ONLINE FILE SHARING

by

PALANIAPPAN RAMANATHAN

B.E., Annamalai University, India, 2004

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2006

Approved by:

Major Professor
Daniel Andresen, Ph.D.

ABSTRACT

File sharing is one of the oldest applications of the internet. One way of sharing files online is for a user to upload files to a common space on the web and others users can download the files from the common web space.

The objective of this project was to design an online file sharing website where users can upload files and other users can download them. To attain this objective an AJAX enabled interactive user interface involving features like versioning control, RSS syndication and extensive search capabilities was developed. To make the website more user friendly, users were given two space-constrained visualizations of their file system to view space occupied by the files and folders, and three AJAX based file management system that works like browsing files on a desktop computer with drag and drop, context menu functionalities etc.

This report discusses the implementation details of the website, and the advantages of having different visualizations of the file system. This report also addresses one frequently asked question regarding file storage; where to store the files, in database as BLOBs or as files in the file system on web server? This report analyzes the time needed to upload, download and search the files stored in both places and discusses the advantages and disadvantages of both techniques in terms of performance, security, integrity, maintenance and code complexity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my Major Professor Dr. Daniel Andresen for guiding me throughout this project. I would also like to thank my other committee members Dr. William J. Hankley and Dr. Mitchell L. Neilsen for helping me in completing this project report

Chapter 1 - Introduction

1.1 Problem

Online File Sharing is practice of sharing files among different users across the internet. Common forms of file sharing are FTP (File Transfer Protocol) model and P2P (Peer-to-Peer) file sharing network. Another common form of sharing files over the internet is for a user to upload files to a website and allow other users to download them from the website. There are a lot of issues to consider when developing such a website.

Users of an online file sharing website who use features like upload, download, share, search etc would want a website that is very interactive and fast and not annoying with a lot of post backs and flashing screens. Another issue is the visualization of their file system where usually users have a limit to upload files. The normal web based file-folder view would be good, but if there are other types of visualizations it would be great. Another important issue to consider is the location where the website stores the uploaded files. Two places where one can store the uploaded files are Database and Server.

1.2 Objective

There were three main objectives in this project. First objective was to build an AJAX enabled online file sharing website which not only reduces the annoying postbacks and loss of control focus, but also gives a faster and more interactive user interface. Moreover to make the website more feature rich, features like RSS syndication, extensive searching (inside documents uploaded), group option to share a file, versioning control to

get back deleted or archived files, organization of the files using folders were added to the website.

Second objective was to give the users different visualizations of their file system. Usually in a file sharing website, users will be given only one option where they can view their files and folders in the traditional windows style folder view i.e. where they have the option to sort their files and folders based on size, type, and time uploaded etc, and navigate through their file system by clicking on the folders. In this website, users were given different visualizations of their file system i.e. one traditional windows style folder view with postbacks as seen in other similar websites, three AJAX based windows style folder view with no postbacks and additional functionalities like right click menus, drag and drop functionalities, and two space-constrained hierarchical visualizations of their file system with which users can know how their files and folders occupy their allotted space.

Third objective was to analyze the issue of file storage. Two common places where files can be stored are database and the web server. In the first option, files can be stored as BLOBs (*B*inary *L*arge *OB*jects) which is the place for storing huge files in the database. Second option is to store the file in the file system on the web server and to store a pointer to the file location in the database. This report analyzes both options and discusses the advantages and disadvantages of both techniques.

1.3 Document Overview

The rest of this documentation first discusses related work in Chapter 2, and then describes the implementation details of the website in Chapter 3. Chapter 4 describes the different visualizations designed and the advantages of having such visualizations.

Chapter 5 presents the results obtained when storing files in database and storing files in the file system, and discusses the pros and cons of both techniques by analyzing them. Chapter 6 presents the conclusions and describes future work.

Chapter 2 - Related Work

2.1 Online File Sharing

There are a lot of file sharing websites online. Some famous sites are www.rapidshare.com, www.megaupload.com, www.xdrive.com, www.box.net etc. www.box.net is an AJAX enabled website with a lot of cool features and is also very interactive. All the websites which serves the purpose of online file storage/sharing usually have a size limit to upload files and some have size limit to download files per hour due to space and bandwidth constraints.

Other forms of file sharing as described in the previous chapter are FTP and P2P. FTP or file transfer protocol is a commonly used protocol for exchanging files over any network that supports the TCP/IP protocol (such as the Internet or an intranet) [15]. The FTP server, running FTP server software, listens on the network for connection requests from other computers. The client computer, running FTP client software, initiates a connection to the server. Once connected, the client can do a number of file manipulation operations such as uploading files to the server, download files from the server, rename or delete files on the server and so on [16]. Most of the browsers present now can act as a FTP client. Common FTP client software's are CuteFTP, SmartFTP and DirectFTP etc. FTP is a common standard for file sharing and is used by a lot of people today.

P2P or Peer-to-Peer network is a type of network in which each workstation has equivalent capabilities and responsibilities. P2P file sharing network is usually used for

4

sharing content files containing audio, video, data or anything in digital format and real-time data. BitTorrent is a famous peer-to-peer file distribution client application. P2P is best known for sharing files online and is more popular than the others methods available.

2.2 File System Views

This section of the document discusses about the research that has been done and the tools that are available for visualizing the file system.

2.2.1 Space-constrained hierarchical visualization

Several types of visualization are out there for visualizing a file system. In a file sharing website where each user has a size limit, a hierarchical visualization would be a very useful visualization compared to other visualizations, since the user would be able to see how his/her files have occupied his/her allotted space. There are lots of hierarchical visualization tools available for visualizing file system in Linux operating systems which can be used as a base model for visualizing file system in a website. The most commonly used hierarchical visualizations techniques are:

2.2.1.1 Space-filling treemaps

The idea of treemaps is to visualize a tree by dividing a rectangle into smaller rectangular objects, one rectangle for each node in the hierarchy. These rectangles have size proportional to some node property (usually file size, if the tree is a file system). [9]. Figure 2.1 shows a simple treemap. fsv (http://fsv.sourceforge.net) is a file system visualizer that's uses the Treemap visualization technique. StepTree is a visualization tool which is also a three dimensional extension of space filling Treemap concept.
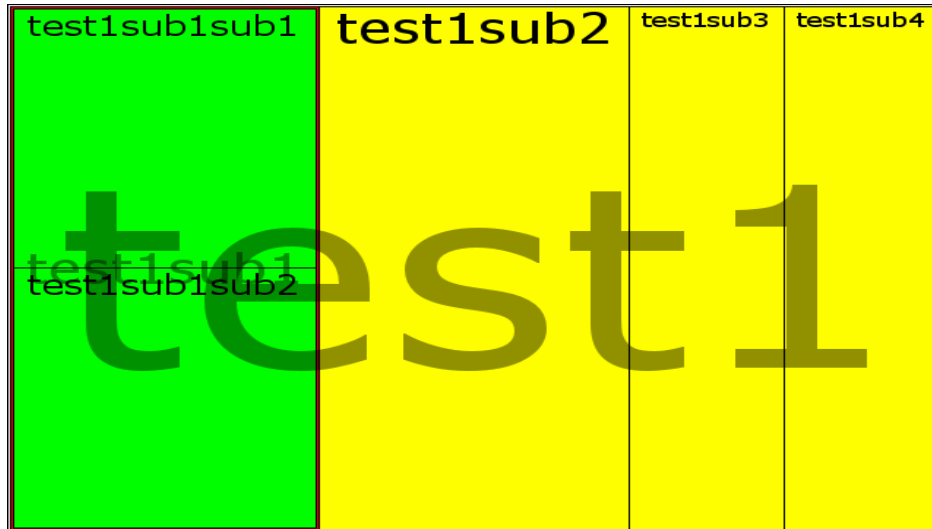
Figure 2.1 Treemap example

2.2.1.2 Squarified treemaps

Squarified treemaps are an extension to the concept of treemaps in which the rectangles are made to look like squares as much as possible [7]. An example of a Squarified treemap can be seen in Figure 2.2. Squarified treemaps are the most popular form of space constrained visualization. Chapter 4 discusses Treemaps in detail.
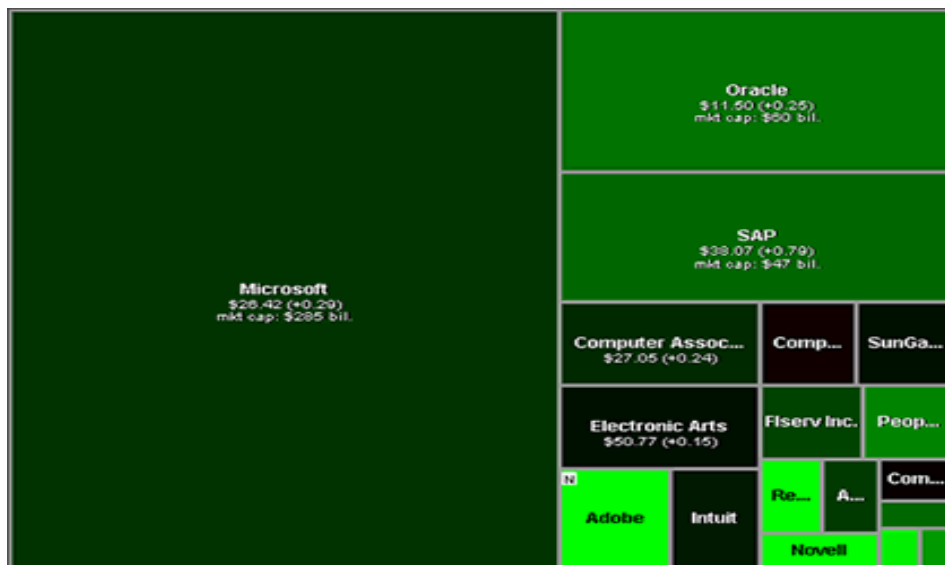
Figure 2.2 Squarified treemap example

2.2.1.3 Cone trees

Cone trees are basically an extension of the normal two-dimensional trees we are used too [6]. The difference is that instead of placing all child nodes along a horizontal line, they are placed on a horizontal circle below the root node.
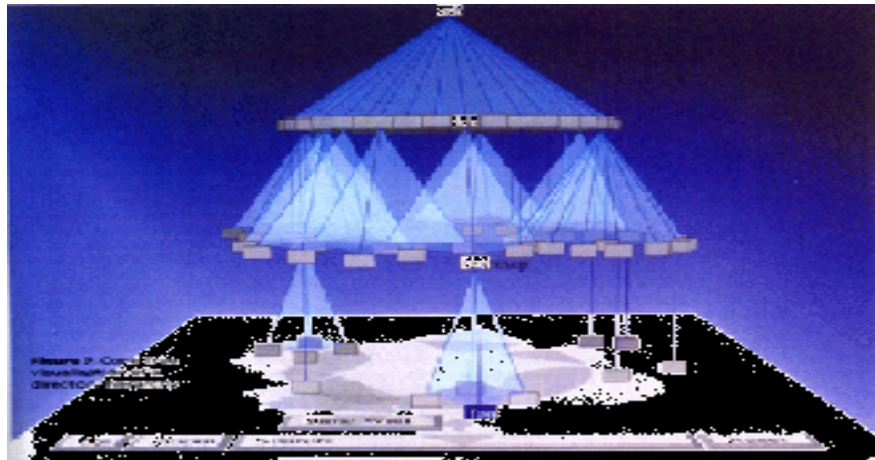
Figure 2.3 Cone tree example

2.2.1.4 Information Cube

In "The Information Cube", every node in a hierarchy is visualized as a semi-transparent cube. The contents of a node are shown as cubes within its cube. What you see if you look at this visualization is lots of cubes within cubes within cubes, and so on [6]. An example of an information cube can be seen in Figure 2.4.

Most of the hierarchical file system visualizers (like fsv, StepTree, XCruiser, tdfsb, etc) available are for Linux/Mac systems. So using this concept of hierarchical file system visualization in a website can be complicated, especially when you consider techniques like cone trees and information cubes which are 3D visualizations. 3D file system visualization in a website where data is being retrieved from a database and using

7

technologies like ASP.NET would be very complicated. It could be done using tools like

Flash. But visualizations based on treemaps or Squarified treemaps can be done using the

technologies and tools that have been used in this project.

Figure 2.4 Information Cube

2.2.2 Traditional file system visualization

Traditional file system visualization is the view in which users get the Microsoft

windows styled folders and files, and they navigate through folders into the child folders

by clicking on a folder and so on. Most of the present websites which visualize a file

system use this method. Until the sudden growth of AJAX, this model was very non-

interactive with its postback for every operation since it had to go to the database where

the details about a particular folder or file were stored. In these past few years, after

AJAX became an important web development technique, all these drawbacks can be

overcome. With these latest technologies, web sites have developed an user interface with

which user can browse his/her file system in the web just like browsing a file system on a

PC by dragging and dropping files into folders etc. In fact it could be made more

interactive and attractive like a file system view on an Apple computer with the JavaScript libraries available today like script.aculo.us and Rico etc.

2.3 Database Vs File System

This question has been asked by a lot of web developers and programmers in the past i.e. where should files be stored – in database as BLOBs or in the file system. Another frequently asked question is where should images that are used in the website stored – in database as BLOBs or in the file system as .gif or .jpeg etc. Answers to both questions are different because images used in a website are usually small in size and are retrieved every time a page is loaded, whereas in a files upload (say an online file sharing/storage) website, files are usually big and are retrieved only when someone requests the file.

People have proposed different solutions to these questions, but no solution has been confirmed as the best solution. The answer usually depends on the requirements of the website.

Chapter 3 - Implementation

This chapter discusses the implementation details of the website developed. It discusses about the latest technologies and tools that were used, the system architecture, database design, functionalities and features available.

3.1 Technologies

Latest technologies and tools were used in developing the website. With these tools and technologies, complex coding can be made very simple.

3.1.1 ASP.NET 2.0 / Microsoft Visual Studio 2005

ASP.NET 2.0 is a technology for building powerful, dynamic Web applications and is part of the .NET framework 2.0. The Microsoft .NET Framework 2.0 is a platform for building, deploying, and running Web Services and applications. ASP.NET 2.0 makes web programming dramatically easier compared to technologies like ASP, JSP, and PHP etc. In fact building web applications with ASP.NET 2.0 is very easier than building them with ASP.NET 1.1.

Web applications can be developed with ASP.NET 2.0 using Visual Studio 2005 which is an advanced integrated development environment developed by Microsoft for building applications that run on Microsoft Windows and the World Wide Web. It has got some great features which were missing in the previous version like better intellisense, configuration of datasets with a few clicks etc.

3.1.2 Microsoft SQL Server 2005

Microsoft SQL Server is a relational database management system from Microsoft. There are lot a features in .NET that are only available for SQL Server like *SqlCacheDependency*, membership, role and session state providers etc. When the project was started, initial plan was to use SQL Server 2005 Express, but it was upgraded to SQL Server 2005 in order to utilize the full-text indexing which is used for searching through BLOBs.

3.1.3 AJAX / ATLAS & Web Services

AJAX is the short form of Asynchronous JavaScript and XML which is a Web development technique for creating interactive web applications. The main concept of AJAX is to process one part of a web page behind the scene with disturbing the other parts of the website.

Atlas is the new Web development technology from Microsoft which integrates client script libraries with the ASP.NET 2.0 server-based development framework. It is the Microsoft's version of AJAX for ASP.NET 2.0. Since the objective of the project was to build an AJAX enabled website, AJAX was used in most of the pages. To decrease the complexity of coding, Atlas was used to send asynchronous calls (using *Atlas Script Manager*) to web services which interacted with the database. Other than sending asynchronous calls to the server, the Atlas control extenders were used to make the website more rich and interactive.

3.1.4 XML / XSLT / XPATH

XML (eXtensible Markup Language) was used as the format for transferring data between the server and client using AJAX. XML returned by the web service was transformed into HTML using XSLT (eXtensible Stylesheet Language Transformation). As discussed in the Features section of this chapter and in next chapter, XML was transformed into HTML with JavaScript incorporated into it using XSLT. Also XML is the preferred format when using AJAX to transfer data between server and client.

3.1.5 JavaScript / Prototype library

For the website to be fast, most of the operations had to be done in the client side rather than on the server side. To do that JavaScript is the best option. Moreover to give a good graphical look to the website, the Prototype JavaScript library was used.

3.2 System Architecture

The system architecture is shown in Figure 3.1. The figure shown is slightly different from the typical n-tier architecture, because the pages when loaded for the first time act similar to the normal asp.net web application, but the user interactions, where the user requests something, are handled in the form of asynchronous calls through Ajax. Hence, the left side of the architecture is used when loading the page for the first time, and to do operations that are not possible through Ajax. The right side of the architecture is used when the user requests something and calls to the server are made behind the scenes.

In the architecture shown below, Client UI is the client browser which is present in the client's machine. The Presentation layer consists of standard ASP.NET web forms,

12

and documents, etc. This layer works with the results/output of the business logic layer

and transforms the results into something usable and readable by the end user.
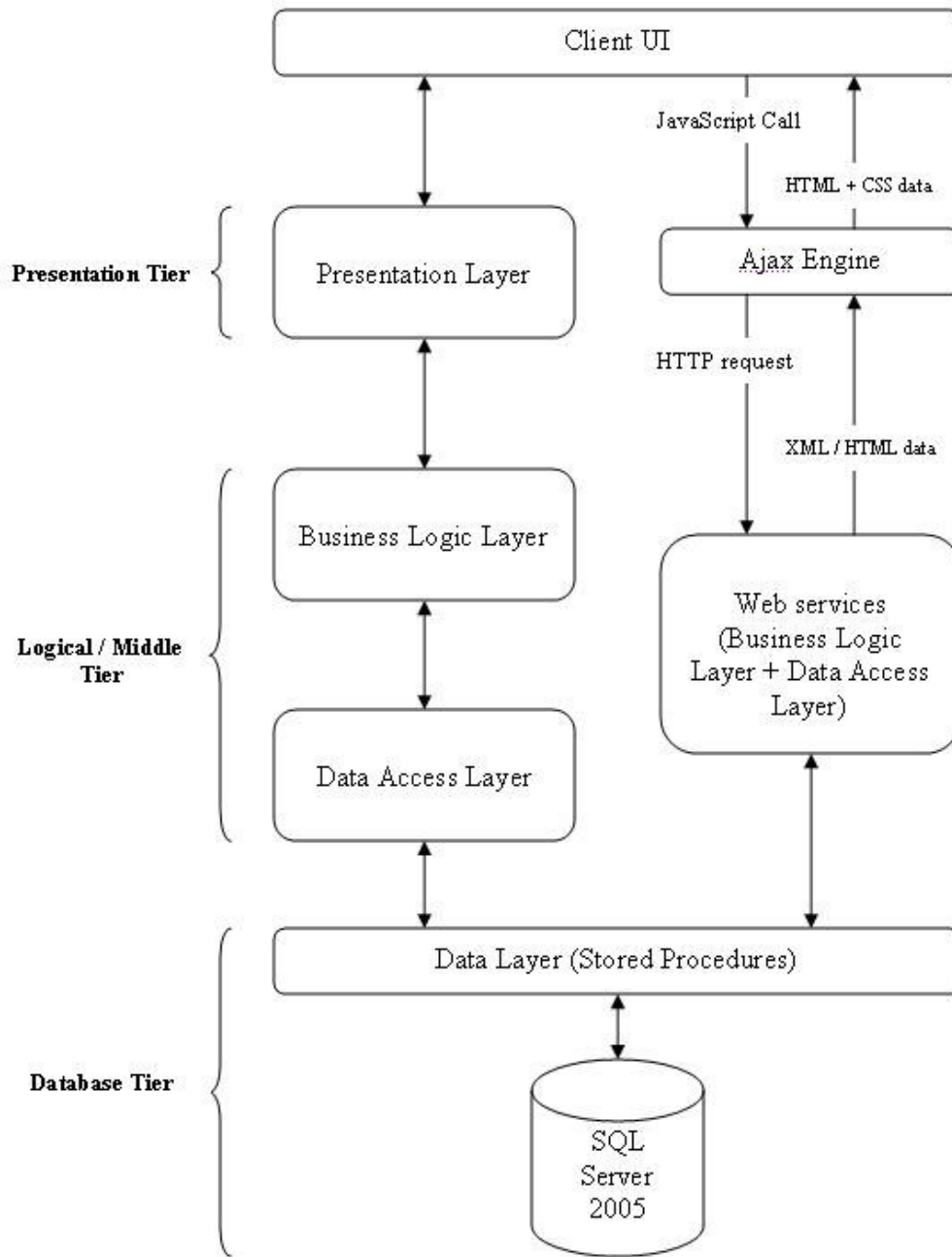
Figure 3.1 System Architecture

Business Logic Layer allows users to share and control business logic by isolating

it from the other layers of the application. The business layer functions between the

presentation layer and data access layers, sending the client's data requests to the database layer through the data access layer.

Data Access Layer provides access to the database by executing a set of SQL statements or stored procedures. This is where generic methods to interface with data is written like creating and opening a SqlConnection object, creating a SqlCommand object for executing a stored procedure, etc. As the name suggests, the data access layer contains no business rules or data manipulation/transformation logic. It is merely a reusable interface to the database [14]. This layer was not available in ASP.NET 1.1. In ASP.NET 1.1 both Business Logic layer and Data Access Layer combined were known as Logical Layer. The Business Logic Layer and Data Access Layer are usually in the App_Code folder in the application server. When using datasets to interact with the database they are separated, but if the interaction is done through C# classes they are combined into one layer.

Database Tier consists of database and data layer which consists of stored procedures to manipulate and retrieve data from the database i.e. SQL Server.

In the above architecture when the user requests something from the server, a JavaScript call is made to the Ajax Engine, which is Atlas here. The Ajax Engine creates an XmlHttpRequest object and sends it to the web service without affecting the control flow of the webpage. The web service interacts with the database by calling the stored procedure, gets the result from the database, converts it to XML and returns XML or HTML data by transforming the XML into HTML using XSLT to the Ajax Engine. If the result returned is XML, Ajax Engine processes it and transforms it to HTML, or else just the returned HTML is send to the client browser.

Apart from these usual advantages like scalability, availability, and ease of integration etc, the n-tier architecture also allows any of the n tiers to be upgraded or replaced independently as requirements or technology change. For example, a change of database from SQL Server to Oracle 10g would only affect the data storage layer and data access layer.

3.3 Database Design

The database schema for the website consists of six tables, out of which one table is used to store BLOBs. So if the files are stored in the file system, the table could be removed. There is one main table to store the primary details of the file uploaded, one table to keep track of various versions of the same file i.e. versioning control, two table for creating groups and sharing a file within that group, and the last table for organizing the files with use of folders.

The tables for managing users, roles, profile properties etc were in-built with ASP.NET 2.0 membership, role and profile providers. The *UserName* column present in the table created by membership provider is referenced by all the tables of the schema shown in Figure 3.2.
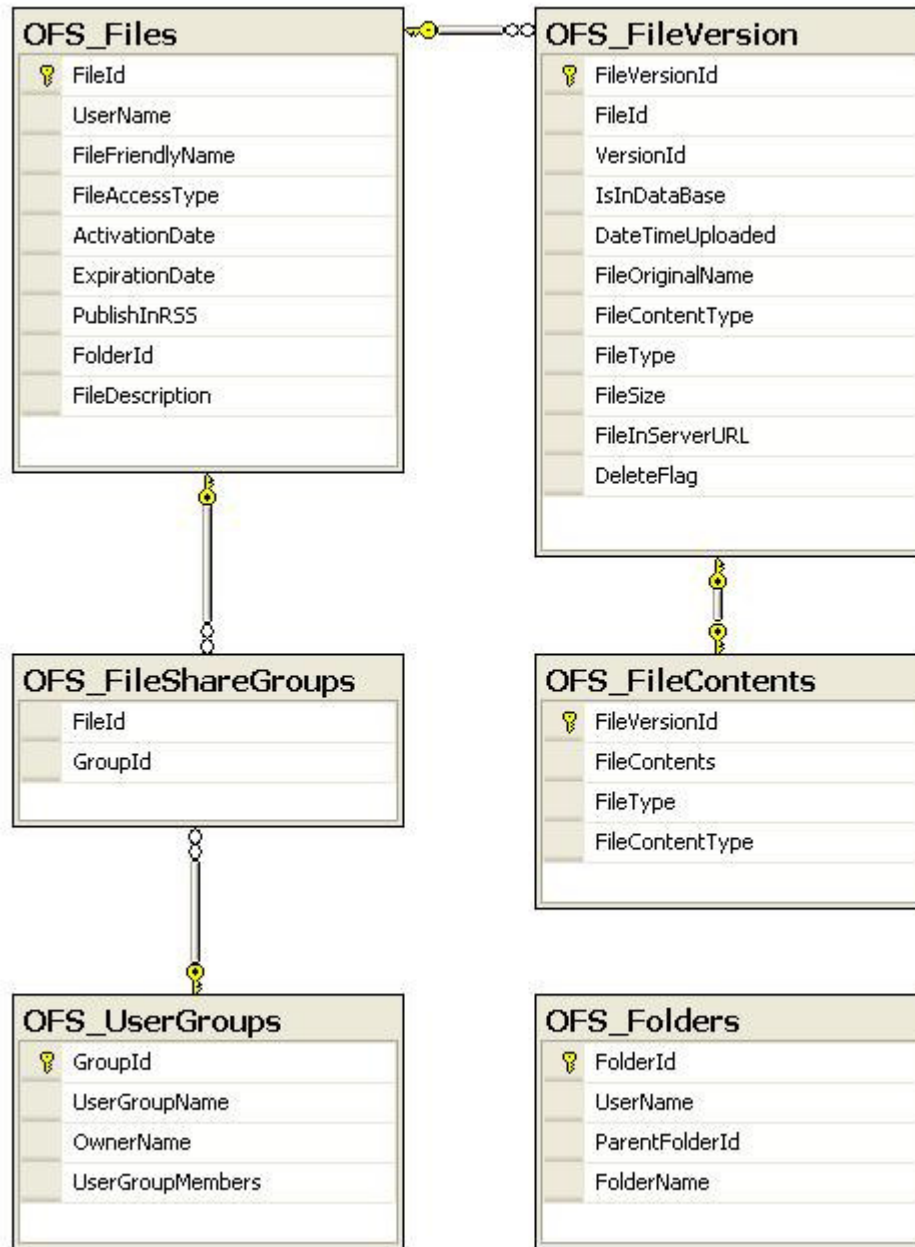
Figure 3.2 Database Design

3.4 Functionalities

Since the project was done for educational purposes, there was no size limit given to users to upload or download files. The main features of the website are:

3.4.1 User Registration

News users can register to upload files and create their own file system. The user management was easily integrated into the website with the providers that ship in with ASP.NET 2.0. Users have their own profile page to update their profile (using profile providers). These pages are Atlas enabled i.e. the page uses AtlasCTK extenders. Other than these users have got option to change their password, request for a new password etc.

3.4.2 Upload Files

This part is the heart of the project where the user can upload a file. Part of this page is Atlas enabled. File upload takes place in the form of a wizard control, where the user does the upload in a step by step manner. When the user uploads a file, he/she will be given a progress bar indicator to show what the status of the upload is. Metadata about the file upload can also be entered in the first step. Second step gives the option to enter the activation and expiration date of the uploaded files. Third step allows the user to organize the files by creating folders and the last step given the option of sharing. i.e. whether the file is public, private, or can be shared by groups.

3.4.3 Share Files

In the last step of the upload process, users can choose to share the file by creating groups. A group will consist of one or more registered users of the website. When a user is in a group, then only he/she will be able to see the file while searching or downloading. A user can have any number of groups. The process of creating groups, adding/deleting

17

members from the group are all Atlas enabled i.e. everything happens without page refreshes.

3.4.4 Version Control

When the users updates a file, the old file will be archived so that, if the user wants to access it later he/she can retrieve that file. Files will not be archived if the user deleted the files or the folder containing the files.

3.4.5 RSS Feeds

Users have an option to publish their RSS feed (RSS 2.0 format) so that others can use the feed to show the latest files changes and uploads in their website. Users will be given an option whether to include a file in the RSS feed while uploading the file.

3.4.6 Search

Another important feature available is the extensive search feature. It is a completely AJAX enabled (both simple search and advanced search) with no page refreshes at all. Concepts used to attain this no page refresh feature were AJAX, XML/XSLT, and web services. Search feature searches not just the file name and metadata about it, but through the document files uploaded like .doc, .xls, .htm, .ppt, .pdf etc. It makes use of the full-text indexing available in SQL Server to search through BLOBs and Microsoft's indexing service to search though the files stored in the server.

3.4.7 File management / File system visualization

As discussed in Chapter 1, one objective of this project was to give the users different visualizations to view and manage their file system. Chapter 4 discusses in more detail about management and visualization of file system.

3.5 Design

3.5.1 Use case diagram

Use case diagrams describe what a system does from the standpoint of an external observer [18]. The use case diagram in Figure 3.3 depicts the relationships among the actors and use cases where actors represent the external entities of the system and use cases represent the functional parts of the system.

In this web application, there are two external actors – Registered and Unregistered users. The 'include' association tells that the use case included includes the task described by the other use case. The different use cases present in the system can be seen in the figure below.

Figure 3.3 Use case diagram

3.5.2 Navigation flow diagram

Figure 3.4 shows the control flow of the entire application. When a user is not logged in he/she will only be able to access the Search and RSS features. Only logged in users will be able to access all the other features like upload/edit/deletion of files, file system visualizations, file search, group management, and profile management etc. All

the users will be able to download the files available or visible to them i.e. for unregistered users, the search results will only return files that are public. Also necessary authentication is done before downloading a file.
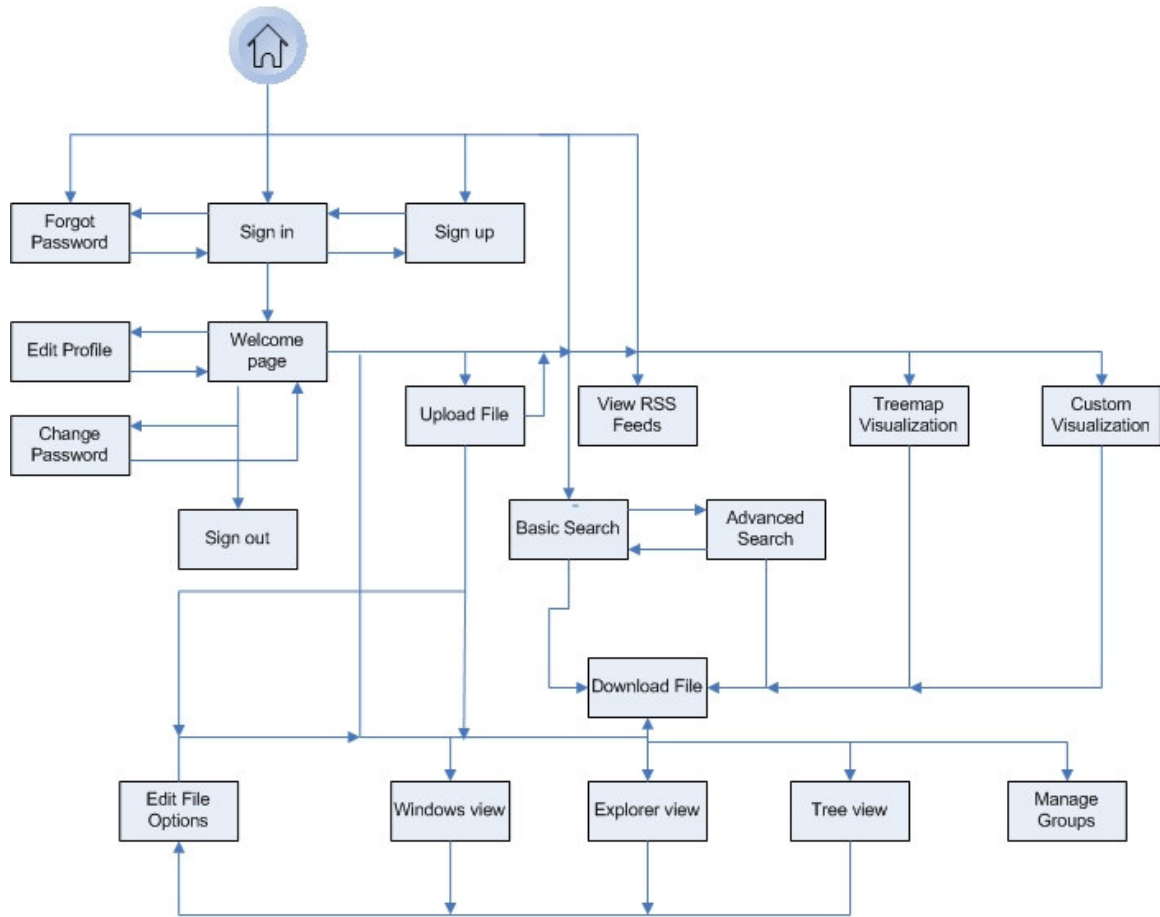
Figure 3.4 Navigation flow diagram

3.5.3 Class diagram

The class diagram in Figure 3.5 depicts the classes, components and their relationships among each other in the web application. It gives a very high level overview of the application. Since the website uses AJAX in most of the pages, web services are utilized the most. Hence there are five web services, out of which three are for file system management i.e. different file system views, one for search service, and one for managing

groups. The different pages access these web services and also the DB files which act as the logical layer i.e. Business Logic Layer + Data Access layer. The web services also act as the logical layer. There are also datasets in which the business logic layer and data access layer are separated. The datasets are usually utilized by the pages when the page is loaded. There is also the ProgressBar component which is used to show the progress of the upload as explained in the last section.

Classes related to user management are not depicted in the figure since all the functionalities are taken care by ASP.NET 2.0. All the classes depicted below except for the Search and RSS classes are available only to registered users.

Figure 3.5 Class diagram

3.6 Testing

3.6.1 ANTS Load Test

ANTS Load is used to predict a web application's behavior and performance under the stress of a multiple user load. It does this by simulating multiple clients accessing a web application at the same time, and measuring what happens. It has the ability to test web services too. It helps in finding out the slowest and fastest objects and pages in the web application. It is very important to do load testing because if the page takes a long time to load, users get frustrated and will leave the site. Hence the response time as well as the download time should be as low as possible.

Test script that was recorded to load test the website created a group, added members to the group, viewed an RSS page, viewed the tree view page in which files were dragged and dropped, renamed and deleted, viewed the windows view page in which files and folders were opened, deleted and renamed, and a search was done to search files in the database and server.

Figure 3.6 shows time required to connect, time to receive the first byte, and time to receive the last byte for the pages visited. The times shown below are in milliseconds, and it can be seen that the time to connect is negligible. Also time to receive the first byte and last byte is less than one second.

#	Page	Time to connect	Time to first byte	Time to last byte	Count
1	http://localhost:1770/OFS/Default.aspx	1	66	246	10
2	http://localhost:1770/OFS/Registered/MyGroups.aspx	1	161	226	5
3	http://localhost:1770/OFS/RSSFeeds.aspx	0	146	224	5
4	http://localhost:1770/OFS/Search.aspx	0	65	134	5
5	http://localhost:1770/OFS/Registered/UserProfile.aspx	0	85	180	5
6	http://localhost:1770/OFS/Registered/AjaxTreeView.aspx	0	691	963	5

Figure 3.6 Test summary information by page

24

Figure 3.7 shows the average timings for each object of the web application. Time required connecting, to receive the first byte and last byte is less than one second even when the number of bytes received is greater than 60,000 bytes. It can also be seen that the average timings to access a web service is significantly less.

#	Name	Time to connect	Time to first byte	Time to last byte	Bytes received	Count
1	GET http://localhost:1770/OFS/Default.aspx	0	69	73	13722	5
2	GET http://localhost:1770/OFS/WebResource.axd	0	14	21	88383	165
3	POST http://localhost:1770/OFS/Default.aspx	1	64	65	14162	5
4	GET http://localhost:1770/OFS/Registered/UserProfile.aspx	0	80	82	24770	10
5	GET http://localhost:1770/OFS/Registered/MyGroups.aspx	1	161	162	24260	5
6	GET http://localhost:1770/OFS/WebServices/OFSGroupService.asmx/js	0	6	6	497	5
7	POST http://localhost:1770/OFS/Registered/MyGroups.aspx	0	56	58	15768	25
8	GET http://localhost:1770/OFS/Images/Save.gif	0	2	2	371	10
9	GET http://localhost:1770/OFS/RSSFeeds.aspx	0	146	148	11912	5
10	GET http://localhost:1770/OFS/Images/feedicon.gif	0	2	3	910	15
11	POST http://localhost:1770/OFS/RSSFeeds.aspx	0	80	98	4148	5
12	GET http://localhost:1770/OFS/Search.aspx	0	65	68	16068	5
13	GET http://localhost:1770/OFS/Validation.js	0	3	4	2998	5
14	GET http://localhost:1770/OFS/WebServices/OFSSearchService.asmx/js	0	7	8	755	5
15	POST http://localhost:1770/OFS/WebServices/OFSSearchService.asmx	0	41	42	2136	30
16	POST http://localhost:1770/OFS/Search.aspx	0	12	13	4874	15
17	GET http://localhost:1770/OFS/Registered/AjaxTreeView.aspx	0	691	711	622662	5
18	GET http://localhost:1770/OFS/WebServices/OFSTreeViewService.asmx/js	0	9	9	848	5
19	POST http://localhost:1770/OFS/WebServices/OFSTreeViewService.asmx	0	20	21	608	40

Figure 3.7 Test summary information by objects

Test results also showed that the time to connect was always less than 50 milliseconds for both pages and objects. Time to receive the first and last byte occasionally went up to one or two seconds, but was usually less than 100 milliseconds.
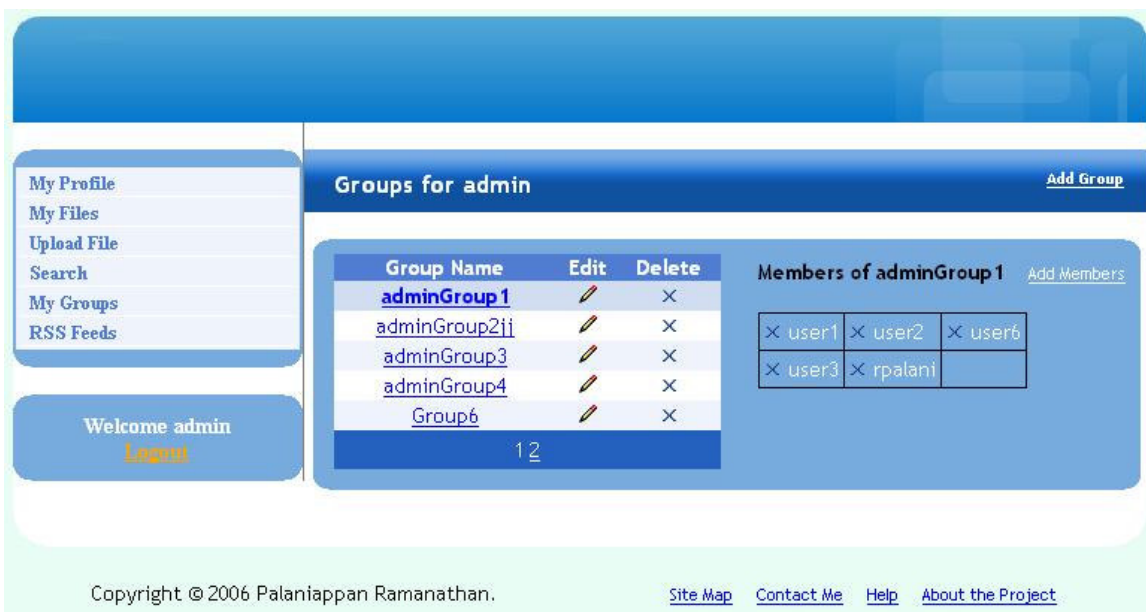
25

3.7 Screen Shots

Figure 3.8 Profile page
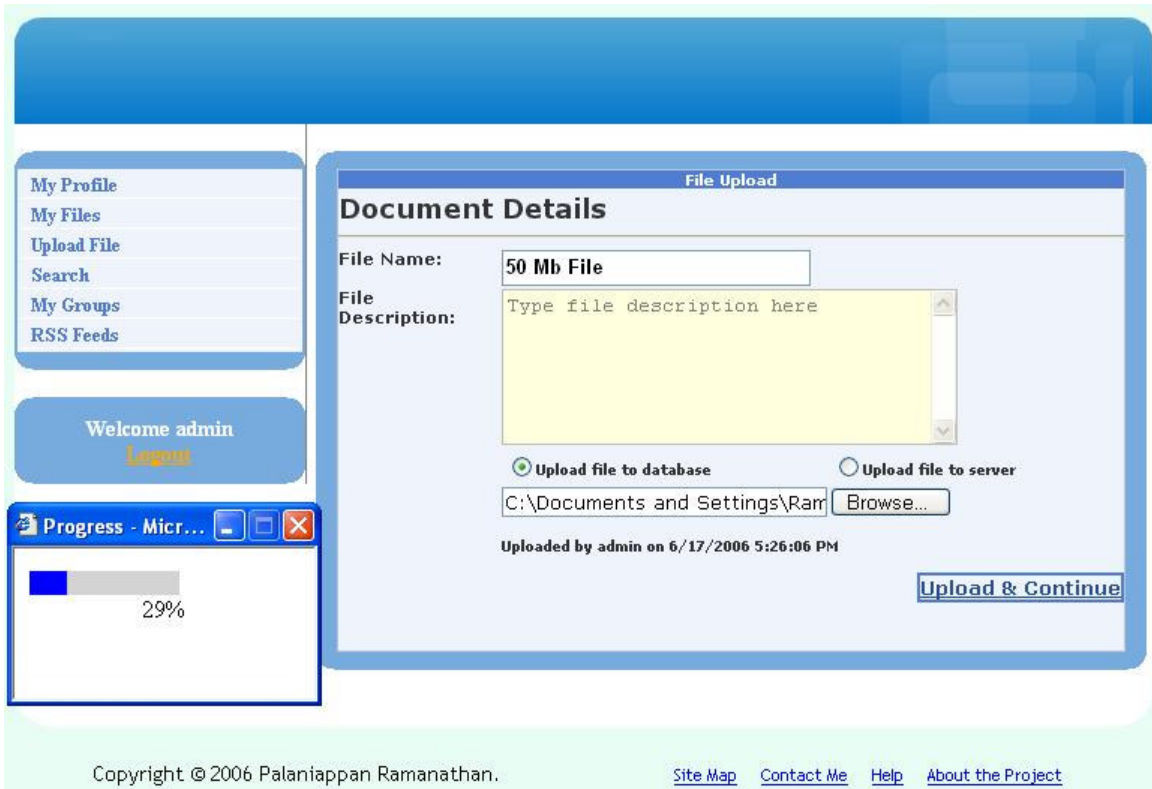

Figure 3.9 User groups

26

Figure 3.10 Upload page

Figure 3.11 Search page

27

Chapter 4 - File System Views

This chapter discusses the different file system visualizations and file management options available to the users. Section 4.1 discusses about the space constrained visualizations in detail i.e. what are the techniques used, implementation details and advantages of such visualizations. Section 4.2 discusses about the traditional file management options available to the users. Section 4.3 gives a comparison of both visualizations with respect to ease of coding, interactivity and user friendliness.

4.1 Space-constrained visualization

Visualization is a method for seeing the unseen [10]. Thus a custom visualization of a file system should allow the users to see something that they cannot see in the traditional visualization. Traditional visualization mentioned here is the file management option available in Microsoft windows and other operating systems.

If the website developed for this project was commercial and not educational, then each user will surely have a size limit he/she can upload files to his/her account. In that case the users would find a space constrained visualization of their hierarchical file system very useful, which is discussed in the next two sub sections.

Developing and integrating file management options in space constrained visualization is possible, but requires a lot of time and coding, which was not an objective of this project. Hence users were only given option to navigate through the file system and view the visualization, and the ability to edit/delete files and folders were not given.

Next two sub sections of this document discusses in detail about the two space-constrained visualizations developed for the project.

4.1.1 Treemap visualization

Treemap concept was initially developed by Ben Shneiderman during 1990 in the HCI Lab at the University of Maryland to show a file system in a space-constrained layout. His initial design simply nested the rectangles, and used borders to show nesting i.e. the technique mapped a tree structure (e.g. file directory) into nested rectangles with each rectangle representing a node. A rectangular area is first allocated to hold the representation of the tree, and this area is then subdivided into a set of rectangles that represent the top level of the tree. This process continues recursively on the resulting rectangles to represent each lower level of the tree, each level alternating between vertical and horizontal subdivision. The parent-child relationship is indicated by enclosing the child-rectangle by its parent-rectangle i.e. all descendents of a node are displayed as rectangles inside its rectangle. Associated with each node is a numeric value (e.g. size of a directory) and the size of a nodes rectangle is proportional to its value [8].

Later many PhD and Masters students contributed to the initial design by adding features like zooming, hue/saturation control, many border variations, and labeling control [8]. Later the application was adapted to different operating systems.

While all the research and tools done on treemaps have been for desktop applications, this project implemented similar treemap visualization for visualizing a file system on a web site. There are several algorithms for placing the rectangles in the treemap. A simple algorithm [6] in shown in Figure 4.1

29

1. Decide on a position and size for the root rectangle that contains all other rectangles. This can be set to fill the entire screen area, for example.

2. Divide the root rectangle into as many sub-rectangles as there are children of the root node. Each of these rectangles is as high as the root rectangle, so they go from its bottom to its top. Each sub-rectangle has width proportional to its size divided by the total size of all children. If the rectangle represents a file, its size is the size of that file in. For a directory, the sum of all sizes of its contents is used.

3. Repeat step 2 and 3 for every sub-node of the root node, with the determined rectangle for that node as root node. Alternate the direction of the subdivision this time, so that sub-nodes are placed vertically if they were placed horizontally this time and horizontally otherwise. Alternating the direction of subdivision makes it possible to see to which level of the hierarchy rectangles belong.

Figure 4.1 Simple treemap algorithm

The disadvantage of the above algorithm is that, when the number are files and folders in a parent folder is many, the algorithm produces thin, elongated rectangles which are difficult to compare and analyze. Hence the algorithm used for this project is different from the above one. It uses the sqaurified treemap algorithm to make the rectangles look like squares as much as possible. This project used the *TreeMapGenerator* library developed by the Microsoft Research Community Technologies team which had the algorithm already implemented and integrated. The paper Squarified Treemaps [7] gives the squarified treemap layout algorithm.

4.1.1.1 Features

This visualization allows the user to look at the file system at any depth i.e. how the files occupy the space at level x. User can go into the subdirectory by clicking on the link to it and open file in a similar manner. To improve the visualization, color of each cell was made distinct from its adjacent cells. This visualization is completely Ajax

enabled i.e. no page refreshes. A sample screen shot of the treemap visualization at level 1 and depth 1 is shown in Figure 4.2.
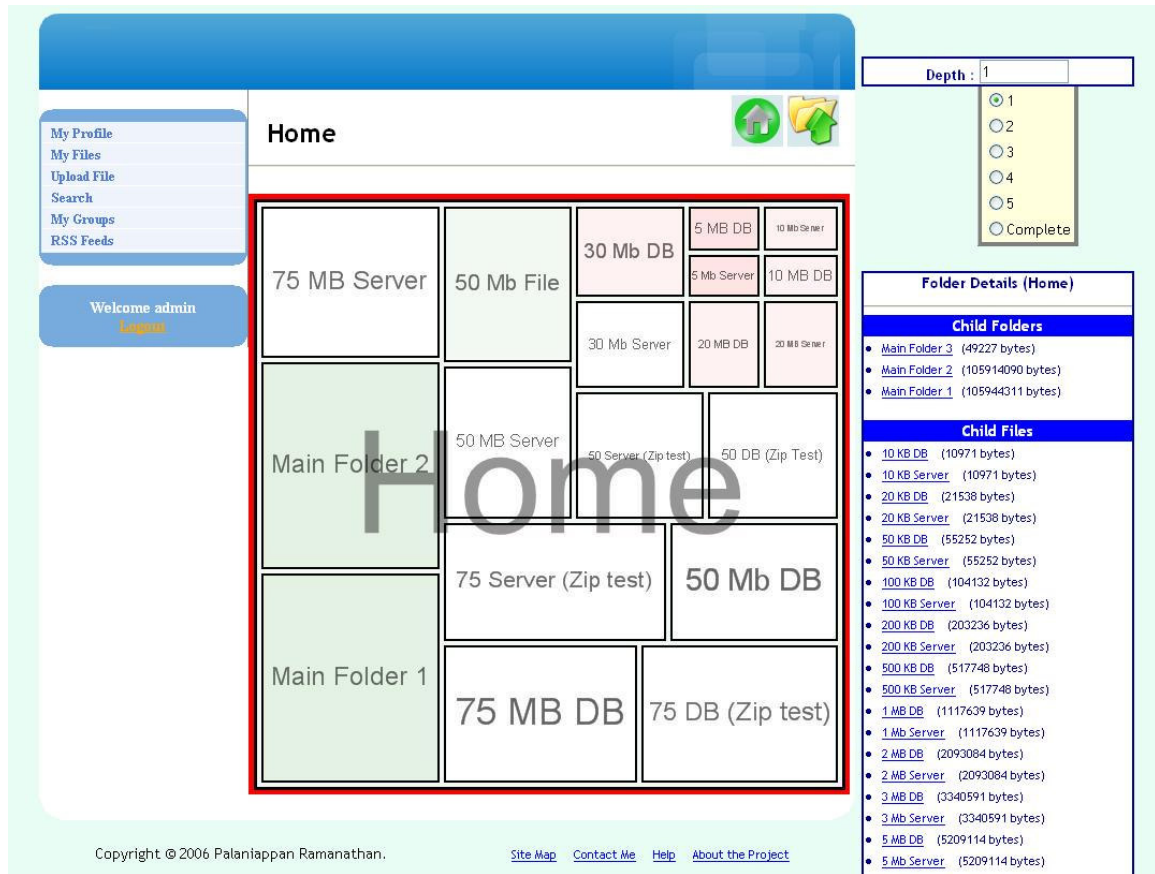
Figure 4.2 Treemap visualization

4.1.1.2 Limitations

Limitations in this implementation of treemap are:

- Features for file manipulation were not implemented i.e. feature to edit or delete a file. Only features to navigate through the file system were implemented.

- Small files or folders of smaller size are not visible i.e. if there is a file which is 100 KB and another file which is 1 KB, the smaller file would not be very visible. It as actually a limitation in all implementations of

31

treemaps in desktop application too, but this limitation has been overcome in desktop application with zooming facility.

4.1.1.3 Problems faced

Some problems faced during the implementation of this version of treemap which used the *TreeMapGenerator* Library from Microsoft were:

- When the feature to increase the depth was implemented, all the folders names were jumbled with its child folders and files. Especially when the depth was set to complete, all the folder names and file names were generated inside its respective rectangle and nothing was legible. In order to overcome this problem only the last level of folder names and file names were generated.

4.1.1.4 Advantages

Treemap visualization has got its own set of advantages and disadvantages. Some disadvantages were discussed in Section 4.1.1.2. Some of its advantages are:

- The user can know how his/her files have occupied his/her allotted space and the amount of free space available.

- Helps the user to know how a particular folder is occupied i.e. the ratio of sizes of child files and child folders in it.

4.1.2 Custom visualization

The custom space constrained visualization was developed based on XCruiser, a file system visualization tool for desktop applications. This visualization is a very limited version of its original with only the basic look and basic functionalities implemented.

4.1.2.1 Features

The file system is visualized with 2D spheres. The parent folder is represented by a big blue 2D sphere, child folders are represented by blue 2D spheres with its area proportional to its size, child files are represented by brown 2D spheres with its area proportional to its size, and empty folders are represented by a small grey 2D sphere. File/Folders details can be viewed by placing the mouse over a sphere, which pops up a hover menu. Files can be opened by clicking on the respective spheres. Navigation into child folders is possible by clicking on a folder which would open a new modal window containing the files and folders in it. Screen shot of the custom visualization shown in Figure 4.3 shows the new modal window opened for navigation and the hover menu for viewing file/folder details.
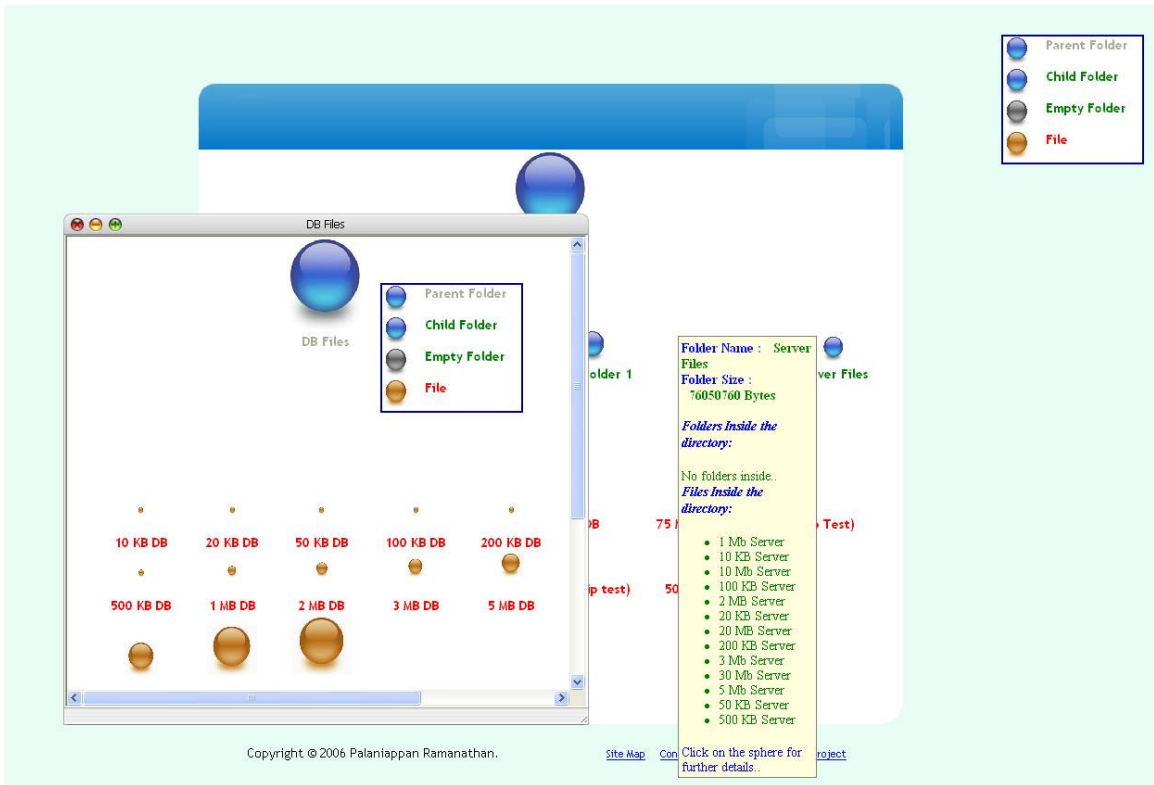

Figure 4.3 Custom visualization

4.1.2.2 Limitations

Due to the limitations of technologies involved, and the complexity of coding needed, some cool features like zooming into a folder or a file, relationships between ancestors, etc were not implemented.

Also people would find it more difficult to figure out the ratio of child folder to parent folder just by looking at it i.e. if the visualization was in the form of a Venn diagram, it would be easier to find out ratio of the size of child folder.

4.1.2.3 Advantages

With the custom visualization, user can see all files and folders available, unlike the treemap visualization where really small files would not be seen. In fact, in this visualization, the user can see even empty folders as grey spheres. User can navigate through any folder to any level since every folder is opened in a new window like the file managers in Linux operating systems. Like all other visualizations implemented for this project, this one is also Ajax enabled, which makes it very user friendly.

4.2 AJAX based file management

Since users are used to managing their files and folders in desktop applications, a desktop style file management would be the easiest way to get used to managing file system in a website. In this website users were given three desktop style interactive AJAX based file managers. It is the traditional file visualization.

4.2.1.1 Windows view

In this traditional view shown in Figure 4.4, users have the option to open/rename/delete folders, option to open/rename/delete/edit files, option to go to home

directory, go back to the parent folder, option to refresh the current folder etc. All the operations involved do not post back, but they give a *loading* message as seen in the figure while the modifications are done to the page. Technologies involved in developing this file manager were Atlas, Prototype library etc.
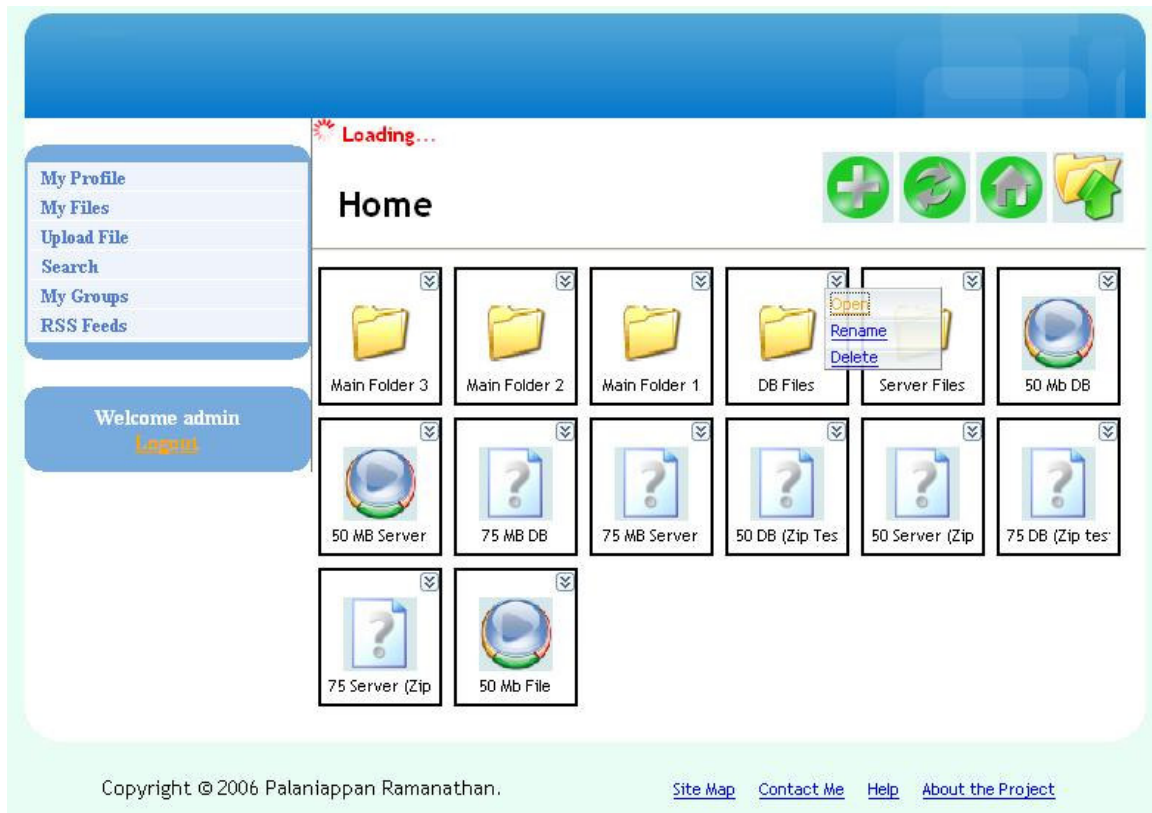
Figure 4.4 Windows view

4.2.1.2 Explorer view

Explorer view is similar to the Windows explorer available in Microsoft windows. In addition to all the functionalities of the above view, Explorer view has got a tree view to navigate more easily as shown in Figure 4.5. Advantage of this view over the Windows view is the navigation i.e. user can go to a child filer without traversing through all its parent folders.
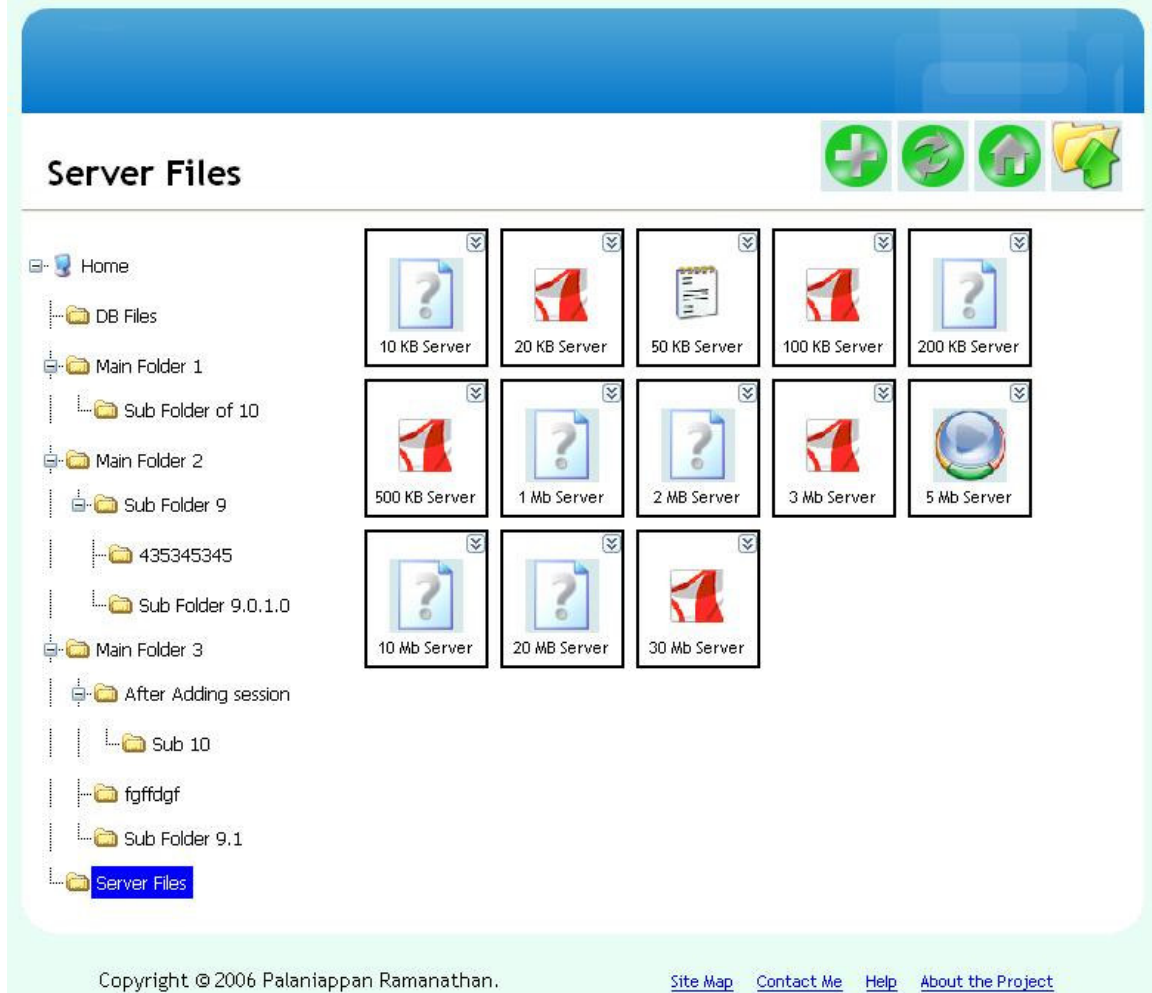
Figure 4.5 Explorer view

4.2.1.3 Drag N Drop Tree view

The desktop feel that was missing in the above two file managers was the drag and drop option and the right click menu option. The Drag N Drop Tree view has got all the features mentioned above. Figure 4.6 and Figure 4.7 illustrate that. Concepts used for developing this file manager were AJAX, XML/XSLT, JavaScript, Prototype Library, and web services etc.

Figure 4.6 Tree view showing drag and drop option

Figure 4.7 Tree view showing context menu

4.3 Comparison

This section of the document compares space-constrained visualization and traditional visualization implemented for this project and in general with respect to usability, features and complexity of coding.

Space-constrained visualizations implemented for this project does not have file management options. So usability wise the traditional visualization would attract users more since they are used to traditional file management. If the space-constrained visualization had the file manager incorporated with it, even then it would take some time for the users to get used to it. If the space constrained visualization is implemented as a 3D visualization with zooming feature, users would find it more interactive i.e. like a combination of TreeV of fsv (file system visualization) and XCruiser. But implementing a 3D visualization with zoom feature as discussed in Chapter 2 would require different technologies and more time. Hence with more features added to the space-constrained visualization, it could be better usability wise, but at the level both visualizations are implemented for this project, the traditional visualization (especially Drag N Drop Tree View) would be more users friendly.

Both traditional and space constrained visualizations implemented have their own unique features. As discussed above, if the file management options are added to the space constrained visualizations, then for sure the space constrained visualizations would have more features. Features of all visualizations implemented have been discussed in the previous sections of this chapter.

Complexity of coding in implementing space-constrained visualizations was not much, since it did not have the file management options (i.e. drag and drop, context menu

etc). But while implementing the traditional visualization, Ajax based file managers required more complicated coding. Thus the level of difficulty (i.e. complexity of coding, time required, lines of code etc) in implementing these visualizations can be rated in ascending order as:

- Traditional file system visualization with postbacks - easiest of all because of the in built features of ASP.NET 2.0

- Space-constrained visualization – easier than the Ajax based file visualization, since the file management option was not implemented. Also since the *TreeMapGenerator* library was used, the need to implement the Treemap algorithm was not needed.

- Ajax based file managers – Time required and the lines of code in implementing this file visualization were more than all the other visualizations.

Chapter 5 - Database Vs File System

This chapter compares database and file system with respect to file storage. As discussed in Chapter 2, the two primary options of file storage in a web application are storing files as BLOBs in database and as files in file system of the web server. This chapter explores both possibilities by first presenting the results obtained from the tests i.e. time required to upload, download and search files present in both places, and then gives a comparison of both techniques.

5.1 Storing files in database

When storing the uploaded file in database, all the file details i.e. file name, file size, file type, share options etc and the BLOB i.e. binary representation of the file, are stored in the database (BLOB data types in SQL Server are image, text and ntext). The data type used to store BLOBs for this project was the *image* data type. Full-text indexing were enabled for the BLOB column, file name column and file description column to speed up the search. BLOBs were inserted to the database using transactions with 32 kilobytes of data being appended to the BLOB in each round. Transactions were used since the file being uploaded can be a large file.

While downloading the file, the binary data (BLOB), content type, file name etc are retrieved from the database, and the binary data is written to the output stream after setting the content type of the output stream. Necessary authentication procedures are done before retrieving and writing the file to the output stream.

5.2 Storing files in file system

When storing the uploaded file in the file system, the file details i.e. file name, file size, file type, share options etc are stored in the database, the file is stored in the file system on the web server inside the uploads directory with a naming mechanism. While saving the file to the server, the file is appended in 32 KB chunks. After saving the file to the file system on the web server, the specially given name for the file in the file system is updated to the database in its respective row i.e. the path to the saved file in server is updated in the database.

While downloading the file from server, if direct access to the file is given by giving the absolute path to the file, then anyone can download any file with a brute force attack. Since the direct access option isn't realistic, the file should be converted to a byte array and should be written to output stream of the response object, after setting the content type and name of the file to the values retrieved from database. Even though this option is slower than the direct access method, it is more realistic where you can allow only authenticated users to download the file.

5.3 Results obtained

In order to analyze the performance of both techniques, a simple testing was conducted and the results were recorded. Tests were done to record the download time, upload time and time needed to search files.

Table 5.1 shows the details of files that were used to test the upload and download time. All the files shown were uploaded five times to the server and five times to the database. Similarly they were downloaded five times from database and server.

File size (Approximate)	File size (Actual)	File Type
10 KB	10 KB	.ml
20 KB	21 KB	.pdf
50 KB	54 KB	.js
100 KB	101 KB	.pdf
200 KB	198 KB	.zip
500 KB	505 KB	.pdf
1 MB	1.06 MB	.zip
2 MB	1.99 MB	.chm
3 MB	3.11 MB	.pdf
5 MB	4.98 MB	.mp3
10 MB	9.75 MB	.exe
20 MB	20.4 MB	.chm
30 MB	30.3 MB	.pdf
50 MB	48.9 MB	.avi
75 MB	74.9 MB	.mdf
100 MB	105 MB	.avi

Table 5.1 – Test suite

5.3.1 File upload results

Files of sizes given in Table 5.1 were uploaded sequentially to database and server and the time needed to do those operations were recorded. All the files were uploaded five times to database and server. Table 5.2 shows the average time (of the five timings recorded) taken to upload files to database and server in milliseconds. It also gives the standard deviation of the timings recorded.

Figure 5.1 gives the graphical representation of the results obtained. In the figure, x-axis represents the file size and y-axis represents the time required to upload the file in milliseconds. From the figure it is very clear that there is only a very negligible difference in the time required to upload a file which is 3 MB or less (since the timing recorded are in milliseconds). But there is a dramatic rise in the time required to store a file in the database which is 5 MB or more. So if the size of the file uploaded is going to be usually small, then there would be no big difference.

42

File size	Average time (Database)	Average time (Server)	Std deviation (Database)	Std deviation (Server)
10 KB	37	57	21	19
20 KB	27	35	23	23
50 KB	23	59	22	39
100 KB	66	35	53	24
200 KB	57	47	61	32
500 KB	79	63	58	39
1 MB	113	89	81	25
2 MB	822	227	1096	134
3 MB	1086	227	1721	73
5 MB	2478	263	3880	40
10 MB	4860	554	4442	91
20 MB	10177	2292	7811	1872
30 MB	13596	3998	7289	1476
50 MB	17918	4672	3122	715
75 MB	38955	7035	20585	1956
100 MB	57978	14732	41924	6535

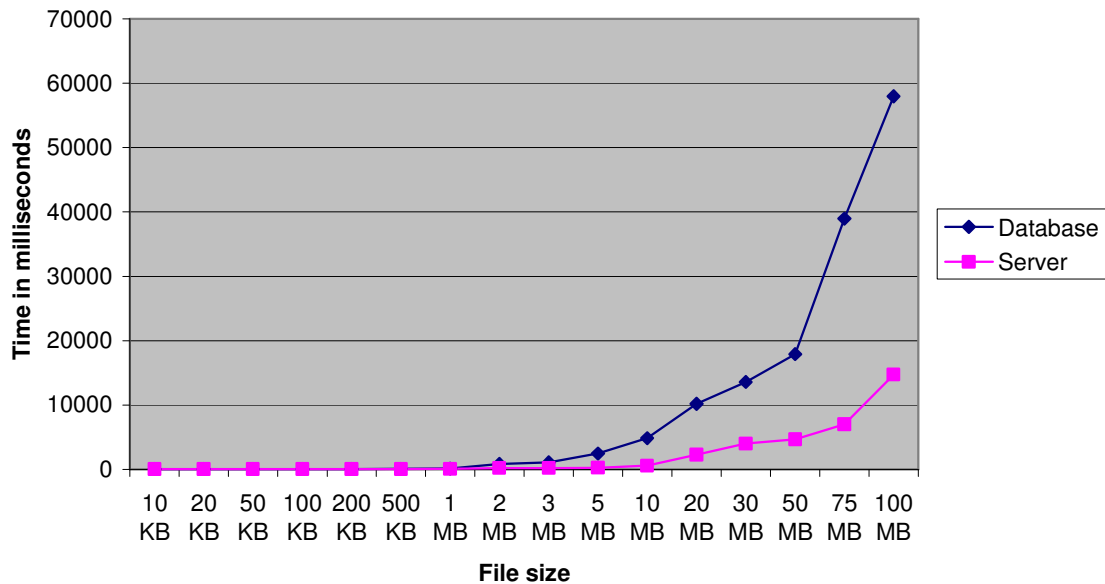**Table 5.2 – Database Vs Server (Upload)**

Figure 5.1 Database Vs Server (Upload)

5.3.2 File download results

Files of sizes given in Table 5.1 were downloaded sequentially from database and server and the time needed to do those operations were recorded. All the files were

43

downloaded five times from database and server. Table 5.3 shows the average time (of the five timings recorded) taken to download files from database and server in milliseconds. It also gives the standard deviation of the timings recorded.

File size	Average time (Database)	Average time (Server)	Std deviation (Database)	Std deviation (Server)
10 KB	82	100	67	91
20 KB	126	182	65	178
50 KB	82	138	29	73
100 KB	120	114	36	53
200 KB	140	196	83	78
500 KB	238	344	58	379
1 MB	270	170	130	82
2 MB	364	208	62	107
3 MB	557	2982	40	1745
5 MB	959	2141	532	1019
10 MB	3269	7877	1127	2955
20 MB	5972	9502	954	3569
30 MB	8107	13623	833	3146
50 MB	11829	17970	2135	1826
75 MB	18070	23386	1817	2202
100 MB	25434	25446	4258	2158

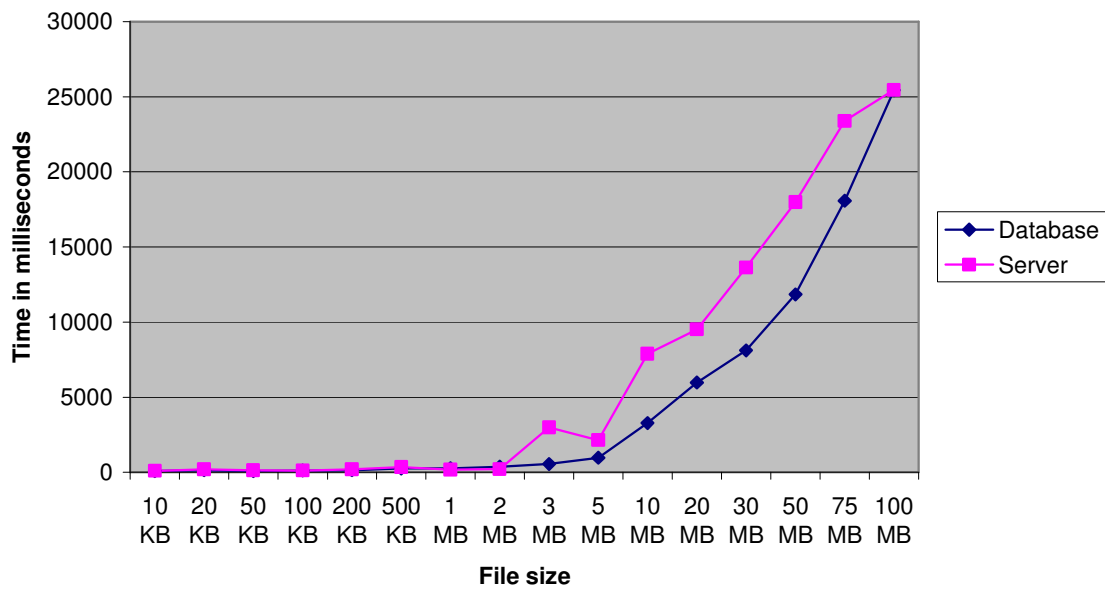**Table 5.3 – Database Vs Server (Download)**

Figure 5.2 Database Vs Server (Download)

44

The timings recorded above, are the time required to read the BLOB from database into a byte array / read the contents of a file in server into byte array, and write it to the output stream of the response object. It can be seen from Table 5.3 and Figure 5.2 that both techniques perform in a similar way, unlike the timings recorded when uploading a file. The server method seems to take a little bit more time than reading a BLOB. If the direct path to the file in server was given instead of reading the contents into a byte array and writing it to output stream, the server method would be much faster than reading a BLOB.

5.3.3 File Search results

In order to compare the search performance of both techniques, twenty searchable files i.e. .doc, .pdf, .xls, .htm, .ppt were uploaded to both database and server. Then the search was done with keywords present inside the documents and files.

Search text	Time taken to search files in database (ms)	Time taken to search files in server (ms)	Number of records returned
Obtention	30	70	1
Requirement	30	420	10
Agents	40	170	2
Wiki	70	120	2
Simulation	30	200	5
Eligibility immigration	40	150	2
Examination schedule	40	270	6
Visualization	30	160	2
Microarray	70	140	1
Compiler content management	90	390	14

Table 5.4 – Database Vs Server – Search comparison

45

Searching through BLOBs in SQL Server was done with the help of full-text indexing, and searching through the files in file system was done with the help of Microsoft's indexing service. Table 5.4 shows the time taken by both techniques to complete the search, number of records returned and the search word used.

As it can be seen in the table above searching through the files in the file system takes more time than searching through BLOBs. Searching files in file system is slower because, unlike the database technique were the search is completely done in one query, in the file system search, the file details stored in the database is searched separately, and the files in the file system are searched separately, and then the results should be compared and combined, which takes a lot of time. Another problem is that, while searching files in the file system, only the files that the searching user can have access should be searched. So each time a file is about to be searched, the control should go to the database to check if the user has got access to the file which would also take a lot of time.

5.4 Comparison

5.4.1 Performance

When a file is stored in the file system and direct access to the file is given, the time needed to download the file would surely be lesser than the time needed to download a file that is stored as a BLOB in database. But, if authentication is necessary before access to download is given, then the method described in Section 5.2 is necessary. In that case both methods take almost the same time to download a file.

As discussed in Section 5.3, while uploading a file to database or server, time taken to upload a file which is 3 MB or less takes almost the same time, since the timings recorded are in milliseconds. But for files which are 5 MB or more the server option seems to perform better.

So, if files that are being stored are usually small, then both methods perform well. If the files stored are the like images that are used in the web site and are visible to all users, then the server option would perform better, because there is no need for any authentication and direct access to the file can be given while downloading.

When searching a through a file, if the file is stored as a blob, full-text indexing of SQL Server can be used and the search would be faster since one query can be used to search the blob, file name, metadata etc. But if the file is stored in the server, then Microsoft's indexing service can be used to search the files, but the need to have two queries (one to search in database and another the file in server), and the time needed to combine the results of both queries would slower the search operation.

5.4.2 Maintenance

Taking backup of the uploaded files would be much easier if the files are stored in database, since it would happen in the process of taking regular backup, whereas if the files are stored in the file system, the backup has to be taken separately. Moreover the copy of multiple files can be stored in the database without coming up with some naming mechanism.

When migrating to a different database (say SQL Server to Oracle), the BLOBs would not be compatible, but if the server is changed there would be no problem.

47

5.4.3 Integrity

When files are stored in database, the database's built-in referential integrity checking can be used to check the integrity of the files, whereas if the files are stored in the file system, then the integrity should be checked manually.

Similarly when performing backup of the database, where files are stored in the file system and its location in the database, extra care should be taken to preserve integrity between the file and the associated database record. If the files are stored as BLOBs then no extra care is needed.

5.4.4 Security

Files stored in database are more secure then files stored in file system. If users find the path of uploads directory, then they can try different names to get access to the files in that folder if there are no security restrictions for the folder. Anyway with the files stored in database users cannot do that.

Moreover files stored in the database are isolated from the file system. Hence if the file system is cracked, it will not affect the files stored in database. If the files are stored in the file system then it could be a security risk. One way to overcome this risk is to store the files outside the document root of the web server.

5.4.5 Code complexity

Storing BLOBs to database, and retrieving them from database is more complicated than storing files in file system and retrieving files from file system. For a novice web developer, procedures involved into storing and retrieving BLOBs is more complicated than the other option. Especially when the file being uploaded is big, and the

file has to be uploaded to the database using transaction by appending chunks of data, it is more complex than just saving a file to the file system.

Lines of code needed to store and retrieve BLOBs are more than storing files to file system and retrieving files from file system. Also with in-built controls like ASP.NET 2.0 file upload control, uploading files to the file system is comparatively very easy.

Chapter 6 - Conclusions and Future work

6.1 Conclusions

An interactive file sharing website with very few page refreshes was developed using ASP.NET 2.0 and SQL Server 2005 as the back end. Users were given multiple views for viewing their file system. There were three AJAX based traditional desktop style file system visualizations with file management options, two space-constrained visualizations without file management options and one traditional file system visualization with file management options. Custom visualizations would take more time for the users to get adjusted, but would be helpful in many ways. Also other features like RSS Syndication, share groups, search etc were integrated.

An analysis was done on where to store the uploaded files – database or file system. Storing files and large data in database is not new. Microsoft's sharepoint products and technologies store all contents, document files, images, etc in SQL Server. Also WinFS (Windows Future Storage) plans to store all the files and data in a relational database derived from SQL Server 2005.

The answer to the above question depends on the requirements of the application. In the website developed for this project, it is clear that the upload time increases for files stored in database as the file size increases, but download time of files, whether stored in database or file system is almost the same. Also database has got more advantages when considering maintenance, security, integrity etc. Considering these facts, the administrator can store the files in the database, if the users do not care much about the

50

upload time, since the administrator would find it easy to administer and maintain the files when stored in the database. But if upload time is taken into consideration, and then file system option would be better. Also if authentication is not required to download file (e.g. images that are used in a web application), then file system option would perform much better while uploading as well as downloading. If the question is asked when developing a document management system, where users will use the search feature to search trough the files than any other feature, then database option would be much better than the file system option, since it is faster in retrieving the search results.

6.2 Future work

The website could be made more interactive by having an AJAX file upload. Right now the file is not uploaded asynchronously, but if the file is uploaded asynchronously, the user could enter other options and details of the files while the file is being uploaded (like attaching a file in GMail).

The space constrained visualization could be made better by adding file management features to it as discussed in the previous sections. Also navigation through the file system can be made better by having features like zooming, preview, etc.

The analysis between database and file system could have been done better, by uploading, downloading and searching files from a different computer that does not host the web application. Analysis would have also been better if files of bigger size (like 1GB or 2 GB) were uploaded and downloaded.

References

[1] Microsoft, "Conserving Resources When Writing BLOB Values to SQL Server", http://msdn2.microsoft.com/en-us/library/3517w44b.aspx

[2] Daniel Anderson, "Paging through Records using a Stored Procedure", June 2003, http://www.aspfaqs.com/webtech/062899-1.shtml

[3] Joe Slovinski, "Advanced UI Design Using XML and XSL", http://www.15seconds.com/issue/010921.htm

[4] Thiru Thangarathnam, "Professional ASP.NET 2.0 XML", 2006

[5] Sikha Saha Bagui & Richard Walsh Earp, "Learning SQL on SQL Server 2005", 2006.

[6] Niklas Höglund, "3D Graphics in the User Interface of a File System Browser", Royal Institute of technology, Sweden, 2004.

[7] Mark Bruls, Kees Huizing, and Jarke J. vanWijk, "Squarified Treemaps", Eindhoven University of Technology, The Netherlands.

[8] Ben Shneiderman, "Treemap for space-constrained visualization of hierarchies", April 2006, http://www.cs.umd.edu/hcil/treemap-history/index.shtml

[9] Ben Shneiderman, "Tree visualization with tree-maps: A 2-D space-filling approach", ACM Transactions on Graphics, 11(1):92–99, 1992.

[10] Klaus Mueller, Yinlong Sun and Penny Rheingans, "Introduction to Visualization", http://www.cs.umbc.edu/~alark1/cmsc435/lectures/Visualization.pdf

[11] MyCongress, "Storing files in a database vs. on the file system", June 2005, http://mycongress.org/blog/2005/06/29/storing-files-in-a-database-vs-on-the-filesystem/

[12] ASPFAQ, "Should I store images in the database or the file system?", August 2000, http://www.aspfaq.com/show.asp?id=2149

[13] Microsoft, "Chapter 11 – Using BLOBs", http://www.microsoft.com/technet/prodtechnol/sql/2000/reskit/part3/c1161.mspx?mfr=true

[14] Thiru Thangarathinam, "N-Tier Web Applications using ASP.NET 2.0 and SQL Server 2005", 2005, http://www.15seconds.com/issue/050721.htm

[15] Wikipedia, "File Sharing", http://en.wikipedia.org/wiki/File_sharing

[16] Wikipedia, "File Transfer Protocol", http://en.wikipedia.org/wiki/File_Transfer_Protocol

[17] Jesse James Garrett, "Ajax : A New Approach to Web Applications", February 2005, http://adaptivepath.com/publications/essays/archives/000385.php

[18] Randy Miller, "Practical UML: A Hands-On Introduction for Developers", April 2003, http://bdn.borland.com/article/0,1410,31863,00.html