3.5 is supplied as the default page when the administrator logs in. Other options visible are activating/deactivating an account and creating a new user. The main purpose of this form is to facilitate adding new email address domains, possibly sent on request, into the database. A list of currently available ones can be seen with the help of the dropdown element,

the location of this view is auth/index.php and is generated from the 'admin page load' method in the auth controller.

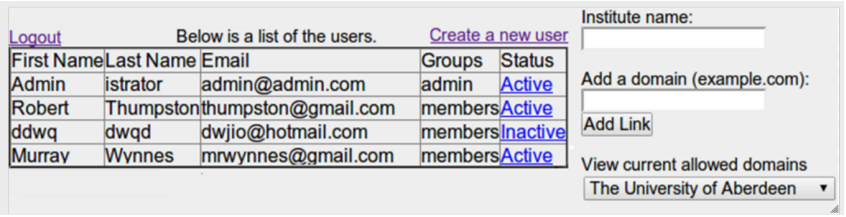


FIGURE 3.5: Administrator control panel showing the 'User table' and 'Institute Email Domain list'

3.2.9 User Passwords

To reduce the cost of authenticating a user we're using a static password which is known and remembered by the user. To diminish leakage of this secret it's important that our system stores no knowledge that can uncover the actual password. Cryptographic one way hash functions can be used to achieve this such that guessing inputs is the only way to break the hash function, i.e. it's computationally infeasible to compute the pre-image of the hash. Assuming H is a hash function acting as a random oracle then $H(p)$ matches an arbitrary length password p of length l, to a fixed length hash of length m uniformly and independently distributed between $[0, 2^{m-1}]$. The function H is guaranteed to have collisions as $m < l$, also only $2^{\frac{m}{2}}$ guesses are required for a successful brute force attack, this is an example of the *birthday paradox* [81]. An example hash function is the *Message-Digest algorithm* (MD5) and it's frequently used even though it has been proven not to be suitable as its not collision resistant. The algorithm has been shown to require $2^{39} < 2^{\frac{128}{2}(ideal)}$ hash operations to brute force by M.Stevens et.al [82]. The *Secure Hash Algorithm* family are other candidates, but a differential collision reduction attack against *SHA1* was demonstrated against in 2005 by Prof. Xiaoyun et.al [83], with only $2^{63} < 2^{\frac{160}{2}}$ hashes needed. The *SHA2* collection of algorithms have varying bit sizes and prevent the previous attacks; they are practically secure against known public cryptanalysis and as an example are used in TLS [84, 85] A replacement standard to SHA1/SHA2 in the SHA family of hash functions was announced by the National Institute for Standards and Technology (NIST) in 2007 [86], however there are still candidate finalists in the selection process and the chosen algorithm has yet to be announced.

Our choice needs to consider the possibility that an attacker somehow obtains a hash or cipher text and is allowed to perform an offline attack with unlimited power against the scheme. In practise public key encryption such as SSL will obfuscate transmission of data and authenticate users. Likewise database encryption

methods can prevent reading specific data instances, but if we have a relational database stored in the cloud then it's entirely possible a password hash output could be exposed. It needs to be as difficult to guess part of the password as it is to guess the whole thing, and each password should be equally secure when considering predictability from password distributions.

3.2.10 BCrypt Security

Therefore we have chosen to implement BCrypt [87], a version of the 64 bit block feistel Blowfish cipher which for 16 rounds repeatedly replaces subkey inputs with its hash output. The 128 bit random salt is integrated into the storage of the password, this is considered a large enough salt space to prevent an adversary using a lookup table. The key schedule *eksblowfish* has a parameterizable expense by running for at least 64×2^{cost}, and it is a one way hash function as "no part of the algorithm can be precomputed without both salt and user-chosen passwords" [87]. It's designed to scale with hardware advancements by tuning the security parameters and increasing the work factor of the function. The implementation itself is efficient because it can use fast register operations like shifts or XOR. However its adaptive so can be chosen to run significantly slower than alternative algorithms thus increasing practical security. This is necessary to thwart modern General purpose computing with graphics processing units which are perfectly suited to implementing this arithmetic in parallel over many cores, an indispensible requirement considering as of June 2012 the world's fastest computer 'Blue Gene' can operate at over 20 Peta Flops [88]. Even though the password size under BCrypt is limited to 55 bytes, the allowed passwords have a max of 20 characters so are not affected. It's possible to select weak keys for the key derivation function, although only second order differential attacks exist and no efficient known plaintext attacks against blowfish have been found [89].

In summary the aim is to increase the time to perform a lookup of the (passwords and salts), and any chosen method increases the time taken to login. The email activation code and forgotten password link are all outputs of the BCrypt algorithm, activated by using the function 'hash code' which takes inputs *identity* and a *UNIX timestamp*.

3.3 Secondary persistent storage security

3.3.1 Login Sessions

As mentioned HTTP is stateless and each request is an independent transaction. Two mechanisms which allow for important data to be saved between login and site pages are: cookies and sessions. Each user accessing our server generates a sessions to be stored. These have unique key identifies so we can match a user up with their details in the database:

```
`session_id` varchar(40) NOT NULL DEFAULT '0',
`ip_address` varchar(45) NOT NULL DEFAULT '0',
`user_agent` varchar(120) NOT NULL,
`last_activity` int(10) unsigned NOT NULL DEFAULT '0',
`user_data` text NOT NULL
```

The client does not have access to this information and we can filter which parts of the user data are required from the session. The 'user data' section is the most exploited attribute in a session and contains at least the user ID, and perhaps latest activity such as temporary form choices. Server sessions are enabled through the use of a long term cookie called 'SESSID', but the sessions themselves expire after two hours for security reasons such as computer sharing. There are two reasons to store the sessions persistently in the database:

- Requested session data from the application can be efficiently searched and filtered from the database, not easily possible with pure cookies.
- Even though we can prevent prevent session fixation attacks, storing user data in the encrypted session cookie has a 4k byte limit.

If the SID is used in form parameters or stored in cookies then fixation is possible. Session fixation is the process of forcing a session identifier (SID) onto a user; an attacker who knows the SID could then steal the session. Within CI, the SID is set to refresh on every controller pass, making later use of an SID invalid. Also only sessions stored in our database are deemed as valid with their associated SID. Further we have decided to check the user's browser agent and I.P address so that only the correct sessions are handled. Aside from needing to generate a valid SID, session attacks generally require less computing power than the the higher risk of general packet sniffing. Protection from man in the middle attacks, where traffic is being intercepted between a client and our server, is given by encrypting packets via Transport Layer Security (TLS) [85]. This is the method to secure HTTP request details from passive and active attackers, with optional

authentication using X.509 certificate chains over the Public Key Infrastructure (PKI). The prefered method of using TLS over TCP is to mutually authenticate server and client to prevent the risk of spoofing attacks.

Even though sessions are also destroyed when the user logs out, there is still a window of attack. For example, imagine a user is reading a private embedded PDF document and also has access to their private messages, both features stated in the objectives. The user may be reading the paper for many hours, but their session identifier should not remain valid for all that time, therefore we've set it to refresh every five minutes. The granularity of refresh still might not be good enough, but any less is costly to generate cookies quickly even using symmetric encryption. A better solution is to encrypt all HTTP traffic on the website, this is expensive to setup, but the return security will prevent many types of attack such as the HTTP session sniffer Firesheep [90].

HTTP sessions are served at the application layer and these are not to be confused with TCP sessions that the server can issue implementing HTTP/1.1. Sessions over TCP are short term and can pipeline requests through ports. By storing HTTP sessions in cookies we need to be aware that browsers might not be secure, specifically a forged HTTP request could perform program functions on behalf of a logged on user. By following a link the user might be authenticated to perform the attack through their populated cookie session. This attack is known as a cross site request forgery (CSRF), and to prevent against this each time a submission form is loaded a random hidden token is generated to match form request with submission. This Anti-CSRF token is generated for every form using:

```
<?php $csrfhash = md5(uniqid(rand());
      form_open('csrf protected form'); ?>
```

The first line generates the MD5 hash of a pseudo random number. The second line inserts the Anti-CSRF hash token into the form but also adds it to the users session cookie for temporary use, when a request without a matching cookie and form Anti-CSRF POST token is submitted it will be invalid and rejected.

3.3.2 Random Numbers

This random number generated by *rand()* is within the range $[0, RAND\_MAX]$, where the upper bound is a prime number specific to the hardware ((2^{31})-1). The default implementation is a linear congruential generator with states of 8 bytes else it becomes a variant of a linear additive feedback number generator [91]. A more appropriate choice of generating uniformly distributed random numbers would be to use the *Mersenne Twister*, which is 4 times faster than rand() and passes the diehard test, along with k-distribution tests reporting 623 tuples per period

equidistributed in an 623 dimensional unit cube. [92, 93]. An implementation to generate random numbers using this algorithm is provided as of PHP 4 in the 'mt_rand()' function [59].

3.3.3 Cookie Encryption

Cookies represent significantly popular targets for adversaries to hack; they might want to read the contents or possibly forge incorrect data. It's imperative that this behaviour is practically impossible, and to make it infeasible the session cookies are encrypted using 256 bit Advanced Encryption Standard (AES 256) in Cipher-block chaining (CBC) mode. The encryption cipher is symmetric and designed for heavy throughput and key re-use. It's promoted as an open standard by NIST, but is susceptible to side channel attacks such as cache based time driven attacks [94] and differential power analysis [95]. However these weaknesses typically require mounting advanced sensory equipment onto the cryptographic device, and software techniques such as a firewall exist which prevent these attacks taking place in cloud environments. Cryptanalysis of the most recent attacks against AES report a reduction by half in the complexity of brute forcing AES 256 using a man in the middle type biclique attack [96]. Despite this AES is a realisticly secure method to apply in this situation as it has been estimated that it would require 50 of the world's fastest computers $1.5 \times (10^{51})$ years on average to search the 256 bit key space [97]. The actual PHP 5.* MCrypt cipher suite provides an implementation of Rijndael, where the block and IV size can vary, but we require AES mode with 128 bit blocks and must use 'MCRYPT_RIJNDAEL_128'. The CBC mode as seen in figure 3.6, where the SID is divided into n 128 bit blocks with padding and the resulting cookie value is provided by equation 3.1 [98].

$$C_i = E_k(M_i \oplus C_{i-1}), i = 1, 2, \ldots, n \tag{3.1}$$

Each cipher text chunk C_i depends on all of the previous cipher texts, Whereas in *electronic code book* mode each block $C_i, i \geq 1$ is independent of $C_j, j \neq i$ and so results in the same encryption of M_i, exposing repetitions and patterns in the message. The *initialization vector* (IV) acts as a mask to prevent pattern analysis on the first, and subsequent, blocks. For this reason the IV should be unique, this is so if an adversary obtains a cipher text and knows the implementation of the AES encryption algorithm then they shouldn't be able to determine any knowledge about the plaintext. This is known as semantic security, and the equivilent definition of polynomial security leads to the assumption that AES in counter mode is indistinguishable against chosen plaintext attack [81].

We will be using encrypted session cookies to link to user session data stored in the database, and also to prevent cross-site request forgery and session hijacking;

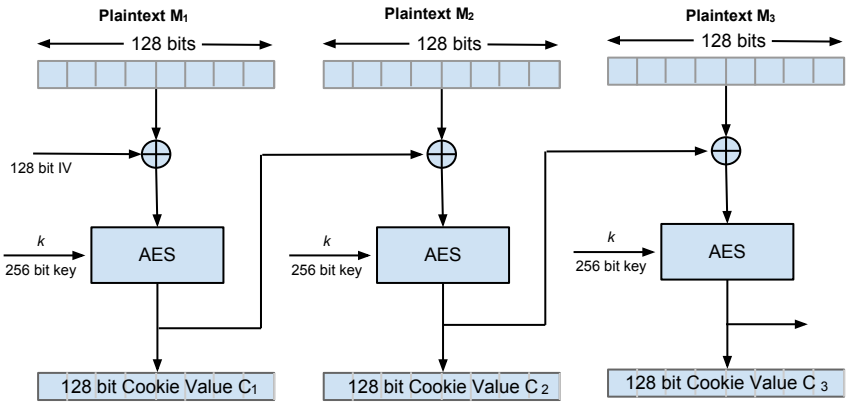


FIGURE 3.6: Advanced Encryption Standard Cipher-block chaining

also possibly to authenticate some OAuth protocols such as the FaceBook connect cookies.

It's important that we understand the need to communicate how cookies our used on our website. The original DIRECTIVE 2002/58/EC [99] is related to the 'processing of personal data' and 'protection of privacy and electronic communications'. It was amended by the directive DIRECTIVE 2009/136/EC [100] under article 5(3) stating that users shall insure the storing of information in non 'strictly neccesary' cookies is only allowed on condition the user has given their consent. However there was a debate as to the nature of this consent, and it has led to the Information commissioner's office releasing further advice and practical examples [101]. This affects our website because if we were using affiliate cookies to track user private information then we would require informed consent, namely prior or implied. Where in the case for implied consent it is important we are satisfied the user's actions are either an 'explicit request' or an 'intended expression' to store and retrieve information on their device. Our PHP session cookies however are exempt from these legislation, storing login details for the sole purpose of 'carrying out or facilitating the transmission of a communication over an electronic communications network' and therefore are 'strictly necessary'. Cookies are set to expire in 8 years, but the sessions attached are much less (2 hours), and most of the options for the cookies can be changed in the global configuration file. The built in CI session class is called by a user() function in the IonAuth model class, which is then assigned to the user variable and stored in the database. A global userdata entry, in the core controller, passes profile details around in the data[] array.

3.4 Final Login Procedure

A recap on the database shows an extended user table, now with the need to remember the following extra information:

```
`ip_address` varbinary(16) NOT NULL,
`activation_code` varchar(40) DEFAULT NULL,
`forgotten_password_code` varchar(40) DEFAULT NULL,
`forgotten_password_time` int(11) unsigned DEFAULT NULL,
`remember_code` varchar(40) DEFAULT NULL,
`last_login` int(11) unsigned DEFAULT NULL,
`active` tinyint(1) unsigned DEFAULT NULL,
`created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

A complete diagram of the steps taken to login a user is shown in figure 3.7. There are three shaded regions:

- The CI and Ion Auth authentication controller response mechanisms, which perform security checks on the GET and POST HTTP requests.
- The actual login process logic which essentially consists of trading off security against user satisfaction. This playground of processes, most of which has been discussed so far, can be repeated and tuned to increase security measures, with the aim to throttle unknown future attacks. Along with the views to serve to potential attackers or legitimate non malicious users so that they may proceed in a fashion the application allows them.
- Successful user validation and how to handle the user based on which security group they belong to. We are designing a colluder network but one that is centralized and not peer to peer, so treating misbehaviour will be dealt with by moderators and the available groups are (User, Moderator or Administrator).

3.4.1 OAuth v2.0

The authentication protocol OAuth v2.0 can be used to access existing infrastructures on behalf of a user, fetch their data without knowing their passwords and create user sessions. We have decided to implement FaceBook's Graph API, which uses the Internet Engineering Task Force (IETF) draft-ietf-oauth(v2-12)[28], and have created a FaceBook application to act as a window, specifying the requested permissions and restricted data that our applications requests. In place of a password previously needed for this behaviour, an access token is granted from an authentication server or the party capable of entrusting access to the resources it owns. Our application plays the role of the client in the Authorization code flow

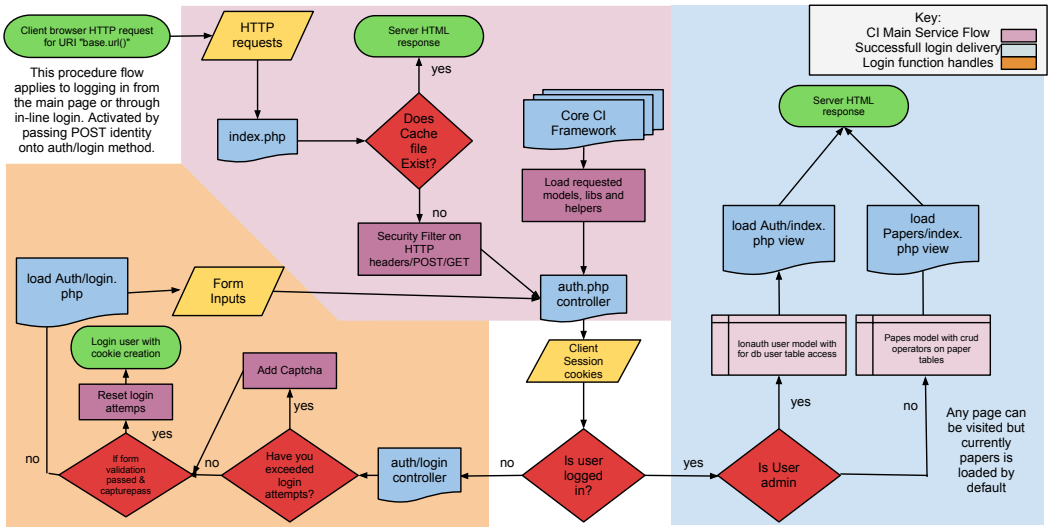


FIGURE 3.7: Procedural flowchart demonstrating how incoming HTTP login requests are processed

(s.4.* v2-12), where our server is assumed secure and can keep the secret *appID* confidential, see helper API's config file. For our application to gain an Access Token (s.1.2 v2-12) a grant such as the authorization code must be issued from the authroization server. A function 'getUserFromAvailableData', authored by *FaceBook Inc*, initiates the redirection URI with authorization code being sent as a request for access & refresh token. The application will need consent for their email address and university only, but as FaceBook has one of the largest user databases in the world there are many opportunities for future expansion using FaceBook's query language; for privacy reasons I've not explored further in this direction. If we were to consider allowing third party access for users credentials as a resource owner then there are many current drafts of OAuth to consider, not just following protocols in (s.2-8 v2-12), but also to prevent against replay attacks with a nonce or timestamps, and eavesdroppers with TLS [102]. The return data

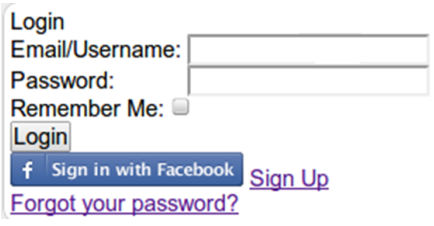


FIGURE 3.8: Example Login page with 1-Click button for simple access

from authentication of my account through FaceBook is shown in figure 3.9, the most important field to our sites developers is the *email* and is highlighted. Also useful is the existence and validation of educational information, this is to mimic

the email domain authentication. Another advantage is that with a degree of confidence we can be assured the user is the correct author of documents, and also is more likely to contribute quality content to the social network.

```
[id] => 639585717
[name] => Murray Wynnes
[first_name] => Murray
[last_name] => Wynnes
[link] => http://www.facebook.com/murray.wynnes
[username] => murray.wynnes
[education] => Array([0] => Array(
    [school] => Array
        (   [id] => 108242009204639
            [name] => University of Bristol )
    [year] => Array
        (   [id] => 201638419856163
            [name] => 2012 )
    [concentration] => Array
        (   [0] => Array
                (   [id] => 197323330286327
                    [name] => Computer Science )
        )
    [type] => College
) )
[gender] => male
[email] => mrwynnes@gmail.com
[timezone] => 1
[locale] => en_GB
[verified] => 1
[updated_time] => 2012-07-30T04:28:21+0000
```

FIGURE 3.9: A return OAuth v2.0 authentication array from a FaceBook App to be validated and tokenized for access

3.5 Scientific Papers

A primary use case is for users to be able to ask questions, receive answers and generally discuss academic material. Before creating the platform to facilitate this we need a method to upload content onto a storage device attached to the Internet. If we refer to Appendix A, we can see that current academic social networks allow the user to add meta data to the publications. The preliminary meta data obtainable without needing user input from a keyboard is: *date uploaded*, the *ID* of the user who uploaded, *file size/format* and anything further we can ascertain from file analysis tools. Primary and preliminary data required to upload material is the 'Title' and 'Authors' data. Secondary input information that is optional is: *date* of paper and *keywords*. It's important that a security protocol is a major consideration to begin with so that it doesn't contradict any design later on. For users to collaborate on papers we are offering the option to flag uploads as 'Private' or 'Public'. By default the 'private' option is opt out, meaning a user must actively choose for content not to be secure, it's important unpublished research is not accidentally released into the public domain.

Considering the public papers case first, before implementing the logical form into the program we need to design the database using entity relationship models. Additionally one can construct a database schema, or set of all relations each

forming a table. An initial database relation and it schema is shown in figure 3.10, where the group entity is a new object and *not* a security group.

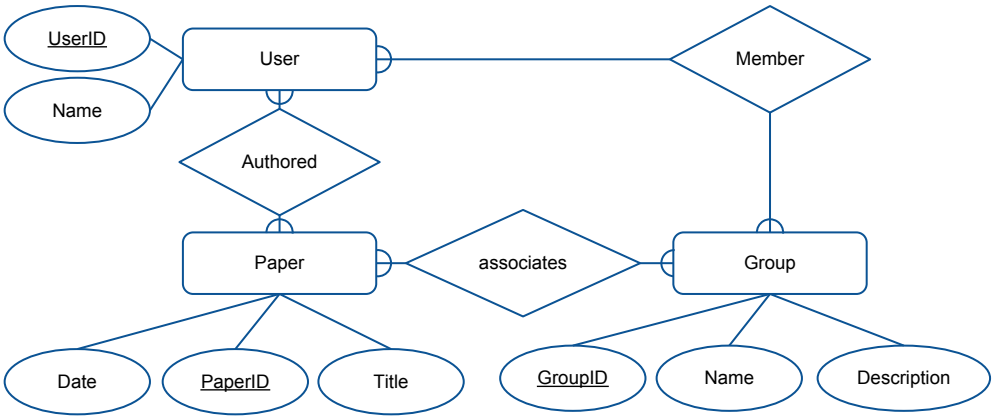


FIGURE 3.10: Entity relationship diagram with the three primary social networking entities: User, Paper, Group

The entity types, which each have at least one non-key attribute, are *User*, *Paper* and *Group*. They each have facts or attributes attached represented by ovals, for example a Paper can have a Date, ID or Title. Each attribute is associated to an entity and has a domain of values, for example 'Date' is the SQL data type TIMESTAMP ('YYYY-MM-DD HH:MM:SS') and its range is upper bounded by the year 2038 UNIX problem [103], where the system time is limited to 32 bits. The uploaded scenario is not incorporated into this conceptual view, as the user that uploads a paper might not be an author and author becomes a subclass of user. It's useful to add relationships between social networking groups as they also form an important part of the system structure. The associations between the entities are depicted with diamond shaped relationship names and the many to many (M:M) is shown using *crow's feet* notation. For example each Paper can be associated to many Groups, or a Group might link too many Papers. If we imagine a real example of when a relationship is enacted upon instances, the attributes can be described on relationships themselves, for example transitional data collected such as 'created by'. There is no strict rule on any of our relationships regarding participation; a new user does not have to have authored any papers. Many hidden attributes are not included that distract the motivation, namely 'Paper Uploader', 'File details', 'Is Private' and 'Keywords'. This E/R designs and similar derivation I've mentioned can all be implemented in SQL. To apply the M:M relationships in MySQL again we will need an intermediate table to prevent key redundancy issues. Each entity has a key attribute 'ID' such that only one instance of the relationship has that value. The key constraint is useful for identifying unique entities such as groups that might share a name. Each entity here becomes a SQL table and the attributes form columns in the table. The relationships can be modelled as foreign keys, for example:

```
ALTER TABLE `User_Papers`
ADD CONSTRAINT `User_Papers_ibfk_1` FOREIGN KEY (`user_id`)
REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `User_Papers_ibfk_2` FOREIGN KEY (`paper_id`)
REFERENCES `Papers` (`Paper_ID`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Shows the constraint for a bridging table between users and papers and shows two foreign keys linked to the primary key IDs of user and paper in the 'users' and 'papers' table. The 'ON DELETE CASCADE' command means each child row in the 'User Papers' table is set to be deleted when the corresponding parent row, linked by the foreign key, is also removed. By locating functions and logic into the database less access calls are made, this will speed up the overall application. The schema for the paper table relation is as follows:

```
`Paper_ID` int(11) NOT NULL AUTO_INCREMENT,
`Paper_title` varchar(300) NOT NULL,
`Paper_slug` varchar(60) NOT NULL,
`Paper_location` varchar(200) NOT NULL,
`Paper_date` date NOT NULL,
`Paper_uploaddate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
`Paper_uploader` int(11) NOT NULL COMMENT 'Reference userID',
`is_private` bit(1) NOT NULL DEFAULT b'1',
PRIMARY KEY (`Paper_ID`),
KEY `Paper_uploader` (`Paper_uploader`)
ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

When a user uploads a paper it should be completed as a single atomic action to preserve consistency of the database. Therefore we are using the InnoDB engine, which follows ACID properties of a transaction to lock the database when we are performing inserts or updates to several tables [104]. As we have a large database used by many people we need a form of concurrency control and InnoDB will handle scheduling in this regard.

3.5.1 Paper Uploading and Storage

To handle papers we have created a paper controller class and associated model class. The requirements for the controller are:

- By default should load all the public papers
- A paper view method for displaying an embedded paper if privacy allows, with all of its meta data; lastly a panel for asking and viewing paper questions.
- A method to create a new paper which generates a HTML form to allow a user to enter meta data
- A method to handle the file uploading process

The Paper model class contains many useful functions allowing the application to interact with the paper objects stored in the database. It's important that we follow a singleton approach when loading the database into the application;

this is to reduce resources being wasted. We achieve this by loading it within the 'config/autoload.php', lots of other PHP classes are parsed here once also.

The paper create method execution looks like follows 3.11, not seen is a 'JQuery UI' date picker to assist with entering a date.

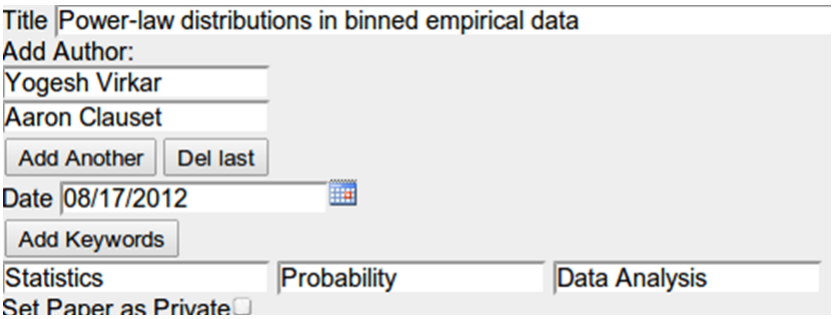


FIGURE 3.11: A view to create a paper with form fields: Title, Authors, Date, Keywords and Private tag.

A user that clicks submit and passes the form validation procedure, where author is required and all inputs are encoded, then proceeds to the upload method. The accepted paper format currently is the Portable Document Format, or the open standard (PDF). The attribute 'Paper location' points to where the paper is stored in the file system.

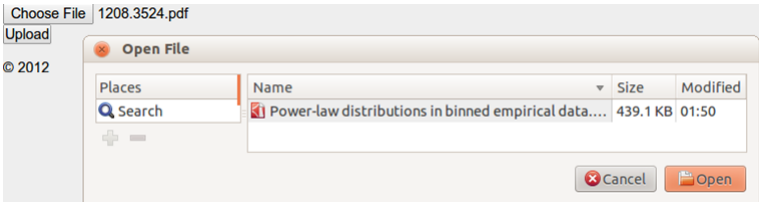


FIGURE 3.12: An example paper being uploaded from a user's file system, with valid size and format

The maximum allowed submission size is 2MB, as specified by default in the PHP configuration files, but this can be changed if users demand larger files. In figure 3.12 we can see the paper upload file having been selected, unsupported formats or file sizes will produce an error message here, forcing a user to reselect a valid file. The final location of where the paper is stored is determined by the privacy options selected, for example if the paper is public then it is stored in the cloud, but if it is local then we encrypt it onto our server. The reason for this is because it's assumed that a much larger proportion of published and public papers will exist and we should be able to handle this demand. It's understood that currently many research archives already exist and which provide extensive meta data. An attractive feature not yet implemented is to create a rich communication method to these sources, and there are more details of this in the 'Future work' chapter.

The code which handles the ultimate location of a paper object is the paper model, and the first step is to add the papers into the database as shown in listing B.8. The ability to upload public papers should attempt to always demonstrate acceptable performance. To deliver this requisite the location for cloud-based persistent storage we have chosen is Amazon S3 [43], which aims to provide storage that's 'scalable, reliable, secure and fast'. It's an Infrastructure as a Service (IaaS), that's virtualized and remotely accessible. The usage is elastic and on-demand, and can be monitored with alarms to email or phone as a method to control the resource usage. Uploaded papers are stored as '1 byte to 5TB' objects in a data directory or 'bucket', ours is located in the European region for increased latency. As data is going to be stored and shared around the world, inevitably its going to enter overseas jurisdictions. This is not a major concern as the Information Commissioner sympathises with at least complying with the *Data Protection Act 1998* (DPA). This is because the DPAs generality is widely applied, through non-EEA countries [105], and by companies in the USA that adhere to safe harbour principles. Fundamental issues with many current popular social networks are that the servers are not located within EU and are exempt from the DPA, thus being able to market user data. A PHP wrapper class is provided by Amazon to query the REST API with operations to update buckets or objects inside. For example when the file is uploaded the paper model stores the file into local storage, then includes the S3 class and puts the file as an object inside the bucket. The paper controller class' view method inally generates a HTML page which looks like 3.13.

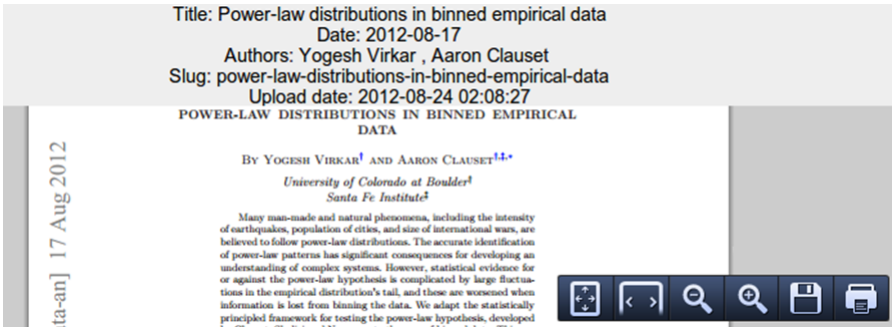


FIGURE 3.13: Paper view page with embedded PDF loaded from S3, and minimal meta data. See B.19

Each bucket is secure by default but access can be controlled by drafting a bucket policy. By specifying the bucket resource, the action to get an object can be allowed to any I.P address. However this is not ideal as bandwidth can be expensive, so there are several steps in place to throttle bandwidth such as browser caching seen in figure 3.14. As well as browsers storing paper objects, an 'active papers' table acts a server side cache, when a public paper is requested an expiring link is issued for the S3 object and pooled into this database table, if the file is requested again the paper is found via that link but if not then a new gateway is opened.

This limits bandwidth leakage and provisionally constrains file access to visitors using our application.

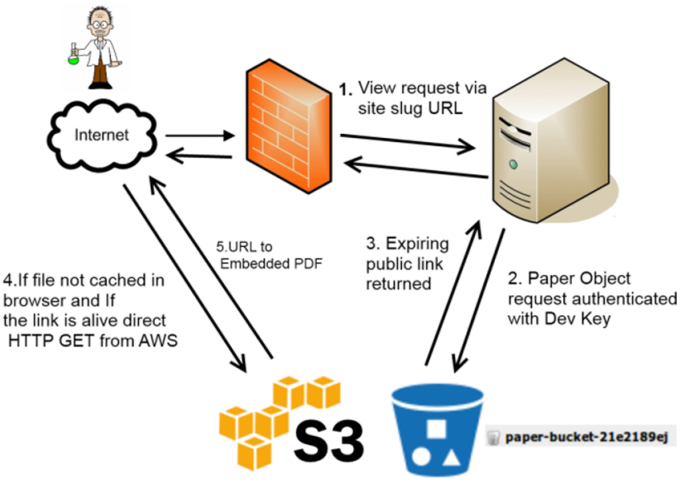


FIGURE 3.14: Steps to fetch a public paper for viewing, resources are stored in a browser cache and pooled by the server

A link to a paper object using 'Google Chrome's inspect element addon is shown in figure 3.15.

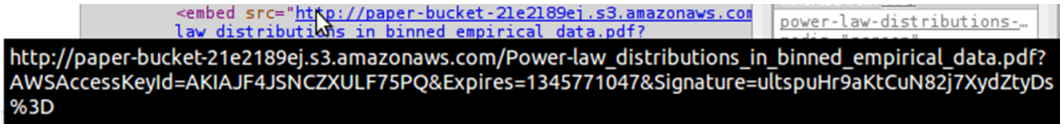


FIGURE 3.15: A URL to a public paper object stored in S3

When the same paper is uploaded twice we can configure S3 to take care of versioning for us so that only the latest content is delivered. In the paper model the code is engineered to provide an entrance to Amazons web services. The terminal process of paper file storage, or transaction methods into S3/local, are executed exactly as in listing B.12. Now users can view the papers, a main objective we need to tackle is a way to ask questions or answer them. The paper controller 'add question' and 'delete question' gives a way to do just this. Each question is stored in the database table 'Questions' with:

```
Question  ID (Integer ),
Text as a variable string of max length 5000 characters,
Author ID (Integer size 11 and Foreign key to UserID),
Date (timestamp)
Reputation karma (integer size 11).plugin
```

Each instance of Question also has a one to many relation with the answers table. An example submitted question is shown in 3.16. A 'what you see if what you get' text editor has been considered for use to bring user friendly text formatting,

but due to limited screen size we opted for a simple and minimalistic input form. Many major web browser support features like spell checking, for example using Firefox I've been notified the text 'stabilize' is incorrectly spelled. By looking at

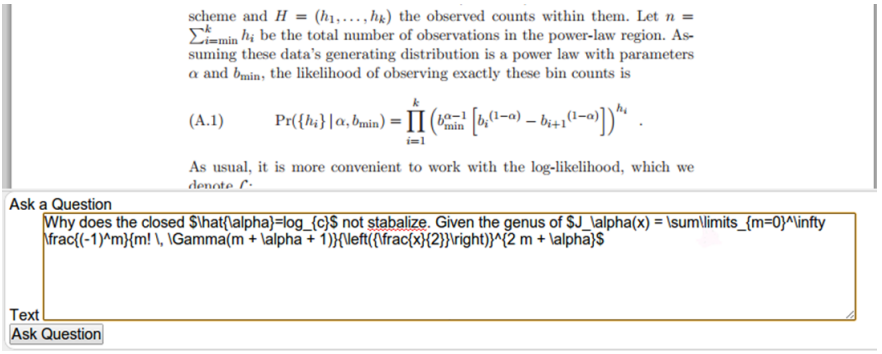


FIGURE 3.16: An unconverted LaTeX question to be asked, relating to the paper

figure 3.16, we can see that the input box contains some mathematics to aid the question, so we need to use HTML-Purify so that the characters are preserved but also to prevent XSS and SQL injection. The ability to display mathematical symbols and equations is provided by using the JavaScript display engine MathJax [106]. Once the question is processed by this code in the browser the result looks like figure 3.17.

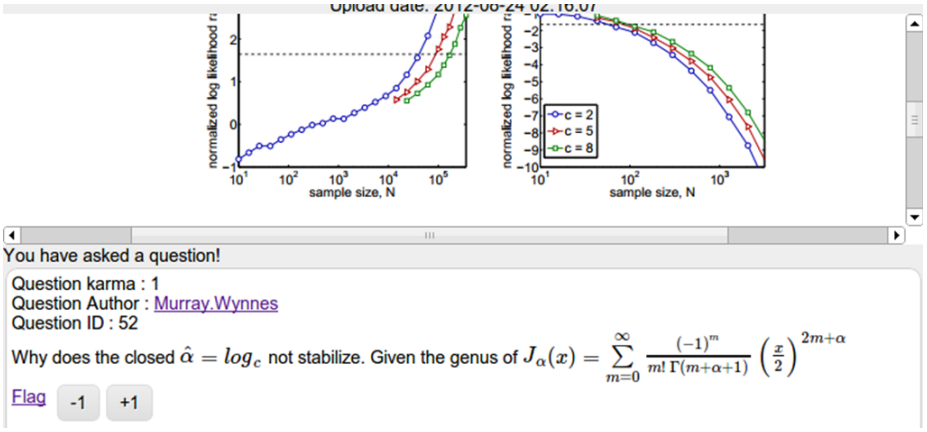


FIGURE 3.17: A submitted question that has been converted into LaTeX or MathML

3.5.2 Private Papers

Material under access control represents a key asset of the system and even partial exposure might impact our reputation or demand legal resolve. If a user doesn't change the default paper privacy options, by selecting the radio group 'is paper

private?', then a paper is assumed private and stored encrypted on the local server. The reason for this is that the content has to be guaranteed to be stored securely, but relying on Amazon's S3 and web services not to be affected by DoS attacks or malware adds an unnecessary extension to risk. The code which handles the ultimate location of a paper object is the paper model, and the first step is to add the papers into the database as shown in listing B.8. The file is therefore stored encrypted using 256 bit AES CBC (3.6), with a binary flag in the 'papers' table to say it is private. Figure 3.18 shows the encrypted files contents at the file location 'www/uploads/papers/$encrypted'. The secret symmetric key is generated by expanding a smaller master key, and neither is distributed through a secure channel as they are used by one party only. Encrypting files to be stored on the server doesn't guarantee they are being requested by the correct user, or that the user even has permission to view them. Also the transmission of the encrypted file over the internet and then through the browser is another security issue which we face. Again we are choosing TLS to at least encrypt transmission, although this may prove a hindrance for scalability as we are by default enabling site wide TLS 1.0. The reason for this is that establishing a TLS connection requires a handshaking protocol that consists of key derivation/exchange procedure, this has a computationally high cost (e.g RSAES-PKCS1-v1_5) for key signatures and validation. These steps need not be taken in full as TLS supports rebooting previously arranged secret keys by matching with client sessions, but as there are equal areas of the application that are public verses private, defaulting to TLS seems appropriate. The steps to enable this are found at the end of the chapter. An unpublished paper is a very important document, which needs to be protected. The integrity of this object is also highly sensitive and shouldn't be changed by anybody other than the authors of the paper. The encrypted paper plain texts stored on the server should remain confidential but also maintain integrity. So far I've only specified the encryption mode and, although combined primitives such as Galois/Counter mode are available, to make sure the same paper is returned upon decryption we can check against the keyed hash message authentication code (HMAC):

```
<?php echo hash_hmac("sha256",$message,$key); ?> //(CHAR 64)
```

3.5.3 Security Models

The security we expect to enforce can be described by a policy which captures its needs. A Bell-La Padula (BLP) model only allows information to flow upwards i.e write up, which is somewhat inline with publishing. So the private academic user cannot talk into the public domain. When one reconsiders, this is not very accurate because educational ideas float in all directions, corresponding to when lecturers teach students specific material, known as *decontextualisation* in the BLP

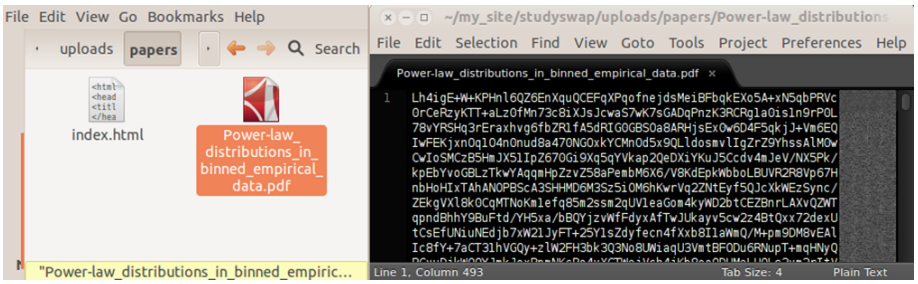


FIGURE 3.18: The location and content view of an encrypted paper p, where $C = MCrypt(p) = AES\_256\_CBC(p)$

model. An application layer security policy validates paper access requests and returns the decrypted file if the user has permission. One way to restrict access to the paper objects could be to use access control lists by attaching a *subject* and *operation* to each file in the format (User, Delete). The operations that authors then perform are well-known and examples are READ,WRITE,APPEND, etc. The subject could also stipulate group rights such as Users or Admin, similar to UNIX file system control. We require a more granular and scalable solution where the users and their permissions are set to change frequently. Therefore the chosen concept to adopt is that of a role-based access control ($RBAC$), where a role represents a job and separates users from privileges. Further a user is assigned a role and never permissions directly, this makes policy administration easier as each roles permissions can be changed without excessive software alterations. The two major roles we have defined are: paper *Contributor* and *Spectator*. A role is similar to a group in that they both handle a collection of users, but we require that groups also have permissions attached, thus we are defining roles [107]. The $RBAC$ is disrectionary as a contributor, assigned to uploader by default, has the permissions to read, update and delete a paper as well as being able to assign any role specific to the file to others. A spectator can only view a paper. Assignments of roles to users are unique for each paper in the same fashion as an access control list [108]. Originally the $RBAC$ introduced role hierarchies where the contributors transitively inherited permissions from spectator roles. However the contributors read user interface displays more meta data on the paper and therefore the inherited read ability is an enhancement of this operation. This can therefore be identified as a two tier hierarchical RBAC which exercises object based dynamic separation of duties [109]. For example a contributor can enter both the permissive state of a contributor and a conflicting spectator but the conflicting privileges are not exercised in parallel on the same paper object. This is known as a $RBAC_2$ system as defined by Sandhu et.al [107], because constraints of mutually exclusive roles are in place. Functions added to the paper model are: 'is_contributor'(B.18) and 'is_spectator' with the goal to determine the active role of a user, they both take arguments $paperID$ and $userID$ and return a Boolean value. This is so that the user can be granted elevated access to the file once they are authenticated

by checking the database role tables. An example view which is generate by the edit_paper_privacy method in the paper controller is shown in figure 3.19. The profile image is loaded from the user's controller, discussed in the next section.

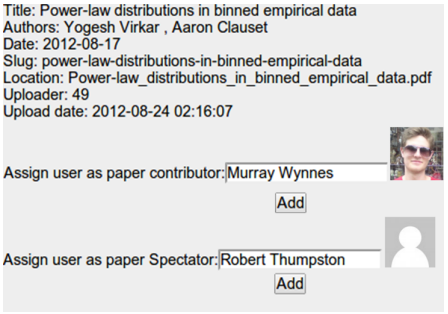


FIGURE 3.19: A form to manage and control which users have access to the papers ['views/edit_paper_privacy.php']

3.6 Social networking features

The main objectives state the need for users to be able to share messages and to express their appearance on the social network. By referring to appendix A we can see that all websites have the ability to create a profile page, join a group and contact others. Therefore these features are the concern of this section with any research relevant to this goal.

3.6.1 Profile Pages

There are many functions that a user subject can be involved with, not just academic material discussions. Therefore we have created a new controller called 'user' which will handle all the manipulation and presentation of user content. We've already visited how users are logged into the site and the prerequisite database tables needed to complete this access. As we are building an academic social network we now need the ability to store lots more content about users and help make the experience as useful and responsive as possible. The major function that the 'user' controller is responsible for is displaying a user profile page, and we shall tailor this to the academic environment by drawing from researching similar sites (Appendix A). The profile pages are stored in the 'Profiles' table with small text fields:

```
`about_me` varchar(200) NOT NULL,
`education` varchar(130) NOT NULL,
`interests` varchar(150) NOT NULL,
`skills` varchar(150) NOT NULL,
`contact_info` varchar(100) NOT NULL
```

We've decided this is a sensible initial set as a balance between privacy, page load speed and ability to express ones self. The *Profile* table is linked to the users 'user ID' via a foreign key. The relationship between the *User* and *Profile* tables is redundant and they could be merged together normalising the database further, but they will be rarely joined and the profile section is set to increase in size making selects on one large table less efficient.

3.6.2 Profile Pictures

Another attribute that we think is vital for an expressive Web 2.0 application is a profile picture. Therefore there is a form to upload a picture, using a CI class to manipulate the image into a thumbnail, stored in '/uploads/pictures'. It's important not to ignore if a service is already provided that solves a problem we face, and so have added optional support for the popular thumbnail content delivery service 'Gravatar' [110], which by default fetches a user's avatar using their email if it exists. The 'view' method in the user controller takes a *username* as an argument, this is uniquely created by concatenating the 'first name', 'last name' and a number if necessary to be unique. The result from this public function is to display the profile page using information from the database; however a different view is served if the owner of the profile page is viewing the page as they are given the option to change the data. My profile page when I'm logged in is shown in figure 3.20.

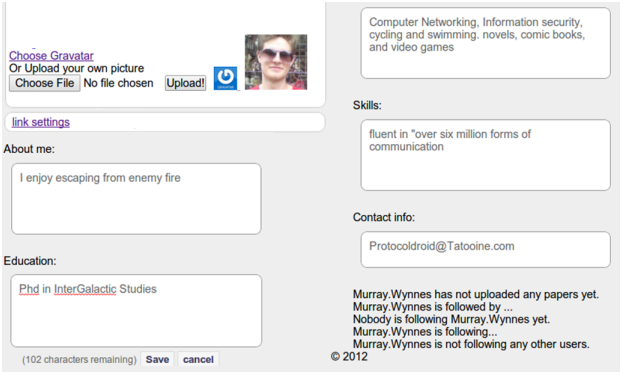


FIGURE 3.20: Profile Page sample view from owners perspective, object loaded from ['views/users/ownprofile.php']

The paper controller also adds methods to 'follow' other users when viewing their profile. Each text field in 3.20 is editable using AJAX so the page does not have to be refreshed when an entry is submitted. The Jeditable jQuery plugin written by Mika Tuupola [111], gives the user a dynamic in place editor, as shown below the education field, so that focus on a particular box enables a character remaining counter. The AJAX targets the set of functions in a 'model handler' to return a

very interactive and non-disruptive front end service. A typical example of this is in listing B.14: to autocomplete text boxes for the user without them needing the refresh their browsers, once they select from a suggestion dropdown list a URL for a thumbnail image is securely retrieved from a hash of their email. However it is inherently insecure as background HTTP requests via asynchronous JavaScript enables further CSRF vulnerabilities. It is vital that each form in the system verifies and meets minimal security standards, and AJAX is not exempt from session riding, with mitigating efforts through header referrer checking and the aforementioned CSRF token:

```
$.ajaxSetup({ data: {csrf_token: $.cookie('csrf_cookie')}});
```

3.6.3 Profile Page Privacy

With respect to a user's privacy we are taking the approach that no information is disclosed without clear consent as to what and how the information is used. This mentality is evident with the papers controller and should be convicted towards maintaining users rights within other areas. Therefore the profile pages, again by default, are completely private and any section can be set to be visible to either anybody visiting the page, only users logged in or nobody at all. The privacy options would possibly be more complex if we had adopted a 'friend' or similar relationship to model connections between user nodes. The privacy options are added into the profile table using 'priv_$Field' names as tiny integers where 0 is public, 1 is logged in and 2 is private. An example of changing privacy data for my profile page is shown in figure 3.21, which is computationally enabled through the user controller shown in listing B.13.

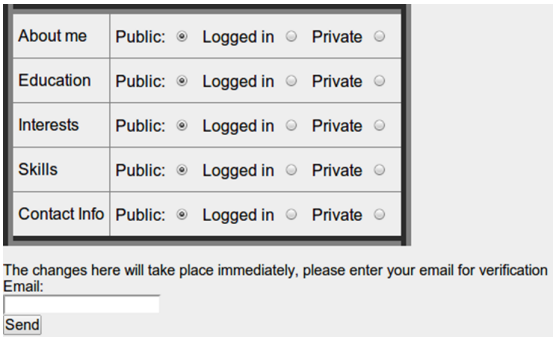


FIGURE 3.21: Profile Page privacy controls form with added email authentication

The email check at the bottom acts as further check so the user can re-evaluate the form, but as this is a security critical page the email is extra user specific authentication information to prevent a CSRF attack.

3.6.4 Groups

I've briefly discussed how the social networking groups are related to the users and papers entities, now I'll discuss the group controller with methods to create a new group or view groups and its associated resources, granted the user has permissions on the objects. A 'create' method in group controller first checks to see if the user is logged in by locating the session cookie data and validating it with a database session. If the user is logged in then a form is generated from the view files that allow the entry of a 'group title', 'description' and area to link users and papers together. All the text fields are again passed through validation checks as a standard precaution.

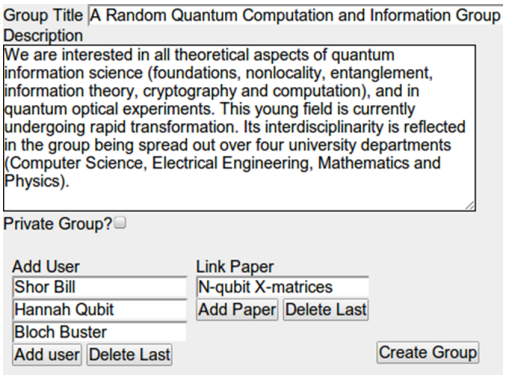


FIGURE 3.22: Group creation view with example data, loaded from ['controllers/groups/create()']

The group create view is seen in figure 3.22, with some preliminary data entered into the form. The database schema gets even more complex now with the groups table and attributes: a 'groups_users' and 'groups_papers' table need further composite foreign keys constraints onto other primary keys. Something not mentioned yet is how all of the content is viewable to users for later perusal. Every controller and model covered thus far opens up its objects by offering public viewing methods which query the database. The example for groups is the set of functions below, each returning various group:

```
public function get_group($group_id)
public function get_users_in_group($group_id,$is_group_public=false)
public function search_groups($search_data){
public function get_papers_in_group($group_id,$is_group_public = false)
```

The classical location for these to be used is when viewing a group on its own page, see 3.23. A journal could associate itself as a group but the scaling profile of the site is designed primarily for a green route to open access (OA) by self archiving.

The group data has been fetched from the group table to be displayed at the top of this HTML view. However less obvious is the need to join the groups table to the 'groups users' and 'groups papers' so that the users' model can be used to fetch

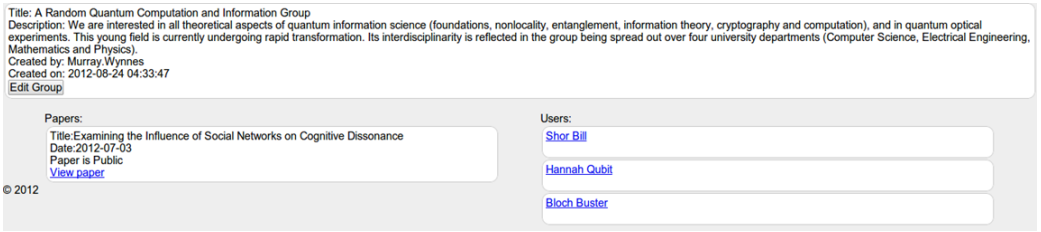


FIGURE 3.23: The created group being viewed, loaded from ['controllers/groups/view($group_id)']

each user by ID and similarly for the papers. If the group is being viewed by the group creator then an 'edit group' button is available underneath the description, this will give the creator a more advanced group creator form seen in 3.24.

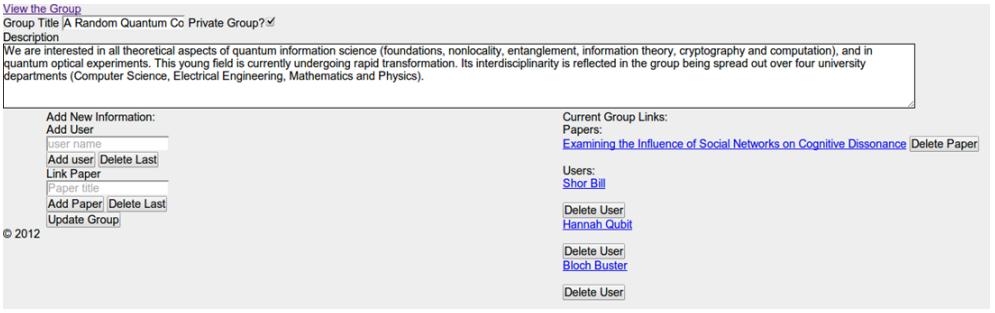


FIGURE 3.24: The group now being edited by the creator, ['controllers/groups/edit_group($group_id)']

When a group is requested to be viewed or edited by a user first the group controller class checks to see if it is set to private or not, if not then it is shown, but if it is a private group then the user must also be a member. This check is shown in the group view method of controllers/group, see listing B.4.

3.7 Messenger

For users to be able to contact each other through our site, a primary objective, a lightweight PHP threaded message library called Mahana [112] is included. A thread consists of a chain of messages ordered by date, and has a list of thread participants. Each message is formed of a text Body, MessageID, Priority and Date. When a user sends a message to another user a new thread is created with both of them added as participants. Then a new message is added with the sender ID notifying the sender address. Finally each message has a 'status flag' for whether or not it had been read by each user whom is a participant in the thread. This can be represented by the following ER diagram 3.25.

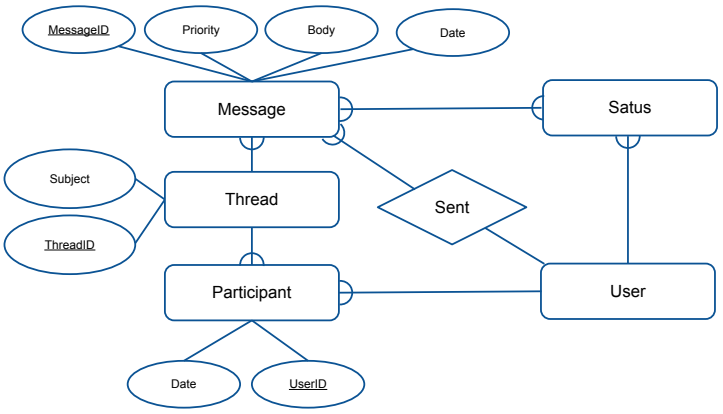


FIGURE 3.25: An entity relationship diagram for the message threading library Mahana [112]

There is no concept of a reply message and the messages are not related to each other, only to a thread. Therefore a thread and its messages can only be ordered by time sent with no added structure, such as whether a message was a reply to a particular other message. A view generated by the first method, loaded when the messenger controller is called, is pictured in 3.26.

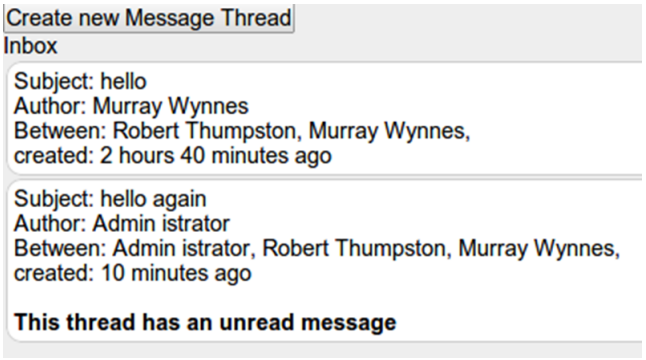


FIGURE 3.26: The view of all a users threads in which they participate, joining of tables Thread, Message, Participant and Status required

Each thread is loaded into a HTML divider and wrapped in a hyper link, directed to the thread view 3.27. By clicking on the thread the method 'view thread' is called which attempts to display the view. Firstly the view request is authenticated by checking the list of participants for the thread against the currently logged in user, if not then they are displayed with status header 404 and a 'page not found' error. Each message is found by joining several tables together and then presented by looping through each record in the resulting SQL query array. To demonstrate the dynamic generation of HTML and JavaScript using PHP see the prior server side code in listing B.6, the PHP instructions with embedded non PHP code will be output in a GET response to the browser, analogous to listing B.7. The output code has no *body*, *html* or *head* tags, ommited them for viewing purposes, but

strictly they form the final composite view and are generated when the template is loaded.

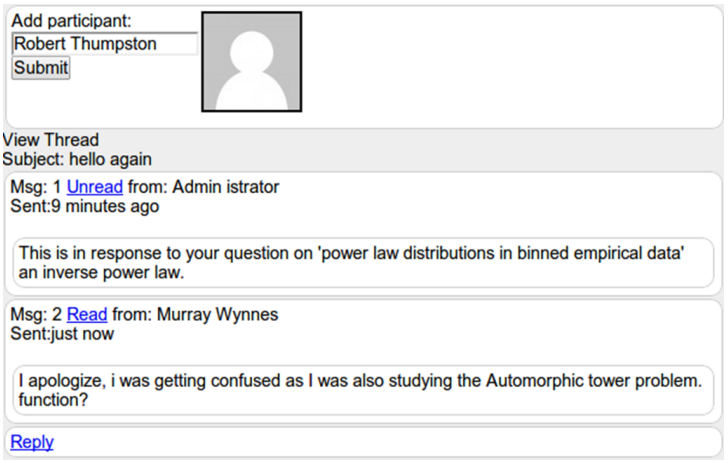


FIGURE 3.27: A view of all a threads messages [views/messenger/view_thread.php]

3.8 Towards a Search Function

A requirement of the system is to be able to search through data records and so users, logged in or not signed up, need a way to do this effectively. It's important to specify that I am not describing a better organization of application resources here, for example to allow autonomous web crawlers, or *spiders*, or other applications to search through social network information. The main data structure that our system uses is a relational database, and it will play an essential role to organise data to be searched. The primary records to be accessed by the search function are groups, papers and users. Therefore we will need to create algorithms which take user input from HTTP requests, sanitize the data and return a formatted SQL response. The search model handles the object procedures and functions and returns SQL queries. An imperative consideration is that private data is not exposed upon searching to users who are not authenticated to see it. The pseudo

code of a search function to demonstrate the data flow of a search request is:

Data: HTTP POST
Result: Search results
submit search form;
initialize validation method;
$searchInput \leftarrow POST(search)$;
if *searchInput* **then**
  filter *searchInput*;
  if *POST(radio)* **then**
    foreach *Table in the set* $\{Group, Paper, User\}$ **do**
      if *POST(radio) = 'Table' selected* **then**
        query ← SELECT all data WHERE validate($TableID, userID$)
        forall the *attributes x of Table* **do**
          WHERE *x* like *searchInput* OR
        end
        FROM *Table* ;
      end
    end
  else
    query ← globalsearch(*searchInput*);
  end
  load searchView(*query*);
end

Algorithm 1: Display search results

The above algorithm for searching is divided amongst a search controller, model and view; this algorithm can easily be replaced to filter options or search diferent objects, a benefit of the MVC 'strategy' design pattern [62]. The validation step is performed so that no private data is revealed unless the user has access. An extract from the search controller, as seen in B.5, reveals the general logic and how this algorithm is converted into PHP.

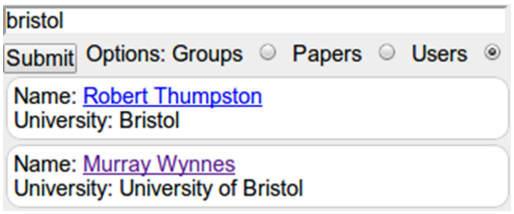


FIGURE 3.28: An example search result with input data 'bristol' and 'User' radio selected [views/search/index.php]

Before constructing a search of the website structure and its data content, it's valuable to understand how incoming URLs are routed depending on their format.

The file hierarchy of the application can be seen in figure 3.29. HTTP requests

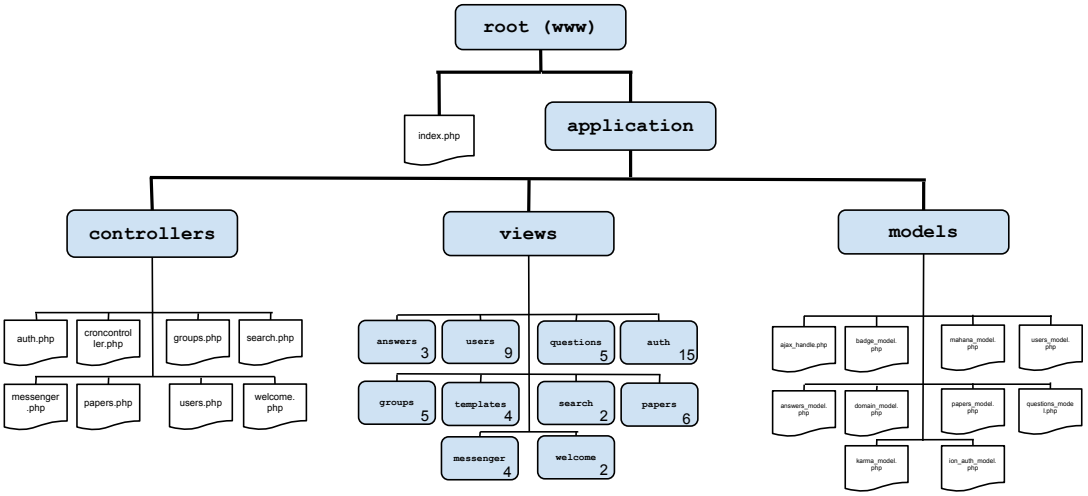


FIGURE 3.29: The file hierarchy of the application, where the view folders contain the number of files shown

are routed by the framework according to the following pattern:

`http://localhost/index.php/controller/method/a1/a2`

An example method 'searchview' in controller 'search' is:

`http://localhost/index.php/search/searchview`

If the arguments $a_1, a_2 \ldots$ exist, they are passed into the method name for the controller. Also we can redirect the control into any other location dynamically, so there is no permanent format for the URL as they can be unpractical as shown above where the word 'search' is undesirably repeated.

For details on how our servers are installed, along with an administrator guide, see Appendix D.1-2. There are also final presentation views, with the 3D style being produced by Firefox's Addon 'tilt'.

Chapter 4

Analysis

4.1 Testing Strategy

An extensive testing strategy is needed to consider all aspects of the systems integrity and security. An efficient and rigorous method to white box test is unit testing. Between project versions the functions and modules are tested by using PHPunit [113]. This allows for systematic fine grained area testing that is comprehensive and leaves out no part of the design. Also by unit testing it means we can refactor the code if it is performing too slowly and check that the code still performs desirably with the many inputs. The individual components can be tested first, such as SQL relational algebra combinations. Then the modules and subroutines that are made up of individual components or functions can be tested. Eventually the whole system behaviour can be checked from the bottom up; this will ensure consistent data types are being used to pass variables around the system.

The fast paced development process has tought us that it's essential to define code so that it doesn't repeat anywhere, especially if it's central to driving application events. In terms of testing the types of test data to be used are: normal data (to be expected), extreme data (far from the boundaries) and incorrect type erroneous data (to invoke the error handler). In hindsight we've learnt development can and should be driven by testing, where tests for units are written before implementation, this not only increases the tests quality but also understanding. A function 'elapsed_date' is to take string/date arguments '*Date sent*' and '*Todays date*', and is used in many locations on the website to calculate how old items are. A test written using PHPUnit to check the output of this can be expressed as such:

```
class DateTest extends PHPUnit_Framework_TestCase
{
  public function testElapsedDateStringformat()
  {
    //Create test dates MM/DD/YY
    $test_date_sent = "01/01/2011";
    $test_date_now = new DateTime('05/07/2011');

    // Assert date output is correct
    $this->assertEquals("4 months 5 days ago",
      elapsed_date($test_date_sent,$test_date_now));
  }
}
```

One may guess the action of the function 'elapsed_date', and whether or not this test passess. In fact the assertion fails as the return string to be sent to a HTML view should is '4 months 5 days ago', and the test is wrong. An improvement on the testing process would be to express function behaviour before writing code, this is known as *Behavior-driven development*, and in the future we would like to adopt a pattern where every member of the team shares a language to specify the required behaviour of units at any level.

There are of course not only tests that need to be run on the server but also front end compatibility checks to be completed. Pages must not contain messy non-compliant tags becoming not well-formed, and CSS validity is required so the site can be viewed across many different browsers. Also we have setup servers which are to be used for feature testing and in the extreme case use server analytics to fuel *bucket testing*, and determine how layouts affect users activity.

4.1.1 Security Testing

There are cryptographic calculations which can be tested against known plain-texts and cipher text pairs to verify the algorithms are implemented correctly. In contrast to white box testing an automated process of finding bugs and exploits which are guaranteed to exist is called *fuzzing*. A semi-automatic fuzz test I've run from within the server uses 'JBroFuzz' [75]. A simple test exhibits status OK return headers over a range of invalid HTTP versions C.5. Another test reveals that a logged in user can identify which ID numbers correspond to private papers in the database C.6.

This is because public papers have no option for privacy and so do not to check whether the user is a contributor, an event costing time for private papers:

$$edit\_paper\_privacy(n) = \begin{cases} h = 200, load(editForm) & \text{if } user \in contributors(n) \\ h = 404 & \text{otherwise} \end{cases}$$

Where h = header status code. The offending database SQL query, and its implementation (B.18), that costs more time is: Select the current user and paper ($\sigma_{userID=\$userID,paperID=\$paperID}$) from paper contributors using natural join:

$$(Paper\_Contrib \bowtie_{contrib\_users.contrib\_groupID=paper.contrib\_groups.contrib\_groupID} Contributor$$

If the firewall doesn't detect and block the fuzzer (perhaps distributed by onion routing) the relational algebra above will expose paper properties. To prevent this side channel analysis we can hide the signal in the time dimension by randomizing the execution time with dummy code or by shuffling operations. Specifically every invalid request to edit paper privacy should return 'Page not found' within a time and byte size which is independent of the underlying branching decisions. This type of attack can be automated with a statistical decision algorithm using the following features:.

```
Hamming distance(integer), Time(real), Response header(categorical)
```

Instances from the fuzz test can be trained to classify an instance based on the feature vector $\vec{x}$. Assuming the features are independent, the learning process will determine a weighted vector $\vec{w}$ for each category k, so that unseen instances can be classified by mapping feature vectors to the real numbers using probabilistic function:

$$y(\vec{x}, \vec{w}) = \sum_i \vec{x_i}\vec{w_i}$$

With this information a malicious user could overflow the database with their own private papers and hope for an out of sync error thus confusing the ID's and being able to view private papers. A solution to this, which we have already implemented, is to insert paper records with locking transactions.

4.1.2 Analysis of tools

The choice of programming language for the server side code has proven to be suitable for our development needs. For security reasons the language can be configured to restrict file functions and directory access. To thwart DoS attacks the application memory limit, file upload size and maximum POST size are reduced,

this is also so the system can fall back on reserved resources. A continuous challenge we face is to follow a team development workflow that we are comfortable with. Fortunately we have access to proficient communication, design and project management tools. For example if we are both working simultaneously on the same source code then git can merge the results together and resolve any conflicts automatically. However if we are working on different files of the project then the observer pattern within the MVC architecture pattern allows all decoupled objects to dynamically adjust and still function, without the code we are working on needing to know of this detail. A more abstract view of the workflow is that we are working on different features at the same time, conflicts here are handled by exploiting features of the distributed open source revision control software git, by:

- Pushing over SSH connection into the hosted optimized repository
- Collaborating with a full and hashed commit history
- Using branches to experiment new code

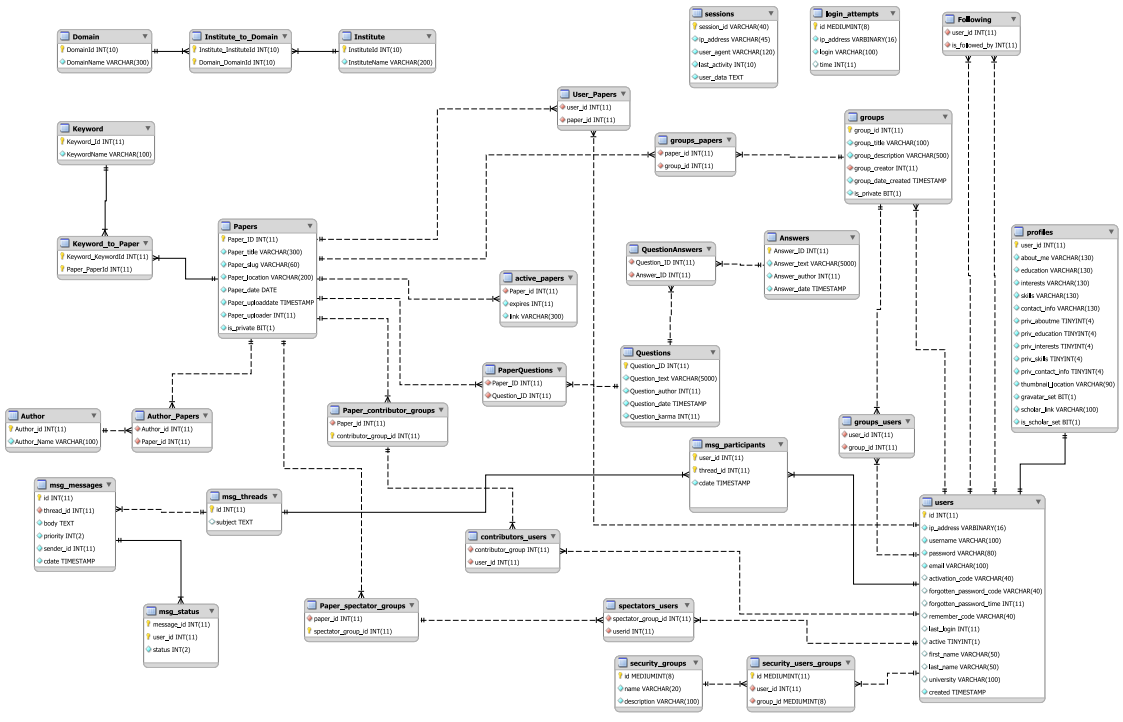


FIGURE 4.1: System Database Entity Relationship Model created using MySQL work bench [http://www.mysql.com]

Figure 4.1 is the entire database schema that the server side code uses; it's access only by the model classes. The primary keys of each relation table are the top

attributes and any foreign key is indicated by a red diamond. In terms of entity relationship cardinalities, this shows all the relationships extended with joining tables for final use. I have mentioned already the groups, users, papers and messenger tables which can be seen as having the most relationships with other tables. How can we tell if this is the best possible design to allow for the huge variety of complex SQL data queries?

We need to organise the tables such that data is consistent and not duplicated anyway. Also the structure should be flexible to allow for arbitrary size entry requirements; this process of data redundancy is called normalisation. The first normal form is to remove all sets of values and make each data values atomic; this is achieved in every table as no sets exist. However problems in this form happen when updating data: when the user's browser agent in sessions if an agent was to be updated then each and every session would need to change every row. Most attributes are functionally dependent on their table keys but for practical reasons this is sometimes not true, a noteworthy example is that a user record is not determined by their I.P Address. The database is also in second normal form as no attribute is partially dependent on the candidate key. If the username was not just unique but formed a candidate key then all user attributes would be partially dependent on the candidate key and the table would not be in second normal form. Finally to be in third normal form no non-key attributes are transitively dependent on the candidate key, and this is the case with many binary field attributes which are non-trivially functional dependent on key data. An example is the scholar link' attribute in the profile table which determines whether or not the Google scholar link and the attribute is_scholar_set'. Therefore as the database stands and evolves it is not optimal for both increased speed and reduced redundancy but is a trade-off between them both. In fact some tables are purposefully denormalised, such as users with I.P addresses, to reduce the number of tables and subsequent expensive join operations.

4.1.3 System Load Testing

Also beneficial to consider is the limits of the database schema, and which key dependencies may cause problems. One way to do this is to test the system load to check if we can actually store the required number of data records and still function efficiently and correct. Auxiliary analysis of the chosen data structures should also be completed. In particular was a relational database the best choice for our needs and whether the table structure be improved. Possibly the best way to test volume loads are by using real data records. This will become possible when the system goes into a beta testing phase. The time taken to load the base classes for the paper controller, which is the longest is 0.034 on an EC2 instance.

The application is 8,407,976 bytes in size, and we can use Code Igniters profiler to identify bottlenecks, such as session data updating, in the SQL execution times:

```
DATABASE:  test   QUERIES: 5  (Hide)
0.0008    SELECT *
FROM (`sessions`)
WHERE `session_id` =  '4105a424df27fab956f911e7ee7fa6f3'
AND `ip_address` =  '::1'
AND `user_agent` =  'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1'
0.0030    UPDATE `sessions` SET `last_activity` = 1347926688,
`session_id` = 'eaef0ab670eb6535044fafca90edd1e3'
WHERE session_id = '4105a424df27fab956f911e7ee7fa6f3'
0.0007    DELETE FROM `sessions`
WHERE `last_activity` < 1347919488
0.0008   SELECT `users`.*, `users`.`id` as id, `users`.`id` as user_id
FROM (`users`) WHERE `users`.`id` =  '1'
LIMIT 1 0.0007 SELECT *
FROM (`Papers`) WHERE `is_private` =  0
```

I have tested the index page for the users, which lists all the current users. This is a good example of SQL data being extracted upon GET request and all of it being printed into a single HTML page without pagination or background loading. The code to create test users is seen in B.17, and i will be testing incrementally loads of 100 users. It takes approximately 29.79 seconds to register, create profiles for and send a hundred users sign up emails, see Appendix C listing C.1. This time is factoring in having to display the users and their details onto the admin control panel view; also the added time taken to run 'mt_rand' for each users name (B.17). To test how the server performs under simulated Internet traffic we could manually wrap up a set of curl instructions concurrently with Posix threads, but I have chosen to use an out-the box multi-threaded http load testing utility called 'siege'. I've run 10 concurrent users repeatedly n times ($10n$ transactions) with GET requests for a list of users of sizes $\{100, 200, \ldots, 1000\}$:

```
$ siege -c 10 -r 10 http://ec2-46-137-33-45.eu-west-1.compute.amazonaws.com
```

The results can be seen in table C.1, and figure C.2. What this shows is that the apache balances the load across our application fairly well, even when a thousand users and their details are fetched from the database. Each request also has to deliver the style sheets and other scripts, as well as load the database, but all transactions are effective within a reasonable amount of time. It is not a common action for a user to be able to request and view a thousand users on one page, therefore this is almost a brute force stress test, and for a micro instance of EC2 a throughput of 0.14MB/s over 100 transactions seems acceptable. Further, a more sensible request is to limit the query to say 30 users, then these can be shown to be loaded from cache in time = 0.0011 seconds (figure C.3). By choosing not to load all one thousand users the system can be seen to overcome computation bottlenecks, see C.4, where limiting the output to 30 users reduces initial send time from 1.1ms to 330ms.

Chapter 5

Evaluation

To assess the extent to which the initial objects were achieved I will now refer back to the objectives section in Chapter 1. In general we have seen much success, but such is the nature of the project there are many features and possible paths which this undertaking and those comparable can explore. Therefore this chapter involves a somewhat limited assessment of only the objects and tasks mapped out, whilst an investigation into further direction is left for the Chapter 'Further Work'. The following objective umbers, and our responses are as follows:

1. The first objective has been achieved by using the IonAuth library, we have obtained a secure method for users to be able to sign up and login. A modern authentication alternative is provided, by logging into existing third party applications using OAUTH v2.0. However it remains unseen if an attack would be successful as the web server has not been subjected to real attacks, for example a distributed denial of service. Basic user authentication of their educational institute is in place, and there is a way to assign users as moderators, currently with limited elevated permissions, if any issues do arise. Users can reset their password their password using email but they cannot currently update their initial user credentials, and currently need to make a new account if they wish to change university or their name.

2. The method of communication on the website can achieved primarily through the threaded messenger controller, which fulfils objective 2; although many secondary methods of interaction exist, such as the ability to follow a user. This works by posting all the users that you are follows recent actions onto the main start up page as a list of activity or site history. Users also have a default user profile page that allows them to post the required details and preliminary content is also retrieved from various APIs.

3. Users can upload only internet media type PDF formats and although this is a popular format, this is a definite weakness in the system. Also the

added Meta data comprises only the title, date, authors and keywords. I personally feel this is sufficient for many documents, but this is not extensive to overcome competitive academic social networks. A particular downfall in this area is that lots of the data can be extracted from the documents or gathered from existing archives and this should be completed. The system is scalable; by storing public material on a cloud storage platform the storage capacity is potentially unlimited.

4. Scientific papers which are entered into the database are searchable using each field in combination only, so the content is archived but to a very poor level. Not having more options filter search criteria and not being able to query the data using a REStful API demonstrated a weakness in the scope of the requirements. The annotation of content will allow for the network to generate an *ontology*. Further concepts such as *semantic diversity* should be considered, this is how related objects, media or events can be correctly labelled and understood for effective propogation [10].

   A user uploading content has the option to release it for public discussion, or if they wish they can assign the roles of contributor or spectator to control its access. However the roles and permissions are static to simplify system design, in particular the permissions are not loosely coupled from the application code so that they can be dynamically altered. There are no limits on the cardinality of the role sets other than the physical capabilities on the centralized server model. Unfortunately the permissions do not cover all of the realistic scenarios, for example there is no capability for any role to revoke another. Therefore temporary delegation of a role is also not permitted. Further restrictions to our policy are the distribution capabilities, in particular when a user requests to activate a role all roles to which they are assigned are returned. To increase the performance and scalability of the system to handle much concurrent activation, only the least amount of privileges required to perform the action should be granted per session in a need-to-know' fashion [114]

5. We've engineered a platform for papers to be discussed, and we have attempted to maximize the effectiveness of the visual requirements from the paper view and questions. However we face an on-going challenge to respond to responsive web design obligations which should be overcome, this is so the interface will cater for ubiquitous mobile device platforms. The text editors are very basic and although the objective has been met in this regard more support for mathematics formatting and education on how to use them needs to be in place.

6. Functions needed to complete an early reputation system have been demonstrated. In particular being able to up-vote or down-vote questions. This

is only used to order the display of the questions and a more effective approach could be installed which takes into account concepts such as ratings bias, freshness and decay [115]. For more details of these a reader should refer to the undertakings of related Thesis by Robert Thumpston, in particular see 'karma_model/badge_model' *not in listings*.

7. Most of the content on the website has the option to be confidential, indeed by default groups and papers are. Although we would like to encourage participation and to show some footprint of users participating in the social network so usernames and profile pictures are viewable by anyone on the internet

8. Users can create unlimited numbers of groups with 100 character titles and 500 char descriptions. Each group can have private and public papers attached at creation or at a later data, probably when the papers are published or new users are added. There is no support to give ownership or a ranking on membership other than the owner.

9. By using email addresses in place for username the system retains up to date contact emails of every single signed up member that is active. Although there is no method for the users to accord or opt-in to any future notifications sent.

10. The institute and domain authentication system is installed to prevent non institute member access, still this does not prevent a user pretending to be known under a different name within their institute or wherever they have a valid email address.

11. So far we have tested the HTML to see if the markup tags are valid and well-formed. Content negotiation will be in place on the server. I believe the layout is not finalized so it is difficult to argue cross browser compatibility.

Objective points 12-14 are best verified upon seeking the relevant chapters.

Chapter 6

Further Work

There are many features which we would have liked to implement, but given the time and resource constraints left to us after planning we have been unable to do so. This chapter summarizes content planned after the initial objectives, or which has been evicted because it has not been covered by the development process yet.

6.1 Scientific Papers

An attractive direction the project could take is to introduce feature to support additional uploaded file formats, such as LaTeXor Open Document format. There is also more content meta data which could be collected such as paper abstracts or digital object identifiers.

6.2 Paper Searching

We also feel strongly about enhancing the search capability of the website. This can be improved in many regards, from customizing filters to indexing all the HTML pages. In terms of the scientific papers themselves, unscientific literature can be represented by text documents and so the location of content should be considered. A handy way to discover new research could be for the database to extract all the text and allow a user to perform a full text search through every word and not just metadata. This would probably be an expensive process as searching selectively ignores all insignificant words. On the other hand it might be worthwhile as an intuitive method to discoverer the long tail through research content identification and not just keywords. Added convenience would be the ability to upload extra documents relating to each each paper, for instance the source code with a link to an online repository or workflow.

6.3 External research library connections

Some of the existing academic social networks attempt to link the users name with obtainable papers through citation searching on various databases. This demonstrates an opportunity for us to focus on creating a more reactive service to reduce the need for manual data entry. Two popular research archives are arXiv [116] with over 780,000 articles, and JournalTOC [117] with 20,000 free scholarly journals. Both have an API to support XML or JSON querying for paper links. There have been many campaigns such as the Open Archives Initiative to promote standards for interoperable content propagation, for example the 'Protocol for metadata Harvesting" and "Object Reuse and Exchange" [118]. More recently efforts from the non-profit research organisation Online Computer Library Center' have accumulated connections with over 10,000 libraries of items [119]. For development purposes, under their record use policy, a basic API with 10,000 queries a day is offered, with more requiring a mutually beneficial relationship where our library would supply the 'WorldCat' network.

6.4 Resource Exposure

In fact there are many ways in which our web server could expose its public information through perhaps a RESTful API or maybe RSS feeds. The MYSQL database could even be converted into semantic resource description framework which is then exposed to the query language SPARQL [120]. Once the mapping is complete it remains to create a vocabulary, and the attraction is this can be used for intelligent referencing, for example author = creator. There are even further technologies endorsed by WC3, and It's in our interests to design not only a web site for end users, but also a web service ready for the future of the web. A challenge we have faced is how do we sensibly design a rich service that can restrict access by behaving similar to a mentioned security model. It has been noted that most social networks designed in the web 2.0 era operate using a resource label policy enforced by the owner. A more elegant and powerful system is now possible using RDF, which will allow us to infer actual meaning from rules defined about subjects and objects, for example a 'frequent co-author' is also a 'co-author'.

6.5 Additional Security

For legal reasons, as mentioned in the administrator policy section, its important to keep server logs which could also be useful to detect misuse of the application. For example a background process or daemon could scan the log files and bans

any IP which shows malicious signs, by dynamically adding rules to the firewall, once such tool to perform this is Fail2ban [121]. Ideally a statistical anomaly behaviour based intrusion detection system will flag misuse detection if the transformations detract from normality. In terms of user password storage the results of a SHA2 function could be computed alongside BCrypt on signup and stored in another password column, then if an algorithm becomes fully broken with a feasible amount of time we can just switch the validation check. Also the usage of TLS is worth considering as the PKI computations pose a serious threat to a scaling model. An alternative protocol is *IPsec*, running at the network layer, the 'Encapsulating Security Payload' (ESP) provides both confidentiality and authentication. An end-to-end secure connection is established from a dynamic *Security Association* between client and server via the *Internet Key Exchange* key management protocol. Unfortunately correct IPSec deployement is complex with firewall conflicts commonplace, which could reduce security.

A main goal for project progression is to improve on current offered services. One way to do this is to upgrade any of the software or hardware requirements when necessary. For example the latest versions of encryption will be supported by performing an upgrade of the PHP and related language versions.

As the application proliferates it will be expected to not just delivered from one static central server to a fixed ideal of consumers. On the contrary it will be accessed from academics around the globe and each layer of the service should adapt to overcome global restrictions and thrive with internationalisation. For example at the front end every page should be shown regional language and each resource can be served using a geographically close content delivery network to reduce latency and strip attached cookies. Also the application will need to evolve; one way is to extend its database of authenticated Institute domain names.

6.6 Email Encryption

When users sign up to the website they are sent a randomly generated token link via email that expires, and when a user visits with that token in a HTTP request we check their session ID to see if it was the same person. In theory this should be enough security, but if a person knows a users email they could reset their password erroneously, this is not a security problem but a more serious issue could be an excessive abuse of this feature within a short time period. A general solution to DOS attacks is to slowdown availability, in our case either with another Capture or my choice one password reset per minute. Potentially many emails could be sent on future releases that include updates to newsletters, content or general notifications. Potentially a selection of these objects contains sensitive data and it would be incredibly beneficial to offer services to encrypt the

private email communication. Inherent to internet traffic, there are many scenarios where a mail server along the route of email transmission could eavesdrop and violate the messages privacy. For example another mail server could generate a message to masquerade the source address identity. To encrypt emails an open Internet standard is OpenPGP [122]. Its based on Phil Zimmermanns PGP and uses public key cryptography on a decentralized web of trust model. The email address and user public key certificates are signed by parties who agree on the binding. An implementation available to our application, under the GPL, is GNU Privacy Guard [123]. The process would work by allowing the user to enter their public key or a link to a public key server where it's held, then email messages sent to them encrypted with this key can only be decrypted by the holder of the corresponding private key, namely the same user.

6.7 Third Party Connection

A system requirement was to exploit current bibliographic data calculated from academics papers and citations. An excellent source for up to data information of this nature is Google Scholar, however they do not currently have an API that allows you to effectively retrieve the data. Therefore I have implemented a section of the page so that when a new user signs up they have the option to add their own bibliographic data if they so wish.

6.8 Extras

As mathematical symbols are mostly independent of spoken language a suitable extension is to provide a MathML plugin if future users wish to be emailed site content. Final considerations, which themselves represent significant projects, are that web browsers appear on many heterogeneous platforms and these allow for operating system level applications to be interacting and depending upon them. The Android operating system offers an open source development stack and increased speed, computational power and availability to applications with embedded browsers. This could be one of many directions that the project take in order to reach potential academics who may be part of the 1 million device activations per day `http://www.android.com/`.

Appendix A

Existing Academic Social Networks

Login/signup Methods Key	
Key	Value
fb	Facebook
gg	Google
reset	Reset password via email
u - [a]	University supplied [authenticated]
Material linking Key	
Key	Value
m	Manual upload
i - [loc]	Option to import from [location]
Social Network Key	
Key	Value
fo	Followers
fr	Friends
g	Groups
m	Messaging
h	History/activity
Personal Expression Key	
Key	Value
pi	Profile images
c - [w/e/l]	Contact [website/email/location]

FIGURE A.1: Key to existing Academic Social Networks table - Research carried out in July 2012

Name and URL	Site slogan	Site Signup/Login methods	Paper viewing or discussions?	Material/Papers linking to user?	Paper metadata (other than date,authors,title)	Restricted access to anything?	Integrated maths in discussion	Social Networking features?	Reputation system?	Personal expression?	Number of users	API?	Extra information
Academia http://academia.edu/	share research	u/reset/find friends/papers. Import friends from fb/gg	pdf entire document on page by service 'sribd'/ no discussions	l - pub med, arXiv. m. microsoft academic paper search	tags appear in the News Feeds	-	-	m, fo, department colleagues	-	c -w, status, add books-papers-talks-teaching-blogs-CV,pu	1.78+ million	-	uni domains i.e bristol.academia.edu
Research Gate http://www.researchgate.net/	Connect, collaborate and discover scientific publications	u - a/reset/add interests/fb login	pdf advanced viewer. Abstract with request full text/public papers are just viewable no	l - pubmed, m with bibtex	keywords, type (journal), abstract, DOI.	papers if not found full text link	-	fo, m, h, projects	RG score	pi , profile completion, info, degree/pu/activity	1.9+ million	in progress	can ask questions with attachements
My Experiment http://www.myexperiment.org/	find, use and share scientific workflows	openID, reset	many different file formats, with the ability to comment	workflows, files, packs	tags, credits, license	full sharing options/licenses	wysiwyg	fr, m, g, a	global user rating out of 5	pi, c -wel , interests, occuptation, field	10k users	extensive REST	mygrid support
Mendeley http://www.mendeley.com/	reference manager and PDF organizer	fb login/reset	save pdf to library	m (drag and drop)	abstract, tags, city, publisher, type, url, doi	unpublished work not in catalog	-	fo, fr, m, g, h - updates	Awards	pi,CV,pu,awards	30k+ visits a day	extensive REST	program to download and syncup

Appendix B

Source Code

```
786 public function domain_check($email)
    {
788  $uni = $this->input->post('university');
     if (!$this->domain_model->domain_authentication($email,$uni))
790  {
      $this->form_validation->set_message('domain_check', 'Please enter a valid
792   email for your institute, if you do not have a university email address please contact us');
      return FALSE;
794  }
     else
796  {
      return TRUE;
798  }
    }
```

LISTING B.1: 'application/controllers/auth.php-domain_check($email)' validation callback function

```
 60 public function domain_authentication($email, $uni)
    {
 62  $email_part = explode("@", $email);
     //call to evaluate if the email domain matches an institute
 64
     $ret = $this->find_all_Institute2domain($uni,$email_part[1]);
 66  if(!empty($ret)){
      return TRUE;
 68  }
     else{
 70   return FALSE;
     }
 72 }
    public function find_all_Institute2domain($uni,$email=false,$limit=0){
 74  $this->db->select('*');
     $this->db->from('Institute');
 76  $this->db->join('Institute_to_Domain', 'Institute.InstituteId =
     Institute_to_Domain.Institute_InstituteId');
 78  $this->db->join('Domain', 'Institute_to_Domain.Domain_DomainId =
     Domain.DomainId');
 80
     $this->db->where('Institute.InstituteName', $uni);
 82
     if($email==true){
 84   $this->db->where('Domain.DomainName', $email);
     }
 86  if($limit>0){
      $this->db->limit($limit);
 88  }
     $query = $this->db->get();
 90  return $query->result_array();
    }
```

LISTING B.2: 'application/models/domain_model.php' Domain authentication model with active record

```
     $this->form_validation->
676  set_rules('first_name', 'First Name', 'required|xss_clean|trim|alpha_numeric');
     $this->form_validation->
678  set_rules('last_name','Last Name','required|xss_clean|trim|alpha_numeric');
     $this->form_validation->
680  set_rules('email','Email Address',
     'required|valid_email|is_unique[users.email]|callback_domain_check');
682  $this->form_validation->
     set_rules('university','university Name','required|xss_clean');
```

```
684  $this->form_validation->
     set_rules('password','Password','required|min_length['.
686  $this->config->item('min_password_length','ion_auth').']|max_length
     ['.$this->config->item('max_password_length','ion_auth').']|matches[password_confirm]');
688  $this->form_validation->set_rules('password_confirm','Password Confirmation','required');
```

Listing B.3: 'application/controllers/auth.php-create user' Signup form validation rules

```
     //check to see if the group is public (0) or private (1)
158  if(ord($group->is_private)){
      if($this->ion_auth->logged_in()){
160    //if private then is the current user a member?
       $ismember = false;
162    $currentusername = $this->data['user_data']['username'];
       $users = $this->group_model->get_users_in_group($id);
164    foreach($users as $key => $user){
        if($user['username'] == $currentusername){
166       $ismember = true;
        }
168    }
      }
170   //If user not on private guestlist then error
       if(!$this->ion_auth->logged_in() || $ismember == false){
172     show_404();
       }
174  }
```

Listing B.4: 'controllers/groups.php' A group view HTTP request being validated agaisnt the current users permissions

```
112  //load the form based on what the user has input,
     $selectedsearch = $this->input->post('searchradios');
114  $searchdata = html_purify($this->input->post('search'));
     $this->data['selected'] = $selectedsearch;
116
     //not guilty until charged/ assume there are results
118  $this->data['zero_results'] = false;

120  //Perform the Search
     if($selectedsearch == 'group'){
122   $this->data['groupradio']['checked'] = TRUE;
      $searchreturn = $this->groupsearch($searchdata);
124   $this->data['groups'] = $searchreturn;
     }
126  if($selectedsearch == 'paper'){
      $this->data['paperradio']['checked'] = TRUE;
128   $searchreturn = $this->papersearch($searchdata);
      $this->data['papers'] = $searchreturn;
130  }
     if($selectedsearch == 'user'){
132   $this->data['userradio']['checked'] = TRUE;
      $searchreturn = $this->usersearch($searchdata);
134   $this->data['users'] = $searchreturn;
     }
136
     //Global search
138  if ($selectedsearch == NULL) {
      $globalsearch = array();
140   if($this->usersearch($searchdata))
       {$globalsearch += $this->usersearch($searchdata);}
142   if($this->papersearch($searchdata))
       {$globalsearch += $this->papersearch($searchdata);}
```

```
144  if($this->groupsearch($searchdata))
      {$globalsearch += $this->groupsearch($searchdata);}
146 }

148 $this->data['button'] = $selectedsearch;
    $this->data['main_content'] = 'search/index';
150 $this->load->view('templates/template', $this->data);
```

LISTING B.5: 'controllers/search.php' controller search form after submit, with request being handled (corresponding search model not displayed)

```
    <?php if($this->ion_auth->logged_in()) { ?>
 2  <div id="infoMessage"><?php echo $message;?></div>
    <div class="text-box" >
 4   <?php if(isset($thread)){ ?>
     <?php echo form_open('messenger/view_thread/'.$thread[0]['thread_id']);?>
 6   <p>Add participant:<br />
     <input type="text" name="add_part" value="" id="add_part"  />
 8   <img id = "profile" src = "" alt = "profile_pic" /></br></p>
     <input type="hidden" name="userid" id="userid" value="" />
10   <p>
    <script>
12  $(function() {
     $.ajaxSetup({ data: {csrf_token: $.cookie('csrf_cookie')}});
14   $("#add_part").autocomplete({
      source: function(request, response){
16     $.ajax({
       url: "http://localhost/studyswap/messenger/user_suggest",
18     data: {
        term: $("#add_part").val()
20     },
       dataType: "json",
22     type: "POST",
       success: function(data){
24      response(data);
       }
26     });
      },
28    select: function( event, ui ) {
       $('#userid').val(ui.item.id);
30     $.post("http://localhost/studyswap/messenger/get_thumbnail",
        { id: ui.item.id }, function(profilelink) {
32       $("#profile").attr("src", profilelink);
        });
34     },
      minLength: 2
36   });
    });</script>
38  <p><?php echo form_submit('submit', 'Submit');?></p>
    <?php echo form_close();?>
40 </div>
    <h1>View Thread</h1>
42  <?php $msgcount = 0; $subprint = 0 ; ?>Subject:
    <?php foreach ($thread as $key=>$message): ?>
44   <?php if(!$subprint): ?>
       echo $message['subject'];
46     $subprint = 1;
     <?php endif;?>
48   <div class="text-box" >
      Msg: <?php echo ++$msgcount; ?> <?php if($message['status'] == 0){
50     echo anchor('messenger/update_status/'.
       $message['id'].'/1/'.$message['thread_id'],'Unread');
52    }
      else{
```

```
54      echo anchor('messenger/update_status/'.
        $message['id'].'/0/'.$message['thread_id'],'Read');
56     } ?> from: <?php echo $message['username']; ?><br/>

58     Sent:<?php echo elapsed_date($message['cdate']);?></br></br>
       <div class="text-box" ><?php echo $message['body']; ?></br>
60     </div>
      </div>
62   <?php endforeach; ?>
     <div class="text-box" >
64    <?php echo anchor('messenger/reply_message/'.$message['id'],'Reply'); ?>
     </div>
66   <?php } ?>
    <?php } ?>
```

Listing B.6: 'views/messenger/view_thread.php' Messenger view server side PHP code

```
    <div id="infoMessage"></div>
 2  <div class="text-box" >
     <form action="http://localhost/studyswap2/messenger/view_thread/13"
 4    method="post" accept-charset="utf-8">
      <div style="display:none">
 6    <input type="hidden" name="csrf_token" value="34e289f1394ac8370ddc19216f9a0f13" />
      </div><p>
 8    Add participant:<br />
      <input type="text" name="add_part" value="" id="add_part"  />
10    <img id = "profile" src = "" alt = "profile_pic" /></br> </p>
      <input type="hidden" name="userid" id="userid" value="" />
12    <p><input type="submit" name="submit" value="Submit"  /></p>
     </form></div>
14   <h1>View Thread</h1>
     Subject:
16   hello<div class="text-box" >
     Msg: 1 <a href="http://localhost/studyswap2/messenger/update_status/15/0/13">
18   Read</a> from: Murray Wynnes<br/>Sent:11 days 2 hours 13 minutes ago</br></br>
     <div class="text-box" >test message</br></div></div>
20   <div class="text-box" >
    <a href="http://localhost/studyswap2/messenger/reply_message/15">Reply</a></div>
```

Listing B.7: 'views/messenger/view_thread.php' Messenger view with dynamiically generated client side HTML with embedded AJAX (autocomplete) unchanged and so detatched

```
public function add_paper($upload_data,$paper_uploader)
{
 $this->load->helper('url');
 $paper_slug =
  url_title($this->session->userdata('paper_title'), 'dash', TRUE);
 $olddate = $this->session->userdata('paper_date');
 $newdate = date('Y-m-d', strtotime($olddate));
 $papertitle = xss_clean($this->session->userdata('paper_title'));
 $paper_privacy = $this->session->userdata('paper_privacy');
 $papers_authors = $this->session->userdata('paper_authors');
 $paper_data = array(
  'paper_title' => $papertitle,
  'paper_slug' => $paper_slug,
  'paper_location' => $upload_data['file_name'],
  'paper_date' => $newdate,
  'paper_uploaddate' => date('Y-m-d H:i:s'),
  'paper_uploader' => $paper_uploader,
  'is_private' => $paper_privacy
 );
 $this->db->trans_start();
 $this->db->insert('Papers', $paper_data);
 $paper_id = $this->db->insert_id();
 foreach($papers_authors as $foo => $author){
  $author_id = $this->insert_author($author);
  $this->insert_author_paper($author_id,$paper_id);
 }
 $this->db->
  insert('User_Papers', array('user_id' => $paper_data['paper_uploader'],
  'paper_id' => $paper_id));
 $this->db->trans_complete();
 if ($this->db->trans_status() === FALSE)
 {
  echo 'Could not add paper.';
  return FALSE;
 }
 else
 {
  //database stores filneame could be local or in s3, told by flag
  if($paper_privacy==FALSE){ //i.e public access
   $this->store_paper_cloud($paper_data);
  }
  else{ //i.e private access
   $this->store_paper_local($paper_id,$paper_data);
  }
 $this->enter_keywords($papertitle,$this->session->userdata('paper_keyword'));
 }
}
```

LISTING B.8: 'models/papers_model.php' Function to insert a new paper, same transaction method for public or private, into database. Then initiate storage.

```
public function s3_getTemporaryLink($path) {
 $accessKey = $this->config->item('accessKey','s3_config');
 $secretKey = $this->config->item('secretKey','s3_config');
 $bucket = $this->config->item('bucket','s3_config');
 $s3 = new S3(array("accessKey" => $accessKey,'secretKey' => $secretKey));
 return $s3->el_s3_getTemporaryLink($accessKey, $secretKey, $bucket, $path);
}
```

LISTING B.9: 'models/papers_model.php' Gather an expring link to public paper

```
public function insert_spectator($new_user_id,$paper_id){
  $this->db->select('spectator_group_id')->from('Paper_spectator_groups')
  ->where('paper_id',$paper_id)->limit(1);
  $query = $this->db->get();
  if($query->num_rows() > 0){
   $ret = $query->row();
   $data = array(
       'spectator_group_id' => $ret->spectator_group_id,
       'userid' => $new_user_id );
   $this->db->insert('spectators_users', $data);
  }
 }
```

LISTING B.10: 'models/papers_model.php' assignment of role 'spectator' to a user relating restricted paper upload

```
private function insert_institute_domain_link($institute_name,$domain_name){
 $Institute_id_row = $this->
  get_ID_from_name('Institute','InstituteName',$institute_name);
 $InstituteId = $Institute_id_row->InstituteId;
 $Domain_id_row = $this->get_ID_from_name('Domain','DomainName',$domain_name);
 $DomainId = $Domain_id_row->DomainId;

 $this->db->select('*');
 $this->db->from('Institute_to_Domain');
 $this->db->where('Institute_to_Domain.Institute_InstituteId', $InstituteId);
 $this->db->where('Institute_to_Domain.Domain_DomainId', $DomainId);
 $query = $this->db->get();

 if ($query->num_rows() == 0)
 {
 $this->db->set('Institute_to_Domain.Institute_InstituteId', $InstituteId);
 $this->db->set('Institute_to_Domain.Domain_DomainId', $DomainId);
 $this->db->insert('Institute_to_Domain');
 }
}
```

LISTING B.11: 'models/domain_model.php' procedure to insert a new Institute and domain email into database, matching delete is similiar

```
private function store_paper_local($paper_id,$paper_data){
 //create a template for the contributor and spectator roless
 $this->db->insert('Paper_contributor_groups', array('Paper_id' => $paper_id));
 $this->db->insert('contributors_users', array('contributor_group' =>
 $this->db->insert_id(),'user_id' => $paper_data['paper_uploader']));

 //create empty spectator group
 $this->db->insert('Paper_spectator_groups', array('paper_id' => $paper_id));

 $this->session->set_userdata('paper_id', $paper_id);

 $uploadFile = "uploads/papers/".$paper_data['paper_location'];

 $encrypted_string = $this->encrypt->encode(file_get_contents($uploadFile));

 $paper_file = str_replace(SELF,'',FCPATH).$uploadFile;
 unlink($paper_file);

 $handle = fopen($paper_file, 'w') or die('Cannot open file:  '.$paper_file);
 fwrite($handle, $encrypted_string);
}
private function store_paper_cloud($paper_data){
 $config = array("accessKey" =>
```

```
220     $this->config->item('accessKey','s3_config'),
        'secretKey' => $this->config->item('secretKey','s3_config'));
222   $s3 = new S3($config);
      $uploadFile = "uploads/papers/".$paper_data['paper_location'];
224
      // Check if our upload file exists
226   if (!file_exists($uploadFile) || !is_file($uploadFile))
       exit("\nERROR: No such file: $uploadFile\n\n");
228
      //upload to the cloud
230   $s3->setEndpoint("s3-eu-west-1.amazonaws.com");
      $bucket = $this->config->item('bucket','s3_config');
232   $inputFile = $s3->inputFile($uploadFile);
      $s3->putObject($inputFile,$bucket,baseName($uploadFile));
234   unlink($uploadFile);
    }
```

LISTING B.12: 'models/papers_model.php' private functions to either encrypt a private paper and setup RBAC, or to send a public object into local S3 bucket

```
    /**
 2   * AUTHenticated customize privacy values
     *
 4   * Must be a slow process function as a change is very important
     *
 6   * @return void
     * @access public
 8   */
    public function edit_privacy()
10  {
     //validate form input
12   $currentuser = $this->data['user_data']['id'];
     $this->data['title'] = "Privacy controller";
14
     $this->form_validation->set_rules('email', 'Email', 'required|valid_email');
16
     if ($this->form_validation->run() == true)
18   {
       $email_test = $this->input->post('email');
20     if($email_test == $this->data['user_data']['email']){

22      //enter selected radio buttons into privacy object
        $privacy_array = array($this->input->post('about_me'),
24                $this->input->post('education'),
                  $this->input->post('interests'),
26                $this->input->post('skills'),
                  $this->input->post('contact_info'));
28
        //enter the values into the database
30     $this->users_model->enter_privacy_settings($privacy_array,$currentuser);

32     //reset the view back to the users own profile
       $this->profile($this->data['user_data']['username']);
34     }
       else{
36     //the email they provided is not theirs
       $validation_fail = true;
38     }
     }
40   if($this->form_validation->run() == false || isset($validation_fail))
     {
42    //display the form
      //set the flash data error message if there is one
44    $this->data['message'] = (validation_errors()) ?
```

```
    validation_errors() : $this->session->flashdata('message');

    $this->data['email'] = array(
      'name' => 'email',
      'id'   => 'email',
      'type' => 'email',
    );
    //get the default privacy settings to preload the radiobuttons
    $defaults = $this->users_model->get_privacy_values($currentuser);
    $this->data['privacy'] = $defaults;

    //render
    $this->data['main_content'] = 'users/edit_privacy';
    $this->load->view('templates/template', $this->data);
  }
}
```

Listing B.13: 'controllers/users.php' Systematic procedure for generating update form regarding a profile pages module appearence

```
<?php class Ajax_handle extends CI_Model {
 public function __construct()
 {
  parent::__construct();
  $this->load->helper('htmlpurifier_helper');
 }

 //auto complete the user field
 function ajax_user_suggest($term){
   $term = html_purify($term);
    $this->db->like("concat(' ',first_name,last_name)" , $term, 'both');
    $this->db->limit(5);
    $query = $this->db->get('users');
    $keywords = array();
    $output = "";
    foreach($query->result() as $row){
  $keywords['id'] = $row->id;
  $keywords['value'] = $row->first_name." ".$row->last_name;
  $output = json_encode($keywords).','.$output;
    }
    return '['.substr_replace($output ,"",-2).']';
 }
 //once a user is selected on the autocomplete dropdown hit them up with a picture
 function get_ajax_thumbnail($userid){
   $userid = html_purify($userid);
   $this->load->model('gallery_model');
   $this->load->model('users_model');
   $user = $this->users_model->get_user_short($userid);
   $email = $user->email;
   if($this->gallery_model->get_profile_type_selected($userid) == 0){
    $link = base_url().'images/thumbs/'.$this->gallery_model->get_profile_pic($userid);
   }
   else{
    $link = $this->users_model->
    get_gravatar($email,$s=80,$d='mm',$r ='g',$img=false,$atts=array());
   }
   return $link;
 }
}
```

Listing B.14: 'models/ajax_handle.php' receptor functions for background AJAX POST linked to form inputs

```
public function clean_uploads(){
 if ($handle = opendir('uploads/decrypted')) {
  while (false !== ($file = readdir($handle))) {
   $filelastmodified = filemtime($file);
   if((time()-$filelastmodified) > 1)
   {
      unlink($file);
   }
  }
  closedir($handle);
 }
}
```

Listing B.15: 'controllers/cron.php' background scheduled cleanup for private paper files

```
# setup - remove previous rules
iptables -F

# allow loopback port for local adapter connects
iptables -I INPUT -i lo -j ACCEPT

# allow us of email
# enable SSH,INCOMINGSMTP,POP3,IMAP,SSMTP,SSLIMAP,IMAP4,SSL-POPTRAFFIC
iptables -A INPUT -p tcp --dport ssh -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 25 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 110 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 143 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 465 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 585 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 993 -i eth0 -j ACCEPT
iptables -A INPUT -p tcp --dport 995 -i eth0 -j ACCEPT

# Allow web browsing
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Block other traffic
iptables -A INPUT -j DROP
iptables -A OUTPUT -j ACCEPT
iptables -A FORWARD -j DROP
```

Listing B.16: Sample iptable firewall rules

```
public function insert_valid_test_user(){
 $username = "testuser".md5(time() + mt_rand(0, 1999999999));
 $email = substr($username, 0, 16)."@bris.ac.uk";

 $additional_data = array(
  'first_name' => substr($username, 0, 16),
  'last_name' => substr($username, 16, 8),
  'university' => 'University of Bristol',
 );
 $password = substr(md5($username),0,20);
 $this->ion_auth->register($username, $password, $email, $additional_data);
}
for ($i = 1; $i <= 100; $i++) {
 $this->insert_valid_test_user();
}
```

Listing B.17: Procedure to stress test the MYSQL database with a population of valid users into database

```
//check to see if a user is a membeer fo the contributer group attached to paper
 public function is_contributor($user_id,$paper_id){
  $this->db->select('*');
  $this->db->from('contributors_users');
  $this->db->join('Paper_contributor_groups', 'contributors_users.contributor_group
  = Paper_contributor_groups.contributor_group_id');
  $this->db->where('contributors_users.user_id', $user_id);
  $this->db->where('Paper_id', $paper_id);

  $query = $this->db->get();
  if($query->num_rows() > 0){
   return true;
  }
  else{
   return false;
  }
 }
```

Listing B.18: 'models/papers_model.php' RBAC check for contributor role permissions before initiating a user session with private paper

```
<div id="paper-further-details">

 Date: <?php echo date('F d, Y', strtotime($paper_item['Paper_date']));?><br/>

 Authors:
  <?php $comma=false; foreach($author[$paper_item['Paper_ID']] as $author_item):
            if($comma) echo ","; $comma=true;?>
            <a href="linktopapersbyauthor">
     <?php echo $author_item['Author_Name']; ?></a>
  <?php endforeach; ?><br/>

 Uploaded by : <a href="linktoprofile">I.M.Todo</a> on
 <?php echo date('F d, Y', strtotime($paper_item['Paper_uploaddate']));?><br/>


  <i class="icon-download-alt"></i> Download<br/>

<ul id="sample-menu-1" class="sf-menu">
 <li class="current">
  <a href="#a"><i class="icon-share"></i> Share <i class="icon-caret-down"></i></a>
  <ul>
   <li><a href="#">Facebook <i class="icon-facebook sf-icon"></i></a></li>
   <li><a href="#">Twitter <i class="icon-twitter sf-icon"></i></a></li>
   <li><a href="#">Email <i class="icon-envelope-alt sf-icon"></i></a></li>
  </ul>
 </li>
</ul>

</div><!--/#paper-further-details-->

</div><!--/#paper-info-->

<?php if(isset($Paperfile)){ ?>
<embed src="<?php echo 'http://localhost/studyswap/uploads/decrypted/'.
$Paperfile; ?>" class="pdf-embed" />
 <?php } else{?>
 <embed src="<?php echo $Paper_URL; ?>" class="pdf-embed" />
<?php } ?>
```

Listing B.19: 'views/papers/view.php' The paper view forms the reader, metdata and header objects as components of the overall composite view

Appendix C

Testing Material

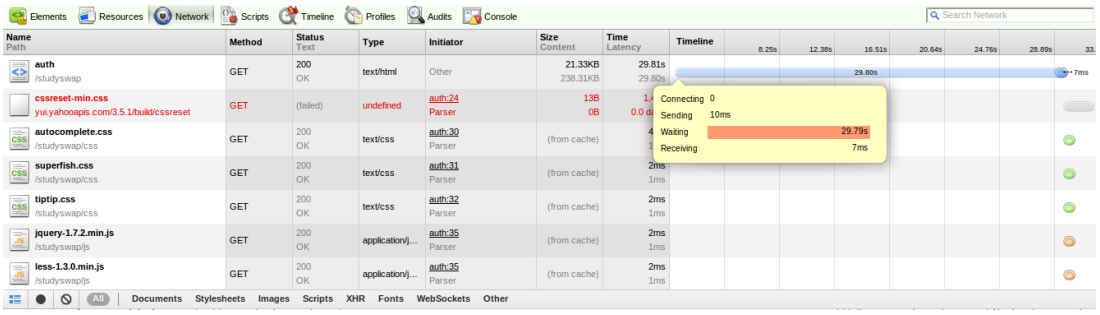


FIGURE C.1: Output from Google Chrome network profiler - time taken to register one thousand users into the database, and send SMTP authentication emails.

TABLE C.1: 100 GET HTTP /1.1 Transactions with 100% Success Rate

Users	Elapsed (s)	(Transactions/s)	Concurrency	Data Transfer(MB)	Throughput(MB/s)
100	12.16	8.22	3.87	0.61	0.05
200	14.27	7.01	6.27	1.02	0.07
300	17.96	5.57	5.32	1.43	0.08
400	19.91	5.02	6.41	1.67	0.08
500	19.42	5.15	6.58	1.88	0.1
600	19.15	5.22	5.95	2.13	0.11
700	21.39	4.68	6.42	2.35	0.11
800	23.88	4.19	6.91	2.54	0.11
900	20.67	4.84	6.95	2.74	0.13
1000	20.92	4.78	6.58	2.94	0.14

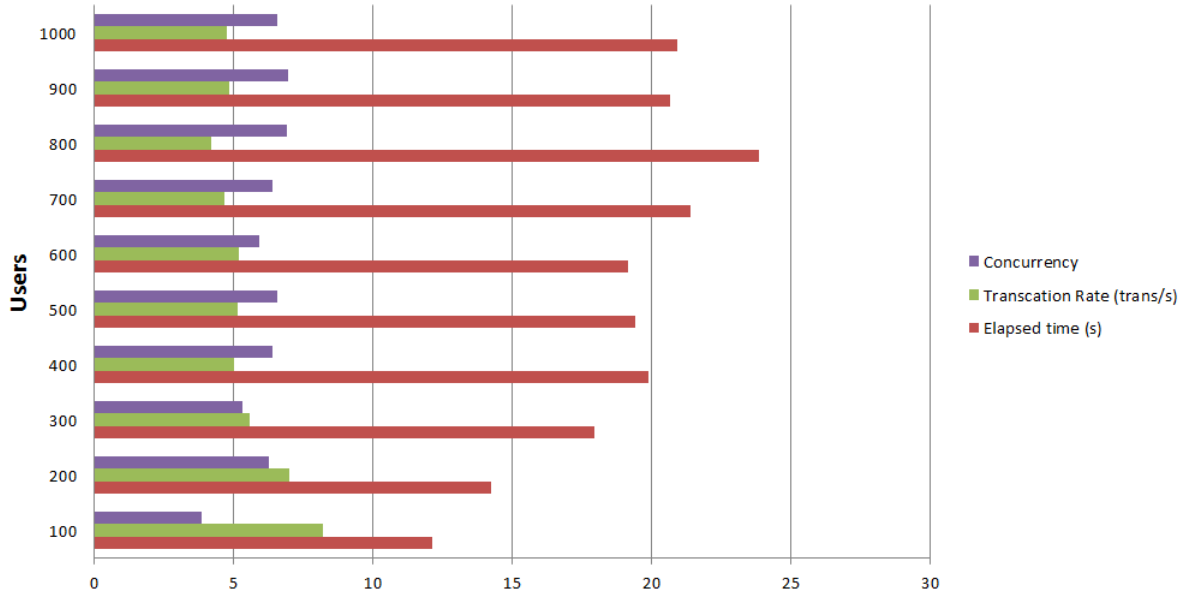


FIGURE C.2: domain/users/index.php (HTTP /1.1 200) Load Testing Chart from table C.1

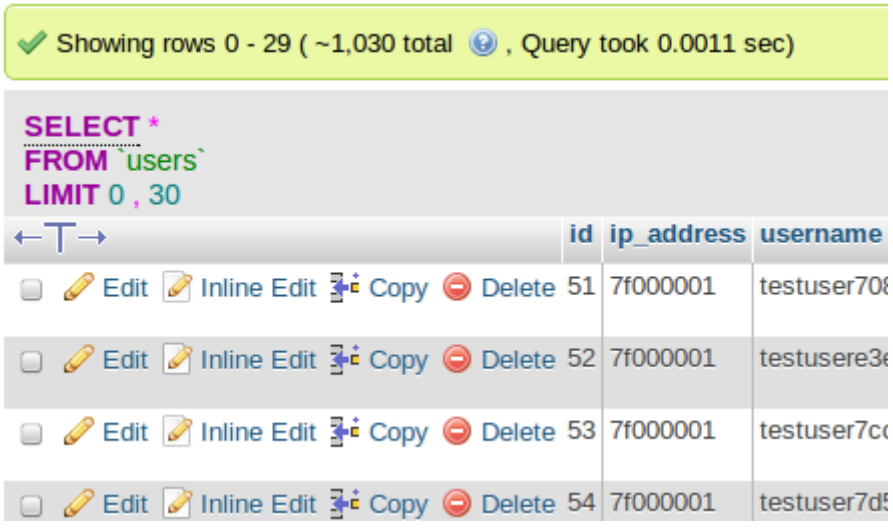


FIGURE C.3: An optimized SQL select on users, with result caching and limit set

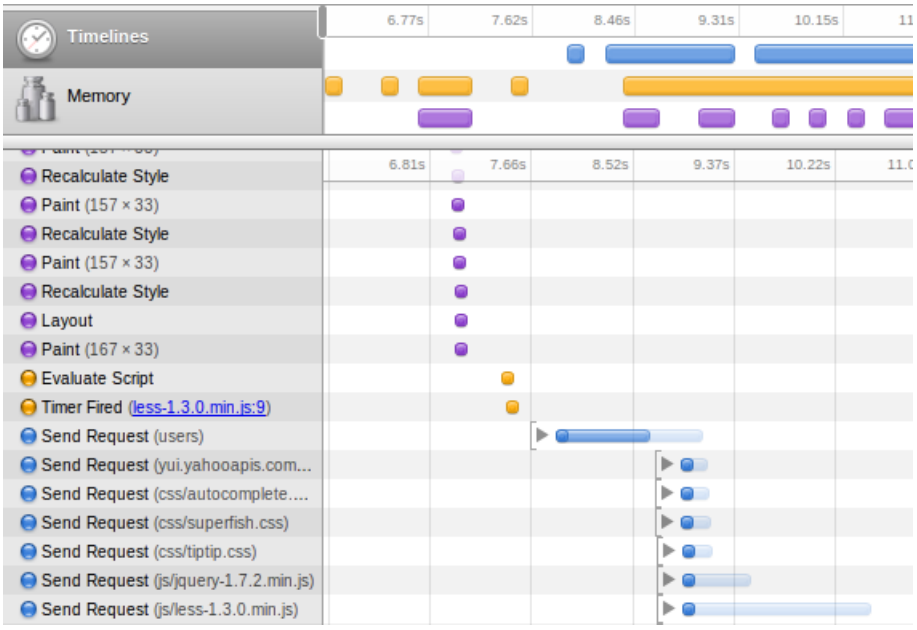


FIGURE C.4: Page load timings before limiting the output of data results

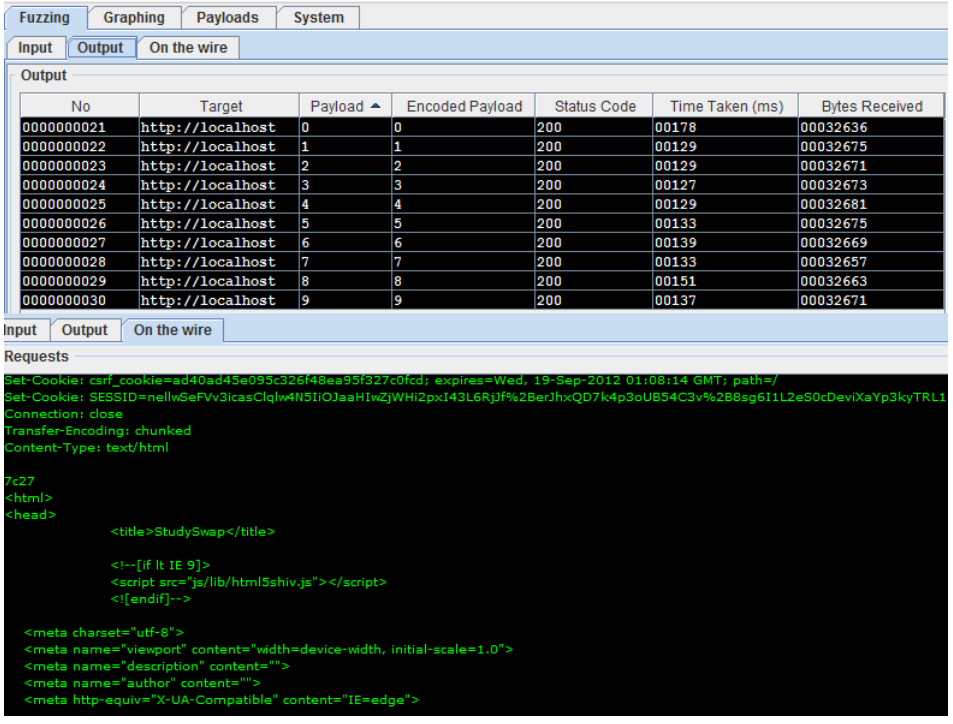


FIGURE C.5: Fuzz testing to show correct handling of invalid HTTP protocol versions

FIGURE C.6: A simple few trace timing attack against the privacy form load logic

Appendix D

Additional Documents

D.1 Server setup

For clients to be able to connect to our service we need to consider options for hosting the site. For testing purposes we have chosen to implement an instance of Amazon Elastic Compute 2 [43], a virtual hosting platform or IaaS. Minimal hardware is provided from a micro instance, such as 613mb of RAM and a preconfigured template called an Amazon Machine Image (AMI) contains the operating system and libraries to be installed. We have chosen to install the 64 bit Ubuntu Cloud Guest Server 12.04.1 AMI, and this initially allows only SSH access using pre-arranged RSA public and private keys. A web service provided by amazon has built in port restrictors labelled as security groups, for example this can be configured to allow all I.Pv6 access on port 80. New analysis on the strength of AMI templates show the difficulty in protecting against unauthorized access [124]. We will be using a LAMP stack where the Apache server can be configured with modules such as *cURL* or a rewrite engine for additional functionality. A difficult type of attack to prevent is a DoS which can even be distributed over many source addresses, a method to mitigate against this is to time out clients or set limits on the multi-processing Apache modules that handle concurrent connections. The software *git* also needs to be installed so that we can push our latest test branch to a bare git repository that has remote origin 'EC2 instance'.

For the structure of the web pages we are conforming to the XHTML 5 standard, with HTML5 so pages are compatible for older browsers. Therefore the server must be configured to handle content negotiation of the header type. This is so that browsers treat pages as the tighter XHTML and view pages correctly. The server database needs to be installed using MySQL, and then seeded with the administrator users, institute lists and security groups. For clients to authenticate the identity of a website a certificate must be issued from a trusted third party to our validated domain. I've tested TLS with Apache2 as seen in figure D.1, by acting as a CA and self signing a certificate using open-ssl; the SSL module and certificate keys then need to be installed onto the server properly. Although there are security issues with HTTPS certificates: one is that certificate authorities (CA) are constantly under attack, another being CAs need to uphold their contracts [125]. A free P2P CA 'CAcert' issues different strength certificates either automatically (weak) by just domain name or manually with real life verification.

A firewall is installed between the server and the Internet to control TCP/IP network traffic access. The software based Linux firewall is known as 'iptables', and handles packets using a chain of rules in order. Listing B.16 shows a firewall configuration with restrictive ingress filters comparable to our deployment servers, but for security reasons details are overlooked. It might be safe now to advertise chosen methods of security and not rely on'security by obscurity', but there are many reasons why this is a really bad idea. For one attacks always get better,

and as this converges with Moore's law of increased computation power, it's only a matter of circumstance which will render our system vunerable and therefore service unsafe. An extreme method of securing server access during development is to use 'port knocking' where the server is set to appear to drop all inbound packets on all ports. Only a correct sequence of incomming packets on agreed ports gains access, by dynamically updating rules to the firewall.

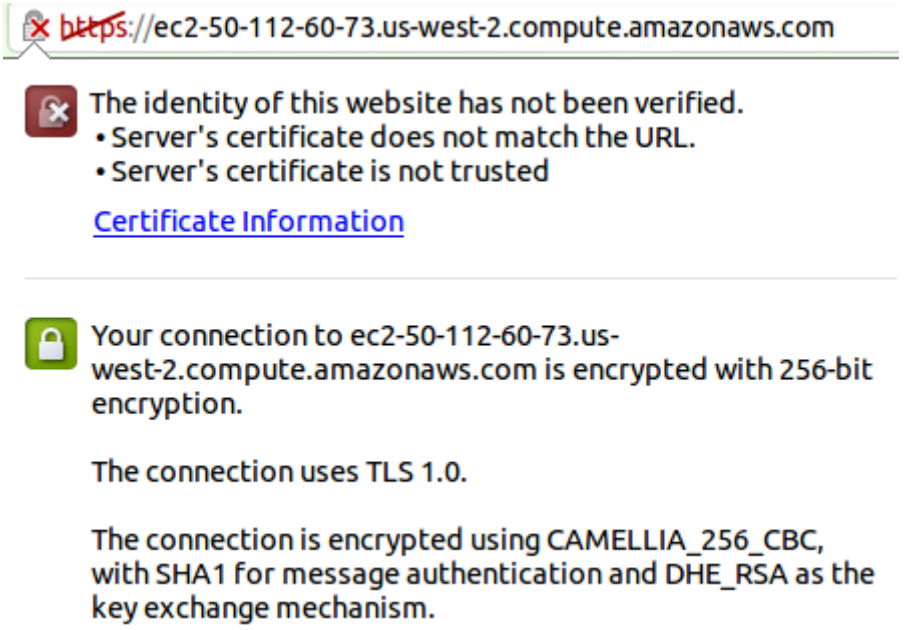


FIGURE D.1: TLS response with unverified certificate identity

D.2 Administrator Guide

D.2.1 System Adminitrator

Information security participants for IT systems can involve the roles for management (senior/IT/security), designers and employees. Naturally there are even different levels of and meaning to the term 'Admin' of our system. The user 'Admin' is allowed application privileges to manage users and moderate institute data. A more elevated level of authority is the system security administrator, which should be granted log access, feasibly an admin SQL account, SSH keys for server shell access to then configure and secure the system, delegating authority to others when necessary. The system will suffer from various forms of erosion: hardware faults, operating system updates, library updates, memory leaks, bugs and errors found in the code. Depending on the platform, it therefore will be necessary for this role to apply software patches. The largest dependency is PHP, which can be updated using a packet manager. The server should be remote so that nobody

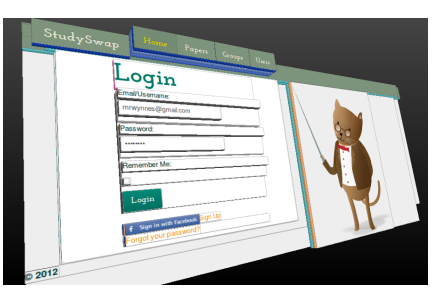
can physically access it, such is the guarantee and nature of a virtual EC2 instance. The database can be securely backed-up automatically using cron jobs or manually with exporting SQL. A system administrator should actively monitor the system, which might involve identifying intrusions attempts such as SQL injection; but its encouraged to resist overly secure measures until these attacks are detected.

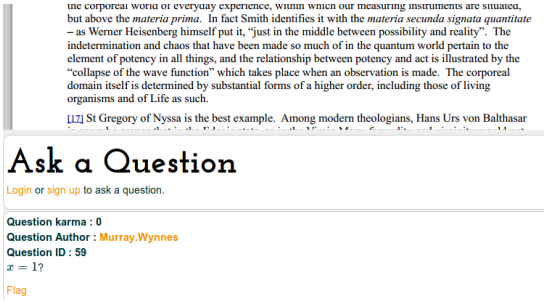
D.3 Legal Assessment

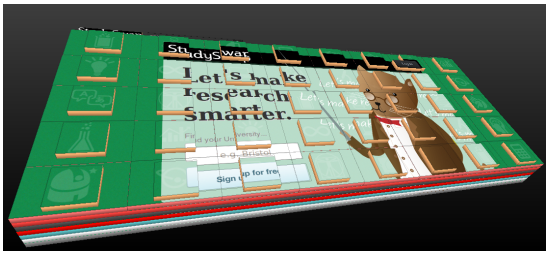
A main objective of the development is to recognize legitimate techniques and to meet all security requirements by providing adequate solutions. The system is non-profit and so terms and conditions and business trading laws will not apply, trademark and domain name issues should also be of insignificant concern. In the event that we decide to register for a domain name its unlikely that, if we trade with a reputable service, that we purchase an already registered domain and require *ICANN UDRP*. More attention will however need to be taken if we decide to extend present services; the contractual relationship is then determined by the licensing and the jurisdiction which can affect intellectual property rights. We have briefly visited the patenting and copyright position of software and our strategy has been to investigate any copyright licensing of code we've used before deciding whether it is appropriate (open source) or proprietary. In order to store any information in which people can be identified a data controller is subject to conformity of the *Data Protection Act* (1998) (DPA), this will induce concerns for inference of data from statistical records. Universities act in according with these laws: a student cannot request their paper as they are do not contain data about them, but marks and comments can be returned. Of more interest to us is that personal research data can be archived if it is believed to have future use, and there is no need to inform the author of this action. A minimalistic approach is to simply act as a controller and avoid any processing of the data or private user items held. Where processing means "amending, augmenting, deleting or re-arranging the data or extracting the information constituting the data. As well as this the data should be obtained when necessary, held for a similar period of time and be secured with the users being told exactly how its used. It's possible the data may never be disclosed, but could be under (s.35 DPA) in the event that the *Further and Higher Education Act* (1992) s.79 is mentioned in a court order. As part of an ongoing customer service post site installation, subjects may access their data if they provide the correct request format and inability for us to respond will result in ICO enforcement under the *Criminal Justice and Immigration Act* (2008).

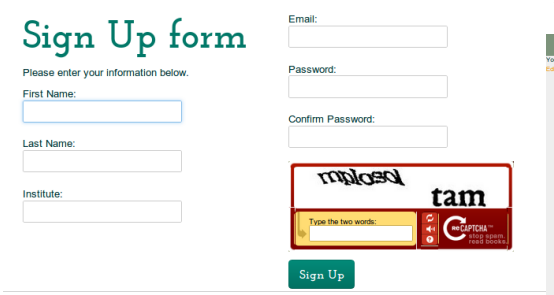
There have been seminal legal updates regarding security, one such being the *Electronic Communications Act* (2000). It provides a legal framework for approving and legally recognizing cryptography. To avoid issues with lack of protection from legal documentation, certain key generation procedures have had to be altered to

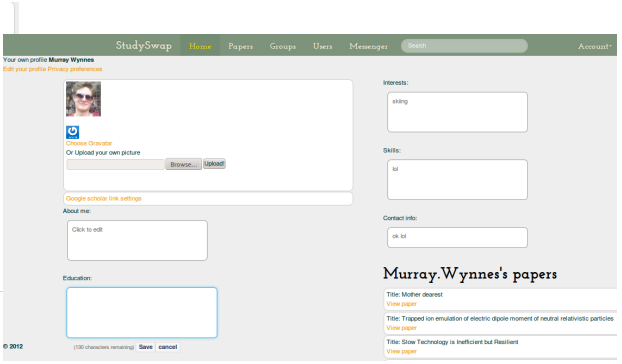
protect entities. For example needing a different key pair for incoming and outgoing messages [126], which may be disclosed under the *RIPA* act. The *Regulation of investigatory powers act* (2000) controls the interception of communications in the workplace and if an individual operates within our institution without authority then they could be in breach of the *Computer Misuse Act*. Any commercial communication on our website must be clearly identifiable as such with full details of promotional offers sent by email in an unambiguous manner. As this generally does not yet apply to us, a more suitable regulation concerning emails is not to send unintended SPAM perhaps during authentication. If a user has an account this may or may not be under a subscription contract depending on our 'End User License Agreement', and any individual who has not explicitly provided subscriber consent should not receive unsolicited marketing. The key legislation in regards to our system is the *Computer Misuse Act* (1990) deeming unauthorized access an offense, especially with the intent to commit an offense or aid one, and also serves to deters unauthorised modification of the contents of the computer system.











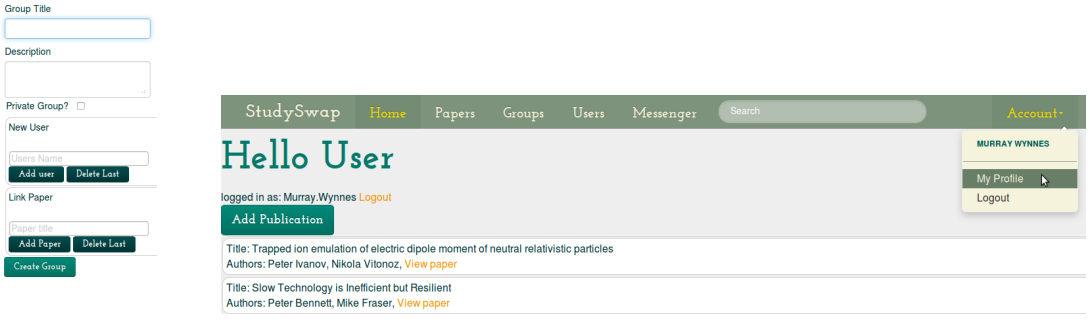


FIGURE D.2: Processed wireframes and sample client end page views

Bibliography

[1] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(40):35–41, March 1977.

[2] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[3] Y.Harande. Author productivity and collaboration: an investigation of the relationship using the literature of technology. 2001.

[4] Report of the Working Group on Expanding Access to Published Research Findings. Accessibility, sustainability, excellence: how to expand access to research publicastions, 2012. URL http://www.researchinfonet.org/wp-content/uploads/2012/06/Finch-Group-report-FINAL-VERSION.pdf.

[5] Innovation Department for Buisness and Skills. Government Response to the Finch Group Report. URL http://bit.ly/QCOu7c.

[6] European Science Foundation MO Forum on Peer Review, 2012. URL http://www.esf.org/activities/mo-fora/peer-review.html.

[7] Public Knowledge Project, 2012. URL http://pkp.sfu.ca/.

[8] M.Bernstein. Reviewing conference papers. 2008. URL http://www.markbernstein.org/elements/Reviewing.pdf.

[9] H.Jiang M.Carpenter, M.Li. Social network research in organizational contexts: A systematic review of methodological issues and choices. *Journal of Management*, 2012.

[10] Adam Wierzbicki. Behavior dynamics in media-sharing social networks by h. vicky zhao, w. sabrina lin and k. j. ray liu. *J. Artificial Societies and Social Simulation*, 15(1), 2012. URL http://dblp.uni-trier.de/db/journals/jasss/jasss15.html#Wierzbicki12.

[11] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005. ISSN 0028-0836. doi: 10.1038/nature03607.

[12] L.Lamport. *LaTeX: A document preparation system, User's guide and reference manual*. 1997. URL http://bbs.dartmouth.edu/~fangq/MATH/download/book/latex.pdf.

[13] *University of Cambridge* latex advocacy. URL http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/latex_advocacy.html.

[14] J.Cain. Online social networking issues within academia and pharmacy education. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2254235/.

[15] G.Merchant. Unravelling the social network: theory and research. *Learning, Media and Technology*, 2011.

[16] Robert K. Merton. The matthew effect in science. *Science*, 159(3810): 56–63, 1968. URL http://www.garfield.library.upenn.edu/merton/matthew1.pdf.

[17] Miray Kas, Kathleen M. Carley, and L. Richard Carley. Trends in science networks: understanding structures and statistics of scientific networks. *Social Netw. Analys. Mining*, 2(2):169–187, 2012. URL http://www.andrew.cmu.edu/user/mkas/papers/snam_paper.pdf.

[18] Katy Börner, Jeegar T. Maru, and Robert L. Goldstone. The simultaneous evolution of author and paper networks, November 2003. URL http://arxiv.org/abs/cond-mat/0311459.

[19] RAE. *Research Assessment Exercise 01/2008: the outcome (December)*. 2008.

[20] Universities UK. *The use of bibliometrics to measure research quality in UK higher education institutions*. 2009. URL http://www.universitiesuk.ac.uk/Publications/Documents/bibliometrics.pdf.

[21] Google - data centers, 2012. URL http://www.google.com/about/datacenters/index.html.

[22] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569–16572, 2005. doi: 10.1073/pnas.0507655102. URL http://www.pnas.org/content/102/46/16569.abstract.

[23] SciVerse - Open to accelerate science - Scopus API. URL http://www.info.sciverse.com/scopus/scopus-in-detail/tools/api.

[24] Catherine Dwyer, Starr Roxanne Hiltz, and Katia Passerini. Trust and Privacy Concern Within Social Networking Sites: A Comparison of Facebook and MySpace. In John A. Hoxmeier and Stephen Hayne, editors, *AMCIS*, page 339. Association for Information Systems, 2007. URL http://csis.pace.edu/~dwyer/research/DwyerAMCIS2007.pdf.

[25] We're quitting facebook. URL http://www.quitfacebookday.com/.

[26] OfCom. Social networking: A quantitative and qualitative research report into attitudes, behaviours and use. 2008. URL http://news.bbc.co.uk/1/shared/bsp/hi/pdfs/02_04_08_ofcom.pdf.

[27] B.Michael M.Michelle, J.Lupe. *The Failure of Online Social Network Privacy Settings*. Columbia University Computer Science Technical Reports, 2011. URL http://academiccommons.columbia.edu/catalog/ac:135406.

[28] OAuth - An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications. URL http://oauth.net/.

[29] Pubmed - more than 22 million citations for biomedical literature from medline, life science journals, and online books, 2012. URL http://www.ncbi.nlm.nih.gov/pubmed.

[30] Open study - ask questions, give help, and connect with over 100,000 students from 170 countries and 1,600 schools., 2012. URL http://openstudy.com/.

[31] Khan academy - watch. practice. learn almost anything for free., 2012. URL http://www.khanacademy.org.

[32] Internet assigned numbers authority. URL http://www.nro.net/ipv6/nro_depletion_deployment_faq.

[33] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly, Beijing, 2007. ISBN 978-0-596-52926-0. URL http://my.safaribooksonline.com/9780596529260.

[34] Berne. *The Law of International Copyright London*. 1886. URL http://www.ipo.gov.uk/types/copy/c-about/c-history/c-history-1886.htm.

[35] Gnu affero general public license, 2007. URL http://www.gnu.org/licenses/agpl-3.0.html.

[36] Elgg - open source social networking engine, 2012. URL http://www.elgg.org/.

[37] Berners-Lee. WWW: Past, present, and future. *COMPUTER: IEEE Computer*, 29, 1996.

[38] R.Ku. The creative destruction of copyright: Napster and the new economics of digital technology. 2002. URL http://papers.ssrn.com/sol3/papers.cfm?abstract_id=266964.

[39] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The BitTorrent P2P file-sharing system: Measurements and analysis. In *Proc. 4th International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA, February 2005.

[40] Cristian Lumezanu, Randolph Baden, Neil Spring, and Bobby Bhattacharjee. Triangle inequality variations in the internet. In Anja Feldmann and Laurent Mathy, editors, *Internet Measurement Conference*, pages 177–183. ACM, 2009. ISBN 978-1-60558-771-4. URL http://dblp.uni-trier.de/db/conf/imc/imc2009.html#LumezanuBSB09.

[41] Le-Hung Vu, Karl Aberer, Sonja Buchegger, and Anwitaman Datta. Enabling secure secret sharing in distributed online social networks. In *ACSAC*, pages 419–428. IEEE Computer Society, 2009. ISBN 978-0-7695-3919-5. URL http://dblp.uni-trier.de/db/conf/acsac/acsac2009.html#VuABD09.

[42] Diaspora - Distributed open source social networking software, 2012. URL https://github.com/diaspora.

[43] Amazon Web Services, 2012. URL http://aws.amazon.com/.

[44] David Thomas and David Heinemeier Hansson. *Agile web development with Rails - a pragmatic guide*. Pragmatic Bookshelf, 2005. ISBN 978-0-9766940-0-7.

[45] Social Go - create a beautiful website, 2012. URL http://www.socialgo.com/.

[46] Carlo Curino, Evan P. C. Jones, Raluca A. Popa, Nirmesh Malviya, Eugene Wu 0002, Samuel Madden, Hari Balakrishnan, and Nickolai Zeldovich. Relational cloud: a database service for the cloud. In *CIDR*, pages 235–240. www.cidrdb.org, 2011. URL http://dblp.uni-trier.de/db/conf/cidr/cidr2011.html#CurinoJPMWMBZ11.

[47] D.Brandtt. Networking and scalability in eve online, 2005. URL http://dl.acm.org/citation.cfm?id=1306839.

[48] Rackspace - cloudhosting, 2012. URL http://www.rackspace.co.uk/.

[49] Kristina Chodorow and Michael Dirolf. *MongoDB - The Definitive Guide: Powerful and Scalable Data Storage.* O'Reilly, 2010. ISBN 978-1-449-38156-1.

[50] Redis - key-value store. URL http://redis.io/.

[51] Cormac Herley and Paul C. van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security and Privacy*, 10(1):28–36, 2012. URL http://dblp.uni-trier.de/db/journals/ieeesp/ieeesp10.html#HerleyO12.

[52] CAPTCHA Mechanism - Telling Humans and Computers Apart Automatically, 2012. URL http://www.captcha.net/.

[53] M.DelMonte. *Email Authentication System.* 2002. URL http://www.google.com/patents/US20040024823.

[54] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network security: private communication in a public world, second edition.* Prentice Hall Press, Upper Saddle River, NJ, USA, second edition, 2002. ISBN 9780137155880.

[55] Simson L. Garfinkel. Email-based identification and authentication: An alternative to pki? *IEEE Security & Privacy*, 1(6):20–26, 2003. URL http://dblp.uni-trier.de/db/journals/ieeesp/ieeesp1.html#Garfinkel03.

[56] CSI. Computer crime and security survey, 2010. URL http://gocsi.com/survey.

[57] Masoud Bekravi, Shahram Jamali, and Gholam Shaker. Defense against syn-flood denial of service attacks based on learning automata. *CoRR*, abs/1208.5037, 2012. URL http://dblp.uni-trier.de/db/journals/corr/corr1208.html#abs-1208-5037.

[58] William Stallings. *Cryptography and network security - principles and practice (4. ed.).* Prentice Hall, 2005. ISBN 978-0-13-111502-6.

[59] PHP Manual, 2012. URL http://www.php.net/manual/en/index.php.

[60] HipHop - PHP Translator into C++, 2010. URL https://developers.facebook.com/blog/post/2010/02/02/hiphop-for-php--move-fast/.

[61] Git - distributed revision control and source code management. URL http://git-scm.com/.

[62] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA, 1995. ISBN 978-0-201-63361-0.

[63] CodeIgniter - an open source PHP web application framework. URL http://codeigniter.com/.

[64] Open Source Initiative OSI - the Open Source License 3.0. URL http://opensource.org/licenses/osl-3.0.php.

[65] Bristol Applications and Tools - Bristol goes Google. URL http://www.bristol.ac.uk/it-services/applications/google-apps/.

[66] P.Leach T.Berners-Lee. RFC Hypertext Transfer Protocol - HTTP/1.1. URL http://tools.ietf.org/html/rfc2616.

[67] Paco Hope and Ben Walther. *Web security testing cookbook - systematic techniques to find problems fast.* O'Reilly, 2008. ISBN 978-0-596-51483-9.

[68] Exploiting a cross-site scripting vulnerability on facebook. URL http://www.acunetix.com/websitesecurity/xss-facebook.htm.

[69] Owasp top 10 most critical web application security risks, 2010. URL https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[70] W3 schools web statistics browser statistics. URL http://www.w3schools.com/browsers/browsers_stats.asp.

[71] R. Sekar Riccardo Pelizzi. Protection, usability and improvements in reflected xss filters. URL http://seclab.cs.sunysb.edu/seclab/pubs/xss.pdf.

[72] HTML Purifier, standards-compliant HTML filtering, 2012. URL http://htmlpurifier.org/.

[73] Joel Weinberger, Prateek Saxena, Devdatta Akhawe, Matthew Finifter, Eui Chul Richard Shin, and Dawn Song. A systematic analysis of xss sanitization in web application frameworks. In Vijay Atluri and Claudia Daz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 150–171. Springer, 2011. ISBN 978-3-642-23821-5. URL http://dblp.uni-trier.de/db/conf/esorics/esorics2011.html#WeinbergerSAFSS11.

[74] MySQL.com and Sun hacked through SQL injection. URL http://nakedsecurity.sophos.com/2011/03/27/mysql-com-and-sun-hacked-through-sql-injection/.

[75] Open Web Application Security Project, 2012. URL https://www.owasp.org.

[76] IonAuth - simple and lightweight auth system based on Redux Auth 2, 2012. URL https://github.com/benedmunds/CodeIgniter-Ion-Auth.

[77] Web cryptography api, 2012. URL http://www.w3.org/TR/WebCryptoAPI/.

[78] Apache License, version 2.0, 2012. URL http://www.apache.org/licenses/LICENSE-2.0.

[79] Campaign monitor - css support for email clients, 2012. URL http://www.campaignmonitor.com/css/.

[80] reCAPTCHA - free anti-bot service to help digitize books. URL http://www.google.com/recaptcha.

[81] M.Bellare S.GoldWasser. Lecture notes on cryptography. University Lecture, 2008. URL http://cseweb.ucsd.edu/~mihir/papers/gb.pdf.

[82] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen K. Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. *IACR Cryptology ePrint Archive*, 2009:111, 2009. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2009.html#StevensSALMOW09.

[83] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005-11-04. ISBN 3-540-28114-2. URL http://dblp.uni-trier.de/db/conf/crypto/crypto2005.html#WangYY05a.

[84] *Secure Hash Standard SHS*. National Institute of Standards and Technology, Gaithersburg, 2008. URL http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf. Federal Information Processing Standard 180-2.

[85] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. URL http://www.ietf.org/rfc/rfc5246.txt. Upyeard by RFCs 5746, 5878, 6176.

[86] National institute˜of˜standards˜and technology. Announcing request for candiyear algorithm nominations for a new cryptographic hash algorithm (SHA-3) Family. Technical report, DEPARTMENT OF COMMERCE, November 2007. URL http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf.

[87] Niels Provos and David Mazires. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91. USENIX, 2003-09-04. ISBN 1-880446-32-4. URL http://dblp.uni-trier.de/db/conf/usenix/usenix1999f.html#ProvosM99.

[88] Top 500 super computer lists - sequoia - bluegene/q, power bqc 16c 1.60 ghz, 2012. URL http://i.top500.org/system/177556.

[89] Orhun Kara and Cevat Manap. A new class of weak keys for blowfish. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2007-09-03. ISBN 978-3-540-74617-1. URL http://dblp.uni-trier.de/db/conf/fse/fse2007.html#KaraM07.

[90] Firesheep - a firefox extension that demonstrates http session hijacking attacks. URL http://codebutler.github.com/firesheep/.

[91] The glibc random number generator. URL http://www.mscs.dal.ca/~selinger/random/.

[92] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998. ISSN 1049-3301. doi: 10.1145/272991.272995. URL http://dx.doi.org/10.1145/272991.272995.

[93] Mersenne twister home page, a very fast random number generator. URL http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.

[94] Michael Neve and Kris Tiri. On the complexity of side-channel attacks on aes-256 - methodology and quantitative results on cache attacks. *IACR Cryptology ePrint Archive*, 2007:318, 2007. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2007.html#NeveT07.

[95] Jens-Peter Kaps and Rajesh Velegalati. Dpa resistant aes on fpga using partial ddl. In Ron Sass and Russell Tessier, editors, *FCCM*, pages 273–280. IEEE Computer Society, 2010. ISBN 978-0-7695-4056-6. URL http://dblp.uni-trier.de/db/conf/fccm/fccm2010.html#KapsV10.

[96] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. *IACR Cryptology ePrint Archive*, 2011:449, 2011. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2011.html#BogdanovKR11.

[97] wikipedia - an estimation of super computers ability to exhaust AES key space. URL http://en.wikipedia.org/wiki/Brute-force_attack#Theoreticallimits.

[98] Razvi Doomun, Jayramsingh Doma, and Sundeep Tengur. Aes-cbc software execution optimization. *CoRR*, abs/1208.3227, 2012. URL http://arxiv.org/ftp/arxiv/papers/1208/1208.3227.pdf.

[99] E. U. Directive. 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector. *Official Journal of the European Communities*, 2002.

[100] E. U. Directive. 2009/136/EC amending Directive 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector and Regulation (EC) No 2006/2004 , Directive 2002/22/EC. *Official Journal of the European Union*, 2009.

[101] Privacy and electronic communication regulations - guidance on the rules on use of cookies and similiar technologies. URL http://bit.ly/NjDScW.

[102] IBM P. Hunt Oracle Corporation T. Lodderstedt, M. McGloin. Oauth 2.0 threat model and security considerations draft-ietf-oauth-v2-threatmodel-07, August 2012. URL http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel-07.

[103] Year 2038 bug/problem on UNIX systems. URL http://en.wikipedia.org/wiki/Year_2038_problem.

[104] Theo Hrder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, 1983. URL http://dblp.uni-trier.de/db/journals/csur/csur15.html#HarderR83.

[105] European Commission - Justice. URL www.ec.europa.eu/justice.

[106] Mathjax is an open source javascript display engine for mathematics that works in all modern browsers. URL http://www.mathjax.org/.

[107] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[108] John F. Barkley. Comparing simple role based access control models and access control lists. In *ACM Workshop on Role-Based Access Control*, pages 127–132, 2002-12-17. URL http://dblp.uni-trier.de/db/conf/rbac/rbac1997.html#Barkley97.

[109] András Belokosztolszki. Role-based access control policy administration. *Technical Report UCAM-CL-TR-586*, University of Cambridge, Computer Laboratory, 2004.

[110] Gravatar, a globally recognized avatar, 2012. URL https://en.gravatar.com/.

[111] Jeditable edit in place plugin for jquery, 2012. URL http://www.appelsiini.net/projects/jeditable.

[112] Mahana messaging library for codeigniter by jr-madsen67. URL https://github.com/jrmadsen67/Mahana-Messaging-library-for-CodeIgniter/.

[113] PHPUnit - unit testing framework for PHP, 2012. URL https://github.com/sebastianbergmann/phpunit.

[114] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. In *Proceedings of the IEEE 63-9*, 1975.

[115] F. Randall Farmer and Bryce Glass. *Building Web Reputation Systems - Ratings, Reviews and Karma to Keep Your Community Healthy.* O'Reilly, 2010. ISBN 978-0-596-15979-5.

[116] Cornell university library, 2012. URL http://arxiv.org/.

[117] JTOC - journal table of contents, 2012. URL http://www.journaltocs.hw.ac.uk/.

[118] Open archives initiative, 2012. URL http://www.openarchives.org/.

[119] Online computer library center - the worlds libraries connected. URL http://www.oclc.org.

[120] W3 - resource description framework and structured query language. URL http://www.w3.org/wiki/RdfAndSql.

[121] Fail2ban - logfile scanner, 2012. URL http://www.fail2ban.org.

[122] The OpenPGP Alliance - based on RFC4880, 2007. URL http://www.openpgp.org/.

[123] GNU Privacy Guard, 2012. URL http://www.gnupg.org/.

[124] Marco Balduzzi, Jonas Zaddach, Davide Balzarotti, Engin Kirda, and Sergio Loureiro. A security analysis of amazon's elastic compute cloud service. In Sascha Ossowski and Paola Lecca, editors, *SAC*, pages 1427–1434. ACM, 2012. ISBN 978-1-4503-0857-1. URL http://dblp.uni-trier.de/db/conf/sac/sac2012.html#BalduzziZBKL12.

[125] Cacert. URL http://wiki.cacert.org/Risk/History.

[126] C.Walker Y.Akdenix, N.Taylor. Regulation of investigatory powers act. *State surveillance in the age of information and rights, (2001) Criminal Law Review, (February)*, 2000.