Working in the Cloud

Web-based Version Control System for Task-oriented Group and Individual Projects

Sheng Yu

September 2011

Report on a project submitted in part fulfilment for the degree of

Master of Science in Information Technology

[Department of Computer Science](http://www.cs.york.ac.uk/)

[The University of York](http://www.york.ac.uk/)

Supervisor: Dr. Stefano Pirandola

*Number of words = x,xxx, as counted by the Microsoft Word’s word count command, excluding appendices.*

# Abstract

Version control is one of the most common ways to manage computer based projects. However, due to the software based design and file-oriented mechanism, the existing version control systems are too large and too complicated for lightweight use, as well as difficult to be deployed in the diversified environments, such as computers without administrator account and mobile devices. In this project, a web-based version control system for task-oriented group and individual projects has been developed as supplement to the existing version control systems by following the standard process of web application design and development: requirements analysis, methodologies, design, implementation and evaluation.

Under the concept of cloud computing, the version control system which this project design and developed is a fully web-based system. It does not need of installation of any software at client side, and can be accessed at anywhere, even though on mobile devices or computer without administrator privilege in installing software, just need a web browser and network connection.

To effective assign works to group members and avoid hassle in overlap editing files, the system has been designed a task-oriented mechanism in dividing work of files as tasks for assignment.

# Statement of Ethics

1.1 Contribute to society and human well-being.

1.7 Personal Data Consideration, Privacy: Login require to all functions

2.7 Improve public understanding of computing and its consequences.

3.2

# Acknowledgements

Contents

[1. Introduction 1](#_Toc303384128)

[2. Literature Review 2](#_Toc303384129)

[2.1 Version control 2](#_Toc303384130)

[2.1.1 Existing version control systems 3](#_Toc303384131)

[2.2 Software-based, web-based and cloud 4](#_Toc303384132)

[2.3 Programming languages 5](#_Toc303384133)

[2.4 Interaction design 6](#_Toc303384134)

[3. Requirements Analysis 8](#_Toc303384135)

[3.1 Project goal 8](#_Toc303384136)

[3.1.1 Web-based 8](#_Toc303384137)

[3.1.2 Task-oriented 9](#_Toc303384138)

[3.2 Key users and user needs (usability Investigation) 9](#_Toc303384139)

[3.2.1 Deployment and Portability 9](#_Toc303384140)

[3.2.2 Accessibility 9](#_Toc303384141)

[3.2.3 Privacy and safety of data 9](#_Toc303384142)

[3.2.4 Response time 9](#_Toc303384143)

[3.2.5 Modifiability 9](#_Toc303384144)

[3.2.6 Platform compatibility 9](#_Toc303384145)

[3.2.7 Reporting 9](#_Toc303384146)

[4. Methodologies 10](#_Toc303384147)

[4.1 Platform as a service (PaaS) 10](#_Toc303384148)

[4.2 Waterfall model 10](#_Toc303384149)

[4.3 Prototype 10](#_Toc303384150)

[4.4 Database design concepts 10](#_Toc303384151)

[4.4.1 Entity-relationship model 10](#_Toc303384152)

[4.4.2 Third normal form (3NF) 10](#_Toc303384153)

[5. Design and implementation 11](#_Toc303384154)

[5.1 Function and mechanism 11](#_Toc303384155)

[5.1.1 Task-oriented design 11](#_Toc303384156)

[5.1.2 Task version and relationship 11](#_Toc303384157)

[5.1.3 Directory version and relationship 11](#_Toc303384158)

[5.1.4 File version and storage 11](#_Toc303384159)

[5.1.5 Error handling 11](#_Toc303384160)

[5.1.6 Login and Safety 11](#_Toc303384161)

[5.1.7 Performance optimization 11](#_Toc303384162)

[5.2 Database model 11](#_Toc303384163)

[5.2.1 Entity-relationship modelling 11](#_Toc303384164)

[5.2.2 Attributes property of entities 11](#_Toc303384165)

[5.3 Interface design 11](#_Toc303384166)

[5.3.1 Style 11](#_Toc303384167)

[5.3.2 Page layout 11](#_Toc303384168)

[5.3.3 Accessibility 11](#_Toc303384169)

[5.4 Prototype design 11](#_Toc303384170)

[5.5 Prototype evaluation 11](#_Toc303384171)

[5.6 Prototype re-design 11](#_Toc303384172)

[5.7 Two layer PHP architecture 11](#_Toc303384173)

[5.8 Compatibility 12](#_Toc303384174)

[5.9 CSS classes multiple use 12](#_Toc303384175)

[6. Evaluation 13](#_Toc303384176)

[6.1 Testing of version control 13](#_Toc303384177)

[6.2 Testing of administration 13](#_Toc303384178)

[6.3 Compatibility testing 13](#_Toc303384179)

[7. Future Works 14](#_Toc303384180)

[7.1 Diff storage and analysis 14](#_Toc303384181)

[7.2 Automatic merging 14](#_Toc303384182)

[7.3 Branching support 14](#_Toc303384183)

[7.4 Distributed system design 14](#_Toc303384184)

[7.5 Multi-level administration 14](#_Toc303384185)

[8. Conclusion 15](#_Toc303384186)

[Appendices 21](#_Toc303384187)

[8.1 Set-up guide 21](#_Toc303384188)

[8.1.1 Environment requirement 21](#_Toc303384189)

[8.1.2 Step-by-step guide of installation 21](#_Toc303384190)

[8.2 Questionnaire of requirement analysis 21](#_Toc303384191)

[8.3 Interface samples 21](#_Toc303384192)

[8.4 Core source code 21](#_Toc303384193)

# Index of Figures

# Index of Tables

[Table 1 – Jabok Nielsen’s framework of system acceptability [27] 7](#_Toc303305510)

# Introduction

Motivation

Introduction of this project

Report structure

# Literature Review

This section was about the literatures review related to this project – a web-based version control system. As this project was to develop an application as a web-based system for doing task-oriented version control for group and individual computer-based projects, the literature review could be divided into these parts: version control, existing systems, web and cloud, programming languages and interaction design. These parts were all about the features of the system.

## Version control

Version control, also be called as revision control or source control, was a method of managing files related to development of projects through their whole life cycle [1]. It is essential for multi-developer projects [2]. Lots of popular version control software includes CVS and Subversion run as client-server model, so they support more than one users working especially programming for a same project, because the multi-clients can be connected to a central version control server to be synchronised [3] [4]. The general features provided by version control include storing each commit/version of files or directories, allowing rollback, showing changes history and assist merge/integration [1]. Based on the features, many kinds or parts of project can use version control to be effective managed, such as projects of software development and documentation writing. Each commit/change can be marked with properties of which author did this commit and what time it committed. The commits also comes with a unique version code to identify the times of changes of it. Users can also add some comments to the version when they commit it, so it is very clear to identify what the users changed in there committed version. When a version commit has been identified as worse than an older version, the changes can be easily roll backed to a previous version at any time [1].

Even though the existing version control systems sound wonderful enough, however there still an important point should be noticed, that is the “concurrent access” problem exists in almost all the version control systems [5]. Every project developer can access an entire project, and have privilege in changing any part of the project. It is hard to assign responsibility as tasks to the developers, so the developers may forget where their positions in developing the projects are. When a developer mixed up his/her range of development, and changed some files which someone else is working for, it may leading to serious problem, or waste of time in combining works manually, even though merge policy can do combination of most their works, the result cannot be guaranteed as the most expected one due to the complicacy of different kinds of working [5] [6] [7]. Some version control system, such as Subversion, has designed the “lock” feature to prevent this problem. However, even though a file has been locked by a developer, other developers can also get copy of it from their local update before file locked, or from archive of older versions. It is hard to restrict this problem under the current version control policy [6].

To solve this problem, this MSc project was aimed to develop a version control system which has “task-oriented” feature avoids developers forget their role in development, also prevents the happen of concurrent editing of files. Task-oriented development supports more than one people working with one project at its separated tasks, without interrupt and overlap. Assignment of tasks for the developers can be confirmed by a discussion in a group of each developer’s strong points and weak points, and finish assigning by group leader in the new system. To achieve task assignment, a schedulable feature may be useful. In the system of this project, the tasks could be set up with relationship with other tasks. In 1910, Henry Gantt first published the concept of predecessor in his famous Gantt chart [8]. It resolved the organisation problem of group collaboration in scheduling very well. This project uses the concept from Gantt chart to scheduling tasks. A task could have a predecessor/father task, which means in order to make sure tasks to be worked in sequence, a task can be start only when its predecessor task already finished.

### Existing version control systems

As coordination and work management were important features of the system, some design can be learnt from operating mechanism of existing version control system. In the following, the popular software-based client-server version control system CVS [3] and Subversion [4] will be analysed by their important features and associated with the idea inspired by them.

#### Unique version number

Every content change of directory or file will generate a new version attached to the directory or file itself. Moving, renaming and deleting will also be considered as a change. Each historical version will be kept for in-case use [4].

#### Atomic commits

For coordination considering, every submitting of change set will generate a version of change history, even only a part of the set has been changed [4]. As the system which this project developed was task-based, so it should be different from the existing SVN versioning mechanism (all files will be updated to a latest same version code after an every submitting command). Task-based feature limited area of change set to tasks, therefore, the versioning mechanism could be designed as: every time of file and directory change will increase the version number of file and directory individually, and the task version will be increased when a submit operation has been executed, no matter how many changes of file and directory have been made, even only renamed a file. For example, when commit the change of one file in a four file task, the version code of the one changed file will be increased by 1, other unchanged file will still remain the it old version code, the version code of the entire task contains this file will also be increased by 1, because the new mechanism is designed to able to track change of each file associated with whole task.

#### Locking

To avoid unsynchronised editing, lock-work-submit-unlock is a very good feature provided by SVN for conflicts free editing [4]. In this project, the task as a minimum assignable unit, should be locked when a user start doing the task. When a task starts by a user, a lock would be set to a task, other user except the user who sets the lock cannot request to do any change of the task, even he/she was already been assigned to the task.

## Software-based, web-based and cloud

Lots of popular version control systems are traditional client-server model based [9], even though some version control systems were built with the distributed approach, like Git[[1]](#footnote-1), however, it is very complicated for small group and individual projects to use, due to it is hard to understand and use for non-experience users [10]. To set up a traditional version control system, it requires sets up three parts which will be configured to work together: centralised server runs server side software, client(s) runs client side software and reliable network connections between server and client(s). Server stores every versions/commits of each files, client(s) stores a copy of latest version of project files and allow user working and changing it for future commit [1] [11] [12]. Network connection is a very important component in version control system, because it connects server and client(s) for both of them can be synchronised to latest status and keep files up-to-date. In normal way of doing a project in a version control system, users as clients usually sign in by client side software to the repository of the project in version control server, download (update) latest version of copy of all files related to the project from server to their own computer, and work for the files. The users may have arrangement in advance to avoid possible overlap working in same files, at least in same class of a programming project or same paragraph of a documentation writing project, because the existing merge algorithm in the most version control systems could not combine overlapped work in different commits without conflicts faultlessly. When a user finished a milestone of working in his/her files, he/she needs to re-login by client software into project repository in the server, commits all the working he/she did to the server. After commit, a new version code will be generated. When other users as clients tried to download/update the project, if newer version of files on the server has been detected after compared to the local version, the files in the client side would be updated to the latest version [1] [13].

The helpful features made version control systems work great for managing many kinds of project. However, it needs to install software at both server side and client side to perform version control actions [1] [13]. For entry-level users, it may be hard for them to install and configure client side software to work with server which providing version control service; For users who often working in different kinds of environment instead of their own computer, such as working in computer lab and on mobile devices like iPad, they may have not privilege to install client side software of version control systems, or the mobile device does not support software for version control.

To allow client side user can working at most environment with version control support, it is important to find out a way deliver the service without need of installing software. By the inspiring of more and more popular cloud computing concepts like Google’s Cloud [14], this project was considered to develop a fully web-based version control system which does not require any installation of client side software. Web-based design can overcomes some drawbacks of software-based design, such as hard to use at anywhere and hard to configure by entry-level user. It can be easily accessed at any computer, even a mobile phone, just need a browser and network access to the version control server. When transfer from traditional version control system to web-based system, user can not only benefit from the “access everywhere” feature, but also gain from an important feature – “easy upgrade at cloud with less disruption”. If the system needs to upgrade to latest release, it does not need to ask user to do upgrade of their client side software like traditional way. To upgrade the web-based system, just need to change the server side software, and all users could start using the new system as usual via their own web browsers [14]. The concept of “network”, “cloud” and “web-based” not indicates the system is only Internet-based. However, it can run at a local area network as a “local cloud”, because the Ethernet supports same technology of Internet’s, such as IP based TCP connection and HTTP protocol, which allows web-based system run at local area network similar to run at Internet [15]. To set up the web-based system in a company-wide network, it needs to allocate a computer as server, configure its Apache, PHP and MySQL running environment, and install web-based version control systems in it. After installation, users can direct access the server’s domain name[[2]](#footnote-2) or IP address, even a private IP address[[3]](#footnote-3), on any browsers by devices connected to the same local area network with the server. Due to the centralised structure and network connection dependency of traditional version control systems, the cost of migration from traditional systems to the new web-based system might be very low – just need do configuration on server side with new web-based system, and tells users to access the new system via their browsers, even mobile browsers, and then continue their working.

## Programming languages

In order to develop the system to be used at web, there are several programming languages available, such as Java/JSP[[4]](#footnote-4), ASP[[5]](#footnote-5), ASP.net[[6]](#footnote-6), CGI[[7]](#footnote-7) and PHP. In these languages, Java as a popular programming language in object-oriented software development can also provide web service by working with JSP on Servlet[[8]](#footnote-8) [16], even though it provides the most object-oriented structure for programming, however, it is hard to set up server side environment and also hard to programming in the scale of this kind of project, and its structure is too complicated for lightweight development, because of its full object-oriented design and great number of components requirement for running [17]; ASP, an Microsoft’s outdated web application engine, which is very popular at the era of Microsoft Windows NT 4.0 and Windows 2000, with many down sides such as high cost, slower speed, lack of library support, low safety design, no debugging support and hard to do migration to other platforms [18]; ASP.net is the latest Microsoft’s web application engine, overcomes many drawbacks existed at ASP, however it still not an open source platform like before, so it still hard to do migration in the future, and high cost in setting up [18]; finally, PHP, is the most shining web programming language and platform with lots of great features, such as fully open-source, object-oriented support, abundant built-in library functions and rich high quality open source resources. It is totally free [19], but still has great features with commercial programming language and platforms. The running environment of PHP is also easy to be set up. A number of pre-configured server kits are available to do one-click installation of PHP running environment [20] [21] [22]. It can run with open source web server software such as Apache[[9]](#footnote-9) and Nginx[[10]](#footnote-10), also Microsoft’s IIS[[11]](#footnote-11) series. Some opponent would say PHP is too simple so it can only be used in developing lightweight application, however, Facebook as one of the largest websites, uses PHP as its main programming language and also did great contributions to make PHP better [23]. Overall, PHP has been chosen as the programming language for development of this project.

As a great partner of PHP in LAMP[[12]](#footnote-12) group, MySQL[[13]](#footnote-13) has been chosen as the database system for development of this project, because it provides great number of features in small installation size with easy configuration. MySQL is open-source and free, also have lots of useful features, such as “view” and “lock”, it was a great choice for different kinds of use and easy to be customized [24].

## Interaction design

Interaction design is “the practice of designing interactive digital products, environments, systems, and services.” [25] It defines and designs the behaviours of human in perceiving and using digital objects [26]. All the researches of interaction design are focus on its objective: to make product easy to be used; to technology happy to be experienced. In order to achieve the objective, there are several things to be finished in sequence: the expectation of different kind of user should be classified for analysis; understanding what behaviours of user would do or already did during their using an interface; understanding their psychological characteristics of using the interface. The relationship between product and user would be built by the interaction design of interface, to help user to do what he/she need via the interface effective and relaxed [26]. In measuring the usability of a product and its interaction design result, Jabok Nielsen[[14]](#footnote-14) has written a framework of system acceptability [27]:

|  |  |
| --- | --- |
| **Learnability** | How easy is it for users to accomplish basic tasks the first time they encounter the design? |
| **Efficiency** | Once users have learned the design, how quickly can they perform tasks? |
| **Memorability** | When users return to the design after a period of not using it, how easily can they are establish proficiency? |
| **Errors** | How many errors do users make, how severe are these errors, and how easily can they recover from the errors? |
| **Satisfaction** | How pleasant is it to use the design? |

Table 1 – Jabok Nielsen’s framework of system acceptability [27]

The five points from Jabok Nielsen describes the measuring standard of usability, also told product designer especially interface designer make sure their works meets the five points to enable good user experience.

In this project, as the aim of the web application is to provide an interface for user easy to discover functions and feels relax in using, interaction design is an important part of this project, because users may have less experience in using the system. Users may not clear how the mechanism work in the system, however this project should make sure them can feel free in using the system with less barriers.

# Requirements Analysis

In the requirement analysis section, it has been investigated in areas about design requirements of this project.

## Project goal

As described above, the goal of this project was about designs and develops a version control system with task-oriented feature for groups and individuals to use it in aiming their computer-based projects via web interface. In comparing with the existing version control systems, the most two important different features of the system of this project are web-based and task-oriented.

### Web-based

The first features, web-based, is influenced by the more and more popular cloud computing concepts. In this era full of cloud computing, computer is becoming a tool of using cloud services. The widely use of internet provides many favourable factors for collaboration as group [28]. For example, in Google’s design of cloud computing, user just need a device with web browser to use nearly all the Google services, no matter what devices you have and where you are [14]. Google Apps is an interest product should be noted. It is a fully web-based enterprise office solution, which contains enough tools for companies to word with it more flexible than using local software. User do not need to do installation of any programme, and do not need to waste time in considering software compatibility between different hardware and software combination. To run an online web-based programme, users just need a mainstream web browser, even though on a tablet or mobile phone, without requirement of hardware or software configuration [14]. In Google Apps, users could work together without sending works anywhere. They just need to create or upload a previous work into Google Docs, and ask the group members to sign in with their own accounts, after that, all the group members can see the entire work and their can change or add new things, or comment of some paragraphs. When a user did his/her changing of the work, the old version would be archived as “version control”, in case roll back to old versions if the change is harmful. Users could working together at their own computers by sign in to a same system without need of sending their work to each other, as well as prevent the confusion of versions of works [29]. Another product should be noted is Google Chrome OS. It is the result of a pure web-based cloud computing concept. The Chrome OS only have a web browser, and the applications running in the OS are all web-based. The netbook loaded with Chrome OS do not need powerful processing capacity, just few basic processing power of web content is enough, because the core of web-based cloud application is running in the cloud side or server side, the user side or client side computer just in role of helping user in interacting with user interface of the application [30]. By experience of the plentiful advantages of cloud computing, as a trend of moving from traditional software-based application to web-based application, this project is aiming in designing and developing an easy to use version control system in providing a fully web-based interface for users can be used at anywhere.

### Task-oriented

In the traditional version control systems, flies are base units to be version controlled by project. When commit changes, all files changed in once would be counted as one version of a project [3]. In the common group working for a computer based project, they may have several members working together. In most cases of computer based project, members in a group are working their works paralleled, and combined/merged at the end when finished working. During the working, they are communicating with each other, and then continue doing their work in files. To improve efficiency of a computer based group work, a better way is to clearly define and design tasks within a project and assign them to group members by analyses points their good at or not, because distinct task assignment can maximum uses group psychosocial traits, as well as give pressure for member for motivation by compare progress and quality of tasks [31]. In the existing file based version control mechanism, group members working in a same project may usually confuse in identify duty of files [7]. To make the “task” can be a part of the version control system, in this project, the system was focus on build a task based mechanism in order to tracking two level of versions in a project: task level versions, directory and file level versions, for tracking directory and file within task assignment.

## Key users and user needs (usability Investigation)

As an application, user is one of the most important things which this project should be focused. As defined in the project title, this system is for computer based group and personal projects with version control. According to my observations, people who doing documentation writing or programming may focus on version control in aiming their work of keep working safe or make collaboration easy.

### Deployment and Portability

For some kind of users, they may be used to working in computer lab or with public computers which have not administrator privilege. In using that kind of computer for work, a big problem is the user does not have any right in installing software. If the user is also rely on using version control system in helping his/her work, he/she may crazy in the truth of no version control client software available for use.

In order to help this kind of user can use the version control system as usual, the deployment of the system especially at client side may not require administrator privilege. At the same time, if the client side software not support in running at different kind of operating systems, it may limits the usage of the system, so it should support different kinds of operating systems.

For people who like to use version control system on a mobile device, such as on netbook, tablet or mobile phone, they may wish to get the similar experience or familiar interface of when they use it on a regular desktop PC. If user can access the system at any platform with same or similar feeling and process, the system would get better result in Jabok Nielsen’s framework of system acceptability [27], which means the system have better learnability for users to found how to use the systems when they first time transfer to a new platform and better efficiency for they to start perform tasks quickly. So the system should have a universal design of interface for different kind of systems.

### Accessibility

In order to serve as much people as the system can, it is very important to give blind user convenience in using their screen reading software to use the system. The W3C has defined standards for better accessibility, which is the “Web Content Accessibility Guidelines” (WCAG). This document has some guidance of to make web page much easier for blind or amblyopia user to use.

### Privacy and safety of data

### Response time

将数据库读进数组 在数组里本地处理

### Modifiability

### Platform compatibility

### Reporting

# Methodologies

## Platform as a service (PaaS)

## Waterfall model

## Prototype

## Database design concepts

### Entity-relationship model

### Third normal form (3NF)

The third normal form is normal form in database normalization, which defined by Edgar F. Codd in 1971 [32] [33]. All the database tables in 3NF should meet the conditions: the table is in 2NF[[15]](#footnote-15) and its attributes except primary key should not dependent on other attribute [32]. Under 3NF, database tables could minimal their redundancy and make the SQL statements could be connected together for multi-table operation much easier [34]. This means if a table in database meets requirement of 2NF, but not meets 3NF, it need to split into several tables. Redundancy attribute, which is the attributes dependent on other attribute, must to move out to a new table to keep avoid mistake/exception in operations (insert, update and delete) may lead by redundancy, also help tables can be isolated and uses SQL statements to be connected by their primary key and foreign key [35].

# Design and implementation

## Function and mechanism

### Task-oriented design

In comparing to the traditional file-based version control unit

The directory and file level is about keep history/versions of all the operations

### Task version and relationship

This project has been titled as “task-oriented”

### Directory version and relationship

### File version and storage

### Error handling

### Login and Safety

### Performance optimization

## Database model

### Entity-relationship modelling

### Attributes property of entities

## Interface design

### Style

## Accessibility

WCAG 1.0

H

P

Table with Thead

Image with alt

Less javascript use

White background high contact

Use OL UL DL LI for list

Title attribute clear

比如按钮 用户需要反馈 颜色

### Page layout

### Accessibility

## Prototype design

## Prototype evaluation

## Prototype re-design

## Two layer PHP architecture

In the design of this system, the PHP files have been divided into two parts: function libraries and controllers. In controller part, the block of PHP code will be placed in the head of each PHP file, before output of HTML code. This architecture

## Compatibility

## CSS classes multiple use

To avoid the CSS file too large and can be load quickly without redundancy, in the system design, the use of CSS has been designed as fully multiple use mode. The CSS multiple use is a lesser known way to reduce CSS file size effectively, which means more than one CSS classes could be referenced in one HTML element at same time by split the class names by a space [36]. For example.

# Evaluation

## Testing of version control

## Testing of administration

## Compatibility testing

# Future Works

## Diff storage and analysis

[37]

## Automatic merging

## Branching support

## Distributed system design

## Multi-level administration

# Conclusion

References

|  |  |
| --- | --- |
| [1] | B. Collins-Sussman, F. W. Brian and C. M. Pilato, Version Control with Subversion, O'Reilly, 2004. |
| [2] | B. Danella, “Rapid Subversion adoption validates enterprise readiness and challenges traditional software configuration management leaders,” 15 May 2007. [Online]. Available: http://www.open.collab.net/news/press/2007/svn\_momentum.html. [Accessed 10 July 2011]. |
| [3] | D. Price, “CVS v1.11.23 Manual,” Ximbiot LLC, 8 May 2008. [Online]. Available: http://ximbiot.com/cvs/manual/cvs-1.11.23/cvs.html. [Accessed 1 July 2011]. |
| [4] | The Apache Software Foundation, “Apache Subversion Features,” [Online]. Available: http://subversion.apache.org/features.html. [Accessed 1 July 2011]. |
| [5] | T. Oakden, “None Concurrent Access in Version Control,” 12 Oct 2009. [Online]. Available: http://forum.unity3d.com/threads/36536-None-concurrent-access-in-version-control. [Accessed 22 July 2011]. |
| [6] | P. Roy, “Understanding Subversion's Problems,” 9 Mar 2011. [Online]. Available: http://ventspace.wordpress.com/2011/03/09/understanding-subversions-problems/. [Accessed 22 July 2011]. |
| [7] | Jbcrouigneau, “Task Oriented Development and Validation Space,” 24 Apr 2009. [Online]. Available: http://www.svnforum.org/threads/36840-Task-oriented-development-and-Validation-space?s=cf3b028492de3003320a35e609f4777b. [Accessed 22 July 2011]. |
| [8] | H. Gantt, Work, Wages and Profit, New York: The Engineering Magazine, 1910. |
| [9] | Shawn, “Which is More Popular (Currently, by Recent Install Base) SVN or CVS?,” 23 Apr 2009. [Online]. Available: http://stackoverflow.com/questions/782375/which-is-more-popular-currently-by-recent-install-base-svn-or-cvs. [Accessed 23 July 2011]. |
| [10] | C. Duan, “Understanding Git Conceptually,” 17 Apr 2010. [Online]. Available: http://www.eecs.harvard.edu/~cduan/technical/git/. [Accessed 23 July 2011]. |
| [11] | T. Spencer, “Setup a Subversion Server in 4 Minutes,” 2 Mar 2007. [Online]. Available: http://www.tonyspencer.com/2007/03/02/setup-a-subversion-server-in-4-minutes/. [Accessed 23 July 2011]. |
| [12] | “Setting Up Subversion,” July 2006. [Online]. Available: http://systhread.net/texts/200607subver.php. [Accessed 23 July 2011]. |
| [13] | D. Thomas and A. Hunt, Pragmatic Version Control Using CVS, Pragmatic Bookshelf, 2003. |
| [14] | Google Inc., “Top ten advantages of Google's cloud,” 2011. [Online]. Available: http://www.google.com/apps/intl/en/business/cloud.html. [Accessed 24 July 2011]. |
| [15] | L. L. Peterson and S. B. Davie, Computer Networks : A Systems Approach, Amsterdam; London: Morgan Kaufmann, 2007. |
| [16] | Oracle, “Java Servlet Technology Overview,” [Online]. Available: http://www.oracle.com/technetwork/java/overview-137084.html. [Accessed 10 July 2011]. |
| [17] | M. Hall, “Building Web Apps in Java: Beginning & Intermediate Servlet & JSP Tutorials,” 2011. [Online]. Available: http://courses.coreservlets.com/Course-Materials/csajsp2.html. [Accessed 10 July 2011]. |
| [18] | TechyShell.com, “ASP – Its Advantages and Disadvantages,” 27 May 2009. [Online]. Available: http://www.techyshell.com/internet/asp-its-advantages-and-disadvantages/. [Accessed 13 July 2011]. |
| [19] | The PHP Group, “The PHP License, version 3.01,” 2010. [Online]. Available: http://www.php.net/license/3\_01.txt. [Accessed 11 July 2011]. |
| [20] | Z:WAMP Group, “Z:WAMP Server Pack,” 7 Nov 2010. [Online]. Available: http://zwamp.sourceforge.net/. [Accessed 21 July 2011]. |
| [21] | R. Bourdon, “WampServer,” 24 Dec 2010. [Online]. Available: http://www.wampserver.com/en/. [Accessed 21 July 2011]. |
| [22] | E. Group, “EasyPHP,” 2011. [Online]. Available: http://www.easyphp.org/introduction.php. [Accessed 12 July 2011]. |
| [23] | B. Shire, “PHP and Facebook,” 3 May 2007. [Online]. Available: http://www.facebook.com/blog.php?post=2356432130. [Accessed 10 7 2011]. |
| [24] | Daniel, “Benefits Of MySQL,” 20 Nov 2010. [Online]. Available: http://benefitof.net/benefits-of-mysql/. [Accessed 13 July 2011]. |
| [25] | A. Cooper, R. Reimann and D. Cronin, About Face 3: The Essentials of Interaction Design, Wiley, 2007. |
| [26] | H. Sharp, Y. Rogers and J. Preece, Interaction Design : Beyond Human-computer Interaction, Chichester: Wiley, 2007. |
| [27] | J. Nielsen, “Usability 101: Introduction to Usability,” [Online]. Available: http://www.useit.com/alertbox/20030825.html. [Accessed 24 7 2011]. |
| [28] | M. Miller, Cloud computing: Web-based applications that change the way you work and collaborate online, Que, 2008. |
| [29] | Google Inc., “Boost productivity with Google-powered collaboration apps,” 2011. [Online]. Available: http://www.google.com/apps/intl/en/business/collaboration.html. [Accessed 27 July 2011]. |
| [30] | S. Pichai, “Introducing the Google Chrome OS,” 7 July 2009. [Online]. Available: http://googleblog.blogspot.com/2009/07/introducing-google-chrome-os.html. [Accessed 27 July 2011]. |
| [31] | S. G. Cohen and D. E. Bailey, “What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite,” *Journal of Management,* vol. 23, no. 3, pp. 239-290, 1977. |
| [32] | E. F. Codd, “Further Normalization of the Data Base Relational Model,” IBM Research Report, New York City, 1971. |
| [33] | R. J. Rustin, Data Base Systems: Courant Computer Science Symposia Series 6, New York City: Prentice-Hall, 1972. |
| [34] | E. F. Codd, The Relational Model for Database Management, Addison-Wesley, 1990. |
| [35] | C. Zaniolo, “A New Normal Form for the Design of Relational Database Schemata,” *ACM Transactions on Database Systems,* vol. 3, no. 7, 1982. |
| [36] | J. Kyrnin, “Use Multiple CSS Classes on a Single Element,” 2011. [Online]. Available: http://webdesign.about.com/od/css/qt/tipcssmulticlas.htm. [Accessed 26 July 2011]. |
| [37] | Oracle, “JavaServer Pages Technology,” [Online]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. [Accessed 2 July 2011]. |
| [38] | S. Chacon, “Git - The Fast Version Control System,” 2011. [Online]. Available: http://git-scm.com/. [Accessed 23 July 2011]. |
| [39] | J. Loeliger, Version Control with Git, O'Reilly, 2009. |
| [40] | R. M. Hinden and B. Haberman, “RFC 4193 : Unique Local IPv6 Unicast Addresses,” Feb 2005. [Online]. Available: http://tools.ietf.org/html/rfc4193. [Accessed 24 July 2011]. |
| [41] | Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. d. Groot and E. Lear, “RFC 1918 : Address Allocation for Private Internets,” Feb 1996. [Online]. Available: http://tools.ietf.org/html/rfc1918. [Accessed 24 July 2011]. |
| [42] | P. Mockapetris, “The Domain Name System,” in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, Nottingham, 1984. |
| [43] | P. Mockapetris, J. Postel and P. Kirton, “Name Server Design for Distributed Systems,” in *Proceedings of the Seventh International Conference on Computer Communication*, Sidney, 1984. |
| [44] | P. Mockapetris, “RFC 882 : Domain Names - Concepts and Facilities,” Nov 1983. [Online]. Available: http://tools.ietf.org/html/rfc882. [Accessed 24 July 2011]. |
| [45] | P. Mockapetris, “RFC 883 : Domain Names - Implementation and Specification,” Nov 1983. [Online]. Available: http://tools.ietf.org/html/rfc883. [Accessed 24 July 2011]. |
| [46] | D. Parrack, “New Twitter.com UI is faster, better,” 15 Sept 2010. [Online]. Available: http://tech.blorge.com/Structure:%20/2010/09/15/new-twitter-com-ui-is-faster-better/. [Accessed 25 July 2011]. |
| [47] | Microsoft Corporation, “ASP.NET Web Pages,” 2011. [Online]. Available: http://www.asp.net/web-pages. [Accessed 2 July 2011]. |
| [48] | A. Sami, “What is New in ASP.NET 4.0, Visual Studio 2010 IDE,” 13 Jan 2010. [Online]. Available: http://www.codeproject.com/KB/aspnet/Whatis\_New\_ASP\_Net\_4.aspx. [Accessed 2 July 2011]. |
| [49] | C. D. Knuckles and D. S. Yuen, Web Applications: Concepts & Real World Design, Hoboken, N.J.: John Wiley & Sons, Inc., 2005. |
| [50] | D. Robinson and A. L. K. Coar, “RFC 3875: The Common Gateway Interface (CGI) Version 1.1,” Oct 2004. [Online]. Available: http://tools.ietf.org/html/rfc3875. [Accessed 10 7 2011]. |
| [51] | The Apache Software Foundation, “Apache HTTP Server Project,” 2011. [Online]. Available: http://httpd.apache.org/. [Accessed 13 July 2011]. |
| [52] | Igor Sysoev, “Nginx,” [Online]. Available: http://nginx.org/en/. [Accessed 13 July 2011]. |
| [53] | Microsoft Corporation, “IIS: Overview,” 2011. [Online]. Available: http://www.iis.net/overview. [Accessed 13 July 2011]. |
| [54] | J. Lee and B. Ware, Open source Web development with LAMP using Linux, Apache, MySQL, Perl, and PHP, Boston: Addison-Wesley, 2003. |
| [55] | Oracle Corporation, “MySQL Standard Edition,” 2010. [Online]. Available: http://www.mysql.com/products/standard/. [Accessed 13 July 2011]. |
| [56] | TutorialsPoint.COM, “Database - Second Normal Form (2NF),” 2011. [Online]. Available: http://www.tutorialspoint.com/sql/second-normal-form.htm. [Accessed 22 July 2011]. |
| [57] | F. Spillers, “How Usable is Jakob Nielsen?,” 7 Apr 2004. [Online]. Available: http://experiencedynamics.blogs.com/site\_search\_usability/2004/04/how\_usable\_is\_j.html. [Accessed 21 July 2011]. |

# Appendices

## Set-up guide

### Environment requirement

### Step-by-step guide of installation

## Questionnaire of requirement analysis

## Interface samples

## Core source code

1. Git is a distributed version control system developed by father of Linux – Linus Torvalds, used for manages development of Linux kernel originally, now for some large scale projects [38] [39]. [↑](#footnote-ref-1)
2. Domain name, an easy to remember name identifies a computer, which have mapping relation (records) to IP addresses, allows user to access various service on a server by only use its domain name, such as http and ftp (A record), mail service (mx record), etc. [44] [45]. [↑](#footnote-ref-2)
3. Private IP address, is IP address in range of pre-reserved network address space. It usually use in local area networks, which can only be accessed by computer in the same network [40] [41]. [↑](#footnote-ref-3)
4. JSP, JavaServer Pages, a technology uses Java language creates dynamic web content [37]. [↑](#footnote-ref-4)
5. ASP, Active Server Pages, is a server-side script engine from Microsoft for dynamically web pages, uses VBScript and JavaScript as server-side programming language [49]. [↑](#footnote-ref-5)
6. ASP.net, is Microsoft’s second generation server-side script engine, uses Microsoft’s .Net Framework as libraries, and also uses object-oriented programming languages such as VB.net and C# [47] [48]. [↑](#footnote-ref-6)
7. CGI, Common Gateway Interface, is a platform-free interface for client to execute application on web server [50]. [↑](#footnote-ref-7)
8. Servlet, is a server-side Java application, generates web pages as interlayer between client request and server response with platform-free and protocol-free features [16]. [↑](#footnote-ref-8)
9. Apache, a common name of Apache HTTP Server, is an open-source web server application which has been used most widely in the world, and can be used in lots of operating systems [51]. [↑](#footnote-ref-9)
10. Nginx (Engine X) is an emerging high performance open source HTTP and proxy server [52]. [↑](#footnote-ref-10)
11. IIS, Internet Information Services, is Microsoft’s Windows-based internet server application [53]. [↑](#footnote-ref-11)
12. LAMP, a powerful bundle of open-source software working together as a web server, includes Linux (operating system), Apache (HTTP server), MySQL (database) and PHP (script language) [54]. [↑](#footnote-ref-12)
13. MySQL, an open-source database system, developed by MySQL AB, now is a part of Oracle [55]. [↑](#footnote-ref-13)
14. Jabok Nielsen is one of the most famous usability consultants in interaction design [57]. [↑](#footnote-ref-14)
15. 2NF, the second normal form, states all the records in database table should be unique identified [33]. The common way of making table meets requirement of 2NF is adding a unique ID to each record as its primary key [56]. [↑](#footnote-ref-15)